ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

IMPLEMENTACIÓN DE UN SISTEMA PROTOTIPO PARA LA GESTIÓN DE CITAS MÉDICAS Y REGISTRO DE HISTORIAS CLÍNICAS DE PACIENTES PARA EL CENTRO MÉDICO "JESÚS DE NAZARETH"

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

JISSELA JOHANA ARCOS MOLINA

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

GALO DAVID RUBIO AMAYA

DIRECTOR: ING. ANDRÉS FERNANDO REYES CASTRO MSc.

CODIRECTOR: ING. FRANKLIN LEONEL SÁNCHEZ CATOTA MSc.

Quito, mayo 2019

AVAL

Certificamos que el presente trabajo fue desarrollado por Jissela Johana Arcos Molina y Galo David Rubio Amaya bajo nuestra supervisión.

ING. ANDRÉS FERNANDO REYES CASTRO MSc.
DIRECTOR DEL TRABAJO DE TITULACIÓN

ING. FRANKLIN LEONEL SÁNCHEZ CATOTA MSc.
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Nosotros, Jissela Johana Arcos Molina y Galo David Rubio Amaya, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

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

JISSELA JOHANA ARCOS MOLINA	GALO DAVID RUBIO AMAYA

DEDICATORIA

Dedico mi trabajo a Dios, a mi mamá Guadalupe M., a mi papá Germán A., a mis hermanos Michelle, Gustavo y Anthony por ser esa inspiración y motivación de todos los días.

Jiss

A mi familia Galo, Chary, Jhastin y Leo ya que han sido mi principal inspiración y sin su apoyo incondicional no sería posible lograr esta meta.

Galo

AGRADECIMIENTO

Agradezco a Dios por todas las bendiciones que día a día derrama sobre mi vida. A mi mamá por llenar de luz los días grises con su amor, por su sacrificio para ayudarme a cumplir mis metas y por enseñarme que a pesar de todo siempre hay que salir adelante. A mi papá por su amor, amistad, apoyo incondicional, por su esfuerzo y trabajo para sacar a nuestra familia adelante.

Gracias a mi hermana Michelle por ser mi mejor amiga, confidente y en especial por confiar en mi incluso cuando yo no lo puedo hacer, a mis hermanos Gustavo y Anthony por ser ese motor para no darme por vencida. Los tres son los obsequios más bonitos que Dios me regaló.

Gracias Galo por tu apoyo, paciencia y amor en todos los proyectos que emprendemos juntos. Siempre estás ahí para recordarme que soy capaz de lograr cualquier cosa que me proponga y no me dejas darme por vencida.

Agradezco a mis tías Rosa e Hilda por ser esas hadas madrinas que sin su ayuda nada sería posible. A mis primas: Laura, Angelita, Daysi, Lady, Amparito y Carmita por cuidarme en los momentos más difíciles.

Gracias a mis amigos que son uno de los regalos más bonitos que la poli me obsequió: Krups, Gisse, Mary, Kathy, Mafer, Less, Majo, Dianita, David, Elvis, Alejo, Fausto, Pato, Edi y todos aquellos con los que compartí muchos años de complicidad y estudio.

Gracias a mi familia espiritual de Cristo Redentor por su paciencia, comprensión, apoyo y oraciones constantes en los momentos difíciles.

Y muchas gracias a todos los maestros de la poli quienes me compartieron sus conocimientos y su ayuda constante, en especial a mi director de tesis el Ing. Andrés Reyes y a mi codirector el Ing. Franklin Sánchez.

Jiss

Agradezco a mis padres Galo y Chary por ser mi ejemplo a seguir, por todos sus sacrificios, consejos los cuales no me han permitido desistir en todo este camino y los valores enseñados.

A mis hermanos Jhastin y Leo por ser mis amigos incondicionales, por todos los momentos que me apoyaron y todas las experiencias que vivimos juntos.

A mi novia Jiss quien me ha soportado casi toda la carrera, por los retos que juntos superamos, por el apoyo y el amor que me ha dado para culminar juntos esta etapa de mi vida.

A mis amigos con los cuales iniciamos este reto, Mary, Kathy, Sammy, Diana por todos los momentos compartidos, el apoyo tanto en lo académico como en lo personal.

A mis amigos de la carrera, Roberto, Liss, Jona, Vane, Alex, Edu por todos los buenos momentos pue hemos pasado, toda la ayuda brindada a lo largo de este tiempo.

A mis profesores Ing. Andrés Reyes e Ing. Franklin Sánchez por la guía, la ayuda prestada y los consejos que me han dado en esta etapa.

Galo

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE SEGMENTOS DE CÓDIGO

RESUMEN

El presente trabajo de titulación consiste en implementar un sistema prototipo para la gestión de citas médicas y registro de historias clínicas de pacientes para el centro médico “Jesús de Nazareth”. El cual se encuentra formado por: un cliente de escritorio, un cliente móvil y un servidor.

En primer lugar, se realizaron varias reuniones con el personal del Centro Médico para elaborar las historias de usuario de cada iteración. Posteriormente, usando estas historias de usuario se definieron los requerimientos funcionales y no funcionales. Después, se procedió a la etapa de diseño iniciando por la arquitectura del sistema, a continuación, se procedió a elaborar los distintos diagramas que permitieron definir los componentes y esquematizar la lógica del prototipo para proceder a realizar el diseño de las interfaces gráficas.

La siguiente etapa fue la implementación de cada iteración: se empezó con la codificación de los servicios utilizando la tecnología WCF. Luego se implementó el cliente de escritorio utilizando C# y XAML. Se realizaron pruebas locales para el servidor para posteriormente subirlo a Azure. Después se implementó el cliente móvil en Android Studio y se realizaron las pruebas del sistema completo.

El presente documento está constituido por cuatro capítulos: El primer capítulo detalla las diferentes herramientas y tecnologías utilizadas. El segundo capítulo corresponde a la metodología y consta del diseño e implementación. El tercer capítulo contiene el resultado y análisis de las pruebas realizadas en cada iteración. Finalmente se enumeran las conclusiones y recomendaciones.

PALABRAS CLAVE: Azure, WCF, Android, Visual Studio, historias clínicas, citas médicas

ABSTRACT

The present project consists of implementing a prototype system for the management of medical appointments and the registration of patient medical records for the medical center "Jesus of Nazareth". The prototype is formed by: a desktop client, a mobile client and a server.

In the first place, several meetings were held with the Medical Center staff to elaborate the user stories for each iteration. Subsequently, using these user stories, the functional and non-functional requirements were defined. Then, the systems architecture were designed, in order to define the components and outline the prototype's logic and to develop the initial design of the graphic interfaces.

The next stage was the implementation of each iteration, started with the coding of services that use WCF technology. Then the desktop client was implemented using C# and XAML. Local tests were performed for the server to later upload it to Azure. After implementing the mobile client in Android Studio, the performance test were performed.

This document consists of four chapters: The first chapter details the different tools and technologies used. The second chapter corresponds to the methodology and consists of the design and implementation. The third chapter contains the result and analysis of the tests performed in each iteration. Finally, the conclusions and recommendations are listed.

KEYWORDS: Azure, WCF, Android, Visual Studio, medical records, medical appointments

1. INTRODUCCIÓN

Los centros médicos tradicionalmente han manejado su información mediante registros escritos, lo cual dificulta su almacenamiento y procesamiento. En la actualidad, esta información se ha incrementado en cantidad y variedad, motivo por el cual, es necesario la automatización del proceso del registro de estos datos.

El centro médico “Jesús de Nazareth” ofrece diferentes tipos de servicio a la comunidad: medicina general, ginecología, odontología, psicología clínica, psicología infantil, traumatología, enfermería y laboratorio clínico. En los últimos años se ha incrementado el número de clientes y esto ha dificultado la asignación de citas, debido al cruce de horarios tanto de pacientes y de profesionales de la salud. Además, estos últimos requieren que se realice una historia clínica integral de los pacientes para poder realizar diagnósticos más acertados y debido a la cantidad de médicos que laboran en el centro la duplicidad de historias clínicas en distintos consultorios. Otra dificultad que se presenta para los pacientes es la necesidad de asistir personalmente al centro de salud o realizar una llamada telefónica para agendar una cita médica.

El presente Proyecto Técnico tiene la finalidad de implementar un sistema que gestione citas médicas, automatizando el proceso de asignación de citas en el centro médico. Por otra parte, permitirá digitalizar y centralizar la información de historias clínicas de los pacientes para su posterior uso en los distintos consultorios.

1.1 OBJETIVOS

El objetivo del presente trabajo de titulación es el de implementar un sistema prototipo para la gestión de citas médicas y registro de historias clínicas de pacientes para el centro médico “Jesús de Nazareth”.

Los objetivos específicos son:

- Analizar las diferentes herramientas y tecnologías que permitirán definir el contexto sobre el cual se desarrollará e implementará el sistema.
- Diseñar cada uno de los elementos que compone el sistema prototipo propuesto.
- Implementar cada uno de los elementos diseñados.
- Validar el sistema prototipo en un ambiente local.

1.2 ALCANCE

El presente Proyecto Técnico propone el desarrollo e implementación de un sistema de registro de historias clínicas de pacientes de manera centralizada y gestión citas médicas para las distintas especialidades del centro médico “Jesús de Nazareth”.

El sistema se lo realizará con un modelo orientado a los servicios, el mismo que poseerá tres componentes principales: Servidor, Cliente de escritorio y Cliente Móvil, como se observa en la Figura 1.1.

Primer componente

El Servidor se lo desarrollará en Microsoft Visual Studio utilizando servicios web WCF [1] con lenguaje de programación C# [2], el cual se lo implementará en la plataforma informática en la nube Microsoft Azure [3]. Para el almacenamiento de datos se utilizará SQL *Server* [4].

Módulos del servidor:

- **Primer módulo:** Login el cual permitirá a los usuarios ingresar al sistema, el cual consta de cinco perfiles: Administrador, Recepcionista, Médico, Laboratorio clínico y Paciente.
- **Segundo módulo:** Administración de usuarios el cual permitirá realizar acciones CRUD (crear, leer, actualizar y eliminar) [5] a todos los usuarios del sistema.
- **Tercer módulo:** Administración de citas el cual permitirá realizar acciones CRUD a las citas médicas dependiendo del perfil de usuario del sistema.
- **Cuarto módulo:** Administración de historias clínicas el cual permitirá realizar acciones CRUD a las historias clínicas dependiendo del perfil de usuario del sistema.
- **Quinto módulo:** Administración de exámenes médicos el cual permitirá realizar acciones CRUD a los exámenes médicos dependiendo del perfil de usuario del sistema.
- **Sexto módulo:** Notificaciones por correo el cual permitirá enviar correos electrónicos a médicos y pacientes indicando las citas previamente asignadas.
- **Séptimo módulo:** Recuperación de contraseña el cual permitirá a los usuarios cambiar o recuperar la contraseña de los usuarios del sistema.

Segundo componente

El cliente de escritorio se implementará dentro del centro médico, se lo realizará en Microsoft Visual Studio con WPF (*Windows Presentation Foundation*) [6] con el lenguaje de programación C#.

Módulos del cliente de escritorio.

- **Primer módulo:** Corresponde a la interfaz de *Login* el cual se ingresará las credenciales de usuario y contraseña para acceder al sistema.
- **Segundo módulo:** Administrador, el cual consta de las interfaces que le permiten al administrador realizar acciones CRUD a los usuarios del sistema.
- **Tercer módulo:** Recepción, consta de las interfaces que permitirán crear y eliminar nuevos pacientes y médicos; modificar su perfil personal; visualizar los médicos disponibles; agendar, imprimir y enviar por correo la información de las citas; recibir y confirmar solicitudes de citas y enviar la notificación hacia la aplicación móvil.
- **Cuarto módulo:** Médicos el cual consta de las interfaces que permiten modificar su perfil personal; visualizar los pacientes asignados y la agenda para la semana; realizar acciones CRUD de fichas médicas e imprimir en formato PDF; agendar citas de interconsulta.
- **Quinto módulo:** Laboratorio clínico el cual consta de las interfaces que permitirán modificar su perfil personal; mostrar las listas de pacientes y una lista de los resultados de los exámenes por entregar; mostrar todos los exámenes que se pueden realizar en el laboratorio y a través de un *checkbox* seleccionar los exámenes que el paciente se realizará; permitir ingresar los resultados de los exámenes e imprimir dichos resultados.

Tercer componente

El cliente móvil se creará para el uso exclusivo de los pacientes y médicos del Centro Médico. Se lo desarrollará para la plataforma Android utilizando Android Studio[7] en su última versión, juntamente con el lenguaje de programación Java y el lenguaje de Marcado Extensible XML (*Extendible Markup Language*) [8].

- **Primer módulo.** Login el cual consta con las interfaces que permitirán acceder a usuarios registrados previamente en el sistema; realizar la recuperación o cambio de contraseña.
- **Segundo módulo.** Interfaces que permitirán leer y actualizar los datos personales del usuario.
- **Tercer módulo.** Dispondrá de un calendario semanal, mostrando las citas asignadas al usuario y los horarios disponibles para agendar una cita en los diferentes consultorios que posee el centro médico. Permitirá ver la información de una cita asignada, además solicitar una cita con los horarios disponibles. En caso de que la solicitud sea aceptada o rechazada llegará una notificación. Además, se exportarán a Google Calendar las citas asignadas.

A continuación, en la Tabla 1.1 se describen los formularios que se implementarán en cada uno de los componentes del sistema.

Tabla 1.1. Descripción de los módulos del sistema.

Cliente de Escritorio	**MODULO 1**
	Formulario 1: Permite el acceso a los usuarios del sistema.
	MODULO 2
	Formulario 2: Muestra los usuarios existentes en el sistema.
	Formulario 3: Tipo de formulario CRUD para usuarios del sistema.
	MODULO 3
	Formulario 4: Se mostrará un calendario de la semana actual, mostrando las agendas de cada consultorio en donde se podrá revisar los detalles de cada cita.
	Formulario 5: Corresponde a un CRUD de citas, además permitirá enviar e imprimir las citas en formato PDF.
	Formulario 6: Corresponde a un CRUD de pacientes.
	MODULO 4
	Formulario 7: Se visualizará la agenda semanal del profesional de la salud correspondiente.
	Formulario 8: Es un CRU de fichas médicas e impresión en formato PDF.
	MODULO 5
	Formulario 9: Mostrará las listas de pacientes y una lista de los resultados de los exámenes por entregar.
	Formulario 10: Mostrará todos los exámenes que se pueden realizar en el laboratorio y a través de un checkbox seleccionar los que el paciente se realizará.
	Formulario 11: Permitirá ingresar los resultados de los exámenes e imprimir dichos resultados.
Cliente Móvil	**MODULO 1**
	Formulario 1: Permitirá el acceso a usuarios registrados previamente en el sistema
	Formulario 2: Permitirá modificar la contraseña.
	MODULO 2
	Formulario 3: Lee y actualiza los datos personales
	MODULO 3
	Formulario 4: Dispondrá de un calendario semanal, mostrando las citas asignadas al usuario y los horarios disponibles para agendar una cita en los diferentes consultorios que posee el centro médico.
	MODULO 4
	Formulario 5: Corresponderá a un CRUD de citas médicas.
	MODULO 5
	Formulario 6: Mostrará la información de las historias clínicas.

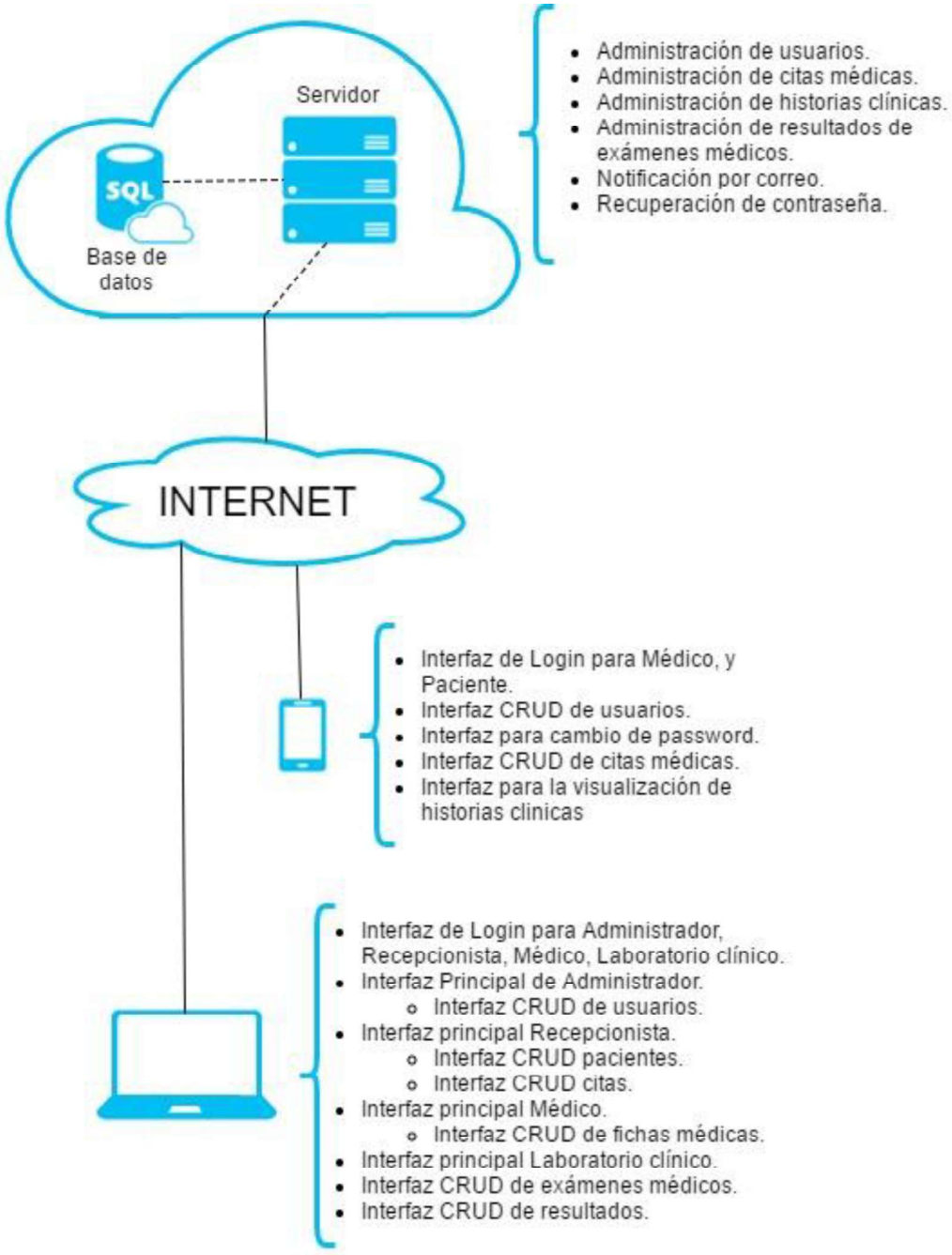


Figura 1.1. Diagrama del sistema prototipo propuesto

1.3 MARCO TEÓRICO

1.3.1 SERVICIOS EN LA NUBE

Los servicios en la nube es un paradigma que ofrece servicios de computación e informática a través de Internet, en donde no es necesario que el usuario conozca la

infraestructura que existe detrás de la computación en la nube, porque pasa a ser una abstracción, “una nube”, en la cual los servicios y aplicaciones crecen con facilidad, a una gran velocidad y existen pocos errores de funcionamiento [9]. En la Figura 1.2 se muestra la arquitectura de la computación en la nube la cual consta de varios servicios (servidores, máquinas virtuales, plataformas de *software*, aplicaciones y almacenamiento de datos) que son accesibles a través de distintos dispositivos (*routers, switches*, computadoras, *smartphones*, entre otros).

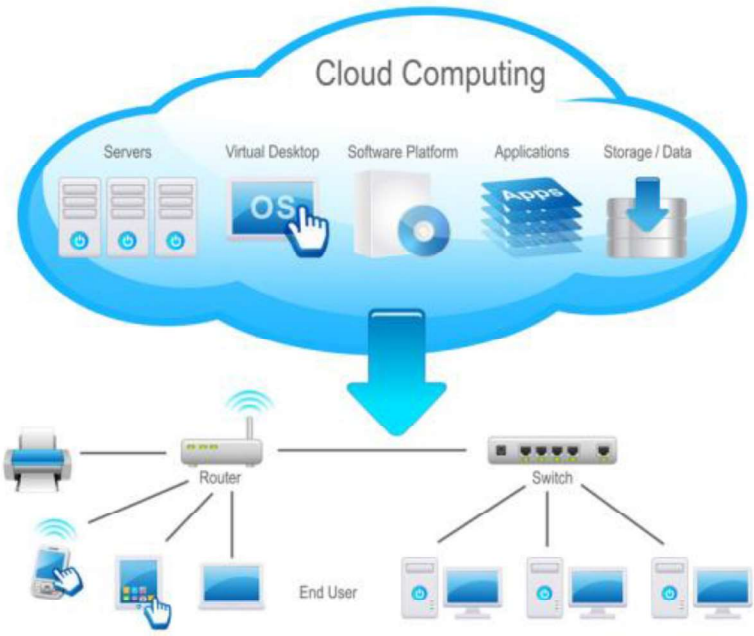


Figura 1.2. Arquitectura de Servicios en la nube [9]

La computación en la nube fundamenta su arquitectura en tres capas, las cuales se visualizan en la Figura 1.3. La infraestructura como Servicio (IaaS-*Infrastructure as a Service*) es un servicio en el cual una organización tiene sus equipos de *hardware*, servidores y otros componentes de red fuera de ella. Todos los problemas relacionados con los equipos deben ser solucionados por el proveedor [10]. La Plataforma como Servicio (PaaS-*Plataform as a Service*) es una plataforma la cual permite a los desarrolladores crear y ejecutar aplicaciones, para la cual se debe tener en cuenta los lenguajes de programación compatibles. Adicional el cliente no necesita instalar, configurar ni dar mantenimiento a sistemas operativos, bases de datos y servidores de las aplicaciones. *Software* como servicio (SaaS-*Software as a Service*) son aplicaciones que no necesitan ser instaladas, ni configuradas, ni realizar el debido mantenimiento de esta en un equipo, el usuario final paga por el alquiler del uso de una distribución de *software* específica.

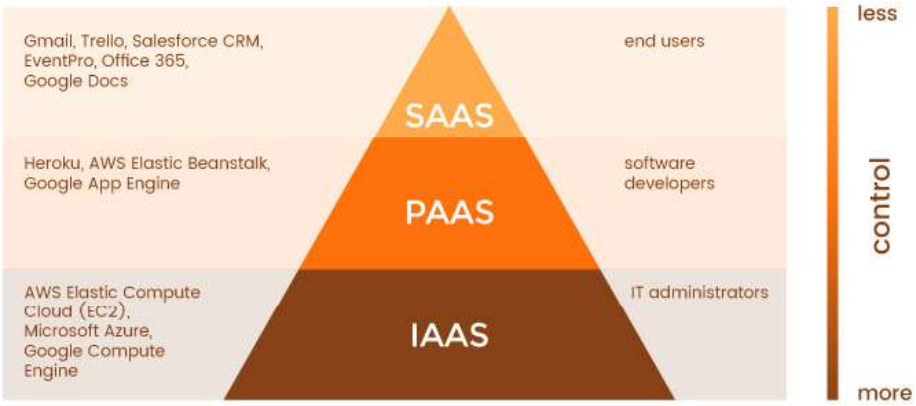


Figura 1.3. Iaas, PaaS, SaaS [11]

1.3.1.1 Ventajas de los servicios en la nube

- Ahorro en adquisición de equipos: no es necesario instalar equipos complejos de hardware, para su funcionamiento los servicios en la nube sólo requieren la instalación de equipos terminales.
- Rapidez en la implementación: no es necesario esperar mucho tiempo para la implementación de las soluciones en las plataformas informáticas.
- Actualizaciones automáticas: se tiene actualizaciones automáticas conservando el trabajo previamente realizado.
- Portabilidad de la información: Un *software* puede ejecutarse en diferentes plataformas o sistemas operativos, los clientes finales utilizan este concepto de manera masiva, al utilizar un *software* tanto en computadoras como en celulares inteligentes.
- Ahorro energético: Al utilizar los servicios en la nube se tiene un ahorro global de energía al disminuir el uso de *hardware* en los sistemas implementados.

1.3.1.2 Desventajas de los servicios en la nube

- Seguridad de la información: al utilizar servicios en la nube toda la información es almacenada y el proveedor puede tener acceso a la misma. Lo cual implica que cualquier fuga de información puede ocasionar un gran impacto negativo.
- No existe disponibilidad sin Internet: La calidad de cobertura de Internet debe ser buena y constante para no tener fallas al momento de utilizar las aplicaciones y servicios en la nube.
- Problemas en la escalabilidad: El uso masivo de estos servicios obliga a los proveedores a dimensionar correctamente su red para que esta tenga un

crecimiento óptimo a las necesidades del mercado y sin provocar degradaciones en los servicios.

1.3.2 PLATAFORMAS INFORMÁTICAS EN LA NUBE

Las plataformas informáticas proveen servicios informáticos como servidores, espacios de almacenamiento, bases de datos, redes, *software*, análisis e inteligencia a través del Internet. Las dos plataformas más utilizadas en la actualidad son Microsoft Azure y AWS (*Amazon Web Service*).

1.3.2.2 AWS

AWS ofrece un extenso conjunto de productos basados en la nube, aplicaciones para cómputo, almacenamiento, bases de datos, así como diversas herramientas para desarrolladores, para administración y seguridad. A continuación, en la Tabla 1.2 y la Tabla 1.3, se muestran los productos que ofrece AWS.

Tabla 1.2. Productos ofrecidos por AWS (parte 1 de 2)

PRODUCTOS	DESCRIPCIÓN
Análisis	Servicio de búsqueda administrada, transmisión de datos en tiempo real, almacenamiento de datos
Integración de aplicaciones	Coordinación de aplicaciones distribuidas, notificaciones push para aplicaciones móviles y SMS, administración de colas de mensajes.
Realidad aumentada y virtual	Creación y ejecución de aplicaciones de realidad virtual y aumentada
Aplicaciones empresariales	Chats, videollamadas y reuniones. Emails y calendarios empresariales protegidos y administrados
Informática	Servidores virtuales en la nube, ejecución y administración de aplicaciones web,
Bases de datos	Servicios de bases de datos relacionales administradas para MySQL, PostgreSQL, Oracle, SQL Server y MariaDB

Tabla 1.3. Productos ofrecidos por AWS (parte 2 de 2)

PRODUCTOS	DESCRIPCIÓN
Herramientas para desarrolladores	Escritura y compilación de códigos, desarrollo e implementación de nuevas aplicaciones AWS.
Game Tech	Alojamiento para servidores de videojuegos.
Internet de las cosas	Conexión de dispositivos a la nube, sistema operativo compatible con IoT para microcontroladores
Soluciones móviles	Creación e implementación de aplicaciones web y móviles

1.3.2.3 Microsoft Azure

Es una plataforma de servicios en la nube que se encuentra alojada en los centros de procesamiento de Microsoft. Permite la creación, implementación y administración de aplicaciones. Además, permite tener una comunicación segura y asociación entre aplicaciones [3]. Entre los servicios, ofrece Infraestructuras como Servicios (IaaS); algunos ejemplos son los servidores de almacenamiento, máquinas virtuales, pagando solamente por el uso y dependiendo de la demanda. Otro servicio son las Plataformas como Servicios (PaaS); bases de datos de alta disponibilidad SQL, CMS (*Content Management System*) para desarrollo de web, *backend* para aplicaciones móviles, evitando el gasto y la complejidad de la adquisición de licencias. Y el *Software* como Servicio (SaaS) como por ejemplo el correo electrónico, los calendarios y las herramientas ofimáticas. En la **¡Error! No se encuentra el origen de la referencia.** se puede observar la diferenciación entre IaaS, PaaS y SaaS en Microsoft Azure. En la Tabla 1.4 se muestra los productos que ofrece Azure con una breve descripción.

Figura 1.4. Servicios ofrecidos por Microsoft Azure [3]

Tabla 1.4. Productos ofrecidos por Microsoft Azure [3]

PRODUCTO	DESCRIPCIÓN
Máquinas Virtuales	Aprovisionamiento de máquinas virtuales de Windows y Linux.
SQL Database	Permite la migración de bases de datos SQL relacionales administradas como un servicio
Azure Cosmos DB	Proporciona bases de datos multimodelo distribuida globalmente para cualquier escala
Cognitive Services	Agrega funcionalidad de API inteligentes para habilitar interacciones contextuales
Windows Virtual Desktop	Proporciona una experiencia de un escritorio virtual en cualquier dispositivo que pueda acceder a la nube
App Service	Permite la creación eficaz de aplicaciones en la nube para Web y dispositivos móviles
Azure Functions	Eventos de proceso con código sin servidor
Blockchin Workbench	Conecta una cadena de bloques a la nube sin ocuparse del trabajo pesado
Machine Learning	Permite crear, entrenar e implementar modelos desde la nube hasta el perímetro

En el siguiente trabajo se hace uso de los siguientes servicios de Microsoft Azure: *App Service* y *SQL Database*. La primera debido a que la comunicación del prototipo se va a realizar a través de servicios web y la segunda porque nos da una base de datos SQL

Server con las características que se adaptan a los requerimientos del sistema. Se eligió esta plataforma porque la mayoría de las herramientas utilizadas en el presente proyecto de titulación son desarrolladas por Microsoft, al igual que Azure, adicional es la opción más económica porque los precios que ofrece son inferiores a AWS [12].

1.3.3 HERRAMIENTAS DE DESARROLLO SELECCIONADAS

A continuación, se describen las herramientas de desarrollo seleccionadas para la implementación del sistema prototipo. El sistema de escritorio se desarrolló para el sistema operativo Windows, porque la gran cantidad de las herramientas elegidas para la implementación del prototipo son desarrolladas por *Microsoft.* En primer lugar, *Visual Studio*, su lenguaje de programación C# y las diferentes tecnologías utilizadas para la configuración del servidor. Después se describe el entorno de desarrollo de Android Studio y el lenguaje de programación para el desarrollo de la aplicación móvil. A continuación, se indica el formato de texto a utilizarse para intercambio de datos entre plataformas diferentes. Finalmente se describen los métodos de programación ágil.

1.3.3.2 Visual Studio

Es un entorno de desarrollo integrado para el sistema operativo Windows, permite escribir código de manera precisa y eficiente, soporta múltiples lenguajes de programación: C++, C#, Visual Basic, Java, entre otros. Adicional tiene nuevas capacidades online con la integración de Microsoft Azure [13].

- **C#**

Es un lenguaje de programación orientada a objetos desarrollado por Microsoft basado en C, muy sencillo con menos de cien palabras clave y una docena de tipos de datos incorporados, pero es altamente expresivo al momento de implementarlo. Además, permite manejar excepciones para la detección y recuperación de errores que pueden surgir en el código. La última versión permite tener compatibilidad con LINQ, lo que ahorra líneas de código, también soporta completamente la sintaxis de WPF que permite la creación de aplicaciones multiplataforma [14]. En la Figura 1.5 se puede visualizar un ejemplo de código de C# el cual imprime en consola el mensaje *Hellow World!.*

```
class Hello
{
    static void Main(string[] args)
    {
        // Use the system console object
        System.Console.WriteLine("Hello World!");
    }
}
```

Figura 1.5. Ejemplo C# [14]

- **WPF**

WPF es un programa desarrollado por Microsoft que permite el desarrollo de interfaces de manera atractiva y sofisticadas con fácil interacción con animación, video o documentos, es decir está destinado a proporcionar una API (*Application Programming Interface*) para aplicaciones web o de escritorio [15]. Sus principales características son:

- ✓ Interfaz gráfica declarativa que crea interfaces de usuario utilizando el lenguaje XAML (*eXtensible Application Markup Language*).
- ✓ Diseño dinámico de las interfaces al momento de trabajar con diferentes resoluciones, tamaños y otras características de la pantalla, WPF lo configura de manera dinámica.
- ✓ Los gráficos en WPF se basan en vectores, por lo que las imágenes usan menos espacio, son escalabilizables sin perder la calidad.
- ✓ Permite a los desarrolladores enlazar y manipular datos dentro de las aplicaciones.

- **XAML *(Extensible Application Markup Languaje)***

XAML es una de las diferentes tecnologías que aparecieron con Windows Vista, como un mecanismo para definir las interfaces de usuario [16]. Es un lenguaje declarativo de Microsoft, proporciona una sintaxis fácilmente extensible y localizable para definir interfaces de usuario separadas de la lógica de aplicación[17]. XAML fue diseñado para soportar clases y métodos de la plataforma de desarrollo .NET. En la Figura 1.6 se tiene un ejemplo del lenguaje XAML y su respectiva interfaz gráfica en la Figura 1.7 en este ejemplo se tiene un botón, asignado el nombre de "*button*", cuyo contenido es la frase "*Hello, world*!", tiene una alineación horizontal hacia la izquierda y vertical hacia arriba, también se define el margen para ubicarlo en la interfaz gráfica.

```
<Button  x:name="button"  Content="Hello,  world!"  HorizontalAlignment="Left"
Margin = "152,293,0,0" VerticalAlignment="Top"/>
```

Figura 1.6. Ejemplo de XAML [18]

Figura 1.7. Interfaz gráfica de un botón con la frase *Hello, world!* [18]

- **WCF (*Windows Communication Foundation*)**

WCF es un *framework*, diseñado para desarrolladores de *software* y web, para crear aplicaciones orientadas a servicios. Usando WCF, se puede enviar datos como mensajes asíncronos de un punto final de servicio a otro [19]. Un punto final puede ser un cliente de un servicio que solicita datos de un punto final de servicio [20]. WCF permite a los desarrolladores crear, alojar, consumir y asegurar servicios utilizando la plataforma de Microsoft. Los mensajes pueden ser tan simples como un solo carácter, una palabra o tan complejos como un flujo de datos binarios. Como indica en [21] estos mensajes tienen un formato de acuerdo con el contrato con en el que tanto el cliente como el servicio están de acuerdo. Para una comunicación exitosa entre el cliente y el servicio, ambas partes deben conocer la Dirección del servicio, la vinculación y los contratos comúnmente conocidos como ABCs de WCF como se observa en la Figura 1.8.

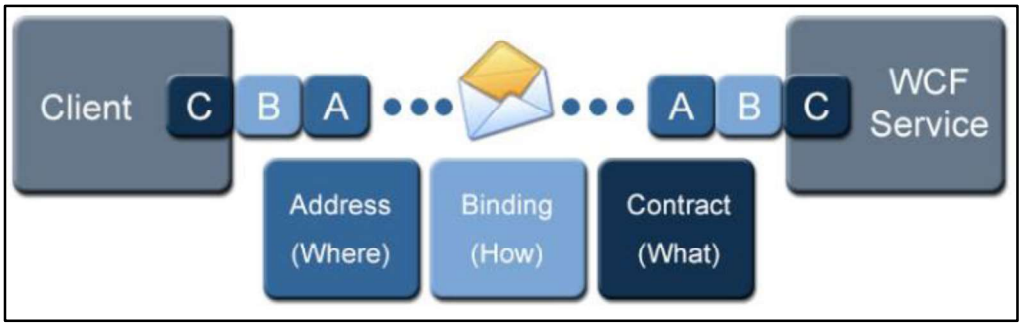


Figura 1.8. Envío de mensajes en WCF [19]

- **LINQ (*Language Integrated Query*)**

LINQ son un conjunto de herramientas desarrolladas por Microsoft que permiten realizar todo tipos de consultas de diferentes fuentes de datos, como, por ejemplo: xmls, bases de datos, etc [22]. Para lograr esto tiene funciones propias para unificar las operaciones más comunes de todos los entornos, es decir se tiene un mismo lenguaje para todas las tareas que se realizan con datos. En C# la sintaxis de escritura de código es similar a SQL, pero se tiene como ventaja la potencia de .net y Visual Studio. En la Figura 1.9 se tiene un ejemplo de una consulta LINQ en C#, en la cual se accede a una colección y se filtran todos los elementos en los cuales la propiedad 1 sea verdadera.

```
var lista = from c in coleccion
            where c.propiedad1 == true
            select c;
```

Figura 1.9. Ejemplo de código de LINQ en C#

1.3.3.3 Android Studio

Es un entorno de desarrollo integrado para el desarrollo de aplicaciones Android, ofrece las siguientes características [23]:

- Un sistema de compilación basado en *Gradle* flexible
- Emulador rápido con varias funciones
- Un entorno unificado en el que se puede realizar proyectos para todos los dispositivos Android
- Reenderizado en tiempo real
- Plantillas para crear diseños comunes en Android
- Consejos de optimización
- Un dispositivo virtual que permite ejecutar y probar aplicaciones Android
- Una de las mayores fortalezas de la plataforma Android es que aprovecha el lenguaje de programación Java y sus librerías

- **Java**

Java es un lenguaje de programación y una plataforma informática, que ha ganado rápidamente aceptación, debido a que es totalmente orientado a objetos, la independencia de plataforma y la posibilidad de trabajo en red [24]. Los programas desarrollados en Java se compilan en el código de bytes de arquitectura neutra y se pueden ejecutar en cualquier plataforma con un intérprete de Java, por lo que tiene una alta portabilidad. En la Figura 1.10 se observa un ejemplo de código de Java el cual imprime en consola el mensaje *Hola Mundo.*

```
1. public class HolaMundo {
2.
3.         public static void main(String[] args) {
4.                 System.out.println("Hola Mundo");
5.         }
6.
7. }
```

Figura 1.10. Ejemplo Java [25]

- **XML**

XML es un lenguaje de marcado de propósito general. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet. permite representar información estructurada en la web (todos documentos), de modo que esta información pueda ser almacenada, transmitida, procesada, visualizada e impresa, por diversos tipos de aplicaciones y dispositivos [26].

1.3.3.4 Bases de datos

Es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización [27]. Un SGBD (Sistema de Gestión de Base de Datos) es un *software* que permite a los usuarios definir, crear, mantener y controlar el acceso a la base de datos; permite a los usuarios definir la base de datos, mediante un lenguaje DDL (*Data Definition Languaje*) y especificar un lenguaje DML (*Data Manipulation Languaje*) que permita insertar, borrar, actualizar, eliminar y extraer información de la base de datos y proporciona un mecanismo de consulta de datos.

- **SQL**

Es un lenguaje de base de datos que permite al usuario: crear la base de datos y las estructuras de relación, realizar tareas de gestión de datos (inserción, modificación y eliminación de datos) y ejecutar consultas simples como complejas [27]. SQL tiene dos componentes principales:

- ✓ Un DDL, para definir la estructura y controlar el acceso de los datos.
- ✓ Un DML, para extraer y actualizar los datos.

Un ejemplo de consulta en SQL se observa en la Figura 1.11 en el cual se utiliza la sentencia *SELECT* para recuperar información de la base de datos

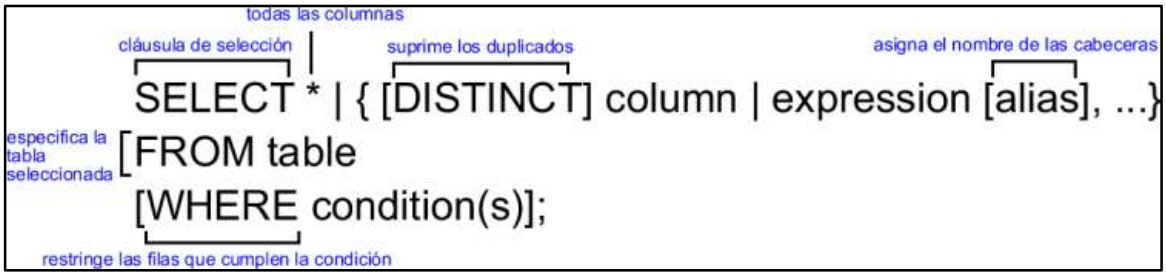


Figura 1.11. Ejemplo de una consulta SQL [27]

- **Microsoft SQL Server**

Microsoft SQL Server es un SGBD de modelo relacional desarrollado por Microsoft sus características principales son:

- Soporta procedimientos almacenados
- Posee un entorno gráfico para administración de datos, permitiendo el uso de comandos DDL y DML de manera gráfica
- Soporta el modo de trabajo cliente-servidor, es decir almacena la información en el servidor y los clientes sólo pueden acceder a dicha información

1.3.3.5 JSON (*JavaScript Object Notation*)

JSON es un formato de texto para intercambio de datos entre plataformas diferentes. JSON está basado en la notación de los objetos en JavaScript, pero no es necesario aprender JavaScript para usar JSON [28]. Es importante su uso porque es generado y leído en diversos ambientes de programación. En la Figura 1.12 se puede observar el formato de un objeto JSON que contiene nombres, un número y un arreglo.

```
{
  "name": "Molecule Man",
  "age": 29,
  "secretIdentity": "Dan Jukes",
  "powers": [
    "Radiation resistance",
    "Turning tiny",
    "Radiation blast"
  ]
},
```

Figura 1.12. Ejemplo JSON [29]

1.3.4 MÉTODOS ÁGILES PARA EL DESARROLLO DE SOFTWARE

Los métodos ágiles para el desarrollo de *software* están basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad del proyecto. En la Figura 1.13 se ilustra un modelo general del proceso para el desarrollo incremental y como se puede observar este proceso implica producir y entregar el *software* en incrementos y no en un solo paquete, cada iteración del proceso produce un nuevo incremento del *software* [30].

Al adoptar un enfoque incremental para el desarrollo de *software* se tiene como ventaja la entrega acelerada de las funcionalidades de mayor prioridad a los clientes y el compromiso del cliente con el sistema. En primer lugar, es importante definir todos los productos entregables del sistema, para obtener todos los requerimientos tanto funcionales como no funcionales del sistema y que los desarrolladores puedan diseñar una arquitectura del sistema, que incluye los diagramas de bases de datos, de casos de uso y todo lo que permita estructurar el diseño. A continuación, se especifica qué incremento se va a añadir al sistema, para posteriormente pasar al desarrollo de éste. Posteriormente se realiza las pruebas unitarias para validar el incremento y seguidamente integrarlo al sistema, se realizan las pruebas de integración respectivas para su respectiva validación. Todo este proceso se repite hasta que el sistema se encuentre completo.

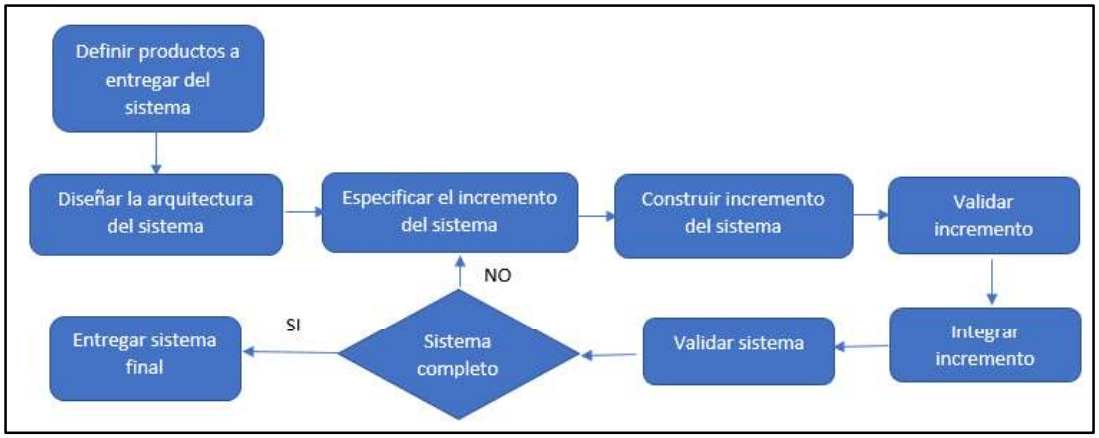


Figura 1.13. Proceso general de desarrollo iterativo [30]

Todos los métodos ágiles se basan en el desarrollo y entrega incrementales; pero proponen diferentes procesos para la realización del *software*; sin embargo, tienen en común un conjunto de principios los cuales se describen en la
Tabla **1.5** y la Tabla 1.6.

Tabla 1.5. Principios de los métodos ágiles (parte 1 de 2)

Principio	Descripción
Participación del cliente	Los clientes deben estar involucrados en todo el proceso de desarrollo, para proporcionar y priorizar nuevos requerimientos del sistema y realizar la evaluación de las iteraciones de este.
Entrega incremental	El sistema se desarrolla en incrementos, el cliente define los requerimientos que se deben añadir en cada incremento

Tabla 1.6. Principios de los métodos ágiles (parte 2 de 2)

Principio	Descripción
Personas, no procesos	Los miembros del equipo de trabajo no tendrán procesos formales, ellos desarrollarán el *software* con sus propios métodos.
Aceptar el cambio	Existirán cambios en los requerimientos del sistema.
Mantener la simplicidad	Debe existir simplicidad tanto en el *software* a desarrollar como en el proceso de desarrollo.

1.3.4.1 Metodología de desarrollo de *software Xtreme Programming* (Programación Extrema)

Xtreme Programming (XP) es el método de desarrollo de *software* ágil más utilizado, en donde todos los requerimientos se expresan como escenarios, los cuales se denominan historias de usuarios.Las características que indica [30] de esta metodología se describen a continuación:

- ✓ **Planificación incremental:** Los requerimientos se registran y se definen el tiempo de entrega dependiendo de la prioridad de las historias de usuario.

- ✓ **Entregas parciales:** Las entregas del sistema deben ser frecuentes e incrementales.
- ✓ **Sencillez en el diseño:** Sólo se realiza el diseño necesario para cumplir con los requerimientos actuales.
- ✓ **Programación en parejas:** Los desarrollos trabajan en parejas, se sientan juntos en la estación de trabajo para el desarrollo del *software* y cada uno verifica el trabajo del otro, proporcionando la ayuda necesaria.
- ✓ **Refactorización:** Es un proceso de mejora de *software* que refactoriza el código continuamente cada vez que se encuentren mejoras de este o cambios de requerimientos para implementarlos inmediatamente.
- ✓ **Cliente presente:** El cliente (usuario final) debe formar parte del equipo de trabajo de desarrollo del *software* y su obligación es formular y actualizar los requerimientos del *software*.
- ✓ **Desarrollo previamente probado:** Se escriben pruebas para nuevas funcionalidades antes de que éstas se incrementen.
- ✓ **Propiedad colectiva**: Todos los miembros del equipo deben trabajan en todas las áreas del sistema y pueden modificar cualquier cosa.

La técnica utilizada por XP para especificar los requisitos del sistema son las historias de usuario: son tarjetas en las cuales el cliente realiza una descripción breve de las características que desea que el sistema tenga, estos deben ser requisitos funcionales como no funcionales [31]. Por otra parte, el tratamiento de las historias de usuario debe ser flexible y dinámico para que puedan añadirse o quitarse características en las diversas iteraciones. No existe una plantilla definida para las historias de usuario, entre los datos importantes que debe contener se encuentra la fecha, número de historia, la prioridad técnica y una descripción de los requerimientos, en esta última característica la granularidad depende la complejidad del sistema.

XP se apoya de las siguientes fases:

- ✓ **Planificación del proyecto**: Se definen las historias de usuario con la participación del cliente, se las clasifica para establecer las iteraciones y elaborar el plan de entrega.
- ✓ **Diseño:** Se elaboran diseños simples y sencillos. Adicional, se contempla el diseño del diagrama de clases y de la base de datos.
- ✓ **Codificación:** En esta etapa los programadores deben adaptarse a los estándares de codificación creados, y elaborar un código consistente, comprensible y escalable

- ✓ **Pruebas:** Se realizan las pruebas unitarias y de aceptación, las primeras las realizan los desarrolladores, las últimas se realizan en conjunto con el cliente para probar que el sistema cumpla con los requerimientos de las historias de usuario.

Todas las fases descritas anteriormente son iterativas, de tal manera que si se tiene un nuevo requerimiento se genera una nueva historia de usuario.

2. METODOLOGÍA

Para el desarrollo del *software* del presente Proyecto Técnico se utilizó la metodología ágil XP, la cual es apropiada para el desarrollo de sistemas orientados a servicios con equipos medianos o pequeños y es capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto. Su filosofía es satisfacer completamente las necesidades del cliente, para lo cual lo integra como una parte del equipo de desarrollo. El ciclo de vida de un proyecto XP es muy dinámico, se puede separar en las siguientes fases: diseño, codificación y pruebas, las cuales se basan en historias de usuario. Por lo expuesto anteriormente se ha considerado esta metodología como la mejor opción para el desarrollo del *software* en el presente Proyecto Técnico. En este capítulo en primer lugar se presenta el diseño y la implementación del sistema.

2.1 DISEÑO

En este apartado se detalla el diseño del sistema prototipo para la gestión de citas médicas y registro de historias clínicas de pacientes para el Centro Médico “Jesús de Nazareth”. En primer lugar, se definieron los requerimientos del sistema, mediante el uso de historias de usuario, y a partir de estas se enlistó los requerimientos funcionales y no funcionales para cada una de las iteraciones. En esta fase se realizaron los diagramas de casos de uso correspondientes para cada módulo. También se indican el diagrama de clases y el diagrama relacional correspondiente a la base de datos. Además, se realizaron los diagramas de actividades del sistema. Para finalizar la etapa de diseño se presentan las interfaces visuales de cada módulo del sistema.

2.1.1 HISTORIAS DE USUARIO E ITERACIONES

Las historias de usuario son utilizadas en las metodologías ágiles para la adquisición de requerimientos, en la metodología XP la historia de usuario es desarrollada juntamente con los usuarios, es por esto que para la elaboración de las mismas se mantuvieron varias reuniones con la directora y los trabajadores del Centro Médico. Debido a que las historias de usuario no tienen un formato definido, para el presente proyecto se presentan los siguientes elementos de las historias de usuario:

- **Autor:** Es el o los desarrolladores que realizan la historia de usuario.
- **Usuario**: Son las personas que indican los requerimientos del sistema para la historia de usuario.
- **Fecha:** Es la fecha en la cual se elaboró la historia de usuario.

- **Código:** Es un código único que identifica la historia de usuario. Está conformada por las siglas “HU”, el número de iteración y el número de historia dentro de la iteración. La descripción del código se indica en la Figura 2.1.

Figura 2.1. Descripción del código de las historias de usuario

- **Nombre de HU:** indica una pequeña descripción de los requerimientos de la historia de usuario.
- **Prioridad:** Indica el grado de prioridad de la historia de usuario dentro del desarrollo del sistema, para lo cual se utiliza la **Tabla 2.1**.

Tabla 2.1. Prioridades de las historias de usuario

Prioridad	Descripción
1	Alta
2	Normal
3	Baja

- **Estimación:** Se presenta la estimación de la cantidad de horas que se requieren para realizar la historia de usuario.
- **Descripción y Observaciones:** Son las descripciones, observaciones y requerimientos que solicitó el usuario.

A continuación, se describen las iteraciones con un resumen de sus respectivas historias de usuario. Todas las historias de usuario se encuentran en el Anexo A, como ejemplo en la Tabla 2.2 se muestra la primera historia de usuario.

2.1.1.1 Primera iteración

Para la primera iteración se entrevistó a la directora del Centro Médico “Jesús de Nazareth” y ella manifestó los primeros requerimientos, los cuales incluyen el módulo *Login* y el de Administración de usuarios, para el sistema de escritorio, con los cuales se elaboró las historias de usuario de la primera iteración. A continuación, en la Tabla 2.3 se encuentra el resumen de las historias de usuario de la primera iteración.

Tabla 2.2. Ejemplo de historia de usuario

Autor:	Jissela Arcos Galo Rubio	**Usuario:**	Directora del Centro Médico
Código de HU:	HU-01-01	**Fecha:**	16/03/2018
Nombre de HU:	Acceso al sistema		
Prioridad:	1	**Estimación:**	20
Descripción y Observaciones: En el módulo Login se debe ingresar el nombre y tipo de usuario. Los tipos de usuario que tendrá el sistema serán: administrador, recepcionista, médico y laboratorista.			

Tabla 2.3. Resumen de las historias de usuario de la primera iteración

Código	Título	Usuario	Prioridad
HU-01-01	Acceso al sistema	Directora del centro médico	1
HU-01-02	Recuperación de contraseña	Directora del centro médico	2
HU-01-03	Actualización de contraseña	Directora del centro médico	2
HU-01-04	Visualización de los usuarios del cliente escritorio	Directora del centro médico	2
HU-01-05	Creación de nuevos usuarios	Directora del centro médico	1
HU-01-06	Visualización de los datos de un usuario	Directora del centro médico	1
HU-01-07	Actualización de los datos un usuario	Directora del centro médico	2
HU-01-08	Eliminación de un usuario	Directora del centro médico	1

2.1.1.2 Segunda iteración

Para la segunda iteración se entrevistó a la recepcionista del Centro Médico “Jesús de Nazareth” para crear las historias de usuario del módulo de Recepción. A continuación, en la Tabla 2.4 se encuentra el resumen de las historias de usuario de la segunda iteración.

Tabla 2.4. Resumen de las historias de usuario de la segunda iteración

Código	Título	Usuario	Prioridad
HU-02-01	Creación y eliminación de nuevos pacientes y médicos por parte de la recepcionista	Recepcionista	1
HU-02-02	Modificación del perfil personal del recepcionista	Recepcionista	3
HU-02-03	Visualización de los médicos disponibles y asignación de citas médicas	Recepcionista	1
HU-02-04	Impresión y envío de las citas por correo electrónico	Recepcionista	2
HU-02-05	Confirmación de solicitudes de citas médicas	Recepcionista	2

2.1.1.3 Tercera iteración

Para elaborar las historias de usuario de la tercera iteración se realizó una reunión entre los desarrolladores y todos los especialistas del Centro Médico (médicos generales, médicos especialistas, psicólogos y odontólogos). El resumen de las historias de usuario de esta iteración se encuentra en la Tabla 2.5 y Tabla 2.6.

Tabla 2.5. Resumen de las historias de usuario de la tercera iteración (parte 1 de 2)

Código	Título	Usuario	Prioridad
HU-03-01	Visualización de todos los pacientes asignados	Especialistas del centro médico	1
HU-03-02	Visualización de la agenda los especialistas	Especialistas del centro médico	1
HU-03-03	Creación de citas médicas	Especialistas del centro médico	1
HU-03-04	Toma de signos vitales	Especialistas del centro médico	2
HU-03-05	Ficha médica de medicina general	Médicos generales del centro médico	1
HU-03-06	Consulta para medicina general	Directora del centro médico	1

Tabla 2.6. Resumen de las historias de usuario de la tercera iteración (parte 2 de 2)

Código	Título	Usuario	Prioridad
HU-03-07	Ficha psicológica	Psicólogos del centro médico	1
HU-03-08	Consulta psicológica	Psicólogos del centro médico	1
HU-03-09	Ficha odontológica	Odontólogos del centro médico	1
HU-03-10	Consulta odontológica	Odontólogos del centro médico	2
HU-03-11	Odontograma	Odontólogos del centro médico	1
HU-03-12	Modificación perfil personal especialistas	Especialistas del centro médico	3

2.1.1.4 Cuarta iteración

Para la cuarta iteración se tuvo una reunión con todos los especialistas del Centro Médico, para definir los requerimientos del módulo que les permite solicitar exámenes médicos. Posteriormente, se realizó una reunión con el laboratorista para elaborar las historias de usuario del módulo del laboratorio clínico. Finalmente, se tuvo una reunión con todos los especialistas para definir los requerimientos de la historia clínica de cada paciente. En la Tabla 2.7 se encuentra el resumen de las historias de usuario.

Tabla 2.7. Resumen de las historias de usuario de la cuarta iteración

Código	Título	Usuario	Prioridad
HU-04-01	Solicitud de exámenes médicos	Especialistas del centro médico	1
HU-04-02	Lista de pacientes con solicitud de exámenes	Laboratorista	1
HU-04-03	Exámenes médicos	Laboratorista	2
HU-04-04	Resultado de exámenes	Laboratorista	2
HU-04-05	Historias clínicas	Laboratorista	1

2.1.1.5 Quinta iteración

Para la quinta iteración las historias de usuario fueron elaboradas con ayuda de los especialistas y la directora del Centro Médico; en esta iteración se recogieron los

requerimientos para el desarrollo de la aplicación móvil. En la Tabla 2.8 se encuentra el resumen de las historias de usuario de esta iteración.

Tabla 2.8. Resumen de las historias de usuario de la quinta iteración

Código	Título	Usuario	Prioridad
HU-05-01	Módulo Login de la aplicación móvil	Especialistas y directora del Centro Médico	1
HU-05-02	Recuperación de contraseña en la aplicación móvil	Especialistas y directora del Centro Médico	2
HU-05-03	Menú de la aplicación móvil	Especialistas y directora del Centro Médico	1
HU-05-04	Modificación del perfil de los usuarios en la aplicación móvil	Especialistas y directora del Centro Médico	1
HU-05-05	Actualización de contraseña en la aplicación móvil	Especialistas y directora del Centro Médico	3
HU-05-06	Agendar citas desde la aplicación móvil	Especialistas y directora del Centro Médico	1
HU-05-07	Visualización del historial médico en la aplicación móvil	Especialistas y directora del Centro Médico	2
HU-05-08	Agenda de especialistas en la aplicación móvil	Especialistas y directora del Centro Médico	3

2.1.2 REQUERIMIENTOS DEL SISTEMA

A continuación, se establecen los requerimientos funcionales y no funcionales del sistema, de acuerdo con las necesidades expresadas en las historias de usuario.

2.1.2.1 Requerimientos Funcionales

Los requerimientos funcionales fueron seleccionados en base a las historias de usuario elaboradas para cada iteración. El código de los requerimientos funcionales consta de las siglas RF, a continuación, el número de iteración y finalmente el número de requerimiento. Esta nomenclatura permite identificar cada requerimiento en las pruebas de

funcionamiento. Los requerimientos funcionales para la primera iteración se muestran en la Tabla 2.9.

Tabla 2.9. Requerimientos funcionales de la primera iteración

Código	Descripción
RF-01-01	Para ingresar al sistema de escritorio el usuario deberá ingresar su nombre de usuario, el tipo de usuario y su contraseña.
RF-01-02	Los tipos de usuario del sistema de escritorio serán: administrador, recepcionista, médico y laboratorista.
RF-01-03	Para crear un nuevo usuario en el sistema se deberá enviar un correo electrónico con un enlace para crear la contraseña.
RF-01-04	El sistema de escritorio debe permitir recuperar la contraseña en caso de olvido.
RF-01-05	El sistema de escritorio permitirá modificar la contraseña actual.
RF-01-06	En el módulo del administrador se podrá visualizar y buscar a todos los usuarios del sistema.
RF-01-07	En el módulo del administrador se podrá crear nuevos usuarios y asignar el tipo de usuario.
RF-01-08	En el módulo del administrador se podrá visualizar y modificar los datos de los usuarios.
RF-01-09	En el módulo del administrador se podrá asignar o eliminar la característica de paciente de los especialistas.
RF-01-10	En el módulo del administrador podrá eliminar a usuarios en el sistema.

Los requerimientos funcionales para la segunda iteración se encuentran en las Tabla 2.10 y Tabla 2.11 .

Tabla 2.10. Requerimientos funcionales de la segunda iteración (parte 1 de 2)

Código	Descripción
RF-02-01	En el módulo de recepción se podrá crear o eliminar nuevos pacientes y médicos.
RF-02-02	En el módulo de recepción se podrá modificar el perfil personal de la recepcionista
RF-02-03	En el módulo de recepción se visualizará la lista de todos los especialistas

Tabla 2.11. Requerimientos funcionales de la segunda iteración (parte 2 de 2)

Código	Descripción
RF-02-04	En el módulo de recepción se podrá agendar citas médicas en los horarios disponibles.
RF-02-05	En el módulo de recepción se podrá imprimir y enviar por correo electrónico a los pacientes la información de las citas agendadas.
RF-02-06	El módulo de recepción permitirá confirmar o denegar las solicitudes de citas médicas enviadas por los pacientes desde la aplicación móvil. Se enviará un correo electrónico con la respuesta.

Los requerimientos funcionales para la tercera iteración se detallan en las Tabla 2.12 y Tabla 2.13.

Tabla 2.12. Requerimientos funcionales de la tercera iteración (parte 1 de 2)

Código	Descripción
RF-03-01	El sistema de escritorio permitirá a los especialistas visualizar todos los pacientes que hayan tenido o tendrán una cita con los especialistas.
RF-03-02	El sistema de escritorio permitirá a los especialistas, a excepción del laboratorista clínico, visualizar un calendario y un horario de las citas médicas que se le han asignado.
RF-03-03	El sistema de escritorio permitirá a los especialistas, a excepción del laboratorio clínico, crear citas médicas para sus pacientes, también pueden agendar citas de interconsultas
RF-03-04	El sistema de escritorio permitirá a ingresar los datos de la revisión de signos vitales como presión, temperatura, edad, peso, estatura y observaciones generales para cada consulta.
RF-03-05	El sistema de escritorio permitirá a los médicos generales y especialistas, a excepción del odontólogo, psicólogo y laboratorista, en la primera consulta ingresar: el motivo de la primera consulta, revisión actual de órganos, antecedentes personales, antecedentes familiares. Todos los datos anteriores serán modificables.
RF-03-06	El sistema de escritorio permitirá a los médicos generales y especialistas visualizar todas las consultas anteriores con el especialista.

Tabla 2.13. Requerimientos funcionales de la tercera iteración (parte 2 de 2)

Código	Descripción
RF-03-07	El sistema de escritorio permitirá a los médicos generales y otros especialistas, a excepción del psicólogo y laboratorista, crear una consulta en cada cita, en la cual consten los signos vitales, el diagnóstico y tratamiento asignados.
RF-03-08	El sistema de escritorio permitirá a los psicólogos en las primeras consultas ingresar la siguiente información del paciente: riesgos psicosociales, anamnesis personal, anamnesis familiar, anamnesis social. Todos los datos anteriores se pueden modificar posteriormente.
RF-03-09	El sistema de escritorio permitirá a los psicólogos crear nuevas consultas e ingresar el diagnóstico, tratamiento y evaluación del paciente.
RF-03-10	El sistema de escritorio permitirá a los odontólogos en la primera consulta ingresar los siguientes antecedentes: propenso a la hemorragia, alergia a algún medicamento, fiebre reumática, diabetes, asma y hepatitis.
RF-03-11	El sistema de escritorio permitirá a los odontólogos ingresar los resultados de los rayos X realizados al paciente con fecha y observación.
RF-03-12	El sistema de escritorio permitirá al odontólogo visualizar el odontograma con los respectivos símbolos que indiquen el estado de las piezas dentales de los pacientes.
RF-03-13	El sistema de escritorio permitirá a los odontólogos crear una consulta odontológica que les permita ingresar el diagnóstico y tratamiento. Adicional debe tener campos para ingresar el presupuesto y abono del tratamiento.
RF-03-14	El sistema de escritorio tendrá un odontograma que debe mostrar las piezas dentales con sus respectivas partes, debe funcionar tanto para niños como para adultos.
RF-03-15	El sistema de escritorio tendrá un odontograma que debe permitir añadir las siguientes características a cada pieza: extracción, caries, sellante, endodoncia, corona, fractura.
RF-03-16	El sistema de escritorio permitirá a todos los especialistas modificar su perfil personal.

Los requerimientos funcionales para la cuarta iteración se encuentran en la Tabla 2.14.

Tabla 2.14. Requerimientos funcionales de la cuarta iteración

Código	Descripción
RF-04-01	El sistema de escritorio permitirá a todos los especialistas solicitar la realización de exámenes de laboratorio. Esta solicitud se podrá imprimir.
RF-04-02	El sistema de escritorio permitirá al laboratorista visualizar la lista de todos los pacientes que tienen una solicitud de exámenes en el centro médico.
RF-04-03	El sistema de escritorio permitirá al laboratorista visualizar de cada paciente los exámenes solicitados, médico solicitante, la fecha de emisión, fecha de realización y la fecha de entrega.
RF-04-04	El sistema de escritorio permitirá al laboratorista subir un documento con los resultados de los exámenes realizados.
RF-04-05	El sistema de escritorio permitirá a los médicos acceder a todas las fichas de un paciente (médica, odontológica, psicológica y exámenes del paciente), excepto a la ficha psicológica.
RF-04-06	El sistema de escritorio permitirá imprimir la historia clínica de los pacientes que está conformada por la ficha médica, la ficha odontológica, la ficha psicológica y los exámenes de cada paciente.

Los requerimientos funcionales para la quinta iteración se encuentran en las Tabla 2.15 y Tabla 2.16.

Tabla 2.15. Requerimientos funcionales de la quinta iteración (parte 1 de 2)

Código	Descripción
RF-05-01	La aplicación permitirá a los usuarios iniciar sesión, para esto deben ingresar su usuario, contraseña y si es paciente o doctor
RF-05-02	La aplicación móvil permitirá a los usuarios recuperar la contraseña.
RF-05-03	La aplicación móvil tendrá un menú con las siguientes opciones: mi perfil, citas e historial médico (sólo para los pacientes).
RF-05-04	La aplicación móvil en la opción de mi Perfil permitirá a los usuarios visualizar o modificar datos personales: nombres, apellidos, lugar de nacimiento, dirección, ciudad, teléfono y celular.

Tabla 2.16. Requerimientos funcionales de la quinta iteración (parte 2 de 2)

Código	Descripción
RF-05-05	La aplicación móvil en la opción de mi Perfil permitirá a los usuarios modificar su contraseña.
RF-05-06	La aplicación móvil permitirá a los pacientes visualizar las citas agendadas.
RF-05-07	La aplicación móvil permitirá a los pacientes solicitar una cita para lo cual debe ingresar los siguientes datos: especialidad, especialista, fecha y horario (sólo podrán seleccionar los horarios disponibles).
RF-05-08	La aplicación móvil permitirá a los pacientes y doctores exportar a Google Calendar las citas asignadas.
RF-05-09	La aplicación móvil permitirá a los pacientes visualizar su diagnóstico y tratamiento de las dos últimas consultas realizadas de cada especialidad.
RF-05-10	La aplicación móvil permitirá a los especialistas visualizar su agenda con las citas médicas asignadas.

2.1.2.2 Requerimientos no funcionales

Los requerimientos no funcionales no describen funciones a realizar, si no son características del prototipo que establecen restricciones en el diseño e implementación [32]. En las **¡Error! No se encuentra el origen de la referencia.** se describen los requerimientos no funcionales de todo el sistema.

Tabla 2.17. Requerimientos no funcionales del sistema

Código	Descripción
RNF-01	Los usuarios del sistema contarán con interfaces amigables e intuitivas.
RNF-02	El tiempo de respuesta de los diferentes módulos deberá ser rápido. Esto también dependerá de la velocidad del Internet.
RNF-03	El sistema proporcionará mensajes de error e informativos orientados al usuario final.
RNF-04	La contraseña de los usuarios se cifrará antes de ser ingresada en la base de datos

2.1.3 ARQUITECTURA DEL PROTOTIPO

La arquitectura del sistema es de tipo cliente-servidor, como se indica en la Figura 2.2 y consta de dos elementos. El primer elemento es el cliente que está constituido por el sistema desarrollado para el sistema Windows y la aplicación Android. Estos consumirán los servicios web WCF del segundo componente, que consta del servidor el cual se lo implementó en la plataforma de la nube Microsoft Azure. En esta arquitectura los dos componentes interactúan enviando y recibiendo mensajes, el cliente envía peticiones al servidor y éste envía las respuestas. Estos mensajes son trasportados mediante servicios web en los métodos del protocolo HTTP y utilizando el formato JSON porque este formato es independiente de cualquier lenguaje de programación.

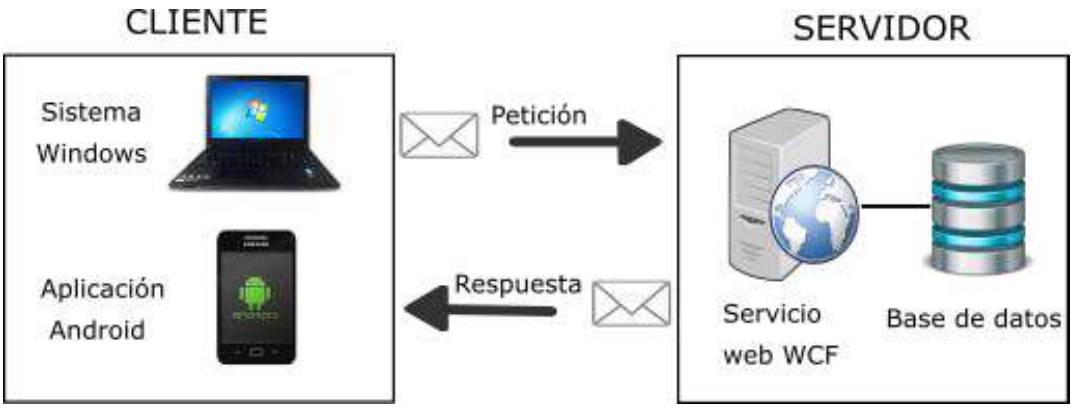


Figura 2.2. Arquitectura del prototipo

2.1.4 DIAGRAMAS DE CASO DE USO

Los diagramas de casos de uso permiten la toma de requisitos del sistema y expresan gráficamente las relaciones entre los diferentes usos de este y sus participantes o actores. Los diagramas de casos de uso se realizaron para cada módulo del cliente de escritorio y del cliente móvil.

2.1.4.1 Diagramas de casos de uso del Cliente de Escritorio

En la Figura 2.3 se encuentra el diagrama de casos de uso del módulo 1. Los especialistas, el administrador o la recepcionista para acceder al sistema deben ingresar su nombre de usuario, seleccionar el tipo de usuario e ingresar la contraseña. También pueden modificar su contraseña y restablecerla en caso de pérdida.

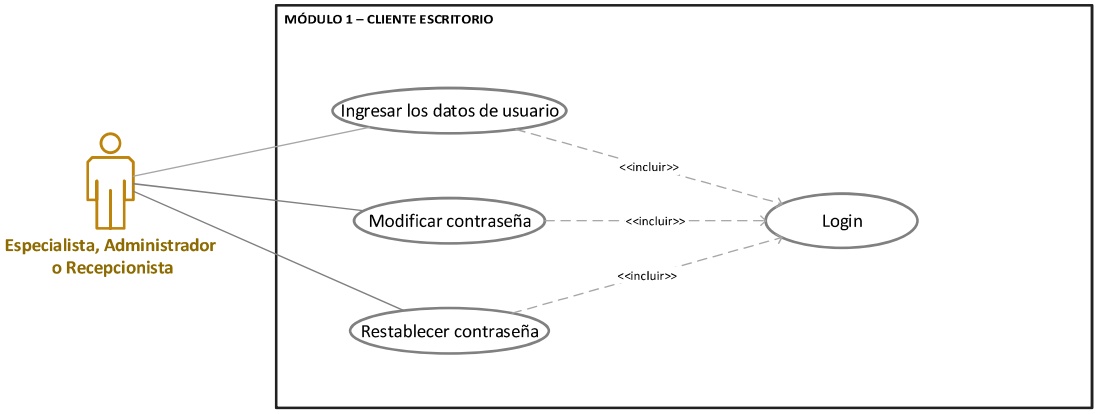


Figura 2.3. Diagrama de casos de uso del módulo 1 del cliente de escritorio

En la Figura 2.4 se encuentra el diagrama de casos de uso del módulo 2. El administrador puede visualizar todos los usuarios del sistema, para esto también se incluyó un buscador. Además, puede crear nuevos usuarios y asignar si son trabajadores, con sus respectivas especialidades. Adicional pueden leer y actualizar la información de los usuarios y también tiene la capacidad de eliminar usuarios.

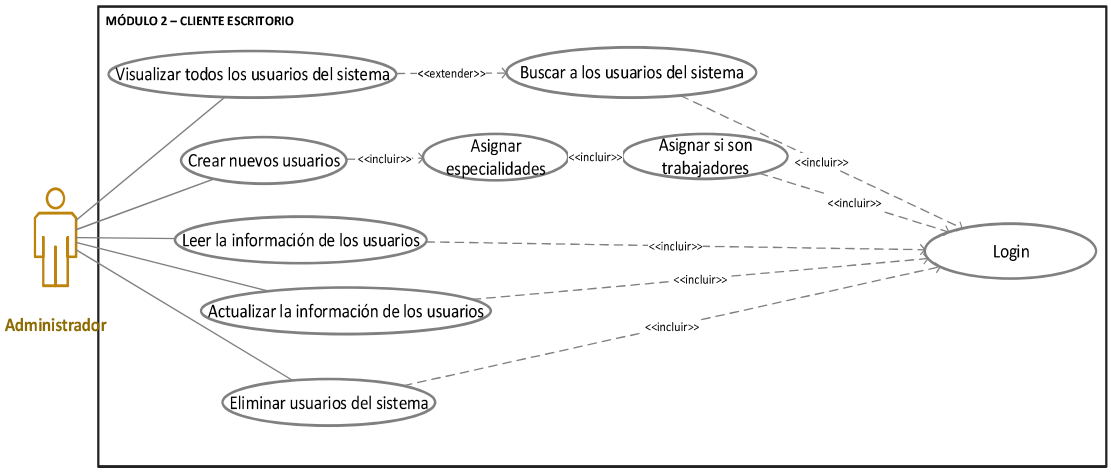


Figura 2.4. Diagrama de casos de uso del módulo 2 del cliente de escritorio

En la Figura 2.5 se encuentra el diagrama de casos de uso del módulo 3. La recepcionista puede visualizar un calendario que le permite crear citas médicas, las cuales podrá enviar por correo electrónico e imprimir el archivo PDF. Adicional, puede visualizar la agenda de cada especialista. Además, la recepcionista puede modificar y eliminar citas médicas. Por otra parte, la recepcionista puede crear o eliminar pacientes y médicos, también puede leer o modificar su información.

En la Figura 2.6 se muestra el diagrama de casos de uso del módulo 4. El especialista puede visualizar su agenda; puede visualizar, crear y modificar fichas de los pacientes y a

su vez puede realizar las mismas acciones con las consultas. También puede ver los resultados de los exámenes médicos de los pacientes.

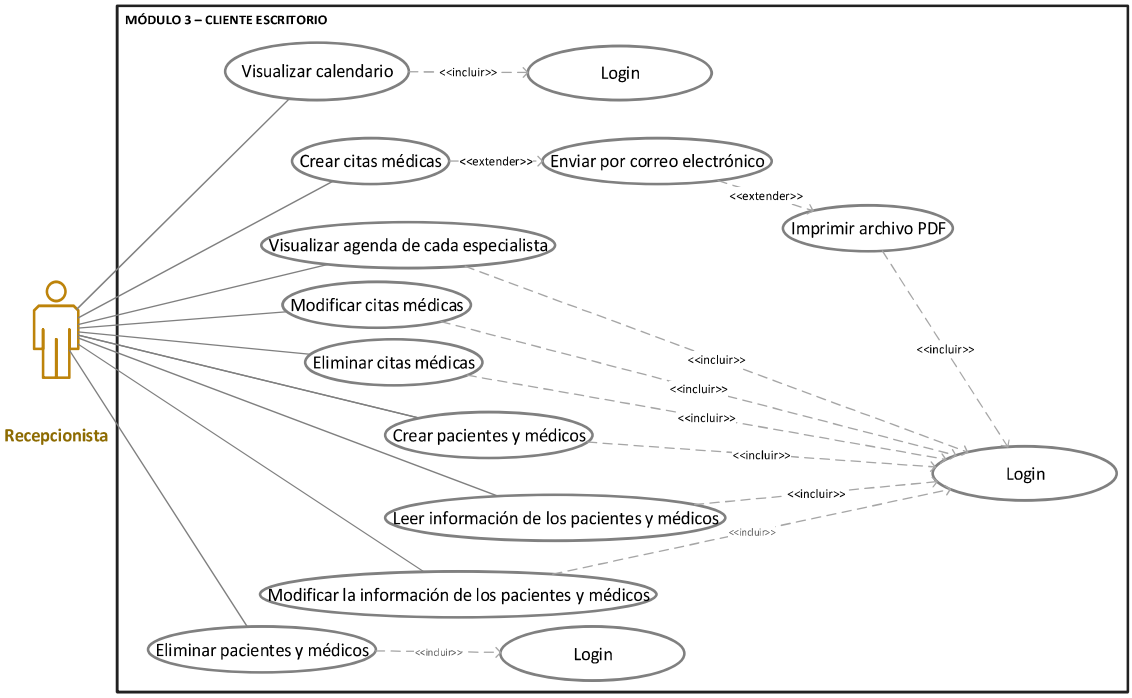


Figura 2.5. Diagrama de casos de uso del módulo 3 del cliente de escritorio

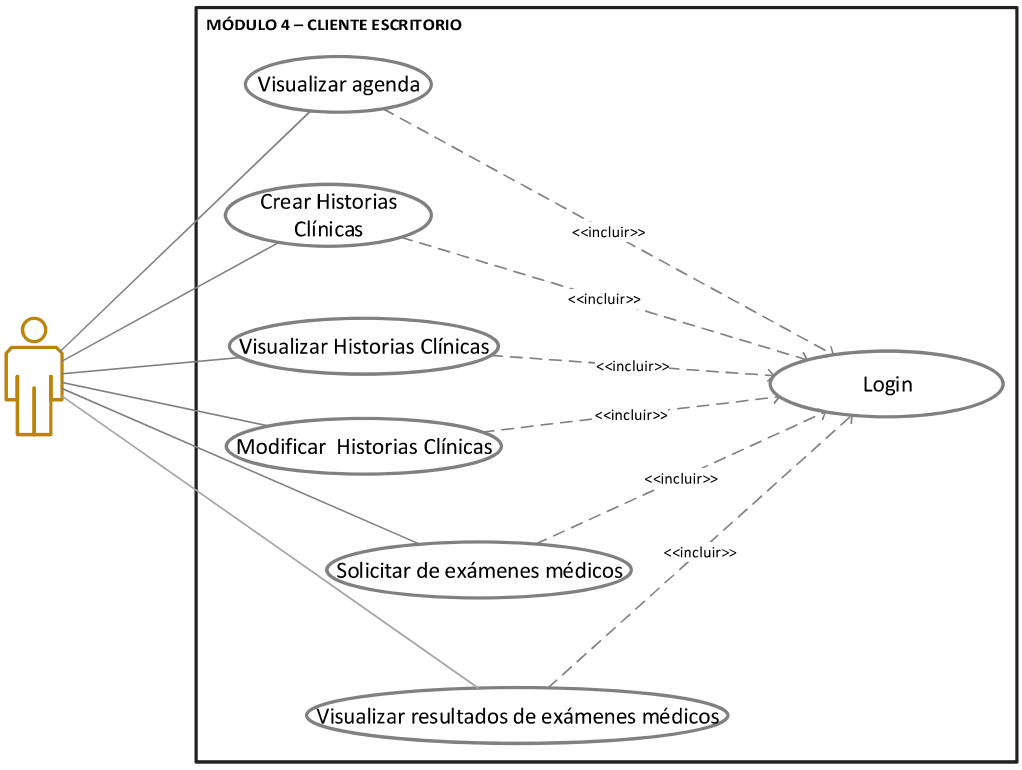


Figura 2.6. Diagrama de casos de uso del módulo 4 del cliente de escritorio

En la Figura 2.7 se encuentra el diagrama de casos de uso del módulo 5. El laboratorista puede visualizar la lista de pacientes que tienen una petición de examen, visualizar la lista de resultados y los exámenes que se puedan realizar. Adicional el laboratorista puede ingresar el resultado de los exámenes e imprimirlos.

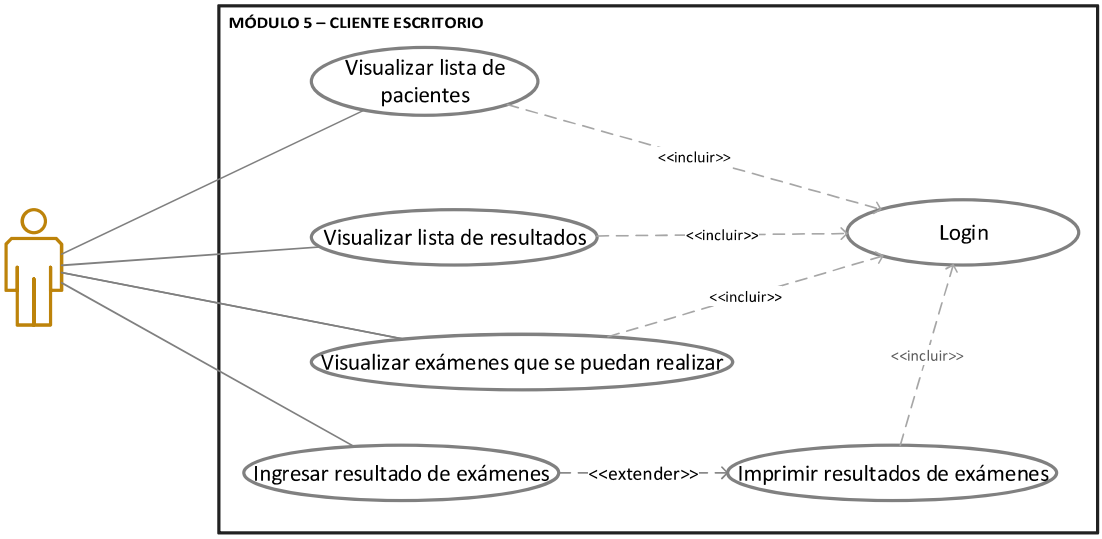


Figura 2.7. Diagrama de casos de uso del módulo 5 del cliente de escritorio

2.1.4.2 Diagramas de casos de uso del Cliente de Móvil

En la Figura 2.8 se encuentra el diagrama de casos de uso del módulo 1 del cliente móvil. Los especialistas y/o pacientes pueden ingresar a la aplicación móvil después de ingresar su nombre de usuario, tipo de usuario y contraseña. Los usuarios también pueden restablecer su contraseña.

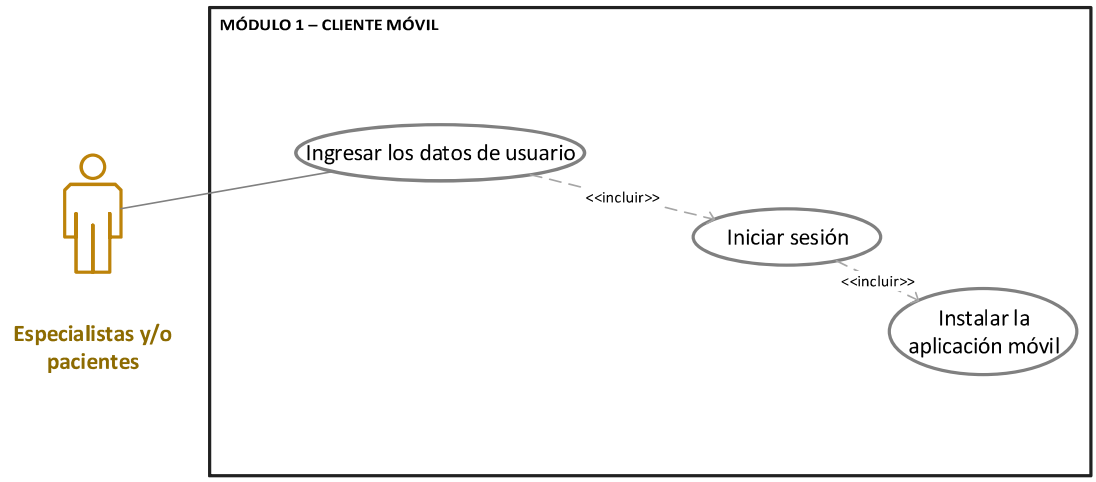


Figura 2.8. Diagrama de casos de uso del módulo 1 del cliente móvil

En la Figura 2.9 se encuentra el diagrama de casos de uso del módulo 2 del cliente móvil. Los especialistas y/o pacientes pueden leer o actualizar los datos personales. Por otro lado también pueden modificar su contraseña.

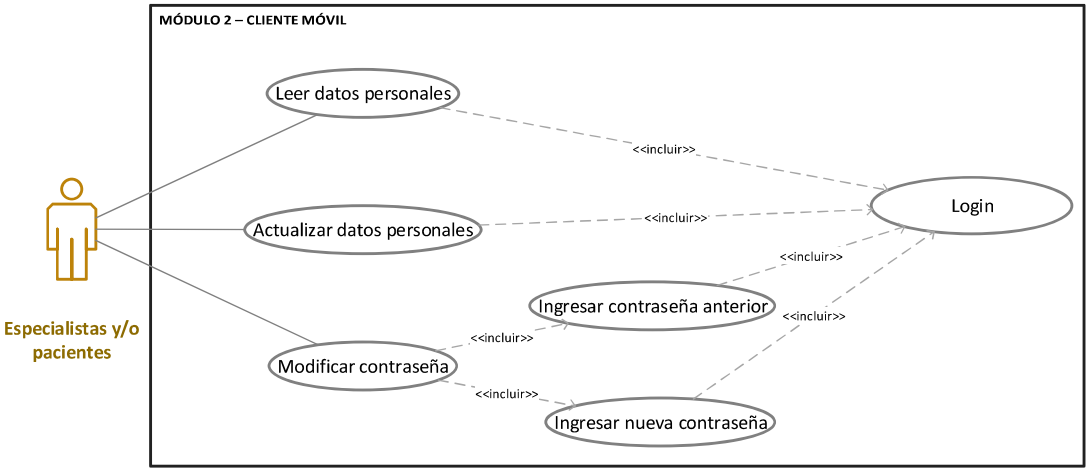


Figura 2.9. Diagrama de casos de uso del módulo 2 del cliente móvil

En la Figura 2.10 muestra el diagrama de casos de uso de los módulos 3 y 4 del cliente móvil. Los pacientes pueden solicitar citas, para lo cual deben seleccionar la especialidad, especialista, fecha y un horario disponible. Los especialistas y pacientes pueden visualizar su agenda de citas y exportarlas a Google Calendar.

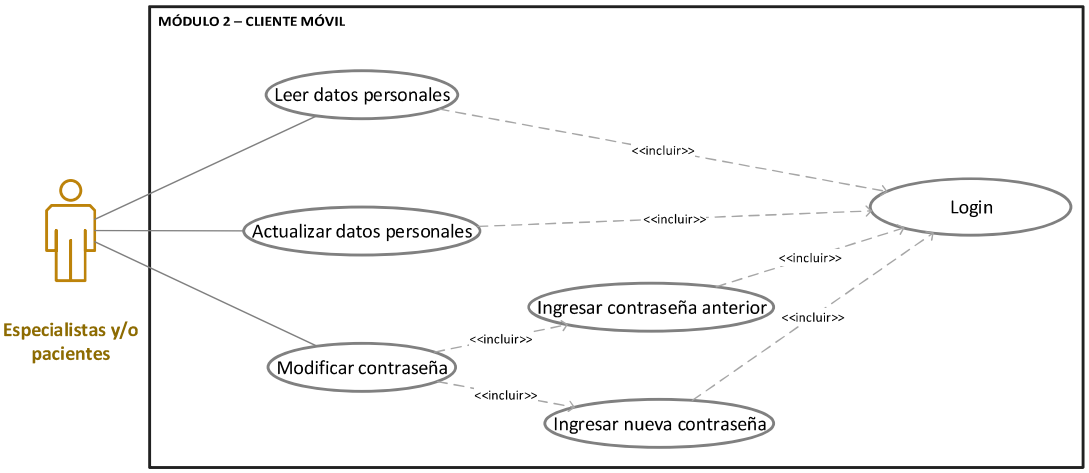


Figura 2.10. Diagrama de casos de uso de los módulos 3 y 4 del cliente móvil

En la Figura 2.11 se encuentra el diagrama de casos de uso del módulo 5 del cliente móvil. Los pacientes al seleccionar la especialidad pueden leer la información de las dos últimas consultas.

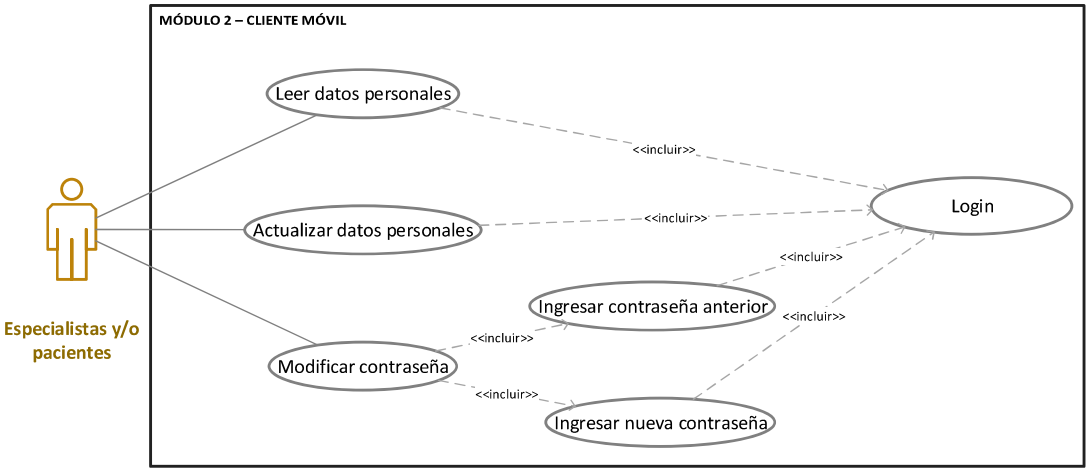


Figura 2.11. Diagrama de casos del módulo 5 del cliente móvil

2.1.5 DIAGRAMA DE BASES DE DATOS

El diagrama de base de datos se aprecia en la Figura 2.12. Este diagrama se ha elaborado utilizando *StarUML*, este programa posee herramientas que permiten modelar diferentes diagramas para el diseño de *software*. En la Figura 2.12 se observan las tablas que almacenan la información que genera el sistema y las relaciones que mantienen entre ellas. A continuación, se realiza una breve descripción de las tablas presentes en el diagrama de bases de datos.

`Usuario`: Almacena todos los detalles del usuario, como el tipo, la contraseña y diferentes estados que se utilizan para recuperación o modificación de la contraseña.

`Personas`: Corresponde a la tabla que almacena todos los datos personales de una persona, la cual puede ser paciente y/o especialista, es decir una persona puede tener varios usuarios, por lo tanto, la tabla `Personas` mantiene una relación de uno a muchos con la tabla `Usuarios`.

`Paciente`: Contiene el *id* del paciente y de la persona.

`Especialista`: Contiene el *id* del especialista y de la persona. Un especialista puede tener varias especialidades, esto se ve reflejado en la relación entre la tabla `Especialista` y `Especialidad`.

`Especialidad`: Esta tabla se utiliza para registrar el *id* de la especialidad, la especialidad y el tipo de usuario.

Cita: Almacena la información correspondiente a una cita médica como es el: horario, el especialista, la especialidad y el paciente

Horario: El horario de una cita debe tener una hora de inicio y una hora fin.

Solicitud_cita: Esta tabla almacena la información necesaria para solicitar una cita.

Ficha_medica: Esta tabla permite guardar la información correspondiente a una ficha para el uso de un médico general, almacena los antecedentes personales y familiares y la revisión actual de órganos.

Consulta_medica: Una ficha médica contiene varias consultas médicas, contienen la información solicitada por los médicos para registrar los datos generados por el paciente en cada consulta.

Ficha_ondontologica: Esta tabla almacena los datos que se necesitan guardar en una ficha odontológica.

Resultado_Rx: Esta tabla contiene los resultados de rayos X que se visualizan en la ficha odontológica. Una Ficha_odontologica puede tener varios Resultados_Rx.

Consulta_odontológica: Esta Tabla almacena información como el diagnóstico, tratamiento, presupuesto y abono que tiene un determinado paciente. La relación entre la tabla Ficha_odontologica y Consulta_odontológica es una a varios.

Odontograma: En la ficha odontológica se debe visualizar el ondontograma en el cual se encuentran los diferentes problemas odontológicos que puede presentar un paciente: extracción, caries, sellantes, endodoncia, corona, fractura y puente.

Ficha_ps_clinica: Los especialistas de psicología solicitaron su propia ficha para registro de la información de sus pacientes, la cual consta de los riesgos psicosociales, las anamnesis personal, familiar y social.

Consulta_psicologica: Esta tabla se utiliza para registrar la información de los pacientes en cada consulta psicológica.

Pedido_examen: Almacena las fechas necesarias para emitir un pedido de un examen de laboratorio.

Resultado: Esta tabla almacena los resultados de los pedidos de examen realizados. Un Pedido_examen puede tener varios resultados.

`Examen`: Detalla la descripción del examen de laboratorio. Tiene una relación unívoca con la tabla `Tipo_examen` y una relación de varios a varios con la tabla `Pedido_examen`.

`Tipo_examen`: Cada examen tiene un tipo de examen específico. En esta tabla se almacena el `id_tipo_examen` y el `tipo_examen`.

2.1.6 DIAGRAMAS DE CLASES

Los diagramas de clases permiten representar la estructura de un sistema en un lenguaje orientado a objetos de manera gráfica. Las clases presentadas en el diagrama serán las que usará el sistema para todos los servicios que permitirán el intercambio de datos. El diagrama de clases del servidor se observa en la Figura 2.13 (ver diagrama en tamaño A3 en el Anexo B).

2.1.7 DIAGRAMAS DE ACTIVIDADES

Los diagramas de actividades se utilizan para modelar el comportamiento del sistema. Un diagrama de actividades tiene dos dimensiones, el eje vertical representa el tiempo y el eje horizontal los diferentes componentes del sistema, el tiempo avanza desde la parte superior del diagrama hacia la inferior.

En la primera iteración, el primer diagrama de actividades realizado es el correspondiente a login, este sirve para el cliente de escritorio como para el móvil (ver Figura 2.14). En primer lugar, el usuario ingresa su nombre de usuario, tipo de usuario y contraseña; estos datos son enviados al servidor para validar la información en la base de datos, si la información es correcta se da acceso al sistema, caso contrario se envía un mensaje de error. En la Figura 2.15 se visualiza el diagrama de actividades para la interacción de la administración de usuarios. En primer lugar, la interfaz gráfica solicita al servidor traer de la base de datos la lista de personas, para que la interfaz la muestre al usuario. También permite al usuario seleccionar una persona. Además, seleccionar y solicitar una acción CRUD que afecte a la persona elegida. El diagrama de actividades para la interacción de administración de citas se visualiza en la Figura 2.16. La interfaz gráfica solicita al servidor que traiga las especialidades de la base de datos. El usuario selecciona la especialidad y la interfaz gráfica solicita traer todos los especialistas en dicha especialidad, después de seleccionar el especialista se debe cargar las citas disponibles para el especialista. Finalmente se debe seleccionar la acción CRUD que se desea realizar sobre la cita seleccionada.

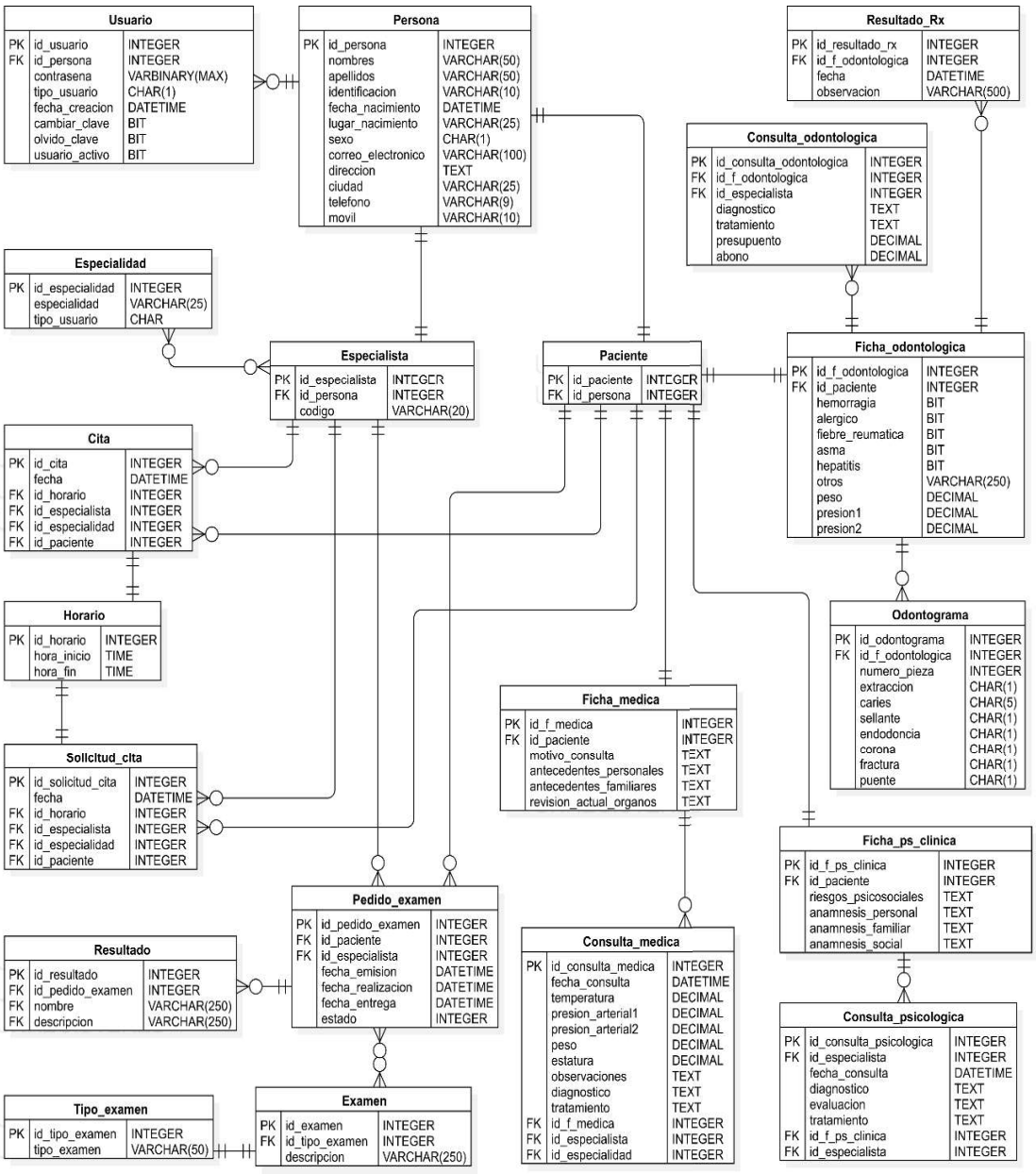


Figura 2.12. Diagrama de bases de datos

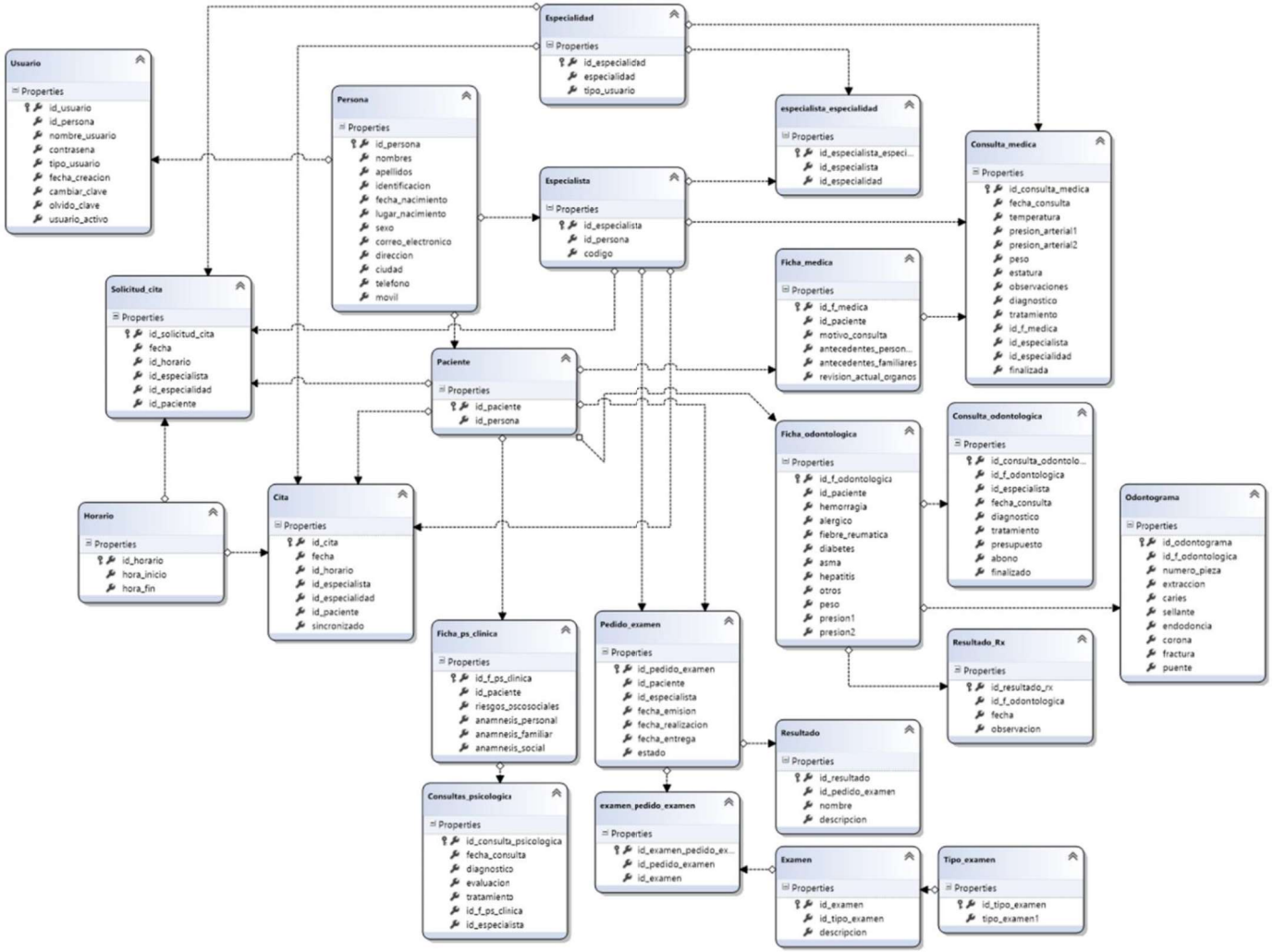


Figura 2.13. Diagrama de clases del servidor

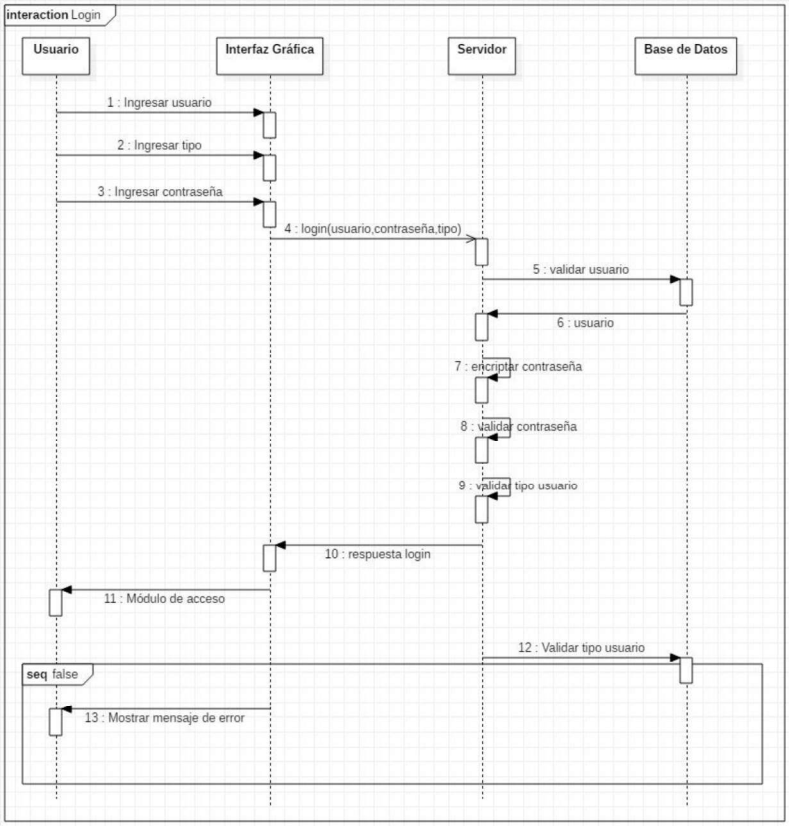


Figura 2.14. Diagrama de actividades de la interacción: Login

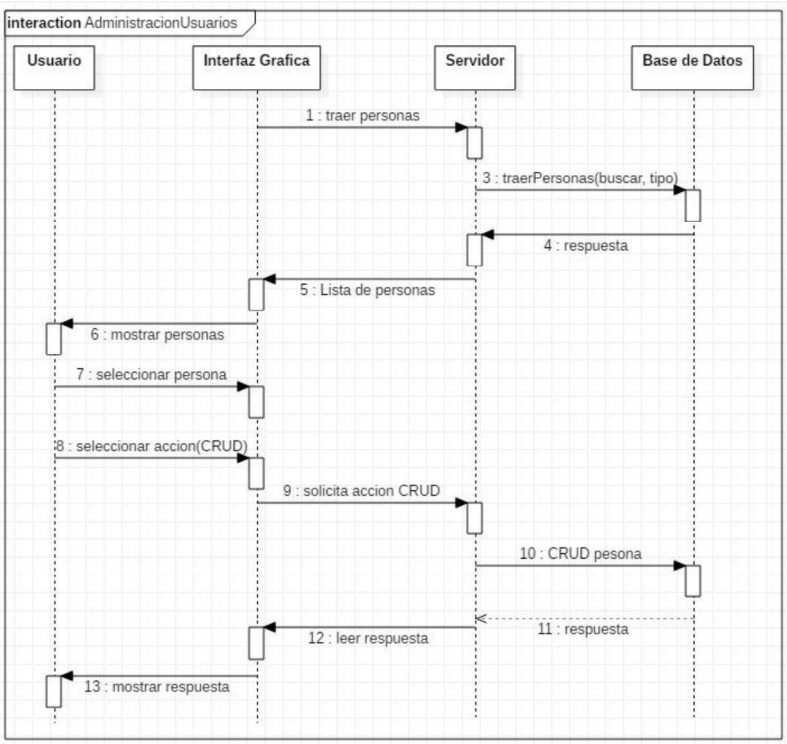


Figura 2.15. Diagrama de actividades de la interacción: Administración de Usuarios

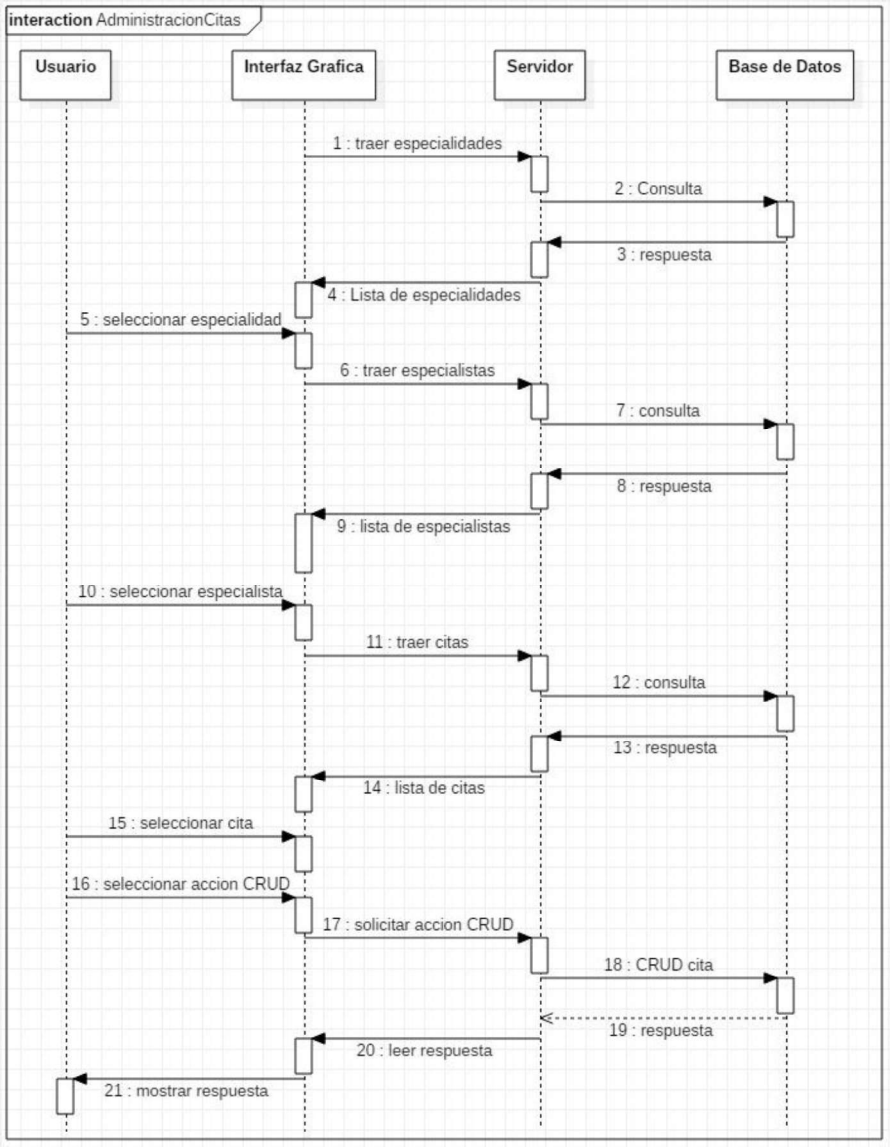


Figura 2.16. Diagrama de actividades de la interacción: Administración Citas

El diagrama de actividades para la interacción de administración de historias clínicas se visualiza en la Figura 2.17. En primer lugar, el médico selecciona al paciente, la interfaz gráfica solicita al servidor traer la historia clínica de éste de la base de datos y esta envía la información a la interfaz gráfica. Posteriormente se selecciona la ficha en la interfaz gráfica y el servidor consulta y trae ésta de la base de datos. La interfaz gráfica solicita traer las consultas correspondientes a cada ficha médica y envía la lista de éstas a la interfaz gráfica. El médico puede seleccionar una acción CRUD con las fichas y consultas del paciente, las cuales son enviadas a la base de datos y ésta envía una respuesta a través del servidor.

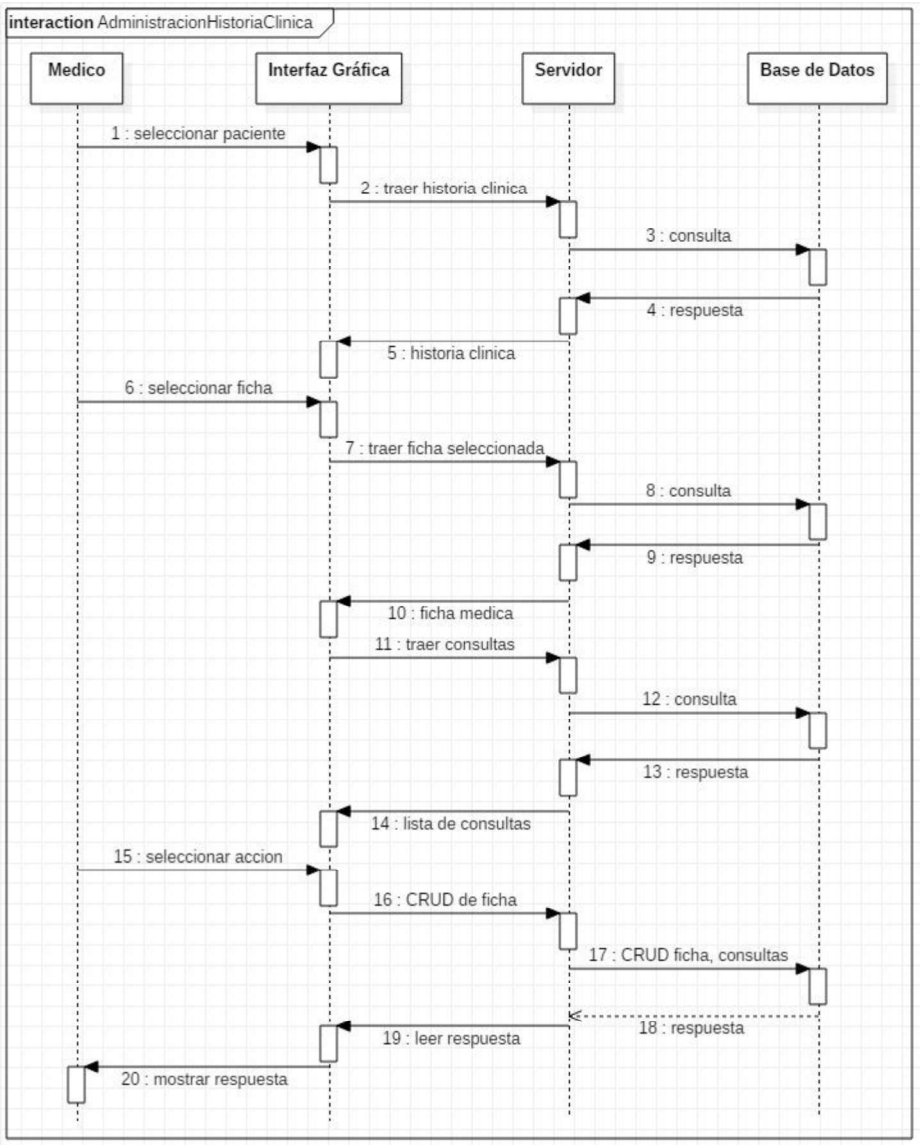


Figura 2.17. Diagrama de actividades de la interacción: Administración Historia Clínica

Otra interacción importante en el sistema es la recuperación de contraseña, el diagrama de actividades de esta interacción se encuentra en la Figura 2.18. En primer lugar, el usuario debe solicitar la recuperación de contraseña en la interfaz gráfica, ésta envía el *id_usuario* del solicitante al servidor en el método *olvidoPassword* el *id_usuario* del solicitante. El servidor envía un correo de recuperación de contraseña con un *link* para acceder a la interfaz web, en la cual se debe ingresar el usuario y código temporal, se valida que los datos ingresados sean correctos para solicitar al usuario su nueva contraseña y la confirmación de la misma, y si las dos contraseñas coinciden se encripta la nueva contraseña, para posteriormente ser almacenada en la base de datos.

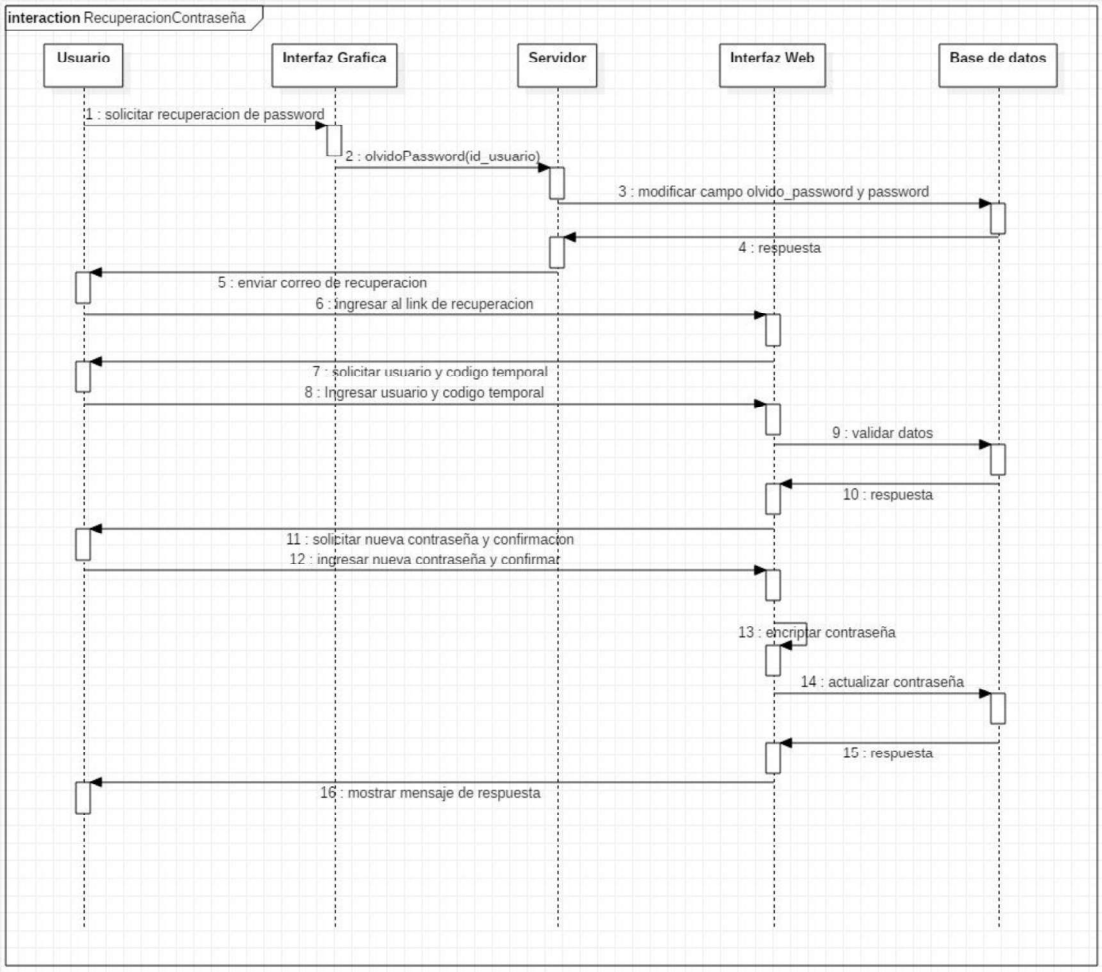


Figura 2.18. Diagrama de actividades de la interacción: Recuperación de Contraseña

2.1.8 DISEÑO DE LAS INTERFACES GRÁFICAS

Los *mockups*[1] del prototipo se diseñaron en base a los requerimientos funcionales del sistema y permitieron elaborar las interfaces de usuario de la aplicación en los sistemas operativos Windows y Android. Todos los *mockups* con su respectiva descripción se encuentran en el Anexo C y a manera de ejemplo se muestran los diseños de las interfaces gráficas realizadas para la primera iteración.

En la Figura 2.19 se encuentra el *mockup* para la página inicial del sistema de escritorio, corresponde al formulario de *Login*, que permite al usuario ingresar su nombre de usuario, tipo de usuario y contraseña para acceder al sistema, adicional tiene un link que permite recuperar la contraseña; en la Tabla 2.18 se encuentra la descripción de los controles que se va a utilizar en esta interfaz. Para la primera iteración también se diseñó la interfaz

1

gráfica principal del administrador que se visualiza en la Figura 2.20 , realizar acciones CRUD y buscar personas, en la Tabla 2.19 se encuentra su descripción.

Al seleccionar cualquiera de las acciones CRUD en la página principal del administrador se abre una interfaz, cuya información depende si la persona cuyos datos se están manejado es paciente o especialista. En la Figura 2.21 se encuentra la interfaz gráfica en caso de que la persona en la que se están realizando las acciones CRUD sea paciente.

En caso de que las acciones CRUD se realicen sobre un especialista del Centro Médico al marcar la opción `Paciente`, al especialista se le agrega el perfil de paciente. Adicional, se genera un código automático que le permite ingresar al sistema con un perfil de especialista dependiendo de las especialidades que tenga. El diseño de esta interfaz gráfica se encuentra en la Figura 2.22 y su respectiva descripción en la Tabla 2.21.

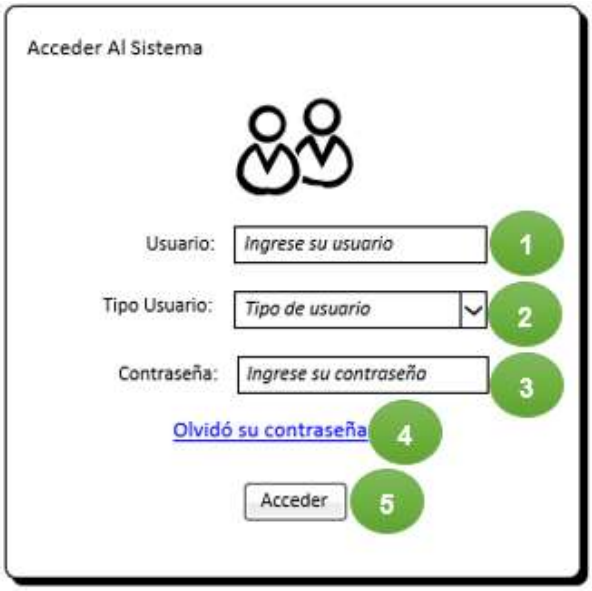


Figura 2.19. *Mockup* interfaz *Login*

Tabla 2.18. Descripción del *mockup* de *Login*

Identificador	Descripción	Control
1	Permite ingresar el nombre de usuario	*TextBox*
2	Permite seleccionar el tipo de usuario	*ComboBox*
3	Permite ingresar la contraseña	*TextBox*
4	Permite dirigirse a la interfaz web para recuperación de contraseña	*Hyperlink*
5	Permite acceder al sistema	*Button*

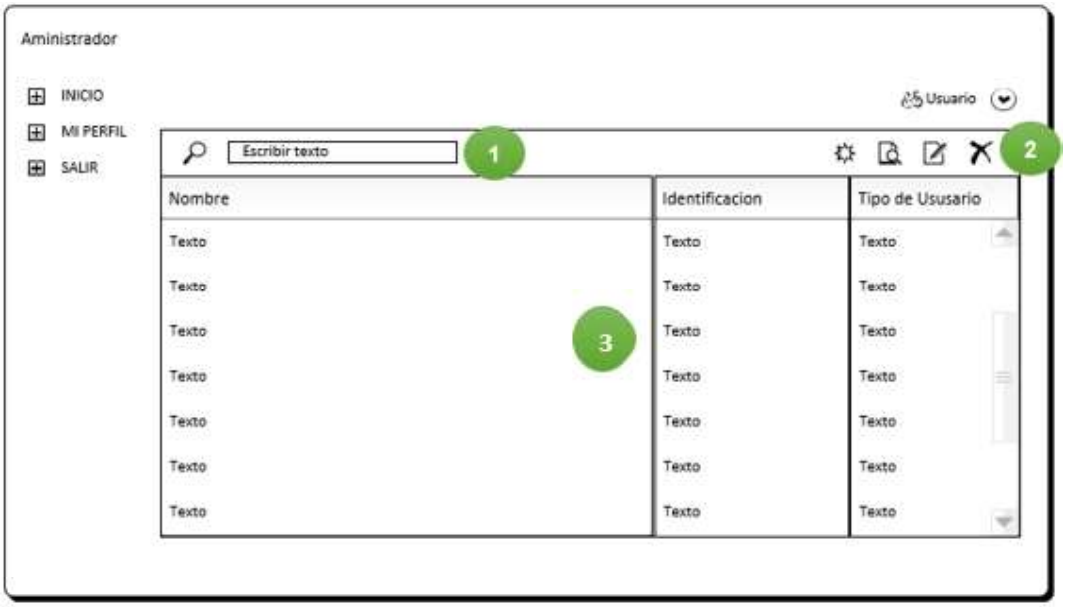


Figura 2.20. *Mockup* interfaz gráfica del administrador

Tabla 2.19. Descripción del *mockup* de *Administrador*

Identificador	Descripción	Control
1	Permite buscar una persona	*TextBox*
2	Permiten realizar acciones CRUD	*Button*
3	Permite visualizar las personas	ListView

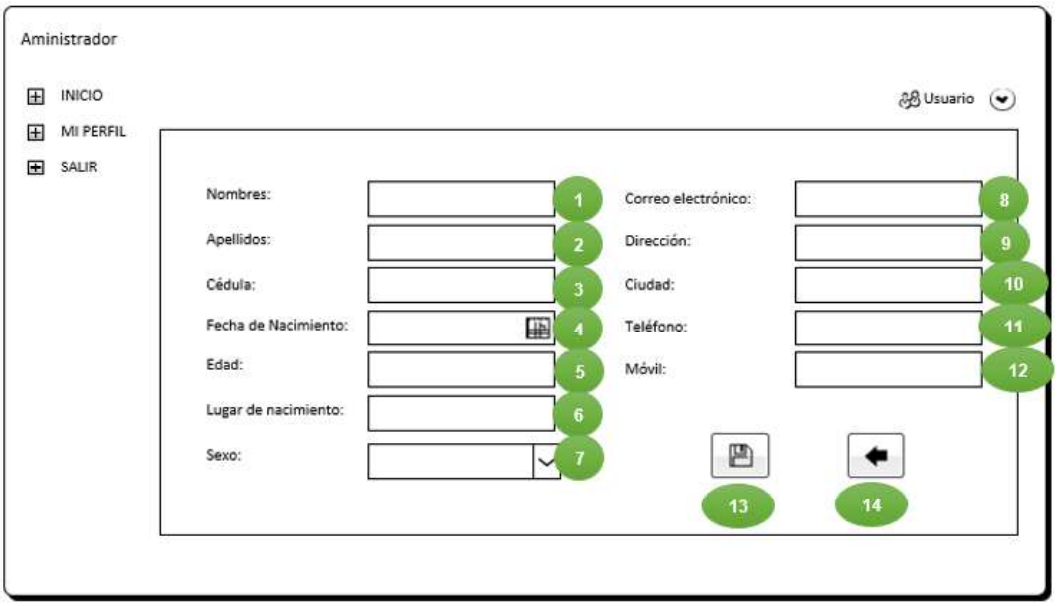


Figura 2.21. Mockup de los datos de un paciente

Tabla 2.20. Descripción del mockup de los datos de un paciente

Identificador	Descripción	Control
1	Permite ingresar los nombres del paciente	*TextBox*
2	Permite ingresar los apellidos del paciente	*TextBox*
3	Permite ingresar la cédula del paciente	*TextBox*
4	Permite ingresar la fecha de nacimiento del paciente	*DatePicker*
5	Permite ingresar la edad del paciente	*TextBox*
6	Permite ingresar el lugar de nacimiento del paciente	*TextBox*
7	Permite seleccionar el sexo del paciente	*ComboBox*
8	Permite ingresar el correo electrónico del paciente	*TextBox*
9	Permite ingresar la dirección del paciente	*TextBox*
10	Permite ingresar la ciudad de residencia del paciente	*TextBox*
11	Permite ingresar el número de teléfono del paciente	*TextBox*
12	Permite ingresar el número celular del paciente	*TextBox*
13	Permite guardar la información registrada	*Button*
14	Permite no guardar los cambios realizados	*Button*

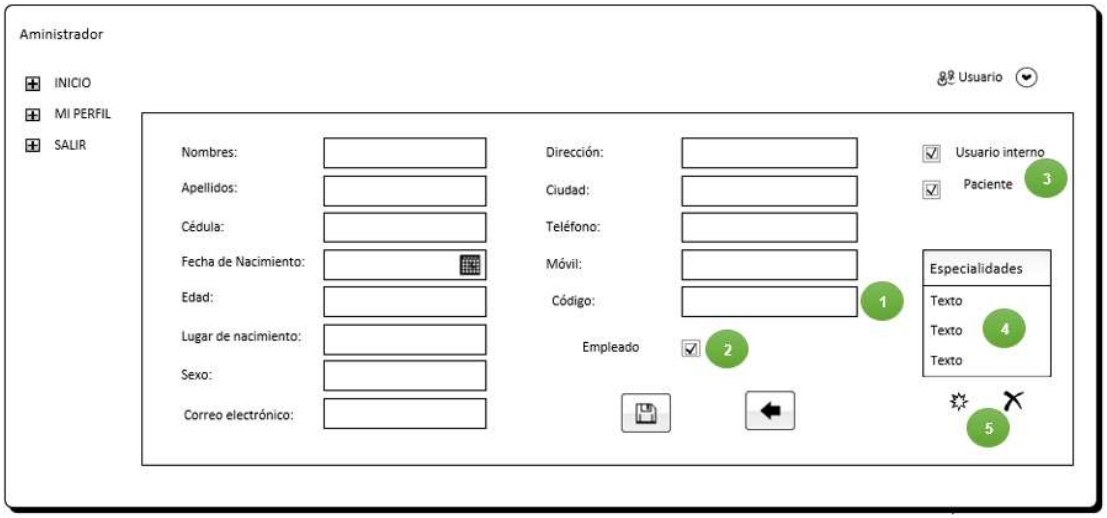


Figura 2.22. Mockup de los datos de un especialista

Tabla 2.21. Descripción del mockup de los datos de un especialista

Identificador	Descripción	Control
1	Permite ingresar el código del especialista	*TextBox*
2	Permiten seleccionar si es empleado	*CheckBox*
3	Permite seleccionar si será además de usuario interno también paciente	*CheckBox*
4	Permite añadir las especialidades del empleado	*ListBox*
5	Permite añadir o eliminar especialidades	*Button*

2.2 CONFIGURACIÓN DEL ENTORNO DE DESARROLLO

En este apartado se describen todas las herramientas que se han instalado previo al desarrollo del código del prototipo.

2.2.1 VISUAL STUDIO ENTERPRISE 2017

El lenguaje de programación utilizado para el cliente de escritorio es C#. Para el desarrollo de interfaces gráficas se utilizó la tecnología WPF juntamente con el lenguaje XAML. Para esto se utilizó el entorno de desarrollo integrado Visual Studio Enterprise 2017 que cuenta con soporte para las opciones descritas anteriormente. La descarga de este *software* se la realiza desde su página oficial. Al momento de instalar Visual Studio se instaló el componente individual *Herramientas de LINQ to SQL* como se observa en la Figura 2.23.

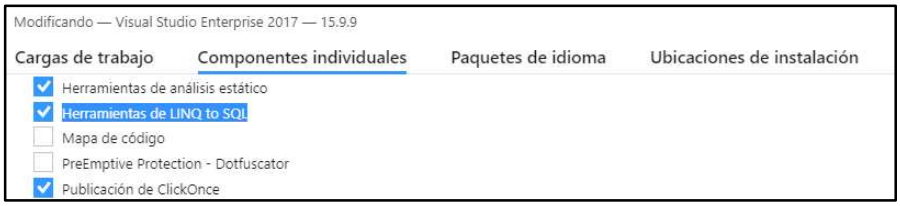


Figura 2.23. Instalación de las Herramientas de LINQ to SQL en Visual Studio.

2.2.2 ANDROID STUDIO 3.3.2

El lenguaje de programación utilizado para el cliente móvil es Java. Para la creación de las interfaces gráficas se usó XML. Para esto se utilizó Android Studio 3.3.2. El *link* de descarga de este *software* se encuentra en su página oficial.

2.2.3 MICROSOFT AZURE

Se desplegó el sistema en la plataforma informática Microsoft Azure. Para utilizar esta plataforma en primer lugar se debe crear una cuenta de Microsoft o utilizar una ya existente.

2.2.4 CONVENCIONES DEL CÓDIGO

Es importante definir la estructura del directorio del proyecto como se visualiza en la Figura 2.24. Esto permite tener un código limpio, ordenado y escalable. Para el desarrollo de código de Visual Studio se crea una solución denominada `SistemaCentroMédico` dentro de esta se crean dos proyectos `ClienteEscritorio` y `Servidor`. En el proyecto `ClienteEscritorio` se crearon cuatro carpetas, la primera es `AccesoSvc` la cual tiene los métodos que permitan al cliente de escritorio acceder al servidor, en la segunda carpeta llamada *Clases* se crearan todas las clases descritas en el diagrama de clases de la Figura 2.13, la tercera nombrada `images` almacena todas las imágenes que se van a utilizar para el diseño del sistema y la cuarta carpeta denominada `Reportes` contiene todos los diseños de reportes que se generen.

En el segundo proyecto se crearon cuatro carpetas, que se indican en la Figura 2.24, en este proyecto la primera es la que almacenó todas las clases del servidor. La segunda carpeta almacena los archivos que permiten el acceso a las tablas de la base de datos, se la denominó `Datos`. La tercera carpeta nombrada `Resultados` guarda los archivos de los resultados de los exámenes médicos. Finalmente, la cuarta carpeta denominada `Servicios` almacena los servicios WCF con sus respectivas interfaces.

Para garantizar la legibilidad del código se optó utilizar ciertos estándares de nomenclatura basados en *Camel Case*, excepto para las tablas de la base de datos, los procedimientos almacenados y las variables donde se utilizó el estándar *Snake Case*. En la Tabla 2.22 se indica cómo se aplicaron los determinados tipos de escritura. Para el desarrollo de la aplicación Android se creó un proyecto denominado `CentroMedico`, no se crearon carpetas adicionales a las que se crean de manera automática.

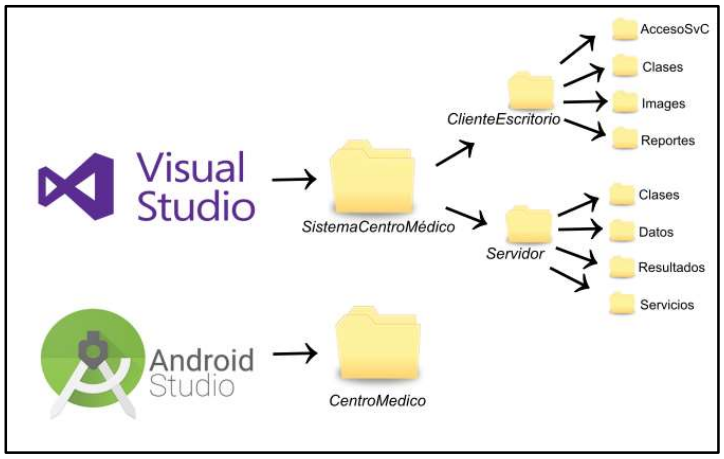


Figura 2.24. Estructura del directorio del proyecto

Tabla 2.22. Estándares de nomenclatura utilizados para el código.

Ítem	Tipo de escritura	Ejemplo
Variables	*Snake Case*	*id_persona*
Métodos	*Lower Camel Case*	*llenarDatos*
Formularios	*Upper Camel Case*	*FrmExamenes*
Clases	*Upper Camel Case*	*ConsultaMedica*
Interfaces	*Upper Camel Case*	*PageCitas*
Componentes Visuales	*Lower Camel Case*	*btnEliminar*
Tablas de la base de datos	*Snake Case*	*Consulta_medica*
Procedimientos almacenados	*Snake Case*	*sp_b_personas*

2.3 IMPLEMENTACIÓN

2.3.1 ITERACIÓN 1

Para la primera iteración se creó la base de datos en Microsoft Azure y se implementó el módulo *Login* y el del Administrador para la aplicación de escritorio. Se dejó a consideración del cliente la decisión de continuar o migrar a otro servicio de hosting una vez terminado el tiempo de periodo de prueba que ofrece la plataforma Azure.

2.3.1.1 Creación de la base de datos

Se creó la base de datos total de todo el sistema. En primer lugar, se inicia sesión en Azure y se procede a seleccionar el servicio *SQL Database* (ver Figura 2.25) y se creó una nueva base de datos sql como se indica en la Figura 2.26. Se procedió a nombrar la base de datos: `SistemaCentroMedico` y se creó un servidor sql: *epnbdd.database.windows.net*, también se agregó la contraseña y nombre de usuario del administrador como se observa en la Figura 2.27. Se esperó a que se complete la implementación para ir al recurso creado y copiar el nombre del servidor. Antes de implementar la base de datos es importante configurar el *Firewall* con las direcciones IP públicas de la red desde donde se va a conectar el sistema a la base de datos, un ejemplo se indica en la Figura 2.28.

Figura 2.25. Creación de una nueva base de datos sql en Azure

Figura 2.26. Configuración del nombre y servidor de la base de datos

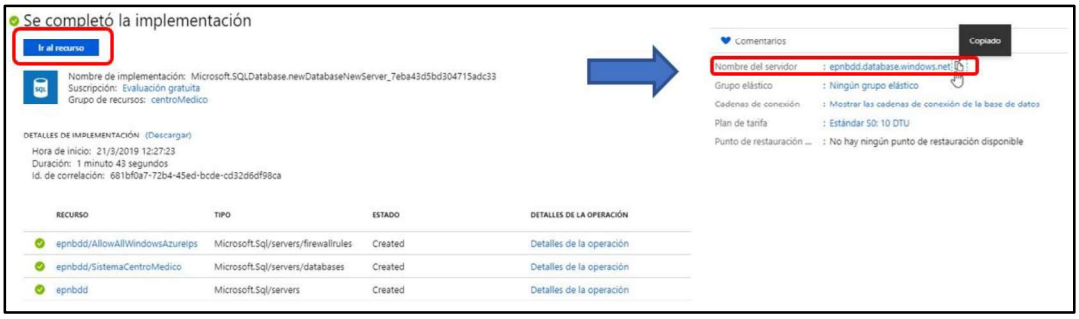


Figura 2.27. Implementación final de la base de datos

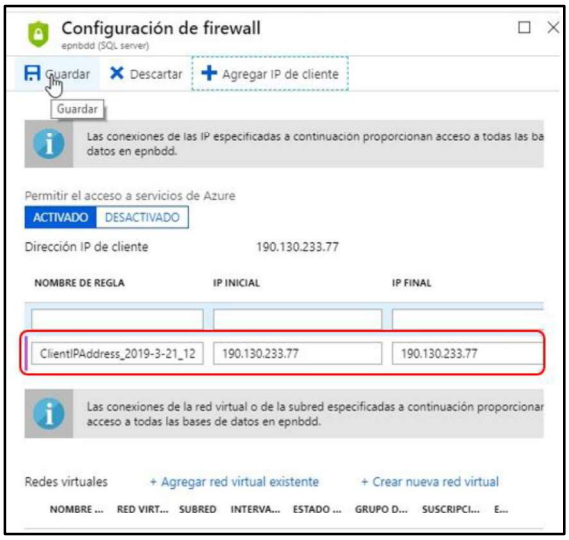


Figura 2.28. Configuración del Firewall de la base de datos

Una vez creada la base de datos en Azure, se procede en Visual Studio a añadir la conexión de base de datos como se visualiza en la Figura 2.29. Se ingresó el nombre del servidor de la base de datos y las credenciales del Administrador y seleccionar la base de datos que se va a utilizar (*Sistema Centro Médico*).

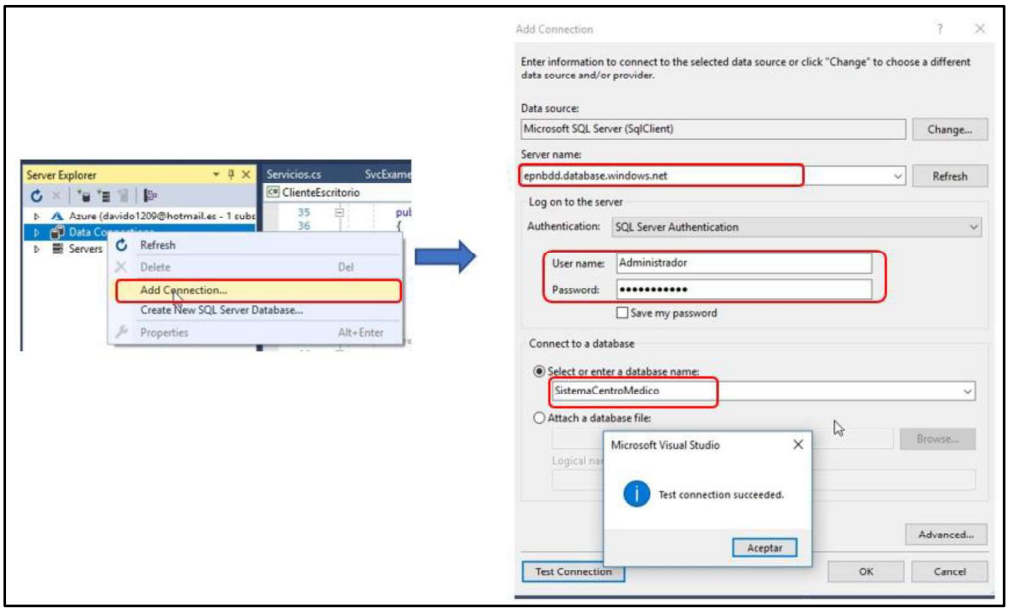


Figura 2.29 Añadir la conexión de la base de datos en Visual Studio

Se creó el script para la creación de la base de datos (Ver Anexo D) esta consulta se ejecutó en Visual Studio como se observa en la Figura 2.30. La primera tabla que se creó fue `personas` como se indica en el Segmento de Código 2.1 en la línea 1 se utiliza el comando `create` para crear una tabla con 12 columnas que corresponden a los datos de las personas que serán registradas en el sistema. Finalmente se puede visualizar todas las tablas de la base de datos en Figura 2.31.

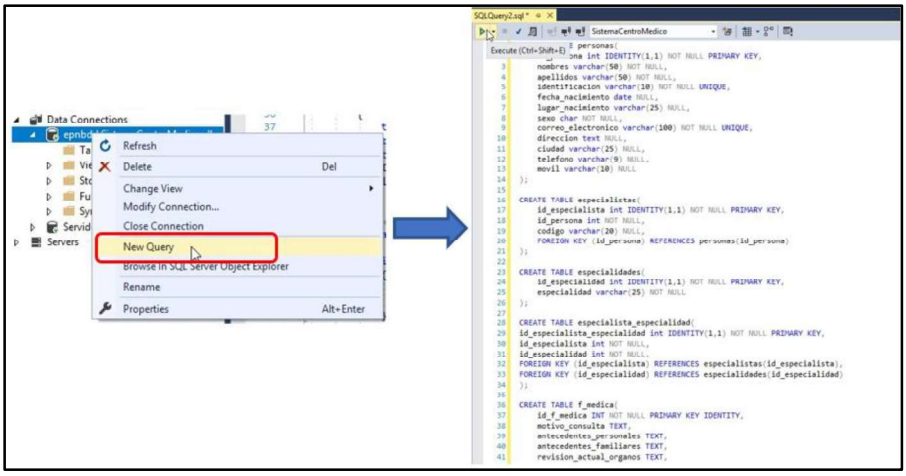


Figura 2.30 Ejecución del script de la base de datos

```sql
CREATE TABLE personas(
    id_persona int IDENTITY(1,1) NOT NULL PRIMARY KEY,
    nombres varchar(50) NOT NULL,
    apellidos varchar(50) NOT NULL,
    identificacion varchar(10) NOT NULL UNIQUE,
    fecha_nacimiento date NULL,
    lugar_nacimiento varchar(25) NULL,
    sexo char NOT NULL,
    correo_electronico varchar(100) NOT NULL UNIQUE,
    direccion text NULL,
    ciudad varchar(25) NULL,
    telefono varchar(9) NULL,
    movil varchar(10) NULL
);
```

Segmento de Código 2.1. Ejemplo de la creación de una tabla

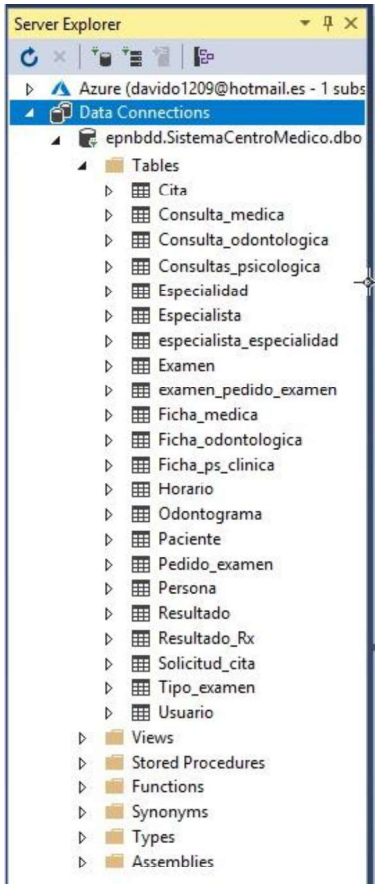


Figura 2.31. Tablas de la base de datos implementada en Azure en Visual Studio

2.3.1.2 Módulo Login

En primer lugar, para el desarrollo del módulo `Login` se realizó el diseño de la interfaz gráfica en base a los mockups realizados anteriormente. Para esto se utilizó el lenguaje xaml. En el Segmento de Código 2.2 se presenta las líneas de código utilizadas para este diseño, el nombre de esta ventana es `MainWindow` y se encuentra dentro del proyecto

`ClienteEscritorio` como se indica en la línea 1. De la línea 2 a la 11 se encuentra la configuración inicial de la ventana de `Login`, para el desarrollo de todas las interfaces gráficas se utilizó los íconos del repositorio web *fontawesome* como se observa en la línea 6. En las líneas 12 a la 29 se definen el número y tamaño de las filas y columnas que se van a utilizar.

```
1  <Window x:Class="ClienteEscritorio.MainWindow"
2          xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3          xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4          xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
5          xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
6          xmlns:fa="http://schemas.fontawesome.io/icons/"
7          xmlns:local="clr-namespace:ClienteEscritorio"
8          mc:Ignorable="d"
9          xmlns:gif="http://wpfanimatedgif.codeplex.com"
10         Title="Login" Height="450" Width="453.151" Icon="images/logo.png"
11         WindowStartupLocation="CenterScreen">
12     <Grid>
13         <Grid.ColumnDefinitions>
14             <ColumnDefinition />
15             <ColumnDefinition Width="125"/>
16             <ColumnDefinition Width="150"/>
17             <ColumnDefinition />
18         </Grid.ColumnDefinitions>
19         <Grid.RowDefinitions>
20             <RowDefinition />
21             <RowDefinition Height="70"/>
22             <RowDefinition Height="35"/>
23             <RowDefinition Height="35"/>
24             <RowDefinition Height="35"/>
25             <RowDefinition Height="30"/>
26             <RowDefinition Height="30"/>
27             <RowDefinition />
28             <RowDefinition />
29         </Grid.RowDefinitions>
```

Segmento de Código 2.2. Diseño de la interfaz del módulo en xaml (parte 1 de 2)

En el Segmento de Código 2.3 se encuentran todos los controles utilizados para el desarrollo de la interfaz gráfica con sus respectivas propiedades. El nombre de usuario será ingresado en un *textbox* denominado `txtUsuario` (ver línea 35). La contraseña será ingresada en un *textbox* denominado `txtPassword`, se utilizó el control `PasswordBox` para que no se visualicen los caracteres que serán ingresados como contraseña y en lugar de éstos aparezca un punto por cada caracter ingresado (ver línea 36). Desde la línea 40 hasta la 45 se encuentra el control *combox* que permite elegir el tipo de usuario. Adicional se colocó un *Hyperlink* en caso de olvido de contraseña en la línea 47. También aparecen mensajes que indiquen los errores que pueden cometer las personas al ingresar sus datos, para esto en la línea 50 se creó un *label* denominado `lblMensaje`. A partir de la línea 52 se encuentran las funciones y características que tiene la imagen para actuar como un botón para acceder al sistema. El resultado del diseño de la interfaz gráfica del módulo *Login* se visualiza en la Figura 2.32.

```
30          <Image Source="images/login.jpg" Grid.Column="0" Grid.Row="0" Grid.ColumnSpan="4" Grid.RowSpan="9"
31                 Opacity="0.3" Stretch="Fill" />
32          <fa:ImageAwesome Icon="Users" Grid.Column="1" Grid.ColumnSpan="2" Grid.Row="1" Margin="10,10,10,10"/>
33          <Label Grid.Column="1" Grid.Row="2"  Foreground="Black" Content="Usuario:" FontSize="14" FontWeight="Bold"/>
34          <Label Grid.Column="1" Grid.Row="4" Foreground="Black" Content="Contraseña:" FontSize="14" FontWeight="Bold"/>
35          <TextBox x:Name="txtUsuario" Text="" Grid.Column="2" Grid.Row="2" Margin="0,5,0,5" FontSize="14"/>
36          <PasswordBox x:Name="txtPassword" Password="" Grid.Column="2" Grid.Row="4" Margin="0,5,0,5"
37                       KeyUp="TxtPassword_KeyUp" GotFocus="TxtPassword_GotFocus" FontSize="14"/>
38          <Label Grid.Column="1" Grid.Row="3" Foreground="Black" Content="Tipo de Usuario:" FontSize="14"
39                 FontWeight="Bold"/>
40          <ComboBox x:Name="cmbTipo" Grid.Column="2" Grid.Row="3" Margin="0,5,0,5" SelectedValue="x:Name" FontSize="14">
41              <ComboBoxItem x:Name="A">Administrador</ComboBoxItem>
42              <ComboBoxItem x:Name="R">Recepción</ComboBoxItem>
43              <ComboBoxItem x:Name="M" IsSelected="True">Médico</ComboBoxItem>
44              <ComboBoxItem x:Name="L">Laboratorio Clínico</ComboBoxItem>
45          </ComboBox>
46          <TextBlock Grid.Row="6" Grid.Column="1" Grid.ColumnSpan="2" HorizontalAlignment="Center" Margin="55,0,64,0">
47                  <Hyperlink x:Name="linkOlvidoPassword" Click="linkOlvidoPassword_Click" >
48                      <Run Text="¿Has olvidado tu contraseña?"/></Hyperlink>
49          </TextBlock>
50          <Label x:Name="lblMensaje" Content="" Grid.Column="2" Grid.Row="5" HorizontalAlignment="Right"
51                 Foreground="Red" Margin="0,0,0,4"/>
52          <fa:ImageAwesome x:Name="btnAcceder" Icon="SignIn" Grid.Column="1" Grid.ColumnSpan="2"
53                           HorizontalAlignment="Center" Grid.Row="7" Width="40" Height="40" Margin="0 0 0 0 "
54                           ToolTip="Iniciar Sesión" MouseEnter="BtnAcceder_MouseEnter"
55                           MouseLeave="BtnAcceder_MouseLeave" MouseLeftButtonDown="BtnAcceder_MouseLeftButtonDown"
56                           MouseLeftButtonUp="BtnAcceder_MouseLeftButtonUp" />
57          <Image x:Name="imgCargando" gif:ImageBehavior.AnimatedSource="images/cargando.gif" Grid.RowSpan="9"
58                 Grid.ColumnSpan="4" Visibility="Hidden" Stretch="Fill" Opacity="0.5"/>
59          <Button x:Name="btnPrueba" Content="Prueba" Click="BtnPrueba_Click" Visibility="Hidden" />
60      </Grid>
61  </Window>
```

Segmento de Código 2.3. Diseño de la interfaz del módulo en xaml (parte 2 de 2)

Figura 2.32. Interfaz gráfica del módulo Login

En el Segmento de Código 2.4 encuentran los servicios declarados para el módulo `Login`. A través de la expresión `[ServiceContract]` se declararon tres servicios en la interfaz `ISvcLogin`: el primer servicio correspondiente al servicio de autenticación se lo denominó `login`. Los otros dos servicios son para la recuperar y cambiar la contraseña: `olvidoPassword` y `cambiarPassword`. Sólo los métodos decorados con `[OperationContract]` están expuestos al cliente. Adicional se utilizó el método POST para la transacción de datos en formato JSON. En las líneas 17, 22 y 27 se encuentran los argumentos de entrada y salida para cada servicio declarado.

```
9   namespace Servidor.Servicios
10  {
11      [ServiceContract]
        1 referencia | 0 cambios | 0 autores, 0 cambios
12      public interface ISvcLogin
13      {
14          [OperationContract]
15          [WebInvoke(Method = "POST", UriTemplate = "login", BodyStyle = WebMessageBodyStyle.WrappedRequest,
16              RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json)]
            1 referencia | 0 cambios | 0 autores, 0 cambios
17          Clases.RespLogin login(String nombre_usuario, string password, char tipo_usuario);
18
19          [OperationContract]
20          [WebInvoke(Method = "POST", UriTemplate = "olvidoPassword", BodyStyle = WebMessageBodyStyle.WrappedRequest,
21              RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json)]
            1 referencia | 0 cambios | 0 autores, 0 cambios
22          Clases.RespLogin olvidoPassword(String nombre_usuario);
23
24          [OperationContract]
25          [WebInvoke(Method = "POST", UriTemplate = "cambiarPassword", BodyStyle = WebMessageBodyStyle.WrappedRequest,
26              RequestFormat = WebMessageFormat.Json, ResponseFormat = WebMessageFormat.Json)]
            1 referencia | 0 cambios | 0 autores, 0 cambios
27          String cambiarPassword(int id_usuario, string password_actual, string password_nuevo);
28      }
29  }
```

Segmento de Código 2.4. Servicios WCF para el módulo *login*

En el Segmento de Código 2.6 se encuentra parte del código desarrollado para el módulo Login, se creó un nuevo hilo, para no bloquear la interfaz principal, a través de la sentencia `new Thread(() =>` para acceder al servicio *login* el cual pide de argumentos de entrada: *usuario, password*, *tipo* y como salida se tiene un objeto de tipo `respLogin` (ver línea 48). Los métodos que permiten al cliente de escritorio comunicarse con el servidor se encuentran en el archivo *Servicios* dentro de la carpeta `AccesoSvc`; a manera de ejemplo en el Segmento de Código 2.5 se encuentra el método `login` el cual envía en formato JSON los datos de entrada solicitados por el servicio WCF y serializa la respuesta para posteriormente convertirla en un objeto de tipo `respLogin`.

En el Segmento de Código 2.6 se encuentra parte del código desarrollado para poder seleccionar el tipo de usuario: administrador (A), recepcionista (R), médico (M) y laboratorista (L), para que si la autenticación se realiza de manera correcta el usuario acceda a su ventana determinada: `MasterAdmin`, `MasterRecepcion`, `MasterMedico`, `MasterLaboratorio` respectivamente (líneas 58, 69, 81 y 93).

En el Segmento de Código 2.7 se encuentra el método `linkOlvidoPassword_Click()` que al dar *click* en el enlace para recuperación de contraseña hace que en primer lugar aparezca un `MessageBox` para confirmar al usuario si desea restablecer la contraseña (ver línea 113). En caso de que la respuesta sea afirmativa se crea el objeto tipo `RespLogin` para obtener la respuesta del servidor el cual envía el correo electrónico al usuario para continuar con el proceso de recuperación de contraseña.

```
15  public static RespLogin login(string nombre_usuario, string password, char tipo)
16  {
17      RespLogin respLogin = new RespLogin();
18      WebClient cliente = new WebClient() { Encoding = Encoding.UTF8 };
19      string json = string.Format(
20          "{{" +
21          "\"nombre_usuario\": \"{0}\"," +
22          "\"password\":\"{1}\"," +
23          "\"tipo_usuario\":\"{2}\"" +
24          "}}", nombre_usuario, password, tipo);
25      cliente.Headers[HttpRequestHeader.ContentType] = "application/json";
26      string result = cliente.UploadString($"{Properties.Settings.Default.ServerName}/Servicios/SvcLogin.svc/login", json);
27      DataContractJsonSerializer serializadorJson = new DataContractJsonSerializer(typeof(RespLogin));
28      respLogin = (RespLogin)serializadorJson.ReadObject(new MemoryStream(Encoding.Unicode.GetBytes(result)));
29
30      return respLogin;
31  }
```

Segmento de Código 2.5. Método `login`

```
46      new Thread(() =>
47      {
48          Clases.RespLogin respLogin = AccesoSvc.Servicios.login(usuario, password, tipo);
49          if (respLogin.Exito)
50          {
51              lblMensaje.Dispatcher.BeginInvoke(new Action(() => { lblMensaje.Content = "Exito"; }));
52              imgCargando.Dispatcher.BeginInvoke(new Action(() => { imgCargando.Visibility = Visibility.Collapsed; }));
53              if (respLogin.Tipo_usuario == 'A')
54              {
55                  this.Dispatcher.BeginInvoke(new Action(() =>
56                  {
57                      this.Hide();
58                      MasterAdmin frmAdmin = new MasterAdmin(respLogin.Id_usuario, respLogin.Id_persona, respLogin.Id_especialista);
59                      txtPassword.Password = "";
60                      frmAdmin.ShowDialog();
61                      this.Show();
62                  }));
63              }
64              if (respLogin.Tipo_usuario == 'R')
65              {
66                  this.Dispatcher.BeginInvoke(new Action(() =>
67                  {
68                      this.Hide();
69                      MasterRecepcion frmRecep = new MasterRecepcion(respLogin.Id_usuario, respLogin.Id_persona,
70                          respLogin.Id_especialista);
71                      txtPassword.Password = "";
72                      frmRecep.ShowDialog();
73                      this.Show();
74                  }));
75              }
```

Segmento de Código 2.6. Selección del tipo de usuario en el módulo *Login* (parte 1 de 2)

```
109     private void linkOlvidoPassword_Click(object sender, RoutedEventArgs e)
110     {
111         if (txtUsuario.Text.Trim() != "")
112         {
113             MessageBoxResult result = MessageBox.Show("¿Desea continuar?", "Restablecer Contraseña", MessageBoxButton.YesNo);
114             if (result == MessageBoxResult.Yes)
115             {
116                 Clases.RespLogin resp = new Clases.RespLogin();
117                 resp = AccesoSvc.Servicios.olvidoPassword(txtUsuario.Text);
118                 MessageBox.Show(resp.Mensaje);
119             }
120         }
121         else
122         {
123             MessageBox.Show("Ingrese su nombre de usuario");
124         }
125     }
```

Segmento de Código 2.7. Método `linkOlvidoPassword_Click`

2.3.1.3 Módulo Administración de Usuarios

La interfaz gráfica para los usuarios con perfil de administrador se denomina `MasterAdmin` y se visualiza en la Figura 2.33. El Segmento de Código 2.8 muestra una parte del código en xaml desarrollado para esta interfaz, desde la línea 41 a la 50 se encuentra un control *StackPanel* para agregar los botones que permiten realizar acciones CRUD sobre una persona seleccionada. Empezando en la línea 53 hasta la 63 se encuentra el *ListView* que contiene una lista denominada `lstPersonas` que contiene el nombre, apellido, identificación y correo de las personas registradas en el sistema.

La interfaz `MasterAdmin` tiene la opción de buscar personas y filtrar la lista según el tipo de persona, para esta opción se utilizó procedimientos almacenados y de esta manera traer de la base de datos información determinada. En el Segmento de Código 2.9 se encuentra el procedimiento almacenado `sp_b_personas`. En las líneas 57 y 58 se declaran las variables a utilizar, desde la línea 61 a la 65 se selecciona a las personas que contengan en sus nombres, apellidos, identificación o correo electrónico el valor de la variable `@buscar` y que su tipo de usuario corresponda al valor de `@tipo`. Todos los procedimientos almacenados utilizados en el desarrollo del sistema se encuentran en el ANEXO E.

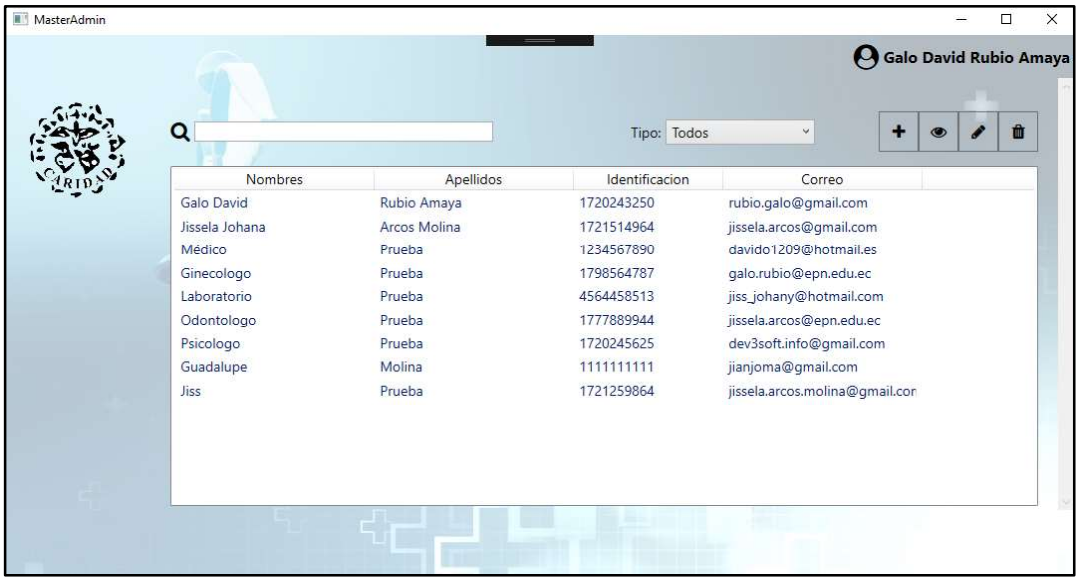


Figura 2.33. Interfaz gráfica *MasterAdmin*

```
41      <StackPanel Grid.Column="1" Grid.Row="1" Orientation="Horizontal" HorizontalAlignment="Right" VerticalAlignment="Bottom"
42                  Margin="0,0,20,0" >
43          <Button x:Name="btnCrear" fa:Awesome.Content="Plus"  VerticalAlignment="Center" Width="40" Height="40"
44                  Click="BtnCrear_Click" FontSize="16" Background="{x:Null}"/>
45          <Button x:Name="btnRead" fa:Awesome.Content="Eye" VerticalAlignment="Center" Width="40" Height="40"
46                  Click="btnRead_Click" FontSize="16" Background="{x:Null}"/>
47          <Button x:Name="btnUpdate" fa:Awesome.Content="Pencil" VerticalAlignment="Center" Width="40" Height="40"
48                  Click="btnUpdate_Click" FontSize="16" Background="{x:Null}"/>
49          <Button x:Name="btnDelete" fa:Awesome.Content="Trash" VerticalAlignment="Center"  Width="40" Height="40"
50                  Click="BtnDelete_Click" FontSize="16" Background="{x:Null}"/>
51      </StackPanel>
52
53      <ListView x:Name="lstPersonas" Grid.Column="0" Grid.ColumnSpan="2" Grid.Row="3" Margin="20,0,20,0"
54                MouseDoubleClick="LstPersonas_MouseDoubleClick" FontSize="14">
55          <ListView.View>
56              <GridView>
57                  <GridViewColumn Header="Nombres" Width="200" DisplayMemberBinding="{Binding Nombres}" />
58                  <GridViewColumn Header="Apellidos" Width="200" DisplayMemberBinding="{Binding Apellidos}" />
59                  <GridViewColumn Header="Identificacion" Width="150" DisplayMemberBinding="{Binding Identificacion}" />
60                  <GridViewColumn Header="Correo" Width="200" DisplayMemberBinding="{Binding Correo_electronico}" />
61              </GridView>
62          </ListView.View>
63      </ListView>
```

Segmento de Código 2.8. Parte del código en xaml del diseño de la interfaz gráfica del módulo de Administración

```
56  CREATE PROCEDURE sp_b_personas
57  @buscar VARCHAR(100),
58  @tipo VARCHAR(1)
59  AS
60  SELECT * FROM Persona
61  WHERE (nombres LIKE '%'+@buscar+'%'
62  OR apellidos LIKE '%'+@buscar+'%'
63  OR identificacion LIKE '%'+@buscar+'%'
64  OR correo_electronico LIKE '%'+@buscar+'%')
65  AND id_persona IN( SELECT u.id_persona FROM Usuario u WHERE u.tipo_usuario LIKE @tipo+'%');
```

Segmento de Código 2.9. Procedimiento almacenado `sp_b_personas`

A continuación, se encuentran enumerados los *UriTemplates* de los servicios WCF declarados en la interfaz `ISvcAdminUsuarios` para la administración de usuarios.

- crearUsuario
- actualizarUsuario
- esUnicoCorreo
- esUnicaIdentificacion
- traerPersonaById
- traerPersonaByIdPaciente
- traerPersonasAll
- traerPersonasTipo
- traerEspecialidadesAll
- traerEspecialidadesByPersona
- traerEspecialidadesByIdEspecialista
- traerEspecialista
- traerEspecialistaByIdPersona
- TraerEspecialistasByIdEspecialidad
- traerEspecialidad
- editarMiPerfil

- traerNomEspecialistasByIdEspecialidad

En la Figura 2.34 se muestra la interfaz gráfica *PersonaCRUD*, la cual se abre al realizar las actividades CRUD sobre una persona seleccionada. En el Segmento de Código 2.10 se encuentra parte del método `llenardatos()` que llama al servicio `traerPersonaById` para cargar los datos de una persona seleccionada en la interfaz *PersonaCRUD*. El Segmento de Código 2.11 muestra el método `Worker_DoWork` el cual se ejecuta en un hilo secundario y consume los servicios `crearUsuario` y `actualizarUsuario`.

Figura 2.34. Interfaz gráfica *PersonaCRUD*

```
95      private void llenarDatos()
96      {
97          try
98          {
99              Clases.Persona persona = new Clases.Persona();
100             persona = AccesoSvc.Servicios.traerPersonaById(idPersona);
101             txtNombres.Text = persona.Nombres;
102             txtApellidos.Text = persona.Apellidos;
103             txtIdentificacion.Text = persona.Identificacion;
104             dtFechaNac.Text = persona.Fecha_nacimiento.Date.ToString();
105             txtLugarNac.Text = persona.Lugar_nacimiento;
106             if (persona.Sexo == 'M')
107                 cmbSexo.SelectedIndex = 0;
108             else if (persona.Sexo == 'F')
109                 cmbSexo.SelectedIndex = 1;
110             txtCorreoElectronico.Text = persona.Correo_electronico;
111             txtDireccion.Text = persona.Direccion;
112             txtCiudad.Text = persona.Ciudad;
113             txtTelefono.Text = persona.Telefono;
114             txtMovil.Text = persona.Movil;
115             List<Clases.Usuario> usuarios = new List<Clases.Usuario>();
116             usuarios = AccesoSvc.Servicios.traerUsuariosByIdPersona(idPersona);
117             foreach (var i in usuarios)
118             {
119                 if (i.tipo_usuario == 'P')
120                     esPaciente = true;
121                 if (i.tipo_usuario != 'P')
122                 {
123                     esEspecialista = true;
124                     txtCodigo.Text = i.Nombre_usuario;
125                 }
126             }
```

Segmento de Código 2.10. Parte del método `llenardatos()`

```
168     private void Worker_DoWork(object sender, DoWorkEventArgs e)
169     {
170         if (modo == 'C')
171         {
172             try
173             {
174                 exito = AccesoSvc.Servicios.crearUsuario(nombres, apellidos, identificacion, fechaNac, lugarNac, sexo,
175                     correo, direccion, ciudad, telefono, movil, esPaciente, esEspecialista, tipoUsuario, l_especialidades);
176             }
177             catch (Exception ex)
178             {
179                 exito = false;
180                 MessageBox.Show("ERROR: " + ex.Message);
181             }
182         }
183         if (modo == 'U')
184         {
185             try
186             {
187                 exito = AccesoSvc.Servicios.actualizarUsuario(idPersona, nombres, apellidos, identificacion, fechaNac, lugarNac,
188                     sexo, correo, direccion, ciudad, telefono, movil, esPaciente, esEspecialista, tipoUsuario, l_especialidades);
189             }
190             catch (Exception ex)
191             {
192                 exito = false;
193                 MessageBox.Show("ERROR: " + ex.Message);
194             }
195         }
196     }
```

Segmento de Código 2.11. Método `Worker_DoWork`

En el Segmento de Código 2.12 se observa una parte del método `crearUsuario()` que se encuentra en el servicio WCF `SvcAdminUsuarios.svc.cs`. En la línea 48 empieza el condicional *if* para el caso de que el usuario creado sea un paciente, en la línea 45 se le genera una clave temporal, para lo cual llama al método `generarClave`, el cual se encuentra en la clase `Cifrado`. En las líneas 58 y 59 se asigna como nombre de usuario el correo electrónico y con el método `encriptarPassword()` se obtiene el *hash* de la contraseña del usuario. En las líneas 67 a la 71 se crea el *string* `mensaje` que corresponde al texto que se envía por correo electrónico al paciente. En la línea 72 se llama al método `enviar()` de la clase `EnviarCorreo`. Para la creación de un especialista el código es similar al del paciente, pero el nombre de usuario se genera automáticamente y se forma con el tipo de usuario (A, M, R o L), la primera letra del nombre y del apellido del especialista y un número secuencial como se indica en el Segmento de Código 2.13.

```
48      if (esPaciente)
49      {
50          Paciente miPaciente = new Paciente();
51          miPaciente.id_persona = miPersona.id_persona;
52          miDB.Paciente.InsertOnSubmit(miPaciente);
53          miDB.SubmitChanges();
54
55          string passwordTemporal = Clases.Cifrado.generarClave();
56          Usuario miUsuario = new Usuario();
57          miUsuario.id_persona = miPersona.id_persona;
58          miUsuario.nombre_usuario = miPersona.correo_electronico;
59          miUsuario.contrasena = Clases.Cifrado.encriptarPassword(Encoding.UTF8.GetBytes(passwordTemporal));
60          miUsuario.tipo_usuario = 'P';
61          miUsuario.fecha_creacion = DateTime.Now;
62          miUsuario.cambiar_clave = true;
63          miUsuario.olvido_clave = false;
64          miUsuario.usuario_activo = false;
65          miDB.Usuario.InsertOnSubmit(miUsuario);
66          miDB.SubmitChanges();
67          string mensaje = string.Format(
68              $"BIENVENID@ {miPersona.nombres} al sistema del Centro Médico Jesús de Nazareth.\n\n" +
69              $"Para completar el registro del usuario: ({miUsuario.nombre_usuario}) su contraseña temporal será: {passwordTemporal}. " +
70              $"Para activar su usuario favor actualice su contraseña ingresando al siguiente enlace: " +
71              $"<a href=\"{Properties.Settings.Default.ServerName}/activarCuenta.aspx/?valor={miUsuario.id_usuario}\">Activar Cuenta</a>");
72          bool correoEnviado = Clases.EnviarCorreo.enviar("Activación Cuenta", mensaje, miPersona.correo_electronico);
73          Console.WriteLine($"correo enviado:{correoEnviado}");
74      }
```

Segmento de Código 2.12. Parte del método `crearUsuario()` para un usuario paciente

```
81      miUsuario.nombre_usuario = tipo_usuario.ToString() + miPersona.nombres[0].ToString()
82          + miPersona.apellidos[0].ToString() + miPersona.id_persona.ToString();
```

Segmento de Código 2.13. Generación del nombre de usuario de un especialista

En el Segmento de Código 2.14 se encuentra la clase `Cifrado` la cual tiene dos métodos para el manejo de las contraseñas: `encriptarPassword()` y `generarClave()`. Adicional se tiene un tercer método denominado `escribirLog()` el cual escribe una nueva línea de texto en el archivo *log.txt* y almacena información sobre los errores que ocurren en el sistema.

```
11  public class Cifrado
12  {
        14 referencias | 0 cambios | 0 autores, 0 cambios
13      public static byte[] encriptarPassword(byte[] bytesPassword)
14      {
15          var sha256 = SHA256.Create();
16          return sha256.ComputeHash(bytesPassword);
17      }
18
        7 referencias | 0 cambios | 0 autores, 0 cambios
19      public static String generarClave()
20      {
21          Random random = new Random();
22          string clave = random.Next(100000, 999999).ToString();
23          return clave;
24      }
25
        63 referencias | 0 cambios | 0 autores, 0 cambios
26      public static void escribirLog(string texto)
27      {
28          string file = HostingEnvironment.MapPath(@"/App_Data/log.txt");
29          File.AppendAllText(file, DateTime.Now.ToString() + ": " + texto + Environment.NewLine);
30      }
31  }
```

Segmento de Código 2.14. Clase `Cifrado`

El sistema de escritorio debe enviar correos electrónicos para la activación de usuarios, cambios de contraseñas y para envío de la información de las citas agendadas para lo cual se desarrolló la clase `EnviarCorreo` (ver Segmento de Código 2.15) que permite enviar correos cuyo remitente es el correo electrónico del Centro Médico como se indica en la línea 21. Desde la línea 23 a la 28 se encuentra configuración del servidor SMTP (*Simple Mail Transfer Protocol*).

Al manejar varios datos es importante validar que la información ingresada sea correcta para lo cual en el Segmento de Código 2.16 se visualiza parte del método `validarFormulario()` este método permite visualizar mensajes de error en caso de que un campo obligatorio se encuentre vacío, un ejemplo se encuentra desde la línea 237 a la 245, en la línea 236 se llama a la clase `ControlErrores` que permite validar un correo electrónico y si los caracteres ingresados corresponden a un valor numérico. Una persona registrada en el sistema debe tener como valores únicos el correo electrónico y el número de identificación, en el Segmento de Código 2.17 se encuentra parte del método

`validarValoresUnicos()` que permite comparar los valores de correo e identificación de una persona con todas las ingresadas en el sistema.

```
10      public class EnviarCorreo
11      {
            10 referencias | 0 cambios | 0 autores, 0 cambios
12          public static bool enviar(string asunto, string mensaje, string destinatario)
13          {
14              MailMessage correo = new MailMessage();
15              correo.To.Add(destinatario);
16              correo.Subject = asunto;
17              correo.Body = mensaje;
18              correo.SubjectEncoding = System.Text.Encoding.UTF8;
19              correo.BodyEncoding = System.Text.Encoding.UTF8;
20              correo.IsBodyHtml = true;
21              correo.From = new MailAddress("centromedico.jesus.nazareth@outlook.com");
22
23              SmtpClient cliente = new SmtpClient();
24              cliente.Credentials = new NetworkCredential("centromedico.jesus.nazareth@outlook.com", "*********");
25              cliente.Port = 587;
26              cliente.EnableSsl = true;
27              cliente.Host = "smtp-mail.outlook.com";
28              try
29              {
30                  cliente.Send(correo);
31                  return true;
32              }
33              catch (Exception ex)
34              {
35                  Console.WriteLine(ex.Message);
36                  return false;
37              }
38          }
39      }
40  }
```

Segmento de Código 2.15. Clase `EnviarCorreo`

```
231     private bool validarFormulario()
232     {
233         bool formOK = false;
234         int contador = 0;
235
236         Clases.ControlErrores errores = new Clases.ControlErrores();
237         if (txtNombres.Text.Trim() != "")
238         {
239             error_nombres.Visibility = Visibility.Collapsed;
240         }
241         else
242         {
243             error_nombres.Visibility = Visibility.Visible;
244             contador++;
245         }
246         if (txtApellidos.Text.Trim() != "")
247         {
248             error_apellidos.Visibility = Visibility.Collapsed;
249         }
250         else
251         {
252             error_apellidos.Visibility = Visibility.Visible;
253             contador++;
254         }
255         if (txtIdentificacion.Text.Trim() != "")
256         {
257             error_identificacion.Visibility = Visibility.Collapsed;
258         }
```

Segmento de Código 2.16. Parte del método `validarFormulario()`

```
357     private bool validarValoresUnicos(int idPerson)
358     {
359         try
360         {
361             bool unicoCorreo = AccesoSvc.Servicios.esUnicoCorreo(txtCorreoElectronico.Text, idPerson);
362             bool unicaIdentificacion = AccesoSvc.Servicios.esUnicaIdentificacion(txtIdentificacion.Text, idPerson);
363             if (unicoCorreo)
364             {
365                 error_correo.Visibility = Visibility.Collapsed;
366             }
367             else
368             {
369                 error_correo.Visibility = Visibility.Visible;
370                 error_correo.ToolTip = $"El correo {txtCorreoElectronico.Text} ya se encuentra registrado";
371             }
372
```

Segmento de Código 2.17. Parte del método `validarValoresUnicos()`

2.3.2 ITERACIÓN 2

En la segunda iteración se realizó el módulo para la recepcionista, en primer lugar, se implementaron las interfaces gráficas diseñadas y después se desarrolló el código para la administración de citas y modificación del perfil personal de la recepcionista.

2.3.2.1 Interfaces gráficas.

La ventana Recepción, la cual tiene tres *pages* (`MiPerfil`, `PageCitas`, `CRUDSignosVitales`). `MiPerfil` que se muestra en la Figura 2.35 en la cual se muestran los principales datos de la recepcionista los cuales se pueden modificar. También se encuentra la opción para actualizar la contraseña como se indica en la Figura 2.36.
La recepcionista gestiona las citas médicas, por lo que tiene acceso al `PageCitas`, el cual se encuentra en la Figura 2.37, aquí visualiza todos los especialistas y horario. También al seleccionar un especialista accede al `CRUDCitas` el cual se encuentra en la Figura 2.38. También en esta *page* se encuentra la opción para ingresar signos vitales y acceder a la *window* `CRUDSignosVitales` que se visualiza en la Figura 2.39. La recepcionista también tiene acceso al *page* `BusquedaPersonas`.

Figura 2.35. *Page* `MiPerfil`

Figura 2.36. Window `CambiarPassword`

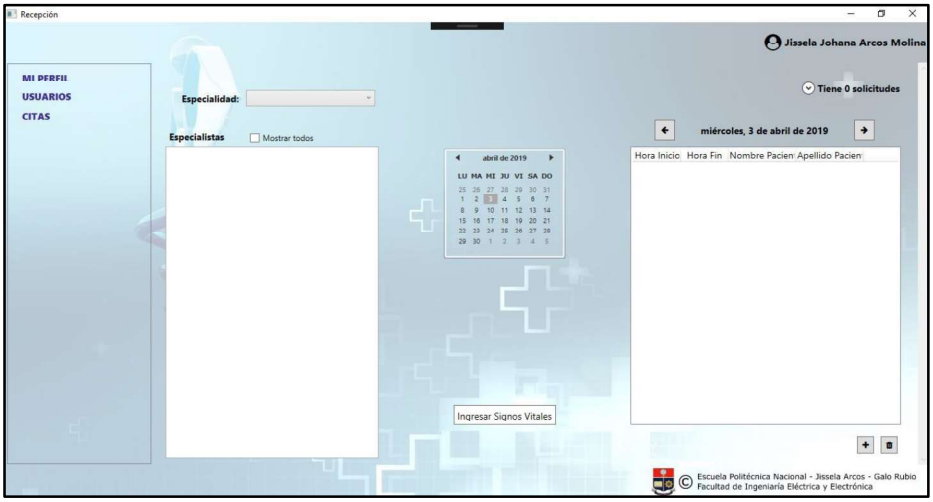


Figura 2.37. `PageCitas`

Figura 2.38. `CRUDCitas`

Figura 2.39. `CRUDSignosVitales`

2.3.2.2 Codificación del módulo recepción

A continuación, se encuentran enumerados los *UriTemplates* de los servicios WCF para la administración de citas:

- traerCitas
- traerHorariosDisponibles
- crearCita
- eliminarCita
- traerSolicitudesCita
- aprobarSolicitudCita
- reporteCita

En la clase `SvcCitas.svc.cs` se encuentran implementados los servicios mencionados anteriormente. En el Segmento de Código 2.18 se encuentra el método `traerHorariosDisponibles()` el cual pide como variables de entrada la `fecha`, el `id_especialista` y el `id_paciente`, y permite traer los horarios disponibles para el paciente y especialista seleccionado, desde la línea 55 hasta la 58 trae los horarios ocupados del especialista y paciente, después desde la línea 64 y 68 se realiza una comparación entre los horarios ocupados y todos los horarios para tener como resultado sólo los horarios disponibles.

En el Segmento de Código 2.19 se encuentra el método `crearCita()`. Desde la línea 82 a la 89 se llenan los atributos de `cita` con los parámetros de entrada. A partir de la

línea 90 hasta la 101 se configura el envío del correo electrónico con los datos de la cita creada.

El método `cambiarPassword()` se encuentra en el Segmento de Código 2.20, permite a los usuarios actualizar su contraseña, para lo cual es necesario que el usuario sepa la contraseña actual. En caso de alguno de los datos ingresados sean incorrectos se envía un mensaje indicando el error.

```
49          public List<Horario> traerHorariosDisponibles(string fecha, int id_especialista, int id_paciente)
50          {
51              List<Horario> horariosDisponibles = new List<Horario>();
52              MiDBDataContext miDB = new MiDBDataContext();
53              try
54              {
55                  var ocupados = (from c in miDB.Cita
56                                  where (c.id_especialista == id_especialista || c.id_paciente == id_paciente)
57                                  && Convert.ToDateTime(fecha).Date == c.fecha.Date
58                                  select c.Horario).Distinct().ToList();
59
60                  var todos = (from h in miDB.Horario
61                               orderby h.id_horario
62                               select h).ToList();
63
64                  foreach (var i in ocupados)
65                  {
66                      todos.Remove(i);
67                  }
68                  horariosDisponibles = todos;
69              }
70              catch (Exception ex)
71              {
72                  Cifrado.escribirLog(ex.TargetSite.ToString() + " - " + ex.Message);
73              }
74              return horariosDisponibles;
75          }
```

Segmento de Código 2.18. Método `traerHorariosDisponibles()`

```
77      public bool crearCita(string fecha, int id_horario, int id_especialista, int id_especialidad, int id_paciente)
78      {
79          MiDBDataContext miDB = new MiDBDataContext();
80          try
81          {
82              Cita cita = new Cita();
83              cita.fecha = Convert.ToDateTime(fecha);
84              cita.id_horario = id_horario;
85              cita.id_especialista = id_especialista;
86              cita.id_especialidad = id_especialidad;
87              cita.id_paciente = id_paciente;
88              miDB.Cita.InsertOnSubmit(cita);
89              miDB.SubmitChanges();
90              string mensaje = string.Format(
91                                  $"Saludos por parte del Centro Médico Jesús de Nazareth.<hr/>" +
92                                  $"Estimad@ {cita.Paciente.Persona.nombres} {cita.Paciente.Persona.apellidos}.<br/><br/> " +
93                                  $"Se ha agendado exitosamente la cita:<br/>" +
94                                  $"<b>Fecha: </b> {cita.fecha.Date.ToShortDateString()} <br/>" +
95                                  $"<b>Hora: </b> {cita.Horario.hora_inicio} <br/>" +
96                                  $"<b>Especialista: </b> {cita.Especialista.Persona.nombres} {cita.Especialista.Persona.apellidos} <br/>" +
97                                  $"<b>Especialidad: </b> {cita.Especialidad.especialidad} <br/> <hr/>" +
98                                  $"MENSAJE ENVIADO AUTOMATICAMENTE");
99              bool correoEnviado = Clases.EnviarCorreo.enviar("Cita Médica Agendada", mensaje, cita.Paciente.Persona.correo_electronico);
100             Console.WriteLine($"correo enviado:{correoEnviado}");
101             return true;
102         }
103         catch (Exception ex)
104         {
105             Cifrado.escribirLog(ex.TargetSite.ToString() + " - " + ex.Message);
106             return false;
107         }
108     }
```

Segmento de Código 2.19. Método `crearCita()`

```
122         public string cambiarPassword(int id_usuario, string password_actual, string password_nuevo)
123         {
124             Datos.MiDBDataContext miDB = new Datos.MiDBDataContext();
125             try
126             {
127                 var usuario = (from usr in miDB.Usuario
128                                where usr.id_usuario == id_usuario
129                                select usr).First();
130
131                 Byte[] hash = Clases.Cifrado.encriptarPassword(System.Text.Encoding.UTF8.GetBytes(password_actual));
132                 if (hash == usuario.contrasena)
133                 {
134                     usuario.olvido_clave = false;
135                     usuario.cambiar_clave = false;
136                     usuario.contrasena = Clases.Cifrado.encriptarPassword(System.Text.Encoding.UTF8.GetBytes(password_nuevo));
137                     miDB.SubmitChanges();
138                     return "Datos actualizados correctamente";
139                 }
140                 else
141                 {
142                     return "El password actual es incorrecto";
143                 }
144             }
145             catch (Exception ex)
146             {
147                 return "Ha ocurrido un error: \n" + ex.Message;
148             }
149         }
```

Segmento de Código 2.20. Método `cambiarPassword()`

2.3.2.3 Configuración de ReportViewer en WPF

ReportViewer es un control de libre distribución que permite incrustar informes en aplicaciones desarrolladas utilizando *.NET Framework*. Para poder utilizarlo en soluciones WPF se debe agregar como un paquete *NuGet* en Visual Studio como se observa en la Figura 2.40. Se desarrollaron varios reportes para el sistema prototipo. El primero fue el de la cita médica el diseño de este se visualiza en la Figura 2.41. También se diseñaron reportes para las fichas médicas, psicológicas, odontológicas y para las solicitudes de exámenes. Para la inserción de texto, tablas y variables se debe configurar *parameters* y *datasets.* Para la creación de los diferentes reportes se creó la *window* `Reporte` en la cual se encuentra el control de *ReportViewer* para esto se agregó el Segmento de Código 2.21 en la programación xaml de la ventana.

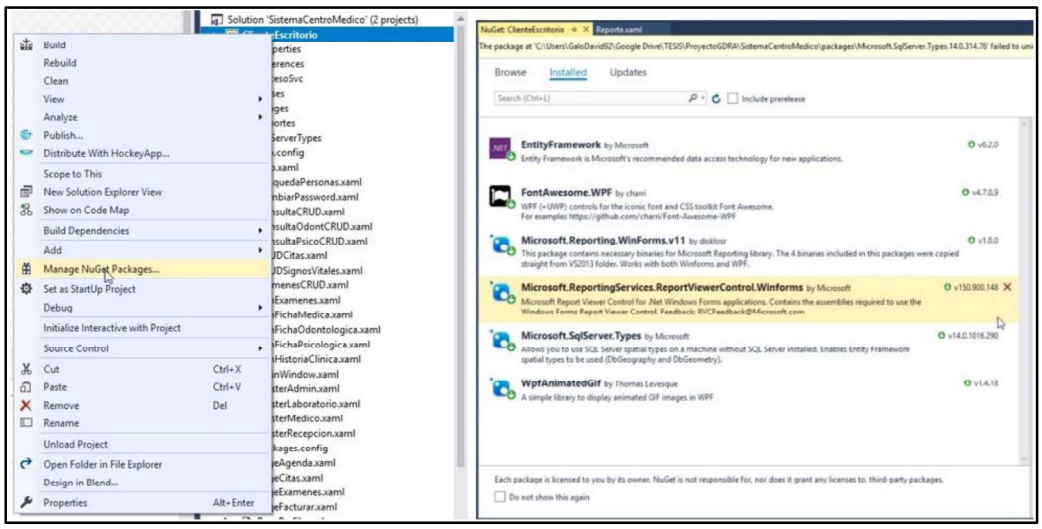


Figura 2.40. Inserción del paquete *NuGet* para utilizar *ReportViewer* en WPF

Figura 2.41. Diseño del reporte para citas médicas.

```
6       xmlns:rv="clr-namespace:Microsoft.Reporting.WinForms;assembly=Microsoft.ReportViewer.WinForms"
```

Segmento de Código 2.21. Referencia a *ReportViewer*

2.3.3 ITERACIÓN 3

En la tercera iteración se realizaron los módulos que utilizarán los especialistas para la administración de historias médicas y para la gestión de citas médicas.

2.3.3.1 Interfaces gráficas

Los especialistas al dar doble *click* sobre un paciente acceden a la ventana `HistoriaClinica`, la cual contiene un menú, en el cual pueden acceder a la ficha médica, odontológica, psicológica o exámenes médicos del paciente cuyos datos identificativos aparecen en la parte superior, como se puede apreciar en la Figura 2.44. En la Figura 2.43 se encuentra la interfaz gráfica diseñada para la ficha médica. Por otro lado, la Figura 2.44 muestra la interfaz gráfica de la consulta médica la cual será utilizada en cada consulta que tenga el paciente. Para los odontólogos se diseñó la ficha correspondiente al *window* `FrmFichaOdontologica` la cual se encuentra en la Figura 2.45 y la Figura 2.46, para cada consulta odontológica se tiene la interfaz correspondiente al *window* consultaOdontCRUD (Figura 2.47). Para la ficha psicológica se diseñó la interfaz gráfica `FrmConsultaPsicológica` que se muestra en la Figura 2.48 y para sus

respectivas consultas la *window* `ConsultaPsicoCrud` que se observa en la Figura 2.49. Todas las interfaces gráficas diseñadas permiten ingresar los datos según los requerimientos solicitados. La historia clínica se compone por todas fichas y exámenes de los pacientes, esta puede ser descargada en formato PDF para lo cual se diseñaron reportes en *ReportViewer.*

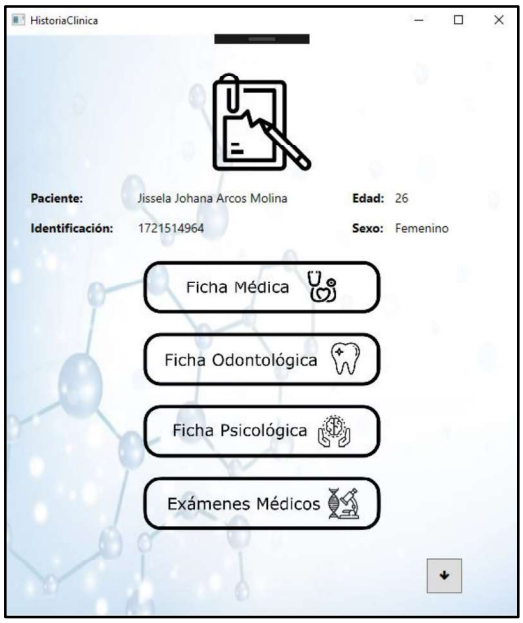


Figura 2.42. *Window* `HistoriaClinica`

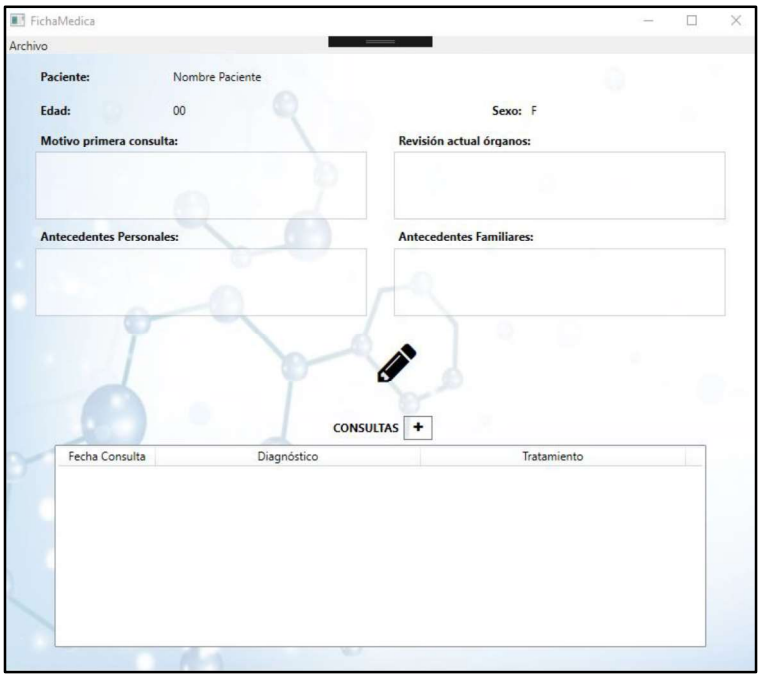


Figura 2.43. *Window* `FichaMedica`

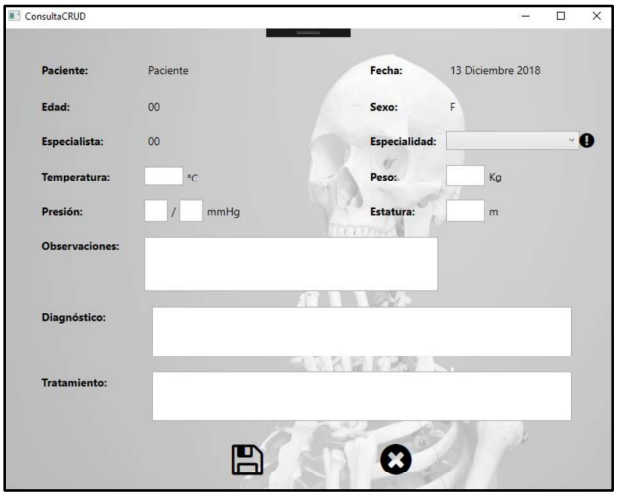


Figura 2.44. *Window* `ConsultaCRUD`

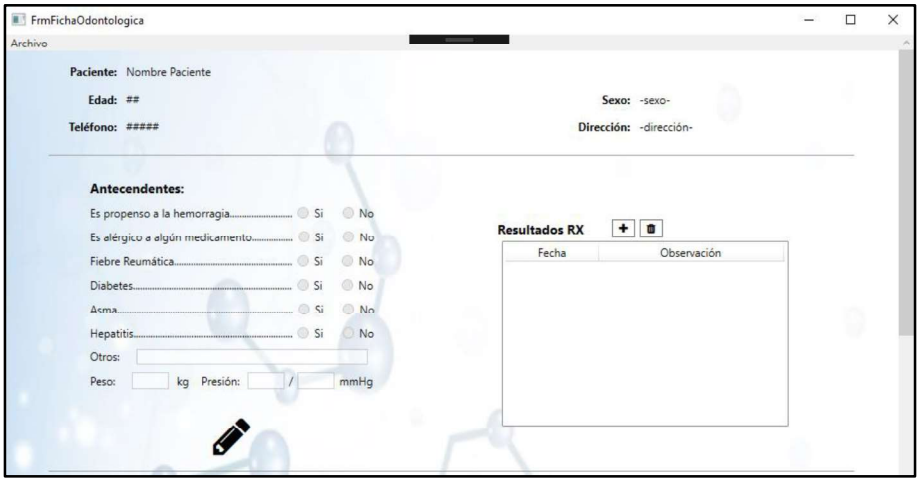


Figura 2.45. *Window* `FrmFichaOdontologica` (parte 1 de 2)

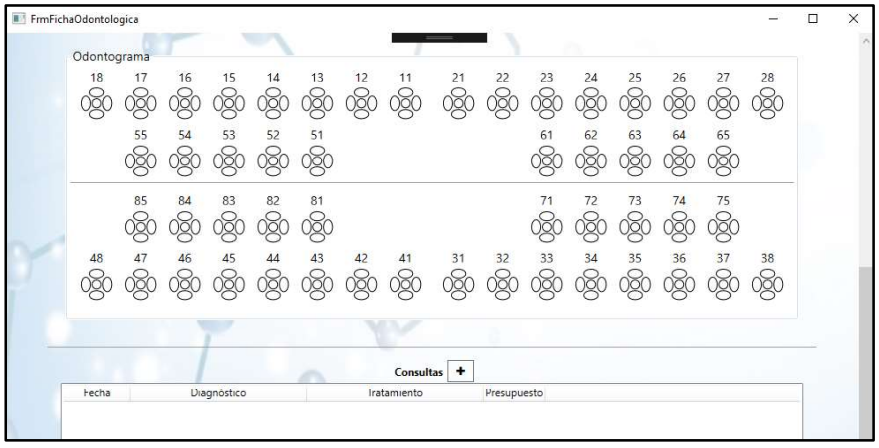


Figura 2.46. *Window* `FrmFichaOdontologica` (parte 2 de 2)

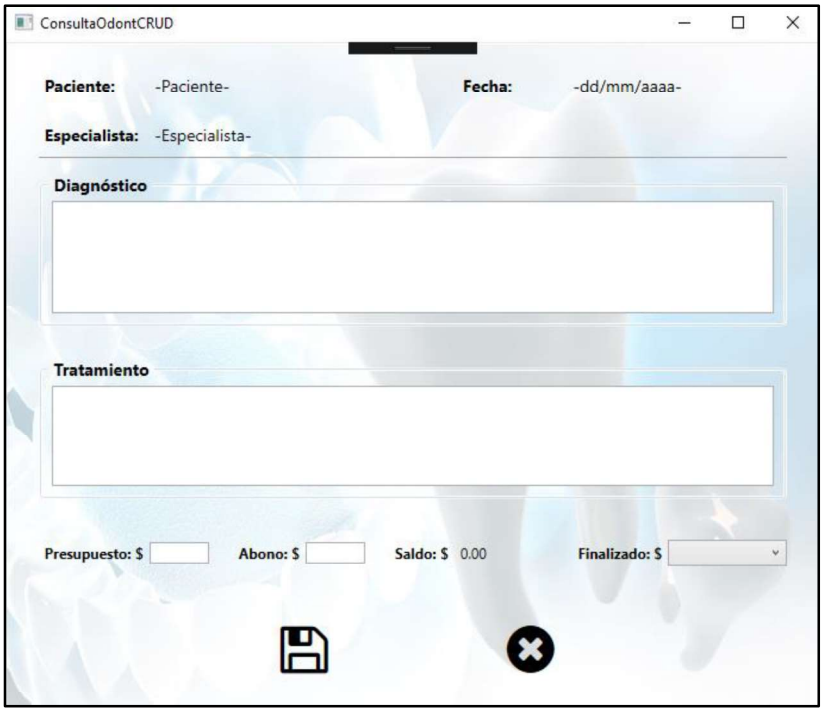


Figura 2.47. *Window* `ConsultaOdontCRUD`

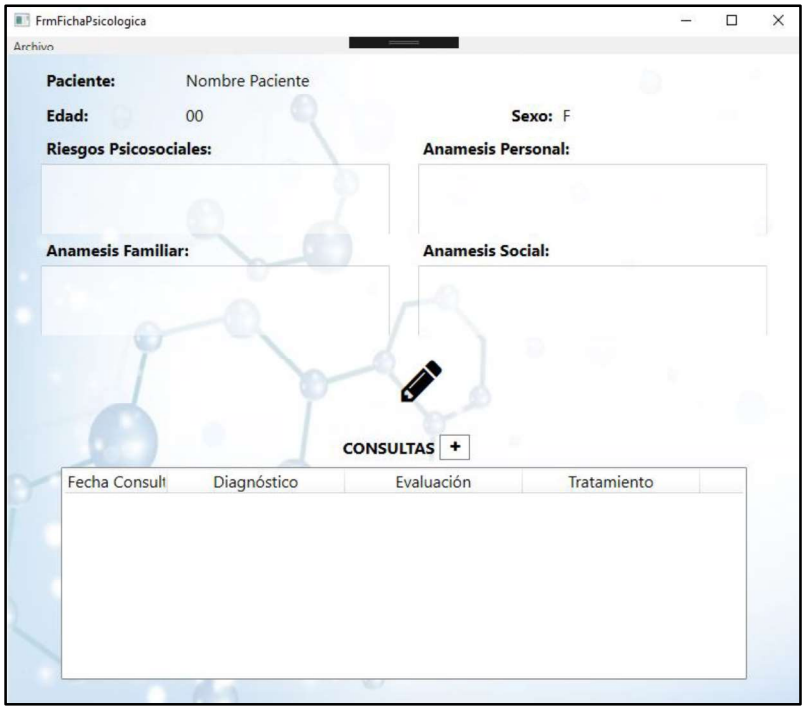


Figura 2.48. *Window* `FrmConsultaPsicológica`

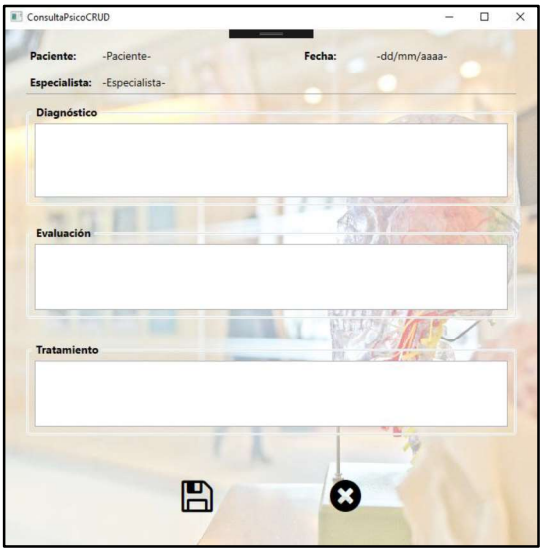


Figura 2.49. *Window* `ConsultaPsicoCrud`

2.3.3.2 Controles de usuario

Para la implementación del odontograma se utilizaron controles de usuario los cuales proporcionan un control vacío que se puede utilizar para crear otros controles, lo que permite reutilizar interfaces gráficas en todo el sistema. En el proyecto `ClienteEscritorio` se agregaron los controles de usuario como se observa en la Figura 2.50. Para el desarrollo del odontograma se crearon dos controles de usuario: `ucPieza` y `ucOdontograma` los cuales se encuentran en la Figura 2.51 y Figura 2.52 respectivamente.

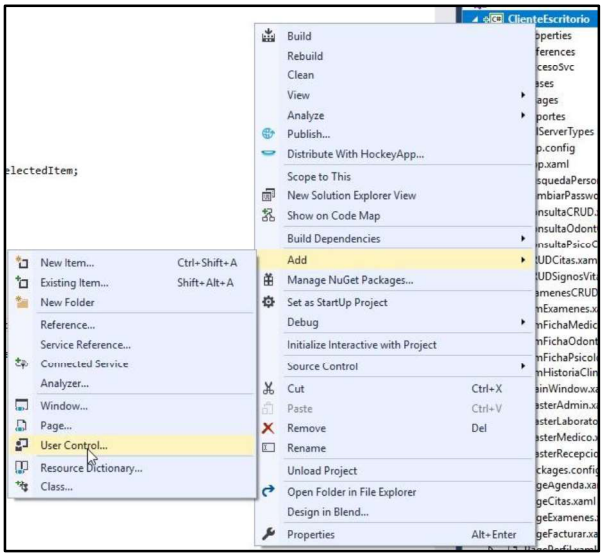


Figura 2.50. Adición de controles de usuarios al proyecto `ClienteEscritorio`

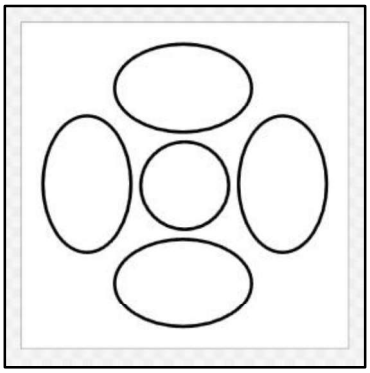


Figura 2.51. Control de usuario `ucPieza`

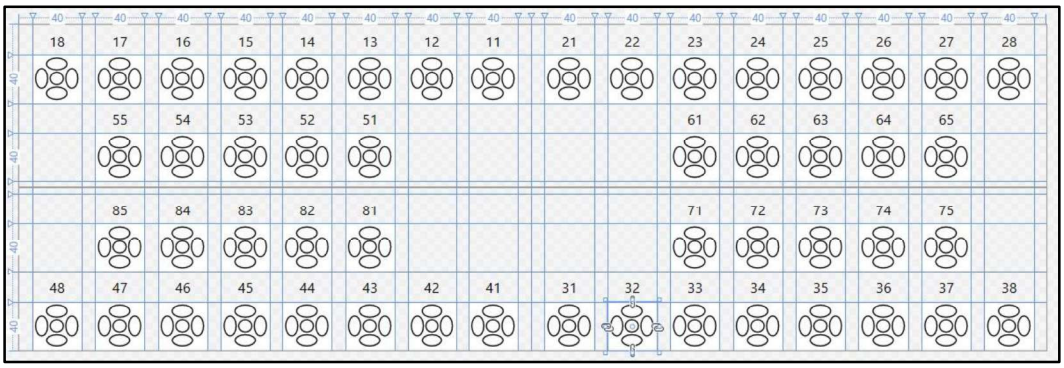


Figura 2.52. Control de usuario `ucOdontograma`

a) Codificación del odontograma

Para que el odontograma permita modificar sus valores en cada consulta se creó el método `editarOdontograma` dentro de la clase `SvcHistoriasClinicas.svc.cs` (ver Segmento de Código 2.22) en las cuales se encuentran todos los procedimientos odontológicos solicitados en los requerimientos (ver líneas 351 y 352), también se realiza la conexión a la base de datos para almacenar los nuevos valores (desde la línea 357).

Cada vez que se llame al odontograma se utiliza el método `llenarPieza` para poner en las piezas las características almacenadas, como se detalla en el Segmento de Código 2.23. Cada procedimiento odontológico tiene tres estados, el estado 0 indica que no se ha realizado ningún procedimiento, el estado 2 indica que está por realizar y el 1 que se ha realizado un determinado procedimiento. Como ejemplo se tiene en el Segmento de Código 2.24 el procedimiento que señala que una pieza dental estaría por extraerse, para lo cual se llamaría al método `OpcPorExtraer` y cambia el estado de la variable extracción a 2 en la línea 271.

```
351         public Odontograma editarOdontograma(int id_f_odontologica, int numero_pieza,
352             char extraccion, string caries, char sellante, char endodoncia, char corona, char fractura, char puente)
353         {
354             MiDBDataContext miDB = new MiDBDataContext();
355             try
356             {
357                 Odontograma odontograma = (from o in miDB.Odontograma
358                                            where o.id_f_odontologica == id_f_odontologica && o.numero_pieza == numero_pieza
359                                            select o).First();
360                 odontograma.extraccion = extraccion;
361                 odontograma.caries = caries;
362                 odontograma.sellante = sellante;
363                 odontograma.endodoncia = endodoncia;
364                 odontograma.corona = corona;
365                 odontograma.fractura = fractura;
366                 odontograma.puente = puente;
367                 miDB.SubmitChanges();
368                 return odontograma;
369             }
```

Segmento de Código 2.22. método `editarOdontograma`

```
54          public void llenarPieza(int id_f_odont, char extrac, string carie, char sell, char endo, char cor, char fract, char puent)
55          {
56              numero_pieza = int.Parse(this.Name.Substring(1, 2));
57              this.id_f_odontologica = id_f_odont;
58
59              if (extrac == '0')
60              {
61                  imgExtraccionR.Visibility = Visibility.Collapsed;
62                  imgExtraccionB.Visibility = Visibility.Collapsed;
63                  extraccion = 0;
64              }
65              else if (extrac == '1')
66              {
67                  imgExtraccionR.Visibility = Visibility.Collapsed;
68                  extraccion = 1;
69                  imgExtraccionB.Visibility = Visibility.Visible;
70              }
```

Segmento de Código 2.23. Parte del método `llenarPieza`

```
261         private void OpcPorExtraer_Click(object sender, RoutedEventArgs e)
262         {
263             try
264             {
265                 var odontograma = AccesoSvc.Servicios.editarOdontograma(id_f_odontologica, numero_pieza, '2', estado_s1.ToString() +
266                     estado_s2.ToString() + estado_s3.ToString() + estado_s4.ToString() + estado_s5.ToString(), sellante.ToString()[0],
267                     endodoncia.ToString()[0], corona.ToString()[0], fractura.ToString()[0],puente.ToString()[0]);
268                 if (odontograma.Id_odontograma > 0)
269                 {
270                     imgExtraccionB.Visibility = Visibility.Collapsed;
271                     extraccion = 2;
272                     imgExtraccionR.Visibility = Visibility.Visible;
273                 }
274             }
275             catch (Exception ex)
276             {
277                 MessageBox.Show("Verifique la conexión a Internet\n\nERROR: " + ex.Message);
278             }
279         }
280
```

Segmento de Código 2.24. Método `OpcPorExtraer`

2.3.4 ITERACIÓN 4

En la cuarta iteración se desarrollaron los módulos para el manejo de exámenes médicos.

2.3.4.1 Interfaces visuales

Las interfaces visuales desarrolladas en esta iteración son: la *page* `FrmExamenes` la cual es la pantalla inicial que observa el laboratorista clínico, en esta se encuentran la lista de todos los exámenes realizados, las fechas y el estado como lo indica la Figura 2.53. En la

Figura 2.54 se encuentra la ventana `ExamenesCRUD` contiene toda la información de los exámenes y tiene una opción para cargar el documento con los resultados. Para seleccionar los exámenes que se deben realizar los pacientes los especialistas tienen acceso a la ventana `SelecExamenes` la cual contiene la lista de todos los exámenes que realizar el centro médico como se visualiza en la Figura 2.55.

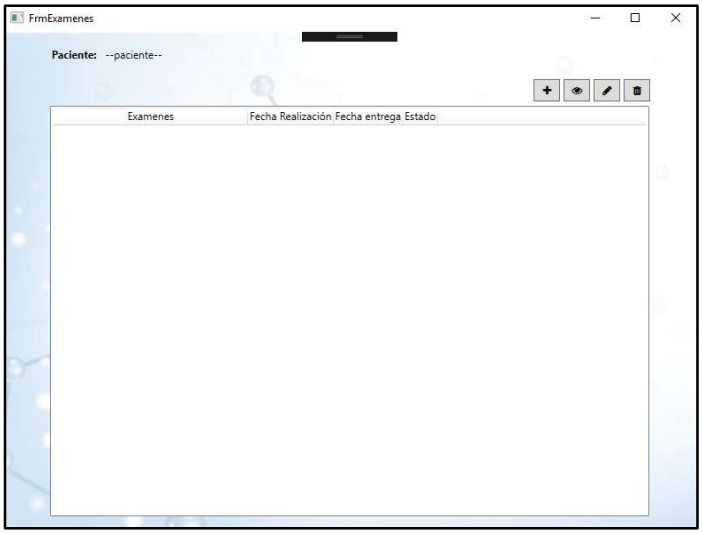


Figura 2.53. *page* `FrmExamenes`

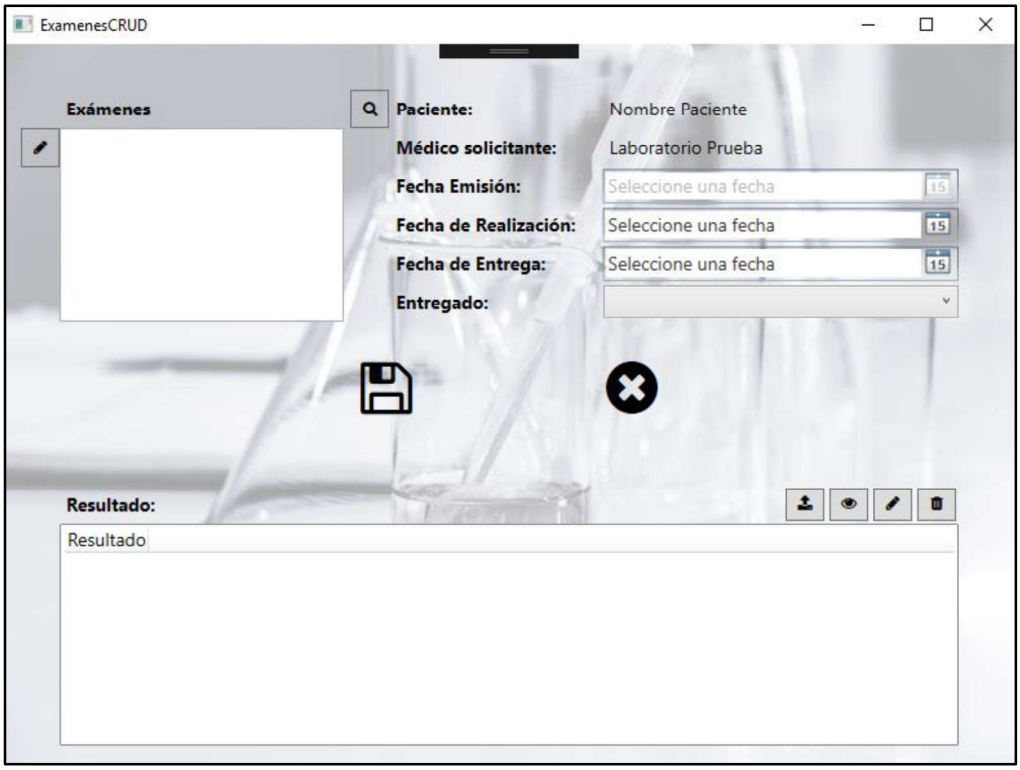


Figura 2.54. W*indow* `ExamenesCRUD`

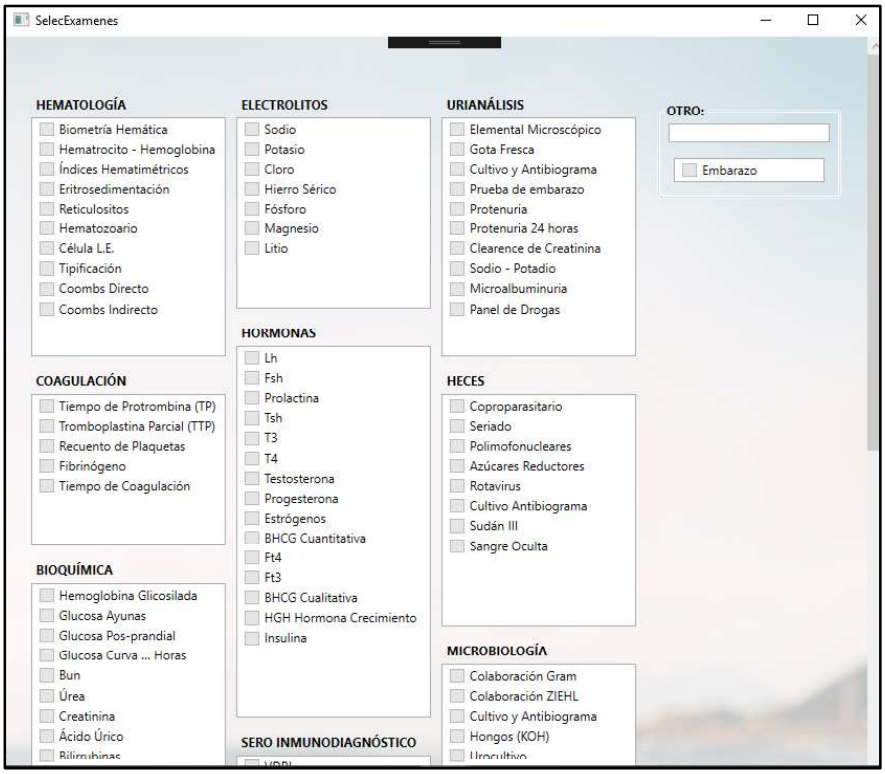


Figura 2.55. *Window* `SelecExamenes`

2.3.4.2 Control FileUpload

FileUpload es un control de ASP.NET que permite seleccionar desde el equipo local un archivo y enviarla al servidor para su manipulación con C#. Se utiliza este control para subir los resultados de laboratorio. En la Figura 2.56 se visualiza la interfaz `ResultadoCRUD`, esta tiene el Control *FileUpload* para permitir al laboratorista subir un archivo con extensión PDF, este debe encontrarse en el computador que está utilizando.

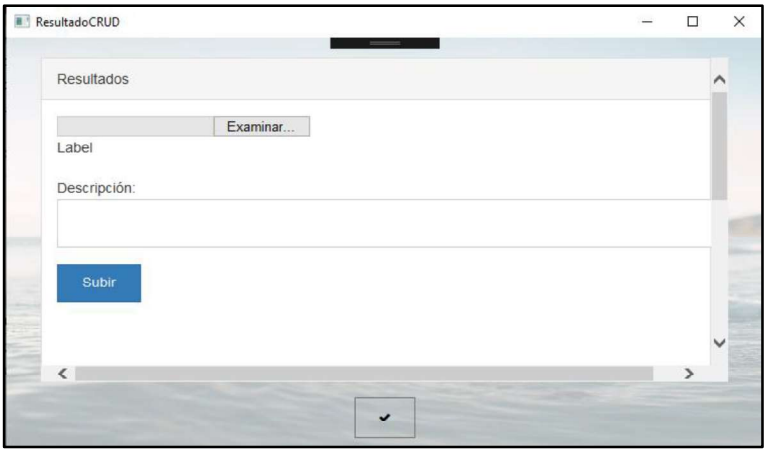


Figura 2.56. *Window* `ResultadoCRUD`

2.3.5 ITERACIÓN 5

En la quinta iteración en primer lugar se publicó el servidor en la plataforma informática de Azure y luego se desarrolló la aplicación Android.

2.3.5.1 Publicación del servidor en Azure

Se procedió a publicar el servidor en la plataforma informática de Azure y debido a que *Visual Studio* es desarrollado por *Microsoft* el proceso de publicación es simple. En primer lugar, se seleccionó el proyecto `Servidor` de la solución y se escogió la opción publicar como se indica en la Figura 2.57, la siguiente ventana que aparece es en la que se debe seleccionar que tipo de publicación se va a realizar, para este proyecto se eligió la opción *App Service*. Para finalizar se seleccionó la cuenta de *Microsoft* a utilizar y las características adicionales que se indican en la Figura 2.58

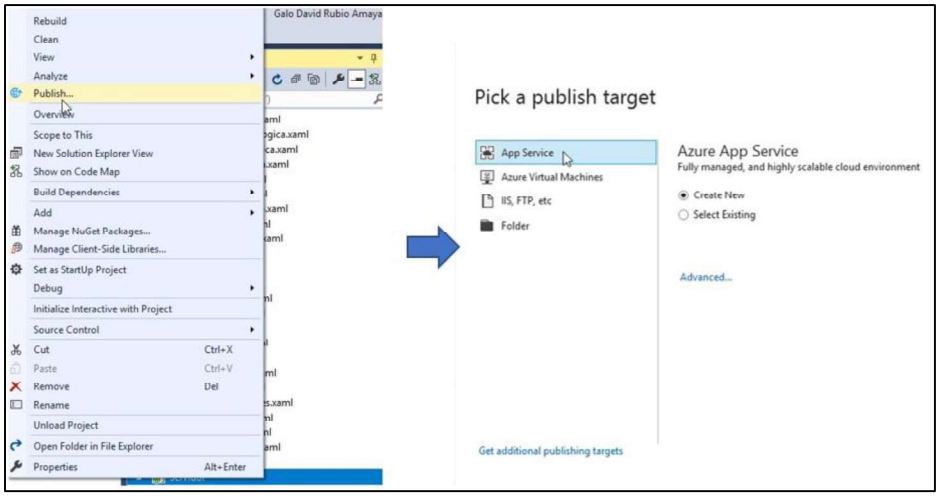


Figura 2.57. Publicación del servidor en Azure (parte 1 de 2)

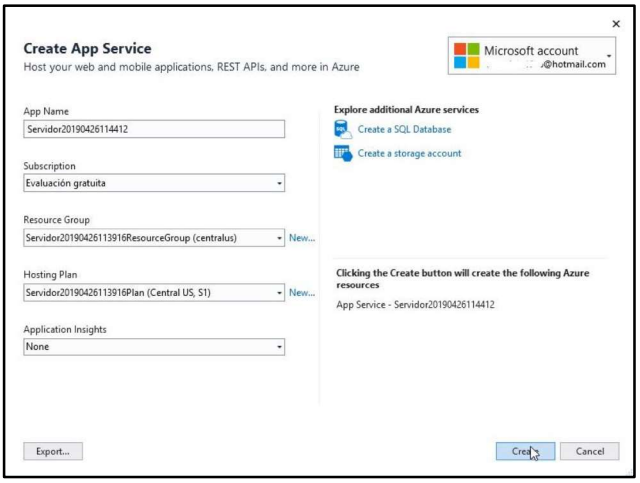


Figura 2.58. Publicación del servidor en Azure (parte 2 de 2)

2.3.5.2 Desarrollo de la aplicación Android

En primer lugar, se creó una aplicación vacía con el nombre `CentroMedico`, cuyo lenguaje de desarrollo es Java y el mínimo *API Level* es Android 4.4 (KitKat) lo que permite un nivel de compatibilidad del 95.3% con los dispositivos Android móviles como se observa en la Figura 2.59. A continuación, en el archivo `AndroidManifest.xml` se definen los permisos que son usados por la aplicación, en el Segmento de Código 2.25 se observa en la línea 5 el permiso para que la aplicación tenga acceso a Internet.

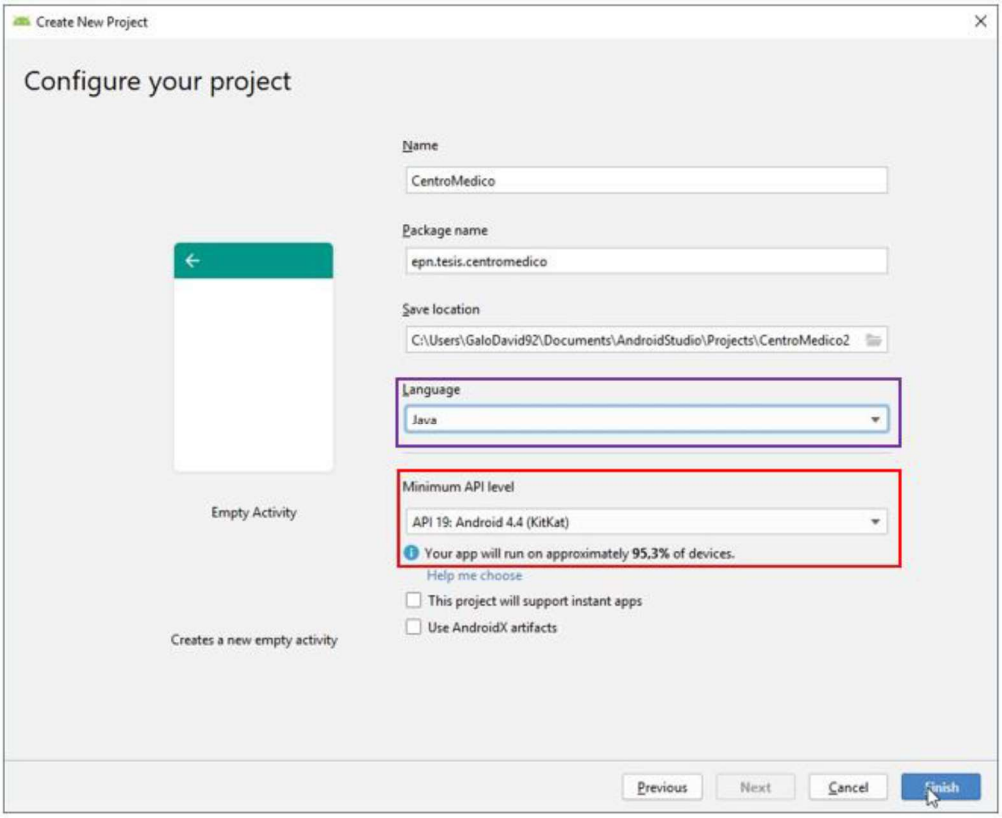
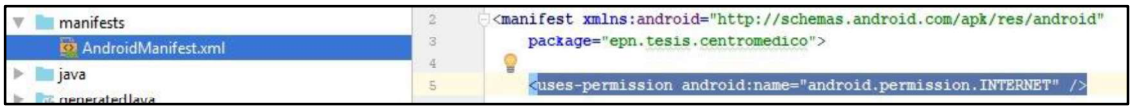


Figura 2.59. Creación de la aplicación Android



Segmento de Código 2.25. Asignación del permiso para que la aplicación tenga acceso a Internet.

En la carpeta *layout* se almacenan todas las interfaces gráficas desarrolladas para la aplicación como lo indica la Figura 2.60. Para el desarrollo de las interfaces se utilizó el lenguaje de marcado xml. La primera interfaz gráfica de la aplicación Android que se desarrolló fue la correspondiente al *Login* la cual se encuentra en la Figura 2.61, todas las demás interfaces se encuentran en el Anexo F, el Segmento de Código 2.26 indica la

codificación en xml detrás de esta interfaz se encuentra en el `activity_main.xml` como se observa se utilizaron controles como *EditText, Switch, ImageButton,* entre otros, en los cuales se va añadiendo características para el diseño visual.

En la Figura 2.62 se encuentran todas las clases que se desarrollaron para la aplicación móvil. En el Segmento de Código 2.27 se encuentra la clase `MainActivity`, que corresponde al punto de entrada de la aplicación, en la que desde la línea 15 a la 18 se declara todas las variables que se va a utilizar. A continuación, desde la línea 24 a la 29 asocia el diseño visual al código Java y se inicializan los componentes principales de la actividad.

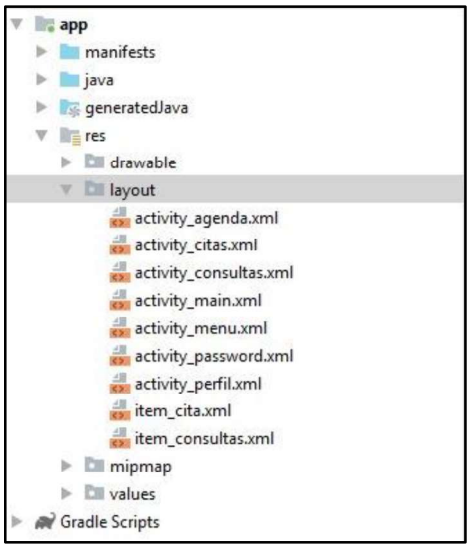


Figura 2.60. Archivos de la carpeta *layout*

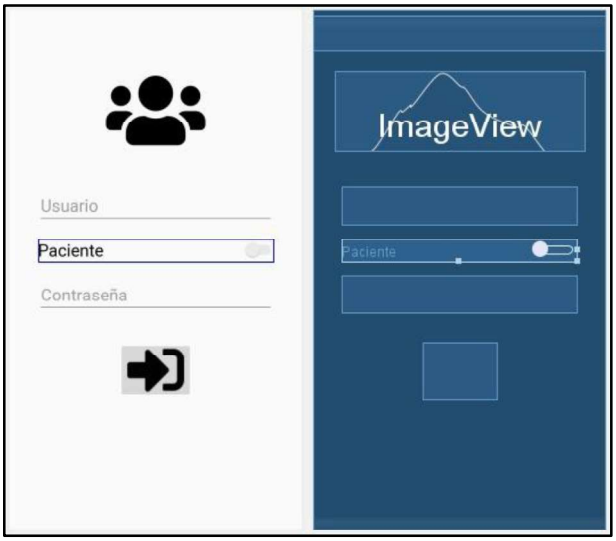


Figura 2.61. Interfaz gráfica del *Login* de la aplicación móvil

```
        <EditText
            android:id="@+id/txtUsuario"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="20dp"
            android:layout_marginRight="40dp"
            android:ems="10"
            android:hint="Usuario"
            android:inputType="textPersonName"
            android:textSize="24sp" />

        <Switch
            android:id="@+id/swMedPac"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="20dp"
            android:layout_marginRight="40dp"
            android:text="Paciente"
            android:textSize="24sp" />

        <EditText
            android:id="@+id/txtPassword"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="40dp"
            android:layout_marginTop="20dp"
            android:layout_marginRight="40dp"
            android:ems="10"
            android:hint="Contraseña"
            android:inputType="textPassword"
            android:textSize="24sp" />

        <ImageButton
            android:id="@+id/btnAcceder"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:layout_margin="40dp"
            app:srcCompat="@drawable/signin" />

    </LinearLayout>
</android.support.constraint.ConstraintLayout>
```

Segmento de Código 2.26. `activity_main.xaml`

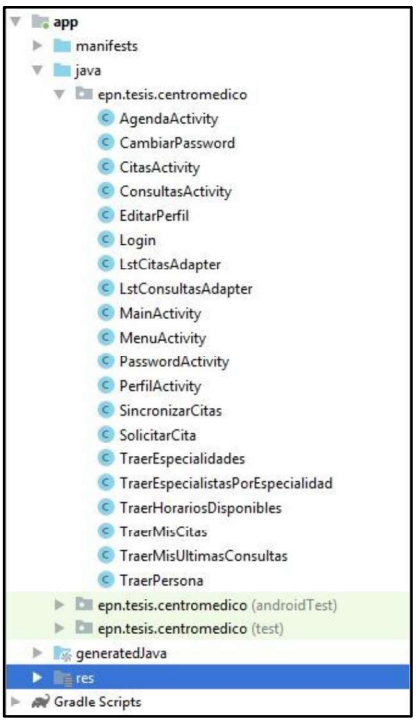


Figura 2.62. Clases de la aplicación Android

```
13  public class MainActivity extends AppCompatActivity {
14
15      EditText txtUsuario;
16      EditText txtPassword;
17      ImageButton btnAcceder;
18      Switch swMedPac;
19
20      String tipo = "P";
21
22      @Override
23      protected void onCreate(Bundle savedInstanceState) {
24          super.onCreate(savedInstanceState);
25          setContentView(R.layout.activity_main);
26          txtUsuario = findViewById(R.id.txtUsuario);
27          txtPassword = findViewById(R.id.txtPassword);
28          btnAcceder = findViewById(R.id.btnAcceder);
29          swMedPac = findViewById(R.id.swMedPac);
30
31          swMedPac.setOnCheckedChangeListener((buttonView, isChecked) -> {
34                  if (isChecked) {
35                      swMedPac.setText("Médico");
36                      tipo = "M";
37                  } else {
38                      swMedPac.setText("Paciente");
39                      tipo = "P";
40                  }
41          });
43          btnAcceder.setOnClickListener(new View.OnClickListener() {
44              @Override
45              public void onClick(View v) {
46                  acceder();
47              }
48          });
49      }
50
51      private void acceder() {
52          String username = txtUsuario.getText().toString();
53          String password = txtPassword.getText().toString();
54          Login login = new Login( context: this, username, tipo, password);
55          login.execute();
56      }
57  }
```

Segmento de Código 2.27. Clase MainActivity

`AsyncTask` es una clase incluida en Android Studio que permite crear tareas asíncronas, lo que implica realizar operaciones en segundo plano y publicar resultados en el subproceso de la interfaz de usuario sin tener que manipular subprocesos. Esta clase posee tres métodos: `onPreExecute, doInBackground, onPostExecute`.

En el Segmento de Código 2.28 se creó la clase `Login` la cual se hereda de la clase `AsyncTask`, de la línea 22 a la 25 se declaran los atributos de la clase *`Login`*, en la línea 25 se ingresa la *url* del servidor con el método que se va a utilizar, a partir de la línea 37 se muestra el constructor de la clase. En el Segmento de Código 2.29 se sobreescribe el método `onPreExecute()` el cual realiza los trabajos previos a la ejecución de una tarea determinada, para este caso en la línea 37 se muestra una barra de progreso.

En el Segmento de Código 2.30 se encuentra el método `doInBackground` donde están las acciones que va a realizar el hilo secundario, de la línea 44 a la 50 crea una conexión http con sus respectivos parámetros, a partir de la línea 52 hasta la 55 crea el objeto json que será enviado, de la línea 58 a la 65 envía la petición al servidor y se cierra la conexión. Desde la línea 67 hasta la línea 84 se observa como el servidor recibe y codifica la respuesta.

```
20  public class Login extends AsyncTask<Void, Void, String> {
21
22      private Context context;//contexto
23      String nombre_usuario, password, tipo_usuario;
24      ProgressDialog progressDialog;//dialogo cargando
25      String url_login = "http://centromedico.azurewebsites.net/Servicios/SvcLogin.svc/login";
26
27      public Login(Context context, String usuario, String tipo, String passwd) {
28          this.context = context;
29          this.nombre_usuario = usuario;
30          this.tipo_usuario = tipo;
31          this.password = passwd;
32      }
```

Segmento de Código 2.28. Atributos y constructor de la clase `Login`

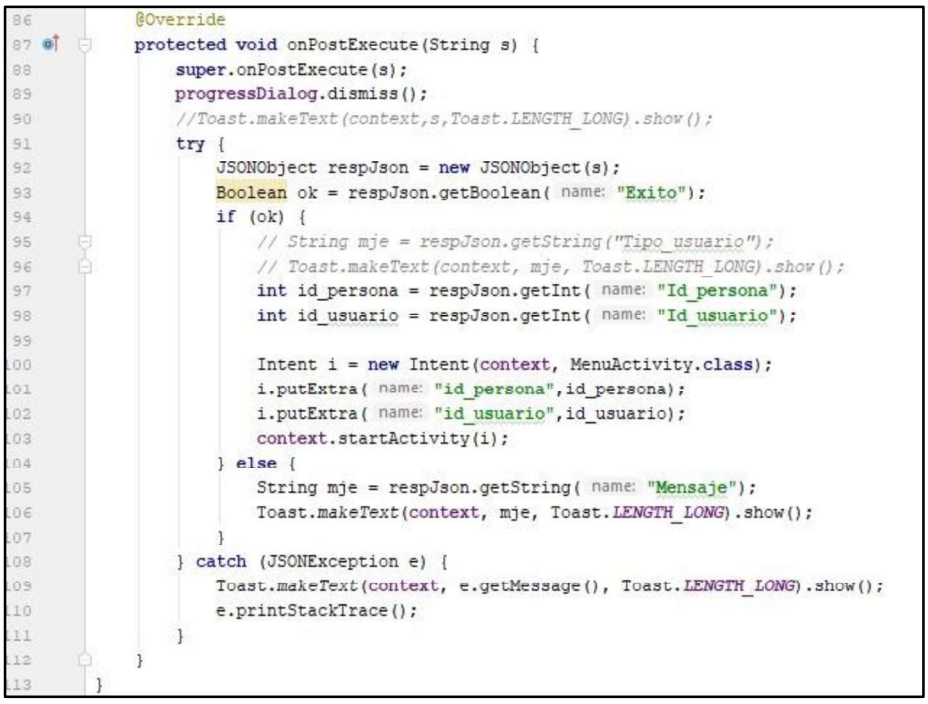
```
34      @Override
35      protected void onPreExecute() {
36          super.onPreExecute();
37          progressDialog = ProgressDialog.show(context,  title: "Procesando Solicitud",  message: "por favor, espere");
38      }
```

Segmento de Código 2.29. Método `onPreExecute()` de la clase `Login`

```
40      @Override
41      protected String doInBackground(Void... voids) {
42          String resp = "";
43          try {
44              URL url = new URL(url_login);
45              HttpURLConnection conn = (HttpURLConnection) url.openConnection();
46              conn.setRequestMethod("POST");
47              conn.setRequestProperty("Content-Type", "application/json");
48              conn.setRequestProperty("Accept", "application/json");
49              conn.setDoOutput(true);
50              conn.setDoInput(true);
51
52              JSONObject jsonParam = new JSONObject();
53              jsonParam.put( name: "nombre_usuario", nombre_usuario);
54              jsonParam.put( name: "password", password);
55              jsonParam.put( name: "tipo_usuario", tipo_usuario);
56
57              // Log.i("JSON", jsonParam.toString());
58              DataOutputStream os = new DataOutputStream(conn.getOutputStream());
59              //os.writeBytes(URLEncoder.encode(jsonParam.toString(), "UTF-8"));
60              String request = jsonParam.toString();
61              os.writeBytes(request);
62
63              os.flush();
64              os.close();
65
66              // resp=String.valueOf(conn.getResponseCode()+" - "+conn.getResponseMessage());
67              InputStream in = conn.getInputStream();
68
69              InputStreamReader reader = new InputStreamReader(in);
70
71              int data = reader.read();
72
73              while (data != -1) {
74                  char current = (char) data;
75                  resp += current;
76                  data = reader.read();
77              }
78              conn.disconnect();
79          } catch (Exception e) {
80              e.printStackTrace();
81              resp = e.getMessage();
82          }
83          return resp;
84      }
```

Segmento de Código 2.30. Método `doInBackground()` de la clase Login

En el Segmento de Código 2.31 se detalla el método `onPostExecute()` este método se ejecuta en el hilo principal al finalizar el método `doInBackground()`. En la línea 89 se oculta la barra de progreso, en la línea 93 se obtiene el valor de la variable booleana `Exito` de la respuesta, si su valor es verdadero se obtienen y se envían los datos recibidos del servidor hacia la actividad `MenuActivity` como se observa desde la línea 94 a la 104, caso contrario se obtiene y se muestra el mensaje de error enviado por el servidor como se muestra a partir de la línea 105.



```
86      @Override
87      protected void onPostExecute(String s) {
88          super.onPostExecute(s);
89          progressDialog.dismiss();
90          //Toast.makeText(context,s,Toast.LENGTH_LONG).show();
91          try {
92              JSONObject respJson = new JSONObject(s);
93              Boolean ok = respJson.getBoolean( name: "Exito");
94              if (ok) {
95                  // String mje = respJson.getString("Tipo_usuario");
96                  // Toast.makeText(context, mje, Toast.LENGTH_LONG).show();
97                  int id_persona = respJson.getInt( name: "Id_persona");
98                  int id_usuario = respJson.getInt( name: "Id_usuario");
99
100                 Intent i = new Intent(context, MenuActivity.class);
101                 i.putExtra( name: "id_persona",id_persona);
102                 i.putExtra( name: "id_usuario",id_usuario);
103                 context.startActivity(i);
104             } else {
105                 String mje = respJson.getString( name: "Mensaje");
106                 Toast.makeText(context, mje, Toast.LENGTH_LONG).show();
107             }
108         } catch (JSONException e) {
109             Toast.makeText(context, e.getMessage(), Toast.LENGTH_LONG).show();
110             e.printStackTrace();
111         }
112     }
113 }
```

Segmento de Código 2.31. Método `onPostExecute()` de la clase `Login`

3. RESULTADOS Y DISCUSIÓN

En este capítulo basado en la metodología XP se describen las pruebas realizadas al sistema, de manera incremental con lo cual se comprobó el cumplimiento de los requisitos funcionales y no funcionales en cada iteración. Desde la iteración 1 a la 4 las pruebas se realizaron con el servidor en el ambiente local que ofrece Visual Studio. Para la iteración 5 se desplegó el servidor en Azure y se realizaron las pruebas en un ambiente de prueba integrado.

3.1 PRUEBAS DE FUNCIONAMIENTO

Las pruebas de funcionamiento fueron realizadas para cada iteración y poseen el siguiente formato:

- **Caso de prueba:** Nombre de la prueba realizada
- **Procedimiento:** Acciones ejecutadas por parte de los usuarios del sistema
- **Resultado esperado:** Describe el comportamiento esperado del sistema al realizar el procedimiento mencionado anteriormente.
- **Resultado obtenido:** Presenta evidencias de los resultados de las pruebas realizadas

3.1.1 PRUEBAS DE FUNCIONAMIENTO ITERACIÓN 1

- **Caso de prueba:** Creación de un nuevo usuario

Procedimiento:

1. El administrador del sistema debe llenar los datos del nuevo usuario
2. Se envía un correo a la dirección registrada por el nuevo usuario con un *link* para la activación del usuario y una contraseña temporal.
3. El usuario debe abrir el *link* enviado e ingresar la contraseña temporal y su nueva contraseña.
4. El usuario al presionar el botón Activar Cuenta debe aparecer el mensaje de éxito.

Resultado esperado: El administrador ingresa los datos obligatorios del nuevo usuario, en caso de que los campos que se vayan a llenar sean incorrectos el sistema no debe permitir guardar la información y muestra mensajes de error. El usuario recibe un correo electrónico con el *link* de activación de su cuenta y su clave temporal. Al acceder al link el nuevo usuario ingresa su contraseña temporal y su nueva contraseña para activar su cuenta en el sistema de escritorio.

Resultado obtenido: En la Figura 3.1 se observa que el administrador debe ingresar todos los datos obligatorios y de manera correcta, caso contrario aparecen mensajes de error en cada campo, como ejemplo se encuentra el mensaje que aparece al ingresar una fecha inválida.

A continuación, se ingresó todos los datos correctamente y el nuevo usuario ha sido creado con éxito como se visualiza en la Figura 3.2. En la Figura 3.3 está el correo electrónico que recibió el nuevo usuario indicándole su nombre de usuario, contraseña temporal y el link de activación. Posteriormente en la Figura 3.4 se encuentra el proceso que siguió el nuevo usuario para activar su cuenta.

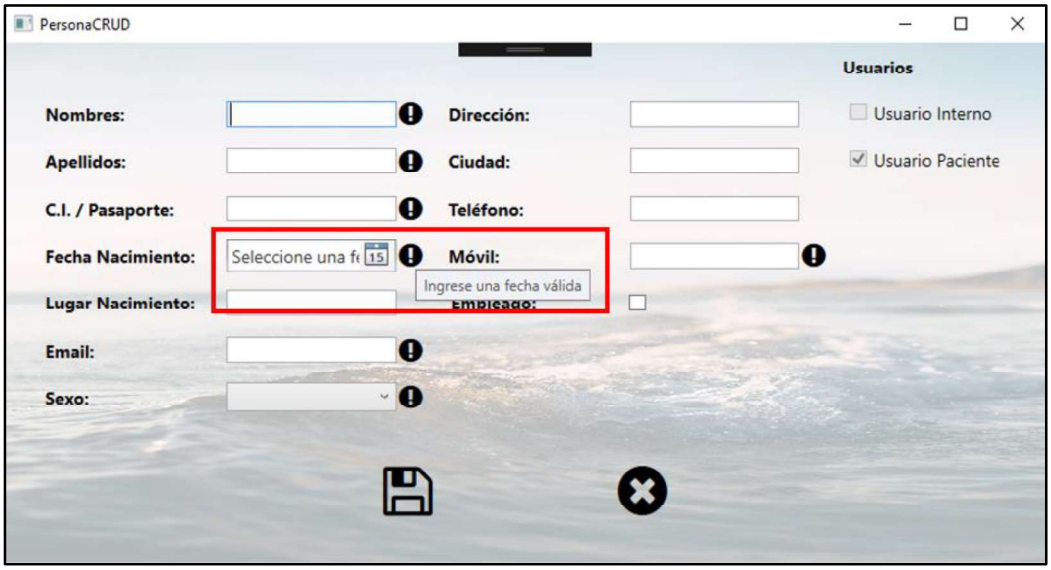


Figura 3.1. Ingreso de datos incorrectos de un nuevo usuario

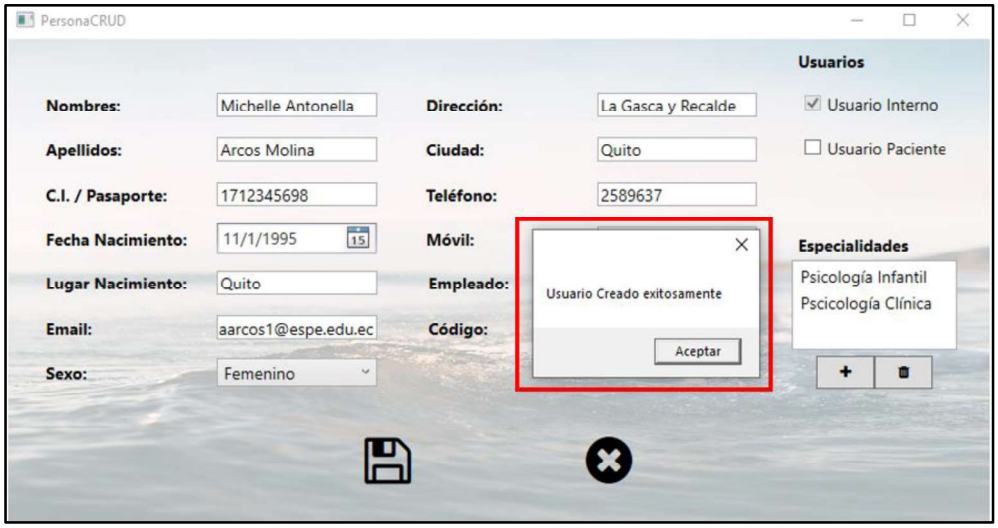


Figura 3.2. Ingreso de datos correctos de un nuevo usuario

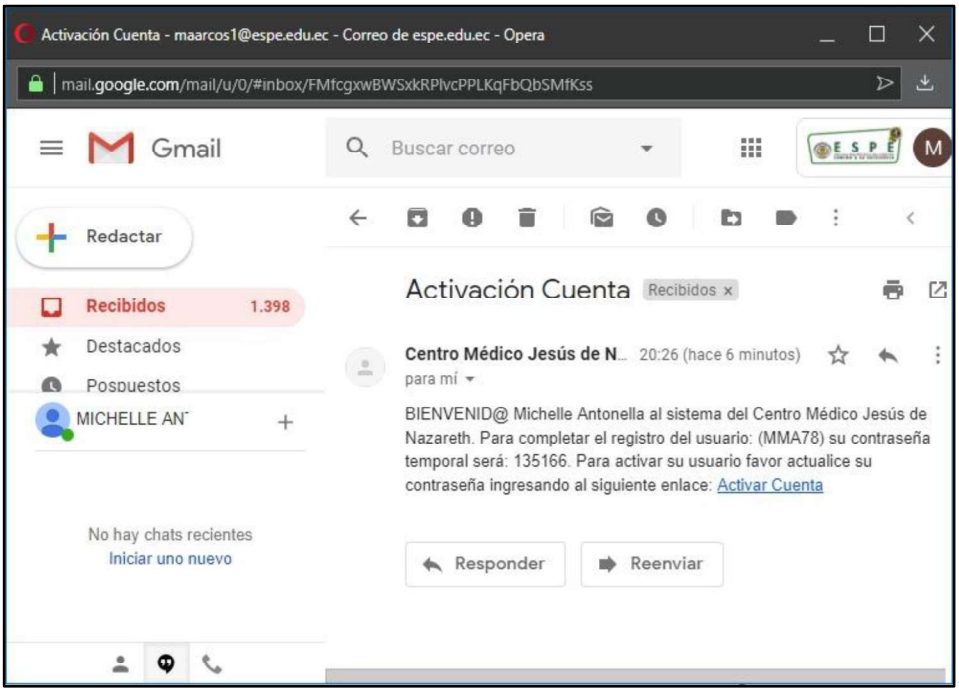


Figura 3.3. Correo electrónico enviado a un nuevo usuario para activación de su cuenta

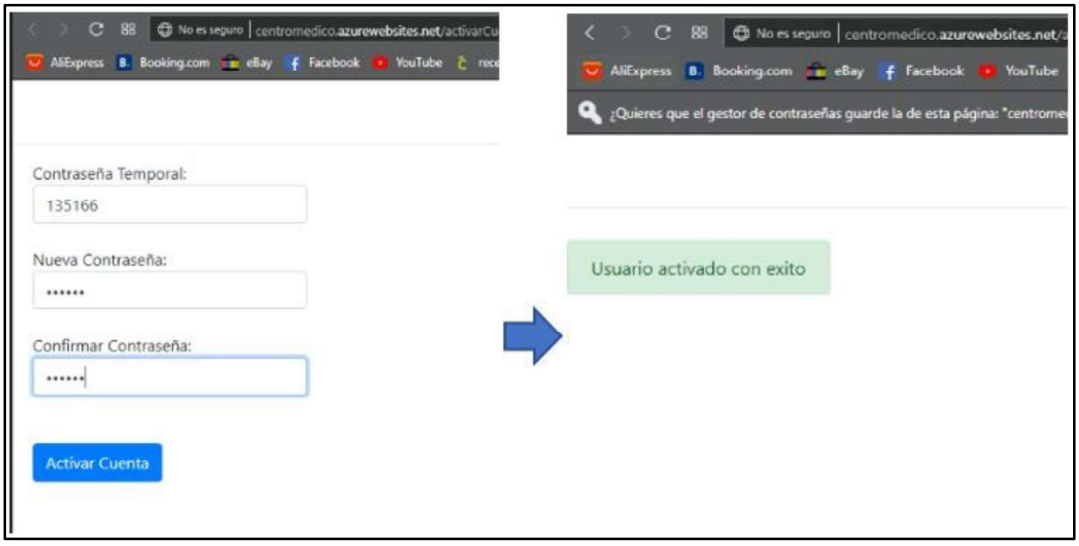


Figura 3.4. Activación de la cuenta de un nuevo usuario.

- **Caso de prueba:** Ingreso de los usuarios al sistema de escritorio

Procedimiento:

1. El usuario ingresa su nombre de usuario y contraseña
2. El usuario selecciona el tipo de usuario que posee: administrador, recepcionista, médico y laboratorio clínico.

3. El usuario selecciona el botón acceder
4. En caso de ingresar alguno de los datos incorrectamente debe aparecer un mensaje de error.

Resultado esperado: El usuario ingresa su nombre y contraseña, además selecciona su tipo de usuario. En caso de que uno de los datos sea ingresado incorrectamente deben aparecer mensajes de error correspondientes.

Resultado obtenido: En la Figura 3.5 se observa el ingreso de datos, adicional se muestra los tipos de usuario que puede seleccionar el usuario (administrador, recepción, médico y laboratorio clínico). En la Figura 3.6 se muestran los mensajes de error que aparecen al ingresar mal cualquiera de los tres datos solicitados.

Figura 3.5. Ingreso de datos de un usuario en el sistema de escritorio

Figura 3.6. Mensajes de error *Login*

- **Caso de prueba:** Recuperación de contraseña

Procedimiento:

1. Ingresar el nombre de usuario en la interfaz del *Login*

2. Dar *click* en el *link* ¿Has olvidado tu contraseña?
3. Preguntar si está seguro de continuar con el proceso
4. Enviar un correo electrónico enviando un *link* de recuperación de contraseña

Resultado esperado: Al dar *click* sobre el *link* de recuperación de contraseña se envía un correo electrónico con instrucciones para recuperar la contraseña.

Resultado obtenido: Como se observa en la Figura 3.7 al dar *click* sobre el *link* ¿Has olvidado tu contraseña? Se envía un correo electrónico para continuar con el proceso como se indica en la Figura 3.8.

Figura 3.7. Recuperación de contraseña en el sistema de escritorio

Figura 3.8. Correo con las indicaciones para recuperar la contraseña

- **Caso de prueba:** Visualización y búsqueda de usuarios

Procedimiento:

1. Ingresar como administrador o recepcionista al sistema de escritorio
2. Se puede visualizar todos los usuarios del sistema
3. Se puede realizar la búsqueda de los usuarios del sistema por tipo y con una barra de búsqueda.

Resultado esperado: Se visualiza el nombre, apellido, identificación y correo electrónico de todos los usuarios. Se filtra usuarios por el tipo y a través de una barra de búsqueda.

Resultado obtenido: En la Figura 3.9 se observa la lista de todos los usuarios del sistema con sus nombres, apellidos, identificación y correo electrónico. Luego en la Figura 3.10 se observa cómo se puede filtrar a los usuarios por tipo y a través de una barra de búsqueda.

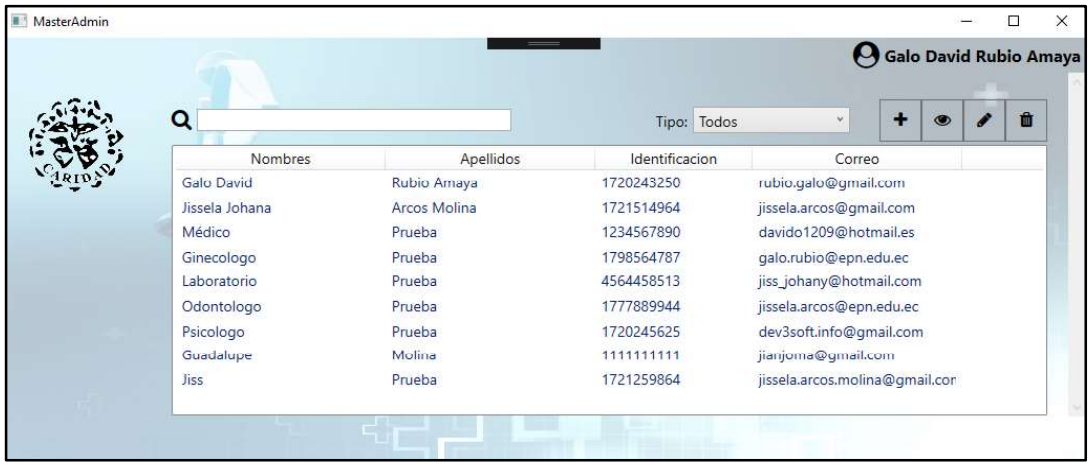


Figura 3.9. Visualización de todos los usuarios del sistema

Figura 3.10. Filtración de los usuarios del sistema

- **Caso de prueba:** Acciones CRUD en usuarios del sistema

Procedimiento:

1. Ingresar como administrador o recepcionista al sistema de escritorio
2. Crear nuevos usuarios
3. Leer la información de los usuarios del sistema

4. Actualizar la información de los usuarios del sistema
5. Desactivar los usuarios del sistema.

Resultado esperado: Realizar acciones CRUD sobre los usuarios del sistema

Resultado obtenido: Se realizaron acciones CRUD sobre los usuarios del sistema. En una prueba anterior se creó nuevos usuarios del sistema. En la Figura 3.11 se observa la lectura y actualización de la información de un usuario del sistema. Posteriormente, en la Figura 3.12 se muestra el proceso para desactivar un usuario y como en la base de datos cambia a *False* la columna `usuario_activo`.

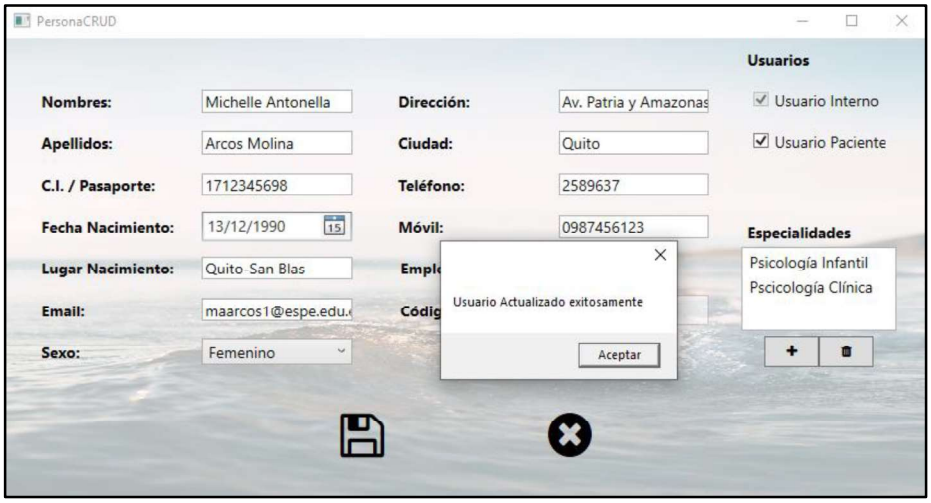


Figura 3.11. Lectura y actualización de los datos de un usuario del sistema de escritorio

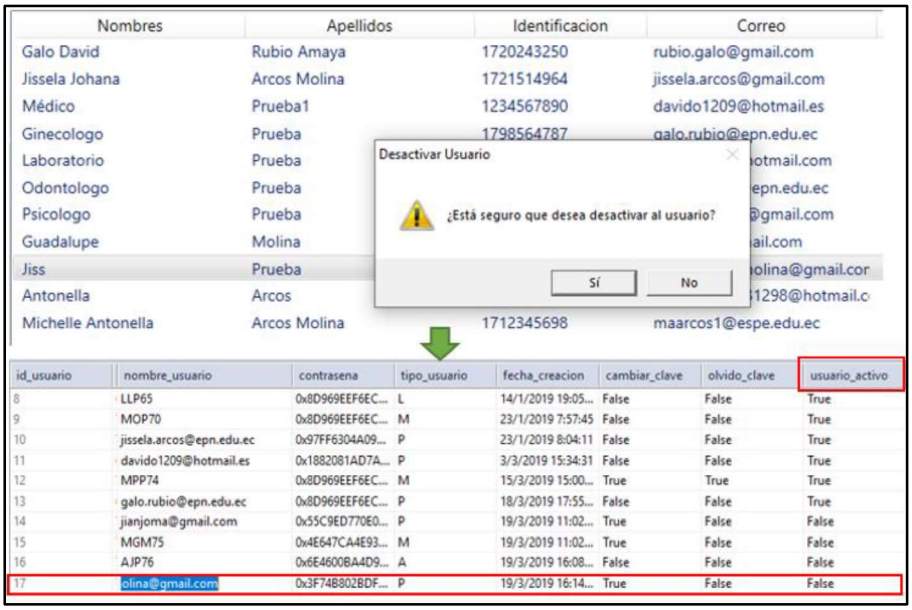


Figura 3.12. Desactivación de los usuarios del sistema de escritorio

3.1.2 PRUEBAS DE FUNCIONAMIENTO ITERACIÓN 2

- **Caso de prueba:** Modificar perfil personal de un usuario del sistema de escritorio

Procedimiento:

1. Los especialistas ingresan a la pestaña Mi Perfil de su módulo
2. Aparece la opción de modificar los datos personales que si pueden modificar
3. Al modificar los datos personales se deben guardar exitosamente

Resultado esperado: Un especialista modifica sus datos personales

Resultado obtenido: Como se observa en la Figura 3.13 el Psicólogo Prueba modificó su dirección, ciudad y teléfono.

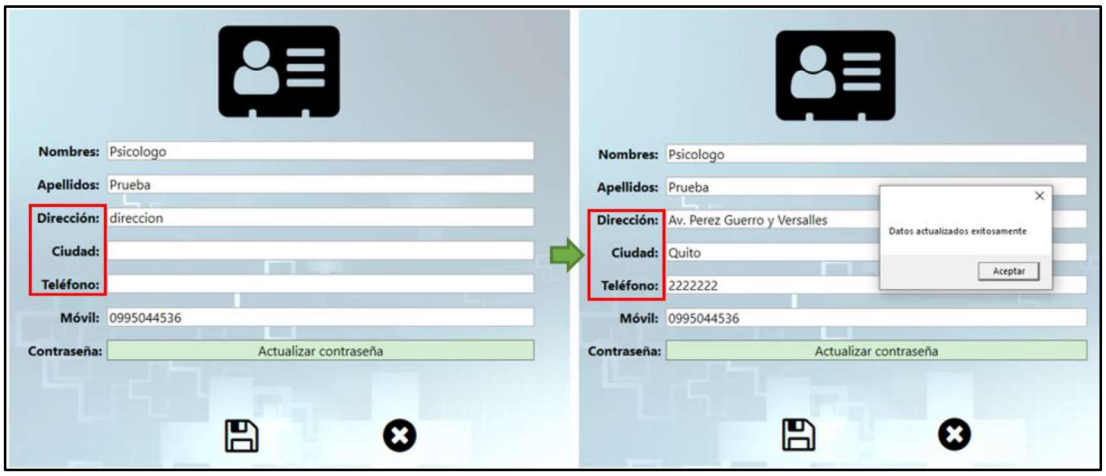


Figura 3.13. Especialista modifica datos personales

- **Caso de prueba:** Modificación de la contraseña de un usuario del sistema de escritorio

Procedimiento:

1. El especialista ingresa a Mi Perfil y presionar el botón Actualizar Contraseña.
2. El usuario ingresa la contraseña actual
3. El usuario escribe la contraseña nueva
4. El usuario escribe la contraseña nueva otra vez para confirmar
5. La nueva contraseña se actualizar correctamente

Resultado esperado: El especialista modifica su contraseña actual, en caso de que la contraseña actual no coincida aparece un mensaje de error y en caso de que la nueva contraseña no coincida tampoco se debe actualizar.

Resultado obtenido: En la Figura 3.14 se muestra los errores cuando no se realiza correctamente el proceso de cambio de contraseña, la primera imagen es cuando las contraseñas nuevas ingresadas no coinciden y la segunda cuando la contraseña actual no es correcta. En la Figura 3.15 la contraseña ha sido modificada correctamente.

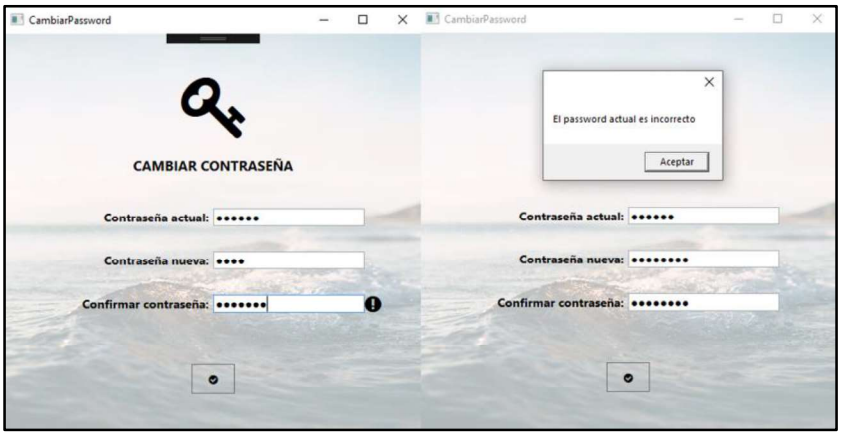


Figura 3.14. Mensajes de error al actualizar la contraseña en el sistema de escritorio

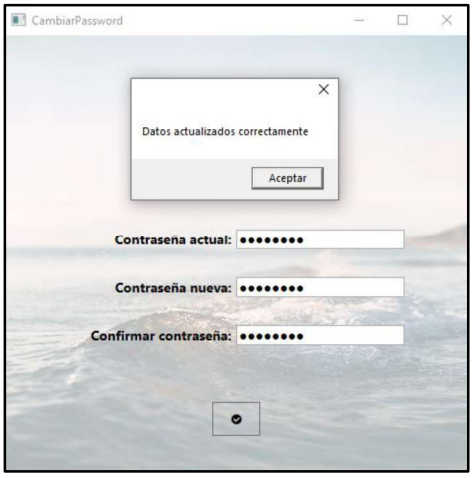


Figura 3.15. Actualización de la contraseña en el sistema de escritorio

- **Caso de prueba:** Visualización de la lista de especialistas

Procedimiento:

1. Ingresar como Recepcionista al sistema de escritorio
2. La recepcionista puede visualizar todos los especialistas del centro médico
3. La recepcionista puede filtrar los especialistas por su especialidad

Resultado esperado: Visualizar la lista de especialistas.

Resultado obtenido: En la Figura 3.16 se observa la lista de los especialistas del centro médico y se puede filtrar por especialidad en este caso por Psicología Clínica.

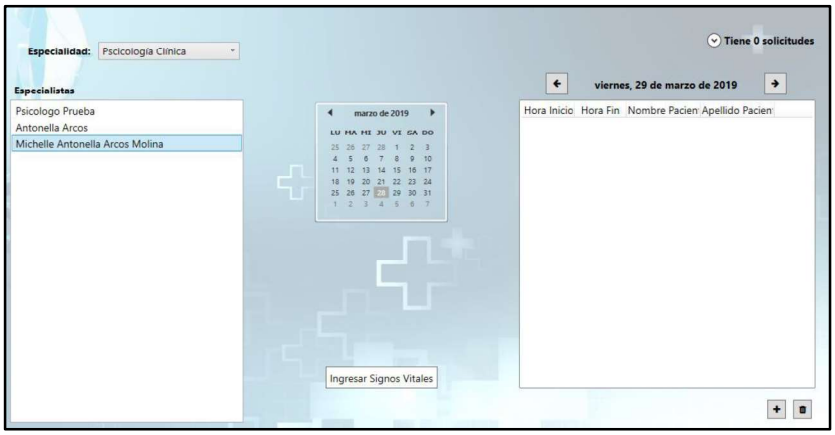


Figura 3.16. Visualización de los especialistas

- **Caso de prueba:** Agendar citas

Procedimiento:

1. Se selecciona el especialista y presionar el signo “+”.
2. Se abre una ventana para ingresar los datos de la cita.
3. Se agenda la cita correctamente y se envía un correo electrónico al paciente con los detalles de la cita.

Resultado esperado: La cita se agenda correctamente y se envía un correo electrónico al paciente. La cita agendada aparece en la agenda del especialista.

Resultado obtenido: Como se observa en la Figura 3.17 la cita de la paciente Jissela Johana Arcos Molina en la Especialidad Psicología Clínica ha sido agendada correctamente.

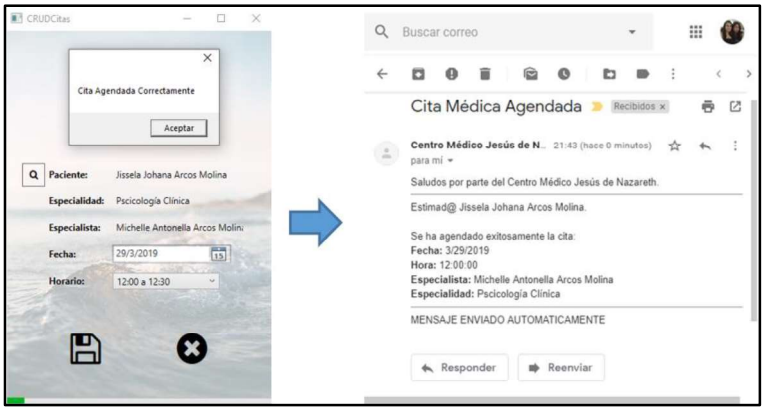


Figura 3.17. Creación de una cita médica

En la Figura 3.18 se muestra como la cita creada aparece en la agenda del especialista.

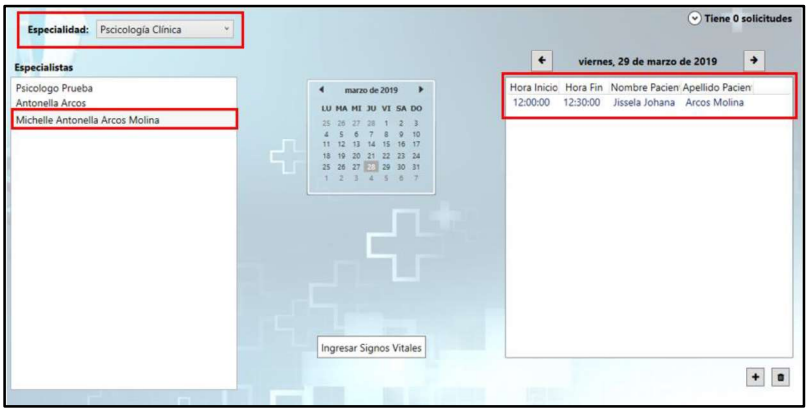


Figura 3.18. Cita en la agenda del especialista

- **Caso de prueba:** Impresión de citas agendadas

Procedimiento:

1. Seleccionar una cita determinada
2. La información de la cita se abre en un reporte
3. Se puede imprimir la cita agendada

Resultado esperado: Impresión de una cita agendada

Resultado obtenido: En la Figura 3.19 se muestra el reporte con la información de la cita agendada y la ventana que permite seleccionar la impresora para imprimir el documento.

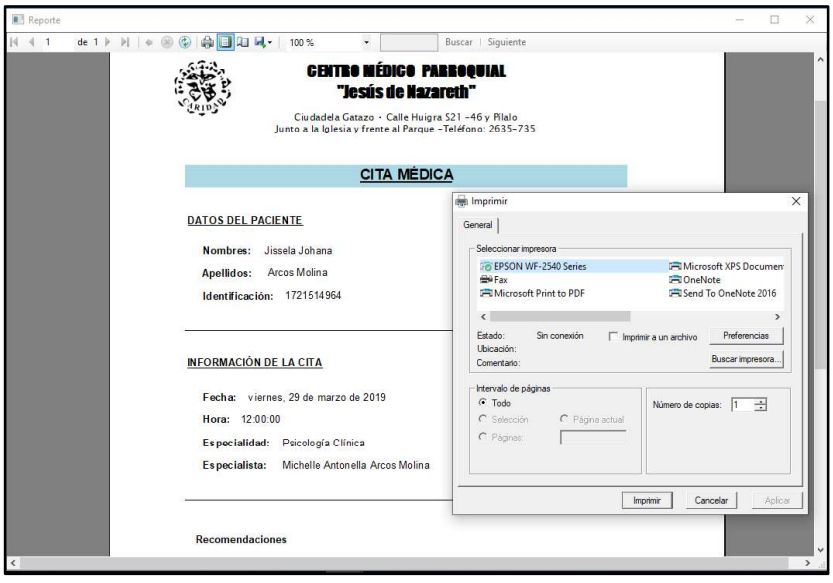


Figura 3.19. Impresión de una cita agendada

3.1.3 PRUEBAS DE FUNCIONAMIENTO ITERACIÓN 3

- **Caso de prueba:** Visualización de la lista de pacientes

Procedimiento:

1. El especialista selecciona la opción MIS CITAS Y PACIENTES
2. Se enlistan los pacientes en una lista

Resultado esperado: Los especialistas visualizan la lista de pacientes

Resultado obtenido: En la Figura 3.20 se observa la lista de pacientes de la especialidad Psicología Clínica.

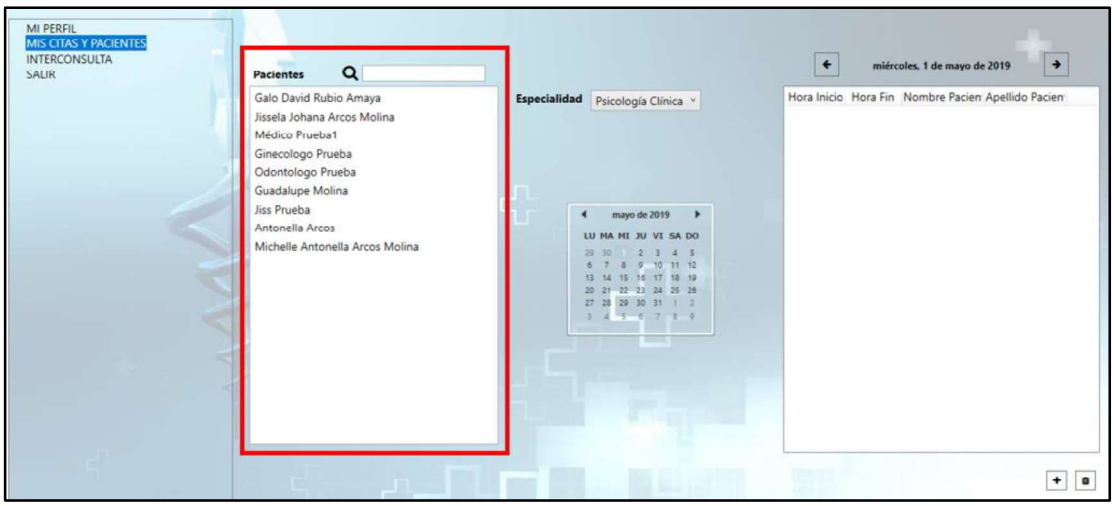


Figura 3.20. Visualización de la lista de pacientes

- **Caso de prueba:** Visualización del calendario y citas médicas agendadas a los especialistas

Procedimiento:

1. El especialista selecciona en su módulo la opción: MIS CITAS Y PACIENTES
2. El especialista visualiza un calendario
3. El especialista al seleccionar el día puede ver las citas agendadas para dicho día

Resultado esperado: Visualización de las citas agendadas en un día determinado

Resultado obtenido: En la Figura 3.21 se observa en la parte central un calendario para seleccionar un día determinado, en la parte derecha se encuentra las citas agendadas para el especialista en el día seleccionado, mostrando la hora de inicio y fin, el nombre y apellido del paciente.

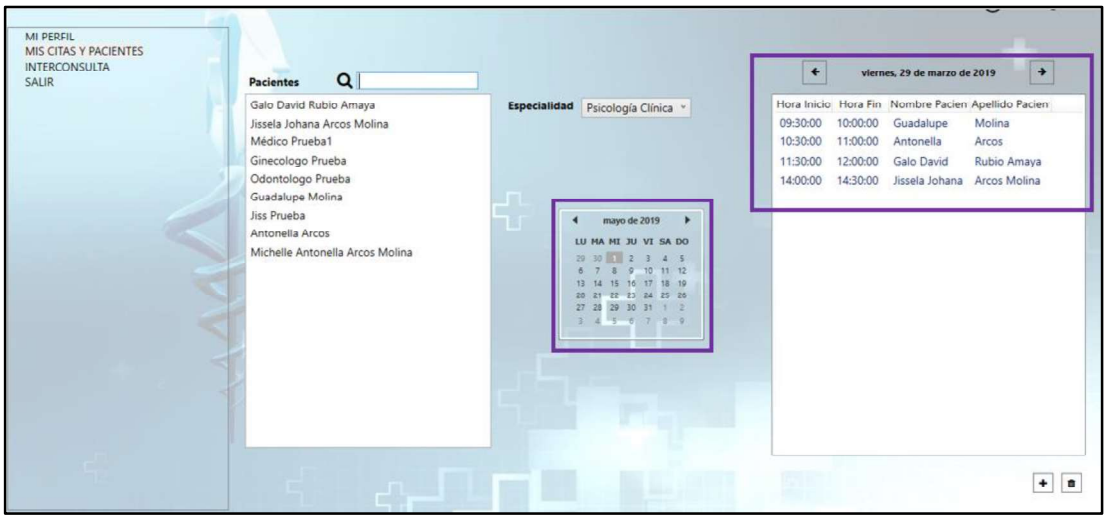


Figura 3.21. Visualización de la agenda de un especialista en el sistema de escritorio

- **Caso de prueba:** Creación de citas interconsultas

Procedimiento:

1. El especialista selecciona la opción INTERCONSULTA
2. El especialista puede seleccionar la especialidad y especialista
3. El especialista puede crear una cita interconsulta.

Resultado esperado: El especialista realiza citas interconsulta

Resultado obtenido: En la Figura 3.22 se observa como el Psicólogo Prueba puede agendar una cita para el Ginecólogo Prueba.

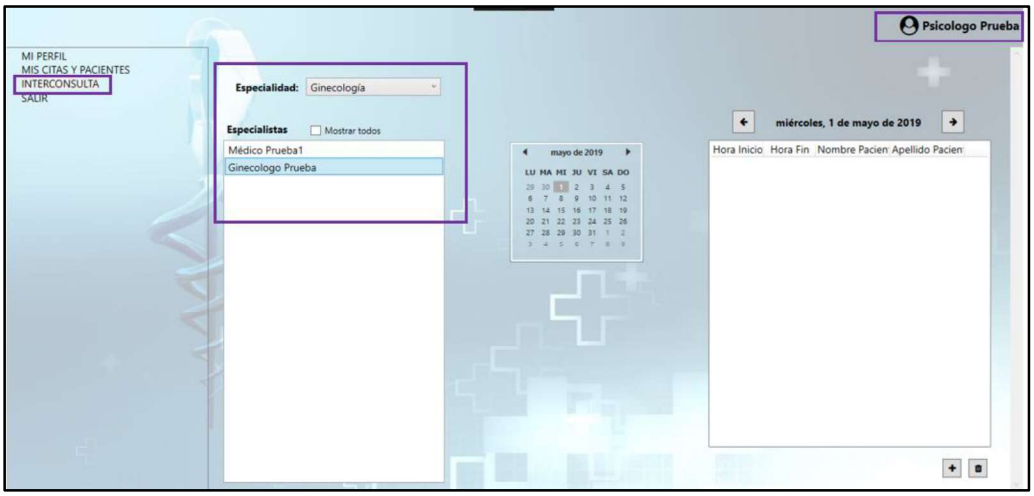


Figura 3.22. Generación de citas interconsulta

- **Caso de prueba:** Ingreso de datos de primera consulta en las diferentes especialidades

Procedimiento:

1. El especialista ingresa al sistema
2. El especialista selecciona un paciente determinado
3. El especialista ingresa los datos de la primera consulta para llenar la ficha determinada

Resultado esperado: Ingreso de datos de la primera consulta para las diferentes especialidades.

Resultado obtenido: Se lograron ingresar los datos de la primera consulta para las diferentes especialidades, como ejemplo en la Figura 3.23 se observa del ingreso de datos para la ficha de medicina general. En la Figura 3.24 se visualiza los datos ingresados para la ficha odontológica y en la Figura 3.25 se detallan los datos que son ingresados en la primera consulta de psicología.

Figura 3.23. Ingreso de datos en una ficha de medicina general

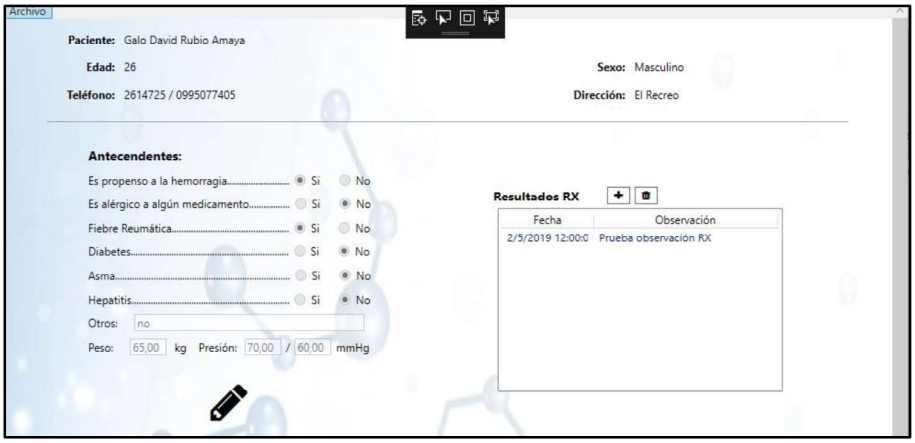


Figura 3.24. Ingreso de datos en una ficha odontológica

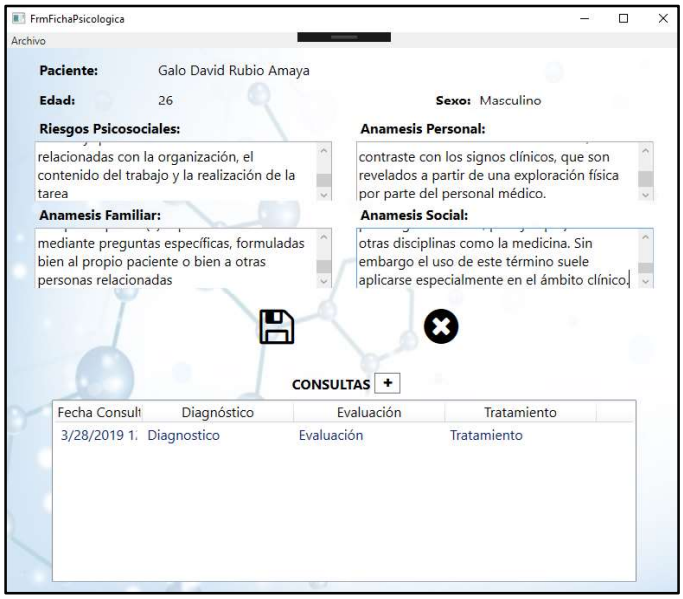


Figura 3.25. Ingreso de datos en una ficha psicológica

- **Caso de prueba:** creación, visualización y edición de consultas

Procedimiento:

1. El especialista ingresa al sistema
2. El especialista en cada consulta debe llenar los datos correspondientes a la nueva consulta

Resultado esperado: El especialista crea, visualiza y edita las consultas.

Resultado obtenido: Los especialistas pueden crear, visualizar y editar las consultas que han realizado. En la Figura 3.26 se muestra la consulta para medicina general, en la Figura 3.27 la consulta para odontología y en la Figura 3.28 la consulta para psicología.

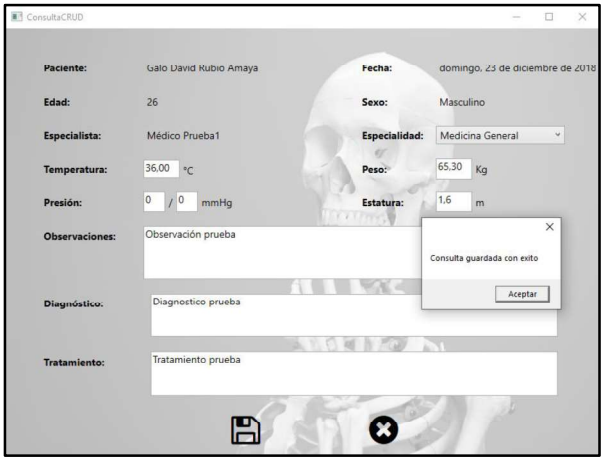


Figura 3.26. Consulta de medicina general

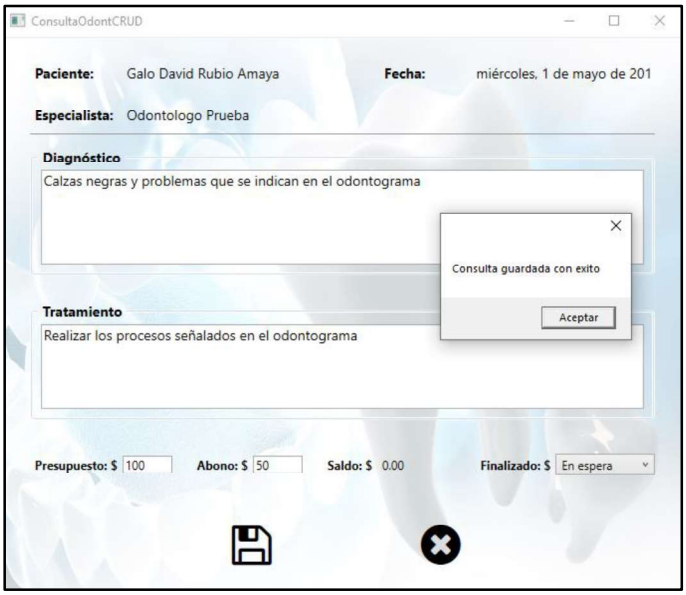


Figura 3.27. Consulta de Odontología

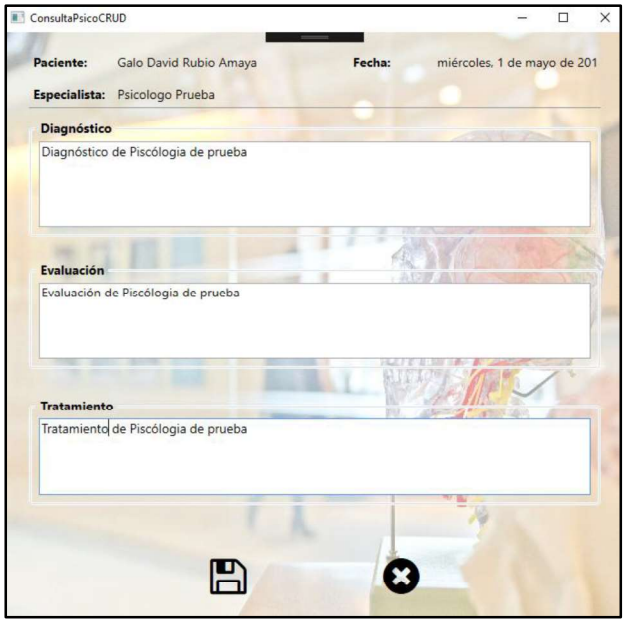


Figura 3.28. Consulta de Psicología

- **Caso de prueba:** Odontograma

Procedimiento:

1. El odontólogo abre la ficha odontológica de un paciente
2. El odontólogo debe seleccionar el procedimiento en la pieza dental determinada

Resultado esperado:

El odontólogo selecciona el procedimiento a realizar o realizado en cada pieza dental.

Resultado obtenido: En la Figura 3.29 se muestra los procedimientos que se pueden realizar en cada pieza dental, a manera de ejemplo se seleccionó la opción Caries/Restauración para seleccionar la parte de la pieza dental en la que se va a aplicar el procedimiento. En la Figura 3.30 se detalla varios de los procedimientos que se han realizado en el odontograma de un paciente y en la Figura 3.31 se tiene la simbología para los procedimientos odontológicos que posee el sistema.

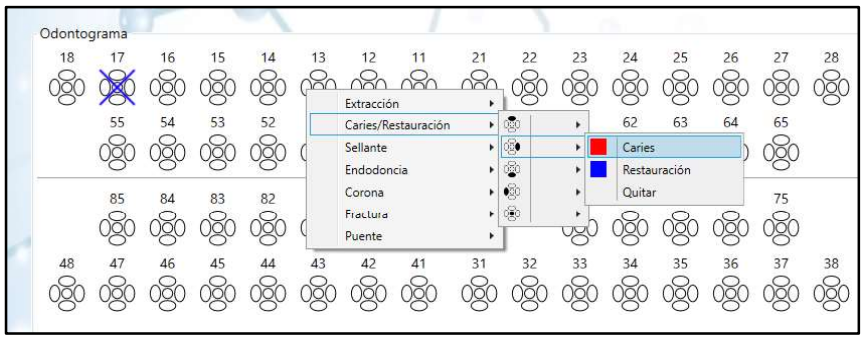


Figura 3.29. Selección de un procedimiento en el odontograma

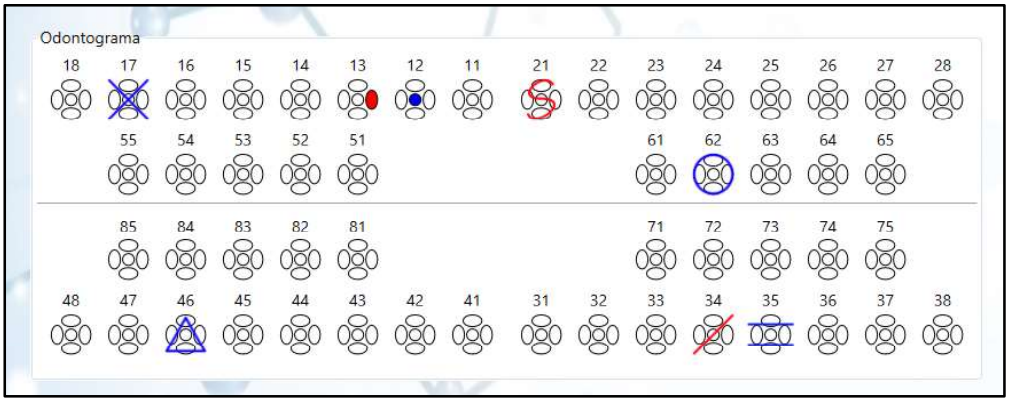


Figura 3.30. Odontograma con varios procedimientos en las piezas dentales

Figura 3.31. Simbología del odontograma

3.1.4 PRUEBAS DE FUNCIONAMIENTO ITERACIÓN 4

- **Caso de prueba:** Solicitud de exámenes de laboratorio

Procedimiento:

1. El laboratorista o los especialistas ingresan a su módulo.
2. Selecciona la opción crear un nuevo examen.
3. Seleccionan los exámenes que debe realizarse el paciente

Resultado esperado: El especialista o laboratorista selecciona los exámenes que el paciente debe realizarse, en caso de que los exámenes no se realicen en el Centro Médico se tiene la opción de imprimir la solicitud de exámenes.

Resultado obtenido: Como se observa en la Figura 3.32 el especialista puede seleccionar los exámenes que el paciente debe realizarse, después se genera un reporte con los exámenes solicitados, el cual tiene la opción de ser impreso (ver Figura 3.33).

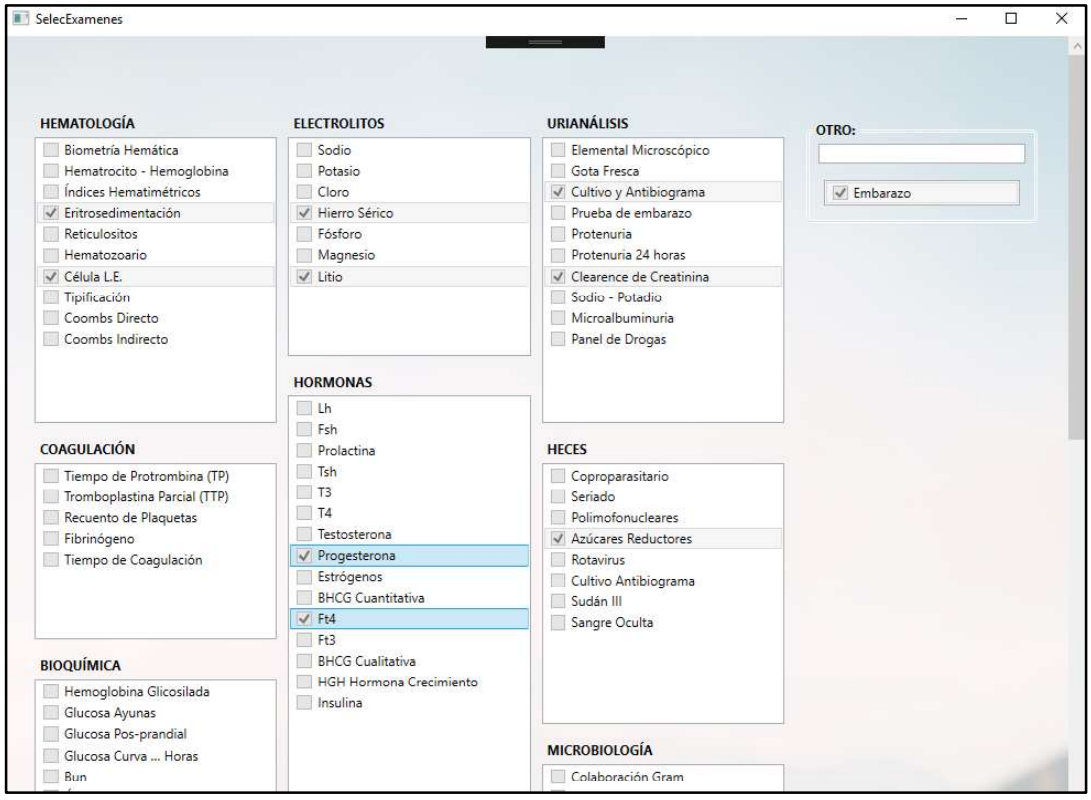


Figura 3.32. Selección de exámenes de laboratorio a realizarse

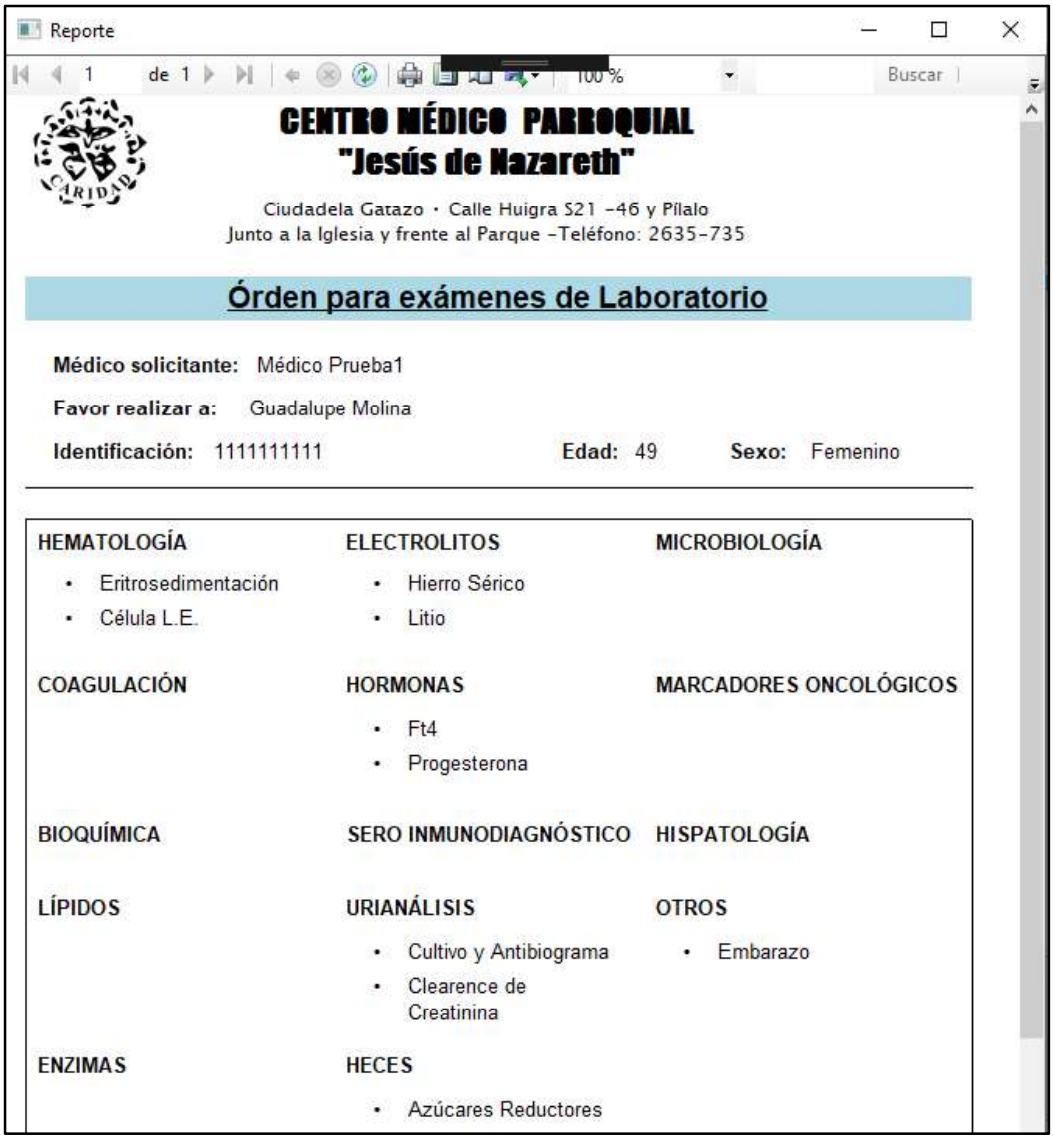


Figura 3.33. Reporte de la solicitud de exámenes médicos

- **Caso de prueba:** Visualización de los pacientes y sus respectivos exámenes de laboratorio

Procedimiento:

1. El laboratorista ingresa a su módulo.
2. El laboratorista visualiza todos los exámenes médicos solicitados en el Centro Médico.

Resultado esperado: El laboratorista puede visualizar todos los exámenes médicos solicitados en el Centro Médico.

Resultado obtenido: Como se observa en la Figura 3.34 el laboratorista puede ver una lista de todos los exámenes de laboratorio solicitados en el Centro Médico, el nombre del paciente, las fechas respectivas y el estado de éste.

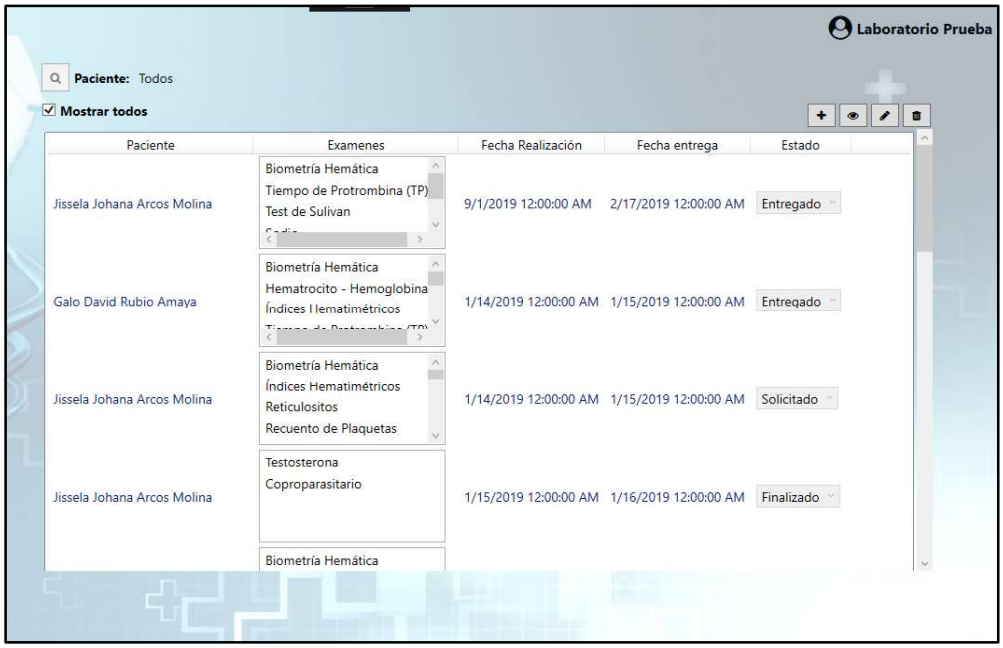


Figura 3.34. Listado de exámenes médicos solicitados en el Centro Médico

- **Caso de prueba:** Carga de resultados de exámenes

Procedimiento:

1. El laboratorista ingresa a su perfil
2. Se selecciona un examen determinado
3. En la opción cargar resultado debe seleccionar el archivo desde su ordenador con los resultados de los exámenes
4. El laboratorista presiona el botón Cargar y espera que aparezca el mensaje de éxito.

Resultado esperado: El laboratorista puede subir los resultados de los exámenes.

Resultado obtenido: El laboratorista selecciona desde su ordenador un documento denominado “ejemplo resultado laboratorio” (ver Figura 3.35). Posteriormente, al presionar el botón `Subir` se carga el documento exitosamente como se visualiza en la Figura 3.36.

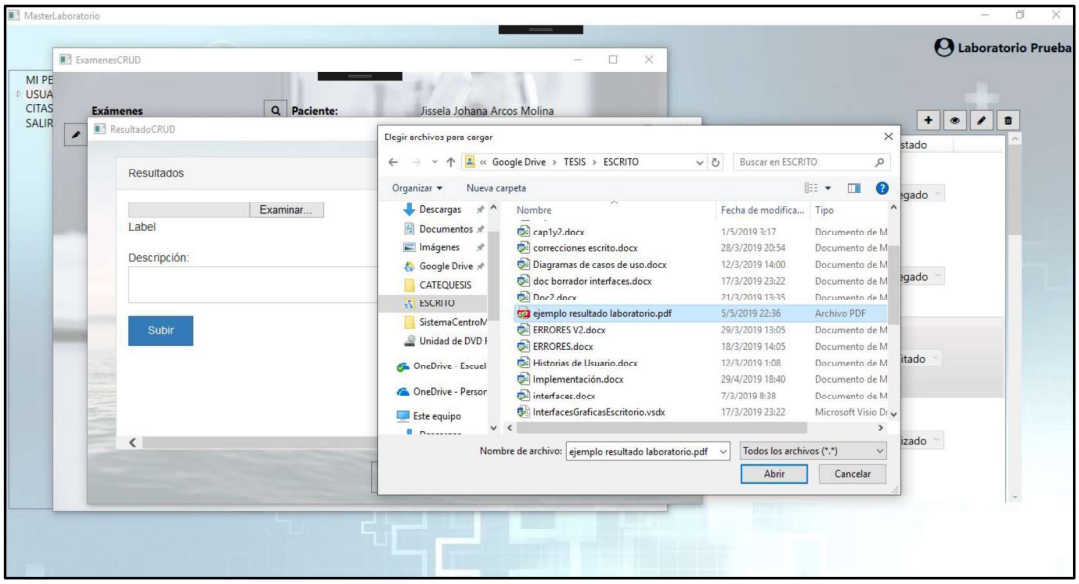


Figura 3.35. Selección de un documento desde el ordenador

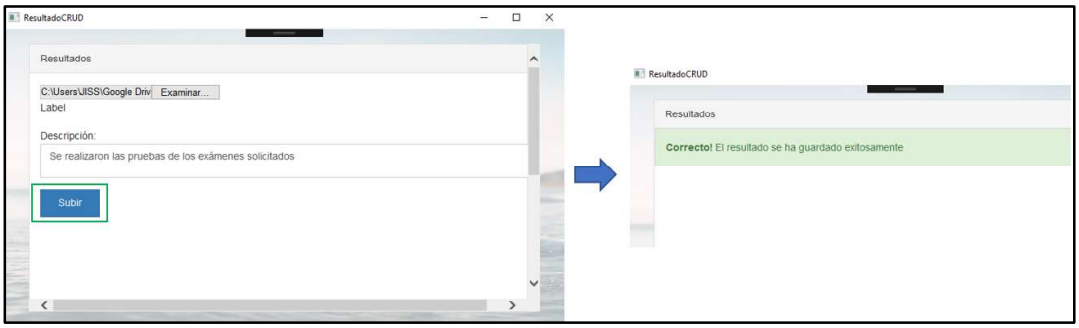


Figura 3.36. Carga exitosa de archivos

- **Caso de prueba:** Acceso a la historia clínica

Procedimiento:

1. El especialista ingresa a su perfil.
2. El selecciona a un paciente determinado y dar doble *click* sobre su nombre.
3. Se abre la ventana de la historia clínica, la cual está formada de la Ficha Médica, Ficha Odontológica, Ficha Psicológica y Exámenes Médicos.
4. En la parte inferior izquierda se encuentra un botón que permita visualizar completamente la historia clínica en un reporte, no se puede visualizar lo de Ficha Psicológica excepto si el especialista tiene un perfil de psicólogo.

Resultado esperado: Visualización de la Ficha Médica en un reporte.

Resultado obtenido: Como se visualiza en la Figura 3.37 el especialista al seleccionar un paciente determinado puede ingresar a su historia clínica la cual costa de: la Ficha Médica,

Ficha Odontológica, Ficha Psicológica y Exámenes Médicos, en la parte inferior izquierda se encuentra el botón para visualizar el reporte de la Historia Clínica completa. En la Figura 3.38 y en la Figura 3.39 se encuentra el reporte de una historia clínica de ejemplo.

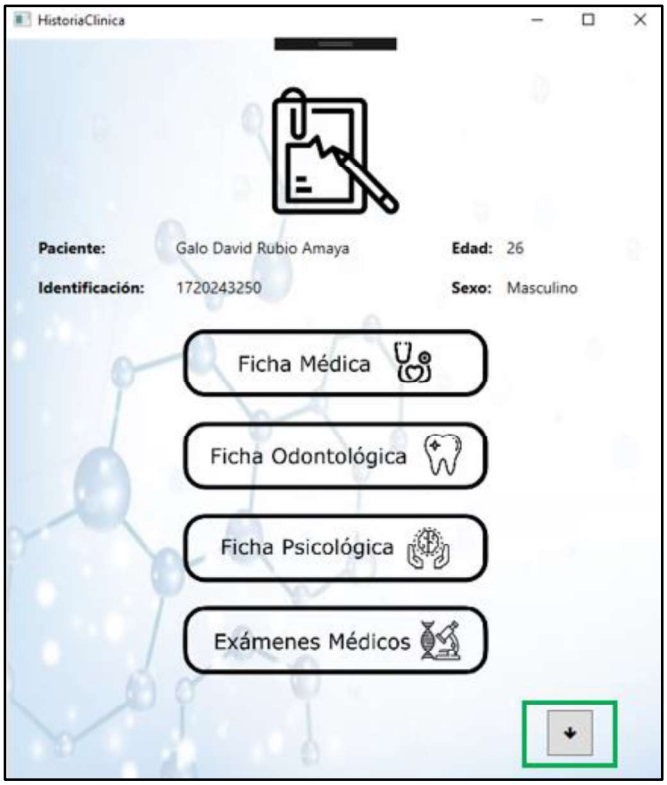


Figura 3.37. Ventana de la Historia Clínica

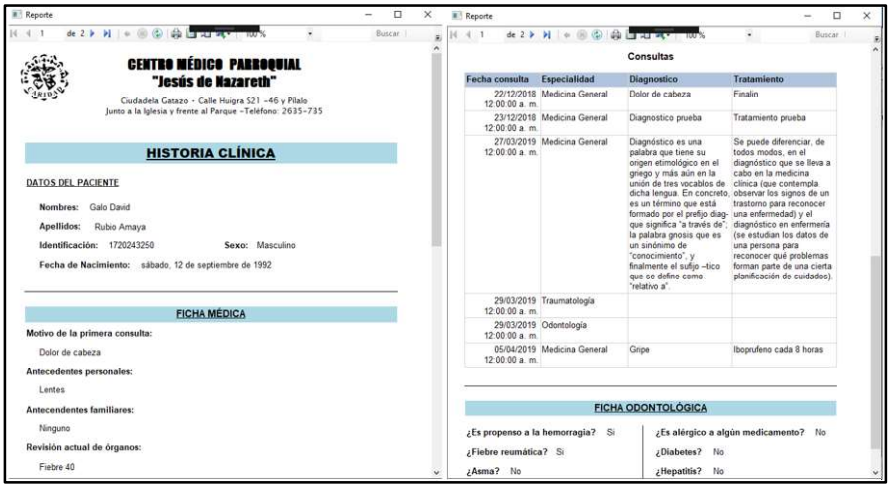


Figura 3.38. Reporte de ejemplo de Historia Clínica (parte 1 de 2)

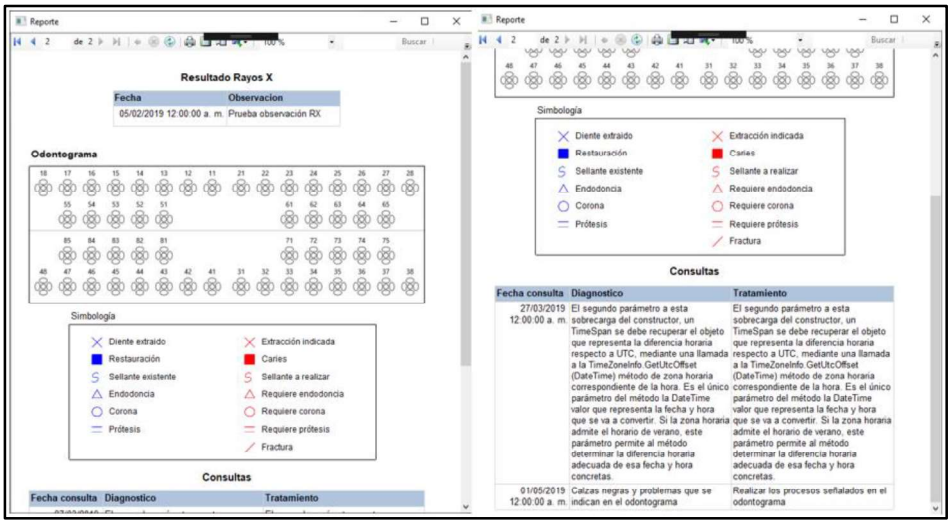


Figura 3.39. Reporte de ejemplo de Historia Clínica (parte 2 de 2)

- **Caso de prueba:** Impresión de la historia clínica de un paciente

Procedimiento:

1. El especialista ingresa a la historia clínica del paciente
2. Presionar el botón para generar el reporte
3. En el reporte seleccionar la opción imprimir y seleccionar la impresora

Resultado esperado: Los especialistas pueden imprimir el reporte de la historia clínica de un paciente determinado.

Resultado obtenido: Como se observa en la Figura 3.40 el especialista puede imprimir el reporte de la historia clínica de un paciente determinado.

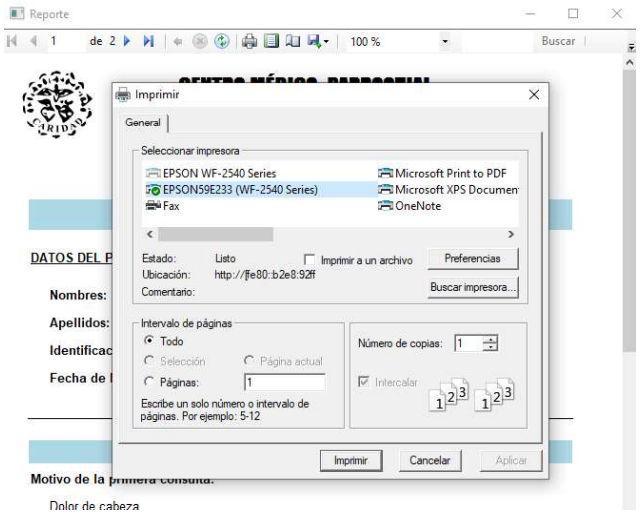


Figura 3.40. Impresión de la historia clínica

3.1.5 PRUEBAS DE FUNCIONAMIENTO ITERACIÓN 5

- **Caso de prueba:** Mensajes de error en el inicio de sesión en la aplicación móvil

Procedimiento:

1. El usuario en la aplicación móvil ingresa incorrectamente algún dato para la autenticación
2. Aparece el mensaje de error correspondiente

Resultado esperado: Mensajes de error correspondientes cuando el usuario inserta los datos incorrectamente para ingresar a la aplicación móvil:

- El usuario no existe
- Contraseña incorrecta
- Verifique el tipo de usuario

Resultado obtenido: Como se observa en la Figura 3.41 aparecen los tres tipos de mensaje de error cuando el usuario ingresa un dato incorrectamente al intentar ingresar a la aplicación móvil.

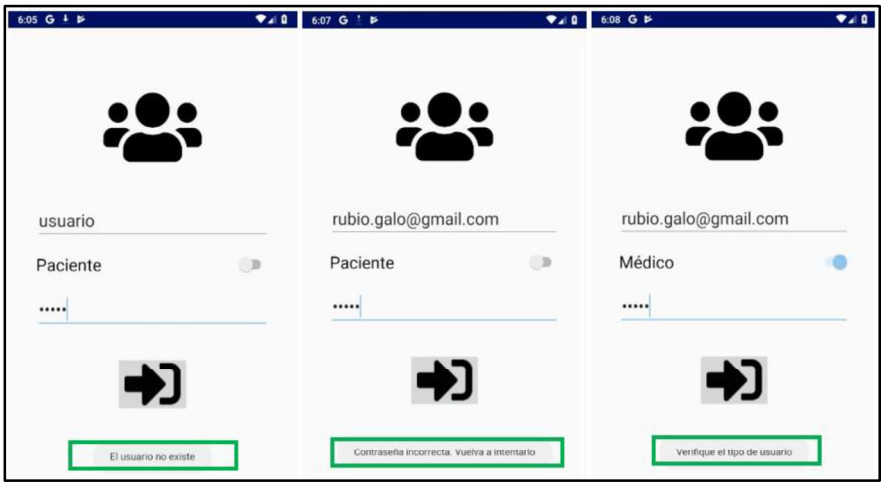


Figura 3.41. Mensajes de error de la interfaz *Login* de la aplicación móvil

- **Caso de prueba:** Inicio de sesión correcta y menú de la aplicación móvil

Procedimiento:

1. El usuario ingresa los datos correctos a la aplicación móvil y logra ingresar
2. Aparece el menú con las tres opciones: Mi Perfil, Citas e Historial Médico

Resultado esperado: Un usuario al ingresar correctamente a la aplicación móvil visualiza el menú.

Resultado obtenido: En la Figura 3.42 se visualiza el menú de la aplicación móvil.

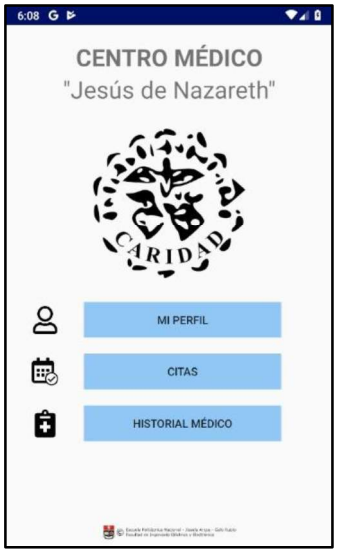


Figura 3.42. Menú de la aplicación móvil

- **Caso de prueba:** Opción Mi Perfil de la aplicación móvil

Procedimiento:

1. El usuario selecciona la opción Mi Perfil de la aplicación móvil.
2. El Usuario ingresa a Mi Perfil y puede visualizar y modificar datos de su perfil personal en la aplicación móvil.

Resultado esperado: El usuario visualiza y modifica datos de su perfil personal.

Resultado obtenido: En la Figura 3.43 se visualiza en la primera imagen los datos del perfil del usuario y en la siguiente imagen se observa la actualización de los datos del perfil.

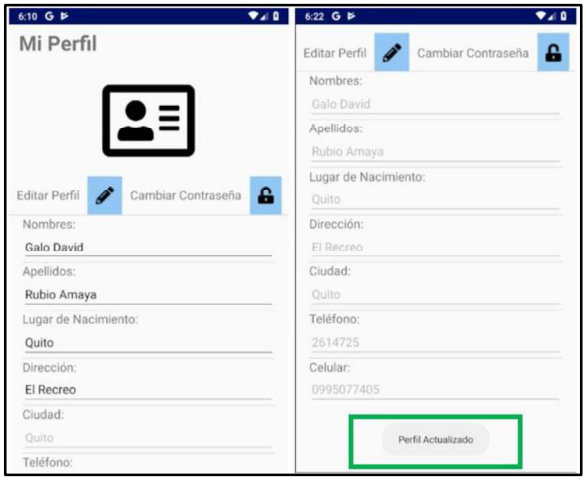


Figura 3.43. Opción Mi Perfil de la aplicación móvil

- **Caso de prueba:** Opción Citas de la aplicación móvil

Procedimiento:

1. El usuario selecciona la opción Mis Citas de la aplicación móvil
2. El usuario puede enviar la solicitud de una nueva cita en la aplicación móvil
3. La recepcionista aprueba la solicitud de la cita enviada
4. El usuario recibe un correo electrónico confirmando la cita agendada y puede visualizarla en la opción Mis Citas de la aplicación móvil.

Resultado esperado: Solicitar citas médicas por medio de la aplicación móvil y visualizar las citas que el usuario tiene agendadas.

Resultado obtenido: Para solicitar una nueva cita se debe llenar los datos requeridos como la especialidad, el médico, la fecha y el horario. A continuación, aparece un mensaje que indica que la cita está solicitada, en el sistema de escritorio la recepcionista recibe la solicitud y debe aprobarla o denegarla, este proceso se observa en la Figura 3.44. Si la solicitud de cita ha sido aceptada el usuario recibe un correo electrónico con la confirmación de la cita y también puede visualizar la cita en la opción Mis Citas de la aplicación móvil como se muestra en la Figura 3.45.

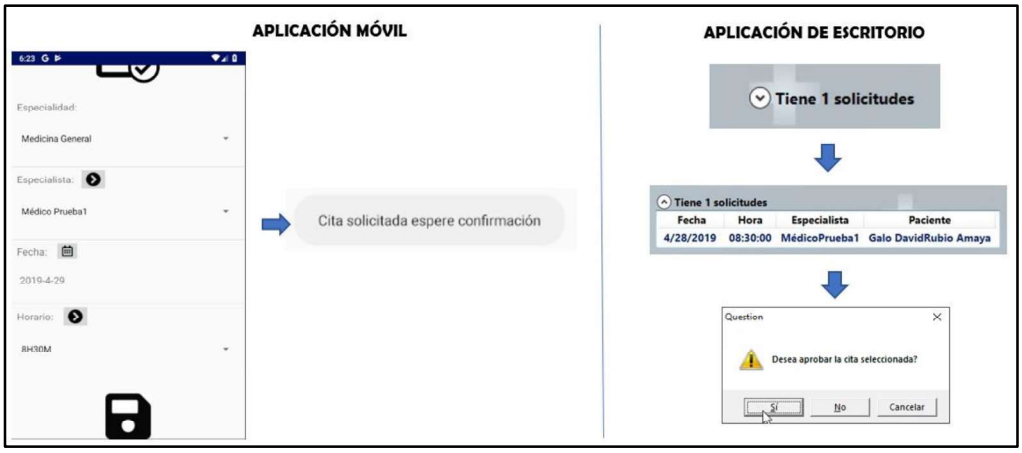


Figura 3.44. Envío y recepción de solicitud de una cita médica.

Figura 3.45. Recepción del correo y la visualización de la cita aprobada.

- **Caso de prueba:** Exportación a Google Calendar las citas asignadas

Procedimiento:

1. El usuario selecciona en la opción Mis Citas del menú, el botón para exportar citas a Google Calendar.
2. Las citas se exportan a Google Calendar.

Resultado esperado: Exportar las citas a Google Calendar.

Resultado obtenido: En la Figura 3.46 se detalla cómo después de presionar el botón exportar citas a Google Calendar esta se exporta correctamente.

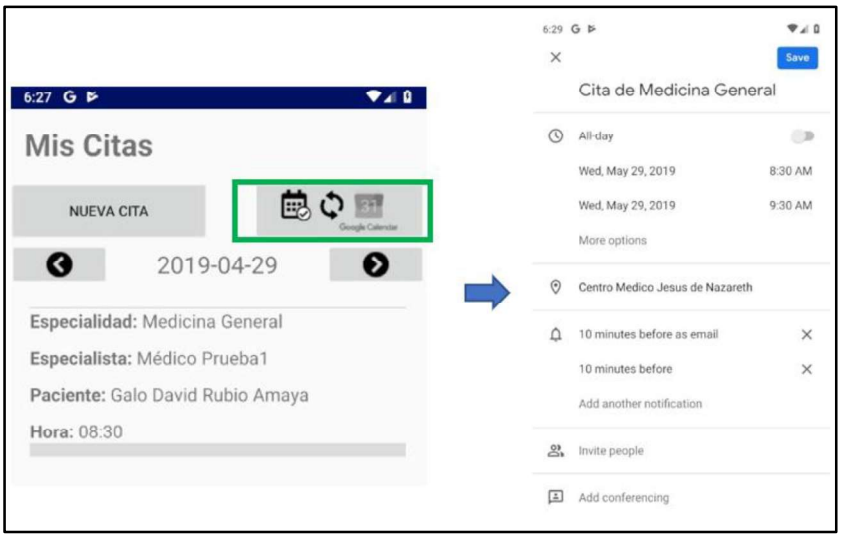


Figura 3.46. Exportación de las citas médicas a Google Calendar

- **Caso de prueba:** Opción historial médico de la aplicación móvil

Procedimiento:

1. El usuario selecciona la opción Historial Médico del menú de la aplicación móvil.
2. El usuario elige la especialidad.
3. Se visualizan las dos últimas consultas del usuario.

Resultado esperado: El usuario visualiza las dos últimas consultas de la especialidad seleccionada.

Resultado obtenido: Como muestra la Figura 3.47 el usuario al presionar el botón Ver Consultas visualizó las dos últimas consultas de Medicina General con su respectivo diagnóstico y tratamiento.

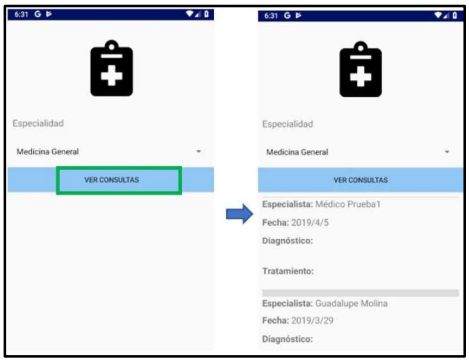


Figura 3.47. Opción Historial Médico de la aplicación móvil

- **Caso de prueba:** Recuperación de la contraseña utilizando la aplicación móvil

Procedimiento:

1. El usuario selecciona en la pantalla inicial el *link* para recuperar contraseña y confirmar que desea continuar.
2. El sistema envía un correo electrónico indicando el procedimiento a seguir.
3. Al ingresar al *link* proporcionado por el correo electrónico continúa el proceso hasta que aparezca el mensaje que confirme que la contraseña se ha actualizado correctamente.

Resultado esperado: Recuperación de la contraseña a través de la aplicación móvil.

Resultado obtenido: En la Figura 3.48 se encuentra los pasos iniciales para el envío del correo electrónico con un *link* para la modificación de la contraseña. La Figura 3.49 indica un proceso exitoso de cambio de contraseña.

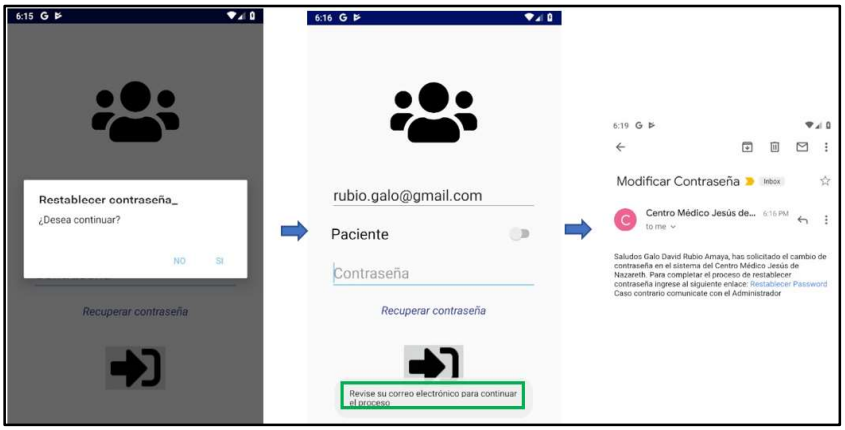


Figura 3.48. Recuperación de contraseña a través de la aplicación móvil (parte 1 de 2)

Figura 3.49. Recuperación de contraseña a través de la aplicación móvil (parte 2 de 2)

3.2 ANÁLISIS DE RESULTADOS

Los resultados que arrojaron las pruebas de funcionamiento fueron satisfactorios, es decir que el sistema prototipo cumple con todos los requerimientos que se definieron a través de las historias de usuario. En la Tabla 3.1 y Tabla 3.2 se detalla el cumplimiento de los requerimientos funcionales en las diferentes iteraciones.

Tabla 3.1. Cumplimiento de los requerimientos funcionales del sistema (parte 1 de 2)

Código Requerimiento	Cumple	Código Requerimiento	Cumple	Código Requerimiento	Cumple
RF-01-01	Si	RF-03-01	Si	RF-04-01	Si
RF-01-02	Si	RF-03-02	Si	RF-04-02	Si
RF-01-03	Si	RF-03-03	Si	RF-04-03	Si
RF-01-04	Si	RF-03-04	Si	RF-04-04	Si
RF-01-05	Si	RF-03-05	Si	RF-04-05	Si
RF-01-06	Si	RF-03-06	Si	RF-04-06	Si
RF-01-07	Si	RF-03-07	Si	RF-05-01	Si
RF-01-08	Si	RF-03-08	Si	RF-05-02	Si
RF-01-09	Si	RF-03-09	Si	RF-05-03	Si
RF-01-10	Si	RF-03-10	Si	RF-05-04	Si
RF-02-01	Si	RF-03-11	Si	RF-05-05	Si
RF-02-02	Si	RF-03-12	Si	RF-05-06	Si
RF-02-03	Si	RF-03-13	Si	RF-05-07	Si

Tabla 3.2. Cumplimiento de los requerimientos funcionales del sistema (parte 2 de 2)

Código Requerimiento	Cumple	Código Requerimiento	Cumple	Código Requerimiento	Cumple
RF-02-04	Si	RF-03-14	Si	RF-05-08	Si
RF-02-05	Si	RF-03-15	Si	RF-05-09	Si
RF-02-06	Si	RF-03-16	Si	RF-05-10	Si

Una característica del sistema de la metodología XP es la participación del cliente en todo el proceso del desarrollo del sistema, es por esto que los requerimientos funcionales como los sistemas operativos seleccionados, los mensajes, entre otros fueron comprobados por el cliente en el transcurso de las distintas iteraciones. En la Tabla 3.3 se muestra el cumplimiento de los requerimientos no funcionales del sistema. Estos se validaron a través de dos encuestas realizadas a los usuarios a través de formularios de *Google*. La primera encuesta se realizó a los trabajadores del centro médico sobre la experiencia al utilizar la aplicación de escritorio, mientras la segunda encuesta fue enviada a los usuarios de la aplicación Android. Las encuestas y sus respectivos resultados se encuentran en el Anexo J.

Tabla 3.3. Cumplimiento de los requerimientos no funcionales

Código Requerimiento	Cumple
RNF-01	Si
RNF-02	Si
RNF-03	Si
RNF-04	Si

Después de la fase de pruebas las correcciones que se realizaron fueron enfocadas a mejorar la experiencia del usuario. En los diferentes módulos del sistema *Windows* se aumentaron los mensajes de error para ayudar al usuario a utilizar correctamente el sistema. También se optimizó el tamaño y ubicación de las diferentes ventanas del sistema de escritorio para que se visualice de mejor manera la información. En la aplicación Android se mejoró el aspecto de los botones y se agregó mensajes de error para ayudar la experiencia del usuario.

4. CONCLUSIONES Y RECOMENDACIONES

4.1 CONCLUSIONES

- Al concluir este trabajo se dispone de un sistema prototipo que gestiona las citas médicas y registra historias clínicas para el Centro Médico “Jesús de Nazareth”. Está compuesto por: un cliente de escritorio, una aplicación móvil y un servidor publicado en la plataforma Azure.

- El cliente de escritorio fue desarrollado para el sistema operativo Windows, se manejó el entorno de desarrollo de Visual Studio, en el cual se utilizó el lenguaje XAML y la tecnología WPF para el desarrollo de las interfaces gráficas. Por otra parte, la codificación de la parte lógica se usó el lenguaje de programación C#. Además, se utilizó WCF para crear, alojar, consumir servicios utilizando la plataforma de Microsoft. El cliente de escritorio fue desarrollado para el uso del personal que trabaja en el Centro Médico y consta de los siguientes módulos: administrador, especialista, recepcionista y el correspondiente al laboratorio clínico. En consecuencia, permite realizar acciones CRUD sobre los usuarios del sistema, por otra parte, permite la gestión de citas y el registro de historias clínicas de los pacientes.

- En el cliente móvil desarrollado para el sistema operativo Android, se utilizó el entorno de desarrollo integrado Android Studio por lo que se utilizaron los lenguajes de programación Java y XML para la codificación de la aplicación. Esta aplicación fue desarrollada para el uso de pacientes y especialistas. A los dos tipos de usuario permite revisar las citas agendadas y solicitar nuevas citas, las cuales deben ser aprobadas por la recepcionista. Además, permite modificar los datos del perfil personal y revisar el diagnóstico y tratamiento de las dos últimas consultas en una determinada especialidad.

- El servidor del prototipo se desarrolló en Microsoft Visual Studio utilizando servicios WCF, el cual se lo implementó en la plataforma informática en la nube Microsoft Azure y para el almacenamiento de datos se utilizó SQL Server.

- El diseño e implementación de este prototipo fue realizado utilizando la metodología ágil de programación XP en la cual todos los requerimientos se expresaron en historias de usuario. También tuvo una planificación incremental con entregas parciales por lo que en el presente Proyecto de Titulación se tuvo cinco iteraciones. Otro aspecto importante es la presencia constante del cliente como parte del equipo de trabajo, es por lo que se mantuvieron varias reuniones

con los especialistas y la directora del Centro Médico durante el desarrollo del sistema prototipo.

- El prototipo implementado satisface las necesidades específicas del Centro Médico “Jesús de Nazareth” en comparación con las versiones disponibles en *software* libre. Un ejemplo de esto es el odontograma realizado con la simbología y características solicitadas por los odontólogos del Centro Médico.
- Las pruebas de funcionamiento realizadas en el sistema confirmaron el correcto funcionamiento de todos los módulos desarrollados, también el cumplimiento de los requerimientos funcionales y no funcionales que se obtuvieron de las historias de usuario.

4.2 RECOMENDACIONES

- Se sugiere la revisión de todos los servicios y las características que ofrecen los proveedores de servicios en la nube, debido a que se debe revisar la compatibilidad con las tecnologías utilizadas, el grado de escalabilidad y las tarifas de consumo, para posteriormente realizar un análisis de costo – beneficio y determinar la mejor opción.

- Se recomienda migrar los módulos desarrollados en WPF hacia ASP.NET u otras tecnologías web similares para que el cliente de escritorio pueda ser utilizado en cualquier sistema operativo.

- Para nuevas versiones del prototipo se recomienda añadir protocolos de seguridad en los servicios WCF, para no enviar las peticiones y respuestas en texto plano y evitar la filtración de información.

- Para ahorrar líneas de código y disminuir el riesgo de errores se recomienda utilizar una librería adicional en Visual Studio para el manejo de objetos JSON en las peticiones hacia el servidor.

- Al cargar los resultados de laboratorio se debe solamente subir archivos con formato PDF porque estos resultados no deben ser modificables y además este tipo de formato reduce el espacio de almacenamiento en el servidor.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] "What Is Windows Communication Foundation | Microsoft Docs." [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wcf/whats-wcf. [Accessed: 03-Feb-2019].

[2] S. Putier, *VB.NET y Visual Studio 2015 : los fundamentos del lenguaje*. ENI, 2016.

[3] "Microsoft Azure: plataforma y servicios de informática en la nube." [Online]. Available: https://azure.microsoft.com/es-es/. [Accessed: 21-Feb-2019].

[4] "SQL Server 2016 | Microsoft." [Online]. Available: https://www.microsoft.com/es-es/sql-server/sql-server-2016. [Accessed: 17-Oct-2017].

[5] A. M. Langer, *Guide to software development : designing and managing the life cycle*. Springer, 2012.

[6] "Crear una aplicación de WPF en Visual Studio | Microsoft Docs." [Online]. Available: https://docs.microsoft.com/es-es/dotnet/framework/wpf/getting-started/walkthrough-my-first-wpf-desktop-application. [Accessed: 16-Oct-2017].

[7] "Download Android Studio and SDK tools." [Online]. Available: https://developer.android.com/studio/index.html?hl=es-419. [Accessed: 04-Nov-2019].

[8] S. Putier, *VB.NET y Visual Studio 2015 : los fundamentos del lenguaje*. ENI, 2016.

[9] "Aplicabilidad del trabajo en la nube (Dropbox, Drive, IBM, etc.) en el ámbito laboral y en la educación – mtievteamblog," 2016. [Online]. Available: https://mtievteamblog.wordpress.com/2016/03/18/aplicabilidad-del-trabajo-en-la-nube-dropbox-drive-ibm-etc-en-el-ambito-laboral-y-en-la-educacion/. [Accessed: 04-Feb-2019].

[10] G. B., "IaaS vs PaaS vs SaaS: A Clear Explanation of Cloud Services," 2018. [Online]. Available: https://rubygarage.org/blog/iaas-vs-paas-vs-saas. [Accessed: 04-Feb-2019].

[11] I. Griffiths, C. Sells, and O. ' Reilly, "Programming Windows Presentation Foundation," 2005.

[12] "Ahorro de costos: SQL Server y Windows Server | Microsoft Azure." [Online]. Available: https://azure.microsoft.com/es-es/overview/azure-vs-aws/cost-savings/. [Accessed: 04-Jul-2019].

[13] “IDE de Visual Studio, editor de código, VSTS y centro de aplicaciones - Visual Studio.” [Online]. Available: https://visualstudio.microsoft.com/es/. [Accessed: 28-Jan-2019].

[14] J. Liberty *et al.*, “Programming C# 3.0 FIFTH EDITION,” 2008.

[15] M. MacDonald and M. MacDonald, *Pro WPF in C# 2008 : Windows presentation foundation with .NET 3.5*. Apress, 2008.

[16] M. Dalal and A. Ghoda, *XAML developer reference*. O’Reilly Media, 2011.

[17] L. A. MacVittie, *XAML in a nutshell*. O’Reilly, 2006.

[18] Microsoft, “Learn how to create a "Hello, world" app (XAML) - Windows UWP applications | Microsoft Docs,” 2018. [Online]. Available: https://docs.microsoft.com/en-us/windows/uwp/get-started/create-a-hello-world-app-xaml-universal. [Accessed: 05-Feb-2019].

[19] W. Anwar, “Introducing Windows Communication Foundation (WCF) Tutorial in Category WCF at EzzyLearning.com,” 2012. [Online]. Available: http://ezzylearning.com/tutorial/introducing-windows-communication-foundation-wcf. [Accessed: 05-Feb-2019].

[20] S. Klein, *Professional WCF programming : .NET development with the Windows Communication Foundation*. Wrox/Wiley Pub, 2007.

[21] “WCF - EcuRed.” [Online]. Available: https://www.ecured.cu/WCF. [Accessed: 03-Feb-2019].

[22] “¿Qué es LINQ? - TuProgramacion.com.” [Online]. Available: http://www.tuprogramacion.com/glosario/que-es-linq/. [Accessed: 21-Feb-2019].

[23] “Conoce Android Studio | Android Developers.” [Online]. Available: https://developer.android.com/studio/intro/?hl=es-419. [Accessed: 05-Feb-2019].

[24] P. Wang, *Java con programación orientada a objetos y aplicaciones www*. México: International Thomson Editors, 1999.

[25] “Cursos oracle,cursos java,master oracle,master java,formación en informatica.”

[26] F. Yergeau and J. Cowan, “Extensible Markup Language (XML) 1.1 (Second Edition).”

[27] T. Connolly and C. Begg, *Sistemas de bases de datos*, Cuarta. España: Pearson,

2005.

[28] L. Bassett, *Introduction to JavaScript object notation : a to-the-point guide to JSON*. .

[29] J. Sepulveda, “Trabajando con JSON | MDN,” 2017. [Online]. Available: https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/JSON. [Accessed: 24-Jan-2019].

[30] I. Sommerville, *Ingeniería Del Software*, Séptima. España: Pearson, 2005.

[31] *Técnica administrativa.* Ciencia y Técnica Administrativa, 2002.

[32] “Requerimientos no funcionales: Ejemplos - La Oficina de Proyectos de Informática.” [Online]. Available: http://www.pmoinformatica.com/2015/05/requerimientos-no-funcionales-ejemplos.html. [Accessed: 05-Jul-2019].

6. ANEXOS

ANEXO A. Historias de usuario

ANEXO B. Diagrama de clases en tamaño A3

ANEXO C. Mockups de las interfaces gráficas

ANEXO D. Script de la base de datos

ANEXO E. Procedimientos almacenados

ANEXO F. Interfaces de la aplicación móvil

ANEXO G. Código Fuente del prototipo desarrollado

ANEXO H. Manual de usuario del sistema

ANEXO I. Diagramas del sistema

ANEXO J. Encuestas realizadas a los usuarios del sistema

ORDEN DE EMPASTADO