ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN SISTEMA PROTOTIPO PARA EL REGISTRO DE ASISTENCIA DE ESTUDIANTES DE LA ESCUELA POLITÉCNICA NACIONAL, BASADO EN RECONOCIMIENTO FACIAL

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

LUIS LEONARDO FÉLIX MORENO

luis.felix@epn.edu.ec

DIRECTOR: FRANKLIN LEONEL SÁNCHEZ CATOTA, M.Sc.

franklin.sanchez@epn.edu.ec

Quito, marzo 2020

AVAL

Certifico que el presente trabajo fue desarrollado por Luis Leonardo Félix Moreno, bajo mi supervisión.

MSc. Franklin Leonel Sánchez Catota
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Luis Leonardo Félix Moreno, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

LUIS LEONARDO FÉLIX MORENO

DEDICATORIA

Este trabajo lo dedico a mi novia, mejor amiga y compañera, que ha sido mi soporte para superar todas las adversidades que se han presentado en mi vida y que me ha dado los momentos más felices que se puede pedir. Te amo

A mi padre por ser la persona que ha sido mi ejemplo a seguir, formándome en carácter, ética y sabiduría.

A mi madre por ser la persona que siempre me ha enseñado a ser mejor cada día.

A mis hermanas, sobrina y sobrino los quiero mucho.

AGRADECIMIENTO

Agradezco a mis padres por darme el apoyo y consejo necesario para cumplir mis metas.

Al Ing. Franklin Sanchez MSc por su guía, compresión y paciencia en todo este proyecto.

A la Escuela Politécnica Nacional y mis profesores que han sido las bases de mi formación académica.

A mis amigos por ser las personas que siempre me han acompañado en las buenas y en las malas, gracias por todo este gran viaje.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE SEGMENTOS DE CÓDIGO

RESUMEN

Este proyecto técnico presenta un prototipo para el registro de asistencia de estudiantes de la Escuela Politécnica Nacional, basándose en reconocimiento facial. Esto brindará a los docentes una herramienta alternativa a la habitual, les permitirá agilizar el proceso del registro de asistencia y centralizará la información en un servidor conectado a una base de datos.

El prototipo se conforma por: una base de datos, un servidor programado en Python y una aplicación cliente que correrá sobre dispositivos móviles Android.

En la base de datos se almacena toda la información de los estudiantes, los usuarios, y los registros de asistencias. El servidor es el encargado de procesar todas las peticiones de los clientes, además de ser el encargado de realizar las consultas a la base de datos, procesar todo el reconocimiento facial y generar los reportes de asistencia. La aplicación cliente es la encargada de enviar al servidor las peticiones que soliciten los usuarios, además es la encargada de capturar las imágenes de los estudiantes para el registro de asistencias y enviarlas al servidor.

Este documento se organiza en 5 capítulos y una sección de Anexos.

En el primer capítulo se da una introducción al prototipo a desarrollar, se muestra los objetivos, el alcance que se ha definido, y se da un marco teórico en el cual se definen y explican todas las herramientas que se utilizaran en el proyecto En el segundo capítulo se muestra la metodología del proyecto, se defienden los requerimientos del sistema, se realizan los diagramas necesarios para definir los procesos del prototipo, todo en base a Scrum. Luego se realiza la implementación.

En el tercer capítulo, se muestra las pruebas realizadas en el prototipo desarrollado, se muestran los resultados obtenidos y se corrigen errores encontrados a lo largo de la implementación. En el cuarto capítulo se da conclusiones del proyecto y recomendaciones para posteriores trabajos. En el quinto capítulo se da las referencias bibliográficas utilizadas y finalmente se muestran los Anexos utilizados.

PALABRAS CLAVE: Reconocimiento Facial, Python, MySQL, Android, OpenCV, Sockets.

ABSTRACT

This technical project introduces a prototype based on facial recognition for student attendance record at the Escuela Politécnica Nacional. This will provide professors an alternative tool that allow them to speed up the attendance registration process and centralize this information on a server connected to a database. The prototype is made up of a database, a server programmed in Python and a client application that will run on Android mobile devices.

In the database all the information of the students, the users, and the attendance records will be stored. The server will be responsible for processing all customer requests. In addition, it will be responsible for making queries to the database, processing all facial recognition and generating attendance reports. The client application will be in charge of send the requests by the users to the server, and will be in charge of capture the images of the students for the attendance record and send them to the server.

This document is organized in 5 chapters and a Supplemental Material section. The first chapter describes an introduction to the prototype that will be developed, the objectives, the defined scope, and a theoretical framework, which defines and explains all the tools that will be used in the project. Then, the second chapter includes the methodology of the project and the system requirements, also it illustrates the necessary diagrams to define the prototype processes, all based on Scrum; after that, the implementation is done. Later, the third chapter describes the performed tests on the prototype and the obtained results. Additionally, the errors found throughout the implementation are corrected in this chapter. Next, the fourth chapter shows the conclusions and recommendations for further projects. Thereafter, the fifth includes the references. Finally, the Supplemental Material is in the last chapter.

KEYWORDS: Facial Recognition, Python, MySQL, Android, OpenCV, Sockets

1. INTRODUCCIÓN

En la actualidad, el sistema de educación superior del país se rige por reglamentos que indican como debe ser el proceso de enseñanza de los estudiantes y que normas se deben cumplir tanto en sistema de calificaciones, duración de las carreras y asistencia a las diferentes actividades académicas que se desarrollen. El tema de las asistencias de los estudiantes se detalla en el artículo 17 del Reglamento Académico expedido por El Consejo de Educación Superior (año 2013), en cual se menciona que todo estudiante debe cumplir con una cantidad de horas de asistencia a diferentes actividades académicas [1].

En cumplimiento del artículo 17 anteriormente mencionado, los diferentes centros de educación superior han creado sus propios reglamentos académicos en los cuales detallan cómo será la asistencia de los estudiantes, por ejemplo, en el Reglamento de Régimen Académico de la Escuela Politécnica Nacional se menciona en el capítulo X artículo 74, que todo estudiante deberá asistir de manera obligatoria y con puntualidad a las diferentes actividades académicas regulares en las fechas establecidas [2]. Por lo tanto, los docentes deberán llevar un registro de asistencias de los estudiantes, para poder corroborar que cada estudiante haya cumplido con lo estipulado en el reglamento.

Por lo mencionado anteriormente, se plantea el desarrollo de un sistema prototipo para el registro de asistencia de estudiantes, utilizando un dispositivo móvil, el cual permitirá al docente, llevar el registro de asistencia de cada materia de forma simplificada, haciendo uso de la cámara del dispositivo y aplicando reconocimiento facial para identificar que estudiantes están presentes en la actividad académica y cuales no; esto servirá como una herramienta de apoyo para los docentes.

1.1 OBJETIVOS

El objetivo general de este Proyecto Técnico es:

- Desarrollar un sistema prototipo para el registro de asistencia de estudiantes de la Escuela Politécnica Nacional, basado en reconocimiento facial.

Los objetivos específicos del Proyecto Técnico son:

- Analizar las herramientas tecnológicas necesarias para el desarrollo del prototipo propuesto.
- Diseñar los componentes del prototipo necesarios para el desarrollo del mismo.

- Implementar los componentes del prototipo con base al diseño realizado
- Analizar los resultados obtenidos de las pruebas ejecutadas en el prototipo.

1.2 ALCANCE

El trabajo de titulación presentado propone el desarrollo de un prototipo para el registro de asistencia de estudiantes, el cual funcionará en un sistema cliente – servidor, por lo que permitirá a varios docentes registrar la asistencia de los estudiantes de cada actividad académica a través de una aplicación móvil. Por lo tanto, se deberá implementar un servidor y un cliente, el cual será la aplicación móvil.

El servidor tendrá como función principal recibir, procesar las peticiones del cliente y realizar las consultas a la base de datos. Específicamente deberá recibir la o las imágenes de los asistentes a las actividades académicas, procesarlas y aplicar el reconocimiento facial de los estudiantes identificados; posteriormente deberá guardar la lista de los estudiantes presentes en la base de datos. Este servidor será desarrollado en lenguaje de programación Python, se utilizará como motor de base de datos relacional MySQL, la librería de mapeo objeto-relación será el ORM PEEWE y como librería para el procesamiento de imágenes, detección y reconocimiento facial se empleará la librería OpenCV para Python.

La aplicación cliente será la responsable de capturar la o las imágenes de los estudiantes presentes en el aula y las enviará al servidor para que estas sean procesadas, también dispondrá de los perfiles de administración y el perfil de cada docente. El cliente se implementará con uso de tecnologías nativas de Android. En la conexión entre cliente y servidor se utilizará sockets proporcionados tanto por Python y por Android; y para el desarrollo de este prototipo se basará en la metodología de desarrollo ágil Scrum [3].

En el sistema prototipo propuesto, la aplicación cliente constará de dos perfiles, uno para el administrador y otro para el docente; el perfil del administrador permitirá realizar el CRUD (*Create, Rename, Update and Delete*) para los docentes, estudiantes, materias, paralelos y actividades académicas; además el administrador será el encargado de enlazar a los docentes y estudiantes con una materia, para poder llevar a cabo la toma de asistencia. Por otra parte, el perfil de docente podrá realizar el CRUD de los estudiantes de cada materia, además podrá ingresar en la base de datos las fechas de cada actividad académica, para posteriormente mediante el uso de la cámara del dispositivo Android,

tomar las fotografías necesarias de los estudiantes presentes y poder enviarlas al servidor para que sean procesadas. Una vez que el servidor realizó el procesamiento de la o las imágenes de una actividad académica, retornará al cliente la lista de los asistentes. Al finalizar el periodo académico o cuando se considere necesario, desde el cliente se podrán generar un reporte en Excel con el historial de asistencias para una materia.

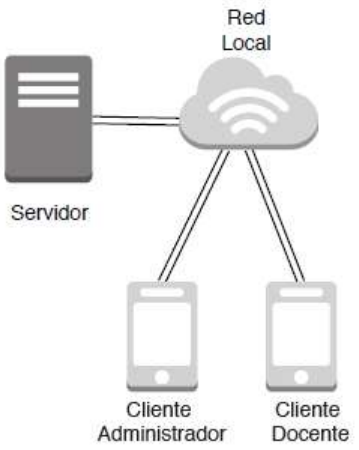

Figura 1.1 Elementos del sistema prototipo.

En la Figura 1.1 se muestra un ejemplo de los componentes de hardware que conforman el prototipo. Los perfiles del cliente compilarán sobre dispositivos Android y se conectarán al servidor por una red local (LAN) para realizar las funciones especificadas anteriormente.

1.3 MARCO TEÓRICO

En esta sección se realizará un análisis del fundamento teórico que servirá como referencia para el desarrollo de este proyecto. En la sección 1.3.1 se presenta un estudio de los fundamentos de sistemas Cliente - Servidor con el cual se desarrolló el prototipo y se analizará porque se escogerá este tipo de sistema. Luego, en la sección 1.3.2 se presenta un estudio de los fundamentos de las bases de datos específicamente de la base MySQL, se presentará ventajas y desventajas con respecto a otras bases de datos. Seguidamente, en la sección 1.3.3 se expondrá las características del lenguaje de programación Python y su implementación para servidores utilizando sockets. Más tarde, en la sección 1.3.4 se analizará las herramientas de mapeo de Objeto – Relación, específicamente de la librería PEEWE para Python. Después en la sección 1.3.5 se estudiará los fundamentos de la detección y reconocimiento facial, enfocándose en la librería de OpenCV para Python. Como penúltimo punto, en la sección 1.3.6 se estudiará la arquitectura de diseño de aplicaciones Android y el uso de sockets en la misma; además se responderá la pregunta: ¿por qué se escogió sockets para su uso en este proyecto?. Finalmente, en la sección

1.3.7 se realizará una introducción a las metodologías ágiles de desarrollo enfocándose en las características principales de la metodología Scrum.

1.3.1 SISTEMAS CLIENTE – SERVIDOR

Los sistemas cliente - servidor (también llamados modelos cliente - servidor) son parte de los *sistemas distribuidos*[1], en los cuales pueden existir dos o más componentes de hardware con su propio software que interactúan entre sí, uno realizando peticiones al otro (cliente) y el otro respondiendo a dichas peticiones (servidor), muchas veces un servidor puede ser cliente de otros servidores. En la Figura 1.2 se muestra un ejemplo de un cliente realizando peticiones a un servidor, y este mismo realizando peticiones a un servidor de base de datos.

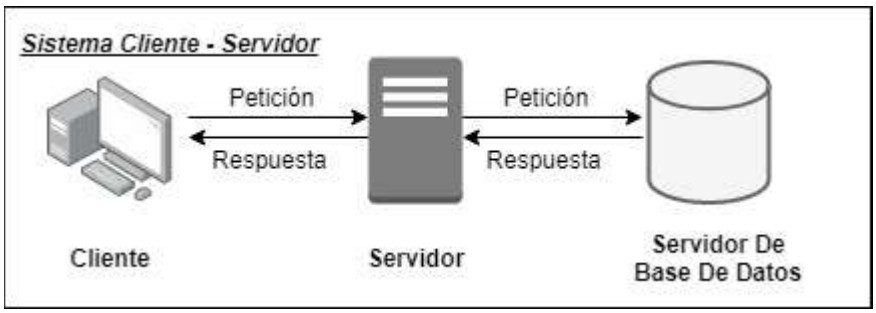

Figura 1.2 Modelo/Sistema Cliente Servidor

Al tener un modelo distribuido se puede hablar de sistemas cliente – servidor multiplataforma, es decir se podrán manejar varios sistemas operativos, diferentes componentes de hardware, varios lenguajes de programación y varias bases de datos. Para esto se deberán tener protocolos previamente establecidos para que se logre mantener la comunicación. [4]

1.3.1.1 Clasificación de los Sistemas Cliente- Servidor según sus capas.

Los sistemas cliente-servidor se clasifican según la división de tareas que cada componente realiza en el sistema, a estos grupos según divisiones de tareas se las llama capas o niveles.

---

1 Sistema Distribuido: conjunto de hardware separado físicamente y conectado a través de una red de comunicaciones, los cuales trabajan conjuntamente como un solo sistema, para realizar tareas programadas.

1.3.1.1.1 Sistema Cliente-Servidor de 2 capas

En este modelo el cliente realiza la petición a un servidor el cual procesa la solicitud y responde utilizando sus propios recursos. A continuación, en la Figura 1.3 se ejemplifica un sistema cliente servidor de dos capas, en el cual el cliente realiza una petición al servidor y este le responde sin necesidad de realizar peticiones a otros servidores. [5]

Figura 1.3 Sistema C/S de 2 capas

1.3.1.1.2 Sistema Cliente-Servidor de 3 capas

En este modelo el cliente (capa 1) realiza la petición a un servidor de aplicaciones (capa 2) el cual funciona como *middleware*[2], ya que necesitará conectarse a otro servidor (capa 3) para completar la petición realizada. En la Figura 1.2, presentada en la sección 1.3.1 ya se ejemplificó como actúa un modelo de tres capas en el cual el cliente realiza una petición al servidor y este realiza a su vez una petición a una base de datos para obtener información y poder responder a la solicitud. [5]

Este modelo es muy utilizado ya que por lo general la mayoría de los sistemas necesitan una base de datos para obtener información de las solicitudes, estas bases de datos pueden estar en el mismo servidor, pero se acceden mediante una dirección de red y un puerto diferente, por ende, se la puede considera una capa más. Si se realiza la conexión a más servidores los cuales estén especializados en una tarea específica, se puede hablar de modelos de N-capas.

[2] **Middleware**: capa de software que permite la compatibilidad entre cliente y servidor para poder establecer la comunicación.

1.3.1.2 Ventajas de los Sistemas Cliente-Servidor

- Permite escalabilidad del sistema ya que se podrán integrar más servicios sin necesidad de alterar los demás ya existentes.
- Permite concentración de información. Con esto nos aseguramos de que todos los clientes posean los mismos datos en tiempo real.
- En temas de seguridad un cliente no deberá conocer todas las IP's y puertos de todos los servidores por lo que el riesgo de hackeo se disminuye.
- Permite el fácil mantenimiento del sistema, ya que se lo puede realizar de forma modular, solo bloqueando ciertas funciones, esto permite que no se tenga que parar el sistema para realizar algún mantenimiento. [46]

1.3.1.3 Desventajas de los Sistemas Cliente-Servidor

- El diseño de la red que transporta la información entre cliente y servidor puede ser un factor de riesgo para que nuestro sistema tenga latencias altas. Por lo tanto, se debe tener una red robusta y una en la cual se considere el número de clientes conectados simultáneamente y el ancho de banda que ocuparán.
- Si el servidor principal el cual realiza la mayoría de procesamiento y es el punto de conexión a otros servidores falla, el sistema puede fallar.
- Se es propenso a ataques de denegación de servicio (DoS), ya que se puede simular las peticiones de los clientes y saturar el sistema. [46]

En toda esta sección se ha puesto en evidencia por qué se escogió un sistema Cliente Servidor, esto se debe a que se tendrá un cliente en un dispositivo móvil Android, el cual se conectará a un servidor que procesará toda la información enviada, y finalmente los resultados se almacenarán en una base de datos. Si no se hubiera escogido este modelo de sistema, cada usuario debería tener la información de la base en su propio dispositivo móvil, además de que el dispositivo debería soportar todo el procesamiento de la detección y el reconocimiento facial; también con un sistema Cliente-Servidor se podrá tener la posibilidad de ingresar con un usuario y contraseña desde diferentes dispositivos sin tener la necesidad de migrar la información.

1.3.2 BASES DE DATOS

Las bases de datos son un conjunto de datos organizados que se encuentran bajo un mismo contexto y que son almacenados para su posterior utilización, estos datos se almacenan en tablas con cabeceras que identifican cada dato, estas cabeceras se

denominan campos; además los datos de una tabla pueden estar relacionados con los de otra tabla por un campo en común, por lo que se les llama base de datos relacional. Una base de datos, en conjunto con un software que le permita administrar, hacer consultas, generar nuevas relaciones y tablas con sus datos se denomina *Sistema de Administración de Base de Datos* (DBMS). [6]

Para organizar y administrar la base de datos se utiliza un Modelo Relacional, este se basa en almacenar los datos en Tablas compuestas por tuplas (filas) y campos (columnas). Las filas serán las instancias de la entidad (objeto, característica) que representa cada tabla, estas instancias tendrán atributos dados por los campos o columnas y describirán las características principales de cada instancia. En la Tabla 1.1 se puede apreciar cómo se forma una tabla de datos que representa una entidad.

Tabla 1.1 Tabla de Entidad Persona

Persona (Entidad)				
Id	**Nombre**	**Apellido**	**Cédula**	**Género**
1	Mishell	Barreno	1711435698	Femenino
2	Luis	Félix	1718162058	Masculino

La relación entre campos en una tabla se denomina esquema, por ejemplo, tomando los datos de la Tabla 1.1 el esquema sería: Persona (Id, Nombre, Apellido, Cédula, Género). Al conjunto de los esquemas de diferentes tablas se los conoce como *Esquema Relacional de Base de Datos.* [7]

1.3.2.1 MySQL

MySQL es un sistema de administración de bases de datos (DBMS), multihilo (puede realizar más de una tarea a la vez) y multiusuario (varios usuarios pueden conectarse simultáneamente a la base de datos para poder realizar consultas). Creado por la empresa MySQL AB, pero actualmente administrada por la empresa Oracle bajo una licencia de software libre bajo modalidad dual o doble licenciamiento.

Está implementado en su mayoría en lenguaje ANSI C y utiliza lenguaje SQL (Structured Query Language) para sus consultas de las bases de datos. Al utilizar el motor no transaccional MyISAM como motor por defecto, sus tiempos de consultas son muy bajos, pero pueden provocar errores en la integridad de los datos en ambientes de alta demanda de modificación.

MySQL permite un sistema de privilegios de usuarios muy completo y flexible, con esto se podrá dar ciertos permisos a los usuarios que accedan a la base, para evitar la mala manipulación de los datos y evitar sentencias como DROP DATABASE que pueden resultar con la pérdida de toda la información. Este DBMS permite hasta 50 millones de registros por base de datos y es multiplataforma, lo que la lleva a ser una herramienta muy poderosa al momento de escoger con qué sistema de bases de datos trabajar. [8]

El DBMS MySQL posee una amplia gama de librerías y drivers compatibles con varios lenguajes de programación, esto permitirá conectar las aplicaciones con los servidores de base de datos, ya sea que se encuentren en la misma máquina o a través de una red. Los lenguajes de programación soportados por MySQL son: Java, Python, JavaScript, C++, C#, C y PHP. También existen ODBCs (Open Data Base Conectivity) los cuales son drivers que cumplen un estándar desarrollado por SQL Acces Group (SAG), que permiten enlazar al servidor de base de datos con algunas aplicaciones como son Excel, software de Big Data o cualquier aplicación que soporte este estándar. [9]

1.3.2.1.1 Ventajas y desventajas de MySQL.

Las ventajas de MySQL son:

- MySQL soporta Tablas hasta de 1 Tera Byte de almacenamiento.
- Es multiplataforma, es decir es soportado por varios sistemas operativos.
- Gracias a MySQL Workbench se posee una interfaz gráfica muy completa para la administración del servidor de base de datos.
- Permite consultas SQL.
- Se puede almacenar sintaxis y contenidos de objetos JSON.
- Posee conectores y ODBCs para conectarse con cualquier aplicación compatible.
- Soporta hasta 32 índices por tabla.
- Gestión de usuarios y niveles de acceso
- Es de software libre, aunque si una empresa requiere un sistema propietario las licencias están disponibles.

- Es multiusuario, es decir permite varios usuarios conectados a la vez. Posee ORM's que ayudan a la conexión de las aplicaciones con el servidor de base datos, ya que añaden librerías para facilitar al acceso y manipulación de datos a través de funciones, objetos y clases. [9]

Las Desventajas de MySQL son:

- No posee Triggers[3], esto debido a que reducen el tiempo de respuesta a las peticiones.
- Los procedimientos almacenados solo se encuentran disponibles a partir de la versión 5 de MySQL
- Se puede producir casos en que usuarios deseen acceder al mismo tiempo a una instancia específica, lo cual conllevará a que alguno no pueda acceder a la información hasta que el otro usuario termine su consulta. Esto también podría llevar a un caso de error de información de la base si es que los usuarios no actualizan sus consultas antes de modificar alguna información. [9]

1.3.2.1.2 Comparativa entre DBMS.

En la Tabla 1.2 se muestra una comparativa entre los principales DBMS, con esto se busca justificar por qué se escogió MySQL para este proyecto de titulación [10] [11] [12]

Tabla 1.2 Comparativa de DBMS

Característica	**DBMS**		
	MySQL	**SQL Server**	**Oracle DB**
Empresa Desarrolladora	Oracle	Microsoft	Oracle
Lenguajes de Programación Aceptados	Java, Python, JavaScript, C++, C#, C, PHP.	C#, .net, Java, Node.JS, C++, PHP, Python, Ruby	Java, .NET, Python, SQL Developer, XE, APEX, Node.JS, c, C++, Ruby, PL/SQL, ROracle, NetServices, Spatial Studio
Tipo de Licencia	Libre y gratuita en su versión Community	Pagada en todas sus versiones	Pagada en todas sus versiones
Requerimiento Mínimos	512 MB de RAM, 1024 MB RAM en VM, arquitectura de 32 o 64 bits.1GB de almacenamiento.	512 MB de RAM, arquitectura de 64bits,6 GB de almacenamiento	1 GB de RAM, arquitectura de 64 o 32 bits, 500 MB de almacenamiento para instalación (Recomendado 10 GB)
Interfaz de Usuario	Si	Si	Si

[3] **Trigger**: también llamado disparador, es un proceso que se ejecuta cuando se cumple alguna condición dentro de la base de datos.

Privilegios A Nivel de Usuario	Si	Si	Si
SO soportados	Windows, Mac y Linux	Windows, Linux y Contendores Docker	Windows, Mac y Linux

Como se pudo analizar en la Tabla 1.2 la mejor opción para este proyecto es MySQL debido a su compatibilidad con Python, su latencia muy baja para responder a las consultas que se le realice, posee una buena interfaz de usuario para poder administrar la base de datos, su conector para Python está integrado en el ORM PEEWE lo cual se explicará más adelante, y como punto predominante es de licencia libre y no se necesita pagar ningún rubro por usar MySQL.

1.3.3 LENGUAJE DE PROGRAMACIÓN PYTHON

Python es un lenguaje de programación interpretado de alto nivel, que utiliza como paradigma la programación orientada a objetos; fue desarrollado por Guido Van Rossum y publicado por primera vez en el año de 1991, actualmente este lenguaje se encuentra en su tercera versión llamada Python 3. Tiene una licencia de software libre llamada Python Software Foundation License (PSFL) que permite modificaciones al código fuente, sin necesidad de realizar algún pago por ello.

Python es un lenguaje multiplataforma por lo que puede ser interpretado en varios sistemas operativos como son Windows, Linux y MAC; aunque también existen versiones para dispositivos móviles (smarth phones, tablets, smarth watchs) y otros tipos de dispositivos.

Python también es soportado por muchas bases de datos tal y como se mencionó en la sección 1.3.2, esto le permite ser uno de los lenguajes favoritos al momento de programar; otros aspectos muy apreciados son su sencilla sintaxis y su manera trabajar matricialmente con los datos organizándolos en tuplas, listas, vectores y matrices. [14]

Las aplicaciones más comunes en las que se aplica Python son [15]:

- **Desarrollo WEB**: Python ofrece para el "internet development" algunos frameworks[4] como son Django o Pyramid, algunos micro-frameworks como Flask y Bottle. También soporta algunas librerías web como son HTML, XML, JSON, FTP,

4 **Framework**: esquema o estructura que se utiliza para desarrollar y organizar programas (software), posee la automatización de algunos procesos comunes.

SMTP, IMAP, entre otros protocolos de internet. Posee una interfaz muy sencilla para el uso de sockets.

- **Programación aplicada a la ciencia y operaciones numéricas**: Python permite la manipulación de datos en forma matricial lo cual la convierte en un lenguaje aplicable al procesamiento de imágenes, procesamiento de video, implantación de redes neuronales, simulaciones de ambientes reales. Además, permite la incorporación de librerías como SciPy (librería que permite realizar operaciones matemáticas complejas para ciencia e ingeniería), Pandas (librería para el modelado y análisis de datos) o IPython (librería para computación paralela y simulaciones).
- **Programación de interfaces gráficas de usuario (GUI):** Python incluye en la mayoría de sus versiones la "TK GUI library", que es una librería para el desarrollo de interfaces de usuario gráficas.
- **Programación para aplicaciones de negocios:** Python dispone de librerías para desarrollar aplicaciones para el comercio electrónico como son Odoo y Tryton.
- **Educación:** Python al ser un lenguaje con una sintaxis muy sencilla es de fácil aprendizaje por lo que es muy adecuada para las personas que se inician en los lenguajes orientados a objetos. Además, posee muchas guías y material referencial que se lo puede encontrar en su página web oficial "www.python.org".

1.3.3.1 Sockets para Python

Los sockets son un bloque de construcción básico para la comunicación ya que son una herramienta que opera en la capa transporte del modelo TCP/IP. Son una abstracción que agrupan propiedades comunes del proceso de comunicación ya sea punto a punto o comunicación multipunto. Un socket puede comunicarse con otros siempre que se encuentre bajo el mismo dominio y comparta los mismos protocolos de conectividad, si se desea comunicar con otros dominios se deberá realizar alguna transformación o traducción.[16]

Los sistemas cliente-servidor son los que más utilizan sockets, los cuales poseen APIs que han simplificado su uso, aunque aún siguen siendo una conexión a nivel de capa transporte (esto significa que se necesita para la conexión una IP y un puerto). Estos sistemas utilizan sockets, ya que muchas veces se busca una conectividad entre distintas plataformas o diferentes programas con varios lenguajes de programación, en los cuales, si se desea utilizar algún framework disponible para la conexión entre cliente y servidor con las

condiciones ya establecidas, se pueden generar problemas al tener objetos con mapas de serialización diferentes.

La API socket para Python está disponible tanto como para su versión 2 y 3, para este proyecto se utilizará Python v.3.6.5 por lo que se explicará la API para dicha versión. Las principales funciones (métodos) que posee son: `socket()`, `bind()`, `listen()`, `accept()`, `connect()`, `send()`, `recv()` y `close()`; estas funciones pueden ser utilizadas para levantar un servidor y un cliente. El servidor será el encargado de escuchar, aceptar y responder las peticiones del cliente, el mismo que será el encargado de realizar la petición de conexión y enviarla al servidor, para posteriormente recibir una respuesta.

Los sockets deben trabajar bajo un protocolo de capa de transporte ya sea TCP (Protocolo de Control de Transmisión) o UDP (Protocolo de datagramas de Usuario). El protocolo escogido para este proyecto de titulación es el TCP ya que se tiene una red IP que trabaja con el mejor esfuerzo, es decir puede haber perdida de paquetes y si el protocolo de capa transporte no es confiable se puede tener perdida de información, la cual no habrá forma de saber que se perdió por parte del servidor o del cliente. [17]

A continuación, en la Figura 1.4 se muestra como es el proceso de comunicación de sockets entre cliente y servidor:

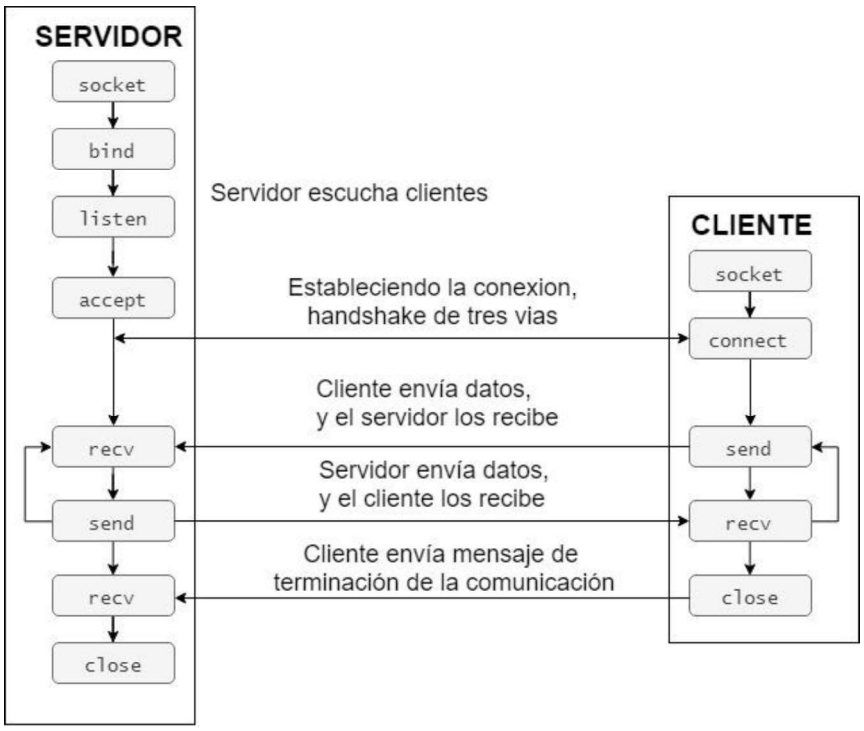

Figura 1.4 Secuencia de sentencias de la API socket

Para establecer la conexión primero el servidor debe crear el objeto socket mediante la función `socket()` en la cual se definirá que protocolos se utilizarán, a más de indicar que ese lado de la conexión es el servidor. Posteriormente al objeto socket se le deberá asignar una dirección IP y puerto para poder escuchar a los clientes, eso se lo hace en la función `bind()`, luego con la función `listen()` ponemos al servidor a escuchar a los clientes, aquí se deberá especificar el número de conexiones simultaneas permitidas, después con la función `accept()` pondremos al servidor en un estado de bloqueo (estado de escucha) hasta que reciba alguna conexión de un cliente. Si se tiene la opción de aceptar conexiones de más de un cliente simultáneamente, el servidor utilizará hilos para procesar las operaciones requeridas.

El cliente para establecer la conexión deberá crear un objeto socket mediante la función `socket()`, en la cual deberá colocar los protocolos y dominios de conexión que permita el servidor, luego mediante la función `connect()` se deberá indicar la IP y el puerto del servidor a la que quiere conectarse. En este momento el cliente enviará la solicitud de conexión al servidor, el mismo que lo aceptará siempre que cumpla con los parámetros requeridos y le responderá con una confirmación de conexión (acuse de recibo), finalmente el cliente volverá a responder un acuse de recibo al servidor indicándole que sí completó la conexión, este proceso se llama handshake (comprobación) de tres vías.

Una vez establecida la conexión se podrá enviar y recibir información por parte del servidor o del cliente según lo necesite la aplicación. Para finalizar la conexión cualquiera de las partes puede enviar un mensaje de desconexión mediante la función close(), con lo que se dará por terminada la conexión.

1.3.4 HERRAMIENTAS DE MAPEO OBJETO – RELACIÓN (ORM)

Los Object Relation Mapping o herramientas de Mapeo Objeto Relación (ORM), son programas disponibles para varios lenguajes de programación que se basan en el paradigma orientado a objetos. Estos programas permiten traducir las entidades expresadas en tablas de una base de datos, en objetos que se pueden invocar muy fácilmente dentro del código . [18]

Si se analiza cómo integrar datos de una base de datos a un programa, generalmente se debería agregar las sentencias necesarias dentro del código de programación, por ejemplo, para bases de datos SQL se puede enviar las consultas como texto plano, pero hay que considerar que muchas veces necesitamos la información de varias Tablas, lo que haría a las consultas mucho más complejas. Como solución a esto los ORMs permiten traducir

tablas con sus respectivas llaves y relaciones, a objetos y clases con los que ahora se podrá acceder a su información como normalmente se lo haría en un lenguaje POO; aunque esto conlleva a que se debe dedicar tiempo al aprendizaje de las librerías que permiten el mapeo. También se debe tomar en cuenta los tipos de datos que se utilizan ya que deben ser los mismos en la base como en las clases. Finalmente, para evitar problemas de serialización al momento de enviar los datos, se debe trabajar en ambas partes con el mismo formato de codificación, por ejemplo UTF-8. [18]

Para Python existen algunos ORMs por ejemplo Django posee su propio ORM para bases SQL, también existe SQLAlchemy, Pony ORM y Peewee [18]. Este último ha sido escogido para ser implementado en este proyecto de titulación, debido a que es un ORM muy fácil de aprender (posee una buena documentación), es compatible con Python 3, las sentencias para consultas son a través de objetos con métodos muy intuitivos y basados en sentencias SQL.

1.3.4.1 ORM Peewee

Peewee es un ORM escrito en Python, compatible con Python 2.6+ y 3.2+, además es compatible con las bases de datos SQLLite, MySQL, Postgresql y extensiones como HSTORE. [19]

Para empezar a utilizar Peewee se debe realizar la conexión a la base de datos desde el programa a través de sus funciones respectivas según la base de datos que se tenga, en nuestro caso al utilizar la base de datos MySQL se utilizarán las sentencias de la librería `peewee` para dicha base. Con estas sentencias se podrá especificar a qué servidor conectarse mediante una IP y un puerto, además se deberá indicar que usuario con su respectiva contraseña se conectará. [20]

Luego de obtener la conexión se procede a mapear las tablas a clases, para esto se deberá modelar una clase padre la cual representará toda la base de datos y varias clases que heredarán de la clase principal, estas clases hijas se las creará para mapear las tablas y entidades de la base de datos, al instanciarlas obtendremos un objeto que representará cada fila de las tablas según corresponda. Se debe cumplir con la definición de clase según POO, es decir, se debe crear constructores en los cuales se deberá indicar que tipo de atributo se tendrá, así como que propiedades cumplirá en la base de datos, por ejemplo, si puede o no tomar el valor de null, si es un valor único, si es una llave primaria, entre otras opciones. [21]

Al momento de realizar las respectivas consultas se deberá instanciar las clases, y ejecutar las sentencias deseadas como son `SELECT`, `DELETE`, `UPDATE`, `JOIN`; todas estas retornarán los objetos con la información solicitada para ser utilizados en la aplicación. Finalmente es aconsejable cerrar la conexión de la base de datos para no tener conflicto con los puertos. [19]

1.3.5 LIBRERÍA OPENCV

Open Source Computer Vision Library u OpenCV es una librería de código abierto, con Licencia BSD[5] que incluye varios algoritmos para Computer Vision (Visión por Computadora), disponible para varias plataformas como son iOS de Apple, Android, Windows y Linux; también disponible para varios lenguajes de programación como son Python, C++, Matlab. entre otros.

Inicialmente OpenCV fue desarrollada como librería, pero el día de hoy ya se le considera una API ya que posee una gran cantidad de algoritmos clasificados en diferentes librerías, que trabajan entre sí sin necesidad de invocarlos. OpenCv tiene una estructura modular, cuyo detalle se muestra en la Tabla 1.3. [22]

Tabla 1.3 Módulos de OpenCV

Módulo	Descripción
Core	Es un módulo que define las estructuras básicas de datos para todos los demás módulos. Incluye todas las funciones básicas para el manejo de matrices multidimensionales.
Imgproc	Es el módulo que permite realizar el procesamiento de imágenes como es filtrado lineal y no lineal, transformaciones de tamaño, operaciones matemáticas con imágenes, compresión de imágenes, conversión de espacio color, histogramas, perspectivas de imágenes, regeneración básica.
Video	Es el módulo que permite manipular secuencias de video y su respectivo análisis. Contiene la función de estimación de movimiento, sustracción o modificación de fondo y algoritmos para la secuenciación y seguimiento de objetos.
Calib3d	Este módulo posee algoritmos de geométrica de múltiples vistas, calibración de tiempos en cámaras y sonidos estéreo, estimación de ubicación de objetos,

[5] **Licencia BSD (Berkeley Software Distribution):** licencia de distribución de software que permite tener muchas libertades al momento de usarla, modificarla y distribuirla; se considera muy cercana al dominio público. Una de sus principales características es que permite la integración en software propietario o no libre.

	algoritmos para manipulación de sonidos estéreo y manipulación de objetos 3D (3 Dimensiones).
Feautures2d	Módulo que permite la utilización de descriptores y detectores de flujos de salida.
Objdetect	Módulo que permite la detección y reconocimiento de objetos con instancias ya definidas en la librería. Los objetos que se pueden detectar son: rostros de personas, ojos, objetos de uso común, animales. Actualmente se permite adicionar nuevos objetos para detección según lo necesite el usuario.
Highgui	Módulo que permite implementar Interfaces de usuario GUI
Videoio	Módulo utilizado para la captura y reproducción de videos utilizando diferentes códecs.
GPU	Módulo que permite acelerar el uso de algoritmos utilizando GPU o unidades gráficas de procesamiento, para esto se necesitará una tarjeta gráfica.
Otros	También existen otros módulos auxiliares para la conexión con servicios y compatibilidad con APIs de otras empresas.

Para este proyecto se utilizarán los módulos Core, Imgproc y en especial Objdetect, ya que permitirán realizar la detección y reconocimiento los rostros de los estudiantes.

Los lenguajes que más utilizan OpenCV y por lo tanto tienen mayor cantidad de documentación son Python y C++. Para este proyecto se escogió utilizar OpenCV con Python ya que facilitan el uso de matrices, procesamiento de imágenes y cálculos matemáticos; además se ha tratado de utilizar solo herramientas de distribución libre por lo que estas opciones son las mejores.

1.3.5.1 Detección facial con OpenCV

Para cumplir con el objetivo de este proyecto es importante definir cuál es el procedimiento para identificar a cada estudiante. El primer paso es la detección facial de los estudiantes presentes en el aula de clase. La detección facial es un proceso de visión por computadora que aplicando un entrenamiento de imágenes puede detectar confiablemente rostros humanos.

Actualmente grandes empresa como Google, Amazon y Facebook han desarrollado sus propias APIs y frameworks para poder realizar detección de objetos y rostros, pero el problema es que muchos de estos algoritmos son licenciados, no son de distribución gratuita, necesitan de un buen conocimiento de redes neuronales para su implementación y requieren hardware con gran capacidad de procesamiento. Por esta razón la librería OpenCV incorpora métodos para la detección de rostros y objetos mediante clasificadores en cascada con el análisis de imágenes o videos; estos métodos son fáciles de utilizar, son

de libre distribución y puede ser ejecutados sin necesidad de tener hardware muy avanzado (se puede ejecutar en dispositivos móviles y hasta en la Raspberry Pi).[23]

Los clasificadores para detección facial son algoritmos que detectan las zonas de la imagen que si corresponden al rostro y cuales no; esto permite limitar los pixeles de la imagen que son importantes. Los clasificadores han sido entrenados con cientos de miles de imágenes de rostros (imágenes positivas) y con imágenes sin rostros (imágenes negativas) para así poder realizar la detención facial. [23]

Los clasificadores en cascada permiten la recolección de información de las imágenes ordenadamente, ya que son una colección de etapas secuenciales en las que se va comparando las imágenes y filtrando los datos erróneos. Inicialmente las imágenes analizadas tendrán resultados muy débiles para predecir otras a partir de sus resultados, por esto se realizan varias etapas de análisis a las imágenes positivas y negativas para encontrar rostros y objetos, con sus respectivas características. En una imagen se puede tener zonas con objetos y rostros, pero también hay zonas que no poseen nada y en general es la mayor parte de la imagen, por lo que no es necesario seguir procesando dicha información, esta diferenciación de cada parte de la imagen permite que en las diferentes etapas se pueda filtrar la información útil para poder llegar a tener un sistema de detección confiable.

Muchas veces se puede dar el caso en el que en una imagen se tengan falsos positivos, es decir detección de zonas sin objetos ni rostros, esto puede agravar el resultado final, pero las siguientes etapas pueden corregir este problema, he ahí la gran importancia de usar clasificadores en cascada. También puede pasar que el algoritmo a utilizar no detecte los rostros u objetos, por lo que una imagen puede dar varios rechazos de información, como resultado a esto se la descarta; este problema no puede ser corregido en las diferentes etapas, pero una solución es tener en cuenta que imágenes se utilizan para el entrenamiento y que algoritmo se utilizará.[24]

En OpenCV existen dos algoritmos disponibles ya entrenados para detención facial o de objetos, estos son Haar Cascade y LBP (Patrón Binario Local). Cada uno tiene su manera de realizar la detección, pero más popular es Haar Cascade por lo que tiene más documentación, un mejor entrenamiento y es más rápido en obtener resultados confiables. Por esta razón se lo escogió para este proyecto de titulación.

1.3.5.1.1 Haar Cascade

Es un algoritmo para la detección facial o de otros objetos, el cual cumple con los pasos definidos por un clasificador en cascada. Haar Cascade que fue propuesto por Paul Viola y Michael Jones en su artículo "Rapid Object Detection using a Boosted Cascade of Simple Features" [25]. En primera instancia se analiza los píxeles de la imagen en matrices cuadradas comparándolas con las funciones Haar[6], esta comparación se la agilita mediante el uso de imágenes integrales, posteriormente se utiliza el algoritmo de aprendizaje de Adaboost [7]para seleccionar las características más importantes y así poder tener un clasificador que permita detectar en este caso rostros confiablemente. [24]

Como se mencionó en el párrafo anterior primero se debe analizar los pixeles de la imagen en matrices cuadradas para identificar las características, para esto se emplean funciones de Haar, en la Figura 1.7 se muestran las representaciones gráficas de dichas funciones:

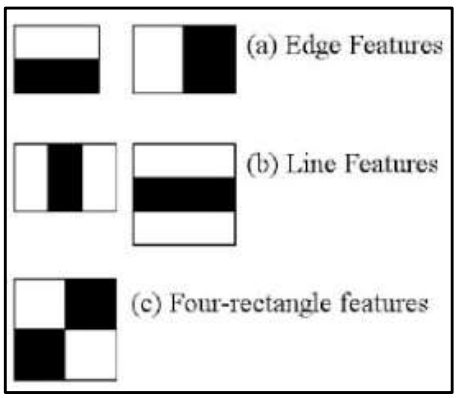

Figura 1.5 Funciones Haar [40]

Las funciones presentadas en la Figura 1.5 sirven para poder identificar las diferentes características de la imagen por ejemplo las dos primeras (a) se utilizan para identificar bordes, los dos siguientes (b) sirven para identificar líneas y la última (c) se utiliza para detectar líneas inclinadas. Matemáticamente se las podría representar por valores expuestos en la Figura 1.6:[26]

[6] Funciones Haar: modelos de pixeles que permiten identificar detalles de una imagen.

[7] Abadoost: algoritmo de aprendizaje automático, para obtener clasificadores robustos.

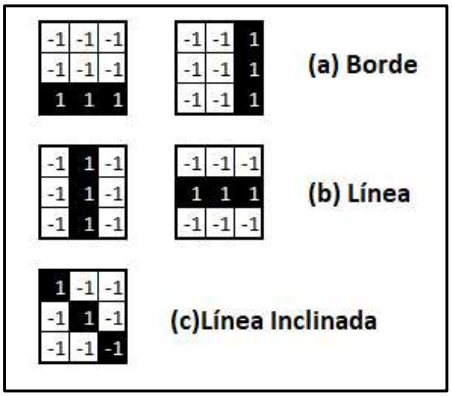

Figura 1.6 Valores Matemáticos de Funciones Haar.

Estas funciones recorrerán toda la imagen para obtener un valor único, resultado de aplicar el concepto de imágenes integrales, con el cuál se analizarán las regiones rectangulares cercanas a la zona de detección (zona clara y oscura de la función Haar), se obtendrá un resumen de la intensidad de pixeles de cada zona y se calculará la diferencia entre estas. La parte de color negro de la función Haar representa píxeles oscuros que podrían ser una línea y la parte blanca representa una zona clara, con esto se podría identificar cuando se tiene líneas de borde o una línea por encima de una zona clara, esto por ejemplo se podría traducir en la zona de los ojos en el rostro, pero hay que considerar que se deben hacer varios análisis hasta llegar a esa conclusión, por eso anteriormente se mencionó que en cada análisis del clasificador en cascada, el entrenamiento se hace más fuerte y confiable. Para el caso de los ojos se podría encontrar otra característica en la que se deduzca que entre los ojos existe una zona clara que corresponde al canal que empieza la nariz, si utilizamos esas dos características más otras encontradas en los demás análisis, se podría determinar finalmente que esa zona en realidad es la que se pensó en un principio.[24][27]

En la Figura 1.7 se puede apreciar cómo se identifican cada zona según van avanzando los análisis, al final se recolectarán todas las características obtenidas y se podrá decir que es la zona de los ojos.

Figura 1.7 Análisis en cascada de características

Otro aspecto para considerar es que en cada análisis se encontrarán muchas características, por eso mediante el algoritmo de Adaboost se podrá escoger las más importantes para el entrenamiento, esto se lo realiza eliminando características no relevantes para que en análisis posteriores no sean tomadas en cuenta.

Finalmente, una vez terminado el análisis y asociando las características más relevantes de la imagen se podrá comparar con las imágenes de entrenamiento para poder detectar la zona total del rostro. Para este proyecto se utiliza este algoritmo para realizar el entrenamiento del sistema con estudiantes y para detectar los rostros de estos en un día de clases. Posteriormente se realizará el reconocimiento facial comparando el entrenamiento ya realizado y los rostros obtenidos de las actividades académicas.

1.3.5.2 Reconocimiento Facial con OpenCV

Una vez que ya se ha obtenido la detección facial de los estudiantes presentes en clases, se deberá identificar cual es la identidad de cada uno. Para esto, anteriormente se debe tener una base de imágenes de entrenamiento de cada estudiante para poder realizar la comparación y posteriormente el reconocimiento. OpenCV proporciona algoritmos para identificar personas como son:[28]

- **LBPH (Histogramas de Patrones Binarios Locales):** Algoritmo que utiliza descriptores e histogramas para indicar que región del rostro es más significativa, posteriormente se comparará con otras imágenes ver la similitud que poseen y consecuentemente identificar a la persona.
- **FisherFaces**: “Es una técnica de reconocimiento de rostros, que tiene en cuenta la luz y las expresiones faciales. Se encarga de reducir las dimensiones de las caras utilizando el método FLD (Discriminant Lineal Fisher). En este sentido, el análisis discriminante de Fisher intenta proyectar los datos de manera que su nueva dispersión sea óptima para la clasificación.”. [28]
- **EigenFaces**: utiliza ortogonalidad dimensional para distinguir que vectores del rostro ofrecen más información para una matriz de datos multidimensional. El reconocimiento se da a partir de la proyección del rostro de la persona sobre un subespacio en el que se comparan las dimensiones de los rostros ya entrenados y el que se quiere reconocer.

Para este proyecto se escogió utilizar el algoritmo LBPH ya que es del que más información se proporciona al ser un método sencillo y sin mucha carga computacional, por lo que es el más utilizado.

1.3.5.2.1 Algoritmo LBPH

El algoritmo de Patrones Locales Binarios (LBP) permite etiquetar pixeles de la imagen, limitando su vecindad con otros y considerando el resultado obtenido como un número binario, su función principal es la clasificación de texturas y fue mencionado por primera vez en 1994.

Al pasar el tiempo y con las diferentes pruebas que han realizado con este algoritmo, se descubrió que si se lo usa con histogramas [8] descriptores de gradientes orientados (HOG), se puede lograr representar imágenes del rostro de una persona con un simple vector.
Este algoritmo necesita definir 4 parámetros fundamentales para su ejecución, en la Tabla 1.4 se muestran cuáles son [29]:

Tabla 1.4 Parámetros LPBH

Parámetro	Descripción
Radio	"Se lo utiliza para construir el patrón binario local circular y es el radio de pixeles alrededor del píxel central. Por lo general, se establece en 1" [29]
Vecinos	Son los pixeles que están dentro del radio definido anteriormente y son utilizados para construir el LBP circular. Por lo general se utilizan 8 vecinos si se posee un radio de 1.
Cuadrícula X	Número de celdas en el eje horizontal, mientras más celdas se ocupen, mayor será la dimensión de vector resultante. Su valor por defecto es 8
Cuadrícula Y	Número de celdas en el eje vertical, mientras más celdas se ocupen, mayor será la dimensión de vector resultante. Su valor por defecto es 8

Una vez definidos que parámetros necesitamos para el funcionamiento de este algoritmo se puede arrancar con el procesamiento. A continuación, se explicarán los pasos que realiza LBPH: [29]

1. **Entrenamiento del algoritmo**: Inicialmente este algoritmo necesitará de un entrenamiento con imágenes de rostros de las personas a identificar, las cuales ya se obtuvieron mediante el algoritmo Haar Cascade. Cada una de estas imágenes deberán estar en escala de grises y también deben estar etiquetadas con un identificador de la persona a las que pertenecen. El identificador debe ser el mismo

[8] **Histograma**: gráfica de barras que presenta la frecuencia con la que se repite un valor.

para cada imagen de una misma persona. Finalizado este entrenamiento se puede proceder al procesamiento de los rostros.

2. **Operaciones LBP**: el primer paso del procesamiento es crear una imagen intermedia que describa mejor a la original, este proceso se debe realizar tanto para las imágenes de entrenamiento como para el rostro a reconocer. Con la imagen intermedia se deberá resaltar las características faciales; para crearla se utiliza una ventana deslizante que toma en cuenta los parámetros de radio y pixeles vecinos. En la Figura 1.8 se muestra el proceso a seguir:

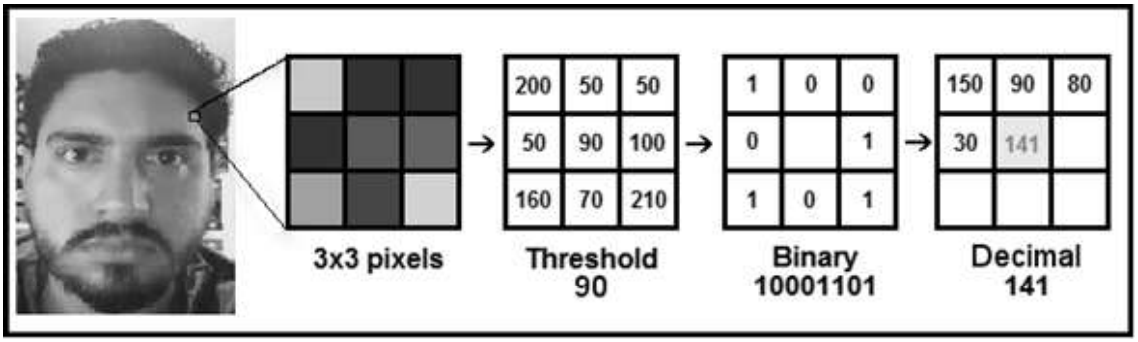

Figura 1.8 Proceso de obtención de imagen intermedia

En la Figura 1.10, se ha tomado como ejemplo una imagen en escala de grises, en la que se ha escogido como ventana deslizante una matriz de 3x3 con parámetros de radio de 1 y vecindad de 1. Posteriormente se obtiene el valor de cada píxel en un rango de 0 a 255, porque estamos trabajando con escala de grises. De la matriz con los valores numéricos se obtiene el valor central que será el umbral con el que se trabajará y tiene un valor de 90.

Con el umbral se calcularán los nuevos valores de los vecinos, para esto se asigna a cada vecino un valor de 1 si supera el umbral y un 0 si es inferior, aquí ya se ha formado el patrón binario circular. Posteriormente se deberá concatenar fila por fila los valores binarios calculados, exceptuando el valor central; el número binario obtenido en el ejemplo es 10001101. Finalmente, el número binario se lo transforma a decimal y este pasa a ser un nuevo píxel que representa la matriz 3x3 en la imagen.

Si se trabaja con otros valores de radio y vecindad, se deberá utilizar interpolación bilineal para encontrar un valor que represente los valores vecinos de una malla cuadrada.

3. **Extracción de los histogramas:** con la imagen LBP intermedia resultante, se puede utilizar los parámetros de cuadricula X, Y para dividir la imagen en varias

matrices de pixeles (Regiones). En la Figura 1.9 se muestra cómo se subdivide la imagen intermedia.

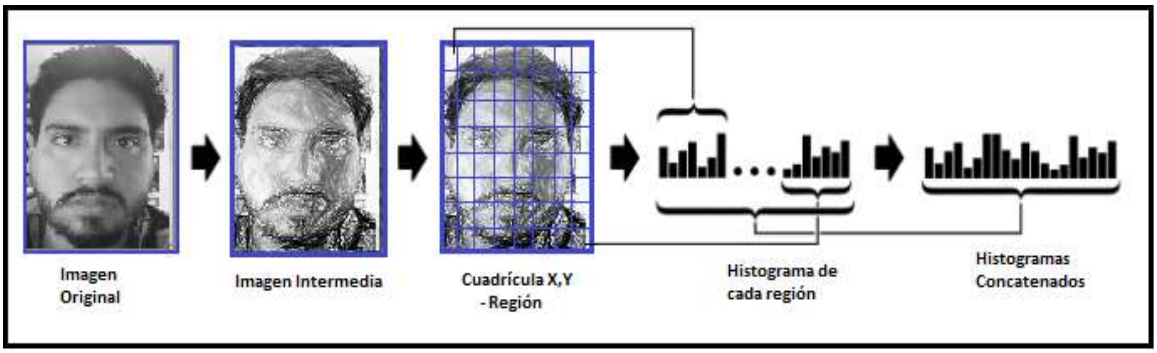

Figura 1.9 Separación de histogramas y concatenación.

A partir de la división en regiones de la imagen intermedia se pueden extraer los histogramas de cada una, se tendrá 256 posiciones posibles ya que se trabaja con imágenes en escala de grises y cada una representará la intensidad de píxel. Después se debe concatenar todos los histogramas ya obtenidos, para encontrar uno nuevo que represente las características más importantes de la imagen original. Una vez realizados estos pasos para las imágenes de entrenamiento se obtendrán un número de histogramas igual a la cantidad de imágenes de entrenamiento; y un histograma para comparar e identificar quien es la persona de la imagen.

4. **Reconocimiento Facial:** al obtener los histogramas del entrenamiento por una parte y de la imagen a reconocer por otra, se puede utilizar varios métodos para calcular la distancia entre los dos, por ejemplo: la distancia Euclidiana, chi-cuadrado, valor absoluto, entre otras. El método más común es la distancia Euclidiana la cual utiliza la fórmula 1.1:

$$D = \sqrt{\sum_{i=1}^{n}(h1_i - h2_i)^2} \qquad (1.1)$$

Donde:

- D = distancia entre histogramas.
- H1= valor del histograma imagen a detectar.
- H2= valor del histograma de imagen de entrenamiento.
- N= número de valores en cada histograma.

Una vez terminado este proceso, el algoritmo nos retornará el histograma etiquetado de la persona que posea una menor distancia con respecto al de la imagen a reconocer, ya que esto significa que es el más parecido. Se deberá definir un valor máximo de tolerancia para aceptar que el resultado del algoritmo fue el correcto. [29]

1.3.6 ARQUITECTURA Y DISEÑO ANDROID

Android es una pila de software que posee un sistema operativo, un middleware y aplicaciones para los usuarios. Fue creado por Google y está destinado a operar en dispositivos móviles inteligentes con pantalla táctil. Su arquitectura se basa en un modelo de capas, en la que cada capa utiliza servicios de la otra. Su desarrollo fue basado en el kernel de Linux y otros softwares de código abierto. En la Figura 1.10 se muestra el diagrama de su arquitectura.[30]

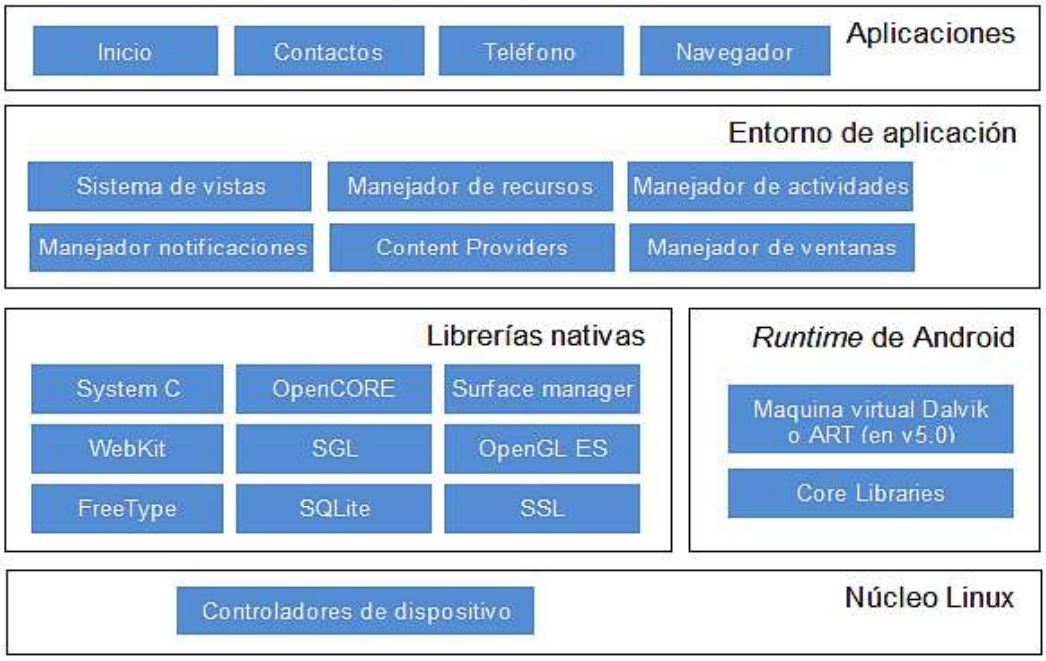

Figura 1.10 Arquitectura de Android [30].

- **Kernel de Linux**: también llamado núcleo, es el encargado de proporcionar servicios como seguridad, administración de memoria, manejo de hilos y multiproceso, uso de la pila de protocolos del sistema, además de conectividad y control de drivers para los componentes de hardware. Esta capa al relacionarse directamente del hardware será la única que dependa del mismo.[31]
- **Runtime Android**: funciona como una máquina virtual de Java inicialmente llamada Dalvik y luego sustituida por ART, está diseñada para operar con poca memoria y procesadores limitados, tiene funciones de optimización de recursos, se basa en registros y permite crear instancias independientes para que cada aplicación corra un proceso. También delega al kernel funciones como threading y manejo de memoria de bajo nivel.[31]

- **Librerías Nativas**: Incluye varias librerías escritas en C/C++ que son indispensables para el manejo de componentes y servicios de Android. También disponen de APIs para el fácil acceso de las aplicaciones a estas librerías nativas, las más importantes son: Sytem C Library, Surface manager, Webkit/Chromium, SGL, Media Framework, Librerías 3D, FreeType, SQLite (motor de base de datos), SSL.[32]
- **Entorno de aplicación**: esta capa de Android permite simplificar la reutilización de componentes, ya que cada aplicación puede disponer de sus recursos para que otras las utilicen, también se permite la modificación de componentes ya desarrollados por el sistema. Estas componentes aprovechan las funcionalidades compatibles de librería de Java. [31]
- **Aplicaciones**: Son los programas que corren en la máquina virtual de Android y proporcionan interacción con el usuario o con el sistema. Por lo general están escritas en java o Kotlin, pero si se desea programar con C/C++ se puede utilizar la API nativa NDK (*Native Development Kit*). [31] [32]

1.3.6.1 Diseño de aplicaciones Android

Para el desarrollo de aplicaciones se tiene un paquete de software llamado Andorid SDK, el cual incluye las herramientas necesarias para el desarrollo (librerías y APIs), depuración, emuladores, compiladores y documentación. Para acceder al SDK se lo puede hacer desde una línea de comandos o un terminal.

Actualmente se han implementado Entornos de Desarrollo Integrados (IDE)[9] para facilitar el desarrollo y ejecución de programas, el más conocido es Android Studio, el cual es recomendado por Google e incluye el SDK de Android y la máquina virtual de Java necesaria para poder correr código Java en el dispositivo. Por tal motivo este proyecto se desarrolló con este IDE. [33]

Android Studio es la plataforma de desarrollo oficial de Android y se lanzó por primera vez al mercado en 2014. Anteriormente se utilizaba Eclipse y se debían instalar algunos Plug Ins (extensiones). Está basado en el software IntellJ IDEA de JetBrains y posee una licencia Apache 2.0. Entre sus principales características permite la construcción de proyectos

[9] IDE: Es un software que facilita un entorno de desarrollo para la programación, depuración, compilación y documentación de aplicaciones.

mediante la herramienta Gradle, permite visualizar un Layout en varias simulaciones de dispositivos y usa herramientas de gestión como Github.[33]

1.3.6.2 Sockets en Android

Android al utilizar el lenguaje de programación Java, para el desarrollo de sus aplicaciones, utilizará la librería de Sockets proporcionada por su máquina virtual. En la sección 1.3.2.2 (Sockets para Python), se abordó como es el funcionamiento de los Sockets en general, ya que siempre habrá un cliente y un servidor que realicen peticiones uno al otro mediante un protocolo de comunicación de capa 4 ya sea TCP o UDP e indicando una IP y un puerto. Lo cual se cumple también para Java.

También existen APIs y Frameworks que permiten realizar comunicaciones entre dispositivos ya sea para un servicio web o para conectarse a un servidor. Estas herramientas facilitan la conexión ya que trabajan a un nivel de capa aplicación, pero requieren que en los dos lados de la comunicación se tenga los mismos métodos de serialización para enviar información, en este proyecto el Servidor se maneja con Python el cual posee una serialización de objetos muy distinta a la que tiene el Cliente que es Android, existen librerías que permiten manejar objetos Java serializados en Python, pero estos utilizan un mapa de traducción que no es 100% confiable, pueden traducir objetos con un margen de error pequeño, el cual puede influir negativamente sobre el proyecto. Esta es la razón de utilizar una comunicación a bajo nivel (sockets), ya que no se necesita compatibilidad entre lenguajes, sino solo compartir un mismo protocolo de comunicación lo cual está estandarizado. En el caso de los Frameworks se puede llegar a tener comunicación, pero estos automáticamente serializarán y deserializarán los objetos lo cual puede ocasionar errores.

Otro aspecto importante para utilizar socktes es que tanto como para Python y Android las librerías han simplificado en gran parte sus funciones, ahora una sola sentencia puede ayudarnos a inicializar el servidor y otra permitirnos conectarnos al mismo desde el cliente.

1.3.7 METODOLOGÍAS ÁGILES DE DESARROLLO

Las metodologías ágiles de desarrollo son procedimientos para realizar un proyecto de software, de tal manera que se pueda entregar al cliente un producto final de manera rápida y eficiente. Estos métodos son utilizados para sistemas que sufren cambios constantemente durante la fase de desarrollo, ya que se entrega al cliente constantemente los avances del proyecto y se pueden generar cambios. [34]

Las metodologías ágiles de desarrollo tienen algunos principios con los que trabajan. En la Tabla 1.5 se presentan dichos principios: [34]

Tabla 1.5 Principios de las metodologías ágiles.

Principio	Descripción
Participación del Cliente	Los clientes se los debe incluir en todo el proceso, para que evalúen los avances y sugieran cambios de ser necesario
Entregas Incrementales	El software se debe desarrollar en iteraciones incrementales, donde el cliente debe especificar sus requisitos a incluir en la siguiente iteración.
No Procesos Generales	Se debe respetar el estilo de cada integrante del equipo de desarrolladores, ya que imponer un proceso general puede provocar que la persona disminuya su eficiencia.
Aceptar Cambios	El sistema se debe desarrollar pensado en que puede haber cambios en sus requerimientos, por lo que debe poder dar paso a dichas modificaciones.
Mantener la Simplicidad	Siempre se debe apuntar a dar soluciones simples ante los problemas que se presenten en el proyecto.

1.3.7.1 Metodología de desarrollo ágil Scrum

Scrum es una metodología ágil y fue desarrollada por Ikuro Nonaka e Hirotaka Takeuchi en la década de los 80. Se basa principalmente en la organización de pequeños grupos de trabajo auto organizados que gestionan su trabajo en base a entregas parciales llamadas Sprints, con el objetivo de que el cliente sea uno de los supervisores del producto final y evitar complicaciones en el proceso de desarrollo. [35]

Scrum posee un grupo de trabajo llamado “Scrum Team”, en la Tabla 1.6 se muestra los integrantes de dicho grupo:[35]

Tabla 1.6 Integrantes Scrum Team

Rol a desempeñar	Descripción
Product Owner	Es el cliente o auspiciante del proyecto

ScrumMaster	Supervisor y jefe del proyecto, es el encargado de velar por el cumplimiento de los Sprints, tiempos de entrega y debe resolver los problemas que se generen en el proyecto.
Equipo de Desarrollo	Son los programadores y testers del sistema.

Una base fundamental de Scrum son las reuniones del equipo de trabajo para controlar el proceso de desarrollo del proyecto, para esto existen reunión prescritas, en la Tabla 1.7 se las detalla.

Tabla 1.7 Reuniones Scrum

Reunión	**Descripción**
Sprint Planning Meeting	Reunión de planificación de todo el proyecto, en especial de los Sprint
Dialy Scrum	Reunión de seguimiento diario
Sprint Review	Reunión para revisión de avances
Sprint Retrospective	Reunión para analizar cómo fue la entrega y que se debe cambiar para la siguiente Sprint Review.

El esquema de funcionamiento de un proyecto desarrollado con Scrum se muestra a continuación en la Figura 1.11:

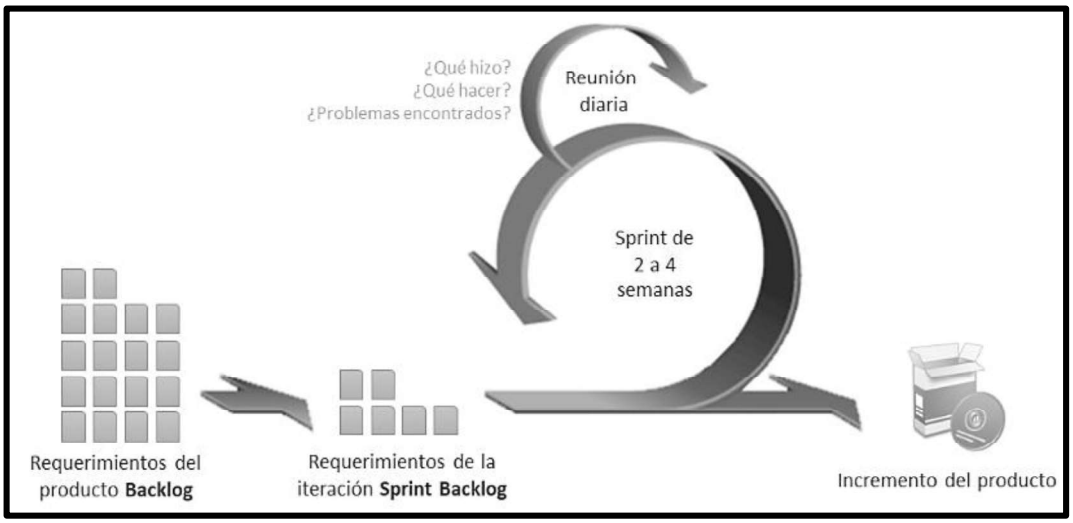

Figura 1.11 Esquema de un Proyecto Scrum [39]

El proceso que se muestra en la Figura 1.11 es el siguiente:

1. Empieza cuando el Product Owner define los requerimientos del sistema que se va a desarrollar, a esto se lo llama Product Backlog. El Scrum Master será el encargado de apoyar al cliente para definir de una forma coherente dichos requerimientos.
2. Los requerimientos se los descompone en pequeños Sprints Backlogs (paquetes de trabajo) y se les asigna un tiempo para ser completados, por lo general el tiempo es de 2 a 4 semanas. Los Sprints Backlogs se los organiza en una reunión Sprint Planning Meeting.
3. El equipo de desarrollo emplea el método de trabajo que más domine, es decir los programadores serán los encargados de decidir los detalles de la programación. También tendrán un Dialy Scrum (reunión diaria), en la que cada integrante expondrá sus avances, las dificultades encontradas y se discutirán las posibles soluciones.
4. Al terminar un Sprint se realiza una Sprint Review para presentar el producto resultante del Sprint Backlog, en esta reunión estará presente el cliente que evaluará el avance, posterior a esto se puede realizar un Sprint Retrospective para analizar que se puede mejorar y como se lo podría lograr.
5. Luego de evaluar el Sprint y el avance, el Product Owner propondrá un siguiente Product Backlog, este proceso se repetirá hasta terminar todo el proyecto.

Para el presente proyecto se adaptará SCRUM al organizar todo el trabajo en Sprints, organizar los integrantes del proyecto en un Equipo SCRUM y sobre todo probar el sistema a medida que se lo implementa.

2. METODOLOGÍA

En este capítulo se mostrará el desarrollo del prototipo. Primero se realizará el diseño del sistema, para esto se definirá la estructura de su arquitectura, se analizarán los requerimientos tanto funcionales como no funcionales, se modelará el prototipo en general y se organizarán los Sprints de trabajo. Como segundo punto se tendrá la fase de implementación; en la cual, siguiendo los Sprints ya definidos, se procederá a implementar la base de datos, codificar los módulos del sistema tanto del servidor como del cliente y finalmente se pondrá en marcha al sistema prototipo.

2.1 DISEÑO DEL PROTOTIPO

En esta sección se realizará el análisis de requerimientos, luego se definirán las historias de usuario, se realizarán los diagramas de casos de uso, se definirá la arquitectura del prototipo, se escogerán las herramientas de desarrollo, se organizará el Product Backlog y finalmente se diseñará el sistema en base a diagramas de historias de usuario, casos de uso, diagramas de clases, secuencia y de actividades.

2.1.1 ANÁLISIS DE REQUERIMIENTOS

Los análisis de requerimientos se los realizó en base a entrevistas realizadas a 5 docentes de la Escuela Politécnica Nacional, el formato de la entrevista se encuentra en el ANEXO A.

Como resultado de las entrevistas se ha obtenido un resumen de los mismos mostrados en la Tabla 1.1, además acoplándose al alcance propuesto para este trabajo de titulación se pudieron obtener los siguientes requerimientos:

- Permitir el control de asistencia de estudiantes mediante reconocimiento facial de fotografías de estudiantes presentes en una actividad académica.
- Diseñar un sistema que centralice la información en un servidor.
- Realizar un CRUD (crear, renombrar, actualizar y borrar) que permita la administración de estudiantes.
- Realizar un CRUD que permita la administración de profesores.
- Considerar el manejo de un usuario Administrador.
- Considerar el manejo de un usuario Profesor.
- Llevar el registro de asistencia clasificado por paralelos según la asignatura dictada.
- Permitir al usuario Administrador modificar todos los aspectos de la aplicación.

- Organizar la asistencia de los estudiantes según la actividad académica, identificada por la fecha en la que se desarrolló.
- Permitir que un profesor pueda ser asignado a varios paralelos de una misma materia
- Permitir que en un paralelo puedan registrarse estudiantes que pertenecen a varias carreras.
- Permitir que un profesor pueda ser asignado a materias y paralelos en varias facultades.
- Permitir que se disponga de un método manual para toma de asistencia, que sirva como método de respaldo.
- Permitir que un usuario pueda acceder desde varios dispositivos móviles a su respectivo perfil y disponer de los servicios permitidos al mismo. Se debe permitir movilidad de usuarios.
- Permitir la generación de un reporte final de asistencia en un documento Excel.
- Enviar el reporte de asistencia a una dirección de correo electrónico.
- Permitir que un estudiante solo sea asignado a una carrera.
- Considerar que se puedan manejar varias facultades y carreras.
- Realizar un CRUD para actividades académicas.
- Manejar más de una fotografía por actividad académica.
- Ingresar fotografías de estudiantes para el entrenamiento del sistema.
- Almacenar las fotografías tomadas para el reconocimiento facial, como respaldo de la asistencia de los estudiantes.

Tabla 2.1 Resultado De Entrevistas

N°	Pregunta	Docente					Total
		1	2	3	4	5	
1	Como docente universitario, ¿Lleva un registro de asistencia de estudiantes por cada materia en la que está a cargo?, ¿de qué forma lo hace?	si Manual	si Digital	si Digital	si Digital	si Digital	Si (100%) - No (0%)
2	¿Cuánto tiempo tarda aproximadamente en tomar la asistencia de una actividad académica?	5 min	3 min	6 min	8 min	5 min	5.4 min
3	¿Conoce algún sistema para control de asistencias que utilice reconocimiento facial?	Si	Si	No	No	Si	Si (60%) - No (40%)
4	¿Considera que sería útil tomar la asistencia a estudiantes mediante reconocimiento facial de una o más fotografías de los asistentes?	si	si	si	si	si	Si (100%) - No (0%)

5	¿Considera necesario el uso de dispositivo móvil para ingresar los datos de la asistencia y una base de datos para almacenarlos?	si	si	si	si	si	Si (100%) - No (0%)
6	¿Considera importante que el sistema tenga la opción de autenticarse mediante un usuario y una contraseña?	si	si	si	si	si	Si (100%) - No (0%)
7	¿Cree usted que las fotografías enviadas al servidor deben almacenarse como respaldo de la información obtenida?	si	si	si	si	si	Si (100%) - No (0%)
8	Para el reconocimiento facial de una persona es importante tener una base de datos de fotografías previas, con las cuales el sistema se entrena. ¿Usted estaría dispuesto a entrenar al sistema antes de empezar a tomar la asistencia de los estudiantes?	si	si	si	si	si	Si (100%) - No (0%)
9	¿Considera importante que al final del periodo académico o cuando usted lo considere necesario, pueda generar un reporte la materia que ha impartido?	si	si	si	si	si	Si (100%) - No (0%)

2.1.1.1 Especificación de requerimientos funcionales y no funcionales

Luego de haber obtenido los requerimientos mediante las respectivas entrevistas se procede a discriminar los requerimientos funcionales y no funcionales.

2.1.1.1.1 Requerimientos funcionales

Los requerimientos funcionales definirán cuales son los servicios que deberá incluirse en el prototipo. Se definieron los siguientes:

- El usuario administrador podrá crear, eliminar y modificar a otros usuarios administradores y profesores.
- El usuario administrador podrá crear eliminar y modificar materias, paralelos para el sistema y asignar a profesores a los mismos.
- El administrador podrá crear, eliminar y modificar actividades académicas.
- El administrador podrá crear, eliminar y modificar estudiantes.
- El administrador podrá inicializar el servidor.
- El administrador podrá crear, eliminar y modificar facultades y carreras.
- El administrador podrá realizar la toma de asistencia de estudiantes de cualquier paralelo, utilizando el método de reconocimiento facial o método manual.
- El usuario administrador tendrá el control total del sistema.

- El administrador podrá ingresar fotografías de estudiantes para el entrenamiento del sistema y posteriormente entrenarlo según estudiante o el sistema en total.
- El usuario Administrador podrá generar reportes de asistencia de cualquier paralelo – materia.
- El usuario profesor podrá crear, eliminar y modificar estudiantes.
- El usuario profesor podrá crear, eliminar y modificar actividades académicas.
- El usuario profesor podrá realizar la toma de asistencia de estudiantes de los paralelos a los que ha sido asignado, utilizando el método de reconocimiento facial o método manual.
- El usuario profesor podrá generar reportes de asistencia de los paralelos – materias a los que fue asignado.
- El usuario profesor podrá cargar en el sistema fotografías de estudiantes para el entrenamiento del sistema y posteriormente entrenarlo estudiante por estudiante.
- Cualquier usuario puede acceder a su perfil desde cualquier dispositivo móvil que posea instalada la aplicación cliente.
- Al momento de generar un reporte se podrá enviar a cualquier dirección de correo electrónico.
- Al momento de realizar la captura de imágenes para el reconocimiento facial se debe manejar más de una fotografía por actividad académica.
- El servidor aceptará la autenticación de administradores mediante su usuario y clave de seguridad.
- La aplicación cliente aceptará la autenticación de usuarios profesores y administradores.
- Los datos deberán almacenarse en una base de datos y estar disponible para todos los usuarios.
- Al momento de realizar la toma de fotografías de los estudiantes presentes, la aplicación debe permitir seleccionar fotografías de la galería del dispositivo.
- El servidor deberá almacenar las fotografías tomadas por el cliente, como respaldo de la información.
- El usuario tanto para profesores y administradores es su número de cédula.
- Se manejarán mensajes de aviso cuando algún dato sea mal ingresado o no se haya podido completar alguna solicitud.
- La aplicación servidor deberá poder levantar su servicio aunque haya cambiado su IP y/o Puerto.
- El cliente debe poder conectarse al servidor, si este ha cambiado su IP y puerto.

- Las claves de acceso deberán no poder visualizarse en el cliente.
- Se deberá mostrar mensajes de error en la aplicación cliente cuando un proceso no se logre cumplir.

2.1.1.1.2 Requerimientos no funcionales

Los requerimientos no funcionales definirán cuales son las propiedades y restricciones prototipo, es decir mostrarán como se cumplirá el funcionamiento del sistema [41]. Se definieron los siguientes:

- Al momento de ingresar números de cédulas solo se aceptarán números.
- Al momento de ingresar direcciones de correo electrónico se deberá validar que estas sean correctas.
- Los mensajes de error deberán ser Alert Dialog
- Validar que las fechas ingresadas tenga un formato valido.
- Mostrar mensaje de error al tener un autenticación fallida.

2.1.2 HISTORIAS DE USUARIO

Las historias de usuario son herramientas utilizadas para especificar las funcionalidades del software o proyecto a desarrollar y describirán los requerimientos proporcionados por el cliente [42]. Como este proyecto se basa en la metodología Scrum, el Product Owner será el encargado de proporcionar la información de funcionalidades al Scrum Manager.

Para realizar las historias de usuario se implementará el formato de ficha o formulario indicado en la Tabla 2.2 [36]:

Tabla 2.2 Formato de historia de usuario

HISTORIA DE USUARIO		
ID:	**Usuario:**	
Programador Responsable:	**Sprint:**	**Número Sprint:**
Nombre Historia:	**Prioridad:**	
Descripción:		
Comentarios:		

Los campos de la Tabla 2.2 son:

- ID: código identificador de la historia de usuario.
- Usuario: indica el usuario que utilizará la funcionalidad a describir.
- Programador Responsable: nombre de la persona encargada de programar esta funcionalidad.
- Sprint: nombre del Sprint en el que se programará esta funcionalidad.
- Número Sprint: número del Sprint en el que se programará esta funcionalidad.
- Nombre Historia: título que describa a la funcionalidad.
- Prioridad: nivel de importancia de la funcionalidad, se tiene tres niveles de prioridad (baja, media y alta).
- Descripción: se describe brevemente y de forma concisa la funcionalidad
- Comentarios: se indica información adicional para entender mejor o aumentar detalles a la funcionalidad.

2.1.2.1 Detalle de historias de usuario.

En la Tabla 2.3 se indica un ejemplo de historia de usuario para el LogIn de los usuarios en la ampliación cliente:

Tabla 2.3 Historia de Usuario

HISTORIA DE USUARIO		
ID: UH-01	**Usuario:** Administrador, Profesor	
Programador Responsable: Luis Félix	**Sprint:** LogIn	**Número Sprint:** 1
Nombre Historia: LogIn en Cliente	**Prioridad:** Alta	
Descripción: Los usuarios deben poder autenticarse con un nombre de usuario y contraseña en la aplicación cliente.		
Comentarios: El usuario deberá ser su número de cédula y no podrá existir dos usuarios del mismo tipo con el número de cédula repetido.		

Las historias de usuario de este proyecto debido a su extensión se adjuntan en el Anexo B.

2.1.3 DIAGRAMA DE CASOS DE USOS

En base a los requerimientos definidos en la sección 2.1.1 se puede implementar el diagrama de casos de uso. Con este diagrama posteriormente se definirán los roles de los usuarios y cuál será su función dentro del sistema. En la Figura 2.1 se incluye el diagrama de caso de usos.

Algunos casos de uso se les ha asignado una relación <include> con tareas que serán parte de los mismos, por ejemplo cuando se realiza un CRUD de un Paralelo, se deberá asignar un profesor al mismo.

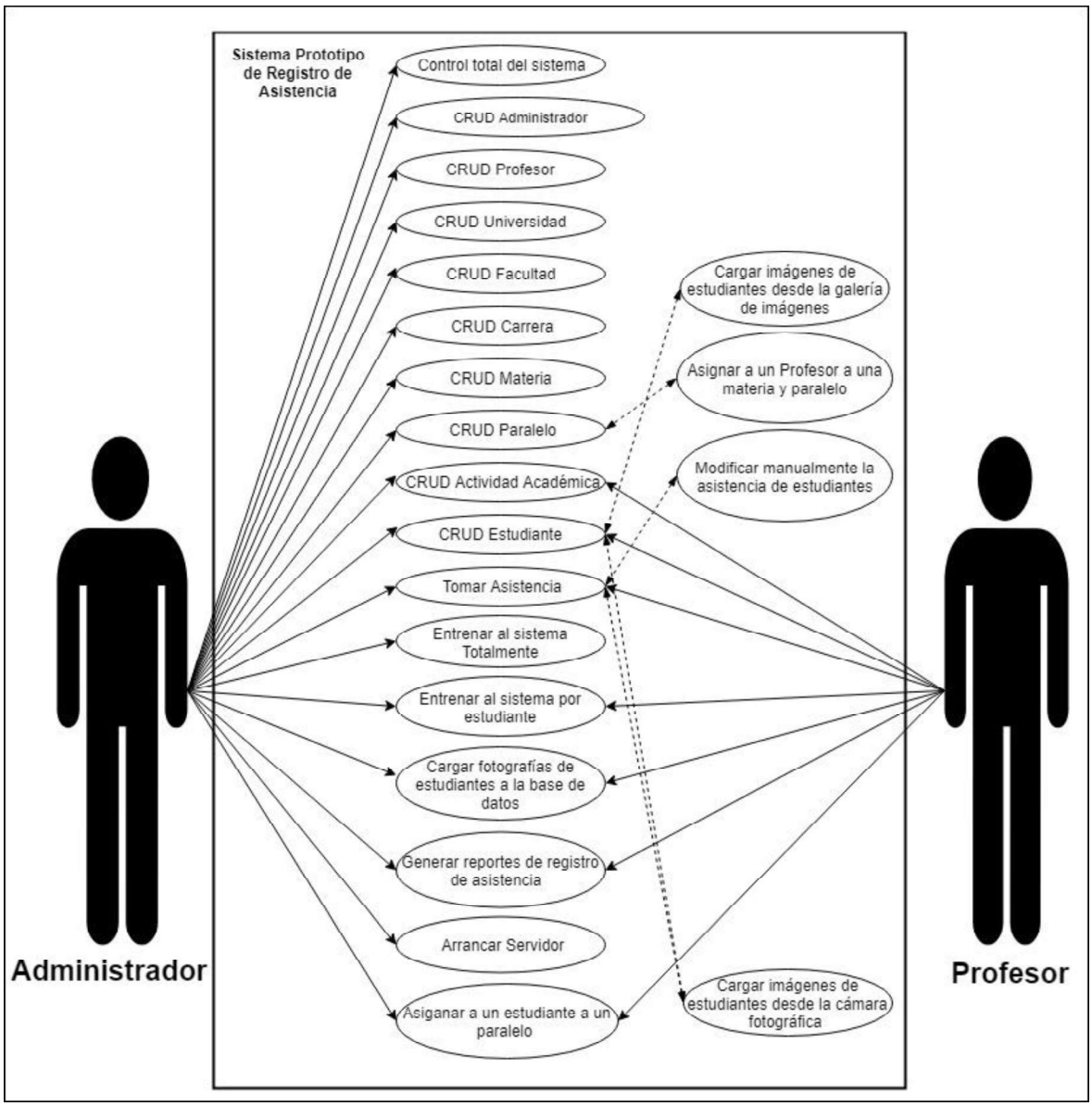

Figura 2.1 Diagrama de caso de usos

2.1.3.1 Roles de usuarios

En base al diagrama de uso de casos se identificaron dos tipos de usuarios para el sistema, en la Tabla 2.4 se indica el rol del usuario y su descripción dentro de la aplicación.

Tabla 2.4 Roles y funciones

Rol	Descripción
Administrador	• Es el encargado de administrar todo el sistema, tendrá máximos privilegios y podrá acceder a todas las funciones del sistema. • Podrá realizar el CRUD de otros usuarios tanto administradores como profesores, podrá realizar el CRUD de universidad, facultad, carrera, materia, paralelo, estudiantes y actividad académica para organizar las asistencias y a los estudiantes. • Podrá realizar la asignación de estudiantes a un paralelo y por ende a su materia asociada. • Podrá realizar la asignación de profesores a un paralelo y por ende a su materia asociada. • Podrá realizar el entrenamiento total y parcial de las fotografías de los estudiantes en el sistema. • Podrá generar reportes de los diferentes paralelos y materias, además de enviarlos vía mail. • Podrá inicializar el servidor. • Podrá realizar la toma de asistencia de estudiantes de cualquier paralelo y materia. • Podrá cargar imágenes desde la cámara del dispositivo móvil y desde la galería de imágenes del mismo.
Profesor	• Podrá realizar el CRUD de estudiantes y actividad académica para organizar las asistencias y a los estudiantes. • Podrá realizar la asignación de estudiantes a un paralelo y por ende a su materia asociada. • Podrá realizar el entrenamiento parcial de las fotografías de los estudiantes en el sistema. • Podrá generar reportes de las materias y paralelos a la cual fue asignado, además de enviar dichos reportes vial mail.

	• Podrá realizar la toma de asistencia de estudiantes de las materias y paralelos a la cual fue asignado. • Podrá cargar imágenes desde la cámara del dispositivo móvil y desde la galería de imágenes del mismo.

2.1.4 ARQUITECTURA DEL PROTOTIPO

La arquitectura de este sistema prototipo está conformada por un servidor y uno o más clientes, los cuales se conectarán mediante una red local. El servidor deberá aceptar las solicitudes de los clientes y responderá a las mismas. El servidor será una consola programada en Python, que se conectará a una base de datos en MySQL para obtener los datos requeridos por los clientes.

Los usuarios del prototipo manejarán una aplicación Android cliente que correrá sobre un dispositivo móvil. Será la encargada de ingresar todos los datos para que sean enviados y procesados por el servidor, tomará las fotografías para el registro de asistencia e ingresará los nuevos usuarios y las diferentes abstracciones que se implementarán para el correcto funcionamiento del sistema.

Mediante una petición de un cliente para generar un reporte de asistencia, el servidor se conectará con un servidor SMTP para enviar dicho reporte vía mail. En la Figura 2.2 se muestra el diagrama del sistema prototipo propuesto.

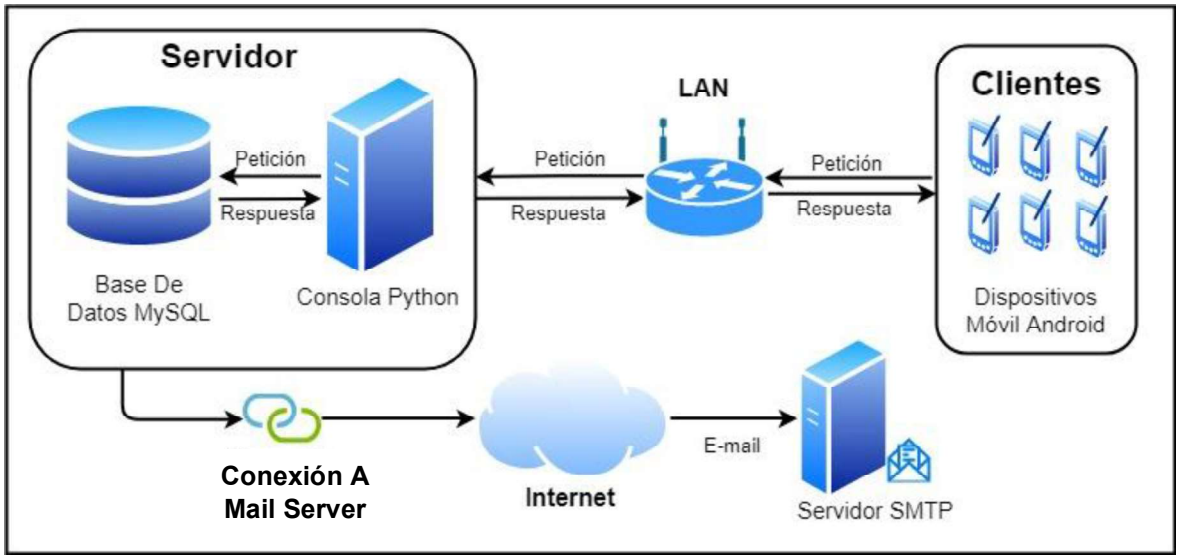

Figura 2.2 Arquitectura del Prototipo

2.1.5 HERRAMIENTAS DE SOFTWARE PARA EL DESARROLLO

En la Tabla 2.5 se indican las herramientas utilizadas para el desarrollo de este sistema prototipo:

Tabla 2.5 Herramientas de desarrollo.

Herramienta		Descripción
Servidor	**Base de datos.**	Para la base datos se utilizará la DBMS MySQL v8.0.13, se utilizará el IDE MySQL Workbench v8.0.13, el cuál brindará una interfaz gráfica para poder manipular la base de datos.
	Implementación de la consola del servidor y su lógica.	Para la programación del servidor se utilizará Python v3.6.5, el cual será instalado mediante Anaconda; como conector para la base de datos se utilizará MySQL-Python 1.2.5. El editor de desarrollo a utilizar es Sublime Text 3.
	Procesamiento de imágenes y Reconocimiento Facial.	Para el procesamiento de imágenes y reconocimiento facial se utilizará la librería OpenCV v.3.3.1; también se utilizará la librería NumPy 1.14.3 para el procesamiento de dato matemáticos y de matrices.
	ORM para la base de datos.	Como ORM se utilizará la librería PEEWE v.3.6.4 para la traducción de la base relacional a objetos.
	Generación de Reportes de Asistencia.	Para el reporte de asistencia se generará un archivo Excel a partir de la librería “openpyxl v.2.5.3” para Python.
	Envío de reporte de asistencia	Para el envío del reporte de asistencia se utilizará la librería SmtpLib para Python para conectarse con un servidor SMTP y poder enviarlo vía mail.
Cliente	**Aplicación cliente.**	El cliente se programará con ayuda del IDE Oficial de Android llamado Android Studio v.3.0, además se desarrollará para la API 21 en adelante y con el lenguaje de programación Java.

2.1.6 PRODUCT BACKLOG

Para el producto Backlog se utilizaron como referencia las historias de usuario de la sección 2.1.2. En la Tabla 2.6 se lo detalla:

Tabla 2.6 Detalle del Product Backlog

ID de Historia de Usuario	Nombre de Historia de Usuario
UH-01	LogIn en cliente.
UH-02	LogIn en servidor.
UH-03	Arrancar servidor.
UH-04	Gestión de usuarios administradores.
UH-05	Gestión de usuarios profesores.
UH-06	Gestión de abstracciones: universidad, facultad, carrera y materia.
UH-07	Gestión de abstracciones: estudiante.
UH-08	Gestión de abstracciones: paralelo.
UH-09	Asignar estudiante a paralelo.
UH-10	Gestión de abstracciones: actividad académica.
UH-11	Gestión de lista de estudiantes presentes.
UH-12	Adición de fotografías de estudiantes en base de fotografías.
UH-13	Entrenamiento del sistema por estudiante.
UH-14	Entrenamiento total del sistema.
UH-15	Tomar asistencia.
UH-16	Generar reporte de asistencia

UH-17	Base de datos

2.1.7 TEAM SCRUM

Para el desarrollo de este prototipo se ha defino ciertos roles para las personas que intervendrán. En la Tabla 2.7 se muestra el rol, el nombre de la persona y su función.

Tabla 2.7 Team Scrum

Rol Scrum	Nombre	Funciones
Product Owner	Franklin Sánchez M.Sc.	Responsable de establecer los requerimientos para el sistema, establece tiempos de entrega y evalúa los mismo.
Scrum Máster	Franklin Sánchez M.Sc.	Responsable de verificar que se cumplan los tiempos de los Sprints, verifica que las actividades desarrolladas cumplan con los objetivos y si surge algún cambio es el encargado de tomar la decisión final.
Equipo de Desarrollo	Luis Félix	Encargado de la implementación del sistema prototipo, se deberá guiar en base a los requerimientos del sistema y de las historias de usuario.

2.1.8 PLANIFICACIÓN Y ORGANIZACIÓN DE LOS SPRINTS

La planificación de los Sprints se lo realizará según el Product Backlog y las historias de usuario; esta organización se la muestra en la Tabla 2.8:

Tabla 2.8 Organización de Sprints

N° Sprint	Nombre Del Sprint	Id de Historia de Usuario	Nombre de La Historia de Usuario	Tiempo Estimado
1	Base de Datos	UH-17	Base de datos	2 semanas
2	LogIn	UH-02	LogIn en servidor.	2 semanas

		UH-03	Arrancar Servidor	
		UH-01	LogIn en cliente.	
3	Usuario Administrador	UH-04	Gestión de usuarios administradores.	2 semanas
4	Abstracciones Parte 1	UH-06	Gestión de abstracciones: universidad, facultad, carrera y materia.	2 semanas
5	Usuario Profesor y Abstracciones Parte 2	UH-05	Gestión de usuarios profesores.	3 semanas
		UH-08	Gestión de abstracciones: paralelo.	
		UH-07	Gestión de abstracciones: estudiante.	
6	Abstracciones Parte 3	UH-09	Asignar estudiante a paralelo.	2 semanas
		UH-10	Gestión de abstracciones: actividad académica.	
7	Entrenar Sistema	UH-12	Adición de fotografías de estudiantes en base de fotografías.	3 semanas
		UH-13	Entrenamiento del sistema por estudiante.	
		UH-14	Entrenamiento total del sistema.	
8	Tomar Asistencia	UH-15	Tomar asistencia.	3 semanas
		UH-11	Gestión de lista de estudiantes presentes.	
9	Reporte Asistencia	UH-16	Generar reporte de asistencia	3 semanas

2.1.8.1 Sprint 1

Al Sprint número 1 se lo ha llamado base de datos en la tabla 2.9 se lo muestra con las tareas a desarrollar.

Tabla 2.9 Sprint 1

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-17	Base De Datos	Servidor	Impementar la base de datos en MySql basándose en el diagrama realcional.

2.1.8.2 Sprint 2

Al sprint número 2 se lo ha nombrado LogIn, en la Tabla 2.10 se lo muestra con las tareas a desarrollar:

Tabla 2.10 Sprint 2

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-02	LogIn en servidor.	Servidor	Crear una interfaz de usuario en consola para poder autenticar y dar acceso a un administrador en el servidor.
			Implementar el código para autenticación.
			Implementar un bucle para que el sistema permita ingresar credenciales detectadas como erróneas.
UH-03	Arrancar Servidor	Servidor	Implementar el código para que el Administrador pueda inicializar el servidor y este no termine su ejecución.
			Implementar código para que el usuario administrador, pueda desistir de arrancar el servidor.
UH-01	LogIn en cliente.	Cliente Android	Crear una interfaz de usuario para poder autenticar y dar acceso a un administrador o Profesor a la aplicación Android.
			Implementar el código para el envío de los datos de autenticación al servidor.

			Implementar un formulario para que el usuario decida a que IP y puerto del servidor desea conectarse
		Servidor	Implementar el código para que el servidor autentique al usuario y responda si posee o no acceso a la app.

A continuación, en la figura 2.3 se muestra el diagrama de actividades para el LogIn del sistema en la aplicación cliente.

2.1.8.3 Sprint 3

Al sprint número 3 se lo ha nombrado Usuario Administrador, en la Tabla 2.11 se lo muestra con las tareas a desarrollar:

Tabla 2.11 Sprint 3

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-04	Gestión de usuarios administradores	Aplicación Android	Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de otros Administradores.
			Implementar el código necesario para que el cliente envíe al servidor la petición de gestión de un usuario Administrador.
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.

A continuación, en la figura 2.4 se muestra el diagrama de actividades para la creación de un usuario Administrador

2.1.8.4 Sprint 4

Al sprint número 4 se lo ha nombrado Abstracciones Parte 1. Se lo ha denominado de esta forma ya que hace alusión a las abstracciones que se han implementado en la base de datos. En la Tabla 2.12 se lo muestra con las tareas a desarrollar:

Tabla 2.12 Sprint 4

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-06	Gestión de abstracciones: universidad, facultad, carrera y materia.	Aplicación Android	Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de la abstracción Universidad.
			Implementar el código necesario para que el cliente envié al servidor la petición de gestión de la abstracción Universidad.
			Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de la abstracción Facultad.
			Implementar el código necesario para que el cliente envíe al servidor la petición de gestión de la abstracción Facultad.
			Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de la abstracción Carrera.
			Implementar el código necesario para que el cliente envié al servidor la petición de gestión de la abstracción Carrera.
			Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de la abstracción Materia.

			Implementar el código necesario para que el cliente envíe al servidor la petición de gestión de la abstracción materia.
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.

A continuación, en la figura 2.5 se muestra el diagrama de actividades para la creación de una universidad, el mismo proceso se cumple para las demás abstracciones.

2.1.8.5 Sprint 5

Al sprint número 4 se lo ha nombrado Usuario profesor y Abstracciones Parte 2, en la Tabla 2.13 se lo muestra con las tareas a desarrollar:

Tabla 2.13 Sprint 5

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-05	Gestión de usuarios profesores.	Cliente Android	Crear una interfaz de usuario para que el administrador pueda realizar el CRUD del usuario profesor.
			Implementar el código necesario para que el cliente envié al servidor la petición de gestión del usuario profesor.
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.
UH-08	Gestión de abstracciones: paralelo.	Cliente Android	Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de la abstracción paralelo.
			Implementar el código necesario para que el cliente envié al servidor la petición de gestión de la abstracción paralelo.
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.

UH-07	Gestión de abstracciones: estudiante.	Cliente Android	Crear una interfaz de usuario para que el cualquier usuario pueda realizar el CRUD de la abstracción estudiante.
			Implementar el código necesario para que el cliente envié al servidor la petición de gestión de la abstracción estudiante.
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.

A continuación, en la figura 2.6 se muestra el diagrama de actividades para la creación de un usuario Profesor, en la figura 2.7 se muestra el diagrama de actividades para la creación de estudiantes, la abstracción paralelo tendrá un similar diagrama que el de estudiante por lo que no se lo mostrará.

2.1.8.6 Sprint 6

Al sprint número 6 se lo ha nombrado Abstracciones Parte 3, en la Tabla 2.14 se lo muestra con las tareas a desarrollar:

Tabla 2.14 Sprint 6

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-09	Asignar estudiante a paralelo.	Cliente Android	Crear una interfaz de usuario para que un Administrador o Profesor puedan realizar el CRUD para asignar a un estudiante a un paralelo.
			Implementar el código necesario para que el cliente envié al servidor la petición de asignar a un estudiante a un paralelo
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.

UH-10	Gestión de abstracciones: actividad académica.	Cliente Android	Crear una interfaz de usuario para que el Administrador pueda realizar el CRUD de la abstracción actividad académica.
			Implementar el código necesario para que el cliente envié al servidor la petición de gestión de la abstracción actividad académica.
		Servidor	Implementar el código necesario para que el servidor procese la solicitud de CRUD que haya recibido.

A continuación, en la figura 2.8 se muestra el diagrama de actividades para la asignación de un estudiante a un paralelo, los procesos CRUD de las abstracciones de este Sprint serán similares a los mostrados anteriormente en los Sprints superiores.

2.1.8.7 Sprint 7

Al sprint número 7 se lo ha nombrado Entrenar Sistema, en la Tabla 2.15 se lo muestra con las tareas a desarrollar:

Tabla 2.15 Sprint 7

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-12	Adición de fotografías de estudiantes en base de fotografías.	Cliente Android	Complementar la interfaz de usuario de estudiante para que un administrador o profesor puedan adicionar fotografías de estudiantes a la base de imágenes para su posterior entrenamiento.
			Implementar el código necesario para que el cliente envíe al servidor las fotografías de los estudiantes para que se adicionen a la base de imágenes.
		Servidor	Implementar el código necesario para que el servidor pueda acceder a las fotografías de los estudiantes.

UH-13	Entrenamiento del sistema por estudiante.	Cliente Android	Complementar la interfaz de usuario de estudiante para que un administrador o profesor puedan realizar el entrenamiento del sistema para el reconocimiento facial estudiante por estudiante.
			Implementar el código necesario para que el cliente envíe al servidor la petición entrenamiento estudiante por estudiante.
		Servidor	Implementar el código necesario para que el servidor entrene a los estudiantes seleccionados por el cliente.
UH-14	Entrenamiento total del sistema.	Cliente Android	Crear una interfaz de usuario para que el administrador pueda entrenar al sistema totalmente.
			Implementar el código necesario para que el cliente envíe al servidor la petición de entrenamiento total del sistema.
		Servidor	Implementar el código necesario para que el servidor pueda entrenar en totalidad al sistema.

En la figura 2.9 se muestra el diagrama de actividades para la adición de fotografías a la base de imágenes y el posterior entrenamiento del sistema.

2.1.8.8 Sprint 8

Al sprint número 8 se lo ha nombrado Tomar Asistencia, en la Tabla 2.16 se lo muestra con las tareas a desarrollar:

Tabla 2.16 Sprint 8

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas

UH-15	Tomar asistencia.	Cliente Android	Crear un interfaz de usuario para que un Administrador o Profesor puedan tomar imágenes de los estudiantes presentes en una actividad académica, para aplicar reconocimiento facial e identificar que estudiantes se encuentran presentes.
			Implementar el código necesario para que el cliente envíe al servidor las fotografías de los estudiantes de una actividad académica, para aplicar reconocimiento facial e identificar que estudiantes se encuentran presentes.
		Servidor	Implementar el código necesario para que el servidor pueda realizar el reconocimiento facial de los estudiantes presentes, según las fotografías enviadas para una actividad académica y pueda responder con la lista de estudiantes presentes.
UH-11	Gestión de lista de estudiantes presentes.	Cliente Android	Complementar la interfaz de usuario de Tomar Asistencia para que, una vez recibida la lista de estudiantes presentes para una actividad académica, se pueda complementar la asistencia o falta de los mismos manualmente.
			Implementar el código necesario para que el cliente envíe al servidor la petición de cambio de la lista de estudiantes presentes para dicha actividad académica.
		Servidor	Implementar el código necesario para que el servidor cambie la lista de

			estudiantes presentes con la nueva recibida por parte del cliente.

En las figuras 2.10 y 2.11 se muestran el diagrama de actividades la toma de asistencia de estudiantes para una actividad académica.

2.1.8.9 Sprint 9

Al sprint número 9 se lo ha nombrado Generar Reporte, en la Tabla 2.17 se lo muestra con las tareas a desarrollar:

Tabla 2.17 Sprint 9

Id de Historia de Usuario	Nombre de Historia de Usuario	Componente	Tareas
UH-16	Generar reporte de asistencia	Cliente Android	Crear un interfaz de usuario para que un Administrador o Profesor puedan generar un reporte de asistencia para una materia y paralelo de los estudiantes presentes en las diferentes actividades académicas y enviarlo a un e-mail.
			Implementar el código necesario para que el cliente envíe al servidor la petición de generación de un Reporte de Asistencia para un paralelo y enviarlo vía mail.
		Servidor	Implementar el código necesario para que el servidor pueda generar el reporte de asistencia para un paralelo y mediante la conexión a un servidor SMTP pueda enviarlo vía mail.

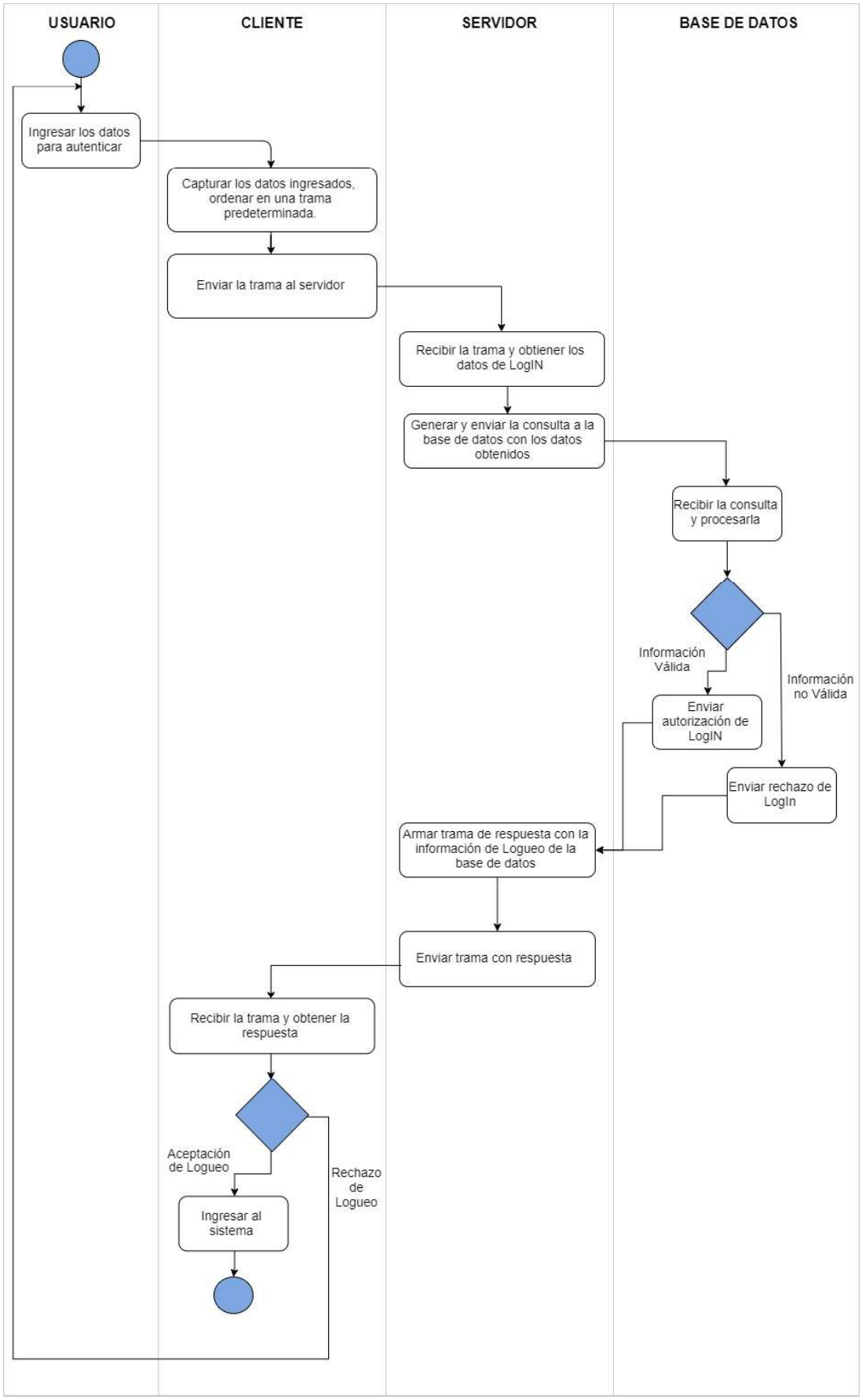

Figura 2.3 Diagrama de actividades LogIn

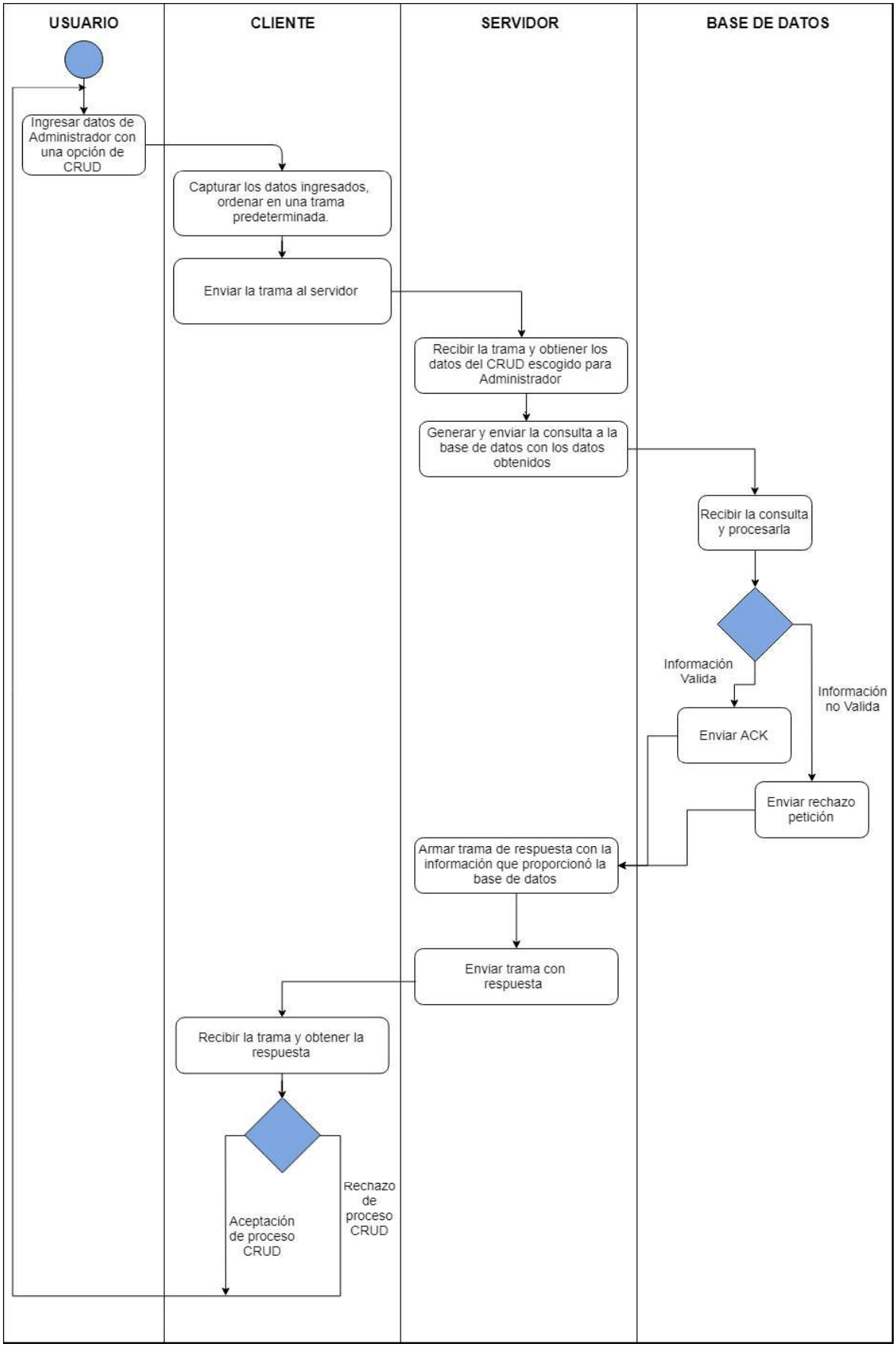

Figura 2.4 Diagrama de actividades para usuario Admin

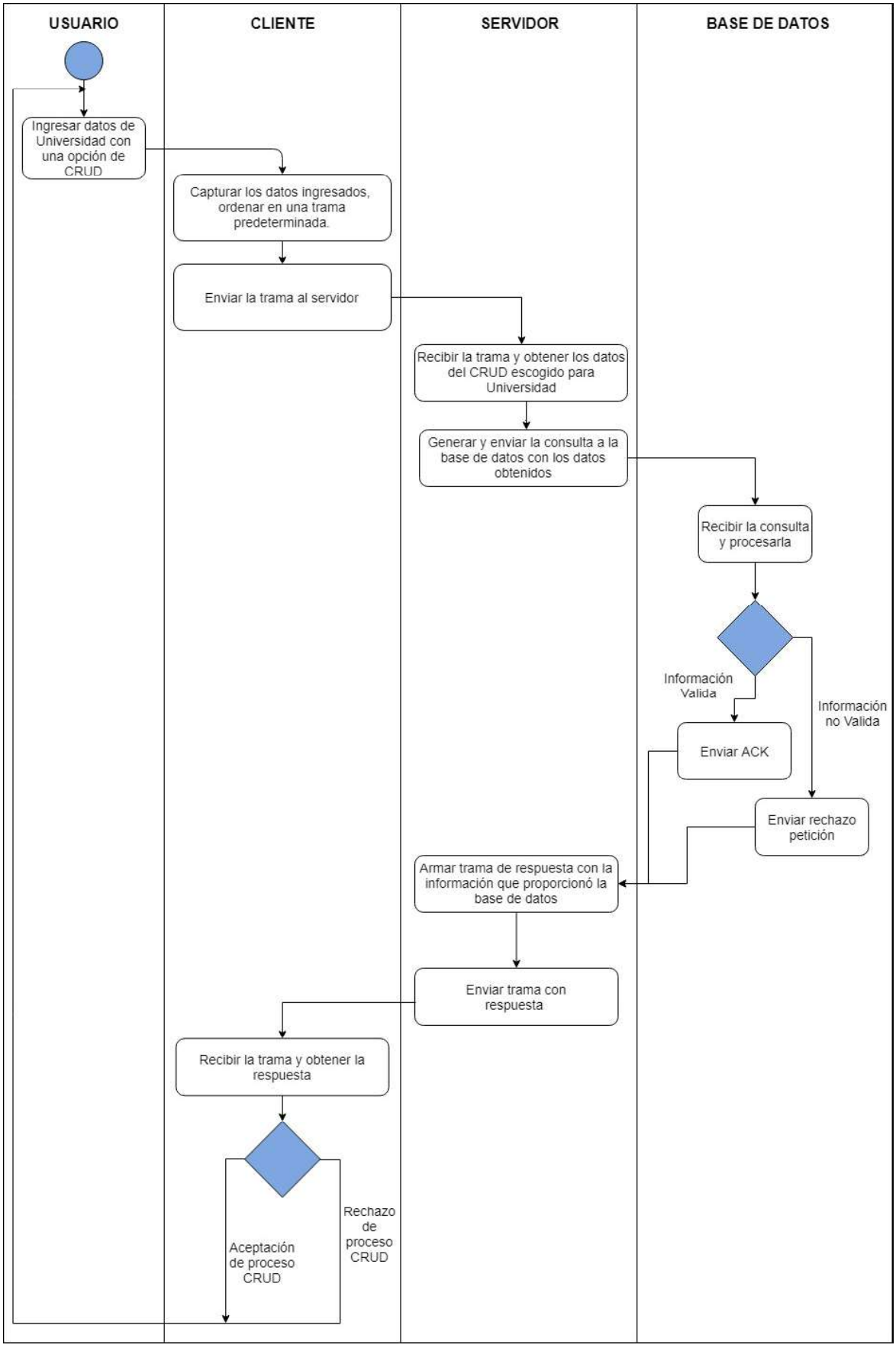

Figura 2.5 Diagrama Actividades CRUD Universidad

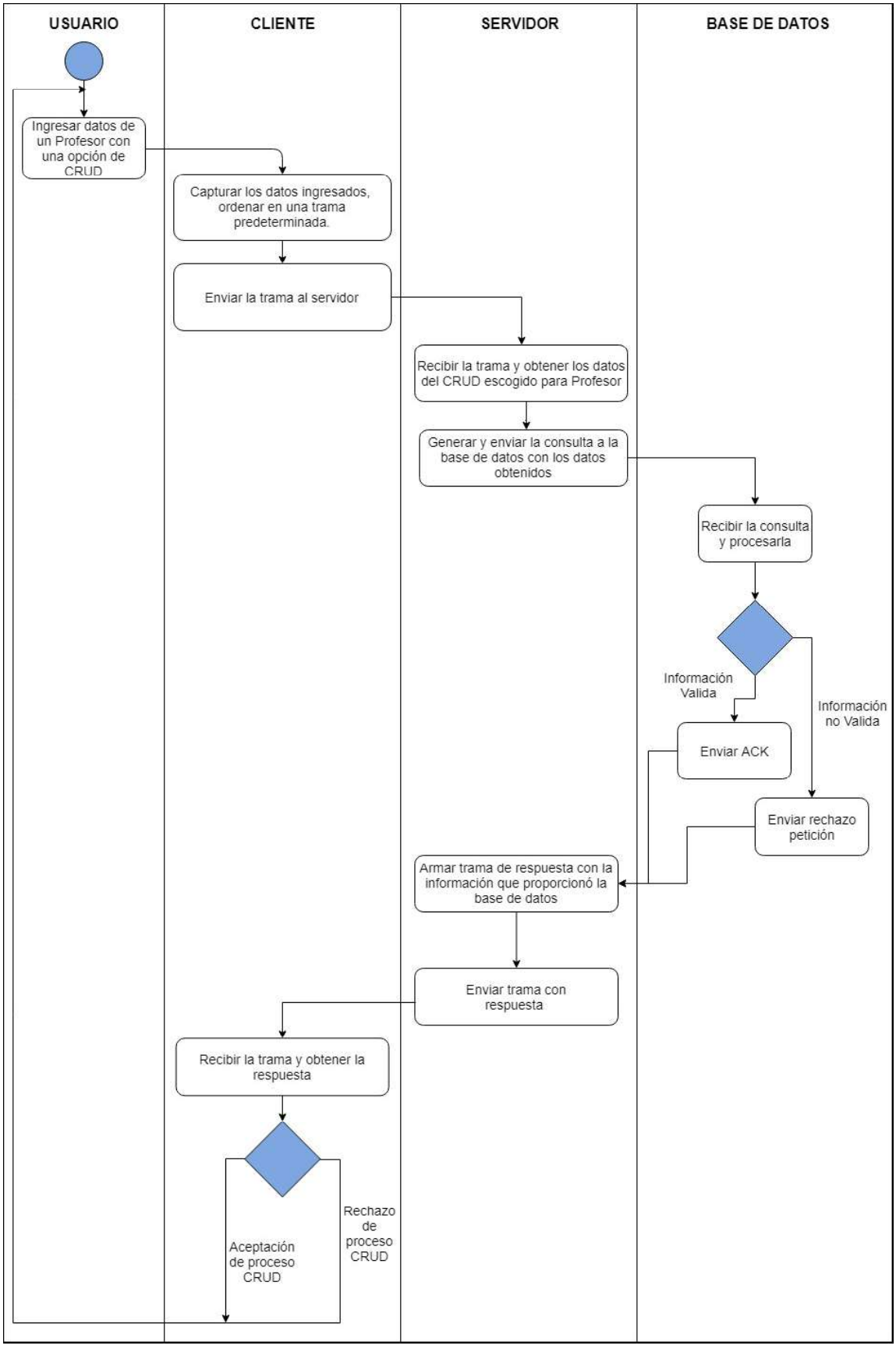

Figura 2.6 Diagrama Actividades para CRUD usuario Profesor

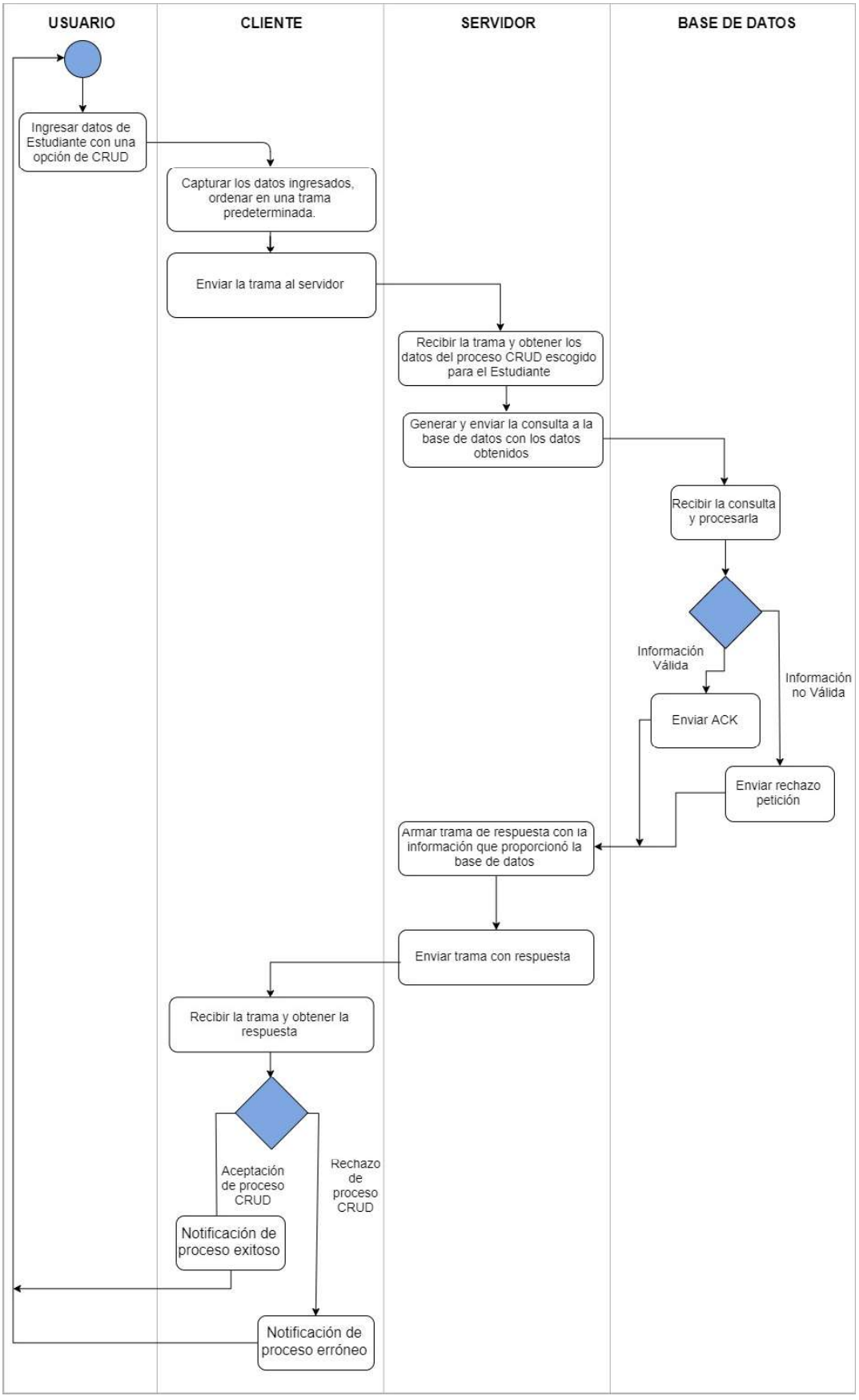

Figura 2.7 Diagrama de actividades CRUD Estudiante

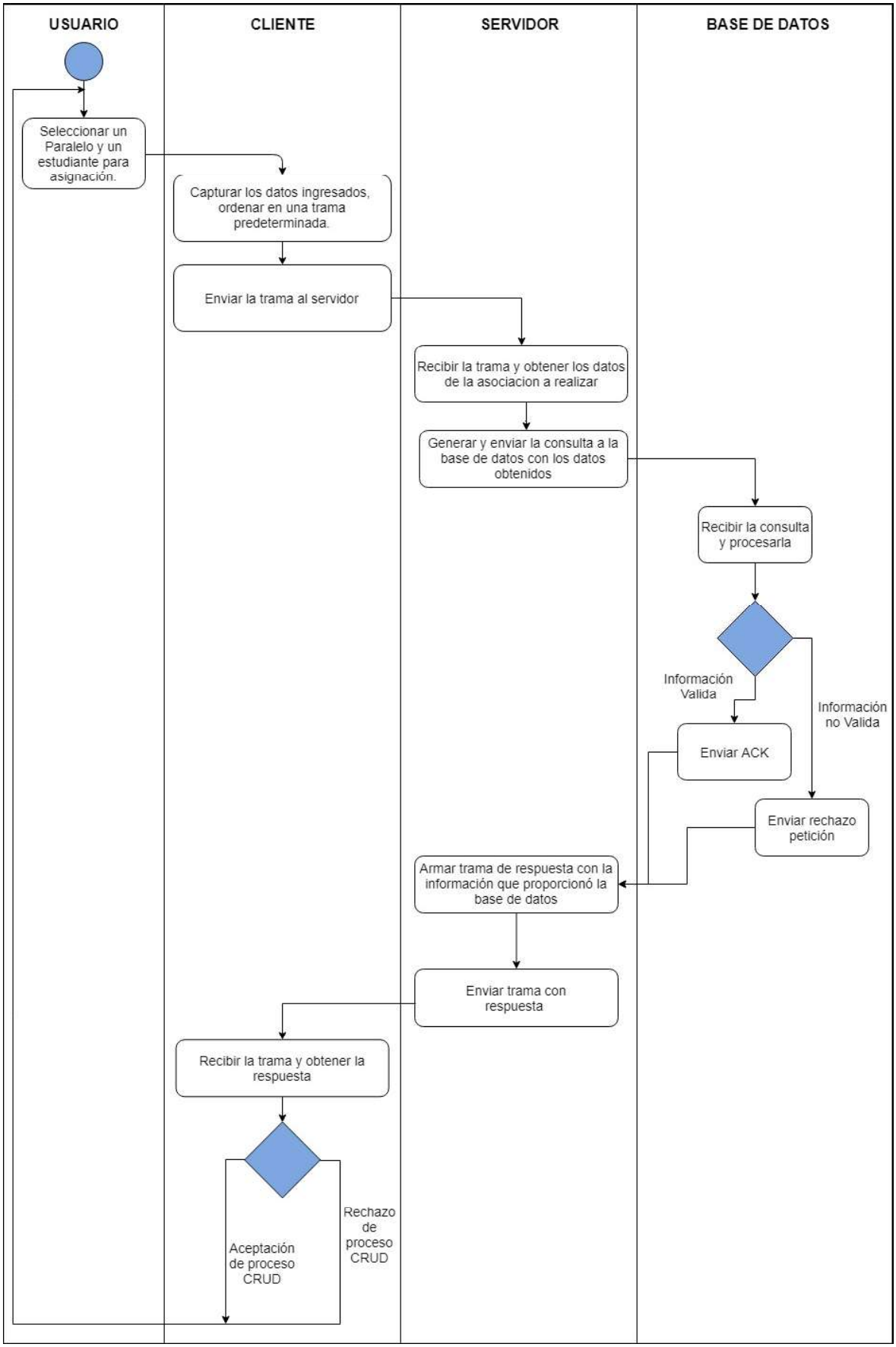

Figura 2.8 Diagrama de actividades Asignación de Estudiante a Paralelo

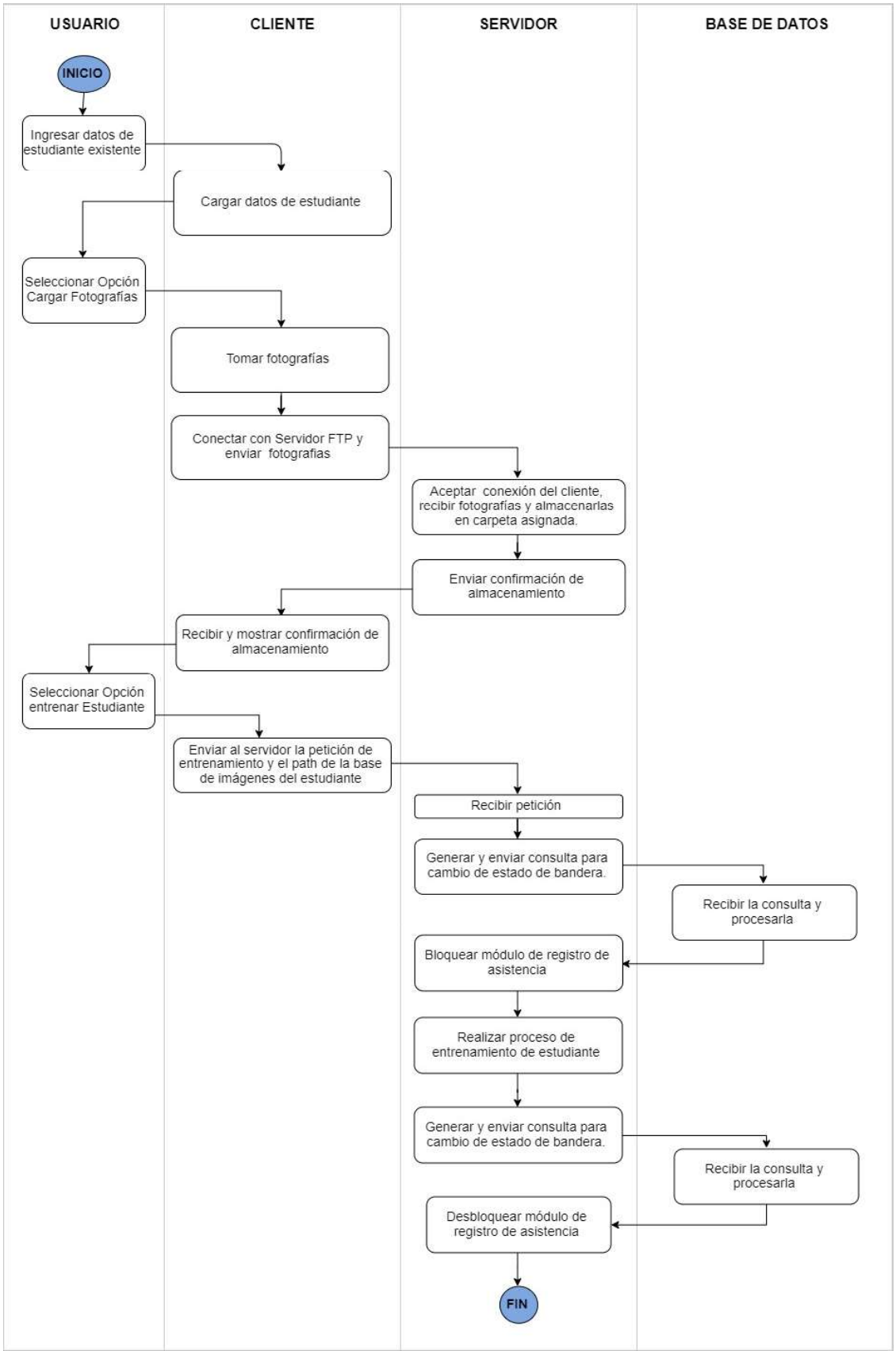

Figura 2.9 Diagrama de actividades de entramiento del sistema y adición de fotografías a la base de imágenes

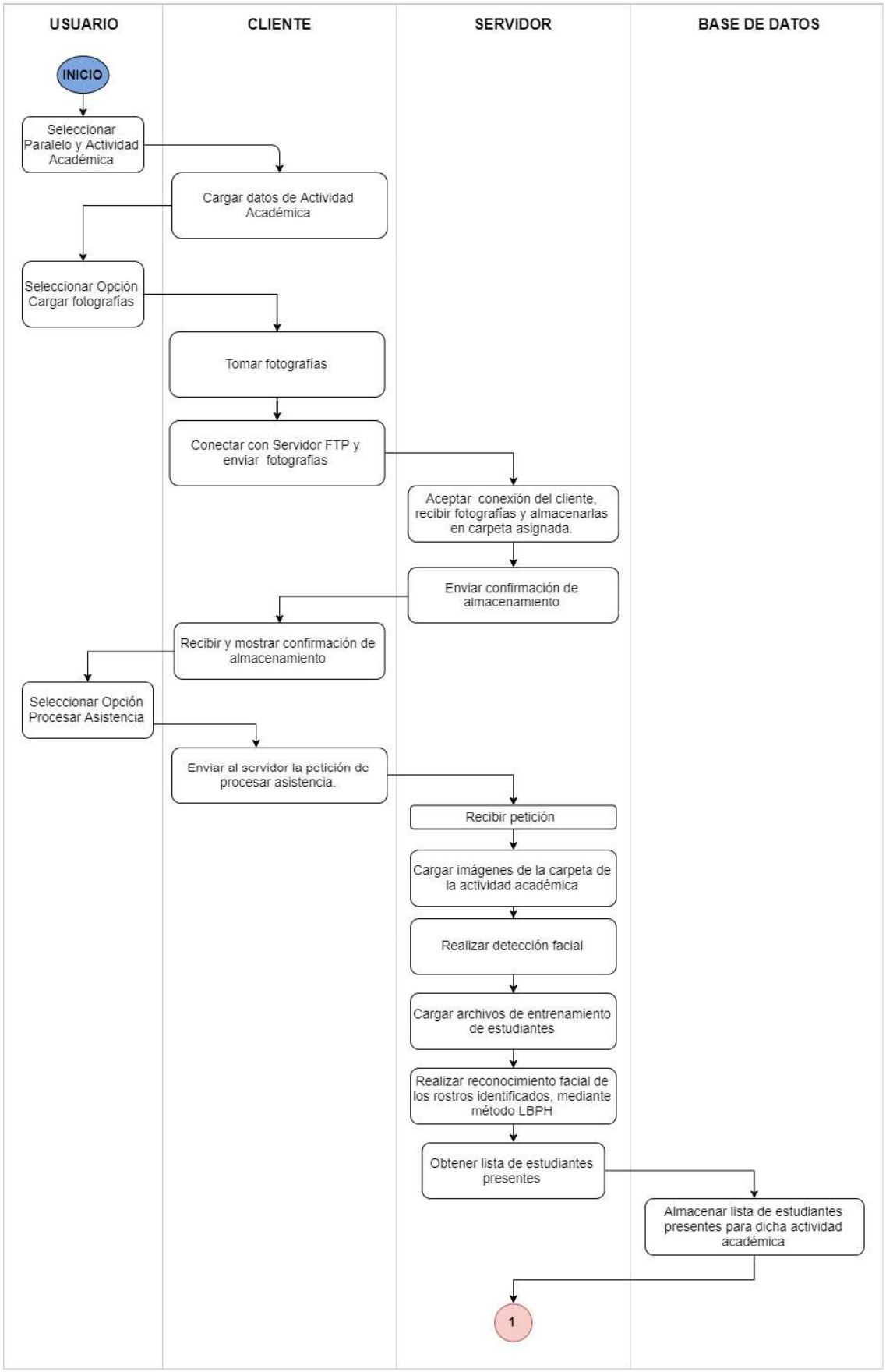

Figura 2.10 Diagrama de actividades Tomar Asistencia Parte I

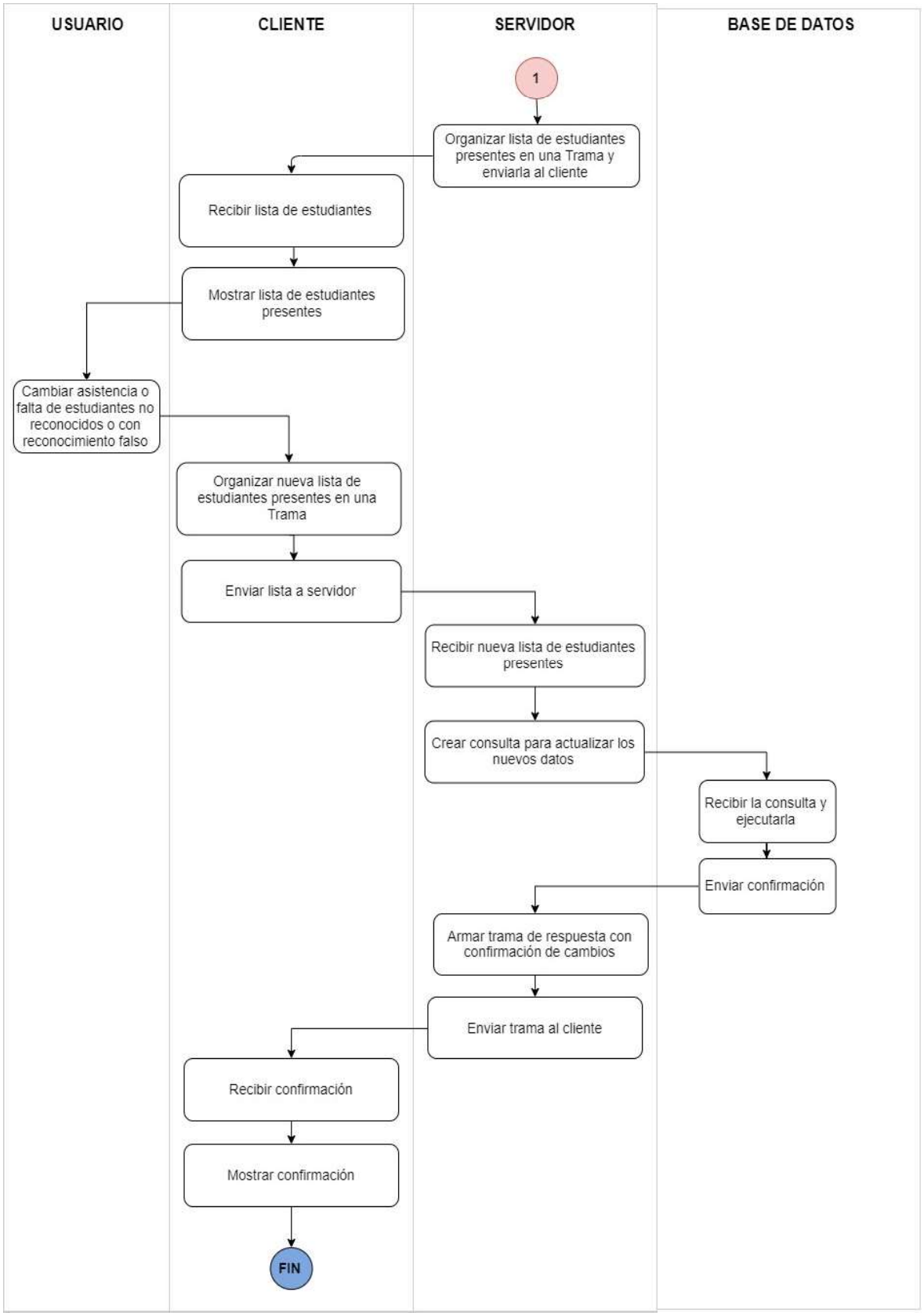

Figura 2.11 Diagrama de actividades Tomar Asistencia Parte II

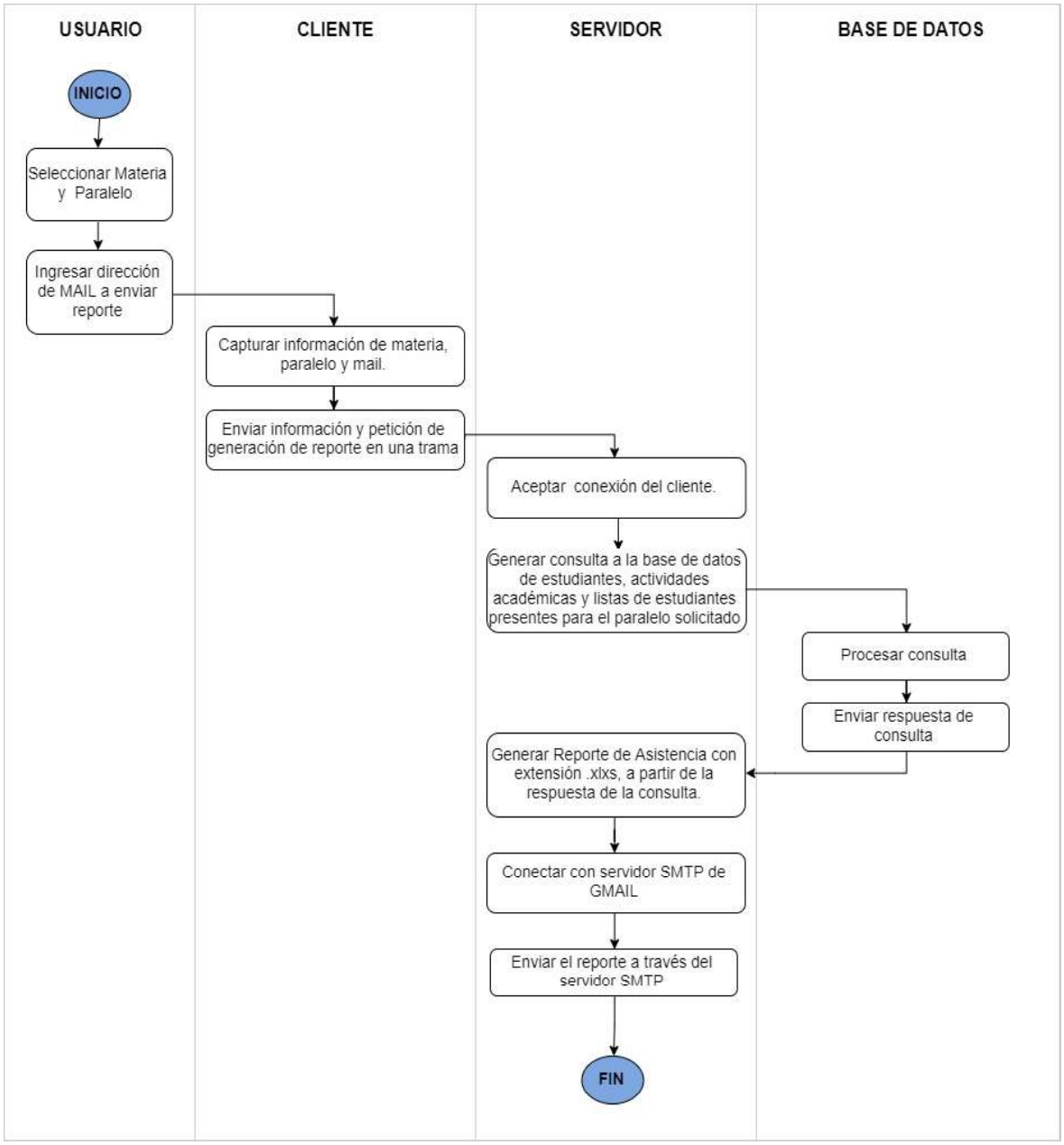

Figura 2.12 Diagrama de actividades Generar Reporte

2.1.9 DISEÑO DE LA BASE DE DATOS

El sistema prototipo a desarrollar constará con una base de datos en MySQL, la cual permitirá almacenar toda la información que se requiera para su correcto funcionamiento.

2.1.9.1 Diagrama entidad relación

En la Figura 2.13 y 2.14 se muestra el diagrama entidad relación, la misma que consta de las siguientes entidades: Universidad, Facultad, Carrera, Estudiante, Materia, Paralelo, Actividad Académica, Lista de Estudiantes Presentes, Profesor y Administrador.

En la Tabla 2.18 se muestran las entidades con sus respectivos atributos:

Tabla 2.18 Entidades y atributos

N°	Entidad	Atributo	Descripción
1	Universidad	idUniversidad	Índice de la entidad.
		nombreUniversidad	Nombre descriptivo de la entidad.
2	Facultad	idFacultad	Índice de la entidad.
		nombreFacultad	Nombre descriptivo de la entidad.
3	Carrera	idCarrera	Índice de la entidad.
		nombreCarrera	Nombre descriptivo de la entidad.
4	Estudiante	idEstudiante	Índice de la entidad.
		cedula	Número de identificación del estudiante
		nombre	Nombre del estudiante
		apellido	Apellido del Estudiante
		numeroUnico	Número único asignado por la universidad
		pathFotografias	Dirección de la ubicación de las fotografías de entrenamiento del estudiante
5	Materia	idMateria	Índice de la entidad.
		nombreMateria	Nombre descriptivo de la entidad.
6	Paralelo	idParalelo	Índice de la entidad.
		nombreParalelo	Nombre descriptivo de la entidad.
7	Actividad Académica	idActividadAcademica	Índice de la entidad.
		fecha	Fecha en la que se desarrolla la actividad académica.
8	Lista de Estudiantes Presentes	idListaEstudiantesPresentes	Índice de la entidad.
9	Profesor	idProfesor	Índice de la entidad.
		cédula	Número de identificación del profesor, será su nombre de usuario.
		nombre	Nombre del profesor
		apellido	Apellido del profesor
		password	Clave para ingresar al sistema
10	Administrador	idAdministrador	Índice de la entidad.

		cédula	Número de identificación del profesor, será su nombre de usuario.
		nombre	Nombre del administrador
		apellido	Apellido del administrador
		password	Clave para ingresar al sistema

De la Figura 2.13 podemos observar que las cardinalidades se forman de la siguiente manera:

- Una Universidad posee varias Facultades; y una Facultad solo puede estar dentro de una Universidad, por lo que la cardinalidad es Varios a 1.
- Una Universidad posee varios Profesores trabajando en ella; y un Profesor solo puede trabajar en una Universidad, por lo que la cardinalidad es Varios a 1.
- Una Facultad posee varias Carreras; y una Carrera solo puede estar dentro de una Facultad, por lo que la cardinalidad es Varios a 1.
- Una Carrera posee varias Materias; y una Materia solo puede pertenecer a una Carrera, por lo que la cardinalidad es Varios a 1.
- Una Materia posee varios Paralelos; y un Paralelo solo puede pertenecer a una Materia, por lo que la cardinalidad es Varios a 1.
- Una Carrera posee varios Estudiantes; y un Estudiante solo puede pertenecer a una Carrera, por lo que la cardinalidad es Varios a 1.
- Un Paralelo posee varias Actividades Académicas; y una Actividad Académica solo puede pertenecer a un Paralelo, por lo que la cardinalidad es Varios a 1.
- Una Actividad Académica posee varias Listas de Estudiantes Presentes; y una Lista de Estudiantes Presentes solo puede pertenecer a una Actividad Académica, por lo que la cardinalidad es Varios a 1.
- Un Estudiante puede pertenecer en varias Listas de Estudiantes Presentes; y una Lista de Estudiantes Presentes solo puede pertenecer a un Estudiante, por lo que la cardinalidad es Varios a 1.
- Un Estudiante puede pertenecer a varios Paralelos; y un Paralelo solo puede tener varios Estudiantes, por lo que la cardinalidad es Varios a Varios. En este caso en el diagrama relacional se deberá crear una Tabla que permita eliminar este tipo de relación.

En la Figura 2.14 se muestra que la entidad `Administrador` no posee relación alguna con otras entidades, esto se debe a que es un usuario que posee un roll de control sobre el sistema, por lo que podría ser cualquier persona que pertenezca o no a la universidad.

2.1.9.2 Diagrama Relacional

Una vez identificadas las entidades que intervendrán en el prototipo se procede a ver cuál es la relación entre cada entidad y que características tendrán sus atributos. Para esto en la Figura 2.15 se muestra el diagrama relacional de la base de datos.

El diagrama relacional muestra que cada entidad se ha traducido en una tabla con una clave primaria o Primary Key, la cual la identifica dentro de una relación de entidades, además en cada relación 1: N, la entidad con cardinalidad N usará como atributo la clave primaria de su contraparte, a esto se lo llamada Foreing Key. Para las entidades todos los índices son auto numerados.

La relación entre `Estudiantes` y `ActividadesAcademicas` tiene una cardinalidad de N: N, para esto se ha creado la tabla intermedia `ListasEstudiantesPresentes`, con esto se ha eliminado dicha relación.

La base de datos tendrá en sus tablas, atributos que no permitirán ser `Null` y tendrán la opción de `UNIQUE`, la cual significa que el valor que tome no puede repetirse en otras instancias. Además, para el registro de número de cédulas se utilizará un atributo `varchar`, ya que estos valores podrían empezar con cero y se perdería el valor original.

La tabla `Administrador` no posee relación alguna con otra, ya que su roll es netamente de supervisión y control del sistema. También se creó una tabla para manejar las banderas de algunos servicios del prototipo, esta tabla se llama `banderasEntrenamientos` y restringe el acceso a la toma de asistencia hasta que no haya finalizado el proceso de entrenamiento.

En la tabla `actividadesAcademicas` el atributo de `fecha` posee el formato AA:MM:DD HH.MM.seg (Años: Mes: día Hora. Minuto y seg).

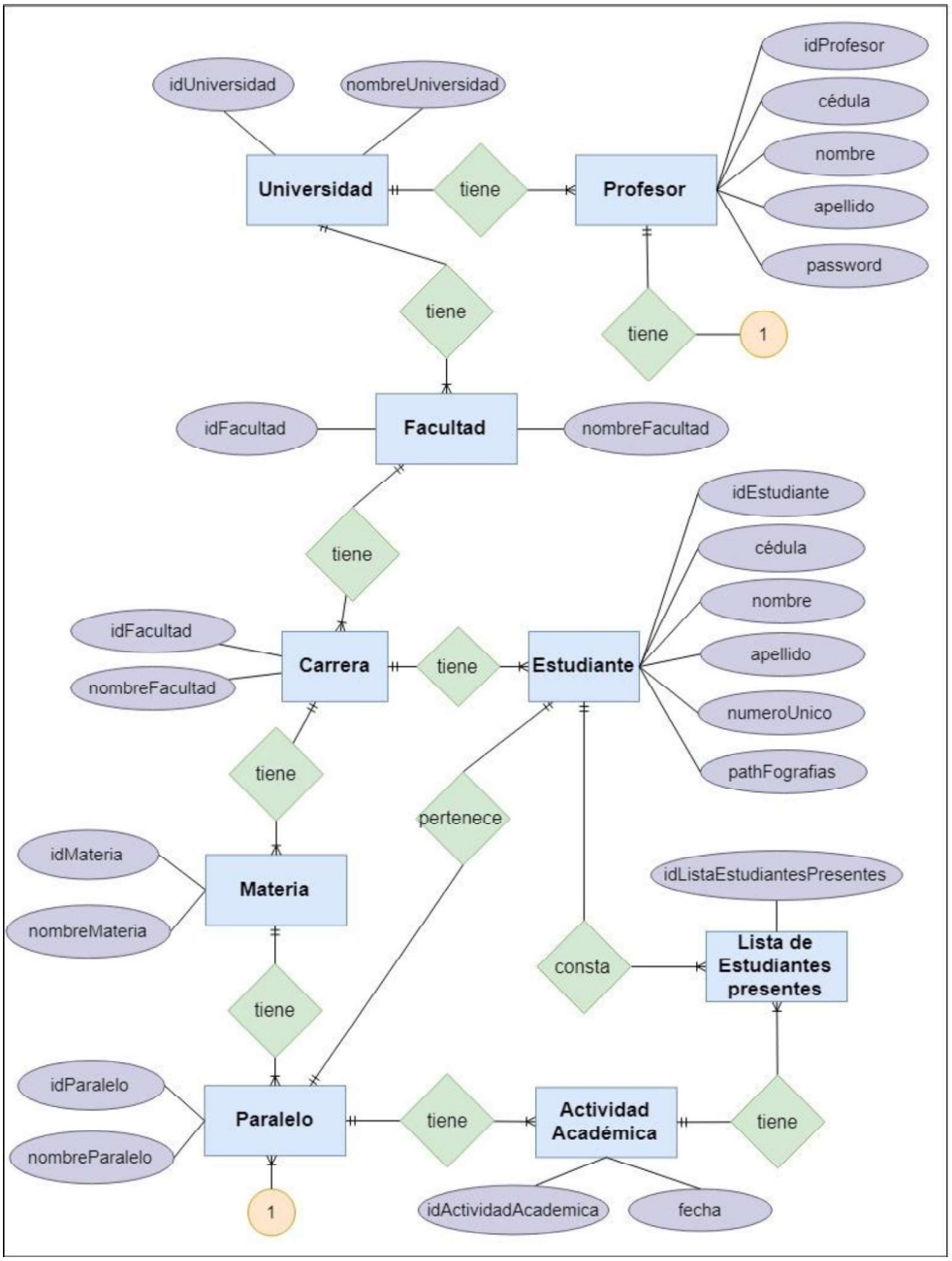

Figura 2.13 Diagrama Entidad- relación (Parte I)

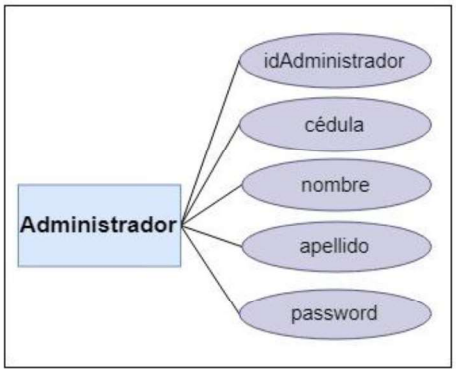

Figura 2.14 Diagrama Entidad-Relación (Parte II)

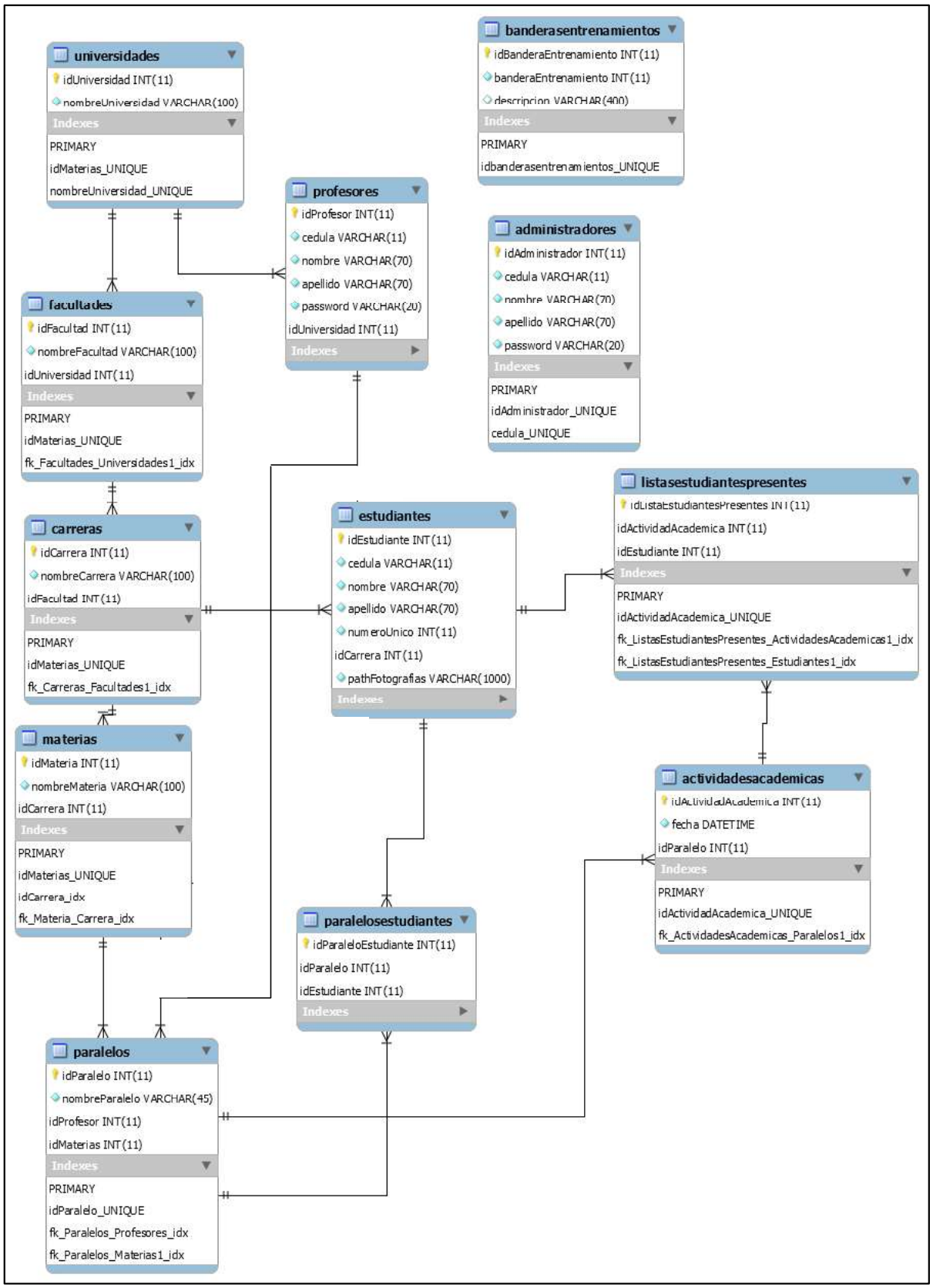

Figura 2.15 Diagrama Relacional

2.1.10 DIAGRAMÁS DE CLASES PARA EL SISTEMA PROTOTIPO

A partir de los diagramas de entidad en el diseño de la base de datos se ha podido generar los diagramas de clases para el prototipo tanto en el cliente Android, como en el servidor.

2.1.10.1 Diagramas para el cliente

Para el diseño del cliente se han generado las clases que se muestran en la Figura 2.16, 2.17 y 2.18 atreves diagramas UML. Para cada figura, las clases se describirán brevemente.

En la Figura 2.16 se muestra la parte 1 del diagrama, ahí se pueden apreciar las siguientes clases:

- `MainActivity`: es la clase principal del cliente, será la encargada de procesar la información para el inicio de sesión de un usuario, deberá tener una interfaz asociada. También deberá ser el punto de partida para la conexión con el servidor, por lo que tendrá los datos de IP y puerto del mismo.
- `MenuPrincipal`: es la encargada de permitir que el usuario pueda escoger una acción a desempeñar en el sistema. Tendrá las opciones a las que se puede acceder según el tipo de usuario logueado.
- `CabeceraPerfilUsuarioConexion`: es una clase que instanciará un objeto, el cual viajará de clase en clase, llevando los datos para la conexión al servidor y el tipo de usuario logueado en el sistema.
- `ProtocoloDatos`: para facilitar la compatibilidad de mensajes entre cliente – servidor; y evitar enviar objetos serializados mediante sockets, se ha decidido crear una clase genérica, la cual será compatible con todos los objetos tanto del cliente como del servidor. Esta clase tendrá un identificador y 10 variables Strings, estas variables mediante un cast se transformarán en los tipos datos a utilizar en la aplicación. Para enviar información al servidor, al objeto Protocolo se lo organizará en una Trama, es decir en un arreglo de Strings concatenados.
- `ConexionSocket`: permitirá crear un objeto para realizar la conexión al servidor mediante sockets. Los datos del servidor los obtendrá de la clase `CabeceraPerfilUsuarioConexion`, y los datos a enviar los obtendrá al concatenar toda la información del objeto `ProtocoloDatos`.

En la Figura 2.17 se muestra la parte 2 del diagrama, ahí se pueden apreciar las siguientes clases:

- `CRUDUniversidad`: Encargado de realizar el proceso CRUD de la entidad Universidad.
- `CRUDFacultad`: Encargado de realizar el proceso CRUD de la entidad Facultad.
- `CRUDCarrera`: Encargado de realizar el proceso CRUD de la entidad Carrera.
- `CRUDMateria`: Encargado de realizar el proceso CRUD de la entidad Materia.
- `CRUDParalelo`: Encargado de realizar el proceso CRUD de la entidad Paralelo.
- `CRUDEstudiante`: Encargado de realizar el proceso CRUD de la entidad Estudiante, el cual tendrá datos de su información básica y su path de fotografías en la base de imágenes.
- `CRUDAdministrador`: Encargado de realizar el proceso CRUD para el usuario Administrador, el cual tendrá como objetivo administrar todos los aspectos del sistema.

En la Figura 2.18 se muestra la parte 3 del diagrama, ahí se pueden apreciar las siguientes clases:

- `CRUDProfesor`: Encargado de realizar el proceso CRUD para el usuario Profesor, el cual tendrá como objetivo registrar la asistencia de estudiantes para las diferentes materias en las que ha sido asignado.
- `CRUDActividadAcademica`: Encargado de realizar el proceso CRUD de la entidad `ActividadAcademica`.
- `CRUDEstudianteParalelo`: clase encargada de enlazar estudiantes con paralelos. Cada paralelo estará enlazado a una materia y poseerá varias actividades académicas con las cuales se realizará el registro de asistencias.
- `TomarAsistencia`: clase encargada de registrar la asistencia de estudiantes para una actividad académica. Como requisito previo se deberá tener una materia, un profesor, un paralelo con estudiantes asignados y una actividad académica.
- `GenerarReporteAsistencia`: clase encargada de generar los reportes de los registros de asistencia para los diferentes paralelos enlazados a materias. Generará reportes en Excel de todas las actividades académicas pertenecientes a un paralelo y las enviará vía mail.
- `AdapterListaEstudiante`: clase que permitirá agregar listas de ítems de estudiantes a las clases `CRUDEstudianteParalelo` y `TomarAsistencia`.
- `ItemEstudianteActivity`: Clase que permite agregar a los Items de ListViews las instancias de la entidad Estudiante.

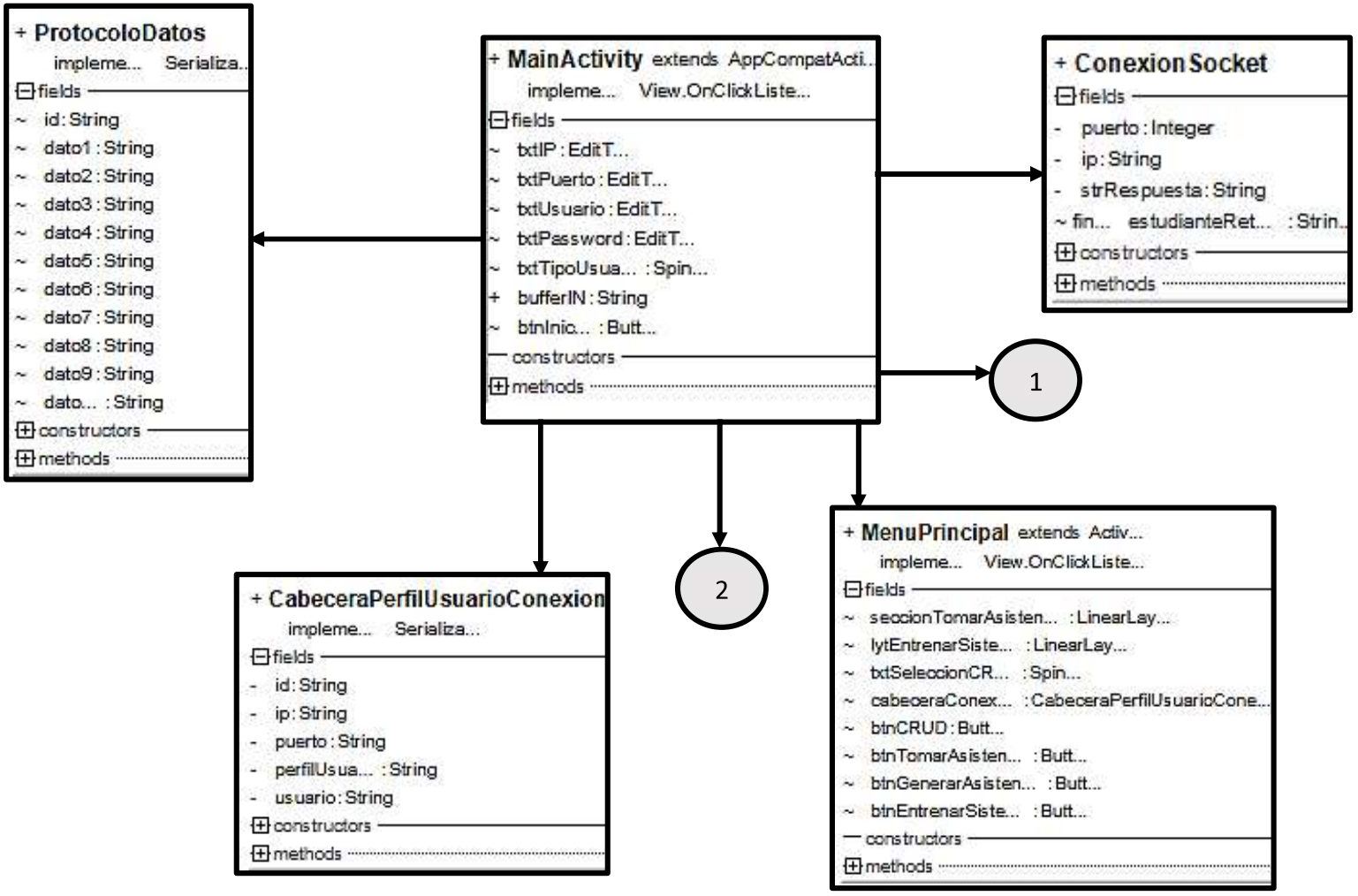

Figura 2.16 Diagrama de Clases Del Cliente Parte 1

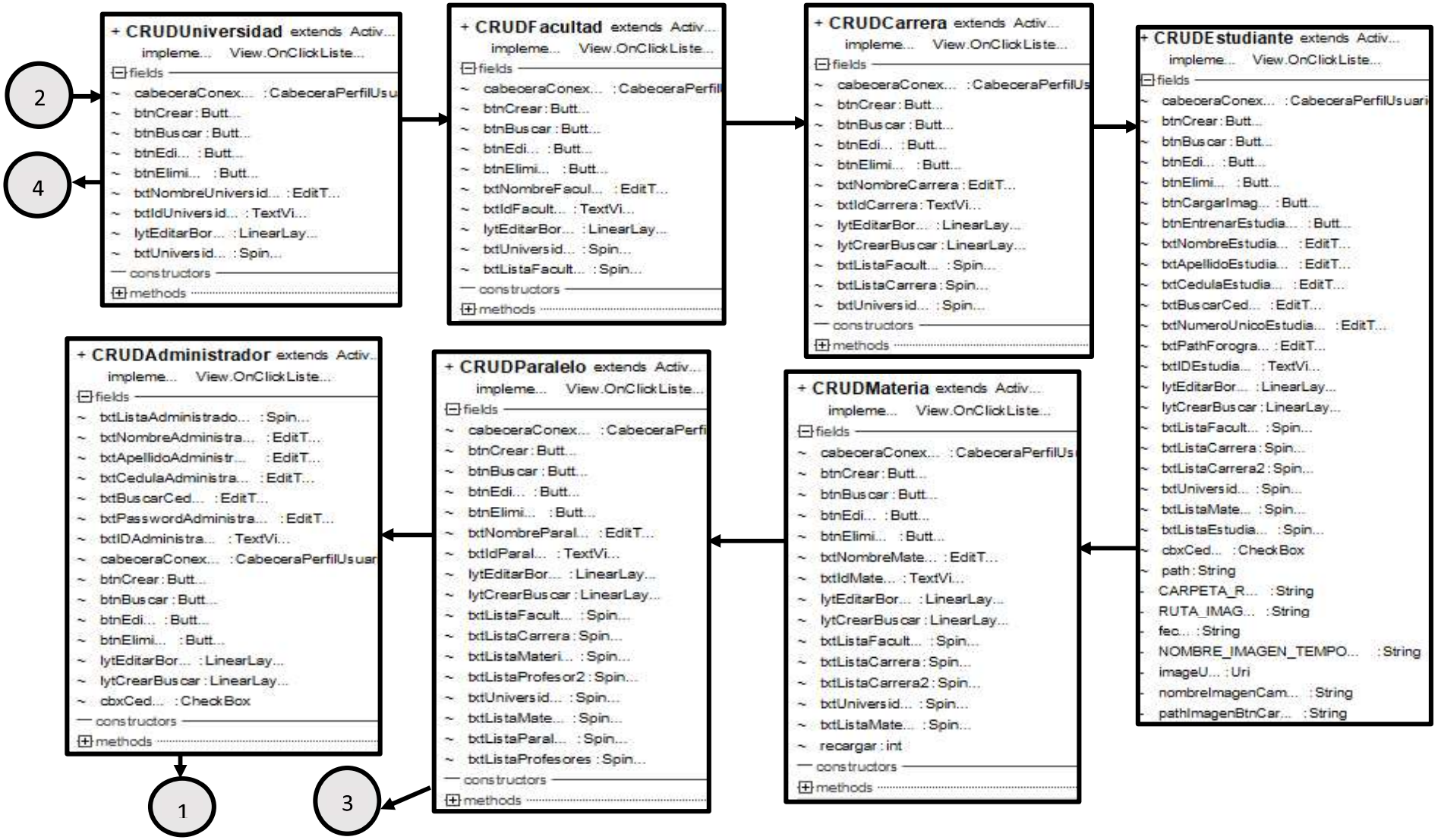

Figura 2.17 Diagrama de Clases Del Cliente Parte 2

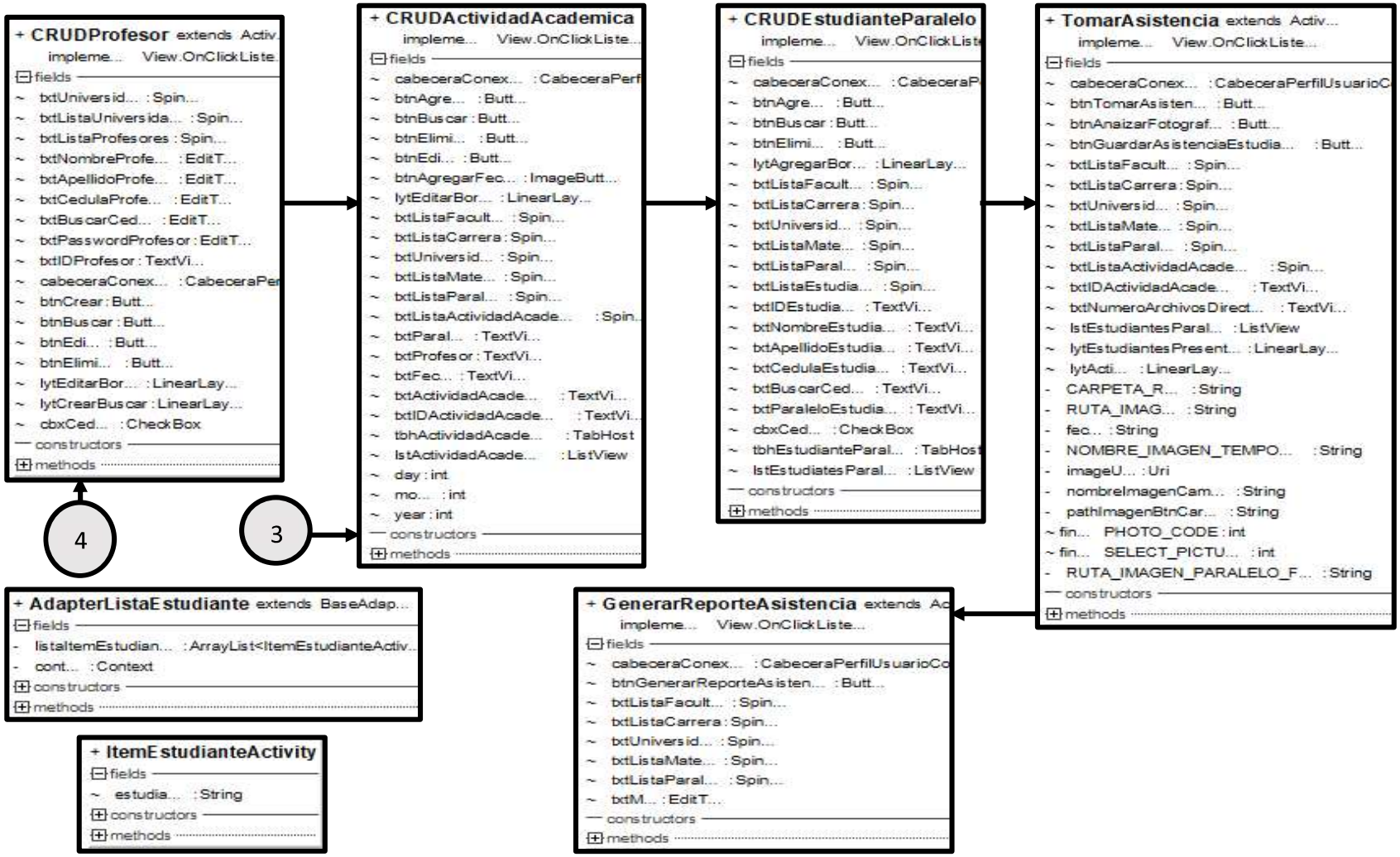

Figura 2.18 Diagrama de Clases Del Cliente Parte 3

2.1.10.2 Diagramas para el servidor

Para el diseño del servidor se han generado las clases que se muestran en la Figura 2.19 diagramas UML, las clases se describirán brevemente.

Los nombres de las clases para el caso del modelamiento con el ORM Peewee deben ser iguales a las que constan en el diagrama relacional, en la Figura 2.9 se pueden apreciar las clases del servidor:

- `Protocolo`: clase encargada de traducir los mensajes que se reciben del cliente y procesarlos según su identificador. Coincide en su constructor y sus atributos con la clase `ProtocolosDatos` del cliente, con esto se ha creado una compatibilidad de objetos en el sistema y se facilitará la comunicación. Tanto el servidor como el cliente intercambiarán estos objetos en forma de arreglos de Strings concatenados llamados Tramas. En la sección de implementación se mostrará cómo se manejará dicha compatibilidad.
- `DBModel`: clase que poseerá una instancia de la conexión a la base de datos y de la cual heredarán todas las clases que intervengan en el modelamiento con el ORM Peewee.
- `administradores`: clase modelada a partir de la tabla `administradores` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `universidades`: clase modelada a partir de la tabla `universidades` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `facultades`: clase modelada a partir de la tabla `facultades` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `carreras`: clase modelada a partir de la tabla `carreras` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `materias`: clase modelada a partir de la tabla `materias` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `paralelos`: clase modelada a partir de la tabla `paralelos` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `actividadesacademicas`: clase modelada a partir de la tabla `actividadesacademicas` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `profesores`: clase modelada a partir de la tabla `profesores` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.

- `estudiantes`: clase modelada a partir de la tabla `estudiantes` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `paraleloestudiantes`: clase modelada a partir de la tabla `paraleloestudiantes` de la base de datos, permitirá acceder a la misma mediante el uso de objetos. Su función principal es llevar el registro de que estudiantes pertenecen a un paralelo.
- `listasestudiantespresentes`: clase modelada a partir de la tabla `listasestudiantespresentes` de la base de datos, permitirá acceder a la misma mediante el uso de objetos.
- `banderasentrenamientos`: clase modelada a partir de la tabla `banderasentrenamientos` de la base de datos, permitirá acceder a la misma mediante el uso de objetos. Su función será la de impedir mediante banderas el uso de la Toma de Asistencia cuando exista algún entrenamiento del sistema.

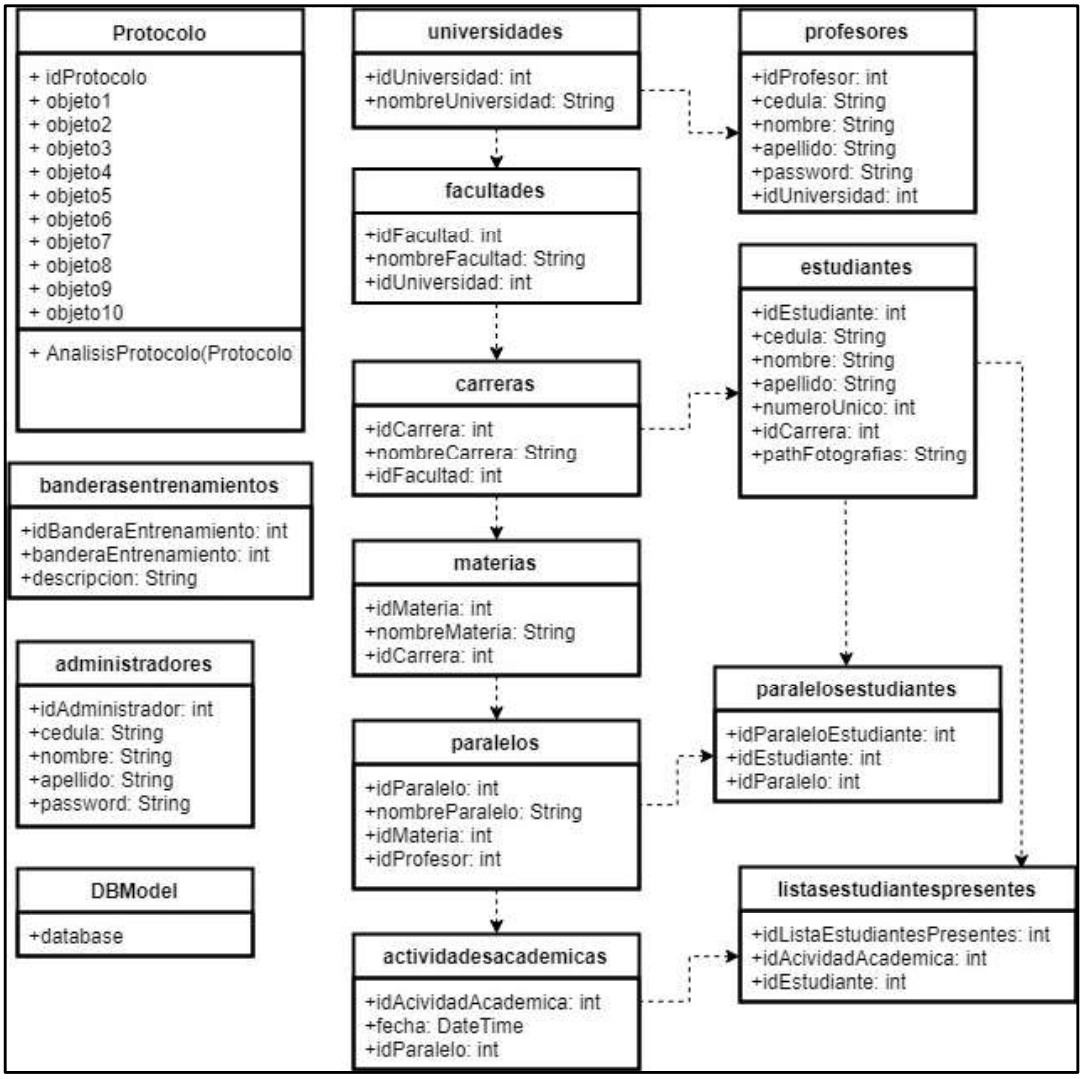

Figura 2.19 Diagrama de clases del servidor

2.1.11 DIAGRAMA DE SECUENCIA

En base al diseño de la base de datos, de las clases tanto del servidor como del cliente, se ha podido modelar el diagrama de secuencia para todo el prototipo, el cual se lo presenta en la Figura 2.20.

En la Figura 2.20 se puede apreciar que el cliente siempre será el que realice peticiones al servidor, y este le responderá según corresponda. El servidor también será el encargado de manipular la base de datos, en ningún caso el cliente se conectará directamente con la con la misma.

Las respuestas del servidor pueden ser confirmaciones de modificación de datos en la base o respuestas a las diferentes consultas que pudo haber solicitado el cliente a través del usuario.

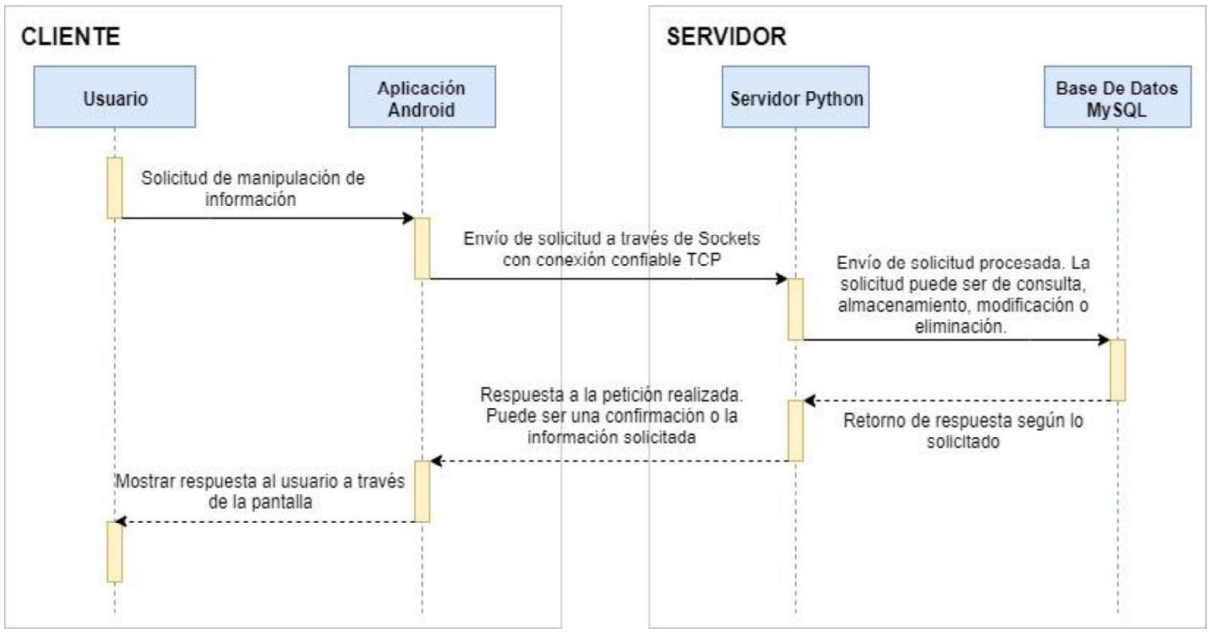

Figura 2.20 Diagrama de secuencia del sistema Prototipo

2.1.12 SKETCHS PRINCIPALES

Como parte fundamental del diseño del prototipo están los Sketchs, que son diagramas genéricos de las interfaces de usuario, en los cuales se detallará que componentes intervendrán y como será su ubicación. En esta sección se presentarán los más importante para el sistema.

En la Figura 2.21 se muestra el diagrama para la interfaz de autenticación, con la cual se ingresará al sistema desde el cliente. Para el LogIn se necesitará información de la IP y puerto del servidor, nombre de usuario, contraseña y el tipo de usuario que se va a loguear.

En la Figura 2.22 se muestra el diagrama para el CRUD de la entidad estudiante, en el cual se incluye el proceso de adición de fotografías a la base de imágenes y el entrenamiento del estudiante en el sistema. En el análisis del proceso de adición de fotografías se buscó tener tiempos de carga de imágenes muy cortos, ya que las fotografías de estudiantes se tomarán desde el dispositivo y se las enviará al servidor para ser almacenadas. La forma de almacenamiento que se escogió es en carpetas, una por estudiante. Cada carpeta se identificará con el número de cédula del estudiante y su Path se almacenará como campo en la tabla estudiante.

Para el almacenamiento de las fotografías en carpetas, se ha decidido crear un servidor FTP en Windows 10, ya que actualmente el proceso de creación del mismo es muy sencillo [37]. El cliente Android se conectará al servidor FTP mediante las librerías `org.apache.commons.net.ftp.FTPClient`, `org.apache.commons.net.ftp.FTPFile` y `org.jibble.simpleftp.SimpleFTP`. Luego de enviar las fotografías, el cliente notificará al servidor que realice el proceso de entrenamiento accediendo al Path del estudiante que corresponda. Gracias a esta modificación en el diseño se logrará optimizar tiempos de envío, no se perderán datos de las imágenes ya que no necesitaremos tablas de deserialización de objetos Java en el servidor Python. Cabe recalcar que el FTP se lo debe ubicar en el mismo computador que el servidor Python.

En la Figura 2.23 se muestra el diagrama para la interfaz llamada Tomar Asistencia, aquí cada usuario seleccionará un paralelo y una actividad académica según corresponda, posteriormente podrá agregar fotografías de los estudiantes presentes a una carpeta que pertenece a cada actividad académica mediante el servidor FTP (es el mismo proceso que se explica en el párrafo anterior). Luego se procesará la asistencia, aquí el servidor primero realizará la detección de rostros de las imágenes y posteriormente realizará el reconocimiento facial de cada estudiante presente, al identificarlos creará una lista de asistencia en la base de datos y la retornará al cliente para que este pueda modificarla manualmente y finalmente guarde los cambios

En la Figura 2.24 se muestra el diagrama para la interfaz de Generar Reporte de Asistencia, en esta interfaz al seleccionar un paralelo asociado a una materia e ingresando una dirección de mail, se realizará una petición al servidor para que genere el reporte de

asistencia de dicho paralelo en extensión .xlsx, y lo envíe mediante un servidor SMTP a una dirección de mail. El servidor SMTP a utilizar será uno proporcionado por GMAIL [38].

Figura 2.21 Sketch LogIn

Figura 2.22 Sketch Módulo Estudiante

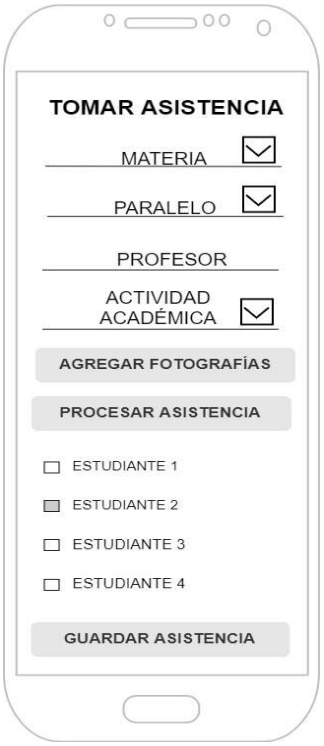

Figura 2.23 Interfaz Tomar Asistencia

Figura 2.24 Interfaz Reporte de Asistencia

2.2 IMPLEMENTACIÓN DEL PROTOTIPO

En esta sección se implementará el sistema prototipo en base al diseño de la sección anterior. El proceso de codificación se ha organizado en base a Sprints, por lo que ya existe un orden predeterminado para crear los diferentes módulos del sistema.

En esta sección se mostrarán las partes más relevantes del proyecto, el código en su totalidad tanto del servidor como del cliente se los mostrará en el ANEXO D y ANEXO E respectivamente.

2.2.1 SPRINT 1

En el Sprint 1 se implementó la base de datos.

La base de datos fue creada en MySQL versión 8.0.13 basándose en el diagrama relacional de la sección anterior. El entorno gráfico Workbench v8.0.13 permite crear la base de datos a partir del diagrama relacional por lo que se utilizó esta función de conversión para agilizar el proceso de relación.

El diagrama a utilizar se muestra en la Figura 2.15 y el proceso a seguir en el IDE es el siguiente:

- Primero: Crear Modelo de base de datos MySQL.
- Segundo: Crear un archivo EER Diagram
- Tercero: Crear el diagrama relacional, es importante considerar las llaves primarias y secundarias, además de la cardinalidad.
- Cuarto: En el menú “Database” escoger la opción “Forward Engenieer” para generar el Script de la base de datos.
- Quinto: Escoger que Tablas se desea crear y que propiedades incluir, para este proyecto se escogió crear Tablas con llaves primarias y secundarias. En este punto se generará el Script de la base de datos.
- Séptimo: Después de haber generado el Script, este es ejecutado automáticamente y creará las tablas de la base con sus respectivas relaciones.

En el Segmento de Código 2.1 se muestra la creación de la base de datos llamada `tesisreconocimientofacialepn`. Para esto se utilizó el comando “`Create SCHEMA IF NOT EXISTS`”, se definió como codificación de caracteres `UTF8mb4` y el collage de caracteres `utf8mb4_ai_ci`.

```
-- -----------------------------------------------------
-- Schema tesisreconocimientofacialepn
-- #Creacion de la base de datos  tesisreconocimientofacialepn
-- -----------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `tesisreconocimientofacialepn`
DEFAULT CHARACTER SET utf8mb4
COLLATE utf8mb4_0900_ai_ci ;
USE `tesisreconocimientofacialepn` ;
```

Segmento de Código 2.1 Script de creación de la base de datos

En el Segmento de Código 2.2 se muestra la creación de la Tabla `profesores`, para esto se utilizó el comando `CREATE TABLE IF NOT EXISTS`, además se definieron sus atributos los cuales son:

- `idProfesor:` es el identificador de la entidad profesor, su valor es único y se crea automáticamente al crear una instancia del mismo. Es del tipo integer de 11 dígitos.
- `cédula`: es un atributo del tipo char de 11 caracteres de longitud. Aunque el número de cédula de una persona es un número, para este proyecto se lo considera una cadena de texto ya que hay números de cédulas que empiezan con 0 y si fueran un entero, ese digito se perdería. Esta cadena tendrá un valor único, es decir no se podrá repetir.
- `nombre`: es un atributo del tipo char de 70 caracteres de longitud, no se aceptan valores null para este atributo, y corresponde al nombre del profesor.
- `apellido`: es un atributo del tipo char de 70 caracteres de longitud, no se aceptan valores null para este atributo, y corresponde al apellido del profesor.
- `password`: es un atributo del tipo char de 20 caracteres de longitud, no se aceptan valores null para este atributo, y corresponde a la contraseña del profesor para acceder al sistema.
- `idUniversidad`: es un atributo del tipo int de 20 dígitos de longitud, y se lo ha creado como Foreing Key de la tabla universidades, no se permiten almacenar valores nulos.

Para esta tabla la actualización de los datos es en cascada, ya que así se podrán actualizar los valores de todas las referencias que posea esta entidad. También se definió como codificación de caracteres `UTF8mb4` y el collage de caracteres `utf8mb4__0900_ai_ci.`

```
-- -----------------------------------------------------
-- Table `tesisreconocimientofacialepn`.`profesores`
-- Creacion de tabla profsores, con clave foranea idUniversidad`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `tesisreconocimientofacialepn`.`profesores` (
  `idProfesor` INT(11) NOT NULL AUTO_INCREMENT,
  `cedula` VARCHAR(11) NOT NULL,
  `nombre` VARCHAR(70) NOT NULL,
  `apellido` VARCHAR(70) NOT NULL,
  `password` VARCHAR(20) NOT NULL,
  `idUniversidad` INT(11) NOT NULL,
  PRIMARY KEY (`idProfesor`, `idUniversidad`),
  UNIQUE INDEX `idAdministrador_UNIQUE` (`idProfesor` ASC) VISIBLE,
  UNIQUE INDEX `cedula_UNIQUE` (`cedula` ASC) VISIBLE,
  INDEX `fk_Universidad_Profesor_idx` (`idUniversidad` ASC) VISIBLE,
  CONSTRAINT `fk_Universidad_Profesor`
    FOREIGN KEY (`idUniversidad`)
    REFERENCES `tesisreconocimientofacialepn`.`universidades` (`idUniversidad`)
    ON UPDATE CASCADE)
ENGINE = InnoDB
AUTO_INCREMENT = 13
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
```

Segmento de Código 2.2 Creación de la Tabla profesores

El Script completo de la base de datos se lo puede encontrar en el Anexo C

2.2.2 SPRINT 2

Para el Sprint 2 se han desarrollado las funciones de autenticación del servidor y el cliente, además se codificó al servidor para que una vez que arranque no se detenga y siempre esté a la escucha de conexiones de los clientes.

2.2.2.1 LogIn del servidor

Para el LogIn del servidor se diseñó la interfaz que se muestra en la Figura 2.25, esta interfaz consta de una consola que aceptará un usuario el cual debe estar registrado en la base de datos como Administrador.

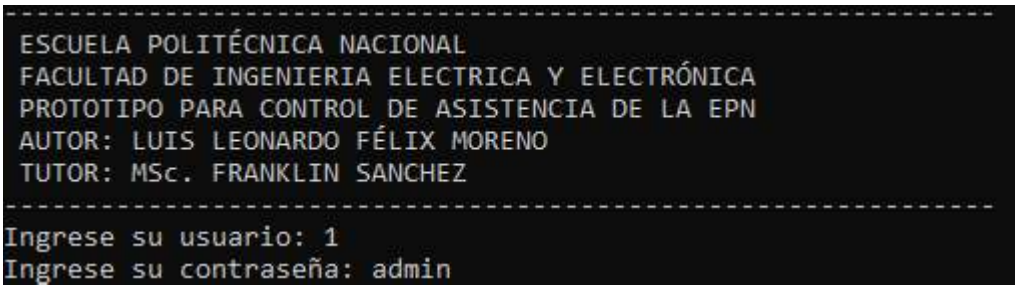

Figura 2.25 Interfaz de LogIn del Servidor

Para la implementación de este código se definió un método llamado `LogIn()` que se muestra en el Segmento de Código 2.3, dicho método se ejecuta desde el Main de la aplicación y retornará un `True` si la autenticación fue correcta o un `False` si no lo fue.

Para realizar la autenticación el usuario deberá ingresar su número de cédula seguido de su contraseña, estos dos atributos se los capturará en dos variables llamadas `user` (ver línea 12) y `password` (ver línea 13), seguido se realizará la conexión a la base de datos para realizar la consulta, dicha conexión se la crea a través de la invocación del Script `ModeloDB` (ver línea 15) que será explicado más adelante. Luego de tener la conexión a la base de datos se procede a ejecutar la consulta con los datos ingresados por el usuario, para esto se utiliza los métodos que proporciona PEEWEE.

La consulta del usuario se la realiza a través de un método `Select`, si coincide tanto en número de cédula y en password, se nos retornará la información completa del mismo, caso contrario retornará un valor `null`. En las líneas de código 18 al 30 se puede apreciar cómo se intenta obtener y manipular los datos del usuario con un `try`, ya que si el valor retornado es null se producirá una excepción que nos indicará que el usuario no existe o ingreso mal sus credenciales.

```
10  def Login():
11      #Login del sistema solo se aceptan administradores como usuarios
12      user = input('Ingrese su usuario: ')
13      password = input('Ingrese su contraseña: ')
14      mDB.db.close()
15      mDB.db.connect()
16      # realizo la consulta mediante el ORM PEEWEE
17      admin = mDB.administradores.select().where(mDB.administradores.cedula==user,
18                                                 mDB.administradores.password== password)
19      try:
20          #Obtengo el usuario administardor si esque exsiste retorno un TRUE para el logueo,
21          # caso contrario rechazo el logueo con un FALSE
22          for userDB in admin:
23              nombre = userDB.nombre
24              apellido = userDB.apellido
25          print ("Bienvenido : " + nombre +" "+ apellido+".")
26          mDB.db.close()
27          return True
28      except:
29          # controlo excepción generada al ingresar mal los valores
30          print ("Usuario no valido. Intente nuevamente")
31          mDB.db.close()
32          return False
```

Segmento de Código 2.3 Método LogIn

Para la conexión y manipulación de la base de datos se necesita las clases traductoras de objetos-relación y un método que nos permita conectarnos a la misma a través de una IP, un puerto, un usuario y una contraseña. Las clases traductoras se alojan en el archivo `ModeloDB.py` y servirán para realizar las consultas a la base de datos; deberán heredar de

una clase que contenga todas las propiedades de un modelo de traducción de PEEWEE. También dentro de este archivo se creó una variable global que es una instancia del método `conexionDPBW()`, el cual con los valores ya mencionados anteriormente realizará la conexión a MySQL. En el Segmento de Código 2.4 se muestra parte de las clases y la variable de conexión (ver línea 7).

En el Segmento de Código 2.5 se puede apreciar el método `conexionDBPW()`, el cual mediante la lectura de un archivo de configuración llamado "`ConectarDB.txt`" podrá obtener la información necesaria para crear la conexión a la base de datos, en la línea de código 8 se puede apreciar que mediante el método `open()` se lee el archivo `ConectarDB.txt`; en la línea 10 se crea la conexión a la base con el método `MySQLDatabase()` y la información del archivo anteriormente cargado, este método se encuentra en la librería `peewee`. Finalmente, en la línea 13 se retorna el objeto de la conexión.

En el Segmento de Código 2.6 se muestra la información del archivo `ConectarDB.txt`.

```
7   db = cDB.conexioDBPW()
8
9   class DBModel(pw.Model):
10      class Meta:
11          database= db
12
13  class administradores(DBModel):
14      idAdministrador= pw.AutoField(unique=True, primary_key= True)
15      cedula= pw.CharField(unique=True)
16      nombre= pw.CharField()
17      apellido = pw.CharField()
18      password = pw.CharField()
19
20  class universidades(DBModel):
21      idUniversidad= pw.AutoField()
22      nombreUniversidad= pw.CharField(unique=True)
23
24  class facultades(DBModel):
25      idFacultad= pw.AutoField()
26      nombreFacultad= pw.CharField(unique=True)
27      idUniversidad = pw.IntegerField(null=True)
```

Segmento de Código 2.4 Clases traductoras ORM

```
3   import peewee as pw
4
5   def conexioDBPW ():
6       # Metodo para la conexion a la base de datos,
7       # datos de conexion se leen del archivo
8       fileDB = open("ConectarDB.txt","r")
9       argDB = fileDB.readlines()
10      db= pw.MySQLDatabase(argDB[9], host=argDB[1].rstrip('\n'),
11                           port=int(argDB[3].rstrip('\n')),user=argDB[5].rstrip('\n'),
12                           passwd=argDB[7].rstrip('\n'))
13      return db
```

Segmento de Código 2.5 método `conexionDBPW`.

```
1  host=
2  192.168.100.229
3  port=
4  3[redacted]6
5  user=
6  [redacted]
7  passwd=
8  [redacted]
9  db=
10 tesisreconocimientofacialepn
```

Segmento de Código 2.6 Archivo `ConectarDB.txt`

2.2.2.2 Inicializar Servidor

Para que el servidor esté siempre funcionando se creó un bucle infinito que permitirá la conexión de los clientes. Para correr el servidor se deberá ingresar al `CMD` de Windows y posteriormente ingresar en la ubicación de la carpeta contenedora de los archivos del servidor llamada "`Tesis_ReconocimientoFacial`", luego se deberá ejecutar el comando "`python server.py`" (ver figura 2.26).

```
C:\Users\LEO\Desktop\Tesis_ReconocimientoFacial>python server.py
-----------------------------------------------------------------
 ESCUELA POLITÉCNICA NACIONAL
 FACULTAD DE INGENIERIA ELECTRICA Y ELECTRÓNICA
 PROTOTIPO PARA CONTROL DE ASISTENCIA DE LA EPN
 AUTOR: LUIS LEONARDO FÉLIX MORENO
 TUTOR: MSc. FRANKLIN SANCHEZ
-----------------------------------------------------------------
Ingrese su usuario: 1
Ingrese su contraseña: admin
Bienvenido : admin admin.


------------------MENU INICIO------------------
        1.- Iniciar servidor
        2 o dif.- Salir servidor

INGRESE UNA OPCION: 1
192.168.100.229
40001
Aceptar conexion..
```

Figura 2.26 Bucle del servidor

En el Segmento de Código 2.7 se muestra el Main del servidor, el cual estará dentro de un bucle infinito gracias a la sentencia `while`. Cuando se arranca el servidor se entra en el primer bucle que se ejecutará indefinidamente hasta que un logueo sea exitoso, esto se lo puede apreciar en las líneas 46 con el `while` infinito, en la línea 47 con la respuesta del método `LogIn()` y en la línea 48 con la verificación del logueo exitoso.

Después de que exista una autenticación exitosa en la línea 52 se esperará que el usuario ingrese el valor de 1 para crear la conexión del servidor y ponerlo a escuchar a clientes

(ver línea 56), finalmente se ejecutará el segundo bucle que aceptará y recibirá las peticiones de los clientes (ver línea 58).

En el Segmento de Código 2.8 se muestra la implementación del método `CrearEscuchar()` que permite al servidor crear la conexión del socket (ver línea 12 y 16) y posteriormente ponerlo a escuchar hasta 5 clientes simultáneos (ver línea 18). Este método para obtener la información de la IP del servidor y el puerto de escucha lee el archivo de configuración `ConectarSVR.txt`. El método finalmente retornará un objeto de conexión de socket que permitirá aceptar conexiones de clientes.

En el Segmento de Código 2.9 se muestra el método `AceptarRecibir()` que tiene como parámetro de entrada el objeto de conexión generado por el método `CrearEscuchar()`. Este método tiene como objetivo aceptar las conexiones de los clientes, procesarlas y generar respuestas de ser necesario. En la línea 26 como método del objeto de conexión se aceptan las peticiones de los clientes, al realizar esta acción se creará un objeto que se lo llamará `conexión`, el mismo que recibirá y enviará los mensajes. Posteriormente en la línea 29 se recibirá los mensajes del cliente y en la línea 31 se los decodificará según Utf-8, con esto se obtendrá en mensaje en texto plano. Dicho mensaje se lo dividirá para crear el objeto genérico llamado Protocolo (explicado en la sección de diseño) y se lo procederá a analizar según el ID que sea recibido.

Para dividir los mensajes se utiliza la sentencia `Split()` (ver línea 34) que permite separar en una lista de Strings los fragmentos de mensajes limitados por algún carácter, en este caso todas las cadenas separadas por “,”.

El servidor tratará de responder a todas las solicitudes generadas por los clientes, por lo que en cada análisis del objeto `Protocolo` se responderá mediante la sentencia `sendall()` (ver línea 43), con una codificación Utf-8 (ver línea 42) .

En el Segmento de Código 2.10, se puede apreciar la clase `Protocolo()`, la cual al instanciarla creará un objeto compatible con la estructura de datos enviada por el cliente. Este objeto tendrá un método llamado `AnalisisProtocolo()` que según el Id del mismo realizará alguna operación en el servidor, tal como un proceso CRUD, un entrenamiento del sistema, una toma de asistencia, entre otras opciones.

```
#--------------MAIN----------------------
#titulos de la aplicación
print("-------------------------------------------------------------------   ")
print (" ESCUELA POLITÉCNICA NACIONAL                                        ")
print (" FACULTAD DE INGENIERIA ELECTRICA Y ELECTRÓNICA                      ")
print (" PROTOTIPO PARA CONTROL DE ASISTENCIA DE LA EPN                      ")
print (" AUTOR: LUIS LEONARDO FÉLIX MORENO                                   ")
print (" TUTOR: MSc. FRANKLIN SANCHEZ                                       ")
print("-------------------------------------------------------------------   ")
x = True
#Lazo infinito para mantener al servidor en constante escucha
while  x == True:
    logVar = Login()
    if logVar == True :
        print("\n\n--------------------MENU INICIO--------------------")
        print("\t1.- Iniciar servidor")
        print("\t2 o dif.- Salir servidor\n")
        indexServer = input ("INGRESE UNA OPCION: ")
        if (indexServer == "1"):
            x= True
            ##################################################################
            socket-cSK.CrearEscuchar()
            while True:
                cSK.AceptarRecivir(socket)
            ##################################################################
        else:
            x=False
```

Segmento de Código 2.7 `Main` del servidor.

```
import socket

def CrearEscuchar():
    fileSVR = open("ConectarSVR.txt","r")
    argSVR = fileSVR.readlines()
    # objeto socket default
    misocket=socket.socket()
    #bind para la direcion del server y el puerto
    print(argSVR[1].rstrip('\n'))
    print(int(argSVR[3]))
    misocket.bind((argSVR[1].rstrip('\n'),int(argSVR[3])))
    #listen ara la cola
    misocket.listen(5)
    return misocket

```

Segmento de Código 2.8 Método `CrearEscuchar()`.

```
def AceptarRecivir(misocket):
    #al aceptar la conexion obtengo un obejoto conexion
    #que me permite manipular dicha conexion.
    print("Aceptar conexion..")
    conexion,addr =misocket.accept()
    print ("Conexión establecida")
    print (addr)
    bufferIN = conexion.recv(16384)
    print(bufferIN.decode('utf-8'))
    cadenaIN= bufferIN.decode('utf-8')
    cadenaIN= cadenaIN.strip("[")
    cadenaIN= cadenaIN.strip("]")
    cadenaIN = cadenaIN.split(", ")
    procesarProtocolo=Protocolo(cadenaIN[0],
                                cadenaIN[1], cadenaIN[2], cadenaIN[3],
                                cadenaIN[4], cadenaIN[5], cadenaIN[6],
                                cadenaIN[7],cadenaIN[8], cadenaIN[9],
                                cadenaIN[10])
    bufferOUT= procesarProtocolo.AnalisisProtocolo()
    print(bufferOUT)
    bufferOUT = bufferOUT.encode('utf-8')
    conexion.sendall(bufferOUT)
    conexion.close()

```

Segmento de Código 2.9 Método `AceptarRecibir()`

La clase `Protocolo()` tiene como constructor 1 String para el ID y 10 Strings para los diferentes campos de los mensajes enviados por los clientes, esto se debe a que muchas veces en los mensajes se enviará varios datos necesarios para realizar las consultas a la base de datos o realizar algún proceso específico en el servidor. No se creó un objeto por cada entidad ya que la extensión del código era demasiada y finalmente con un objeto genérico se podía manipular todos los datos sin problema, si algún dato debe ser de otro tipo diferente a String, se realizará la conversión respectiva.

```
#clase que procesa las peticiones de los clientes segun su identificador
class Protocolo (object):

    #se define el objeto protocolo que segun su id realizara varias acciones
    def __init__(self, idprocolo, objeto1, objeto2, objeto3, objeto4, objeto5, objeto6, objeto7, objeto8, objeto9, objeto10):
        self.idprocolo = idprocolo
        self.objeto1= objeto1
        self.objeto2 = objeto2
        self.objeto3 = objeto3
        self.objeto4 = objeto4
        self.objeto5 = objeto5
        self.objeto6 = objeto6
        self.objeto7 = objeto7
        self.objeto8 = objeto8
        self.objeto9 = objeto9
        self.objeto10 = objeto10

# funcion que orquesta toda la logica del servidor
    def AnalisisProtocolo(Protocolo): ...
    #Login
            if Protocolo.idprocolo=="0" and Protocolo.objeto3== "Administrador":
                mDB.db.close()
                print("Inicio de sesion APP android de administrador")
                mDB.db.connect()
                admin = mDB.administradores.select().where(mDB.administradores.cedula==Protocolo.objeto1,
                                                           mDB.administradores.password== Protocolo.objeto2)
                mDB.db.close()
                if len(admin) == 0:
                    return "False"
                if len(admin) > 0 :
                    return "True"
```

Segmento de Código 2.10 Clase `Protocolo()`

2.2.2.3 LogIn del cliente

Inicialmente la aplicación Android se ha tomado la decisión de llamarla “Control de Asistencia” y poseerá el logo mostrado en la Figura 2.27.

Figura 2.27 Ícono de la aplicación Android

Para la autenticación del cliente se creó la interfaz de usuario mostrada en la Figura 2.25, fue creada en Android Studio, al ser este proyecto un prototipo la parte visual del proyecto no será muy elaborada.

Figura 2.28 Interfaz de LogIn del cliente

En el Segmento de Código 2.11 se muestra toda la lógica del logueo, que se la ubicó en el botón de INGRESAR a través de su evento `onClick()`. Primero se obtiene los datos ingresados por el usuario y se los guarda en el objeto `CabeceraPerfilUsuarioConexion`,

este objeto se enviará clase por clase para que se pueda identificar que usuario se logueo en el sistema, y a que IP + Puerto se realizó la conexión. Seguido se instanciará el objeto genérico `ProtocoloDatos` con los datos del usuario, se lo enviará al servidor que tendrá un objeto compatible con el mismo. Después se instanciará el objeto `ConexionSocket`, el cual con la información del objeto genérico más la información del objeto `CabeceraPerfilUsuarioConexion` realizará la conexión al servidor, enviará el mensaje en texto plano codificado en UTF-8 y esperará alguna respuesta de ser necesario.

```
50      @Override
51      public void onClick(View v) {
52          // Metodo que realiza el LogIn de cliiente, captura los datos y los envia, ademas crea un objeto que se enviará
53          // a las demas calses para que puedan obetener la IP del servidor y que usuario se ha logueado
54          try {
55              switch (v.getId()) {
56                  case R.id.btnIniciar:
57                      if(txtUsuario.getText().toString().equals("")){
58                          Toast.makeText( context: MainActivity.this, text: "Error: Ingrese Un Usuario ", Toast.LENGTH_SHORT).show();
59                          break;
60                      } else {
61                          String id = "0";
62                          CabeceraPerfilUsuarioConexion cabeceraPerfilUsuarioConexion = new CabeceraPerfilUsuarioConexion(id,
63                                  txtIP.getText().toString(), txtPuerto.getText().toString(),
64                                  txtTipoUsuario.getSelectedItem().toString(), txtUsuario.getText().toString());
65                          ProtocoloDatos stringEnvio = new ProtocoloDatos(id, txtUsuario.getText().toString(),
66                                  txtPassword.getText().toString(),
67                                  txtTipoUsuario.getSelectedItem().toString());
68                          ConexionSocket socket = new ConexionSocket(txtIP.getText().toString(),
69                                  Integer.parseInt(txtPuerto.getText().toString()));
70                          socket.SendLogin(stringEnvio, context: MainActivity.this, cabeceraPerfilUsuarioConexion);
71                          break;
72                      }
73              }
74          }catch (Exception e)
75          {
76              Toast.makeText( context: MainActivity.this, text: "Error Al Conectar", Toast.LENGTH_SHORT).show();
77          }
78      }
```

Segmento de Código 2.11 Evento `onclick()`

En el Segmento de Código 2.12 se aprecia el método para enviar la solicitud de autenticación por parte del cliente, dicho método se almacena en la clase `ConexionSocket()`, y es el encargado de enviar la petición de logueo al servidor, luego espera una respuesta y procesa la información recibida. Si el logueo es exitoso ingresa al sistema, caso contrario muestra un mensaje de error.

Para realizar la conexión al servidor se crea un hilo (ver línea 76) en el cual se corre toda la lógica de envío y recepción, seguido se crea el socket (ver línea 82) con ayuda de la librería `java.net.Socket`; una vez creado se debe obtener un `OutputStream` para poder enviar la información de autenticación, luego se espera la recepción de confirmación con el evento `post`() (ver línea 90 al 107).

En el evento `post`() se codifica toda la lógica que debe cumplir la aplicación una vez reciba la respuesta que puede ser afirmativa o negativa. Si es afirmativa se autoriza el ingreso al sistema y se abre el layout del menú Principal de la aplicación (ver línea 94 al 97); caso

contrario se muestra un mensaje de error (ver línea 98 al 100). Finalmente se debe cerrar la conexión creada con el comando `close()`.

```
72      public void SendLogin(final ProtocoloDatos objIN, final Context context,
73                            final CabeceraPerfilUsuarioConexion cabeceraPerfilUsuarioConexion) {
74          // Metodo que envia al srvidor la peticion de logueo, y espera una confirmacion
75          final Handler handler = new Handler();
76          Thread thread = new Thread(new Runnable() {
77              @Override
78              public void run() {
79                  final String [] msg=objIN.GetString();
80                  try {
81                      // Conexion al servidor
82                      Socket s = new Socket(ip,puerto);
83                      OutputStream out = s.getOutputStream();
84                      //ObjectOutputStream output= new ObjectOutputStream(out);
85                      PrintWriter output = new PrintWriter(out);
86                      output.print(Arrays.toString(msg));
87                      output.flush();
88                      BufferedReader input = new BufferedReader(new InputStreamReader(s.getInputStream()));
89                      final String st = input.readLine();
90                      handler.post(new Runnable() {
91                          @Override
92                          public void run() {
93                              Intent intento = new Intent(context,MenuPrincipal.class);
94                              if (st.equals("True")){
95                                  Toast.makeText(context, text: "Ingreso Exitoso. Bienvenido", Toast.LENGTH_SHORT).show();
96                                  intento.putExtra( name: "CabeceraPerfilUsuarioConexion",cabeceraPerfilUsuarioConexion);
97                                  context.startActivity(intento);
98                              } else if (st.equals("False")) {
99                                  Toast.makeText(context, text: "Ingreso Fallido Usuario Y/O Contraseña Incorrecto. Intente Nuevamente",
100                                         Toast.LENGTH_SHORT).show();
101                             } else if (st.equals(null)) {
102                                 Toast.makeText(context, text: "Sin Respuesta Del Servidor", Toast.LENGTH_SHORT).show();
103                             } else {
104                                 Toast.makeText(context,st,Toast.LENGTH_SHORT).show();
105                             }
106                         }
107                     });
108                     output.close();
109                     out.close();
110                     s.close();
111                 } catch (IOException e) {
112                     e.printStackTrace();
113                 }
114             }
115         });
116
117         thread.start();
118     }
```

Segmento de Código 2.12 Hilo de conexión - Autenticación

En el lado del servidor la información se recibe por el método `AceptarRecibir()`, es instanciada en el objeto `Protocolo` y analizada por el método `AnalisisProtocolo()` según la ID que tenga. En dicho método se utilizarán consultas a través de la librería del ORM PEEWEE que simplifican la sintaxis de las consultas y las maneja como métodos de los objetos instanciados. En la sección de código 2.13 se muestra el logueo para un usuario profesor, en las líneas 44 y 45 se puede apreciar la sintaxis de la consulta `select` a la base de datos. Finalmente se retornará un True o False para que el cliente pueda saber que acción realizar.

```
31    def AnalisisProtocolo(Protocolo): ...
36    #Login
37    #
38            #  | idProtocolo , objeto1      ,    objeto2     , objeto3
39            #  | cabecera    ,    cedula    ,   contraseña   , tipo de usuario
40            if Protocolo.idprocolo=="0" and Protocolo.objeto3== "Administrador":
41                mDB.db.close()
42                print("Inicio de sesion APP android de administrador")
43                mDB.db.connect()
44                admin = mDB.administradores.select().where(mDB.administradores.cedula==Protocolo.objeto1,
45                                                           mDB.administradores.password== Protocolo.objeto2)
46                mDB.db.close()
47                if len(admin) == 0:
48                    return "False"
49                if len(admin) > 0 :
50                    return "True"
```

Segmento de Código 2.13 Método Análisis Protocolo

2.2.3 SPRINT 3

Para el Sprint 3 se creó la interfaz de usuario y la lógica para la gestión de usuarios Administradores. No se ha creado restricción de este tipo de usuarios. Cada Administrador podrá tener acceso a toda la información y módulos del sistema como ya se lo ha descrito en secciones anteriores.

En la aplicación Android del cliente se ha diseñado la interfaz de usuario tomando en cuenta los requerimientos ya propuestos; se la muestra en la Figura 2.29.

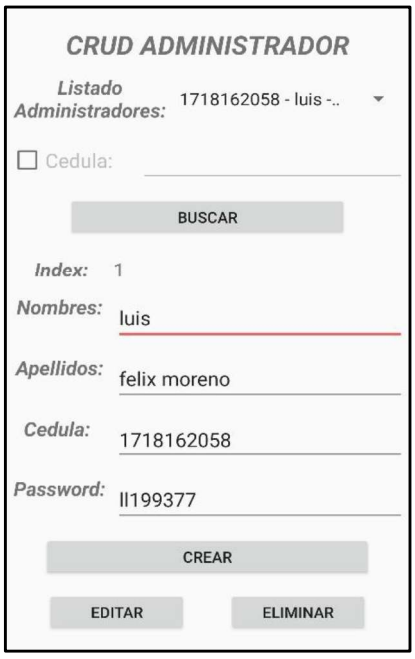

Figura 2.29 interfaz de Administrador

La interfaz mostrada en la Figura 2.29 permite realizar el proceso CRUD de un usuario Administrador, el layout consta de un botón de búsqueda que permite obtener la información de un usuario ya sea de una selección de lista de Administradores o de una búsqueda en base al número de cédula. Esta búsqueda y carga de información permite editar datos o eliminar al usuario de la base de datos.

Para el proceso de creación no es necesario buscar y cargar un usuario ya existente. Solo al ingresar al formulario se debe llenar los campos y seleccionar la opción de Crear.

En el Segmento de Código 2.14 se muestra el método `OnCreate()` de la clase `CRUDAdministrador()`, la cual hereda de la clase `Activity` e implementa el evento `OnClickListerner()`, el cual se disparará al pulsar algún botón. En las líneas de código 16 al 23 se ha creado las variables que representarán a cada componente del Layout; en la línea 30 se implementó el código para realizar el paso de valores entre formularios para el objeto `CabeceraPerfilUsuarioConexion` en el cual se recibe la información el usuario que está conectado, la IP del servidor y el puerto de escucha. En la línea 37 se enlaza al botón `btnCrear` del layout con la variable `btnCrear`(se han mantenido los mismos nombres para evitar confusiones), y en la línea 39 se le ha asignado el evento `OnClickListener`.

```
13  // Clase para la gestion de suarios Administradores
14  public class CRUDAdministrador extends Activity implements View.OnClickListener{
15
16      Spinner  txtListaAdministradores;
17      EditText txtNombreAdministrador, txtApellidoAdministrador, txtCedulaAdministrador,
18              txtBuscarCedula, txtPasswordAdministrador;
19      TextView txtIDAdministrador;
20      CabeceraPerfilUsuarioConexion cabeceraConexion;
21      Button btnCrear, btnBuscar, btnEditar, btnEliminar;
22      LinearLayout lytEditarBorrar, lytCrearBuscar;
23      CheckBox cbxCedula;
24
25      @Override
26      protected void onCreate(Bundle savedInstanceState) {
27          super.onCreate(savedInstanceState);
28          setContentView(R.layout.activity_crudadministrador);
29
30          cabeceraConexion = (CabeceraPerfilUsuarioConexion)getIntent().getExtras()
31                              .getSerializable( key: "CabeceraPerfilUsuarioConexion");
32          lytEditarBorrar = findViewById(R.id.lytEditarBorrar);
33          lytEditarBorrar.setVisibility(View.INVISIBLE);
34          lytCrearBuscar= findViewById(R.id.lytCrearBuscar);
35
36          // asigancion botones
37          btnCrear = (Button) findViewById(R.id.btnCrear);
38          btnBuscar = (Button) findViewById(R.id.btnBuscar);
39          btnCrear.setOnClickListener(this);
40          btnBuscar.setOnClickListener(this);
```

Segmento de Código 2.14 CRUD Administradores Método On Create

En el Segmento de Código 2.15 se muestra el método on Click() para los botones del Layout, para saber que botón fue presionado y que acción realizar, se utiliza el id que se obtiene del view de la clase y el Layout al momento de ejecutar el evento presionar un botón; al tener el id del botón sabremos qué acción se debe ejecutar, para esto se agregó una función `switch`(), para que ejecute un Segmento de Código según el id que reciba, toda esta programación se encuentra en la línea 97. En las líneas 106 a la 116 se ejecuta la acción de crear un administrador, en las líneas 108 al 110 se instancia un objeto genérico `ProtocoloDatos()` para enviar de forma genérica la información al servidor, dicho objeto obtendrá la información de los TextBox que lleno el usuario con información y tendrá un identificador para que el servidor sepa que acción realizar en este caso se envía con ID = 61.

En la línea 111 se instancia un objeto `ConexionSocket()` el cual creará la conexión con el servidor, para esto recibirá la información de la IP y puerto del servidor. En la línea 112 se envía el objeto `ProtocoloDatos` para que sea enviado al servidor y se espere una respuesta, también se envía el contexto de la clase para que el objeto `ConexiónSocket` pueda acceder a las características del mismo, y muestre la respuesta del servidor.

```
91      @Override
92      public void onClick(View v) {
93          //Evento onClick se ejecuta al presionar un boton
94          ProtocoloDatos stringEnvio;
95          ConexionSocket socket;
96          try {
97              switch (v.getId()) {
98                  // 6_= Administrador
99                  // 61 = crear
100                 // 62 = buscar
101                 // 63 = editar
102                 // 64 = ELiminar
103                 // 65 = CargarAdministradores
104                 // 66 = CargarAdministradoresCedula
105
106                 case R.id.btnCrear:
107                     // Boton Crear, envia la información de los textbox para crear un nuevi Administardor
108                     stringEnvio = new ProtocoloDatos( id: "61", txtIDAdministrador.getText().toString(),
109                             txtNombreAdministrador.getText().toString(),txtApellidoAdministrador.getText().toString(),
110                             txtCedulaAdministrador.getText().toString(), txtPasswordAdministrador.getText().toString());
111                     socket = new ConexionSocket(cabeceraConexion.getIp(), Integer.parseInt(cabeceraConexion.getPuerto()));
112                     socket.SendAdministrador(stringEnvio, context: CRUDAdministrador.this);
113                     lytEditarBorrar.setVisibility(View.INVISIBLE);
114                     LimpiarTXT();
115                     ActualizarSpinners();
116                 break;
117
```

Segmento de Código 2.15 Método `onClick`()

Para las demás opciones del CRUD, incluyendo la búsqueda de Administradores existentes se sigue el mismo procedimiento. Primero se instancia sus componente en el método onCreate(), luego a los botones de cada proceso se le asigna el evento onClickListener(), tercero se le asigna una función a cada botón cuando se ejecute un

evento como se mostró anteriormente. Finalmente, toda la información recopilada es enviada con un identificador al servidor para que sea procesada y se espera una respuesta.

En el objeto `ConexionSocket` cuando se desea obtener la lista de Administradores se lo envía a través del método `LoadAdministrador()` y cuando se envía la información al servidor para que sea procesada se lo hace mediante el método `SendAdministrador()`, esto se lo realiza ya que los dos métodos enviarán datos al servidor y esperarán una respuesta, pero en el caso de `LoadAdministrador()` cargará una lista de administradores y en el de `SendAdministrador()` solo recibirá una confirmación de CRUD exitoso.

En el Segmento de Código 2.16 se muestra cómo se envía la información al servidor, al igual que en el Sprint 1 se recibe el objeto genérico y el contexto de la clase `CRUDAdministrador()` (ver línea 601), seguido se crea un hilo que manejará la conexión en él envío y esperará una respuesta (ver línea 604). Tercero se crea la conexión al servidor con la IP y el puerto del objeto instanciado (ver línea 609) y se envía la información (líneas 610 al 613). Después de enviar la información se espera una respuesta mediante un `BufferReader` (ver línea 614), luego al buffer de respuesta se lo lee línea por línea y se obtiene un String de respuesta (ver línea 615). A la respuesta se le ejecuta una función `post()` para que realice una acción después de recibir el mensaje del servido (ver línea 616).

El servidor enviará una cadena de texto que a simple vista no tendrá coherencia ya que toda la información estará unida, para esto en la línea 620 se separa el String con la función `Split()` y utilizando los caracteres " ," como delimitadores. Seguido en las líneas 621 a la 627 se muestra la información de los administradores en un Spinner.

En el Segmento de Código 2.17, en las líneas 630 a la 632 se cierran las conexiones y los buffers. Finalmente, en la línea 638 se da el comando para ejecutar en hilo.

En el Segmento de Código 2.18 se muestra la función `SendAdministrador()`, en la cual se envía una solicitud CRUD y se espera una respuesta. El procedimiento es el mismo que para la función `LoadAdministrador()`, solo cambia en las sentencias que se ejecutan en la función `post()`(líneas 660 a la 686), ya que se tomará en cuenta que puede haber una respuesta de CRUD correcta, incorrecta, o de carga de información de un usuario en los TextBoxs.

En el servidor la petición del cliente se la procesa en base al identificador que se recibe, la cadena de texto enviada por el cliente es descompuesta para obtener toda la información necesaria y poder procesarla en base de datos.

En el Segmento de Código 2.19 se puede apreciar cómo se procesa una cadena en el servidor según su identificador, para esto previamente se la descomprime utilizando la función `split`(), luego pasa a la función `AnalisisProtocolo`() y según el ID se ejecuta alguna acción, en este caso si se posee un id = 61 se agregará un nuevo administrador (Líneas 659 al 662) y se enviará la confirmación mediante el mismo método que en el Sprint1. En el Segmento de Código 2.20 se muestra la sentencia para modificar un administrador y el en Segmento de Código 2.21 se muestra la sentencia para eliminarlo.

```
601  public void LoadAdministrador(final ProtocoloDatos objIN, final Context context) {
602      //Cargo la info de Profesor en CRUD porfesor
603      final Handler handler = new Handler();
604      Thread thread = new Thread(new Runnable() {
605          @Override
606          public void run() {
607              final String [] msg=objIN.GetString();
608              try {
609                  Socket s = new Socket(ip,puerto);
610                  OutputStream out = s.getOutputStream();
611                  PrintWriter output = new PrintWriter(out);
612                  output.print(Arrays.toString(msg));
613                  output.flush();
614                  BufferedReader input = new BufferedReader(new InputStreamReader(s.getInputStream()));
615                  final String st = input.readLine();
616                  handler.post(new Runnable() {
617                      @Override
618                      public void run() {
619                          Intent intento = new Intent(context,CRUDFacultad.class);
620                          String[] profesor = new String[1000];
621                          profesor=(st.split( regex: ", "));
622                          Spinner txtListaAdministradores =
623                                  (Spinner) ((Activity)context).findViewById(R.id.txtListaAdministradores);
624                          List<String> listaAdministrador = new ArrayList<>();
625                          Collections.addAll(listaAdministrador,profesor);
626                          ArrayAdapter<String> adapterSpinner = new ArrayAdapter<>(
627                                  context,android.R.layout.simple_spinner_item,listaAdministrador);
628                          txtListaAdministradores.setAdapter(adapterSpinner);
629                      }
630                  });
```

Segmento de Código 2.16 Método `LoadAdministrador` de `ConexionSocket` Parte I

```
631                      output.close();
632                      out.close();
633                      s.close();
634                  } catch (IOException e) {
635                      e.printStackTrace();
636                  }
637              }
638          });
639          thread.start();
640      }
```

Segmento de Código 2.17 Método `LoadAdministrador` de `ConexionSocket` Parte II

```
handler.post(new Runnable() {
    @Override
    public void run() {
        Intent intento = new Intent(context,MenuPrincipal.class);

        if (st.equals("True")){
            Toast.makeText(context, text: "Proceso CRUD Exitoso", Toast.LENGTH_SHORT).show();
        } else if (st.equals("False")) {
            Toast.makeText(context, text: "Proceso CRUD Erroneo. Intente Nuevamente",
                    Toast.LENGTH_SHORT).show();
        } else if (st.equals(null)) {
            Toast.makeText(context, text: "Sin Respuesta Del Servidor",
                    Toast.LENGTH_SHORT).show();
        } else {
            String[] administrador = new String[1000];
            administrador=(st.split( regex: ", "));
            EditText txtNombreAdministrador, txtApellidoAdministrador, txtCedulaAdministrador,
                    txtBuscarCedula, txtPasswordAdministrador;
            TextView txtIDAdministrador;
            txtIDAdministrador = ((Activity)context).findViewById(R.id.txtIDAdministrador);
            txtNombreAdministrador = ((Activity)context).findViewById(R.id.txtNombreAdministrador);
            txtApellidoAdministrador = ((Activity)context).findViewById(R.id.txtApellidoAdministrador);
            txtCedulaAdministrador = ((Activity)context).findViewById(R.id.txtCedulaAdministrador);
            txtPasswordAdministrador= ((Activity)context).findViewById(R.id.txtPasswordAdministrador);
            txtIDAdministrador.setText(administrador[0].toString());
            txtNombreAdministrador.setText(administrador[1].toString());
            txtApellidoAdministrador.setText(administrador[2].toString());
            txtCedulaAdministrador.setText(administrador[3].toString());
            txtPasswordAdministrador.setText(administrador[4].toString());
        }
    }
```

Segmento de Código 2.18 Método post `SendAdministrador`

```
administrador= mDB.administradores.update({mDB.administradores.nombre:Protocolo.objeto2,
                                           mDB.administradores.apellido:Protocolo.objeto3,
                                           mDB.administradores.cedula:(Protocolo.objeto4),
                                           mDB.administradores.password:Protocolo.objeto5}).where(mDB.administradores.idAdministrador==
                                                                                                  int(Protocolo.objeto1))
rowUpdate=administrador.execute()
```

Segmento de Código 2.19 Sentencia `Update`

```
        print(Protocolo.objeto1)
        administrador= mDB.administradores.delete().where(mDB.administradores.idAdministrador==
                                                          int(Protocolo.objeto1))
        rowUpdate=administrador.execute()
```

Segmento de Código 2.20 Sentencia `Delete`

Para los CRUDs de este proyecto se utilizará la misma lógica del administrador en el servidor, lo único que cambiará son los campos requeridos, pero todos se recibirán en forma de String, y seguirán la lógica en la que el servidor recibe la petición del cliente, la procesa y obtiene el ID, en base a dicho identificador se consultará a la base de datos y responderá con una lista de datos de objetos, una confirmación o un rechazo.

La lógica de los CRUDs y las interfaces de usuario en el cliente, también seguirán la misma lógica que del `CRUDAdministador.java`, lo único que cambiará serán los campos que posee cada objeto.

2.2.4 SPRINT 4

En el cuarto Sprint llamado “Gestión de abstracciones: universidad, facultad, carrera y materia” se crearon las interfaces de usuario en base a los bosquejos de la sección 2.1.12 para su parte visual y para la parte lógica se utilizó el mismo proceso que en el Sprint 2 para el usuario Administrador.

2.2.4.1 CRUD Universidad

Inicialmente para creación de la abstracción Universidad solo se solicita un nombre capturado por un TextBox, ya que en la base de datos ese es su único campo. También mediante un Spinner se lista las universidades ya creadas, para que el usuario pueda cargar su información mediante un botón de “Cargar o Buscar” y posteriormente se pueda realizar un proceso de Update o eliminación. En la Figura 2.30 Se muestra la interfaz del `CRUDUniversidad` almacenado en el archivo `activity_cruduniversidad.xml`

Figura 2.30 Interfaz `CRUDUniversidad`

En la parte del cliente el objeto `ProtocoloDatos` es encapsulado y enviado al servidor para este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para un usuario Administrador.

2.2.4.2 CRUD Facultad

Para la creación de la abstracción Facultad se solicita un nombre capturado por un TextBox y mediante un Spinner se debe escoger una universidad anteriormente creada. También mediante otro Spinner se lista las facultades pertenecientes a una universidad, para que el usuario pueda cargar su información mediante un botón de "Cargar o Buscar" y posteriormente se pueda realizar un proceso de Update o Eliminación. En la Figura 2.31 se muestra la interfaz del `CRUDFacultad` almacenado en el archivo `activity_crudfacultad.xml`

Figura 2.31 Interfaz CRUDFacultad

En la parte del cliente el objeto `ProtocoloDatos` es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para un usuario Administrador.

2.2.4.3 CRUD Carrera

Para la creación de la abstracción Carrera se solicita un nombre capturado por un TextBox, mediante un Spinner se debe escoger una universidad anteriormente creada y mediante otro Spinner se deberá escoger a que facultad de la universidad seleccionada pertenece la carrera a crear. También mediante otro Spinner se lista las carreras, para que el usuario pueda cargar su información mediante un botón de "Cargar o Buscar" y posteriormente se pueda realizar un proceso de Update o eliminación. En la Figura 2.32 se muestra la interfaz del `CRUDCarrera` almacenado en el archivo `activity_crudcarrera.xml`.

Cuando un usuario escoge una opción de universidad, automáticamente se cargarán en el Spinner correspondiente las facultades existentes para dicha abstracción, lo mismo sucede al escoger una facultad, se mostrará en el Spinner las carreras pertenecientes a dicha facultad.

En la parte del cliente el objeto `ProtocoloDatos` es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para un usuario Administrador.

Figura 2.32 Interfaz `CRUDCarrera`

En el Segmento de Código 2.21 se muestra cómo se cargarán en un Spinner las carreras de una facultad. Para esto inicialmente se debió haber creado un Spinner que en este caso se lo ha llamado `txtListaFacultad` y también se le debió haber asignado el evento `ItemSelected` (ver línea 105). Seguido se lo debe programar para cuando se seleccione una facultad, se carguen en otro Spinner las carreras asociadas a la misma, esto coincide con la base de datos ya que la Tabla carreras y facultades tienen una relación.

En la fila 106 se crea un adaptador para el evento. Este adaptador estará a la escucha de que se seleccione o no una carrera; en las líneas 107 a la 128 se ha programado la lógica a ejecutar cuando en el Spinner se seleccione una opción disponible. El procedimiento a realizar es el mismo que se ha utilizado para cargar la lista de administradores, lo único que cambia es el ID que se le asigna al objeto `ProtocoloDatos` y el método `LoadCarrera()`, ya que la información que maneja dicha abstracción es diferente a la de Administrador.

El evento seteado al Spinner evento para cargar lista de universidades y lista de facultades, al igual que se lo utilizará para las demás interfaces de usuario; no se mostrará dicha programación ya que sería redundar en la lógica utilizada.

En la línea 130 se implementó el método `onNothingSelected`, el cual se ejecutará cuando no se seleccione ningún ítem del Spinner; para todos los casos este método permanecerá vacío ya que no realizará ninguna acción, pero su implementación es un requisito del evento.

```
104     // Se carga la información de las carreras al selecionar una facultad
105     txtListaFacultad.setOnItemSelectedListener(
106             new AdapterView.OnItemSelectedListener() {
107                 public void onItemSelected(AdapterView<?> parent, View view,
108                                            int pos, long id) {
109                     if (lytEditarBorrar.getVisibility()==View.VISIBLE){
110                         DisableTxt();
111                     } else {
112                         try {
113                             // Solicito al servidor la lista de carreras para la
114                             // facultad seleccionada
115                             ProtocoloDatos stringEnvio;
116                             ConexionSocket socket;
117                             stringEnvio = new ProtocoloDatos( id: "37",  dato1: "",  dato2: "",
118                                     txtListaFacultad.getSelectedItem().toString(),
119                                     txtUniversidad.getSelectedItem().toString());
120                             socket = new ConexionSocket(cabeceraConexion.getIp(),
121                                     Integer.parseInt(cabeceraConexion.getPuerto()));
122                             socket.LoadCarrera(stringEnvio,  context: CRUDCarrera.this);
123                         } catch (Exception e) {
124                         }
125
126                         lytEditarBorrar.setVisibility(View.INVISIBLE);
127                         EnableTxt();
128                     }
129                 }
130                 public void onNothingSelected(AdapterView<?> parent) {
131                 }
132             });
```

Segmento de Código 2.21 Evento `setOnItemSelected`

2.2.4.4 CRUD Materia

Para la creación de la abstracción Materia se solicita un nombre capturado por un TextBox, mediante Spinners se deberán escoger: una universidad, una facultad y una carrera, anteriormente creadas. También mediante otro Spinner se listarán materias, para que el usuario pueda cargar su información mediante un botón de “Cargar o Buscar” y posteriormente se pueda realizar un proceso de Update o Eliminación. En la Figura 2.33 se muestra la interfaz del `CRUDMateria` almacenado en el archivo `activity_crudmateria.xml`.

Cuando un usuario escoge una opción de universidad, automáticamente se cargarán en el Spinner correspondiente las facultades existentes para dicha abstracción, lo mismo sucede al escoger una facultad y carrera.

En la parte del cliente el objeto `ProtocoloDatos` es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para el usuario Administrador.

Figura 2.33 Interfaz `CRUDMateria`

2.2.5 SPRINT 5

En el SPRINT 5 llamado “Usuario Profesor y Abstracciones Parte 2” se han creado las interfaces para el usuario Profesor y las abstracciones Paralelo y Estudiante, la lógica de programación para el usuario y abstracciones es la misma para las ya creadas en la sección anterior, por lo que explicarlas a fondo sería redundar en las ideas ya planteadas.

2.2.5.1 Usuario Profesor

Para la creación del usuario Profesor se solicitan los datos determinados en la base de datos como son: nombre, apellido, cédula y password, todos capturados por TextBoxs; y mediante un Spinner se deberá escoger una universidad ya que si revisamos el diagrama relacional los usuarios profesores deberán pertenecer a una universidad previamente creada.

Mediante un Spinner se listarán los Profesores ya creados para poder seleccionarlos y cargar su información mediante un botón de “Cargar o Buscar”, para realizar el proceso de Update o Eliminación. Otro método creado para la búsqueda de Profesores es mediante su número de cédula, para esto se deberá activar un Checkbox llamado `cbxCédula`, el cual activará un Textbox que permitirá ingresar el número de cédula, finalmente se deberá pulsar el botón “Cargar o Buscar”. En la Figura 2.34 se muestra la interfaz del CRUDProfesor almacenado en el archivo activity_crudprofesor.xml.

Cuando se une a un profesor a una universidad, el usuario ya no podrá pertenecer a otra institución, pero podrá ser asignado a cualquier materia y paralelo de dicha universidad. No podrán existir dos o más Profesores con cédulas iguales, pero si se podrá crear un usuario Administrador con la misma cédula.

En la parte del cliente el objeto `ProtocoloDatos` es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para el usuario Administrador.

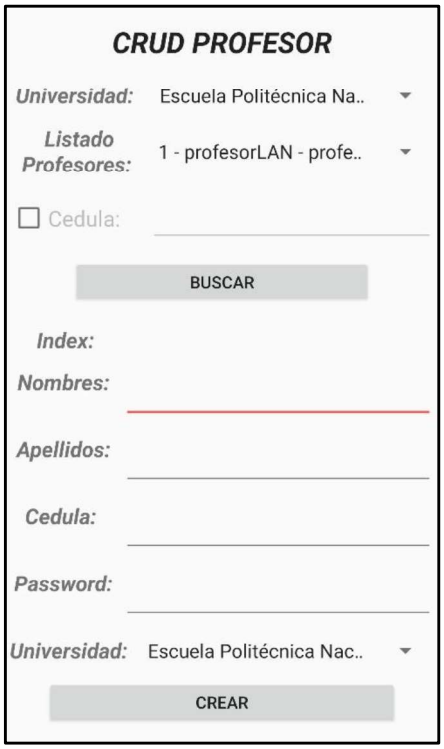

Figura 2.34 Interfaz `CRUDProfesor`

2.2.5.2 CRUD Paralelo

Para la creación de la abstracción Paralelo se solicita un nombre capturado por un TextBox, mediante Spinners se deberán escoger: una universidad, una facultad, una carrera, una

materia y un profesor, anteriormente creados. También mediante otro Spinner se listarán los paralelos disponibles para la materia seleccionada; esto para que el usuario pueda cargar su información mediante un botón de “Cargar o Buscar” y posteriormente se pueda realizar un proceso de Update o Eliminación. En la Figura 2.35 se muestra la interfaz del CRUDParalelo almacenado en el archivo activity_crudparalelo.xml.

Cuando un usuario crea un paralelo se deberá escoger en la parte de información de la abstracción una materia y un profesor, esto se debe a que un paralelo según el diagrama relacional estará ligado a los mismos.

En la parte del cliente el objeto ProtocoloDatos es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para el usuario Administrador.

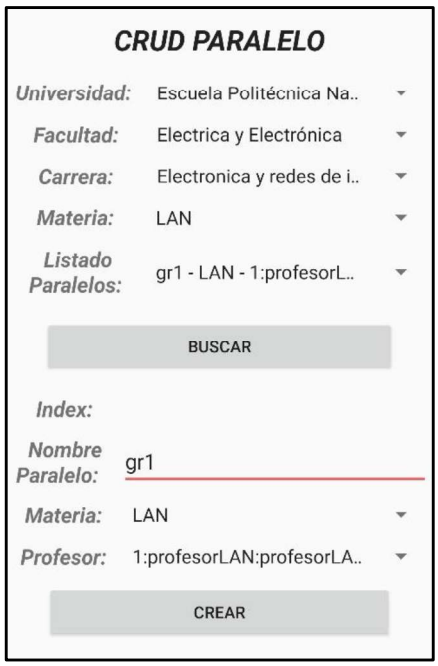

Figura 2.35 Interfaz `CRUDParalelo`

2.2.5.3 CRUD Estudiante

Para la creación de la abstracción Estudiante se solicita la información de: nombre, apellidos, cédula, número único capturado por TextBoxs, mediante Spinners se deberán escoger: una universidad, una facultad y una carrera, anteriormente creadas. También mediante otro Spinner se listarán los estudiantes disponibles para la carrera seleccionada; esto para que el usuario pueda cargar la información mediante un botón de “Cargar o Buscar” y posteriormente se pueda realizar un proceso de Update o Eliminación. En la

Figura 2.36 se muestra la interfaz `CRUDEstudiante` almacenado en el archivo "`activity_crudestudiante.xml`".

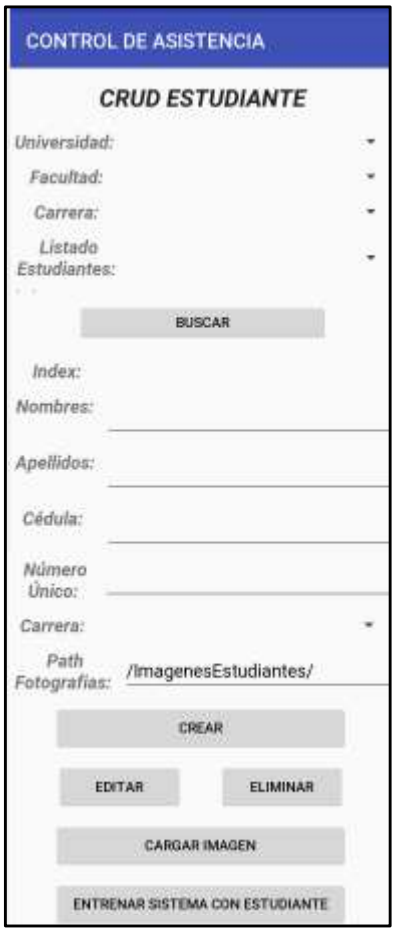

Figura 2.36 Interfaz `CRUDEstudiante`

Cuando un usuario crea un Estudiante se deberá escoger en la parte de información de la abstracción una Carrera, esto se debe a que un Estudiante según el diagrama relacional estará ligado a la misma. También se generará automáticamente el path("/ImagnesEstudiantes/cédula") donde se creará la base de imágenes del estudiante, adicionalmente se ha añadido el botón para agregar fotografías a dicha base y el botón para entrenar el sistema con el estudiante, pero dichos botones se programarán en los Sprint posteriores.

En la parte del cliente el objeto `ProtocoloDatos` es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó para el usuario Administrador.

2.2.6 SPRINT 6

En el SPRINT 6 llamado “Abstracciones Parte 3” se han creado las interfaces y parte lógica para las abstracciones Paralelo-Estudiante y Actividad Académica, la lógica de programación de las abstracciones es la misma para las ya creadas en la sección anterior, por lo que explicarlas a fondo sería redundar en las ideas ya planteadas.

2.2.6.1 Asignar estudiante a paralelo

En el diagrama relacional de la base de datos, al realizar la cardinalidad entre estudiantes y paralelos se observó que había una relación de muchos a muchos, por lo que se creó la tabla `estudiantes Paralelo;` para satisfacer dicha Tabla se ha creado una interfaz para asignar estudiantes a un paralelo.

En la Figura 2.37 se muestra la interfaz “Asignar Estudiantes A Paralelo” ubicada en el archivo activity_crudestudiante_paralelo.xml. La interfaz creada permitirá seleccionar un paralelo enlazado a una materia y adicionalmente se podrá escoger un estudiante ya sea de una lista de estudiantes creados o realizando una búsqueda por su número de cédula. Al cargar la información seleccionada se podrá realizar la acción de enlazar a un estudiante a dicho paralelo o eliminarlo mediante botones.

Al seleccionar cada paralelo automáticamente se cargará la lista de los estudiantes enlazados, al seleccionar un estudiante se podrá eliminarlo. Para crear dicha lista se utilizó un ListView, en el cual se cargará una lista de objetos Estudiantes a través de una clase adaptadora.

En los segmentos de código 2.22 y 2.23 se muestra el método `CargarListView(),` que se ejecuta cuando se activa el evento `OnItemSelectedListener` del Spinner que contiene la lista de paralelos. En las líneas 480 a la 483 se realiza una comprobación (mediante un `if`) para verificar la existencia de las abstracciones universidad, facultad, carrera y paralelo; si alguna de estas abstracciones no existiera en el ListView se cargará un mensaje de error, a través de un `ArrayAdapter` (ver línea 488) que contendrá el contexto del layout, el formato visual del ListView, y los valores en cadenas de texto a mostrar. Finalmente en la línea 499 asignamos el ArrayAdapter a la lista mediante el método `setAdapter().`

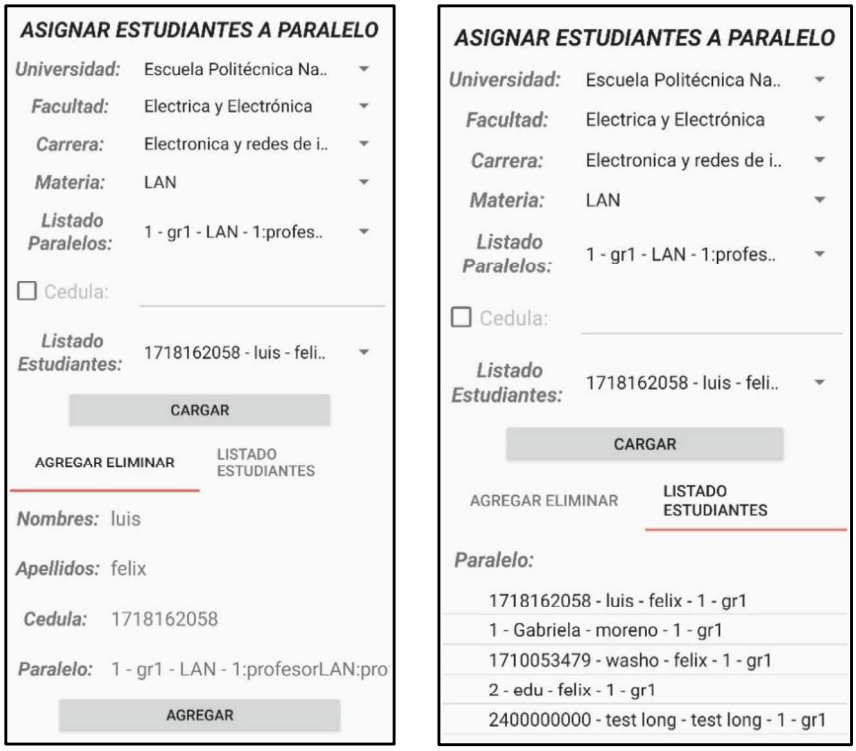

Figura 2.37 Interfaz Asignar Estudiante a Paralelo

```
474    private void CargarListView(){
475        // Metodo que verifica si existe las abstracciones superioes a paralelo,
476        // si exsisten las abtracciones las caragrá, caso contraio cargara en el
477        // Spinner un mensaje de Error.
478        ProtocoloDatos stringEnvio;
479        ConexionSocket socket;
480        if(txtUniversidad.getSelectedItem().toString().equals("False") ||
481                txtListaFacultad.getSelectedItem().toString().equals("False") ||
482                txtListaCarrera.getSelectedItem().toString().equals("False") ||
483                txtListaParalelo.getSelectedItem().toString().equals("False")) {
484            Toast.makeText( context: this,
485                    text: "Error no se encontró ninguna Facultad, Carrera, Materia o Paralelo" ,
486                    Toast.LENGTH_SHORT).show();
487            String[] valores = new String[]{"No se ha seleccionado un Paralelo"};
488            ArrayAdapter<String> adapter = new ArrayAdapter<~>
489                    ( context: this,android.R.layout.simple_expandable_list_item_1,valores) {
490                @Override
491                public View getView(int position, View convertView, ViewGroup parent) {
492                    View view = super.getView(position, convertView, parent);
493                    ViewGroup.LayoutParams parametros = view.getLayoutParams();
494                    parametros.height = 100;
495                    view.setLayoutParams(parametros);
496                    return view;
497                }
498            };
```

Segmento de Código 2.22 Método CargarListView Parte I

```
499         lstEstudiatesParalelo.setAdapter(adapter);
500     } else {
501         String[] paraleloEstudiante = new String[100];
502         paraleloEstudiante = txtListaParalelo.getSelectedItem().toString().split( regex: " - ");
503         stringEnvio = new ProtocoloDatos( id: "102", paraleloEstudiante[0]);
504         socket = new ConexionSocket(cabeceraConexion.getIp(),
505                 Integer.parseInt(cabeceraConexion.getPuerto()));
506         socket.LoadEstudianteParalelo(stringEnvio, context: CRUDEstudianteParalelo.this);
507         txtParaleloEstudiante.setText(txtListaParalelo.getSelectedItem().toString());
508     }
```

Segmento de Código 2.23 Método CargarListView Parte II

Si el `if` de las líneas 480 a la 483 resulta en un `false`, significa que existen paralelos y sus abstracciones superiores, en ese caso se crea un objeto `ProtocoloDatos` con el id de consulta de estudiantes por paralelos que en este caso es 102 y la información del paralelo (ver línea 503). En la línea 504 se crea la conexión al servidor y en la línea 506 mediante la función `LoadEstudianteParalelo`() del objeto socket se envía al servidor el objeto `ProtocoloDatos` para realizar la consulta de los estudiantes asignados a dicho paralelo, posteriormente se cargará en el ListView.

En los segmentos de códigos 2.24 y 2.25 se puede apreciar el método `LoadEstudianteParalelo()`. Este método es similar a `SendLogin`() mostrado en el Segmento de Código 2.12; en el cual primero se define la programación para el hilo, luego se crea la conexión con el servidor y se envía el objeto `ProtocoloDatos`, se espera una respuesta. Con la respuesta en la línea 920 se crea un ListView y se lo enlaza con el componente del Layout, en la línea 922 se separa el String que se recibió y en línea 923 se crea el adaptador `ArrayAdapter`, se le asigna la lista a mostrar, se le da parámetros visuales y finalmente al Listview se le asigna el adaptador. Finalmente se ejecuta el hilo.

Para todos los casos el servidor recibirá la trama a analizar y según el ID recibido consultará a la base y responderá según corresponda.

2.2.6.2 CRUD Actividad académica

Para la creación de la abstracción Actividad Académica se solicita la información de: paralelo, y profesor que se cargarán en TextBoxs automáticamente al escoger un paralelo, también se necesitará una fecha. Mediante Spinners se deberá escoger: una universidad, una facultad, una carrera y un paralelo, anteriormente creados. En otro Spinner se listarán las actividades académicas disponibles para el paralelo seleccionado; esto para que el usuario pueda cargar la información mediante un botón de “Cargar o Buscar” y

posteriormente pueda realizar un proceso de Update o Eliminación. En la Figura 2.38 se muestra la interfaz del `CRUDActividadAcademica` almacenado en el archivo `activity_crudactividadAcademica.xml`.

Cuando un usuario crea una actividad académica para un paralelo en la parte de la información de la abstracción se deberá escoger una fecha obtenida por un `DataPickerDialog()`, como se muestra en la Figura 2.39. Las actividades académicas se listarán en un ListView, al igual que en la sección 2.2.6.1 se listó a los estudiantes.

En la parte del cliente el objeto ProtocoloDatos es encapsulado y enviado al servidor para que este según su ID procese la petición requerida, siguiendo la misma lógica que se utilizó en la sección anterior.

```
902     public void LoadEstudianteParalelo(final ProtocoloDatos objIN, final Context context) {
903         //Cargo la info de Facultad- CArrera en CRUDCarrera
904         final Handler handler = new Handler();
905         Thread thread = new Thread((Runnable) () → {
908                 final String [] msg=objIN.GetString();
909                 try {
910                     Socket s = new Socket(ip,puerto);
911                     OutputStream out = s.getOutputStream();
912                     PrintWriter output = new PrintWriter(out);
913                     output.print(Arrays.toString(msg));
914                     output.flush();
915                     BufferedReader input = new BufferedReader(new InputStreamReader(s.getInputStream()));
916                     final String st = input.readLine();
917                     handler.post(new Runnable() {
918                         @Override
919                         public void run() {
920                             ListView lstEstudiatesParalelo =
921                                     ((Activity)context).findViewById(R.id.lstEstudiantesParalelo);
922                             String[] valores = st.split( regex: ", ");
```

Segmento de Código 2.24 Método LoadEstudianteParalelo Parte I

```
923                             ArrayAdapter<String> adapter = new ArrayAdapter<>
924                                     (context,android.R.layout.simple_expandable_list_item_1,valores) {
925                                 @Override
926                                 public View getView(int position, View convertView, ViewGroup parent) {
927
928                                     View view = super.getView(position, convertView, parent);
929                                     ViewGroup.LayoutParams parametros = view.getLayoutParams();
930                                     parametros.height = 100;
931                                     view.setLayoutParams(parametros);
932                                     return view;
933                                 }
934                             };
935                             lstEstudiatesParalelo.setAdapter(adapter);
936                         }
937                     });
938                     output.close();
939                     out.close();
940                     s.close();
941                 } catch (IOException e) {
942                     e.printStackTrace();
943                 }
944         });
946
947         thread.start();
948     }
```

Segmento de Código 2.25 Método LoadEstudianteParalelo Parte II

Figura 2.38 CRUD Actividad Académica

Figura 2.39 Date Picker Dialog

2.2.7 SPRINT 7

En el SPRINT 7 llamado “Entrenar Sistema”, se han desarrollado las interfaces de usuario y la parte lógica de programación para el entrenamiento del sistema con fotografías de estudiantes en el cliente y el servidor. Las fotografías se las almacena en carpetas en el servidor, organizadas por los números de cédula de los estudiantes.

2.2.7.1 Adición de fotografías de estudiantes al servidor FTP.

En el servidor se almacena las fotografías de estudiantes en carpetas, las cuales el cliente las enviará conectándose al servidor FTP. A estas carpetas se las ha llamado base de imágenes.

En el cliente se recolectarán las imágenes para que sean enviadas al servidor por esto en la Figura 2.36 se puede apreciar que existe el botón “CARGAR IMAGEN”, la lógica de este botón se muestra en el Segmento de Código 2.26. Para cargar la imagen en el servidor se dispone de dos opciones, la primera es tomar una fotografía con la cámara del dispositivo y la segunda es cargar una imagen ya existente desde la galería de imágenes. En la línea de código 330 se crea un `AlertDialog` (ver Figura 2.40) para que nos muestre estas opciones, en la línea 331 se define el título con el método `setTitle()`, y en la línea 332 se define los Items que se incluirán en el `AlertDialog` y el evento al cual responderá que en este caso es `Dialoginterface.OnClickListener()`. En las líneas 335 hasta la 347 se define el evento `OnClick()`, con el cual al momento que el usuario selecciona la opción “Tomar Fotografía” ejecutará la método `AbrirCamara()`; al seleccionar la opción “Seleccionar Fotografía” mediante un `Intent` se abrirá la galería de imágenes; y si selecciona la opción “Cancelar” se cerrará el `AlertDialog`.

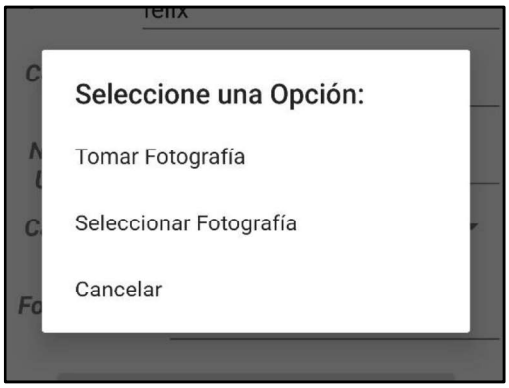

Figura 2.40 AlertDialog

```
326     case R.id.btnCargarImagen:
327         //lanzo los activities de la camara o la galeria
328         final CharSequence [] opciones = {"Tomar Fotografía",
329                 "Seleccionar Fotografía", "Cancelar"};
330         final AlertDialog.Builder builder = new AlertDialog.Builder( context: this);
331         builder.setTitle("Seleccione una Opción: ");
332         builder.setItems(opciones, new DialogInterface.OnClickListener() {
333             @Override
334             // Creo la logica para abrir la camara o la galeria del dispositivo movil
335             public void onClick(DialogInterface dialogInterface, int seleccion) {
336                 if(opciones[seleccion]=="Tomar Fotografía"){
337                     AbrirCamara();
338                 } else if (opciones[seleccion]=="Seleccionar Fotografía"){
339                     Intent intent = new Intent(Intent.ACTION_PICK,
340                             MediaStore.Images.Media.EXTERNAL_CONTENT_URI);
341                     intent.setType("image/*");
342                     startActivityForResult(intent.createChooser(intent,
343                             title: "Selecione APP De Imagen: "), SELECT_PICTURE);
344                 } else if (opciones[seleccion]=="Cancelar") {
345                     dialogInterface.dismiss();
346                 }
347             }
348         });
349         builder.show();
350         break;
```

Segmento de Código 2.26 Lógica del botón `btnCargarImagen`

Para poder trabajar con la cámara del dispositivo y la galería de imágenes, primero se debe dar ciertos permisos a la aplicación en el archivo `AndroidManifest.xml`, en el Segmento de Código 2.27 se muestran los permisos necesarios que son:

- `android.permission.CAMERA` (ver línea 10)
- `android.permission.READ_EXTERNAL_STORAGE` (ver línea 9)
- `android.permission.WRITE_EXTERNAL_STORAGE` (ver línea 8)
- `android.permission.USE_CREDENTIALS` (ver línea 11)

Con estos permisos se podrá acceder a las funciones mediante `Intents`.

```
8     <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
9     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
10    <uses-permission android:name="android.permission.CAMERA" />
11    <uses-permission android:name="android.permission.USE_CREDENTIALS" />
```

Segmento de Código 2.27 Permisos `AndroidManifest`

Para tomar la fotografía del estudiante con la cámara se ejecuta el método `AbrirCamara()`, que se lo muestra en el Segmento de Código 2.28. En la línea 371 se crea un `Intent` el cual con la propiedad `MediaStore.ACTION_IMAGE_CAPTURE`, abrirá la cámara fotográfica y capturará la imagen. Para esto, en la línea 372 se crea un archivo vacío, al cual se le asigna una ruta de la carpeta donde debe ser almacenada la imagen capturada, en las líneas 375

a la 382 se verifica si la ruta de la carpeta ya existe, mediante un `if()` y utilizando el método `exists()` que retornará un booleano; si el resultado es un `false`, se crea la ruta especificada para el archivo; y si el `if()` da como resultado un `true`, se crea el nombre de la imagen con la extensión `jpg`. Una vez que ya nos aseguramos que la ruta y el nombre de la imagen existan, en la línea 384 creamos un archivo con la información ya obtenida del path completo, en las líneas 385 y 386 se obtiene el URI [10] del archivo creado mediante la función `FileProvider.getUriForFile()`. Esto se lo hace ya que en la línea 388 para almacenar la fotografía resultante del Intent se debe especificar el URI de la imagen. Finalmente, en la línea 389 se lanza el Intent, creando una constante para saber que se utilizó la cámara y se espera una respuesta (ver línea 389).

Para obtener una fotografía de la galería de imágenes del dispositivo creo un `Intent` con la acción `ACTION_PICK`, para capturar imágenes mediante su URI desde el almacenamiento del dispositivo. Finalmente se deberá lanzar el `Intent` con una constante que servirá como identificador, mediante la función `startActivityForResult()`. La lógica de programación está especificada en el Segmento de Código 2.26 en las líneas 338 a la 343.

Una vez que se ha obtenido la imagen se la procede a enviar al servidor FTP, para esto se utiliza el método `onActivityResult`(), que se ejecuta al guardar la fotografía en el dispositivo. En el Segmento de Código 2.29 se muestra la lógica utilizada. Para ambos casos solo se debe obtener la ruta de la imagen, para que en la función `CargarImagen`() se conecte con el servidor y se la envíe.

Para enviar la fotografía al servidor FTP, se utiliza el método `CargarImagen`(), mostrado en el Segmento de Código 2.30. Para esto en la línea 411 se crea el objeto FTP, en la línea 413 se conecta al servidor utilizando sus credenciales de Logueo en Windows, en la línea 417 se ingresa a la ruta de la carpeta donde se almacenará la imagen, dicha ruta se la obtiene de la información cargada del estudiante de la base de datos, si no existiera dicha ruta en el servidor, la función `cwd`() la creará. En la línea 419 se almacena la imagen con la función `stor`() y finalmente en la línea 421 se cierra la conexión.

[10] URI: cadena de caracteres que se utiliza para ubicar recursos.

```
369     private void AbrirCamara() {
370         // abre la camara del dispositivo
371         Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
372         File file = new File(RUTA_IMAGEN);
373         String nombreImagen="";
374         String pathImagenFinal="";
375         boolean directorioCreado = file.exists();
376
377         if (!directorioCreado){
378             directorioCreado= file.mkdirs();
379         }
380         if (directorioCreado){
381             nombreImagen = ObtenerFechaNombre()+".jpg";
382         }
383         pathImagenFinal=RUTA_IMAGEN+ File.separator + nombreImagen;
384         File imagen = new File(pathImagenFinal);
385         Uri uri = FileProvider.getUriForFile( context: this,
386                 authority: this.getApplicationContext().getPackageName() + ".provider", imagen);
387         pathImagenBtnCargar=pathImagenFinal;
388         intent.putExtra(android.provider.MediaStore.EXTRA_OUTPUT, uri);
389         startActivityForResult(intent, PHOTO_CODE);
390     }
```

Segmento de Código 2.28 Método `AbrirCamara()`

```
392     @Override
393     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
394         // metodo que me permite envar las imagenes depsues de que se tomo o se selecciono
395         super.onActivityResult(requestCode, resultCode, data);
396         if(resultCode == RESULT_OK && requestCode == SELECT_PICTURE){
397             imageURI = data.getData();
398             CargarImagen(getRealPathFromURI(imageURI));
399
400         } else  if(resultCode == RESULT_OK && requestCode == PHOTO_CODE){
401             CargarImagen(pathImagenBtnCargar);
402
403         }
404
405     }
```

Segmento de Código 2.29 Método `onActivityResult()`

```
408     public void CargarImagen(String path){
409         // Método que envia la imagen al servidor FTP
410         try {
411             SimpleFTP ftp = new SimpleFTP();
412             // CoNexio al servidor FTP con el puerto 21.
413             ftp.connect(cabeceraConexion.getIp(),  i: 21,  s1: "LEO",  s2: "sunshine");
414             ftp.bin();
415             // CAMBIO EL DIRECTORIO DONDE SE ALAMCENARÁ LA IMAGEN, SI EL PATH NO EXISTE
416             // SE LO CREA
417             ftp.cwd(txtPathForografia.getText().toString());
418             // Cargo la imagen
419             ftp.stor(new File(path));
420             //Cierro la conexion
421             ftp.disconnect();
422         }
423         catch (IOException e) {
424             Toast.makeText( context: this,  text: "Error  FTP: " + e, Toast.LENGTH_SHORT).show();
425         }
426
427     }
```

Segmento de Código 2.30 Método `CargarImagen()`

Para recibir las imágenes se creó un servidor FTP (File Transfer Protocol) con las credenciales de Logueo de Windows (Usuario y Password). Actualmente Windows 10 permite crear un FTP para que clientes almacenen archivos de forma centralizada. Esto remplaza al protocolo SMB[11] (Server Message Block), pero el manejo será similar ya que se utiliza carpetas compartidas. Para este sistema prototipo se levantó el servidor FTP con las características mostradas en la Figura 2.41 y su nombre será "Tesis_ReconocimientoFacial", el puerto de conexión será el 21, el arranque de este será automático, estará codificado mediante UTF-8 y la ruta de acceso física será "C:\Tesis_ReconocimientoFacial".

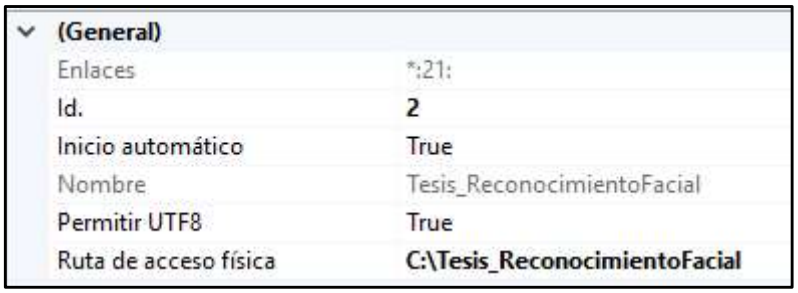

Figura 2.41 Características del Servidor FTP

El servidor FTP se lo utilizará para almacenar las fotografías de estudiantes (llamado base de imágenes), las fotografías de las actividades académicas registradas y los reportes de asistencia generados.

En la Figura 2.42 se muestra la organización en carpetas de la base de imágenes de estudiantes, la ruta de acceso es "C:\Tesis_ReconocimientoFacial\ImagenesEstudiantes" y cada estudiante creado tendrá una carpeta identificada por su número de cédula donde se almacenan todas sus fotografías.

[11] SMB: Protocolo para el uso de carpetas compartidas en Windows.

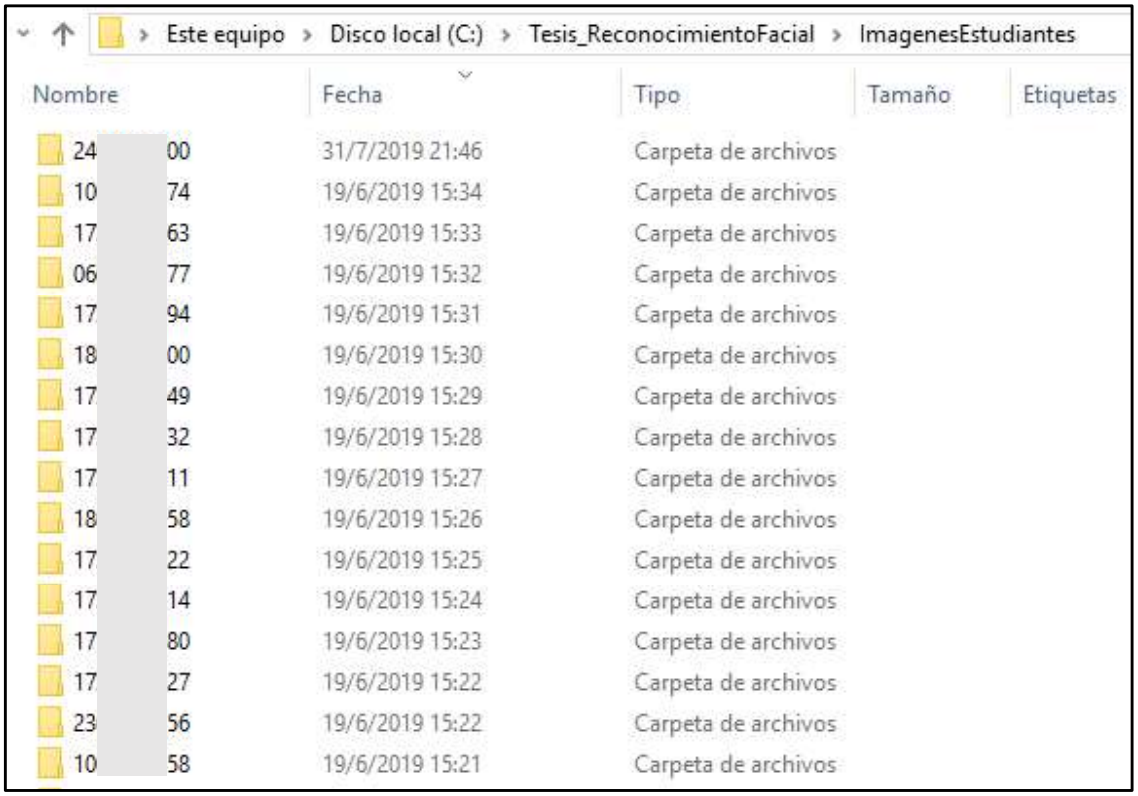

Figura 2.42 Servidor FTP (base de imágenes)

2.2.7.2 Entrenamiento del sistema por estudiante

Para que el sistema pueda reconocer el rostro de los estudiantes, debe realizar un entrenamiento con fotografías de los mismos, para esto en el CRUD Estudiante al momento de cargar la información, se activa un botón para el Entrenamiento del Sistema por Estudiante. En la Figura 2.36 se puede apreciar la interfaz donde se muestra el botón de entrenamiento.

El cliente apoyándose del objeto `ProtocoloDatos` y a través del método `SendEntrenamiento` del objeto `ConexionSocket`, enviará un id para el entrenamiento y el número de cédula del estudiante al servidor, con esto se podrá ubicar la carpeta con las fotografías a entrenar.

Adicionalmente cuando un entrenamiento se está realizando, el servidor impide que un usuario pueda acceder al módulo Tomar Asistencia, para esto se utiliza la Tabla `banderasentrenamientos` de la base de datos, la cual tiene un campo que varía entre 1 y 0, con esto se puede realizar una verificación y bloquear el acceso de ser necesario.

En el Segmento de Código 2.31, el servidor al recibir la trama con el ID 144, interpretará que debe realizar un entrenamiento por estudiante, para esto leerá el número de cédula

recibido y creará un subproceso (ver línea 1537) ejecutando el archivo `ModuloEntrenamientoRf.py`. Con el subproceso, se evita que el prototipo se bloquee cuando se entrena.

```
1529    #Entrenamiento Sistema
1530        #Protocolo.objeto1 == Lista cedulaEstudiantes
1531            if Protocolo.idprocolo == "144":
1532                # Entrenamineto por estudiante, se crea un subpreceso.
1533                mDB.db.close()
1534                mDB.db.connect()
1535                print("Entrenamiento Estudiantes Parcial")
1536                print("estudiante a Entrenar : " + Protocolo.objeto1)
1537                subProceso = subprocess.Popen(["python", "ModuloEntrenamientoRF.py",
1538                                               Protocolo.objeto1] )
1539                mDB.db.close()
1540                return "None"
```

Segmento de Código 2.31 Ejecución del subproceso de Entrenamiento

El Script "`ModuloEntrenamientoRF.py`" inicialmente utilizará las librerías mostradas en el Segmento de Código 2.32. Las principales librerías serán `cv2`, la cual permitirá el uso de OpenCV; la librería `os`, que permitirá manipular las rutas de acceso a la base de datos de imágenes y `numpy`, la cual manipula los datos en forma de matrices.

```
1  import cv2
2  import os
3  import numpy as np
4  import random
5  import peewee as pw
6  import ConexionDB as cDB
7  import ModeloDB as mDB
8  import sys
9  import array as ar
10 from PIL import Image
```

Segmento de Código 2.32 Librerías de `ModuloEntrenamientoRF.py`

En el Segmento de Código 2.33, cuando se inicia mediante un `Main` el subproceso para entrenar estudiantes, primero se cambiará las banderas para bloquear la Toma de Asistencia, esto se lo realizará mediante una consulta `Update` (ver línea 66) a la base de datos. Posteriormente mediante un `if` se verificará que la bandera sea igual a 1 (ver línea 72), y se procederá con el entrenamiento. Primero se recibirá un String con los números de cédula de los estudiantes a entrenar y se la transformará a una lista (ver línea 74), luego en la línea 76 ejecutará la función `EntrenamientoEstudiantes()` que recibirá como parámetro de entrada la lista ya mencionada. Una vez finalizado el entrenamiento, se cambia el estado de la bandera a cero (ver línea 77) y se termina el subproceso.

```
62  if __name__ == '__main__':
63      #Recivo un str con todos las cedulas de estudiantes a entrenar
64      mDB.db.close()
65      mDB.db.connect()
66      bandera= mDB.banderasentrenamientos.update(
67          {mDB.banderasentrenamientos.banderaEntrenamiento:0}).where(
68          mDB.banderasentrenamientos.idBanderaEntrenamiento==1)
69      rowUpdate=bandera.execute()
70      print("Filas alteradas Inicio: " + str(rowUpdate))
71      mDB.db.close()
72      if rowUpdate > 0:
73          listaEdtudiantes=[]
74          listaEdtudiantes=(sys.argv[1].split(" - "))
75          try:
76              EntrenamientoEstudiantes(listaEdtudiantes)
77              bandera= mDB.banderasentrenamientos.update(
78                  {mDB.banderasentrenamientos.banderaEntrenamiento:1}).where(
79                  mDB.banderasentrenamientos.idBanderaEntrenamiento==1)
80              rowUpdate=bandera.execute()
81              print("Bandera alterada FINAL: " + str(rowUpdate))
82              print("------SE HA TERMINADO EL ENTRENAMIENTO----------")
83              mDB.db.close()
```

Segmento de Código 2.33 Main del subproceso de entrenamiento

En el Segmento de Código 2.34 y 2.35 se muestra la función `EntrenamientoEstudiantes()`, la cual recibe como parámetro de entrada la lista de estudiantes a entrenar. En dicha función:

- Primero se deberá cargar las rutas en las que se almacenan los estudiantes, para esto se carga el archivo "`PathReconocimientoFacial.txt`" (ver líneas 18 y 19).
- Segundo se realiza el entrenamiento por estudiante, para esto se aplica a una función `for` (ver línea 23) a la lista recibida, en este caso solo recibirá un estudiante, pero posteriormente en el entrenamiento total del sistema, se recibirán a más estudiantes.
- Tercero en la línea 27, se cargará la ruta de la carpeta de la base de datos de imágenes del estudiante, y en la línea 29 se listará todos los archivos contenidos en la misma y se los almacenará en una lista llamada `lstImagenesEstudiantes.`
- Cuarto, a la lista de archivos se las irá cargando una por una, en busca de las imágenes de los estudiantes para el entrenamiento, para esto se utiliza un `for`. En la línea 33, se puede apreciar cómo se lee la imagen con la función `cv2.imread(image_path)`, que tiene como parámetro de entrada la ruta del archivo.
- Quinto, se verifica que el archivo cargado sea una imagen y en las líneas 35, 36 y 37 se transforma la imagen a escala de grises ya que es un requisito de la función proporcionada por OpenCV para el entrenamiento.

- Sexto, si la imagen ya se encuentra en escala de grises en la línea 39 a 42, se procede a ejecutar la función para encontrar el rostro del estudiante en la imagen, dicha función se llama `detect_face()` y recibe como parámetro de entrada el Clasificador en Cascada que se utilizará y la imagen a entrenar. En este caso se utilizará el clasificador Haar Cascade, como se mencionó en la metodología. Esta función retornará una matriz con los pixeles del rostro encontrado y las coordenadas que se deben utilizar para encontrar dicho rostro en la imagen original, las variables son `face y rect.`
- Séptimo, una vez obtenida la matriz y el id del estudiante, se continua con las demás imágenes. Una vez finalizado se guarda la información de rostros e identificadores en los archivos `FaceTraining.npy` y `LabelTraining.npy` (ver líneas 49 y 50). Con esto al momento de realizar un reconocimiento facial, no se necesitará entrenar al sistema nuevamente, sino solo leer dichos archivos. Cabe recalcar que existirá un archivo rostros e identificadores por estudiante.

La función `detect_face()` permite realizar la detección del rostro de una persona, como se indicó en párrafos anteriores, se utilizará el algoritmo Haar Cascade. En el Segmento de Código 2.36 se muestra dicha función, en la cual:

- Primero, en la línea 53 se convierte a la imagen en escala de grises.
- Segundo, en la línea 54 se ejecuta la detección de rostros mediante la función `f_cascade.detecMultiScale()`, la cual recibirá como parámetros de entrada la imagen en escala de grises, un factor de re escalamiento de la imagen, la cantidad de pixeles vecinos que tomaré en cuenta para la comparación Haar, y un tamaño mínimo de imagen aceptada para evitar el pixelado. Los valores a utilizar en estos parámetros se los describirá en la sección de resultados, ya que es necesario ajustarlos hasta encontrar un funcionamiento óptimo. Finalmente, esta función retornará las coordenadas de la imagen que contienen al rostro del estudiante.
- Tercero, en la línea 59 se recorta la imagen con las coordenadas obtenidas anteriormente.
- Cuarto, se retornará la matriz con el rostro del estudiante y las coordenadas para ubicarlo.

```
#entrenamiento
def EntrenamientoEstudiantes(idEstudiantes):
    #entreno al sistema con fotogrfias de los alumnos segun su ID,
    #al final genero un archivo .np para lectura de datos
    #idEstudiantes=Cedula
    fileRF = open("PathReconocimientoFacial.txt","r")
    argRF = fileRF.readlines()
    faces = []
    labels = []
    print(idEstudiantes)
    for idEstudiante in idEstudiantes:
        faces = []
        labels = []
        label = (idEstudiante)
        direccionEstudiante = argRF[2] + "\\" + idEstudiante
        print("Dirección Estudiante: "+direccionEstudiante)
        lstImagenesEstudiantes = os.listdir(direccionEstudiante)
        aux=0
```

Segmento de Código 2.34 Función `EntrenamientoEstudiantes()`. Parte I

```
    for image_name in lstImagenesEstudiantes:
        image_path = direccionEstudiante + "\\" + image_name
        image = cv2.imread(image_path)
        if image is not None:
            imageInfo = Image.open(image_path)
            if (imageInfo.mode == "L"):
                face = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            else:
                face, rect = detect_face(
                    cv2.CascadeClassifier(
                        r'C:\opencv\sources\data\haarcascades\haarcascade_frontalface_default.xml'),
                    image)
            if face is not None:
                cv2.imwrite("train\\t"+ label+"-"+str(aux)+".jpg",face)
                face = cv2.resize(face,(240,240))
                faces.append(face)
                labels.append(label)
                aux = aux + 1
    np.save(argRF[2] + "\\" + idEstudiante+"\\FaceTraining.npy",faces)
    np.save(argRF[2] + "\\" + idEstudiante+"\\LabelTraining.npy",labels)
fileRF.close()
```

Segmento de Código 2.35 Función `EntrenamientoEstudiantes()`. Parte II

```
def detect_face(f_cascade, colored_img, scaleFactor = 1.3):
    #  Deteccion de las imagenes de entrenamiento
    img_copy = colored_img.copy()
    gray = cv2.cvtColor(img_copy, cv2.COLOR_BGR2GRAY)
    faces = f_cascade.detectMultiScale( gray,scaleFactor, minNeighbors=4,minSize=(190, 190))
    if (len(faces) == 0):
        return None, None
    auxFor= 1
    for (x, y, w, h) in faces:
        cv2.rectangle(img_copy, (x, y), (x+w, y+h), (0, 255, 0), 2)
    return gray[y:y+w, x:x+h], faces
```

Segmento de Código 2.36 Función `detect_face()`

2.2.7.3 Entrenamiento total del sistema

Para el entrenamiento total del sistema se debe ingresar al cliente como un usuario administrador y en el "`activity_menu_principal.xml`" (ver Figura 2.43) del menú principal se mostrará un botón llamado "Entrenamiento Total Del Sistema", el cual al presionarlo enviará al servidor la petición de entrenamiento (con id 147), mediante la función `SendEntrenamiento`() del objeto `ConexionSocket`.

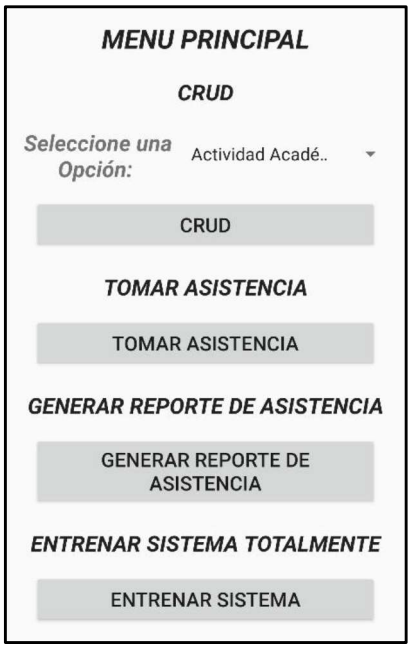

Figura 2.43 Menú Principal

Para mostrar el botón de entrenamiento, se debe restringir la vista de los componentes según el usuario logueado, en el Segmento de Código 2.37 se muestra cómo se los restringe, para esto se utiliza el objeto `cabeceraConexion`, el cual contiene qué tipo de usuario que está en el sistema actualmente.

En el servidor se analiza la trama en la cual solo se recibe el id 147, posteriormente se obtiene mediante una consulta todos los números de cédula de los estudiantes registrados en el sistema y se los envía como parámetro de entrada en el subproceso `ModuloEntrenamientoRF`. En el Segmento de Código 2.38 se muestra dicho proceso.

Para el entrenamiento del sistema, se utiliza el mismo procedimiento mostrado en la sección 2.2.7.2, por lo tanto, no se lo explicará de nuevo.

```
45      //CArgo Activity segun perfil logueado
46      if (cabeceraConexion.getId().equals("0")){
47          if (cabeceraConexion.getPerfilUsuario().equals("Administrador")){
48              //Cargo El spinner para CRUD
49              txtSeleccionCRUD = findViewById(R.id.txtSeleccionCRUD);
50              ArrayAdapter <CharSequence> adapterSpinner = ArrayAdapter.createFromResource( context: this,
51                      R.array.tipoCRUDAdmin, android.R.layout.simple_spinner_item);
52              adapterSpinner.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
53              txtSeleccionCRUD.setAdapter(adapterSpinner);
54              seccionTomarAsistencia.setVisibility(View.VISIBLE);
55              lytEntrenarSistema.setVisibility(View.VISIBLE);
56
57
58          } else if (cabeceraConexion.getPerfilUsuario().equals("Profesor")){
59              //Cargo El spinner para CRUD
60              txtSeleccionCRUD = findViewById(R.id.txtSeleccionCRUD);
61              ArrayAdapter <CharSequence> adapterSpinner = ArrayAdapter.createFromResource( context: this,
62                      R.array.tipoCRUDProfesor, android.R.layout.simple_spinner_item);
63              adapterSpinner.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
64              txtSeleccionCRUD.setAdapter(adapterSpinner);
65              seccionTomarAsistencia.setVisibility(View.VISIBLE); //Visible
66              Toast.makeText( context: this,  text: "Profesor", Toast.LENGTH_SHORT).show();
67              lytEntrenarSistema.setVisibility(View.INVISIBLE);
68          }
```

Segmento de Código 2.37 Restricción de componentes según usuario Logueado

```
1541  if Protocolo.idprocolo == "147":
1542      mDB.db.close()
1543      mDB.db.connect()
1544      print("Entrenamiento Estudiantes Total")
1545      estudiantesTotal = mDB.estudiantes.select()
1546      listaCedulasEstdudiantes=[]
1547      if (len(estudiantesTotal)>0):
1548          for auxEstudiante in estudiantesTotal:
1549              listaCedulasEstdudiantes.append(str(auxEstudiante.cedula))
1550          strEnvio= []
1551          strEnvio = " - ".join(listaCedulasEstdudiantes)
1552          print("Estudiantes a entrenar")
1553          print(strEnvio)
1554          subProceso = subprocess.Popen(["python", "ModuloEntrenamientoRF.py", strEnvio] )
1555          mDB.db.close()
1556      return "None"
```

Segmento de Código 2.38 Entrenamiento total del sistema

2.2.8 SPRINT 8

En el SPRINT 8 llamado “Tomar Asistencia”, se han desarrollado las interfaces de usuario y la parte lógica de programación para el registro de asistencia de los estudiantes para cada actividad académica. Este Sprint se lo dividió en dos partes, la primera será la toma de asistencia mediante fotografías de los estudiantes presentes; la segunda será un registro manual para dar redundancia al sistema y evitar errores.

Cabe recalcar que un usuario profesor solo podrá tomar asistencia de los paralelos a los que ha sido asignado, pero el usuario administrador podrá hacerlo de cualquier paralelo.

2.2.8.1 Tomar asistencia a estudiantes

En la Figura 2.44 se muestra la interfaz gráfica del archivo "`activity_tomar_asistencia.xml`" para la toma de asistencia de estudiantes. Como requisito previo se debe tener creada una actividad académica y estudiantes asignados a un paralelo. Un usuario al seleccionar un paralelo, una actividad académica y posteriormente dar un tap en el botón "`Tomar Fotografía Paralelo`" podrá enviar al servidor las fotografías de los estudiantes presentes, no hay un límite de fotografías a enviar.

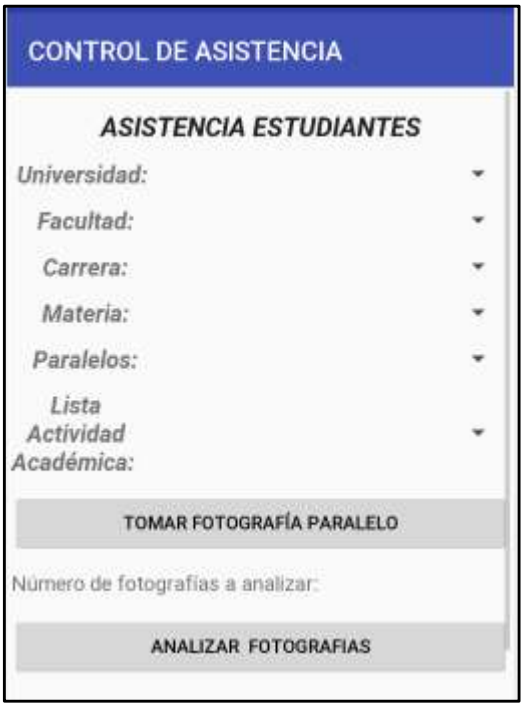

Figura 2.44 Interfaz Asistencia Estudiantes

El procedimiento del envío de fotografías es el mismo que el mostrado en la sección 2.2.7.1 por lo que no se lo explicará nuevamente. La única diferencia es que se almacenarán las fotografías de las actividades académicas en la carpeta "`/ImagenesAsistencia/IdActividadAcademica`" del servidor FTP, cada actividad académica estará identificada por el ID de la misma.

Una vez cargadas las imágenes de los estudiantes presentes se puede analizarlas, esto se lo realiza presionando el botón llamado "`Analizar Fotografías`". Al servidor se le envía el ID 142 y el id de la actividad académica, en el objeto `ProtocoloDatos` que posteriormente se lo convierte en String. Para el envío se utiliza el método `LoadEstudiantesTomarAsistencia()` del objeto `ConexionSocket`.

El método `LoadEstudiantesTomarAsistencia()` esperará que el servidor responda con la lista de estudiantes presentes para que posteriormente se los cargue en la lista del paralelo y se pueda modificar la asistencia manualmente, este proceso se lo explicará en la sección 2.2.8.2.

En el servidor se ejecutará el script llamado "`ModuloReconocimientoFacial.py`". Inicialmente se ejecuta la función `HiloReconocimientoFacial()`, que recibe como parámetro de entrada el Id de la actividad académica a analizar, dicha función se muestra en los segmentos de código 2.39 y 2.40. El procedimiento que sigue es el siguiente:

- Primero, en las líneas 149 y 150 se carga la información de la ruta donde se encuentran las carpetas de las actividades académicas en la base de imágenes. Para esto se carga el archivo "`PathReconocimientoFacial.txt`".
- Segundo, en la línea 151 se obtiene el path de las imágenes que se encuentran en la carpeta de la actividad académica y se los almacena en una lista llamada `listaFotografiasActividadAcademica`.
- Tercero, en la línea 157 se obtiene los números de cédula de los estudiantes que pertenecen al paralelo que representa la actividad académica. Para esto se ejecuta el método `ObtenerEstudiantes()` que realiza una consulta SELECT JOIN a la base de datos. En el Segmento de Código 2.41 se muestra dicha consulta.
- Cuarto, en la línea 158 se ejecuta la función `prepare_training_data()` que recibirá la lista de estudiantes obtenida anteriormente y cargará los archivos de entrenamiento de los estudiantes. Retornará una matriz con los rostros de los estudiantes y otra matriz con sus identificadores.
- Quinto, en la línea 165 se crea un objeto llamado `face_recognizer` a través de la función `cv2.face.LBPHFaceRecognizer_create()`; este objeto permitirá realizar el reconocimiento facial, mediante la comparación de las imágenes de entrenamiento con las de la actividad académica.
- Sexto, en la línea 168 se realiza el entrenamiento de la función con los estudiantes del paralelo a analizar. Se utiliza la función `train()`.
- Séptimo, en las líneas 169 a la 173, mediante un lazo `for` se realiza el reconocimiento facial fotografía por fotografía, para esto se utiliza la función `AnalizarFotografiasActividadAcademica()`, que recibirá como parámetro de entrada el path de la fotografía, el objeto `face_recognizer` y el id de la actividad académica. Al completar el reconocimiento facial, la función almacenará en la base de datos la lista de los estudiantes presentes.

- Octavo, una vez terminado el reconocimiento facial, el servidor obtendrá la lista de los estudiantes (ver líneas 173 a la 184) presentes y la retornará al cliente para que pueda modificarla manualmente.

```
147 def HiloReconocimientoFacial(idActividadAcademica):
148     #Segun el id de actividad academica obtengo la lista de estuddiantes a analizar en un paralelo
149     fileRF = open("PathReconocimientoFacial.txt","r")
150     argRF = fileRF.readlines()
151     listaFotografiasActividadAcademica = ObtenerPathFotografiasActividadAcademica(idActividadAcademica)
152     if listaFotografiasActividadAcademica == "False":
153         print("No se encontraron fotografias")
154         #return "Error sin fotografias"
155     else :
156         print("Preparación de datos")
157         idEstudiantes= ObtenerEstudiantes(idActividadAcademica)
158         faces, labels = prepare_training_data(idEstudiantes)
159         print(labels)
160         print("Data prepared")
161         #print total faces and labels
162         print("Total faces: ", len(faces))
163         print("Total labels: ", len(labels))
164         # objeto pra crear el reconocimeinto LBPHF
165         face_recognizer = cv2.face.LBPHFaceRecognizer_create()
166         print("Proceso face_recognizer inicializado")
167         labels = list(map(int, labels))
168         face_recognizer.train(faces, np.array(labels))
169         for auxFotografia in  listaFotografiasActividadAcademica:
170             AnalizarFotografiasActividadAcademica(
171                 argRF[1].rstrip('\n')+ "\\" + idActividadAcademica+ "\\" +auxFotografia,
172                 face_recognizer,
173                 idActividadAcademica)
174     mDB.db.close()
```

Segmento de Código 2.39 Función `HiloReconocimientoFacial()`. Parte I

```
172 mDB.db.connect()
173 listaEstudiantePresentefinal = mDB.listasestudiantespresentes.select(mDB.listasestudiantespresentes,
174     mDB.estudiantes).join(mDB.estudiantes,
175     on=(mDB.estudiantes.idEstudiante==mDB.listasestudiantespresentes.idEstudiante)).where(
176     mDB.listasestudiantespresentes.idActividadAcademica == int(idActividadAcademica)).order_by(
177     mDB.estudiantes.apellido,mDB.estudiantes.nombre)
178 estudiantesPresentesreturn=[]
179 datosEstudiante=[]
180 for estudiantePresente in listaEstudiantePresentefinal:
181     datosEstudiante.append(str(
182         estudiantePresente.estudiantes.cedula) + " - " +
183         estudiantePresente.estudiantes.apellido+ " - " + estudiantePresente.estudiantes.nombre)
184 estudiantesPresentesreturn = ", ".join(datosEstudiante)
185 if (len(estudiantesPresentesreturn)==0):
186     return "ESTNOP"
187 print("Estudiante a retornar: \n")
188 fileRF.close()
189 mDB.db.close()
190 return estudiantesPresentesreturn
```

Segmento de Código 2.40 Función HiloReconocimientoFacial(). Parte II

```
110     estudiantesQuery = mDB.actividadesacademicas.select(mDB.paralelosestudiantes,
111         mDB.estudiantes).join(mDB.paralelosestudiantes,
112         on=(mDB.paralelosestudiantes.idParalelo==mDB.actividadesacademicas.idParalelo)).switch(
113         mDB.actividadesacademicas).join(mDB.estudiantes,
114         on=(mDB.estudiantes.idEstudiante==mDB.paralelosestudiantes.idEstudiante)).where(
115         mDB.actividadesacademicas.idActividadAcademica==idActividadAcademicaAux)
```

Segmento de Código 2.41 Consulta Select Join para obtener cédula de estudiantes pertenecientes a un paralelo

```
117 def AnalizarFotografiasActividadAcademica(pathFotografiaActividadAcademica, face_recognizer,
118     idActividadAcademica):
119     print("Prediccion de imagen Actividad Academica: " + pathFotografiaActividadAcademica)
120     idEstudiantePresente = []
121     #cargo imagen a analizar
122     test_img1 = cv2.imread(pathFotografiaActividadAcademica)
123     if test_img1 is not None:
124         cv2.imwrite('lastPhoto.jpg',test_img1)
125         predicted_img1, idEstudiantesPresentes = predict(test_img1, face_recognizer)
126         if predicted_img1 is not None:
127             print("Prediction complete")
128             # Al predecir a mayor valor menor la prescision de la imagen
129             AlmacenarEstudiantesPresentes(idEstudiantesPresentes, idActividadAcademica)
```

Segmento de Código 2.42 Función `AnalizarFotografiasActividadAcademica()`

La función `predict()` realiza el reconocimiento facial de las fotografías y se la muestra en el Segmento de Código 2.43, su proceso es el siguiente:

- Primero, en la línea 75 se realiza la detección de los rostros de la imagen a analizar, el procedimiento es el mismo que se mostró en la sección 2.2.7.2.
- Segundo, cada rostro detectado se debe comparar con el entrenamiento realizado anteriormente, para esto se utiliza la función `predict()` (ver línea 80) del objeto `face_recognizer` que se lo obtiene como parámetro de entrada. Esta función retornará un vector de dos valores, el primer valor corresponderá al identificador del estudiante con el que se encontró mayor coincidencia y el segundo valor corresponderá a la distancia de similitud calculada (mientras menor sea el valor, más exacta es la coincidencia).
- Tercero, para el vector resultado de la función `predict()` se debe realizar algunas comprobaciones, ya que el sistema debe reducir notoriamente los reconocimientos falsos, para esto se ha fijado que la distancia de coincidencia máxima permitida sea de 50 (ver línea 84). También se ha realizado una corrección a los identificadores retornados por la función, ya que no se considera números más grandes que un INT, en la sección de resultados y pruebas se lo analizará.
- Cuarto, en la línea 97 se retornará la imagen analizada y los números de cédula de los estudiantes presentes.

```
def predict(test_img,face_recognizer):
    face_recognizerAux = face_recognizer
    img = test_img.copy()
    face, rect = detect_facePredict(cv2.CascadeClassifier(r'C:\opencv\sources\data
    idEstudiantePresente =[]
    if face is not None:
        #Face son los rostros del la deteccion realizada anteriormente
        for x in face:
            label= face_recognizer.predict(x)
            print(label)
            #Agregar un if para controlar que el factor de ralcion
            #no sea mayor a 50(MUCHOS ERRORES).
            if label[1] <= 50.00:
                if (label[0] < 0):
                    # si predict me devuelve un int negativo,
                    #para transformar a Long Int, sumo 4294967296
                    auxLabel = int(label[0]) + 4294967296
                    idEstudiantePresente.append(auxLabel)
                else:
                    label_text=(label[0])
                    idEstudiantePresente.append(label_text)
            else :
                print("Estudiante con cedula: " +
                str(label[0]) + " posee indice de realación muy alto (> 50):" +
                str(label[1]) + " ; estudiante no tomado en cuenta")
        return img, idEstudiantePresente
```

Segmento de Código 2.43 Función `predict()`

2.2.8.2 Gestión de lista de estudiantes presentes

Cuando el servidor ha analizado las fotografías de los estudiantes, retornará la lista de estudiantes presentes al cliente para que se pueda gestionar manualmente las asistencias de los mismos, en caso de que se necesite realizar algún cambio. Para esto en el Layout "`activity_tomar_asistencia.xml`" se carga un ListView con un grupo de CheckBox's que representará a cada estudiante del paralelo y se seleccionará automáticamente a los presentes en la actividad académica. Con esto se puede realizar cambios al registro de asistencia y finalmente guardarlos dando clic en el botón "`Guardar Asistencia de Estudiantes`" enviando al servidor la trama con ID 146 y los números de cédula de la nueva lista de estudiantes presentes.

En la Figura 2.45 se muestra la interfaz para gestión manual de asistencia.

Figura 2.45 Interfaz Asistencia de Estudiantes con Registro Manual de Asistencia

2.2.9 SPRINT 9

En el Sprint 9 llamado “Generar reporte de asistencia”, se han desarrollado la interfaz de usuario y la parte lógica de programación para la creación del registro de asistencia de los estudiantes de un paralelo, tomando en cuenta las actividades académicas creadas. Para esto el cliente generará una solicitud al servidor y este creará una hoja en Excel con la información de las asistencias para el paralelo solicitado, luego la enviará vía mail.

Cabe recalcar que un usuario profesor solo podrá generar reportes de los paralelos a los que ha sido asignado, pero el usuario administrador podrá hacerlo de cualquier paralelo.

En el cliente la interfaz de usuario creada se muestra en la Figura 2.46 y está almacenada en el archivo activity_generar_reporte_asistencia.xml. Para que un usuario genere el reporte de asistencia, debe seleccionar una materia y un paralelo de los Spinner mostrados en la Figura, además deberá ingresar un mail valido para que se lo pueda enviar.

Cuando un usuario da clic en el botón “Generar Reporte”, el cliente envía una petición al servidor de la misma forma que ha enviado las demás peticiones, es decir mediante una trama en forma de String, pero en este caso con ID 143 y anexando el paralelo seleccionado además del mail ingresado.

Figura 2.46 Interfaz Reporte de Asistencia

Al momento de enviar la trama al servidor se realiza una validación del mail ingresado; en el Segmento de Código 2.44 se muestra la función de validación de mail, la función retorna un `booleano` y se utiliza un objeto `Pattern` con su función `matcher().matches()` para obtener el resultado.

```
216     private boolean validarEmail(String email) {
217         Pattern pattern = Patterns.EMAIL_ADDRESS;
218         return pattern.matcher(email).matches();
219     }
```

Segmento de Código 2.44 Función validarMail()

La trama recibida por el servidor es procesada con el ID 143 por el objeto `Protocolo`, en el Segmento de Código 2.45 muestra las sentencias a ejecutar. En las líneas 1526 a la 1528 se ejecuta el subproceso "`GenerarReporteAsistencia.py`", con esto se evita el bloqueo del sistema y se permite atender otras solicitudes en paralelo. Como parámetro de entrada para este subproceso se envía el Id del paralelo y la dirección de mail a enviar el reporte.

```
1520  #GENERAR Y ENVIAR LISTA DE ASISTENCIA
1521      #Protocolo.objeto1 == idParalelo
1522      #Protocolo.objeto2 == mail
1523          if Protocolo.idprocolo == "143":
1524              mDB.db.close()
1525              print("Generar y enviar Lista de asistencia")
1526              subProceso = subprocess.Popen(["python",
1527                  "GenerarReporteAsistencia.py",
1528                  Protocolo.objeto1, Protocolo.objeto2] )
1529              return "None"
```

Segmento de Código 2.45 Ejecución de subproceso `GenerarReporteAsistencia.py`

En el script `GenerarReporteAsistencia.py` se utilizaron las librerías mostradas en el Segmento de Código 2.46, algunas de estas ya se las ha utilizado en otros scripts, pero la más importante es la librería `openyxl`, la cual proporciona los métodos y las herramientas para crear y manipular hojas de Excel.

```
1  import cv2
2  import os
3  import numpy as np
4  import base64
5  import peewee as pw
6  import ConexionDB as cDB
7  import ModeloDB as mDB
8  import sys
9  #libreria que permite manipular hojas de Excel
10 import openpyxl
11 from openpyxl.styles import Font, Color
12 from openpyxl import Workbook
13 from openpyxl.chart.shapes import GraphicalProperties
14 from openpyxl.drawing.line import LineProperties
15 from openpyxl.chart.axis import ChartLines
16 from openpyxl.chart.label import DataLabelList
17 # los métodos a emplearse con chart
18 from openpyxl.chart import (
19     LineChart,
20     BarChart,
21     Reference,
22     label
23 )
24 from openpyxl.chart.axis import DateAxis
25 import EnvioReporteMail as sendMail
```

Segmento de Código 2.46 Librerías del Script `GenerarReporteAsistencia.py`

El punto inicial de este Script es la función `main()`, mostrada en el Segmento de Código 2.47, su función es la siguiente:

- Primero, en la línea 201 se ejecuta el método `DuplicarPlantilla()` que copia la plantilla creada en Excel, con la cual se facilita la generación del reporte. En esta función se duplica el archivo "`PlantillaAsistencia.xlsx`" y se lo guarda en la carpeta "C:\Tesis_ReconocimientoFacial\ReportesAsistencia\" que es la ubicación del servidor FTP. Con esto se evita que la plantilla original sea alterada y genere

errores posteriormente. El archivo "PlantillaAsistencia.xlsx" se lo muestra en la Figura 2.47.

- Segundo, se ejecuta la función CargarPlantilla(),que recibirá como parámetro de entrada el id del Paralelo a generar el reporte. Esta función se encargará de acceder al archivo duplicado, llenar la información de la materia, el paralelo y el profesor; también cargará la información de la lista de estudiantes y sus asistencias según corresponda la actividad académica. Finalmente se realizará una suma de asistencias y se obtendrá un gráfico estadístico.
- Finalmente se ejecutará la función EnviarReporte() que obtendrá como parámetro de entrada el mail a enviar el archivo, la ruta del archivo y el nombre del mismo. Esta función ejecutará el script sendMail.EnviarReporte(), que se lo explicará más adelante.

```
200 if __name__ == '__main__':
201     #main del subproceso
202     DuplicarPlantilla()
203     rutaArchivo,nombreArchivo = CargarPlantilla(int(sys.argv[1]))
204     EnviarReporte(sys.argv[2],rutaArchivo,nombreArchivo)
205
```

Segmento de Código 2.47 Función Main del subproceso GenerarReporteAsistencia.py

	A	B	C	D
1	CONTROL DE ASISTENCIAS EPN			
2				
3	Materia:			
4	Paralelo:			
5	Profesor:			
6				
7		Estudiantes		Fechas
8	Cédula	Apellido	Nombre	
9				

Figura 2.47 PlantillaAsistencia.xlsx

La función CargarPlantilla() como se explicaba anteriormente generará el reporte con la información solicitada, para esto se realizará la siguientes acciones:

a) En el Segmento de Código 2.48, en las líneas 37 y 38 abrirá el archivo y se situará en la Hoja 1. Luego en las líneas 42 a la 49, mediante una consulta anidada a la

base de datos obtendrá la información de las fechas de las actividades académica, el nombre del profesor y la materia del paralelo solicitado.

```
36  def CargarPlantilla(idParaleloIn):
37      excel_document = openpyxl.load_workbook(r'C:\Tesis_ReconocimientoFacial\ReportesAsistencia\test.xlsx')
38      hojaAsistencia = excel_document['Hoja1']
39      mDB.db.close()
40      print("Busqueda fechas")
41      mDB.db.connect()
42      fechas= mDB.actividadesacademicas.select(mDB.paralelos,mDB.materias,
43          mDB.actividadesacademicas,mDB.profesores).join(mDB.paralelos,
44          on=(mDB.actividadesacademicas.idParalelo==mDB.paralelos.idParalelo)).switch(
45          mDB.actividadesacademicas).join(mDB.materias,
46          on=(mDB.materias.idMateria==mDB.paralelos.idMaterias)).switch(
47          mDB.actividadesacademicas).join(mDB.profesores,
48          on=(mDB.paralelos.idProfesor==mDB.profesores.idProfesor)).where(
49          mDB.paralelos.idParalelo==idParaleloIn).order_by(mDB.actividadesacademicas.fecha)
```

Segmento de Código 2.48 `GenerarReporteAsistencia`(). Parte I

b) En el Segmento de Código 2.49, en las líneas 57 a la 59 se escribe la información de la materia, el paralelo y el profesor en las celdas B3, B4 y B5 de la Hoja1. Luego se escribirá la información de las fechas de las actividades académicas empezando en la celda D8 hacia la derecha, según sea necesario.

```
57      hojaAsistencia['B3']= fechas[0].materias.nombreMateria
58      hojaAsistencia['B4']= fechas[0].paralelos.nombreParalelo
59      hojaAsistencia['B5']= (fechas[0].profesores.nombre + " " + fechas[0].profesores.apellido)
60
61      print("numero fechas: " + str(len(fechas)))
62      mDB.db.close()
63      # escribo las fechas de cada actividad academica
64      for x in range(len(fechas)):
65          y = x+4
66          # escribo fecha
67          hojaAsistencia.cell(row=8,column=y).value= fechas[x].fecha
68          hojaAsistencia.cell(row=8,column=y).font= Font(bold=True)
69          #aumento una columna, se inserta la columna antes del valor especiicado
70          hojaAsistencia.insert_cols(y+1)
71      # sumo 4 porque tengo tres columnas de encabezado antes
72      #y una para que no sobre escriba con el ultimo
73      hojaAsistencia.cell(row=8,column=len(fechas)+4).value= "TOTAL"
74      hojaAsistencia.cell(row=8,column=len(fechas)+4).font= Font(bold=True)
75      mDB.db.close()
```

Segmento de Código 2.49 `GenerarReporteAsistencia`(). Parte II

c) En el Segmento de Código 2.50, en las líneas 78 a la 85 se carga la información de los estudiantes según el paralelo indicado. En las filas 88 a la 97 se carga el número de cédula, el apellido y el nombre de cada estudiante en la plantilla. En las filas 99 a la 106 se rellena la Tabla de asistencias con ceros.

d) En el Segmento de Código 2.51, en las líneas 110 a la 117 se carga la asistencia de los estudiantes según la actividad académica. En las líneas 119 a la 128 identifico las filas en las que se encuentra cada estudiante. En las líneas 130 a la 138, según la fila en la que se encuentre cada estudiante y la celda de la fecha de la actividad académica cambio el valor de la celda por un 1, lo cual implica que el estudiante asistió a clases.

e) En el Segmento de Código 2.52, en las filas 141 a la 142 en la celda después de la última actividad académica de cada estudiante, se realizará una suma de todas las asistencias, en las líneas 152 a la 154 se puede observar la fórmula de la suma de asistencias.

f) Se genera el gráfico de barras que permitirá observar cómo ha sido la asistencia de los estudiantes, para esto se utiliza la función `BarChart`(), a la cual se le indicará que nombre, ejes, formato, color de línea tendrá. Para señalar los datos a graficar se deberá crear un objeto `Reference`() con el rango de las celdas a utilizar. Posteriormente se le agregará la referencia al gráfico con la función `add_data`().

g) Finalmente, en el Segmento de Código 2.53 se muestra cómo se guarda el archivo con el nombre de la materia, el paralelo y del profesor; y se retorna la información de la ruta del archivo y su nombre.

```
77      print("Busqueda estudiantes")
78      estudiantesParalelo= mDB.paralelos.select(mDB.paralelos,
79          mDB.paralelosestudiantes,mDB.estudiantes).join(
80          mDB.paralelosestudiantes,
81          on=(mDB.paralelos.idParalelo==mDB.paralelosestudiantes.idParalelo)).switch(
82          mDB.paralelos).join(mDB.estudiantes,
83          on=(mDB.paralelosestudiantes.idEstudiante==mDB.estudiantes.idEstudiante)).where(
84          mDB.paralelos.idParalelo==idParaleloIn).order_by(mDB.estudiantes.apellido,
85          mDB.estudiantes.nombre)
86      for x in estudiantesParalelo:
87          print(x.nombreParalelo + " - " + x.estudiantes.apellido + " - " + x.estudiantes.nombre)
88      #Agrego los estudiantes a la plantilla
89      y=8
90      for x in estudiantesParalelo:
91          #Agrego cedula
92          hojaAsistencia.cell(row=y+1,column=1).value= x.estudiantes.cedula
93          #Agrego apellido
94          hojaAsistencia.cell(row=y+1,column=2).value= x.estudiantes.apellido
95          #Agrego nombre
96          hojaAsistencia.cell(row=y+1,column=3).value= x.estudiantes.nombre
97          y=y+1
98      #Relleno de ceros la tabla
99      auxFila=0
100     auxColum=0
101     for fila in range(len(estudiantesParalelo)):
102         auxFila = auxFila +1
103         auxColum=0
104         for columna in range(len(fechas)):
105             auxColum = auxColum+1
106             hojaAsistencia.cell(row=8+auxFila,column=3+auxColum).value= 0
```

Segmento de Código 2.50 `GenerarReporteAsistencia`(). Parte III

```
    print("Busqueda estudiantesPresentes")
    mDB.db.connect()
    estudiantesPresentes= mDB.listasestudiantespresentes.select(mDB.actividadesacademicas,
        mDB.estudiantes).join(mDB.actividadesacademicas,
        on=(mDB.actividadesacademicas.idActividadAcademica==
            mDB.listasestudiantespresentes.idActividadAcademica)).switch(
        mDB.listasestudiantespresentes).join(mDB.estudiantes,
        on=(mDB.listasestudiantespresentes.idEstudiante==mDB.estudiantes.idEstudiante)).where(
        mDB.actividadesacademicas.idParalelo==idParaleloIn)
    #Obtengo las filas en las que se encuentra cada estudiante
    filaEstudiante = []
    all_rows= excel_document['Hoja1']
    for estudiantePresente in estudiantesPresentes:
        for fila in all_rows:
            for columna in fila:
                if (columna.value==estudiantePresente.estudiantes.cedula):
                    #obtengo la fila del estudiante
                    filaEstudiante.append((columna.row,
                        estudiantePresente.estudiantes.cedula,
                        estudiantePresente.actividadesacademicas.fecha))
    #segun la fila, el estudiante y la fecha ubico la asistencia
    for fechaAsistencia in filaEstudiante:
        for columna in all_rows:
            for celda  in columna:
                #hago la busqueda en las celdas y comapro la fecha
                #para obtener la letra de la columna
                if (celda.value==fechaAsistencia[2]):
                    # segun la letra de la columna mas la
                    #fila del estudainte ya guardada seteo la asistencia
                    hojaAsistencia[celda.column+ str(fechaAsistencia[0])].value=1
```

Segmento de Código 2.51 `GenerarReporteAsistencia()`. Parte IV

```
#obtengo el total de asistencias
totalAsistenciaEstudiante = []
for columna in all_rows:
        for celda  in columna:
            #hago la busqueda en las celdas
            if (celda.value=="TOTAL"):
                # segun la letra de la columna mas la fila
                #del estudiante ya guardada seteo la asistencia
                totalAsistenciaEstudiante=[celda.column,celda.row]
x = 1
for estudiante in range(len(estudiantesParalelo)):
    filaAux=int(totalAsistenciaEstudiante[1]) + x
    hojaAsistencia[totalAsistenciaEstudiante[0]+
    str(filaAux)].value='=SUM('+chr(68)+str(filaAux)+
    ':'+chr(68+len(fechas)-1)+str(filaAux)+')'
    x=x+1
```

Segmento de Código 2.52 `GenerarReporteAsistencia()`. Parte V

```
    #Guardo el archivo
    nombreArchivo = fechas[0].materias.nombreMateria + "_" +
     fechas[0].paralelos.nombreParalelo + "_"+
     (fechas[0].profesores.nombre + " " + fechas[0].profesores.apellido)
    rutaArchivo = (r"C:\Tesis_ReconocimientoFacial\ReportesAsistencia\\"+
        nombreArchivo+".xlsx")
    print("Ruta--> " + rutaArchivo)
    excel_document.save(rutaArchivo)
    return rutaArchivo,nombreArchivo
```

Segmento de Código 2.53 `GenerarReporteAsistencia()`. Parte VI

Una vez generado el archivo se procede al envío del mismo mediante el Script "`EnvioReporteMail.py`" a través de su función `EnvioMail()` citado anteriormente. Para esto inicialmente se creó una cuenta de correo electrónico en Gmail, sus credenciales se han guardado en el archivo "`MailAE.txt`", ubicado en la carpeta de trabajo del servidor. En la Figura 2.48 se muestran dichas credenciales que servirán para acceder a la cuenta de Gmail mediante las librerías `smtplib` y `getpass`.

```
1  #Mail para envio de asitencia de estudiantes
2  mail:
3  asistencia.estudiantes.rf.epn@gmail.com
4  password:
5  [redacted]
```

Figura 2.48 Archivo `MailAE.txt`

En la Figura 2.48 se ha ocultado la clave de acceso a la cuenta de Gmail por motivos de seguridad. Otra configuración necesaria para poder lograr enviar el mail a través de nuestro servidor es cambiar el nivel de seguridad de la cuenta de Google, en la Figura 2.49 se muestra dicha configuración.

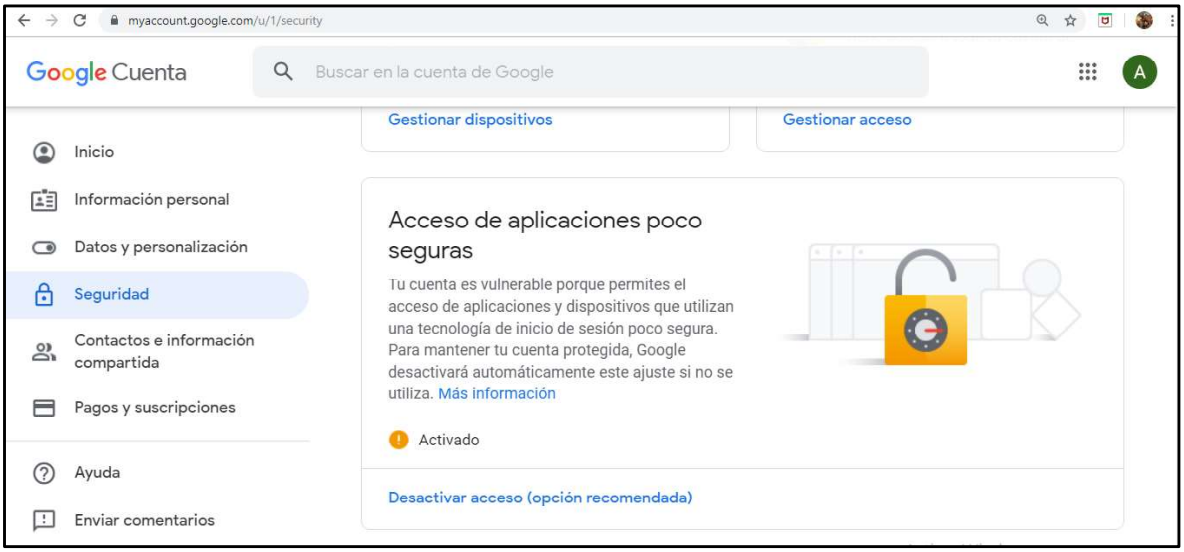

Figura 2.49 Cambio de configuración de seguridad en la cuenta de Google

Luego de haber creado y configurado correctamente la cuenta en Gmail, se procede a crear el Script, en el Segmento de Código 2.54 se muestran las librerías utilizadas, entre las más importantes tenemos `smtplib`, `getpass` para poder acceder al servidor SMTP que proporciona Gmail y la librería `email.mime` para poder crear el mensaje y enviar archivos adjuntos.

```
1  # Librerias para enviar mail
2  import smtplib, getpass
3  from email.mime.multipart import MIMEMultipart
4  from email.mime.text import MIMEText
5  from email.mime.base import MIMEBase
6  from email.encoders import encode_base64
7  import peewee as pw
8  import ConexionDB as cDB
9  import ModeloDB as mDB
10 import os
```

Segmento de Código 2.54 Librerías del archivo EnviarReporteMail.py

En los segmentos de código 2.55 y 2.56, se muestra el procedimiento a seguir para enviar el archivo vía mail, y es el siguiente:

- Primero, la función `EnvioMail()` recibe como parámetro de entrada el mail de destinatario, la ruta del archivo y el nombre del archivo (ver línea 12).
- Segundo, se abre y se lee el archivo "MailAE.txt" para obtener el nombre y la contraseña de la cuenta de mail (ver líneas 16 y 17).
- Tercero, en las líneas 20 a la 24 se crean los componentes del mensaje, en Strings se guardan la dirección del remitente, la dirección del destinatario, el asunto, el mensaje y la ruta del archivo a adjuntar.
- Cuarto, en la línea 26 se crea un objeto `smtplib.SMTP()` llamado `userGmail`, este objeto se lo instancia con la dirección del servidor SMTP de Gmail que es "`smtp.gmail.com`" y el puerto de acceso "`587`". En la línea 28 se ejecuta el protocolo de cifrado de Gmail con la función `starttls()`.
- Quinto, en la línea 30 se inicia sesión en el servidor con la función `LogIn()` del objeto `userGmail`, y en la línea 31 se setea el nivel de depuración en 1.
- Sexto, en las líneas 33 a la 36 se instancia con los String creados en el tercer procedimiento la cabecera del mensaje. En las líneas 38 y 40 se instancia el mensaje.
- Séptimo, en las líneas 42 a la 50 se verifica la existencia del archivo a adjuntar y se lo añade al mensaje.
- Octavo, en la línea 53 se envía el mail con la función `sendMail()` detallando el remitente, el destinatario y la cabecera del mensaje.
- Noveno, en la línea 56 se cierra la conexión.

```
12  def EnvioMail(mailDestinatario,pathIN,nombreArchivo):
13      #Envia el reporte generado previamente al mail solicitado en la APP,
14      # No se permite tildes
15      #tanto en el nombre de la materia como en el nombre del profesor
16      fileMAE = open("MailAE.txt","r")
17      argMAE = fileMAE.readlines()
18      print(argMAE[2].rstrip('\n'))
19      #Cuerpo del mail
20      remitente= argMAE[2].rstrip('\n')
21      destinatario= mailDestinatario
22      asunto = "Reporte de Asistencia: " + nombreArchivo
23      mensaje = " Estimad@ Usuario\n\n Se ha enviado el reporte de asistencias
24      pathAdjunto = pathIN
25      # direccion y puerto de mail GMAIL
26      userGMAIL= smtplib.SMTP('smtp.gmail.com',587)
27      #Protocolo de cifrdo de gmail
28      userGMAIL.starttls()
29      #user, pass gmail
30      userGMAIL.login(argMAE[2].rstrip('\n'),argMAE[4])
31      userGMAIL.set_debuglevel(1) # muestra la depuracion de envio 1=true
32      #Cabecera mail
33      headerMail = MIMEMultipart()
34      headerMail['Subject'] = asunto
35      headerMail['From'] = remitente
36      headerMail['To'] = destinatario
37      # mensaje a HTML
38      mensaje = MIMEText(mensaje, "plain")
39      # se incluye el manesaje
40      headerMail.attach(mensaje)
```

Segmento de Código 2.55 Función `EnvioMail()`. Parte I

```
41
42      #Comprobacion de existencia de archivo a adjuntar
43      if (os.path.isfile(pathAdjunto)):
44          archivoAdjunto= MIMEBase("application","octet-stream")
45          #obtener y leer contenido de archivo
46          archivoAdjunto.set_payload(open(pathAdjunto, "rb").read())
47          encode_base64(archivoAdjunto)
48          archivoAdjunto.add_header("Content-Disposition",
49              'attachment; filename="%s"' % os.path.basename(pathAdjunto))
50          headerMail.attach(archivoAdjunto)
51
52      #Envio mail
53      userGMAIL.sendmail(remitente,destinatario,headerMail.as_string())
54
55      #cierro conexion
56      userGMAIL.quit()
```

Segmento de Código 2.56 Función EnvioMail(). Parte II

3. RESULTADOS Y DISCUSIÓN

En este capítulo se presentarán las pruebas de funcionamiento e integración del sistema prototipo. Las pruebas realizadas se las presentará siguiendo el orden de programación según los SPRINTS de la sección 2.2. Luego se presentarán pruebas en un ambiente construido y un resumen de la evolución del sistema al entrenarlo con varias fotografías.

También se incluye al final de este capítulo una sección de corrección de errores encontrados en la codificación del prototipo.

3.1 PRUEBAS DE FUNCIONAMIENTO DE MÓDULOS DEL SISTEMA

Las pruebas realizadas fueron validadas en los diferentes módulos del sistema.

Los dispositivos móviles con los que se probó la aplicación cliente son un Samsung S9 y un Samsung A10.

3.1.1 SPRINT 1

Para verificar el funcionamiento de la base de datos se realizó las siguientes pruebas:

- Primero, se verificó la conexión al servidor de la base de datos desde el programa Workbench de MySQL. En la Figura 3.1 se puede apreciar el estado de conexión de la base de datos.

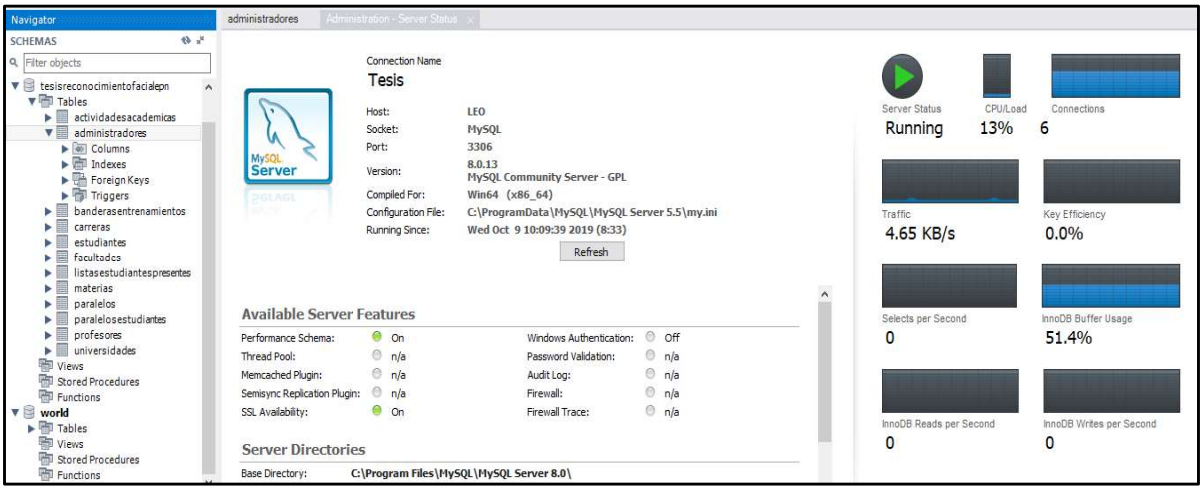

Figura 3.1 Conexión al servidor de base de datos

- Segundo, se realizó consultas a la base de datos. Específicamente se creó un usuario administrador genérico para posteriores pruebas y luego se realizó una consulta SELECT del usuario. En el Segmento de Código 3.1 se muestra la consulta

INSERT INTO para crear el usuario y la consulta SELECT para visualizarlo. En la Figura 3.2 se muestra el resultado de la consulta SELECT.

```
INSERT INTO tesisreconocimientofacialepn.administradores (`cedula`, `nombre`, `apellido`, `password`)
VALUES ('1', 'admin', 'admin', 'admin');
SELECT * FROM tesisreconocimientofacialepn.administradores
WHERE tesisreconocimientofacialepn.administradores.cedula = 1;
```

Segmento de Código 3.1 Consultas a la base de datos

	idAdmi	cedula	nombre	apellido	password
	1	1	admin	admin	admin
▶*		NULL	NULL	NULL	NULL

Figura 3.2 Resultado de la consulta SELECT

3.1.2 SPRINT 2

Para el SPRINT 2 se realizaron pruebas de funcionamiento para los diferentes módulos creados tanto en el cliente como en el servidor.

3.1.2.1 LogIn del servidor

Se realizaron pruebas de funcionamiento para el LogIn del Servidor con el usuario genérico creado en la sección 3.1.1. En la Figura 3.3 se muestra el inicio de sesión exitoso del servidor, con esto se prueba que si existe conexión con la base de datos.

```
C:\Users\LEO\Desktop\Tesis_ReconocimientoFacial>python server.py
------------------------------------------------------------
 ESCUELA POLITÉCNICA NACIONAL
 FACULTAD DE INGENIERIA ELÉCTRICA Y ELECTRÓNICA
 PROTOTIPO PARA CONTROL DE ASISTENCIA DE LA EPN
 AUTOR: LUIS LEONARDO FÉLIX MORENO
 TUTOR: MSc. FRANKLIN SANCHEZ
------------------------------------------------------------
Ingrese su usuario: 1
Ingrese su contraseña: admin
Bienvenido : admin admin.
```

Figura 3.3 LogIn del servidor

También se validó un LogIn erróneo, el servidor soporta dicho error, se lo muestra en la Figura 3.4

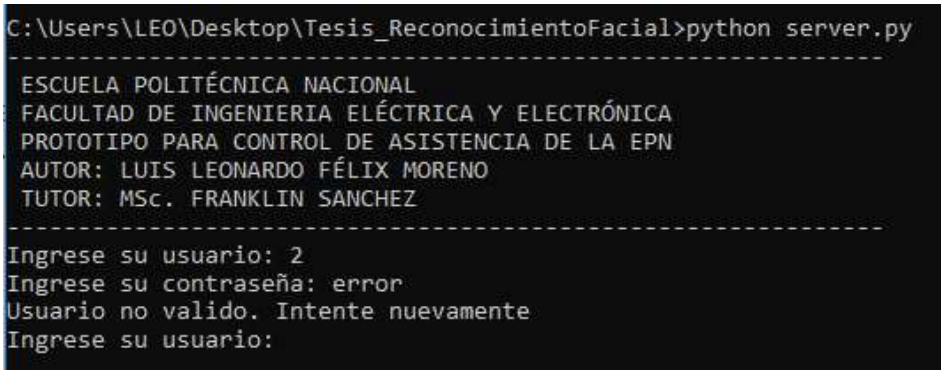

Figura 3.4 Validación de LogIn erróneo

3.1.2.2 Inicializar Servidor

En la Figura 3.5 se muestra como el servidor al tener un LogIn exitoso, permite inicializar sus procesos al seleccionar 1 en el menú. En este estado el servidor empieza el bucle para aceptar conexiones de los clientes.

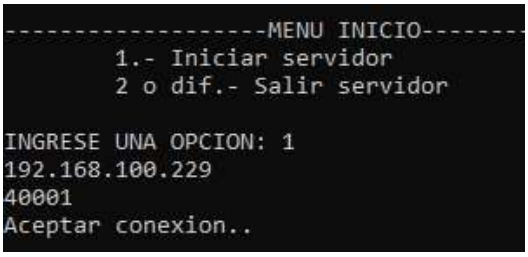

Figura 3.5 Inicializar Servidor

En la Figura 3.6 se muestra la terminación del proceso si se ingresa otra opción diferente a 1 en el menú de inicio.

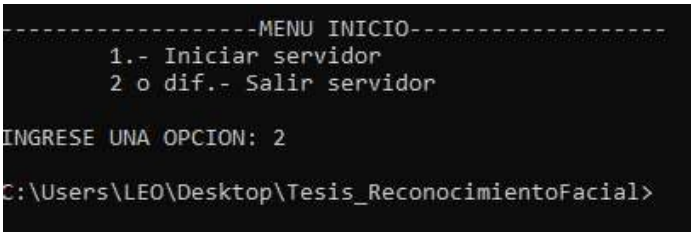

Figura 3.6 Ingreso de una opción diferente a 1 en el menú de inicio

3.1.2.3 LogIn Cliente

Se realizaron pruebas de funcionamiento para el LogIn del Cliente con las credenciales del usuario genérico creado en la sección 3.1.1. En la Figura 3.7 y 3.8 se muestra el inicio de sesión exitoso del cliente, con esto se prueba que la conexión con la base de datos y la comunicación entre cliente y servidor son correctas.

Figura 3.7 LogIn del cliente con usuario Administrador

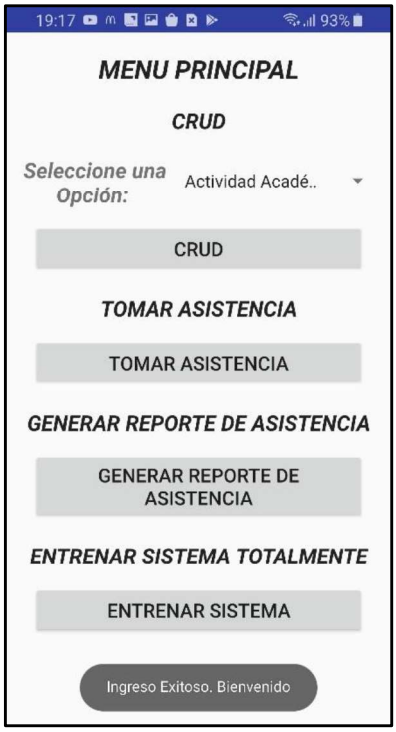

Figura 3.8 Ingreso exitoso al sistema

En la Figura 3.9 se muestra un LogIn erróneo, el sistema soporta este error.

Figura 3.9 Ingreso fallido al cliente

3.1.3 SPRINT 3

En el Sprint 3 se realizaron pruebas de funcionamiento para el módulo del usuario Administrador. Se comprobó que las operaciones CRUD sean exitosas.

En la Figura 3.10, se muestra cómo se crea un nuevo usuario Administrador, en la Figura 3.11 se muestra en la base de datos como se ha creado dicho usuario.

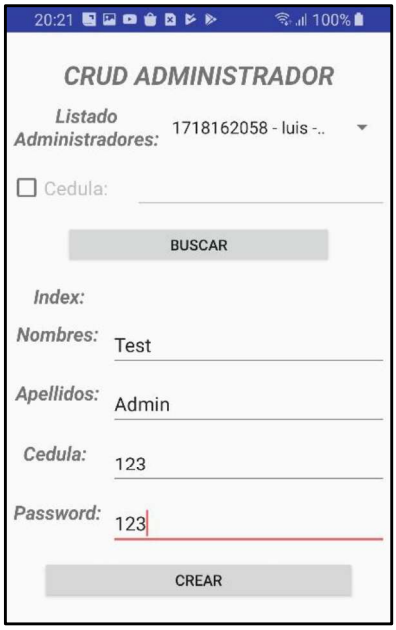

Figura 3.10 Creación de usuario Administrador

	idAdministrador	cedula	nombre	apellido	password
▶	9	123	Test	Admin	123

Figura 3.11 Adición del usuario en base de datos

En la Figura 3.12 se muestra la operación CRUD de editar la información de un usuario, se han modificado el apellido, el número de cédula y la contraseña. En la Figura 3.13 se muestra el cambio de la información en la base de datos.

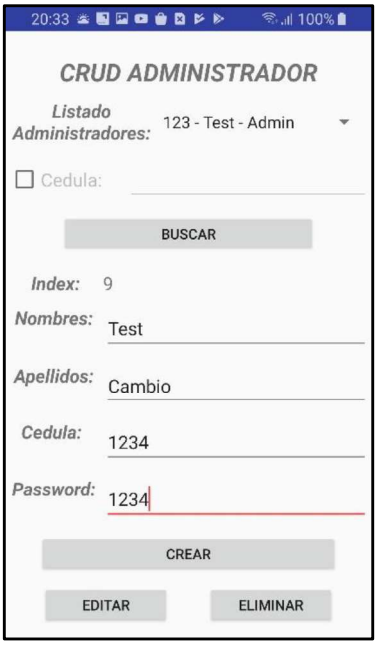

Figura 3.12 Update de usuario administrador

	idAdministrador	cedula	nombre	apellido	password
▶	9	1234	Test	Cambio	1234

Figura 3.13 Cambio de información en la base de datos

Cuando un usuario carga la información de un administrador puede borrarlo de la base de datos, para esto debe dar clic en el botón “Eliminar” que se puede apreciar en la Figura 3.12. En la Figura 3.14 se muestra el usuario eliminado desde la base de datos.

	idAdministrador	cedula	nombre	apellido	password
	1	1718162058	luis	felix moreno	ll199377
▶	4	1	admin	admin	admin
*	NULL	NULL	NULL	NULL	NULL

Figura 3.14 Usuarios administradores

Cuando un proceso CRUD es exitoso, la información cargada en el formulario se elimina para que se pueda realizar otro proceso y se muestra un mensaje de éxito, en la Figura 3.15 se muestra dicho procedimiento.

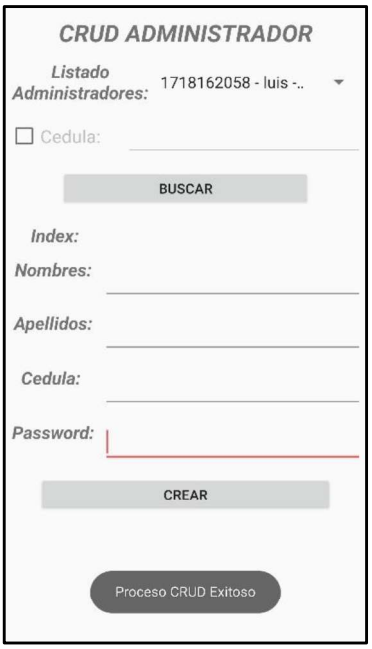

Figura 3.15 Mensaje de éxito en proceso CRUD

Cuando un proceso CRUD es fallido, la información cargada en el formulario se elimina para que se pueda realizar otro proceso y se muestra un mensaje de error, en la Figura 3.16 se muestra dicho procedimiento.

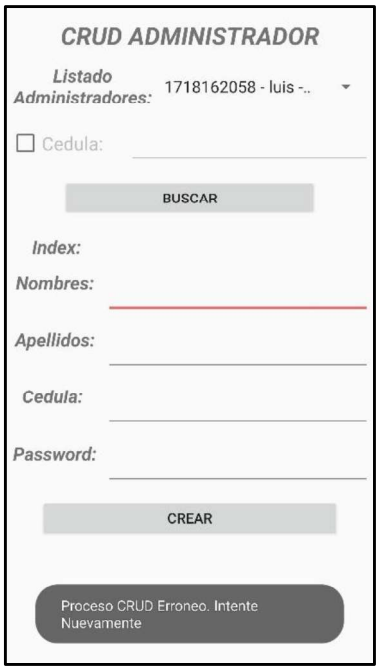

Figura 3.16 Mensaje de error en proceso CRUD

El proceso CRUD mostrado se cumple para los demás módulos del sistema prototipo, por lo tanto, solo se mostrará procesos que sean diferentes y necesarios.

3.1.4 SPRINT 4

Para los módulos CRUD Universidad, CRUD Facultad, CRUD Carrera y CRUD Materia; se cumple el mismo procedimiento que el CRUD Administradores para todos sus procesos, por lo tanto, repetirlos no se lo considera necesario.

En la Figura 3.17 se muestra como en el CRUD Materia, al seleccionar en un Spinner la opción de universidad, en el Spinner de facultad se cargan las opciones pertenecientes a dicha universidad, así pasará para todos los demás Spinners.

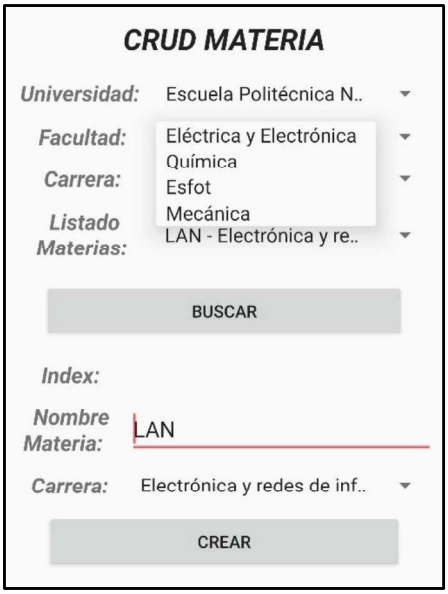

Figura 3.17 Carga automática de información.

En la Figura 3.18 se muestra como se ha controlado la excepción generada al cargar un Spinner de una instancia que no contenga información. Se lo carga con un False.

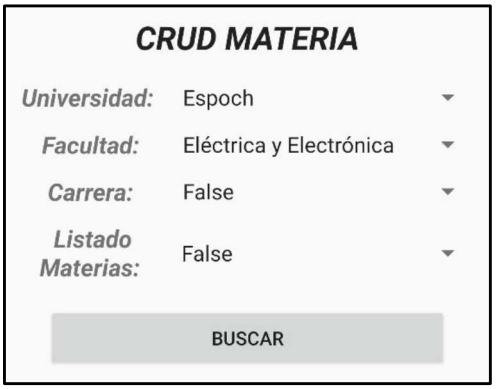

Figura 3.18 Error al cargar una instancia sin valores

La carga de información automática se la ha implementado en todos los módulos, por lo que mostrarlos en cada uno de ellos no será necesario.

3.1.5 SPRINT 5

Para los módulos de usuario Profesor, CRUD Paralelo y CRUD Estudiante se cumple el mismo procedimiento que el CRUD Administradores para todos sus procesos, por lo tanto, repetirlos no se lo considera necesario.

Al crear un usuario Profesor se ha realizado la validación de LogIn del mismo. En la Figura 3.19 se puede observar los profesores creados en la base de datos. En la Figura 3.20 y 3.21 se muestra el proceso de LogIn y la restricción que se le da al usuario de algunos módulos.

idProfesor	cedula	nombre	apellido	password	idUniversidad
1	1	profesorLAN	profesorLAN	12345678	1
2	2	prfesorWAN	profesorWAN	12345678	1
11	1718162058	luis	felix	12345678	1
12	0502398886	Franklin	Sanchez	clavedePRUEBA	1
NULL	NULL	NULL	NULL	NULL	NULL

Figura 3.19 Usuarios profesores

Figura 3.20 LogIn usuario profesor

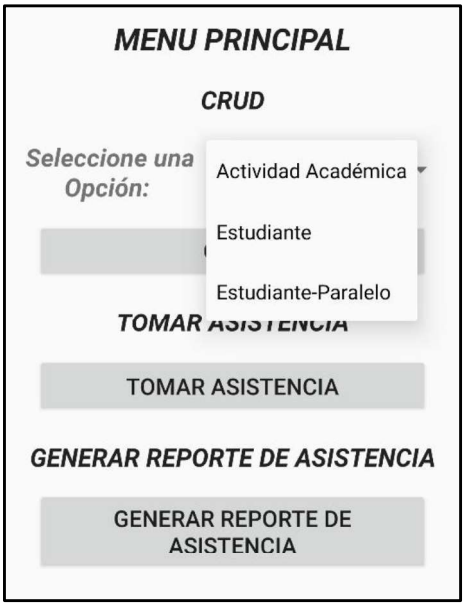

Figura 3.21 Menú principal de usuario profesor con módulos restringidos

En la Figura 3.22 se muestra que, para evitar errores de tipo texto, en variables del tipo Int, se ha limitado el teclado a un tipo numérico para TextBoxs que lo requieran. Esto se lo ha implementado en toda la aplicación cliente.

Figura 3.22 Limitación del texto ingresado en el campo usuario a solo numérico

3.1.6 SPRINT 6

Para el SPRINT 6 se realizó las pruebas de funcionamiento de los diferentes módulos según corresponda.

3.1.6.1 Asignar estudiante a paralelo

En este módulo se permite asignar a un estudiante a un paralelo, en la Figura 3.23 se muestra dicho procedimiento y en la Figura 3.24 se muestra al estudiante agregado en la lista de paralelo seleccionado.

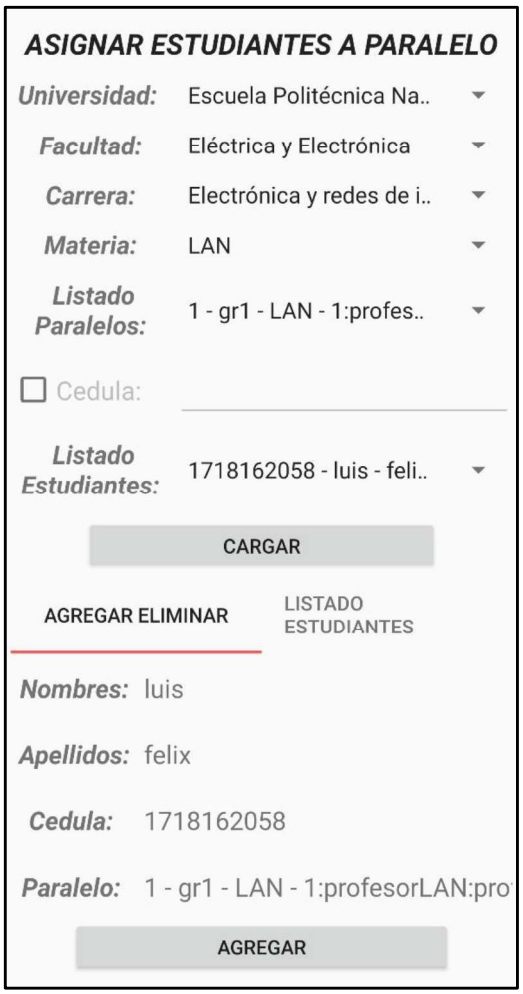

Figura 3.23 Agregar estudiante a paralelo

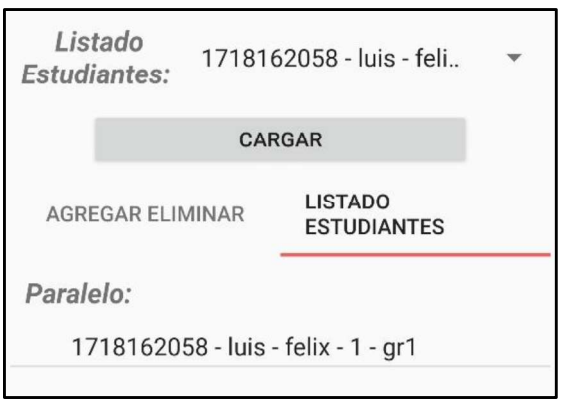

Figura 3.24 Lista de estudiantes por paralelo

En la Figura 3.25 se muestra el proceso de eliminación del estudiante de paralelo asignado.

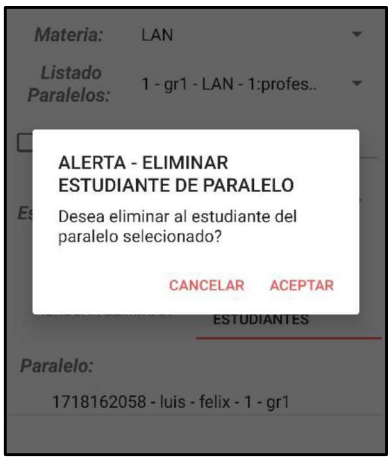

Figura 3.25 Eliminación de estudiante de paralelo asignado

3.1.6.2 CRUD Actividad Académica

Para el proceso de CRUD de actividad académica se ha utilizado la misma lógica que en los demás módulos del cliente, por lo tanto, no se los mostrará.

En la Figura 3.26 se muestra el AlertDialog creado para seleccionar la fecha de una actividad académica, esto se lo implementó para controlar la excepción de formato de fecha.

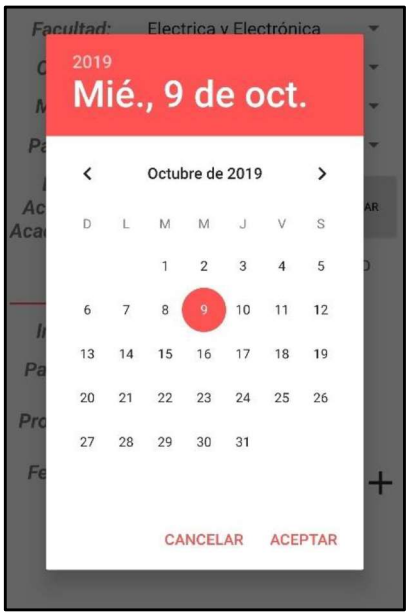

Figura 3.26 AlertDialog para seleccionar una fecha

En la Figura 3.27 se muestra el listado generado de actividades académicas para un paralelo seleccionado, en el módulo correspondiente.

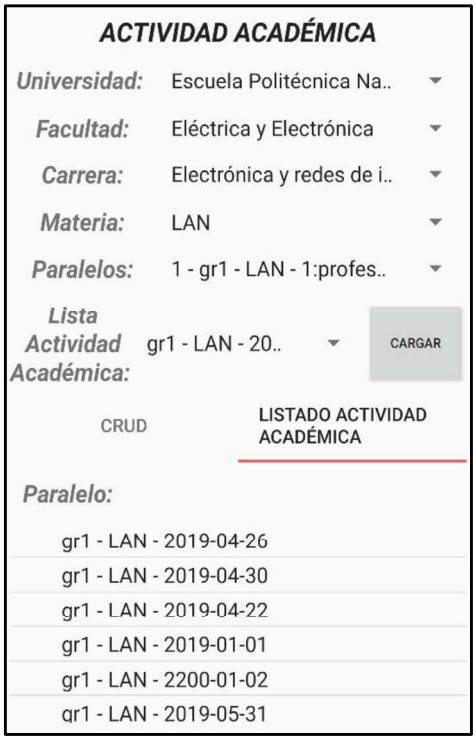

Figura 3.27 Listado de actividades académicas para un paralelo seleccionado

3.1.7 SPRINT 7

Para el SPRINT 7 se realizó las pruebas de funcionamiento de los diferentes módulos según corresponda.

3.1.7.1 Adición de fotografías de estudiantes al servidor FTP

En el módulo de CRUD estudiante se puede agregar fotografías a la base de imágenes de estudiantes mediante la conexión al servidor FTP.

En la Figura 3.28 se puede apreciar que el servidor FTP está corriendo, además se puede observar las carpetas que contiene el mismo.

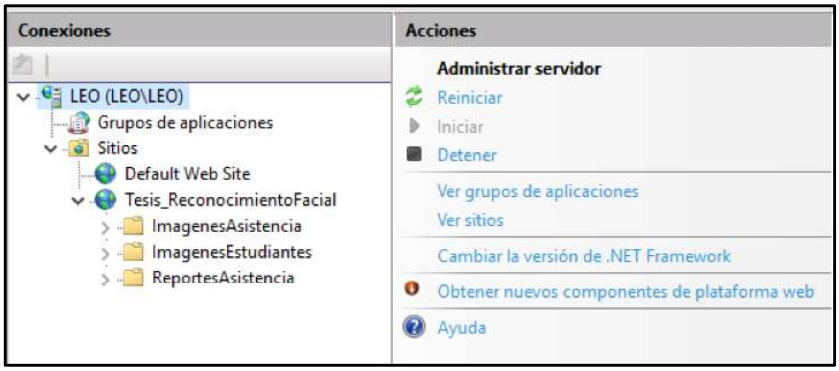

Figura 3.28 Administración del servidor FTP

En la Figura 3.29 se puede apreciar como en el módulo CRUD Estudiante (ver Figura 2.32) al dar clic en el botón “CARGAR IMAGEN” se despliega un AlertDialog para poder seleccionar una de las dos opciones de carga disponible, la primera opción es desde la cámara del dispositivo (ver Figura 3.30) y la segunda es desde la galería de imágenes (ver Figura 3.31).

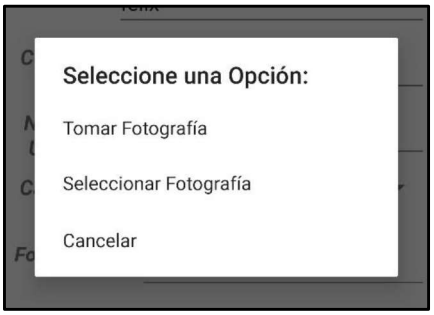

Figura 3.29 AlertDialog para cargar imágenes en CRUD Estudiante

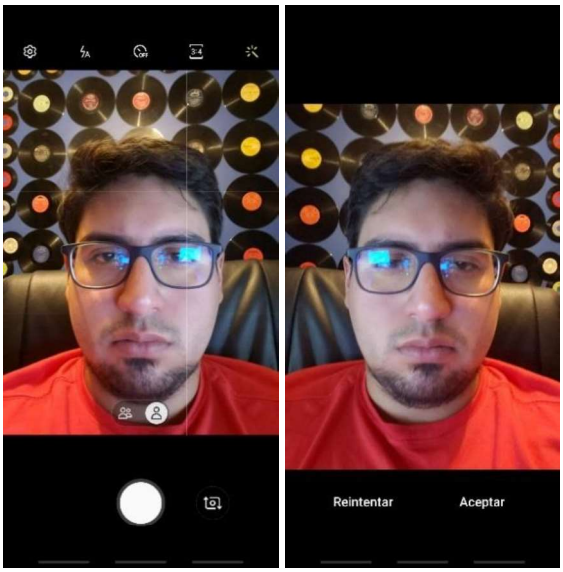

Figura 3.30 Secuencia para tomar una imagen y confirmación de carga

Figura 3.31 Selección de imagen desde galería de imágenes

En la base de imágenes se puede confirmar la carga de las imágenes anteriormente mostradas (ver Figura 2.32).

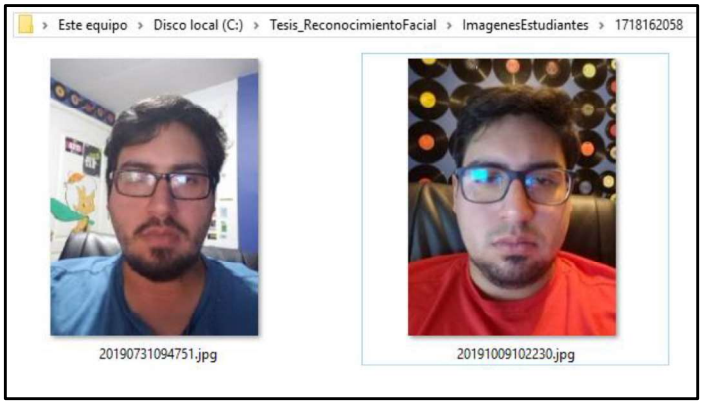

Figura 3.32 Imágenes cargadas en la base de imágenes del estudiante

3.1.7.2 Entrenamiento del sistema por estudiante

Para el entrenamiento del sistema por estudiante como requisito previo se debe tener fotografías cargadas de cada estudiante, para acceder a esta función en el módulo CRUD Estudiante (ver Figura 2.32) al cargar la información de un estudiante se podrá dar clic en el botón “Entrenar Sistema con Estudiante”, luego de desplegará un mensaje confirmando que se realiza el entrenamiento. Este proceso se lo muestra en la Figura 3.33.

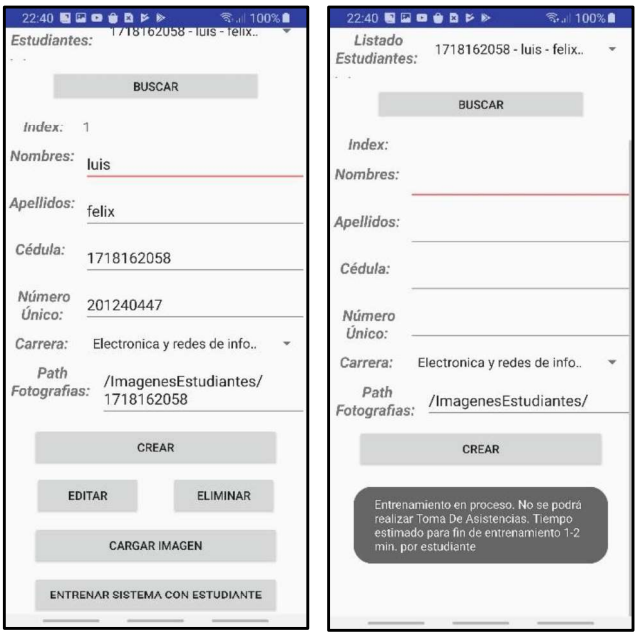

Figura 3.33 Proceso de entrenamiento del sistema por estudiante

Al final del entrenamiento, que durará dependiendo la cantidad de fotografías ingresadas y su peso en bytes, se crearán los archivos “FaceTraining.py” y “LabelTraining.py”, los cuales fueron explicados en la sección 2.2.7.2. En la Figura 3.34 se muestran los archivos creados en la carpeta del estudiante.

Figura 3.34 Base de imágenes de estudiante

3.1.7.3 Entrenamiento total del sistema

Para el entrenamiento total del sistema, el usuario debe estar logueado como Administrador, en el menú de inicio se mostrará un botón llamado “Entrenar Sistema” (ver Figura 3.8). Al dar clic sobre dicho botón se desplegará un AlertDialog para dar una confirmación ya que todo entrenamiento en el sistema bloqueará el módulo “Tomar Asistencia”. En la Figura 3.35 se muestra dicha alerta.

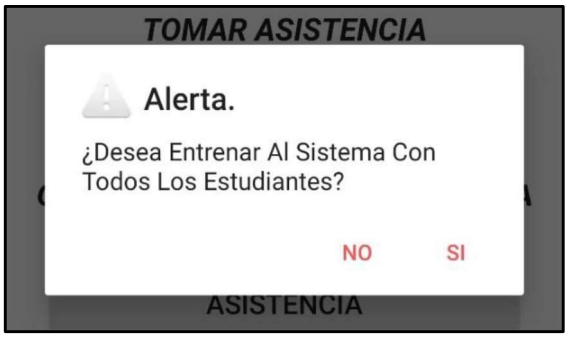

Figura 3.35 AlertDialog Entrenar Sistema

El proceso de entrenamiento será el mismo que en el entrenamiento individual, solo que en el servidor se entrenarán a todos los estudiantes registrados (ver Figura 3.36).

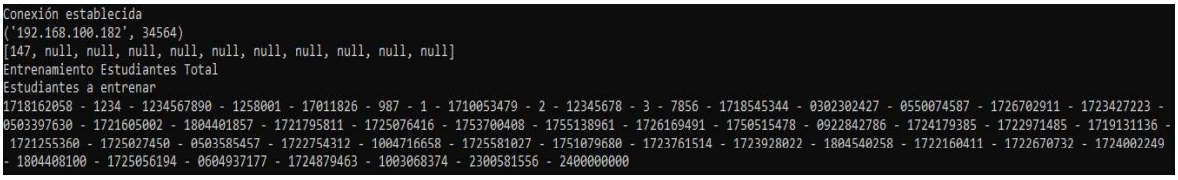

Figura 3.36 Entrenamiento total vista desde el servidor

En la Figura 3.37 se muestra el mensaje desplegado al intentar acceder al módulo “Tomar Asistencia” cuando el entrenamiento se está ejecutando.

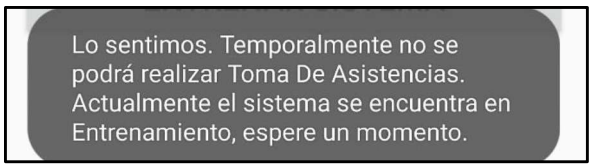

Figura 3.37 Mensaje de bloqueo en entrenamiento del sistema

3.1.8 SPRINT 8

Para el SPRINT 8 se realizó las pruebas de funcionamiento de los diferentes módulos según corresponda.

Para registrar la asistencia a estudiantes, primero se debe tener creada una actividad académica, luego se debe ingresar al módulo “Tomar Asistencia” y se debe cargar las imágenes de los estudiantes presentes para dicha actividad académica, este proceso es similar al mostrado en la sección 3.1.7.1 por lo que no se lo mostrará.

En la Figura 3.38, se muestra el módulo “Tomar Asistencia”, en el cual se ha cargado un paralelo y una actividad académica. Luego se deberá cargar las imágenes dando clic en el botón llamado “Tomar Fotografía Paralelo” (ver Figura 3.39). Finalmente, se deberá dar clic en el botón “Analizar Fotografías” para realizar el registro de asistencia de estudiantes aplicando reconocimiento facial.

Figura 3.38 módulo Tomar Asistencia

Figura 3.39 Base de imágenes de actividad académica

Para esta prueba se han cargado 4 estudiantes, en la siguiente sección se realizarán pruebas en ambientes reales, en las que se explicará más a fondo los resultados, en esta sección solo se está probando que los módulos funcionen correctamente.

En el servidor se cargarán solo los estudiantes que pertenecen a dicho paralelo, en la Figura 3.40 se muestra como el servidor realiza esta acción.

```
Analizar fotografias
Preparación de datos
Estudiantes según paralelo
Dirección estudiante: C:\Tesis_ReconocimientoFacial\ImagenesEstudiantes\1718162058
Dirección estudiante: C:\Tesis_ReconocimientoFacial\ImagenesEstudiantes\1
Dirección estudiante: C:\Tesis_ReconocimientoFacial\ImagenesEstudiantes\2
Dirección estudiante: C:\Tesis_ReconocimientoFacial\ImagenesEstudiantes\1710053479
```

Figura 3.40 Estudiantes según paralelo

Una vez finalizado el reconocimiento facial y almacenado los resultados en la base de datos, se retornará al cliente la lista de los estudiantes presentes en dicha actividad académica, para que este los muestre en una lista de CheckBoxs. Esta lista se podrá modificar manualmente.

En la Figura 3.41 se muestra la lista de estudiantes presentes almacenada en la base de datos.

idListaEst	idActividadAcademica	idEstudiante	cedula
2035	11	19	1
2036	11	1	1718162058
2037	11	21	2
2038	11	20	1710053479

Figura 3.41 Lista de estudiantes presentes en base de datos

En la Figura 3.42 se muestra la lista de estudiantes presentes en el cliente.

Figura 3.42 Lista de estudiantes presentes

Finalmente, el cliente deberá dar clic en el botón “Guardar Asistencia de Estudiantes” si ha realizado algún cambio, para que se refleje en la base de datos.

3.1.9 SPRINT 9

Para el Sprint 9 se realizó pruebas para la generación de Registros de asistencia en Excel y posteriormente enviarlos vía mail.

Se realizó una prueba de generación de reporte en el módulo “Generar Reporte de Asistencia”, se seleccionó un paralelo y se ingresó una dirección de mail válida. En la Figura 3.43 se muestra dicho módulo con la información necesaria.

Una vez generado el reporte de asistencia (ver Figura 3.44) se lo almacenará en la carpeta “/ReportesAsistencia” del servidor FTP (ver Figura 3.45); y se lo enviará vía mail mediante la dirección de correo electrónico especificada en el archivo “MailAE.txt” ubicada en el espacio de trabajo del servidor. En la Figura 3.46 se muestra el mail recibido en la cuenta de correo especificada en la Figura 3.43.

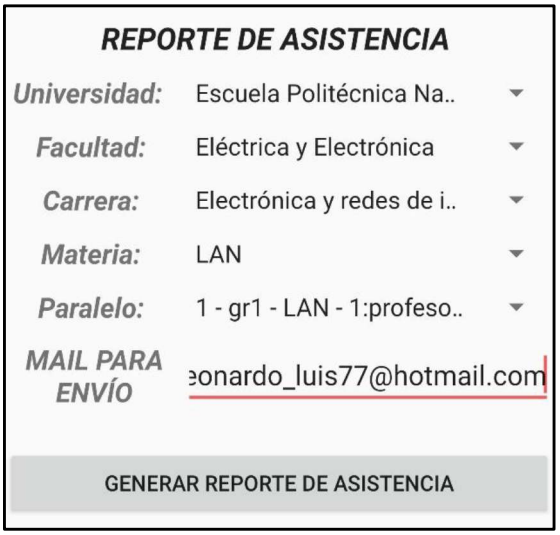

Figura 3.43 Módulo reporte de asistencia

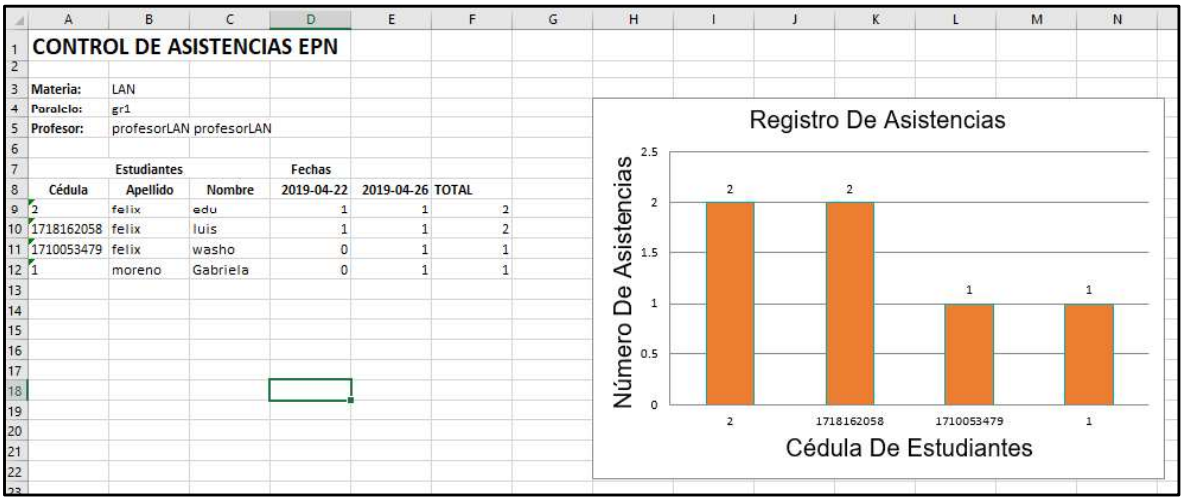

Figura 3.44 Reporte de asistencia

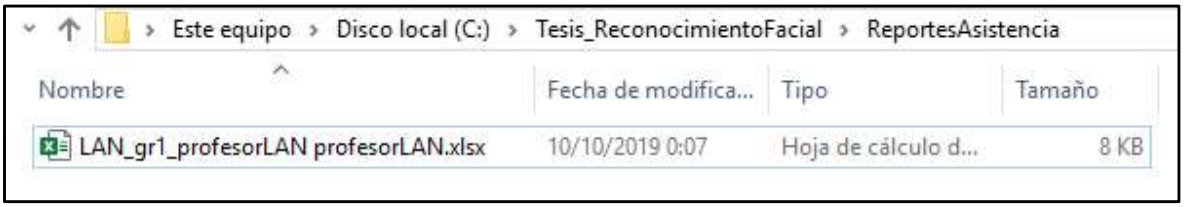

Figura 3.45 Carpeta "/ReportesAsistencia" del servidor FTP

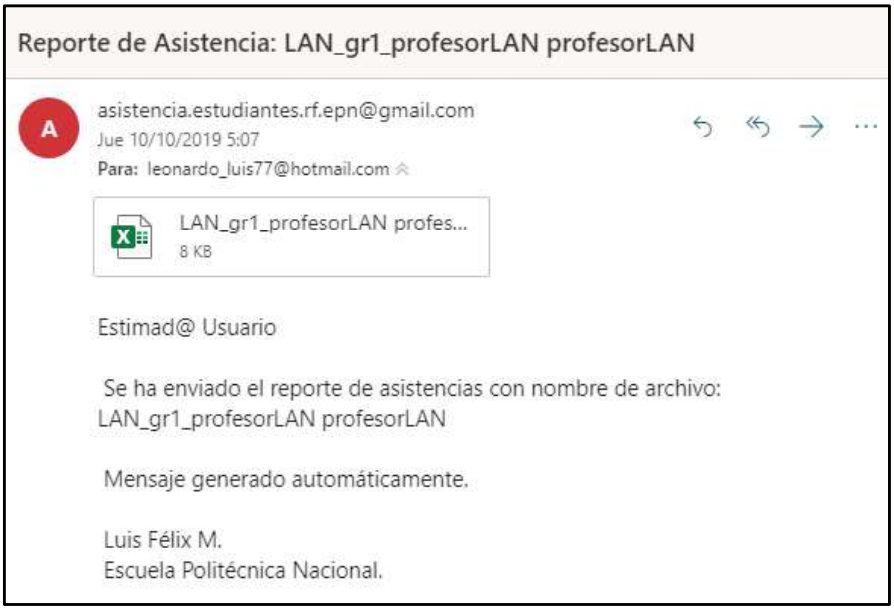

Figura 3.46 Mail enviado desde el servidor

En el módulo de generación de reporte al ingresar una dirección de mail no válida es decir sin un "@", el sistema mostrará un aviso de error que se muestra en la Figura 3.47.

Figura 3.47 Mensaje de error, Email no válido.

3.2 PRUEBAS EN AMBIENTE REAL

Después de validar todos los módulos del sistema y probar su correcto funcionamiento, es necesario realizar pruebas en ambientes reales, es decir, se debe evaluar como el prototipo actuará ante un uso continuo, con una carga de información constante y sobre todo como será su comportamiento ante el manejo por parte de los usuarios.

Para esto se decidió probar el sistema prototipo en dos grupos de estudiantes de la EPN, el primero es el GR1 de la materia Programación Avanzada (llamado en adelante grupo de prueba 1), que consta de 20 estudiantes; y el segundo es el GR2 de la materia de Programación (llamado en adelante grupo de prueba 2), que consta de 17 estudiantes. El tiempo aproximado de prueba fue de un mes, en el cual primero se realizó una sesión fotográfica para obtener la base de imágenes de los estudiantes y con esto entrenar al sistema, luego se recolectaron diferentes fotografías de los estudiantes presentes para cada actividad académica y se aplicó reconocimiento facial para distinguirlos.

El espacio físico para las pruebas del prototipo fueron las aulas de clases en las que los estudiantes tienen sus actividades académicas normales, en la Figura 3.48 se muestra una fotografía de la misma.

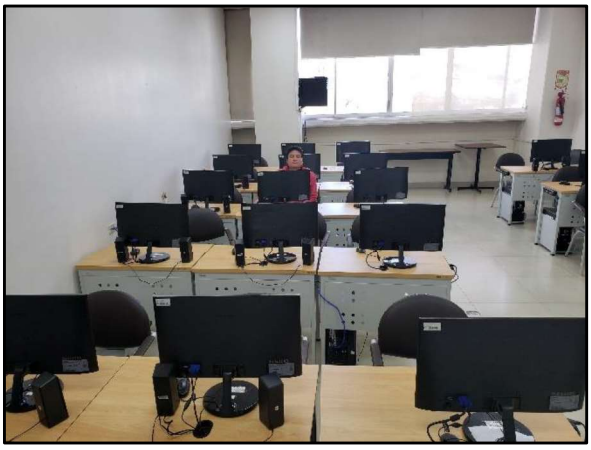

Figura 3.48 Laboratorio de computación "E". FIEE – EPN

La infraestructura utilizada para estas pruebas se muestra en la Tabla 3.3:

Tabla 3.1 Infraestructura utilizada

N°	Equipo	Marca - Modelo	Descripción
1	Computador	Dell Inspirion N4050	Computador en el cual fue alojado el servidor.
2	Router	Nexxt ARN02304UB	Router utilizado para crear la red inalámbrica.
3	Celular	Samsung S9 SM-G9601U1	Dispositivo móvil, en el cual fue instalado la aplicación cliente.

Para la red de pruebas instalada en el laboratorio de cómputo, se utilizó las siguientes características:

- Red: 192.168.0.0/24
- IP router: 192.168.0.1/24
- IP servidor: 192.168.0.2:4001
- IP servidor FTP: 192.168.0.2:21-20
- IP dispositivo móvil: IP dada por el protocolo DHCP del router.
- IP Puerto WAN para salida a internet: 172.xx.xx.44/23
- SSID red WLAN: F---x

3.2.1 INGRESO DE ESTUDIANTES EN EL SISTEMA

En base a la lista de estudiantes para los dos paralelos se procedió a ingresarlos al sistema a través de la aplicación. En las Figuras 3.49 y 3.50 se muestran los estudiantes registrados en la base de datos.

idEstudiante	cedula	nombre	apellido	numeroUnico	idCarrera	pathFotografias
47	0503585457	Adonis Jared	Aguirre Intriago	201720464	16	/ImagenesEstudiantes/0503585457
48	1722754312	Luis Ramón	Amendano Clabon	200910832	16	/ImagenesEstudiantes/1722754312
49	1004716658	Ernesto Javier	Andrade Salgado	201720585	16	/ImagenesEstudiantes/1004716658
63	2300581556	Anthony Andres	Armijos Arboleda	201721430	16	/ImagenesEstudiantes/2300581556
50	1725581027	Kevin Alejandro	Barros Travez	201810759	16	/ImagenesEstudiantes/1725581027
27	1718545344	Roy Boir	Cabrera Garcia	201620231	3	/ImagenesEstudiantes/1718545344
28	0302302427	Luis Fernando	Cantos Muñoz	201520011	3	/ImagenesEstudiantes/0302302427
51	1751079680	Santiago David	Carvajal Vivanco	201721229	16	/ImagenesEstudiantes/1751079680
52	1723761514	Jorge Vladimir	Chalco Simbaña	201710475	16	/ImagenesEstudiantes/1723761514
53	1723928022	Jonathan Andrés	Chamorro Tito	201710485	16	/ImagenesEstudiantes/1723928022
54	1804540258	Pablo Alejandro	Chavez Pérez	204721381	16	/ImagenesEstudiantes/1804540258
29	0550074587	Estalin Wilfrido	Chicaiza Caza	201620118	15	/ImagenesEstudiantes/0550074587
55	1722160411	David Rolando	Corella Logacho	201810582	16	/ImagenesEstudiantes/1722160411
30	1726702911	Alexis Xavier	Criollo Guallichico	201721041	3	/ImagenesEstudiantes/1726702911
56	1722670732	Carlos Andrés	Cuadrado Solis	201810598	16	/ImagenesEstudiantes/1722670732
31	1723427223	Esteban Martín	Fierro Quintana	201620324	3	/ImagenesEstudiantes/1723427223
32	0503397630	Marcelo Jesús	González Borja	201620087	3	/ImagenesEstudiantes/0503397630
33	1721605002	Johnny Alexander	Gualoto Pulupa	201510584	15	/ImagenesEstudiantes/1721605002
57	1724002249	Brandon Patricio	Jiménez Cajas	201720828	16	/ImagenesEstudiantes/1724002249
58	1804408100	Edgar Sebastián	Llumaca Salazar	201721377	16	/ImagenesEstudiantes/1804408100
34	1804401857	Adrian Ricardo	Martinez Morales	201620685	3	/ImagenesEstudiantes/1804401857
35	1721795811	Christian Alejandro	Morán Galarza	201520308	15	/ImagenesEstudiantes/1721795811
59	1725056194	Anghelo Sebastián	Navarrete Cumbal	201720907	16	/ImagenesEstudiantes/1725056194
60	0604937177	Kevin Alexander	Paguay Cali	201720495	16	/ImagenesEstudiantes/0604937177
36	1725076416	Bryan Steven	Peña Maza	201710544	3	/ImagenesEstudiantes/1725076416
37	1753700408	Stefano Andrés	Rodríguez Mosquera	201620659	3	/ImagenesEstudiantes/1753700408
61	1724879463	Bryan David	Rodríguez Villacres	201720891	16	/ImagenesEstudiantes/1724879463
38	1755138961	Steven Leonardo	Rondal Enríquez	201620664	15	/ImagenesEstudiantes/1755138961

Figura 3.49 Estudiantes registrados. Parte I

39	1726169491	Carlos Daniel	Santacruz Tituana	201610687	15	/ImagenesEstudiantes/1726169491
40	1750515478	Nelson Fernando	Sotomayor Sotom...	201721177	3	/ImagenesEstudiantes/1750515478
41	0922842786	Jean Carlos	Suarez Aray	201720519	15	/ImagenesEstudiantes/0922842786
42	1724179385	Jean Carlos	Tandazo Tandazo	201321358	15	/ImagenesEstudiantes/1724179385
43	1722971485	Jorge David	Torres Guerrero	201620305	3	/ImagenesEstudiantes/1722971485
44	1719131136	Francisco Xavier	Valdez Lovato	201420266	3	/ImagenesEstudiantes/1719131136
45	1721255360	Kevin Esteban	Valle Morales	201610326	3	/ImagenesEstudiantes/1721255360
46	1725027450	Ramon Andres	Zambrano Bermello	201420543	15	/ImagenesEstudiantes/1725027450
62	1003068374	Gabriel	Zufferey Palaguerra	201720528	16	/ImagenesEstudiantes/1003068374

Figura 3.50 Estudiantes registrados. Parte II

3.2.2 ENTRENAMIENTO DEL SISTEMA

Una vez definido el ambiente de pruebas e ingresados los estudiantes a la base de datos, se procedió a realizar una sesión fotográfica con los estudiantes para obtener la base de imágenes y con eso entrenar al sistema. Se definió un mínimo de 10 imágenes por

estudiante y posteriormente con los primeros registros de asistencia se aumentó más fotografías. La cantidad de fotografías se las escogió basándose en lo propuesto en la metodología del plan de trabajo de este proyecto y a su referencia bibliográfica a la que apunta.

Algunas expresiones faciales se han definido para que, en el entrenamiento de los estudiantes, se considere que una persona nunca tendrá la misma expresión cuando se tome la fotografía para el reconocimiento facial en una actividad académica. En la Tabla 3.4 se define dichas expresiones para las fotografías, y en la Figura 3.51 se puede apreciar un ejemplo de la base de imágenes inicial para un estudiante.

Tabla 3.2 Detalle de fotografías a tomar para base de imágenes

Cantidad de fotografías	Descripción de la expresión
2	Fotografías con estudiante viendo a la cámara de frente, con seriedad. Similar a una fotografía de cédula.
1	Fotografía con estudiante viendo a la cámara de frente y sonriendo levemente
1	Fotografía con estudiante viendo a la cámara con su rostro levantado levemente y con seriedad.
1	Fotografía con estudiante viendo a la cámara con una inclinación a la derecha y con seriedad.
1	Fotografía con estudiante viendo a la cámara con una inclinación a la izquierda y con seriedad.
1	Fotografía con estudiante viendo a la cámara con su rostro mirando al piso levemente y con seriedad.
1	Fotografía con estudiante viendo a la cámara de frente y cerrando sus ojos.
1	Fotografía con estudiante viendo a la cámara de frente, cerrando sus ojos y sonriendo levemente.
1	Fotografía con estudiante viendo a la cámara de frente y sonriendo, amostrando sus dientes.

Figura 3.51 ejemplo de base de imágenes inicial de un estudiante

Una vez cargadas las imágenes de los estudiantes se procedió a entrenar al sistema, para esto en la aplicación cliente se realizó un "Entrenamiento Total del Sistema". Los parámetros para la detección facial en la función `detectMultiescale()` del script `ModuloEntrenamiento.py` fueron: el número mínimo de cuadros vecinos analizados fue de 4, el factor de escalamiento de las imágenes fue 1.3 y se definió un tamaño mínimo de imagen de 194x194 pixeles. Estos valores fueron seleccionados después de algunos ajustes a los entrenamientos realizados para evitar detecciones falsas.

El proceso de entrenamiento fue explicado en las secciones 3.1.7.2 y 3.1.7.3.

3.2.3 TOMA DE ASISTENCIAS A ESTUDIANTES

Luego de haber completado con éxito el entrenamiento del sistema, se procedió a realizar el registro de asistencia de los estudiantes para las diferentes actividades académicas, en la Tabla 3.5 se muestran las fechas en las que se tomó las asistencias para el grupo de pruebas 1, y en la Tabla 3.6 se muestran las fechas en las que se tomó las asistencias para el grupo de pruebas 2.

Tabla 3.3 Actividades Académicas para el grupo de pruebas 1

N°	Fecha de Actividad Académica
1	19/6/2019
2	21/6/2019
3	22/6/2019
4	24/6/2019
5	25/6/2019
6	26/6/2019
7	2/7/2019
8	3/7/2019

9	8/7/2019
10	10/7/2019
11	17/7/2019
12	29/7/2019
13	30/7/2019
14	31/7/2019

Tabla 3.4 Actividades Académicas para el el grupo de pruebas 2

N°	Fecha de Actividad Académica
1	19/6/2019
2	21/6/2019
3	22/6/2019
4	26/6/2019
5	10/7/2019
6	17/7/2019
7	18/7/2019
8	29/7/2019

Para cada actividad académica se tomó alrededor de 4 fotografías, en las cuales se trató de que cada una se abarque entre 4 y 6 estudiantes aproximadamente, con esto se garantiza que los rostros detectados no serán inferiores a 190x190 pixeles, lo cual es una restricción del sistema. En la Figura 3.52 se muestran las fotografías tomadas para la actividad académica del 31 de julio del 2019, del grupo de pruebas 1; y en la Figura 3.53 se muestran las fotografías tomadas para la actividad académica del 29 de julio del 2019, del el grupo de pruebas 2.

Figura 3.52 Actividad académica del 31 de julio del 2019, del grupo de pruebas 1

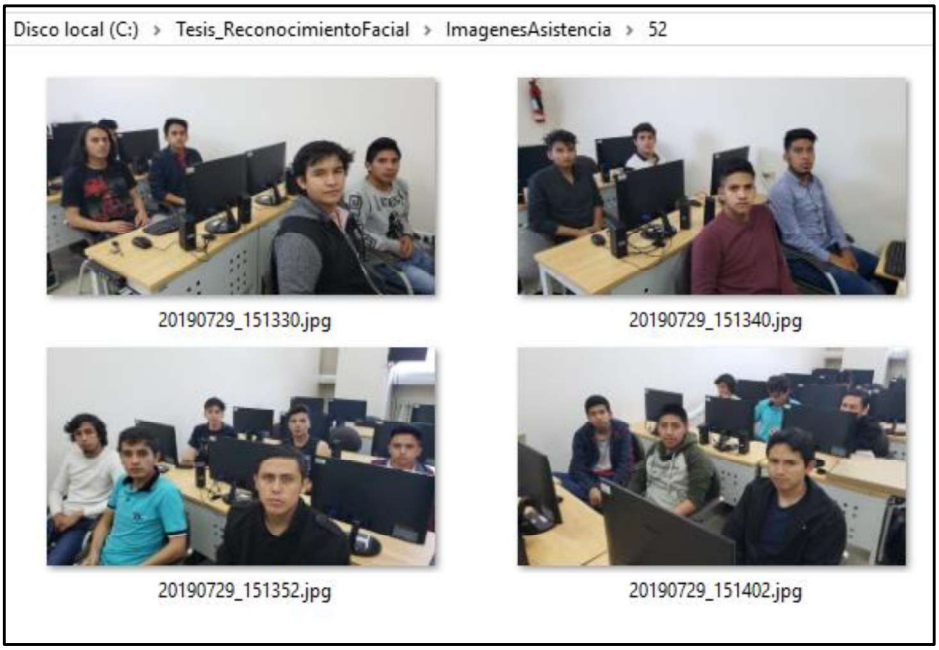

Figura 3.53 Actividad académica del 29 de julio del 2019, del grupo de pruebas 2

Para todas las actividades académicas, luego de haber cargado las imágenes de los estudiantes presentes respectivamente, se analiza las mismas para poder generar el registro de asistencia de ese día y posteriormente cambiar manualmente alguna asistencia de ser necesario. En las Figuras 3.54 y 5.55 se muestran las listas de estudiantes presentes para los dos paralelos en una actividad académica.

En los primeros análisis de fotografías el resultado del registro de asistencia era muy variable en las primeras pruebas se encontró que se llegaba a un 50% de reconocimiento de los estudiantes, y en algunas ocasiones el sistema realizaba reconocimientos falsos, ya que aún faltaba entrenamiento de los estudiantes, para solucionar esto al momento de detectar un rostro, se guardó dichas imágenes y se las agregó a la base de imágenes de los estudiantes respectivamente, además se controló el entrenamiento para que estas imágenes que ya solo poseían rostros no vuelvan a ser analizadas y se las almacene automáticamente en los archivos de lectura. Luego de estas correcciones se llegó a un 100% de reconocimiento. En la Figura 3.56 se muestra la base de imágenes de un estudiante con las nuevas fotografías añadidas. Después de varios análisis y pruebas se llegó a la conclusión de que con 15 fotografías aproximadamente por estudiante, el sistema logra reconocerlo sin dar fallas, pero se recomienda tener el mayor número posible.

Otra acción tomada en el sistema al momento de realizar el reconocimiento facial fue redimensionar sin perder definición a las fotografías de los rostros de los estudiantes, el valor configurado fue 240 x 240 pixeles, con esto nos aseguramos en que la comparación

entre imágenes del entrenamiento con las de actividades académicas no den un valor errado por tener fotografías con diferentes tamaños.

Todos los cambios realizados contribuyeron a que el sistema en las últimas actividades académicas reconozca a todos los estudiantes presentes. Además, se mejoró los tiempos de análisis ya que, al inicio, el sistema se demoraba en analizar todas las fotografías y para retornar las listas de estudiantes presentes al servidor demoraba entre 4 y 5 minutos; después con los cambios realizados se demora 1 minuto máximo.

Figura 3.54 Lista de estudiantes presentes para la actividad académica del 31 de julio del 2019, grupo de pruebas 1

Figura 3.55 Lista de estudiantes presentes para actividad académica del 29 de julio del 2019, del el grupo de pruebas 2

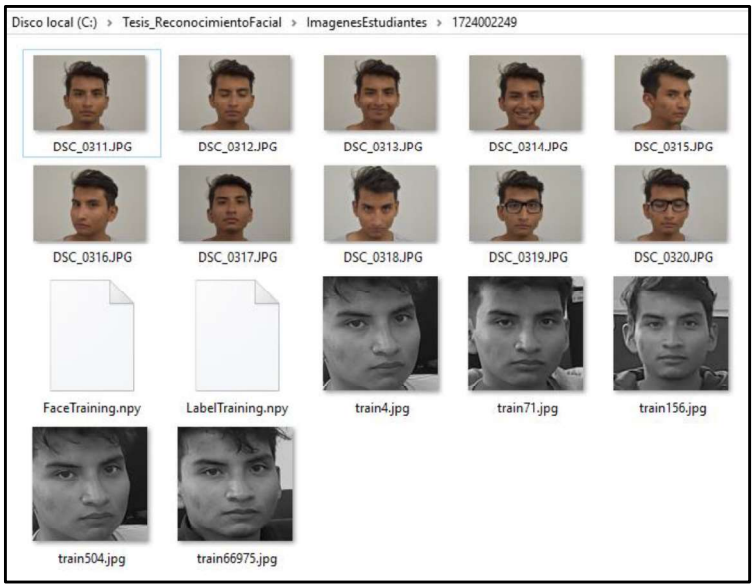

Figura 3.56 Base de imágenes actualizadas de estudiante

Luego de haber realizado la toma de asistencias para todas las actividades académicas planificadas y haber obtenido con éxito los resultados esperados, se generaron los reportes

de asistencia para los dos paralelos y se los envió vial mail. En la Figura 3.57 se muestra el archivo de Excel con los registros de asistencia para el paralelo GR1 de Programación Avanzada y en la Figura 3.58 para el paralelo GR2 de Programación. No se muestran todas las actividades académicas ya que el espacio es limitado para tener una buena visibilidad del archivo, pero se muestra el total de asistencias por estudiante. En los anexos se adjuntará los registros completos.

El modelo de mail recibido se para los registros de asistencia se lo muestra en la Figura 3.46 de la sección 3.1.9.

En el ANEXO F, se adjunta la Base de Imágenes de Estudiantes. En el ANEXO G, se adjunta las Imágenes de Estudiantes Presentes En Actividad Académica. En el ANEXO H, se adjunta los Reportes de Asistencia.

CONTROL DE ASISTENCIAS EPN

Materia:	Programacion Avanzada
Paralelo:	gr1
Profesor:	Franklin Sanchez

Estudiantes

Cédula	Apellido	Nombre	2019-07-30	2019-07-31	TOTAL
1718545344	Cabrera Garcia	Roy Boir	0	1	**9**
0302302427	Cantos Muñoz	Luis Fernando	0	0	**3**
0550074587	Chicaiza Caza	Estalin Wilfrido	1	1	**12**
1726702911	Criollo Guallichico	Alexis Xavier	1	1	**12**
1723427223	Fierro Quintana	Esteban Martín	1	1	**11**
0503397630	González Borja	Marcelo Jesús	1	1	**14**
1721605002	Gualoto Pulupa	Johnny Alexander	0	0	**6**
1804401857	Martinez Morales	Adrian Ricardo	1	1	**11**
1721795811	Morán Galarza	Christian Alejandro	1	1	**10**
1725076416	Peña Maza	Bryan Steven	1	1	**12**
1753700408	Rodríguez Mosquera	Stefano Andrés	1	1	**11**
1755138961	Rondal Enríquez	Steven Leonardo	1	0	**10**
1726169491	Santacruz Tituana	Carlos Daniel	1	1	**10**
1750515478	Sotomayor Sotomayor	Nelson Fernando	1	1	**12**
0922842786	Suarez Aray	Jean Carlos	1	1	**11**
1724179385	Tandazo Tandazo	Jean Carlos	1	1	**12**
1722971485	Torres Guerrero	Jorge David	1	1	**11**
1719131136	Valdez Lovato	Francisco Xavier	1	1	**11**
1721255360	Valle Morales	Kevin Esteban	1	1	**11**
1725027450	Zambrano Bermello	Ramon Andres	1	1	**10**

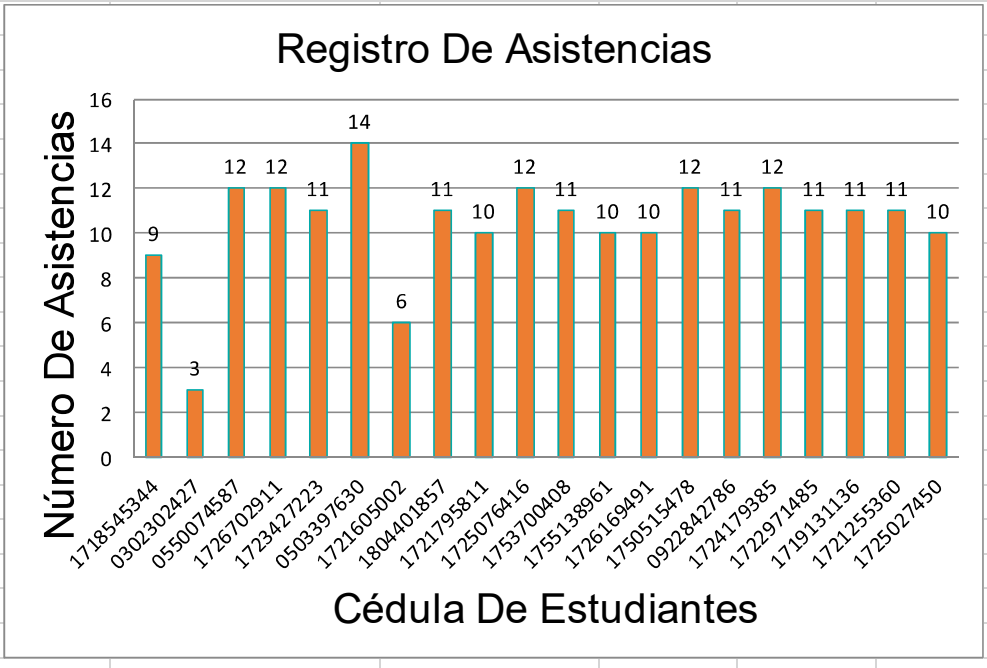

Figura 3.57 Registro de asistencia para el paralelo GR1 de Programación Avanzada

CONTROL DE ASISTENCIAS EPN

Materia:	Programacion
Paralelo:	gr2
Profesor:	Franklin Sanchez

Estudiantes

Cédula	Apellido	Nombre	2019-07-18	2019-07-29	TOTAL
0503585457	Aguirre Intriago	Adonis Jared	1	1	6
1722754312	Amendano Clabon	Luis Ramón	1	1	8
1004716658	Andrade Salgado	Ernesto Javier	1	1	8
2300581556	Armijos Arboleda	Anthony Andres	1	1	8
1725581027	Barros Travez	Kevin Alejandro	1	1	6
1751079680	Carvajal Vivanco	Santiago David	1	1	6
1723761514	Chalco Simbaña	Jorge Vladimir	1	0	4
1723928022	Chamorro Tito	Jonathan Andrés	1	1	7
1804540258	Chavez Pérez	Pablo Alejandro	1	1	6
1722160411	Corella Logacho	David Rolando	1	1	8
1722670732	Cuadrado Solis	Carlos Andrés	1	1	8
1724002249	Jiménez Cajas	Brandon Patricio	1	1	6
1804408100	Llumaca Salazar	Edgar Sebastián	1	1	8
1725056194	Navarrete Cumbal	Anghelo Sebastián	1	1	6
0604937177	Paguay Cali	Kevin Alexander	1	1	8
1724879463	Rodríguez Villacres	Bryan David	1	1	6
1003068374	Zufferey Palaguerra	Gabriel	1	1	7

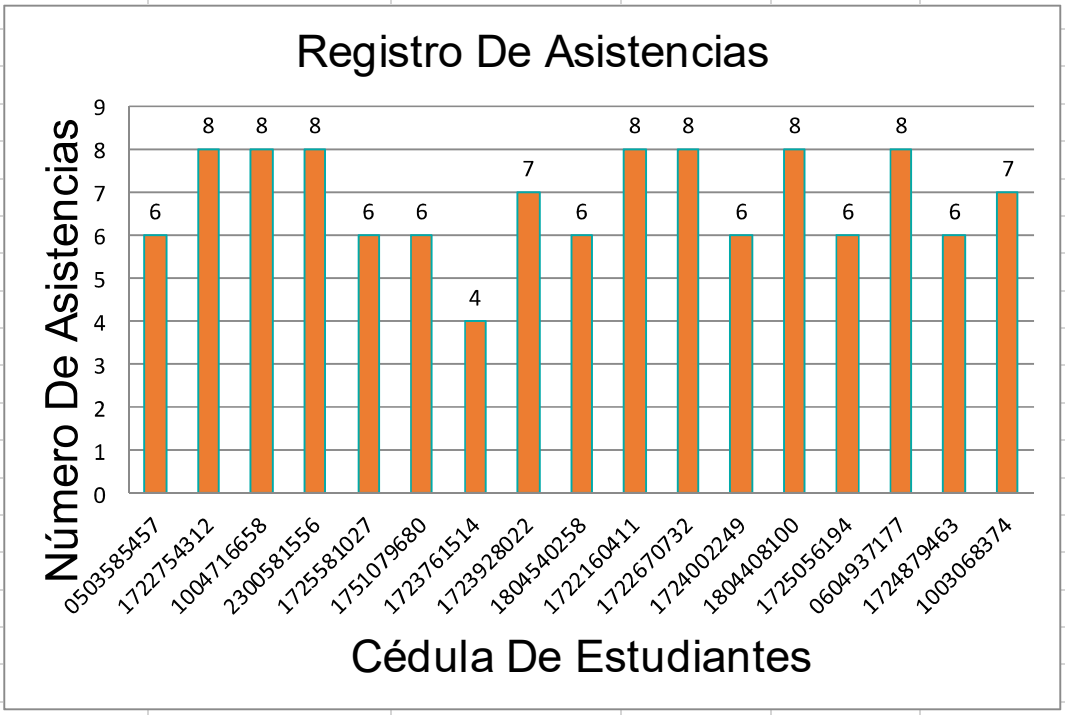

Figura 3.58 Registros de asistencia para el paralelo GR2 de Programación

3.3 ANÁLISIS DE LOS RESULTADOS DEL RECONOCIMIENTO FACIAL OBTENIDOS

Después de haber entrenado al sistema y realizado las diferentes pruebas del mismo, es importante reflejar los resultados obtenidos para determinar en qué porcentaje se cumplen el reconocimiento facial.

Para estas pruebas se utilizó la distancia media entre pixeles que retorna la función de reconocimiento, a este valor se lo transformó en porcentaje tomando en cuenta que el límite fijado en el Segmento de Código 2.43 equivale a 0% de coincidencia.

En la tabla 3.5 se muestra el porcentaje de coincidencia entre el estudiante y el entrenamiento existente, cuando se analiza solo un rostro a la vez para entrenamientos con varios números de fotografías, las pruebas se realizaron para 4 personas.

Tabla 3.5 Análisis del sistema para reconocimiento individual

	Porcentaje de coincidencia (%)		
	Número de fotografías entrenadas		
	5 Fotografías	10 Fotografías	15 Fotografías
Estudiante 1	26.19%	36.51%	69.45%
Estudiante 2	23.81%	40.55%	68.39%
Estudiante 3	17.78%	34.73%	56.88%
Estudiante 4	22.66%	49.28%	72.62%

En la tabla 3.6 se muestra el porcentaje de coincidencia entre el estudiante y el entrenamiento existente, cuando se analiza varios rostros a la vez para entrenamientos con varios números de fotografías, se muestra el resultado para 6 personas de un total de 20 estudiantes.

Tabla 3.6 Análisis del sistema para reconocimiento Grupal

	Porcentaje de coincidencia (%)		
	Número de fotografías entrenadas		
	5 Fotografías	10 Fotografías	15 Fotografías
Estudiante 1	0%	20.45%	38.45%
Estudiante 2	3.46%	26.21%	52.37%
Estudiante 3	0%	23.85%	43.11%
Estudiante 4	0%	18.59%	41.59%

Estudiante 5	7.46%	21.44%	40.68%
Estudiante 6	4.82%	16.22%	46.89%

Como conclusión, aunque el sistema en las pruebas anteriormente presentadas haya sido satisfactorio, los valores de coincidencia de la base de imágenes y las fotografías analizadas no son altos debido a que el entrenamiento aun no es lo suficientemente robusto, esto se lo puede apreciar ya que a medida que se incrementa las fotografías en la base de imágenes porcentaje de coincidencia aumenta en los dos casos.

3.4 CORRECCIÓN DE ERRORES

A continuación, se muestra en forma de Tablas las correcciones realizadas en cada sprint y finalmente en el ambiente de pruebas realizado.

Tabla 3.7 Corrección de errores Sprint 1

SPRINT 1	
Error detectado	**Corrección**
Base de datos no reconoce las Foreing Key	Se asignó manualmente Foreing Key en toda la base

Tabla 3.8 Corrección de errores Sprint 2

SPRINT 2	
Error detectado	**Corrección**
Cédulas de usuarios no consideran el cero inicial.	Se cambió en la base de datos los campos de cédulas de Int a String.
Passwords para ingreso a la aplicación cliente son visibles.	Se cambió el tipo inputType del TextBox a un textPassword.
Se puede ingresar en la aplicación cliente letras en el campo de usuario el cual corresponde a el número de cédula	Se cambió el inputType del TextBox a number (numérico).

Al iniciar sesión en la aplicación cliente se debía ingresar la IP y el Puerto del servidor	Se prefijó una IP y un puerto para evitar que el cliente ingrese esta información.

Tabla 3.9 Corrección de errores Sprint 3

SPRINT 3	
Error detectado	**Corrección**
Problemas al buscar a un usuario administrador solo por su número de cédula	Se agregó un Spinner con todos los usuarios creados
Al realizar una operación CRUD, se debía borrar manualmente todos los campos del formulario.	Una vez realizada cualquier operación CRUD, la información del formulario se elimina totalmente.

Tabla 3.10 Corrección de errores Sprint 4

SPRINT 4	
Error detectado	**Corrección**
Se crean excepciones en la carga automática de información en los Spinners, cuando una entidad superior no tiene entidades inferiores instanciadas, por ejemplo, si una universidad no posee facultades, se produce un error.	Para controlar la excepción si una entidad superior no posee instancias inferiores se carga en el Spinner un False.

Tabla 3.11 Corrección de errores Sprint 5

SPRINT 5	
Error detectado	**Corrección**
Usuario profesor posee acceso a todos los módulos del sistema	Se limitó algunos módulos para el usuario profesor
En el módulo de estudiante al ser extenso, se perdía visibilidad de algunos componentes	Se agregó un ScrollBar para todos los módulos.

Tabla 3.12 Corrección de errores Sprint 6

SPRINT 6	
Error detectado	**Corrección**
Al agregar un estudiante a un paralelo, era difícil ver que estudiantes ya estaban agregados.	Se agregó un ListView para visualizar que los estudiantes asignados a un paralelo seleccionado.
Dificultad al eliminar estudiantes de los paralelos asignados.	En el ListView de estudiantes por paralelo, al seleccionar a un estudiante se permite eliminarlo de dicho paralelo.
Al ingresar la fecha manualmente en el CRUD Actividad Académica se generaban errores de formato.	Se agregó un AlertDialog que permite seleccionar una fecha de un calendario.

Tabla 3.13 Corrección de errores Sprint 7

SPRINT 7	
Error detectado	**Corrección**
Al tomar fotografías con la cámara del dispositivo móvil, en ciertos casos se	Los nombres de las fotografías se los genera en base a la fecha, hora, minutos, segundos y

crea un conflicto con el nombre del archivo si se utiliza nombres aleatorios.	milésimas de segundo en el que fue creado el archivo.
Cuando se realizaban entrenamientos y se realizaba toma de asistencias al mismo tiempo en una misma aplicación, se producían excepciones y caída del servidor.	Se limitó el uso del módulo Tomar Asistencias con banderas, cuando el sistema se encuentra en entrenamiento.
Cuando un entrenamiento fallaba, las banderas no cambiaban sus estados y se bloqueaba el sistema	Se controló las excepciones generadas y al producirse una la bandera cambia su estado.
Al tomar fotografías con el teléfono de prueba Samsung A10, las imágenes salían muy borrosas, ya que el teléfono no posee estabilizador de imágenes.	Se decidió que para la fase de pruebas en el ambiente construido solo se utilizará el Samsung S9.

Tabla 3.14 Corrección errores Sprint 8

SPRINT 8	
Error detectado	**Corrección**
Al cargar fotografías de una actividad académica, era difícil recordar el número de imágenes cargadas o saber si existían precargadas.	Se agregó un TextView que indica cuantas fotografías se encuentran cargadas para la actividad académica seleccionada.
Después de realizar el análisis de las fotografías de una actividad académica y recibir la lista de estudiantes presentes, no se podía cambiar el estado de estudiantes que no hayan sido identificados.	Al seleccionar un paralelo se carga la lista entera de los estudiantes con un estado de ausentes, y al recibir la lista de estudiantes presentes se cambian los estados de ausentes a presentes según corresponda.
Al cambiar el estado de asistencia de un estudiante de presente a ausente en la	Al guardar los cambios de las asistencias, al servidor se envían la lista de estudiantes

Tabla de `EstudiantesPresentes` no se elimina al estudiante.	presentes y ausentes, con esto se los puede eliminar de la Tabla `EstudiantesPresentes`.

Tabla 3.15 Corrección errores Sprint 9

SPRINT 9	
Error detectado	**Corrección**
Error al enviar mail vía SMTP, se retorna un mail con mensaje de origen desconocido.	En la configuración de la cuenta de Google, se dio permiso a aplicaciones externas de conectarse vía SMTP.
Si la red utilizada no poseía salida a internet, se producía una excepción.	Se controló la excepción para que el servidor no se afecte si el mail no puede ser enviado
En el módulo "Generar Reporte de Asistencia", en el TextBox de la dirección de mail, se podía ingresar cualquier texto.	Se creó una función que verifique que se ingrese un mail que contenga una @.

Tabla 3.16 Corrección de errores Pruebas Reales

PRUEBAS EN AMBIENTE REAL	
Error detectado	**Corrección**
Al ingresar un número de cédula mayor a un INT, el sistema en el reconocimiento facial al reconocer a ese usuario retornaba un identificador con un número negativo.	Este error se produce al ingresa un número mayor a un INT el cual es 2'147,483,647 en positivo, la función del reconocimiento no trabaja con Long INT por lo que devolvía un número que cubría su exceso de bits con números negativos. Cuando se produce este error al número negativo se le suma 4'294,967,296, y se corrige el problema
Para el entrenamiento del sistema y toma de asistencias, se utilizó un factor	Después de algunas pruebas se encontró que el valor de 1.3 es el más adecuado.

de escala de 1.5 en la detección de rostros, pero se pasaba por alto muchos rostros. Luego se utilizó un factor de escala de 1.01, pero el sistema generaba muchas detecciones falsas.	
Al realizar el reconocimiento facial de los estudiantes, la distancia de relación superaba en muchas ocasiones el umbral de 50.	Se ajustó todas las imágenes de rostros a 240x240 pixeles, con esto ya no se daba errores por diferencias de tamaño

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Al final del presente proyecto técnico se ha obtenido de forma satisfactoria un sistema prototipo que permite llevar un registro digital de las asistencias de estudiantes de la Escuela Politécnica Nacional, basándose en reconocimiento facial.

- Al final del proyecto se pudo validar que la arquitectura definida para el sistema logró satisfacer todos los requerimientos planteados, con lo que se obtuvo un sistema estable y funcional.

- Gracias al ORM PEEWE se pudo mapear la base de datos en clases para mejorar el acceso a la información desde el servidor y evitar ingresar consultas SQL en el código fuente, ya que al utilizar consultas anidadas y que poseen varias líneas código, mezclar lenguajes de programación, muchas veces se vuelve confuso.

- Gracias a la librería OpenCV para Python, se ha logrado implementar satisfactoriamente el reconocimiento facial de estudiantes asistentes, usando el clasificador en casada Haar para la detección facial y el algoritmo LBPH para el reconocimiento facial.

- El uso de subprocesos e hilos en el sistema prototipo tanto para el servidor, como para el cliente fue primordial, ya que esto evitó el bloqueo del sistema al procesar peticiones con un gran contenido de información.

- La aplicación Android del cliente, permite cargar imágenes al servidor FTP mediante la librería “org.apache.commons.net.ftp.FTPClient”, la cual contribuyó a implementar un sistema robusto y sin pérdida de datos al enviar imágenes desde el cliente al servidor, ya que no se utilizaron tablas de serialización para tratar de aproximar las conversiones realizadas por lenguajes de programación, en vez de eso se utilizó un protocolo que sin importar el lenguaje de programación, la plataforma o el sistema operativo que se utilice, posee las mismas reglas de envío y recepción de información.

- Gracias al sistema prototipo se ha proporcionado una herramienta alternativa para que los profesores ya no lleven sus registros de asistencia de una manera tradicional, ahora podrán llevarlos digitalmente, organizados y disponibles en

cualquier dispositivo que posea la aplicación y conexión a la red donde corra el servidor.

- La librería “openyxl” facilitó la creación y manipulación de las plantillas en Excel de los registros de asistencia, desde el servidor y con comandos en Python. Esto permitió que se pueda presentar los registros en una forma visual y organizada de tal manera que pueda ser útil para un profesor.
- El uso de la librería “smtplib” permitió que se logre enviar vía E-mail, los registros de asistencia generados para un paralelo seleccionado. Esto contribuyó a dar herramientas útiles a los profesores para que se aproveche al máximo el sistema y fomente su uso.
- La creación de un objeto genérico común para poder comunicar al cliente y al servidor ayudó a que en el sistema se evite crear objetos innecesarios y se pueda tener un uso óptimo del sistema.
- El registro manual de asistencia usado como herramienta complementaria al reconocimiento facial, permitió mitigar errores en el registro de asistencia, producidos por reconocimientos falsos de estudiantes sin un entrenamiento adecuado.
- El entrenamiento de estudiantes en el sistema es la clave fundamental para un reconocimiento satisfactorio y sin errores. Al iniciar este proyecto se planificó el uso de 10 imágenes para el entrenamiento, pero al probar el sistema se llegó a la conclusión de que se necesitan mínimo 15 fotografías para no tener errores.
- Basar este proyecto en la metodología Scrum ha permitido realizar cambios de forma oportuna, al evaluar los avaneces a medida que se desarrollaba cada Sprint, con esto se evitó que el diseño del prototipo fracase.
- Los diferentes errores presentados a lo largo del proyecto han servido para fortalecer aspectos funcionales del mismo, por ejemplo, gracias a que se corrigió la sentencia que permite realizar el reconocimiento facial, ahora se pueden utilizar identificadores mayores a un INT.
- En un formulario es importante cargar previamente la información que el usuario podría utilizar, con esto se lo limita a utilizar información que exista en la base de datos o en el sistema.

4.2 RECOMENDACIONES

- Para posteriores proyectos se recomienda la implementación de una aplicación de escritorio o un servicio web para realizar la administración del sistema, ya que el ingreso o administración de una gran cantidad de información en un dispositivo móvil puede ser difícil, debido a que este es compacto y la visualización muchas veces no es la adecuada para las tareas ya mencionadas.

- Se recomienda para el diseño de los Layouts en la aplicación Android, utilizar plantillas predefinidas, con esto se podrá tener una mejor presentación visual del proyecto, además de organización, también se podrá mantener un estilo uniforme en todo el proyecto, y podría simplificar el proceso de implementación ya que algunas funciones como por ejemplos menús se simplifican.

- Para fases posteriores complementarias a este proyecto se recomienda crear el usuario Estudiante y módulos que le permitan visualizar estadísticas de sus asistencias.

- Se recomienda para proyectos futuros, tomar en cuenta las excepciones que se han mencionado en este documento, para que el cliente o el servidor no dejen de funcionar por errores como ingreso de información errónea, envió de información incompleta o errores en las conexiones entre cliente y servidor. .

- Se recomienda para proyectos futuros, realizar pruebas con paralelos que posean un número de estudiantes mayor a 30, para poder analizar los resultados obtenidos y poder dar actualizaciones al sistema de ser necesario.

- Se recomienda que se considere el uso de identificadores de sujetos menores al valor de un INT para el uso de la función de reconocimiento facial, esto para asegurar que no se retornen identificadores negativos y el sistema los ignore.

- Se recomienda al momento de realizar la detección facial de personas, considerar que calidad de imagen se posee ya que al utilizar un factor de escalamiento de la imagen cercano a 1 (1.05 por ejemplo), y si esta posee detalles muy pequeños con una imagen de resolución baja podremos tener detecciones falsas

- Se recomienda para el reconocimiento facial manejar imágenes con una resolución y medida similar ya que así nos aseguraremos de que el reconocimiento sea los más acertado, caso contrario se recomienda entrenar al sistema con fotografías de

varios tamaños y resoluciones para que se pueda cubrir el mayor número de posibilidades en tamaños y resoluciones posibles.

- Se recomienda para proyectos futuros revisar las librerías de reconocimiento facial para identificar, cual es el tamaño máximo del identificador de reconocimiento a utilizar.
- Se recomienda para futuros proyectos utilizar en la base de datos atributos de preferencia String tanto para números de cédulas, números de teléfonos, claves de acceso, códigos postales. Con esto se evitará eliminar dígitos como el cero al inicio de cada atributo.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] CONSEJO DE EDUCACION SUPERIOR, REGLAMENTO DE RÉGIMEN ACADÉMICO, Quito, 2013.

[2] CONSEJO POLITÉCNICO EPN, «REGLAMENTO DE RÉGIMEN ACADÉMICO DE LA ESCUELA POLITÉCNICA NACIONAL,» QUITO, 2017.

[3] K. SCHWABER, Agile Software Development With Scrum, Pearson International, 2002.

[4] G. Coulouris, J. Dollimore, S. Dormido y T. Kindberg, Sistemas Distribuidos. Conceptos y Diseño, Madrid: Pearson Education, 2001.

[5] J.-F. Pillou, «Redes - Arquitectura Cliente/Servidor en 3 niveles,» [En línea]. Available: https://es.ccm.net/contents/147-redes-arquitectura-cliente-servidor-en-3-niveles. [Último acceso: 6 08 2019].

[6] I. Gilfillan, La Biblia De MySQL, España: Anaya Multimedia, 2003.

[7] Natsys, Todo sobre MySQL: Libro ideal para ingresar en el mundo de la base de datos MySQL, Natsys, 2014.

[8] Oracle, «Connectors and APIs,» [En línea]. Available: https://dev.mysql.com/doc/index-connectors.html. [Último acceso: 07 08 2019].

[9] UDLAP, «Modelo relacional,» [En línea]. Available: http://ict.udlap.mx/people/carlos/is341/bases03.html. [Último acceso: 06 08 2019].

[10] Oracle, «Database from Oracle,» [En línea]. Available: https://www.oracle.com/database/. [Último acceso: 07 08 2019].

[11] Microsoft, «SQL Server 2019,» [En línea]. Available: https://www.microsoft.com/es-es/sql-server/sql-server-2019. [Último acceso: 07 08 2019].

[12] Oracle, «MySQL,» [En línea]. Available: https://www.mysql.com/. [Último acceso: 07 08 2019].

[13] A. Hinojosa, Python Paso a paso, Madrid: Grupo Editorial RA-MA, 2016.

[14] Python, «Applications for Python,» [En línea]. Available: https://www.python.org/about/apps/. [Último acceso: 07 08 2019].

[15] S. Leffler, R. Fabry, W. Joy y P. Lapsley, «An Advanced 4.4BSD Interprocess Communication Tutorial,» California.

[16] N. Jennings, «Socket Programming in Python (Guide),» [En línea]. Available: https://realpython.com/python-sockets/#background. [Último acceso: 08 08 2017].

[17] J. Alarcón, «¿Qué es un ORM?,» 28 02 2018. [En línea]. Available: https://www.campusmvp.es/recursos/post/que-es-un-orm.aspx. [Último acceso: 12 08 2019].

[18] Peewee, «Peewee - Project description,» [En línea]. Available: https://pypi.org/project/peewee/2.1.6/. [Último acceso: 12 08 2019].

[19] C. Leifer, «Docs Peewee - Databse,» [En línea]. Available: http://docs.peewee-orm.com/en/latest/peewee/database.html. [Último acceso: 12 08 2019].

[20] C. Leifer, «Docs Peewe- Quickstart,» [En línea]. Available: http://docs.peewee-orm.com/en/latest/peewee/quickstart.html#quickstart. [Último acceso: 12 08 2019].

[21] OpenCV, «OpenCV's Introduction,» [En línea]. Available: https://docs.opencv.org/3.0-beta/modules/core/doc/intro.html. [Último acceso: 12 08 2019].

[22] V. Tabora, «Face Detection Using OpenCV With Haar Cascade Classifiers,» [En línea]. Available: https://becominghuman.ai/face-detection-using-opencv-with-haar-cascade-classifiers-941dbb25177. [Último acceso: 12 08 2019].

[23] W. Berger, «DEEP LEARNING HAAR CASCADE EXPLAINED,» [En línea]. Available: http://www.willberger.org/cascade-haar-explained/. [Último acceso: 12 08 2019].

[24] P. Viola y M. jones, «Rapid Object Detection using a Boosted Cascade of Simple,» Cambridge, 2001.

[25] W. Chin-Feng, «What’s the Difference Between Haar-Feature Classifiers and Convolutional Neural Networks?,» 4 08 2018. [En línea]. Available: https://towardsdatascience.com/whats-the-difference-between-haar-feature-classifiers-and-convolutional-neural-networks-ce6828343aeb. [Último acceso: 0812 2019].

[26] OpenCv, «Face Detection using Haar Cascades,» 23 02 2018. [En línea]. Available: https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html. [Último acceso: 12 08 2019].

[27] C. Esparza, C. Tarazona, E. Sanabria y D. Velazco, «RECONOCIMIENTO FACIAL BASADO EN EIGENFACES, LBHP Y FISHERFACES EN LA BEAGLEBOARD-xM,» Santander, 2015.

[28] K. Salton do Prado, «Face Recognition: Understanding LBPH Algorithm,» 10 11 2017. [En línea]. Available: https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b. [Último acceso: 13 08 2019].

[29] Universidad Carlos III de Madrid, «Arquitectura Android,» [En línea]. Available: https://sites.google.com/site/swcuc3m/home/android/generalidades/2-2-arquitectura-de-android. [Último acceso: 13 08 2019].

[30] Universidad Politécnica De Valencia, «Arquitectura de Android,» [En línea]. Available: http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/99-arquitectura-de-android. [Último acceso: 13 08 2019].

[31] Android Developers, «Arquitectura de la plataforma,» [En línea]. Available: https://developer.android.com/guide/platform?hl=es.

[32] Universidad Politécnica De Valencia, «Instalación del entorno de desarrollo,» [En línea]. Available: http://www.androidcurso.com/index.php/recursos/31-unidad-1-vision-general-y-entorno-de-desarrollo/100-instalacion-del-entorno-de-desarrollo. [Último acceso: 13 08 2019].

[33] V. Gómez, «Introducción a las Metodologías Ágiles,» 14 06 2015. [En línea]. Available: https://instintobinario.com/introduccion-a-las-metodologias-agiles/. [Último acceso: 13 08 2019].

[34] D. Calvo, «Metodología SCRUM (Metodología ágil),» 07 04 2018. [En línea]. Available: http://www.diegocalvo.es/metodologia-scrum-metodologia-agil/. [Último acceso: 13 08 2019].

[35] Scrum Manager, «Historia de usuario,» [En línea]. Available: https://www.scrummanager.net/bok/index.php?title=Historia_de_usuario. [Último acceso: 24 08 2019].

[36] M. Navas, «Cómo configurar servidor FTP en Windows 10,» 01 05 2016. [En línea]. Available: https://www.profesionalreview.com/2016/05/01/configurar-servidor-ftp-windows-10/. [Último acceso: 09 09 2019].

[37] G Suite, «Enviar correo electrónico desde impresoras, escáneres o aplicaciones,» [En línea]. Available: https://support.google.com/a/answer/176600?hl=es. [Último acceso: 10 09 2019].

[38] Computer Hope, «Programming language,» 30 06 2019. [En línea]. Available: https://www.computerhope.com/jargon/p/programming-language.htm. [Último acceso: 07 08 2019].

[39] D. Calvo, «Scrum,» 04 2018. [En línea]. Available: http://www.diegocalvo.es/wp-content/uploads/2018/04/Metodolog%C3%ADa-SCRUM.png. [Último acceso: 13 11 2019].

[40] Medium, [En línea]. Available: https://miro.medium.com/max/640/0*FfWvzaLCs5evm3vU.jpg. [Último acceso: 13 11 2019].

[41] Requeridos Blog, «Requerimientos Funcionales y No Funcionales, ejemplos y tips,» 16 04 2018. [En línea]. Available: https://medium.com/@requeridosblog/requerimientos-funcionales-y-no-funcionales-ejemplos-y-tips-aa31cb59b22a. [Último acceso: 16 11 2019].

[42] Tenstep, «SCRUM: Cómo escribir historias de usuarios sin morir en el intento,» [En línea]. Available: https://www.tenstep.ec/portal/articulos-boletin-tenstep/41-scrum/253-scrum-como-escribir-historias-de-usuarios-sin-morir-en-el-intento. [Último acceso: 16 11 2019].

[43] V. Carlos, «Entorno cliente/servidor,» 23 11 2017. [En línea]. Available: https://es.ccm.net/contents/148-entorno-cliente-servidor. [Último acceso: 03 02 2020].

ANEXOS

Índice de anexos:

ANEXO A

FORMATO PARA ENTREVISTAS A PROFESORES

DESARROLLO DE UN SISTEMA PROTOTIPO PARA EL REGISTRO DE ASISTENCIA DE ESTUDIANTES DE LA ESCUELA POLITÉCNICA NACIONAL, BASADO EN RECONOCIMIENTO FACIAL

FICHA DE ENTREVISTA

La información proporcionada en esta entrevista será confidencial y para uso exclusivo del proyecto en desarrollo.

Elaborado Por:	Luis Leonardo Félix Moreno
Institución:	Escuela Politécnica Nacional
Fecha:	
Número de Pregunta:	Pregunta
1	Como docente universitario, ¿Lleva un registro de asistencia de estudiantes por cada materia en la que está a cargo?, ¿de qué forma lo hace?
2	¿Cuánto tiempo tarda aproximadamente en tomar la asistencia de una actividad académica?
3	¿Conoce algún sistema para control de asistencias que utilice reconocimiento facial?
Se propone la implementación de un prototipo que permita a Profesores llevar el registro de asistencia de estudiantes mediante un dispositivo móvil, aplicando reconocimiento facial. El prototipo contará con un servidor centralizado, una base de datos y una aplicación Cliente en Android, que utilizará la cámara del mismo para obtener fotografías de los estudiantes. En base a lo expuesto, por favor responder a las siguientes preguntas:	
4	¿Considera que sería útil tomar la asistencia a estudiantes mediante reconocimiento facial de una o más fotografías de los asistentes?
5	¿Considera necesario el uso de dispositivo móvil para ingresar los datos de la asistencia y una base de datos para almacenarlos?

6	¿Considera importante que el sistema tenga la opción de autenticarse mediante un usuario y una contraseña?
7	¿Cree usted que las fotografías enviadas al servidor deben almacenarse como respaldo de la información obtenida?
8	Para el reconocimiento facial de una persona es importante tener una base de datos de fotografías previas, con las cuales el sistema se entrena. ¿Usted estaría dispuesto a entrenar al sistema antes de empezar a tomar la asistencia de los estudiantes?
9	¿Considera importante que al final del periodo académico o cuando usted lo considere necesario, pueda generar un reporte la materia que ha impartido?

ANEXO B

HISTORIAS DE USUARIOS.

HISTORIA DE USUARIO		
ID: UH-01	**Usuario:** Administrador, Profesor	
Programador Responsable: Luis Félix	**Sprint:** LogIn	**Número Sprint:** 1
Nombre Historia: LogIn en Cliente	**Prioridad:** Alta	
Descripción: Los usuarios deben poder autenticarse con un nombre de usuario y contraseña en la aplicación cliente.		
Comentarios: El usuario deberá ser su número de cédula y no podrá existir dos usuarios del mismo tipo con el número de cédula repetido.		

HISTORIA DE USUARIO		
ID: UH-02	**Usuario:** Administrador	
Programador Responsable: Luis Félix	**Sprint:** LogIn	**Número Sprint:** 1
Nombre Historia: LogIn en Servidor	**Prioridad:** Alta	
Descripción: El usuario Administrador debe poder autenticarse con un nombre de usuario y contraseña en el servidor.		
Comentarios: El usuario deberá ser su número de cédula y no podrá existir dos Administradores con el número de cédula repetido.		

HISTORIA DE USUARIO		
ID: UH-03	**Usuario:** Administrador	
Programador Responsable: Luis Félix	**Sprint:** LogIn	**Número Sprint:** 1
Nombre Historia: Arrancar Servidor	**Prioridad:** Alta	
Descripción: El usuario Administrador deberá arrancar el servidor par que se pueda utilizar el sistema		
Comentarios: Después de la autenticación en el servidor el usuario deberá arrancar el servidor, caso contrario no existirá comunicación con el cliente.		

HISTORIA DE USUARIO		
ID: UH-04	**Usuario:** Administrador	
Programador Responsable: Luis Félix	**Sprint:** Usuario Administrador	**Número Sprint:** 2
Nombre Historia: Gestión de usuarios Administradores.	**Prioridad:** Alta	
Descripción: El usuario Administrador será el único que podrá crear, modificar y eliminar otros usuarios Administradores.		
Comentarios: No se limita el número de Administradores que pueda aceptar el sistema.		

HISTORIA DE USUARIO		
ID: UH-05	**Usuario:** Administrador	
Programador Responsable: Luis Félix	**Sprint:** Usuario Profesor	**Número Sprint:** 4
Nombre Historia: Gestión de usuarios Profesores.	**Prioridad:** Alta	
Descripción:		

El usuario Administrador será el único que podrá crear, modificar y eliminar usuarios Profesores.
Comentarios: Tomar en cuenta que un profesor pertenece a una sola universidad.

HISTORIA DE USUARIO		
ID: UH-06	**Usuario:** Administrador	
Programador Responsable: Luis Félix	**Sprint:** Abstracciones	**Número Sprint:** 3, 4
Nombre Historia: Gestión de abstracciones: universidad, facultad, carrera y materia.	**Prioridad:** Alta	
Descripción: El sistema requiere abstraer ciertos Objetos para la mejor organización del sistema y de la información, estos son universidad, facultad, carrera y materia. Se debe poder gestionar dichos objetos.		
Comentarios: Tomar en cuenta que los profesores pertenecerán a una universidad y los estudiantes a una carrera.		

HISTORIA DE USUARIO		
ID: UH-07	**Usuario:** Administrador, Profesor	
Programador Responsable: Luis Félix	**Sprint:** Abstracciones	**Número Sprint:** 4
Nombre Historia: Gestión de abstracciones: estudiante.	**Prioridad:** Alta	
Descripción: El sistema requiere abstraer el Objeto estudiante para la mejor organización del sistema y de la información. Se debe poder gestionar dicho objeto.		
Comentarios:		

Tomar en cuenta que un estudiante pertenece a una carrera, y toma varias materias, las cuales se organizan por paralelos.

HISTORIA DE USUARIO		
ID: UH-08	**Usuario:** Administrador	
Programador Responsable: Luis Félix	**Sprint:** Abstracciones	**Número Sprint:** 5
Nombre Historia: Gestión de abstracciones: paralelo.	**Prioridad:** Alta	
Descripción: El sistema requiere abstraer el Objeto paralelo para la mejor organización del sistema y de la información. Se debe poder gestionar dicho objeto.		
Comentarios: Tomar en cuenta cada paralelo estará asignado a un profesor ya que cada profesor puede dictar una materia, pero tener varios paralelos.		

HISTORIA DE USUARIO		
ID: UH-09	**Usuario:** Administrador, Profesor.	
Programador Responsable: Luis Félix	**Sprint:** Abstracciones	**Número Sprint:** 5
Nombre Historia: Asignar Estudiante a Paralelo.	**Prioridad:** Alta	
Descripción: El sistema debe poder asignar varios paralelos a un estudiante.		
Comentarios: Tomar en cuenta que un estudiante de una carrera puede tomar asignaturas de otras carreras.		

HISTORIA DE USUARIO	
ID: UH-10	**Usuario:** Administrador, Profesor.

Programador Responsable: Luis Félix	**Sprint:** Abstracciones	**Número Sprint:** 5
Nombre Historia: Gestión de abstracciones: actividad académica.	**Prioridad:** Alta	
Descripción: El sistema requiere abstraer el Objeto actividad académica para poder crear el día de clases en el que se va a tomar lista. Se debe poder gestionar dicho objeto.		
Comentarios: Cada actividad académica debe estar relacionada con un paralelo y deberá constar de una fecha.		

HISTORIA DE USUARIO		
ID: UH-11	**Usuario:** Administrador, Profesor.	
Programador Responsable: Luis Félix	**Sprint:** Tomar asistencia	**Número Sprint:** 7
Nombre Historia: Gestión de lista de estudiantes presentes.	**Prioridad:** Alta	
Descripción: El sistema debe permitir gestionar la lista de estudiantes presentes para una actividad académica.		
Comentarios: Para cada actividad académica existirá una lista de estudiantes que asistieron a dicha clase, esto servirá para poder cambiar una asistencia por falta o viceversa, si existe algún error en el reconocimiento facial.		

HISTORIA DE USUARIO		
ID: UH-12	**Usuario:** Administrador, Profesor.	
Programador Responsable: Luis Félix	**Sprint:** Entrenar	**Número Sprint:** 6
Nombre Historia:	**Prioridad:** Alta	

Adición de fotografías de estudiantes en base de fotografías.	
Descripción: El sistema debe permitir añadir a una base de fotografías, las fotografías de los estudiantes para que posteriormente se realice el entrenamiento del sistema	
Comentarios: Esta base de fotografías deberá estar organizada en carpetas según el estudiante, la aplicación cliente deberá poder enviar las fotografías a dicha base.	

HISTORIA DE USUARIO		
ID: UH-13	**Usuario:** Administrador, Profesor.	
Programador Responsable: Luis Félix	**Sprint:** Entrenar	**Número Sprint:** 6
Nombre Historia: Entrenamiento del sistema por estudiante.	**Prioridad:** Alta	
Descripción: Después de añadir las fotografías de una estudiante a la base de fotografías, de deberá poder realizar el entrenamiento del sistema para dicho estudiante.		
Comentarios: Esta acción de entrenamiento será bloqueante para el módulo de Toma de Asistencia. También se deberá poder realizar varios entrenamientos simultáneos.		

HISTORIA DE USUARIO		
ID: UH-14	**Usuario:** Administrador.	
Programador Responsable: Luis Félix	**Sprint:** Entrenar	**Número Sprint:** 6
Nombre Historia: Entrenamiento total del sistema.	**Prioridad:** Alta	
Descripción: Después de añadir las fotografías de una estudiante a la base de fotografías, de deberá poder realizar el entrenamiento total del sistema.		
Comentarios:		

Esta acción de entrenamiento será bloqueante para el módulo de Toma de Asistencia.

HISTORIA DE USUARIO		
ID: UH-15	**Usuario:** Administrador, Profesor.	
Programador Responsable: Luis Félix	**Sprint:** Tomar asistencia	**Número Sprint:** 7
Nombre Historia: Tomar Asistencia.	**Prioridad:** Alta	
Descripción: Se deberá crear un módulo para la toma de asistencia de estudiantes para una actividad académica, los métodos a utilizar deberán ser mediante reconocimiento facial o de forma manual.		
Comentarios: Se deberá controlar que un profesor solo pueda tomar asistencia de los paralelos a los que fue asignado y el administrador podrá realizar de cualquier paralelo. Las fotografías para el reconocimiento facial deberán cargarse desde la galería de imágenes del dispositivo móvil o desde la cámara del mismo.		

HISTORIA DE USUARIO		
ID: UH-16	**Usuario:** Administrador, Profesor.	
Programador Responsable: Luis Félix	**Sprint:** Reporte de asistencia	**Número Sprint:** 8
Nombre Historia: Generar Reporte de Asistencia.	**Prioridad:** Alta	
Descripción: Se deberá crear un módulo generar reportes de asistencia por paralelo, el reporte deberá ser un archivo Excel y deberá constar de la información de la materia, paralelo, profesor y la lista de estudiantes con su respectiva asistencia. Dicho archivo se deberá enviar mediante correo electrónico.		
Comentarios: Se deberá controlar que el usuario profesor solo generar reportes de los paralelos a los que fue asignado y el administrador podrá generar de cualquier paralelo.		

HISTORIA DE USUARIO		
ID: UH-17	**Usuario:** Administrador.	
Programador Responsable: Luis Félix	**Sprint:** Base de datos	**Número Sprint:** **1**
Nombre Historia: Base de datos.	**Prioridad:** Alta	
Descripción: Se deberá crear la base de datos que permita almacenar toda la información de la aplicación.		
Comentarios: Se deberá permitir que la base de datos se ubique en el mismo servidor o en otro		

ANEXO C

SCRIPT DE LA BASE DE DATOS

Debido a la extensión del ANEXO C, el script de la base de datos se muestra en el CD adjunto.

ANEXO D

CÓDIGO FUENTE DEL SERVIDOR

Debido a la extensión del ANEXO D, el código fuente del servidor se muestra en el CD adjunto.

ANEXO E

CÓDIGO FUENTE DEL CLIENTE

Debido a la extensión del ANEXO D, el código fuente del cliente se muestra en el CD adjunto.

ANEXO F

BASE DE IMÁGENES DE ESTUDIANTES.

Debido a la extensión del ANEXO F, una muestra de la base de imágenes de estudiantes se muestra en el CD adjunto.

ANEXO G

IMÁGENES DE ESTUDIANTES PRENSENTES EN ACTIVIDAD ACADÉMICA.

Debido a la extensión del ANEXO g, una muestra se muestra en el CD adjunto.

ANEXO H

REPORTES DE ASISTENCIA.

Debido a la extensión del ANEXO H, los reportes de asistencia se muestra en el CD adjunto**.**

ORDEN DE EMPASTADO