ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

SISTEMA PROTOTIPO PARA LA EMISIÓN Y VERIFICACIÓN DE LA INTEGRIDAD DE DOCUMENTOS ACADÉMICOS

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

EDUARDO DANIEL GUZMÁN ENRÍQUEZ

DIRECTOR: Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO

CODIRECTOR: Ph.D. ANA MARÍA ZAMBRANO VIZUETE

Quito, julio 2020

AVAL

Certificamos que el presente trabajo fue desarrollado por Eduardo Daniel Guzmán Enríquez, bajo nuestra supervisión.

Ph.D. FELIPE LEONEL GRIJALVA ARÉVALO
DIRECTOR DEL TRABAJO DE TITULACIÓN

Ph.D. ANA MARÍA ZAMBRANO VIZUETE
CODIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

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

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

EDUARDO DANIEL GUZMÁN ENRÍQUEZ

DEDICATORIA

Este trabajo esta dedicado a:

A mis padres, Eduardo Guzmán y Myrian Enríquez
por su amor,
y ser mi impulso
para obtener cada meta trazada.

Eduardo Daniel

AGRADECIMIENTO

Primero a Dios quien a sido mi fortaleza para poder culminar esta meta.

Un agradecimiento especial a mis padres quienes a través de su paciencia, amor y buenos valores han sido un baluarte en mi vida.

Al Ph.D. Felipe Grijalva por cada consejo y toda la ayuda brindada en la elaboración de este proyecto.

De corazón gracias.

Eduardo Daniel

ÍNDICE DE CONTENIDO

RESUMEN

El presente proyecto de titulación propone el diseño y desarrollo de un sistema prototipo para la emisión y verificación de la integridad de documentos académicos. Este prototipo busca mejorar el proceso de emisión de los documentos académicos de las universidades, desarrollando un sistema web que automatice dicho proceso en donde, el receptor del documento pueda verificar su integridad, teniendo así la certeza de la autenticidad del mismo.

El prototipo tiene tres módulos. En el primer módulo se realiza la petición de documentos por parte del estudiante. El segundo módulo es el encargado de procesar la solicitud del estudiante y generar el documento académico. Una vez que el módulo haya terminado el proceso de generación del documento académico, enviará un correo electrónico al estudiante notificando que el documento se generó exitosamente, adjuntando el código hash asociado al mismo. El tercer módulo permite la descarga del documento académico ingresando el código hash asociado.

El prototipo fue programado con Laravel y usa como base de datos MongoDB. Una vez implementado el prototipo un grupo de voluntarios realizó pruebas al sistema reportando un altísimo ahorro de tiempo en la obtención del documento comparado al sistema manual. Además de generar una excelente experiencia de usuario y un alto rendimiento del sistema.

PALABRAS CLAVE: integridad, *hash*, *captcha*.

ABSTRACT

This project proposes a design and development of a prototype system for the issuance and verification of the integrity of academic documents. This prototype seeks to improve the process of issuing academic documents from universities, developing a web system that automates this process, where the recipient of the document can verify its integrity, thus being certain of its authenticity.

This prototype has three modules. In the first module the student requests for documents. The second module is in charge of processing the student´s request and generating the academic document. Once the module has finished the process of generating the academic document, it will send an email to the student notifying that the document was successfully generated, attaching that hash code associated with it. The third module allows the academic document to be downloaded by entering the associated hash code.

The prototype was programed with Laravel and uses MongoDB as a datebase. Once the prototype was implemented, a group of volunteers carried out tests on the system, reporting a very high time saving in obtaining the document compared to the manual system. In addition to generating an excellent user experience and high system performance.

KEYWORDS: *integrity*, *hash*, *captcha*.

1. INTRODUCCIÓN

El proceso de emisión de documentos académicos por parte de las universidades que no cuenta con un sistema digital completo para este proceso, es lento y no garantiza la integridad de los documentos académicos emitidos a las personas que reciben dichos documentos. Un ejemplo de esto es cuando el estudiante aplica a un empleo y como uno de los requisitos se le pide su histórico escolar, el estudiante hace las gestiones del caso y obtiene el documento con sus respectivas firmas y sellos. La empresa a la que se le entrega el documento académico no posee un medio que le garantice la autenticidad del documento, ya que actualmente replicar el diseño de un documento, duplicar sellos, falsificar firmas en documentos, editar documentos PDF no es tarea difícil. En otras palabras, las firmas o un sello no garantizan la integridad del mismo [1].

Ante esta problemática, el presente proyecto tiene como propósito fundamental, generar documentos académicos de una manera inmediata, mediante una plataforma en la cual se pueda verificar la integridad de los mismos [2,3]. Los documentos académicos que se podrán generar en el sistema serán definidos en el alcance.

De acuerdo a [4], un documento tiene integridad si el documento no ha sido alterado desde su emisión hasta llegar a su destino. Por lo tanto, es importante mencionar que el presente proyecto se enfocará exclusivamente en verificar la integridad de los documentos académicos que son emitidos por el prototipo.

Este proyecto se lo hace para resolver falencias en el proceso de emisión de documentos académicos de las universidades, que no cuentan con un sistema automatizado.

Primero, no existe actualmente alguna manera de comprobar la integridad de los documentos académicos emitidos por estas universidades; ya que en dichos establecimientos se sigue usando la firma y sello en el proceso de la emisión de los mismos. El sistema prototipo propuesto busca ser el modelo inicial para futuras implementaciones y así solventar este problema, ya que el receptor del documento académico emitido por la universidad, podrá verificar en el sistema la integridad del mismo. [1]

Segundo, el proceso de emisión de los documentos académicos es poco ágil; por lo general el estudiante tiene que acercarse personalmente a solicitar la emisión del documento académico. Luego esperar días hasta que el documento esté listo, debido a que el documento debe pasar por diferentes autoridades para su respectiva firma o sello.

Si una universidad implementa este sistema prototipo se simplificará el proceso de emisión de los documentos académicos, facilitando a los estudiantes la generación de los mismos, desde un ordenador con unos cuantos clics, sin tener que acudir a la secretaría de la facultad, generando los siguientes beneficios para la universidad:

- Optimizar el aprovechamiento de los recursos humanos. Generalmente quien se encarga de la emisión de los documentos académicos en las universidades es una secre-

taria la cual ocupa gran parte de su tiempo en gestionar los documentos académicos. Al simplificar el proceso de emisión de los documentos a la secretaria se le podrá asignar nuevas responsabilidades.

- Se disminuirá considerablemente el uso de los insumos de impresión, obteniendo una reducción de costos para la universidad.

1.1. OBJETIVO GENERAL

Diseñar un sistema prototipo para generar y verificar la integridad de documentos académicos.

1.2. OBJETIVOS ESPECÍFICOS

- Analizar las tecnologías y lenguajes necesarios para el sistema prototipo.
- Diseñar cada módulo del sistema prototipo para la emisión y verificación de la integridad de documentos académicos.
- Codificar los módulos del sistema prototipo.
- Realizar pruebas de funcionalidad en el sistema prototipo.

1.3. ALCANCE

El objetivo de este prototipo no es alterar cómo se valida la información, para generar los documentos académicos sino aportar con nuevas tecnologías para optimizar la forma en que estos se los realizan. Por lo tanto, el proyecto no realizará ninguna actividad que realiza el departamento Registrador de la Universidad.

Se desarrollará un sitio web en donde los estudiantes podrán generar documentos académicos y los receptores de los documentos académicos puedan verificar la integridad de los mismos. (ver Figura 1.1)

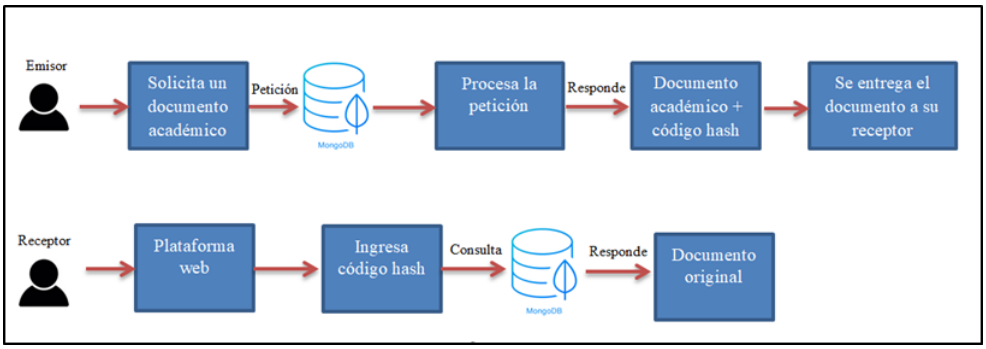


Figura 1.1: Diagrama de bloques

Específicamente el sistema web lo conformará los siguientes módulos:

El primer módulo se encargará de la generación de los documentos académicos, a este módulo los estudiantes ingresarán con sus respectivas credenciales. En la primera pantalla (ver Figura 1.2) del módulo se seleccionará el tipo de documento que se quiere generar, los documentos académicos que se podrán generar son los siguientes:

- Certificado de aprobación del plan de estudios.
- Histórico escolar.
- Certificado de promedio general.
- Certificado de créditos aprobados.
- Certificado de pertenecer a la Universidad.
- Certificado de agotó segunda matrícula.
- Certificado de agotó tercera matrícula.

Cabe destacar que este prototipo se alimentará de una base de datos con datos ficticios, para simular una integración con la base de datos de la universidad.

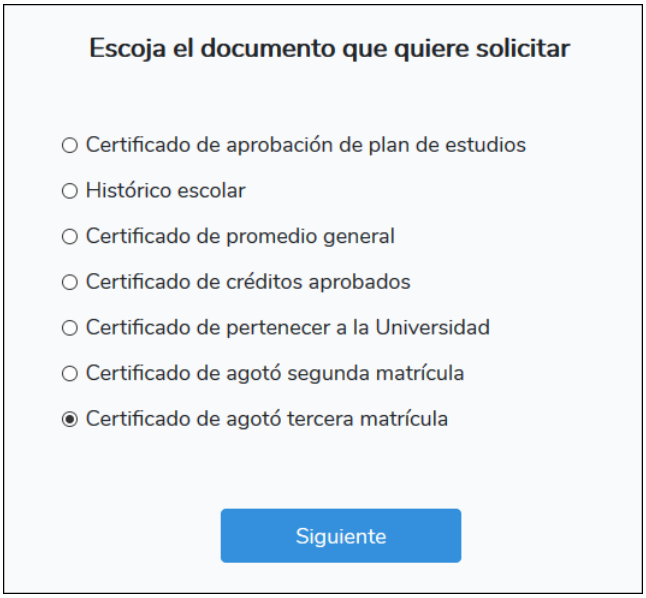


Figura 1.2: Tipo de documentos

Completada esta pantalla se pasará al paso 2, que es una pantalla meramente de confirmación. Finalizado todo este proceso, el módulo genera una petición para generar el documento al servidor.

El segundo módulo recibirá las peticiones del anterior paso, y realizará una consulta en la base de datos NoSQL orientado a documentos MongoDB, y en base a los datos del estudiante y de un banco de plantillas procesará la información de la petición, generando el

documento académico con su respectivo código único, haciendo uso de una función hash [5]. La función hash permite al sistema asegurar la integridad de los documentos académicos [2, 3]. Una vez listo el documento el módulo enviará un email al estudiante notificando que el documento ha sido generado correctamente, adjuntando el código hash asociado al documento solicitado, para poder ser descargado posteriormente en la plataforma web. El correo que le llegará al estudiante se asemejará a la Figura 1.3.

Figura 1.3: Notificación de email.

El tercer módulo de la aplicación web tendrá una pantalla similar a la que se muestra en la Figura 1.4. Para acceder a esta pantalla en el sitio web no es necesario hacer un login. La pantalla tendrá una caja de texto por medio del cual se ingresará el código que se genera en el módulo 2, además el usuario deberá completar la prueba captcha el cual permite determinar si la petición fue realizada por un humano [6]. Una vez completado estos pasos se enviará una petición al servidor, el cual hará una consulta en la base de datos y devolverá el documento asociado a este código para poder ser descargado. Este módulo será usado por el estudiante para obtener su documento y por el receptor del documento académico a fin de obtener el original el cual podrá ser comparado con el documento recibido y así tener la certeza y confianza de que el documento entregado no fue alterado.
Una vez terminada la fase de implementación se realizará la fase de pruebas (QA) en la cual se ejecutarán pruebas de funcionalidad con un grupo de estudiantes universitarios.

El producto final de este proyecto consistirá en una aplicación web, que permitirá la verificación de la integridad de los documentos académicos emitidos por el prototipo.

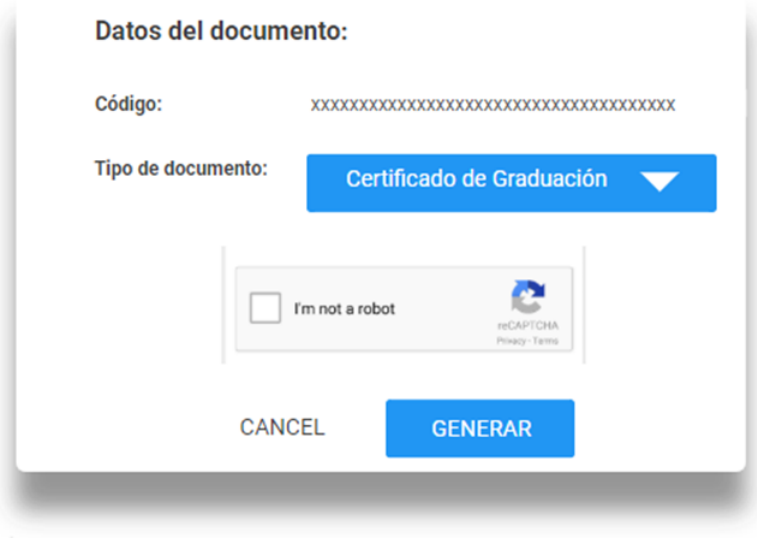


Figura 1.4: Descarga del documento.

1.4. MARCO TEÓRICO

El propósito de este capítulo es proporcionar una descripción acerca de la integridad de documentos y las funciones hash. En la Sección 1.4.3 se detalla la importancia del uso de la prueba *captcha*. En la Sección 1.4.7 se explica la metodología a usar en el presente prototipo. Finalmente en las Secciones 1.4.4, 1.4.5 y 1.4.6 se detallan las tecnologías y herramientas necesarias para la implementación de este.

1.4.1. INTEGRIDAD DE LA INFORMACIÓN DIGITAL

El paso de tener la información física a tenerla en forma digital nos brinda capacidades tales como almacenamiento y recuperación en línea, procesos de búsqueda eficientes y envío de datos a nivel mundial. Pero los datos digitales también traen consigo un problema importante, a saber, la facilidad con la que se puede copiar y manipular dicha información. Por lo tanto, ha surgido la necesidad de tecnologías que puedan salvaguardar la integridad y autenticidad de los datos digitales. La preservación de la integridad de los datos digitales es particularmente importante cuando están sujetos a posibles ataques criminales concertados [7].

Si la información que se envía de un punto A a un punto B es la misma se dice que la información está íntegra, es decir que en el trascurso del envío de información no ha sido alterada.

Para poder verificar la integridad de la información se usa diferentes tecnologías como:

- Funciones hash
- Encriptación de los datos
- Blockchain

1.4.2. FUNCIONES HASH (FUNCIONALIDADES O USOS DE LAS FUNCIONES HASH)

Es una función matemática que transforma un mensaje en una cadena de texto constante, generalmente más pequeña, que es la salida de la función, llamada un código hash [8]. Un hash en otras palabras es la huella de un archivo por lo tanto si un archivo se modifica va a cambiar su código hash. El objetivo principal de una función hash es verificar que el archivo o mensaje enviado no ha sido modificado hasta llegar al receptor, es decir preservar la integridad del mismo.

Otra aplicación de un código hash es ocultar la información sensible, por ejemplo la contraseña de un usuario al acceder a un determinado servicio.

Las aplicaciones más comunes de las funciones hash están orientadas a la seguridad informática y a la criptografía. Debido a que, el resultado de las funciones hash es similar a una huella digital de un texto, de ahí que el hash tiene una gran importancia en los procesos de emisión de certificados, firmas digitales, generación de claves y verificación de password.

Las funciones hash deben cumplir con las siguientes propiedades [9]:

- Unidireccionalidad: Una vez generado el código hash de cierto documento o texto, debe ser computacionalmente imposible encontrar el texto o archivo original a partir de dicho código hash, es decir, que esta función no tenga función inversa.
- Compresión: a partir de un mensaje de cualquier longitud, la función hash debe tener una longitud fija siempre menor a la del mensaje original.
- Facilidad de cálculo: La generación de esta función no debe significar ninguna complejidad a nivel computacional.
- Difusión: La función debe ser compleja, si el mensaje original es alterado así sólo sea por un caracter, la función hash resultante del texto alterado será completamente distinta.
- Colisión simple: será computacionalmente imposible encontrar dos funciones hash parecidas o similares.

A continuación en la Figura 1.5 se muestra el procesamiento general de una función hash
La función hash es determinista, es decir que partiendo de una misma entrada siempre se obtiene la misma salida. Sin embargo, el interés de estos algoritmos reside en que partiendo de entradas distintas se obtienen salidas distintas. En un buen algoritmo HASH es inadmisible que un conjunto reducido de modificaciones no altere el código resultante, ya que se podrían realizar retoques en el documento sin que fuesen detectados, debido a que no se garantiza la integridad [10].

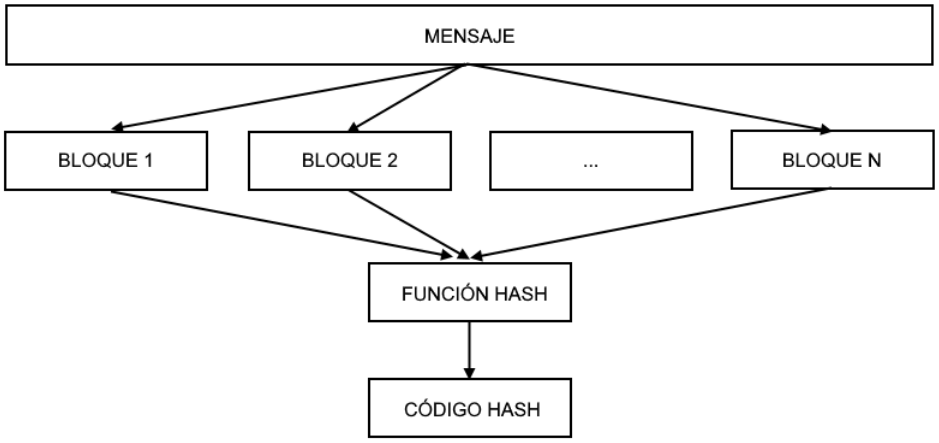


Figura 1.5: Procesamiento la función hash [10].

SHA (Secure Hash Algorithm)

Consiste en cinco funciones hash diseñadas por la Agencia de Seguridad Nacional (NSA) y publicadas por el Instituto Nacional de Estándares y Tecnología (NIST). Uno de ellos es SHA-2. SHA-2 es un conjunto de funciones hash seguras que incluyen SHA 224, SHA 256, SHA 384 y SHA 512 desarrollado por la NSA para proporcionar un mayor nivel de seguridad que el algoritmo SHA-1. SHA 224 y SHA 256 son algoritmos similares basados en una longitud de palabra de 32 bits que producen funciones de 224 y 256 bits. SHA 384 y SHA 512 se basan en palabras de 64 bits y producen funciones de 384 y 512 bits. En el presente trabajo se utilizará SHA 256 debido a las características antes mencionadas.

1.4.3. PRUEBA CAPTCHA

En la implementación de un sitio web, es de suma importancia la protección del sistema ante ataques de sistemas automatizados o *bots*, los cuales están tratando de apoderarse de las contraseñas de los usuarios, de publicar contenido *spam* y de modificar el contenido web [11]. Para que un sistema pueda proteger sus datos de forma segura se usa la prueba captcha.

La prueba captcha es una solución que permite demostrar que un solicitante es una persona y no un robot esto lo hace analizando el tráfico entrante de la aplicación, para así determinar si el tráfico proviene o no de sistemas automatizados. Para el presente proyecto se usará google ReCaptcha v3 el cual elimina la necesidad de pruebas interactivas, como en versiones anteriores. Este sistema otorga una puntuación para cada solicitud a la aplicación, calificando que tan probable es que la solicitud provenga de sistemas automatizados [12]. El puntaje toma valores de 0 a 1, el valor 1 indica que es muy probable que sea una buena interacción. Por el contrario, el valor de 0 indica que es muy probable que el tráfico lo haya generado un *bot* y por defecto se usa como umbral el valor de 0.5.

El algoritmo que emplea ReCAPTCHA v3 aprende al ver tráfico en tiempo real, detecta las iteraciones que tiene el usuario con el sistema y predice la probabilidad de que la solicitud

haya sido generado por un sistema automatizado. Por ejemplo en la situación de un inicio de sesión si ReCAPTCHA v3 indica que la solicitud proviene de un *bot*, el sistema puede tomar medidas como pedir autenticación de 2 factores o simplemente negar la solicitud, esto para evitar ataques de relleno de credenciales.

1.4.4. HERRAMIENTAS Y TECNOLOGÍAS POR UTILIZAR EN LA CAPA DE DATOS

La capa de datos es donde se encuentran los datos de la aplicación y es la que está a cargo de acceder a los mismos, por lo que generalmente la conforma un gestor de base de datos. A grandes rasgos una aplicación puede seleccionar dos tipos de base de datos relacional y no relacional. La diferencia principal entre estos dos tipos de base de datos es que en las relacionales los datos siguen una estructura fija mientras que en las no relacionales los datos no siguen una estructura fija. Este prototipo se lo hace para cualquier institución académica por lo que los datos no van a tener una estructura fija, por ende se va a utilizar una base de datos no relacional. Entre las base de datos no relacionales la más popular es MongoDb por lo que se optó esta opción. A continuación se detallan las herramientas y tecnologías que va usar este prototipo para la implementación del presente prototipo.

1.4.4.1. MongoDB

MongoDB es un sistema de bases de datos no relacionales. MongoDB fue diseñado para la web moderna, ya que esta posee enorme cantidad de datos no estructurados, además necesita poder escalar horizontalmente de una forma sencilla y sin que afecte a su rendimiento [13]. Entre sus principales clientes están empresas como Google, Verizon, Adobe y Sega.

Una de las soluciones de MongoDB es MongoDB Comunity Server, este es el software que usará el presente prototipo que a su vez incluye MongoDB Shell, que permite administrar e interactuar con la base de datos desde una línea de comandos.

Estructura

Es una base de datos orientada a documentos, guarda los datos en formato BSON que no es mas que un JSON codificado. Un registro en MongoDB se denomina *documento* estos son similares a los objetos JSON. MongoDB almacena los *documentos* en colecciones, estas son análogas a las tablas en bases de datos relacionales [14]. La Tabla1.1 muestra una analogía de cómo estructura los datos MongoDB vs las bases de datos relacionales.

Tabla 1.1: Analogía entre MongoDB y bases de datos relacionales.

MongoDB	Base de datos relacional
Base de datos	Base de datos
Colección	Tabla
Documento	Fila o registro
Campo	Columna

Índices

Un índice en MongoDB permite al momento de hacer una consulta en la colección, evitar escanear cada documento de una colección, es decir realizar consultas eficientes [15]. Un índice tipo *unique* en MongoDB asegura que los campos indexados no almacenen valores duplicados, es decir no existirá dos documentos con el mismo campo, al cual se le asignó un índice tipo unique [16].

1.4.4.2. Paquete Laravel MongoDB

Este paquete permite la comunicación de una aplicación hecha en Laravel con MongoDB, mediante el uso de *Eloquent*, la cual es una característica de Laravel para trabajar de forma fácil e intuitiva con la base de datos [17]. Para poder trabajar con este paquete es necesario instalar el Driver MongoDB PHP como se indica en la documentación oficial del paquete [18].

1.4.5. HERRAMIENTAS Y TECNOLOGÍAS POR UTILIZAR EN LA CAPA DE NEGOCIO

La capa de negocio es la que contiene la lógica de la aplicación, permitiendo recibir las solicitudes del usuario, procesarlas y enviar la información. Esta capa solicita información a la capa de datos, para que en base a las reglas y lógica del sistema envíe información a la capa de presentación, para que se muestre posteriormente la información correspondiente al usuario. Para que el proyecto este disponible en internet se necesita de un servidor para esto existen diferentes empresas que proporcionan este servicio como Microsoft, Amazon, Hostinger, etc. Para el presente proyecto se escogió el servicio EC2 de Amazon Web Service para que sea el servidor que aloje nuestro proyecto, dado que posee un plan gratuito por un año que incluye entre las principales características:

- 30GB de almacenamiento interno.
- 1GB de memoria RAMM
- Fireway
- Certificado SSL gratis

Uno de los lenguajes mas usados para la realización de páginas web es PHP, según varios portales mas de la mitad de sitios en la web usan PHP. Por la extensa documentación que existe y por su importancia en la web se escogió PHP para la codificación del proyecto. Adicionalmente se va a usar un framework de php llamado Laravel ya que proporciona seguridad y una gran cantidad de librerias. Laravel se basa en la arquitectura modelo vista controlador, que hace que el código tenga un orden y sea fácil de comprender. Las herramientas y tecnologías que se va a usar en esta capa se detalla a continuación:

1.4.5.1. Amazon Elastic Compute Cloud

Ec2 es una parte central de la plataforma de cómputo en la nube de la empresa Amazon denominada Amazon Web Services.

Posee un entorno informático virtual conocido como instancia, que no es más que un computador en la nube, en donde se puede configurar: cpu, memoria, sistema operativo y capacidad de red. El ingreso a la instancia se lo hace de manera segura mediante pares de claves, AWS en este caso almacena la clave pública y el usuario la almacena como una clave privada en un lugar seguro [19].

Su configuración de seguridad, redes, almacenamiento, permite escalar hacia arriba o hacia abajo la instancia para controlar los cambios en los requisitos o picos de popularidad, con lo que se reduce la necesidad de prever el tráfico. Esto proporciona a los clientes el poder desarrollar e implementar aplicaciones escalables en menor tiempo.
Consta de un firewall que permite especificar los protocolos, puertos y los rangos de direcciones desde las cuales se admitirá el tráfico.

1.4.5.2. Laravel

Laravel es un framework de código abierto que permite desarrollar aplicaciones y servicios web usando uno de los lenguajes más populares de internet php. Su objetivo es desarrollar aplicaciones en php de una forma elegante y simple, evitando aplicaciones con una estructura de control de flujo compleja e incompresible. Esta hecha con la arquitectura Modelo Vista Controlador, esta arquitectura separa la lógica del negocio, los datos de la aplicación y la interfaz de usuario en tres partes [20]:

- **Modelo**: Contiene la representación de los datos del sistema.
- **Vista**: Interfaz que interactúa directamente con el usuario.
- **Controlador**: Actúa de intermediario entre la Vista y Modelo, capturando las solicitudes del usuario para procesarlas e interactuar con el modelo, solicitando o actualizando los datos de la aplicación.

A continuación se explican algunas características y componentes de Laravel [21]:

- **Sistema de rutas:** Es un sistema intuitivo de rutas en donde se define las diferentes direcciones que va a tener la aplicación y los métodos HTTP asociado a las mismas, como POST y GET.
- **Middlewares:** Encargadas de agregar seguridad a las rutas de la aplicación, permite verificar los permisos de un usuario antes de poder acceder a un recurso.
- **Modelos:** Es el componente que permite la comunicación con la base de datos, es decir consultar información e ingresarla.
- **Eloquent:** Es el ORM que incluye Laravel, proporciona una forma fácil de interactuar con la base de datos gracias a que aporta con gran cantidad de funciones. Evita hacer consultas en el lenguaje nativo de la base de datos.
- **Controladores:** Es el lugar donde se define toda la lógica de la aplicación como el manejo de solicitudes y todo su procesamiento.

1.4.5.3. Composer

Es un manejador de dependencias para el lenguaje de programación PHP [22], evita al usuario tener que descargar cada dependencia de forma manual. Genera un archivo composer.json, en donde especifica requisitos y versiones mínimas de los paquetes para una aplicación, esto ahorra tiempo al usuario ya que no debe estar analizando la compatibilidad de cada paquete e instalando cada paquete.

1.4.5.4. Git

Git es esencialmente un sistema de control de versiones, de distribución libre y de código abierto, diseñado para tener un registro y control en los cambios que va a tener un proyecto a lo largo del tiempo. Git crea un repositorio local, que es donde se guardará todos los cambios que se han realizado en un proyecto. Si un cambio en el código produce un error, con Git es fácil identificarlo ya que podemos saber que líneas de código son diferentes entre una versión de la otra. [23]

1.4.5.5. Github

Permite alojar a los desarrolladores un repositorio de Git en la nube, para que otras personas puedan colaborar en proyectos con ellos, ya sea que estén abiertos a la contribución pública o cerrados para que colegas específicos trabajen en un proyecto privado.
La idea no es diferente a la forma en que Google Docs le permite alojar sus archivos de procesamiento de texto y hojas de cálculo, y como los abre a la colaboración. Los desarrolladores no trabajan juntos en los mismos documentos en tiempo real o realizan cambios directamente en el navegador. En Git cada colaborador va a tener una copia del repositorio en donde pueden realizar cualquier cambio y gestionar las versiones como desee. Cuando quiera hacer algún cambio en el repositorio principal, este cambio debe ser aprobado por el administrador del proyecto para que se haga efectivo. [24]

1.4.6. HERRAMIENTAS Y TECNOLOGÍAS POR UTILIZAR EN LA CAPA DE PRESENTACIÓN

La capa de presentación contienen todas las interfaces del sistema, estas son las que interactúan directamente con el usuario final, esta capa en otras palabras es la parte visual del sistema que muestra los datos a los usuarios y captura las solicitudes de los usuarios. Para la codificación de esta capa se va a usar las siguientes herramientas:

1.4.6.1. Bootstrap

Bootstrap es un *framework* basado en HTML, CSS y JavaScript para crear páginas y aplicaciones web. Es un proyecto gratuito y de código abierto, alojado en GitHub , y creado originalmente por Twitter.
Bootstrap facilita el desarrollo de aplicaciones responsivas ya que sus clases de CSS ya están configuradas para este fin. El diseño responsivo hace posible que una página web o

aplicación detecte el tamaño y la orientación de la pantalla del visitante y adapte automáticamente la pantalla.

A los diseñadores y desarrolladores web les gusta Bootstrap porque es flexible y fácil de trabajar. Sus principales ventajas son:

- Facilita el diseño responsivo.
- Mantiene una amplia compatibilidad con el navegador.
- Es muy fácil de usar y rápido de aprender.
- Ofrece una amplia extensibilidad utilizando JavaScript, y viene con soporte incorporado para complementos jQuery y una API de JavaScript.
- Bootstrap se puede usar con cualquier IDE o editor, y con cualquier tecnología e idioma del lado del servidor, como ASP.NET, PHP y Ruby on Rails.

1.4.6.2. Sistema de plantillas Blade

Es un motor de plantillas que viene incluido en Laravel. Algo que le destaca de otros motores de plantillas, es que permite usar código PHP en sus vistas. Genera archivos con la extensión ".blade.php", a estos Laravel les llama vistas, que no es más que la interfaz que interactúa con el usuario. Estos archivos contienen código HTML, CSS, JavaScript y PHP. Todas las vistas de blade se compilan en el código PHP simple y se almacenan en caché hasta que se modifican, lo que significa que Blade agrega esencialmente cero sobrecarga a su aplicación [25].

1.4.7. METODOLOGÍA KANBAN

Es un método visual para controlar la gestión de los procesos, para poder alcanzar la máxima eficacia y eficiencia en el desarrollo de las tareas. El sistema Kanban se encarga de controlar y organizar las tareas necesarias para poder llevar a término la ejecución de los procedimientos que forman parte de un proyecto. Una de las mejores ventajas que ofrece este sistema es que permite mejorar la gestión sin tener que realizar cambios relativamente grandes en la estructura organizativa de un proyecto.

Kanban busca limitar el trabajo en curso, ya que está demostrado que, cuanto más trabajo en curso se gestione a la vez, los índices de calidad disminuyen drásticamente. Aumentar el trabajo en curso implica generar mayor cantidad de errores al proyecto, como consecuencia de la poca capacidad de concentración que los desarrolladores podrán dedicarles a las tareas. En general se puede decir que el Kanban, al trabajarlo con estricta disciplina limita la cantidad de trabajo a realizar, retorna mayores índices de calidad y un tiempo de servicio bastante menor.

Kanban hace una división del trabajo por medio de etapas, las más comunes son: Lista de tareas, Tareas en proceso, Tareas realizadas. Cada tarea estará pasando a la etapa de Tareas en proceso, conforme se vayan finalizando las tareas de mayor prioridad.

2. METODOLOGÍA

En el presente proyecto de titulación se diseña y desarrolla un sistema prototipo para la emisión y verificación de la integridad de documentos académicos. Para esto, el proyecto se desarrollará en dos etapas el diseño (ver Sección 2.1) y la implementación (ver Sección 2.2). En la primera etapa de diseño se establecerán todos los requerimientos y propiedades que debe tener el prototipo para luego diseñar cada capa que conforma el mismo. El prototipo va a estar dividido en tres capas Capa de Datos, Capa de Negocio y Capa de Presentación. La etapa de implementación detallará la codificación de la etapa de diseño.

2.1. DISEÑO

2.1.1. PLANTEAMIENTO DEL TABLERO KANBAN

El presente tablero Kanban presenta las diferentes fases por la que el prototipo deberá pasar hasta su culminación (ver Figura 2.1). En donde se observa que las tareas relacionadas a la fase de investigación están finalizadas por ende dichas tareas se ubican en la columna “Tareas finalizadas”. Las tareas relacionadas a la etapa del diseño están en curso por ende se ubican en la columna “Tareas en curso”. Las tareas relacionadas con pruebas finales y presentación de resultados están en estado pendiente por los que se ubican en la columna “Lista de tareas”.

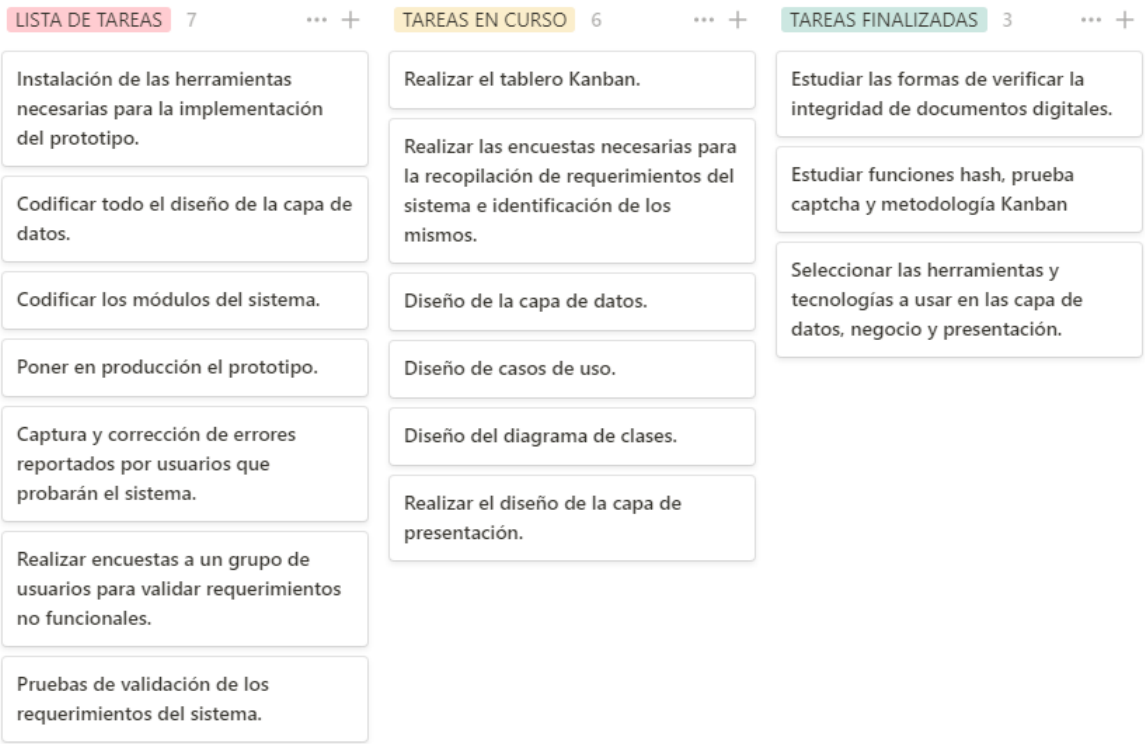


Figura 2.1: Tablero Kanban etapa de diseño.

2.1.2. ENCUESTAS PARA LA OBTENCIÓN DE REQUERIMIENTOS

En las universidades que no poseen un sistema web para la emisión automatizada de documentos académicos, el proceso de emisión es manual y generalmente siguen los pasos que se describen a continuación:

- El estudiante acude a la universidad solicita el documento académico.
- Luego el funcionario encargado de recibir el documento académico valida la información en su sistema para verificar si es posible la emisión del documento.
- Si pasa la validación anterior, el funcionario realiza la gestión para conseguir las firmas y sellos requeridos.
- Una vez que el documento académico tenga los sellos y firmas requeridas, el documento está listo para que el estudiante lo retire.

Cabe recalcar que cuando el documento está listo, el estudiante no recibe ninguna notificación, por consiguiente el interesado debe estar dando seguimiento a dicho proceso.

Partiendo de esta hipótesis se realizó unas encuestas a un grupo de estudiantes universitarios, en la Plataforma de Formularios de Google a fin de obtener los requerimientos funcionales y no funcionales. Para la realización de las antes mencionadas encuestas, cada usuario como requisito debió iniciar sesión con su cuenta de Gmail [26].

El objetivo de las encuestas es obtener la opinión de los estudiantes a fin de de automatizar el proceso para la emisión de documentos académicos. Conocer que tan conformes están con el proceso actual de emisión de documentos académicos, y que actividades cree que deben mejorar en el proceso actual, con el fin de diseñar un sistema que supla los problemas del proceso actual y satisfaga los requerimientos y necesidades de los estudiantes.

Se tuvo la colaboración de 68 estudiantes de diferentes universidades del Ecuador, para realizar la encuesta. Las preguntas con sus respectivos resultados se muestran a continuación.

Las preguntas y resultados de las encuestas fueron los siguientes:

1. **¿A qué universidad pertenece?**
   La pregunta fue contestada por 67 estudiantes, de los cuales 65 estudiantes que representan el 97.01 % son de universidades públicas y 2 estudiantes que representan el 2.98 % son de instituciones privadas. El gráfico de los resultados de la pregunta se visualizan en la Figura 2.2

2. **¿En tu universidad todos los documentos académicos se solicitan en un mismo lugar?**
   El 67.6 % afirma que no se solicitan los documentos en un mismo lugar y el 32.4 % dice que si se solicita en un mismo lugar los documentos. Por lo tanto el proceso actual para

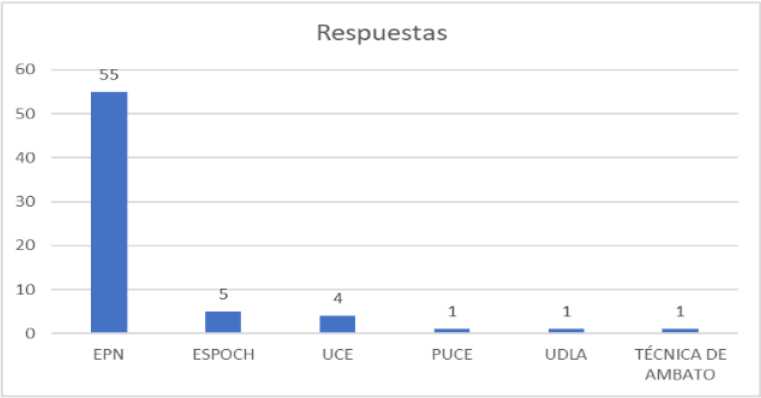


Figura 2.2: Pregunta 1. Universidades participantes

la emisión de documentos académicos no es ágil para el estudiante, de tal forma se evite estar de departamento en departamento averiguando en donde se debe obtener su documento académico. Los resultados de la pregunta se visualizan en la Figura 2.3

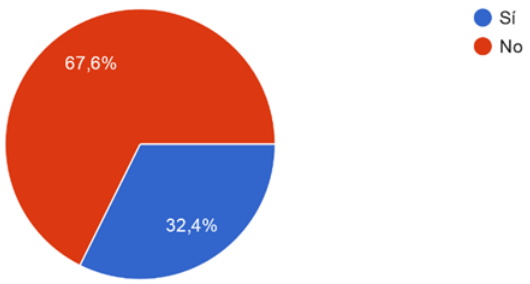


Figura 2.3: Pregunta 2. Obtención de documentos académicos

3. **¿Los documentos académicos se obtienen de manera rápida en tu universidad?**
   La mayoría de los encuestados, que representa el 80.9 % afirma que la emisión de los documentos académicos no es rápida y oportuna. Por este motivo sería un punto negativo del actual proceso de emisión documental que tienen algunas universidades, por consiguiente el presente prototipo debe tener la funcionalidad de emitir documentos académicos de manera rápida y oportuna. El gráfico de los resultados de la pregunta se visualizan en la Figura 2.4

4. **¿Cuánto tiempo tarda la entrega de un documento académico solicitado?**
   El 80 % de los encuestados afirma que la entrega de los documentos académicos tarda más de 5 días y el 16.4 % expresa que tarda de 2 a 5 días. Por lo tanto, se establece el principal problema a solucionar por el prototipo por consiguiente dicho prototipo debe estar diseñado para que automatice el proceso manual de la emisión de los documentos académicos para así reducir los tiempos de entrega. El gráfico de los resultados de la pregunta se visualizan en la Figura 2.5

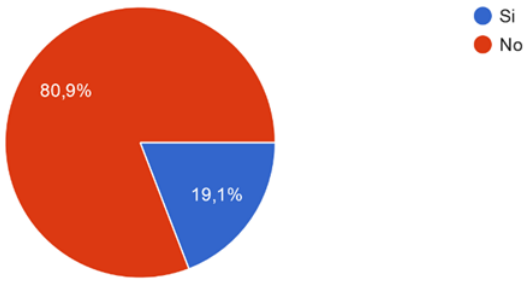


Figura 2.4: Pregunta 3. Agilidad en la entrega de documentación

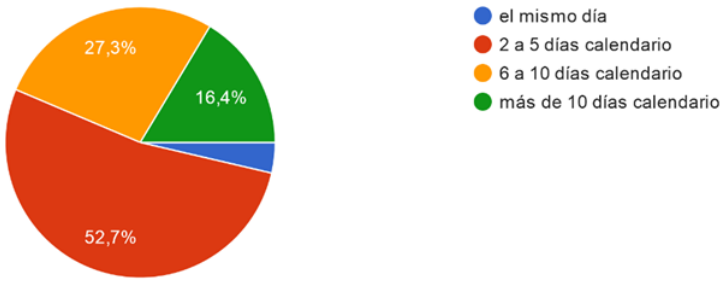


Figura 2.5: Pregunta 4. Tiempo de entrega de documentación

5. **¿El proceso para solicitar algún documento académico es:**
   Pedir el documento al funcionario encargado quién hace la gestión para conseguir las firmas y sellos requeridos, y una vez conseguidos el documento está listo?
   Gran porcentaje de los encuestados que representa el 67.3 % afirma que es correcta la hipótesis planteada en la pregunta, seguido por quienes desconocen del tema que representan el 27.3 %. Por lo tanto esta hipótesis contribuirá en el desarrollo del prototipo que va a automatizar el proceso emisión documental. El gráfico de los resultados de la pregunta se visualizan en la Figura 2.6

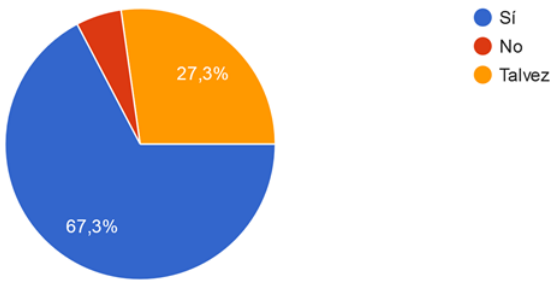


Figura 2.6: Pregunta 5. Proceso de emisión documental

6. **¿Crees que es necesario la creación de un aplicativo web el cuál permita hacer**

la solicitud de los documentos académicos en línea y que reduzca de manera significativa el tiempo de entrega de documentos?
La totalidad de los encuestados afirma que un aplicativo web para la emisión de documentos académicos es necesario, por lo tanto se concluye que el prototipo a diseñar debe suplir este requerimiento de los estudiantes. El gráfico de los resultados de la pregunta se visualizan en la Figura 2.7

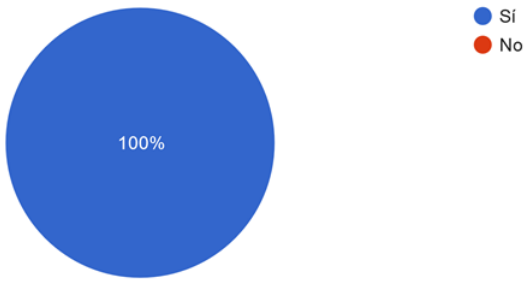


Figura 2.7: Pregunta 6. Opinión acerca de aplicativo web

7. **¿Qué tan útil calificaría si el aplicativo web generara documentos con la misma validez que los físicos?**
Casi la totalidad de los encuestados representado por el 90.9 % considera muy útil que los documentos creados por el aplicativo web tengan la misma validez que los físicos. De lo antes expresado se desprende que la mayoría de los estudiantes ven de gran utilidad esta característica para el futuro diseño del prototipo. Los resultados de la pregunta se visualizan en la Figura 2.8

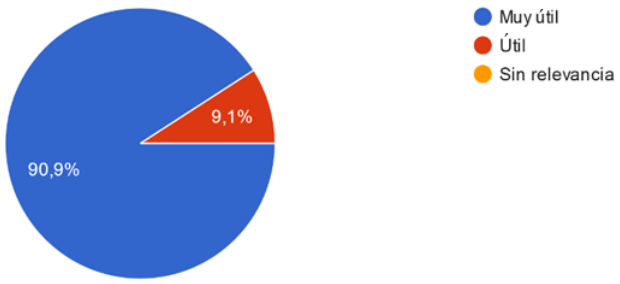


Figura 2.8: Pregunta 7. Utilidad del aplicativo web

8. **¿Considera conveniente que la aplicación web verifique la integridad de los documentos académicos?**
Considerando que la mayoría de los encuestados representados por el 81.8 % considera conveniente la característica que se plantea en la pregunta, es de vital importancia que el prototipo verifique la integridad de los documentos académicos. Es decir que el receptor tenga la seguridad de que el mismo no haya sido alterado es decir que sea

auténtico. El gráfico de los resultados de la pregunta se visualizan en la Figura 2.9

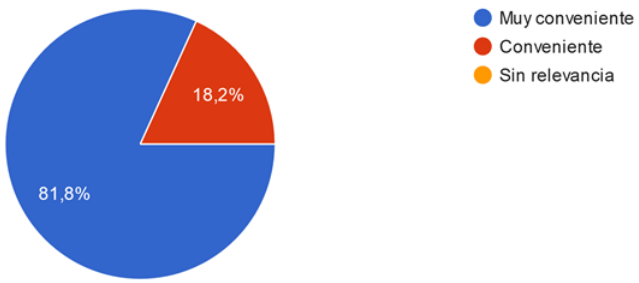


Figura 2.9: Pregunta 8. Integridad de documentos emitidos

2.1.3. REQUERIMIENTOS NO FUNCIONALES

Los requerimientos no funcionales son propiedades del sistema que van a permitir al prototipo cumplir de manera correcta todos sus servicios y funcionalidades. En base a las encuestas realizadas (ver Sección 2.1.2) se concluye los requerimientos no funcionales listados en la Tabla 2.1.

Tabla 2.1: Requerimientos no funcionales

	RF	Descripción
Presentación	RNF001	La interfaz deberá ser rápida, amigable e intuitiva. Es decir que el tiempo de carga de la página, navegación dentro de las diferentes pestañas sean veloces y que la plataforma sea fácil de usar para el usuario.
Autenticación	RNF002	El acceso al módulo de solicitud de documentos académicos debe estar solo accesible a usuarios registrados.
Rendimiento	RNF003	La entrega de documentos académicos debe ser menor a un día.
Seguridad	RNF004	Las credenciales de los usuarios y los documentos académicos no deben ser enviados en texto plano.
Escalabilidad	RNF005	El prototipo debe ser implementado en un servidor de tal manera que sea posible el mejoramiento de su hardware como son: la memoria RAM, procesador y espacio de almacenamiento.

2.1.4. REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales detallan los servicios y características que tendrá el prototipo además se describe la interacción que tendrá cada tipo de usuario con el sistema.

En base a los resultados de las encuestas realizadas (ver Sección 2.1.2) se concluyen los requerimientos funcionales listados en la Tabla 2.2. Dichos requerimientos considera dos tipos de usuarios: el estudiante, que es el solicitante del documento académico y el usuario

público, que es el receptor del documento; estos conceptos se profundizarán en la Sección 2.1.6.
Por otro lado existen dos tipos de documentos académicos que se puede descargar; uno es el entregable que tiene un pie de página con el código hash el cual solo puede ser descargado por el estudiante, y el segundo no tiene pie de página y puede ser descargado por la persona que posea un código hash asociado a un documento académico.

Tabla 2.2: Requerimientos funcionales

RF	Descripción
RF001	El prototipo permitirá al usuario tipo estudiante hacer *login* en la plataforma.
RF002	El usuario del tipo estudiante, puede solicitar un documento académico bajo las siguientes restricciones: Se puede generar cada tipo de documento una vez al día y como máximo 5 veces por semestre.
RF003	El usuario tipo estudiante a lo largo del tiempo va a generar varios documentos académicos. El prototipo le da la posibilidad de visualizar la lista de los últimos documentos académicos solicitados.
RF004	El prototipo enviará un email al usuario tipo estudiante notificando la generación exitosa del documento académico y el código hash asociado al mismo.
RF005	El usuario tipo estudiante puede descargar el documento académico entregable, para esto debe iniciar sesión e ingresar el código hash asociado al mismo.
RF006	El usuario de tipo público puede descargar un documento académico sin necesidad de hacer *login*, simplemente usando el código asociado al documento.

Basado en los requerimientos que se plantearon anteriormente, el prototipo va a estar conformado por tres módulos, los mismos que se detallan a continuación:

- **Módulo 1: Módulo de petición de documentos académicos.** Este Módulo es el que se encarga del sistema de autenticación. Muestra al usuario tipo estudiante un histórico de documentos solicitados, además realiza las validaciones para ver si el usuario tipo estudiante puede o no solicitar un documento académico.

- **Módulo 2: Módulo de generación de documentos académicos** Este módulo es el encargado de obtener los datos necesarios para la generación del documento académico, para poder generar el documento. Este envía un email al estudiante cuando el documento esté listo.

- **Módulo 3: Módulo de descarga de documentos académicos**. Permite la descarga de los documentos cuando el usuario ingresa un código hash asociado al mismo.

A continuación, se clasifica cada requerimiento funcional por módulo y se detalla una estimación de tiempo que tomará la realización de cada módulo.

El requerimiento RF001 hace referencia a todo el sistema de autenticación, basado en lo detallado en la Sección 1.4.5.2. Se plantea que este prototipo va a usar el módulo de autenticación de Laravel por ende se debe investigar como acoplar este sistema con la base de

datos a usar, debido a esto la estimación para este requerimiento fue de 8 horas.
Para la implementación del requerimiento RF002 es necesario realizar algunas validaciones en el sistema, esto implica realizar varias consultas de diferentes niveles de complejidad en la base de datos. Como conclusión la estimación de tiempo para la realización de este requerimiento es de 45 horas. Para codificar el requerimiento RF003 hace falta hacer una consulta a la base de datos y ordenar los datos en una tabla, se estima que para la realización de este requerimiento es necesario 5 horas. En la Tabla 2.3 se muestra la estimación de tiempo que tomará la realización de cada requerimiento del Módulo de petición de documentos académicos.

Tabla 2.3: Estimación de tiempo del Módulo de petición de documentos académicos.

RF	**Descripción**	**Estimación en horas**
RF001	El prototipo permitirá al usuario tipo estudiante hacer login en la plataforma	15
RF002	El usuario del tipo estudiante, puede solicitar un documento académico bajo las siguientes restricciones: Se puede generar cada tipo de documento una vez al día y como máximo 5 veces por semestre.	40
RF003	El usuario tipo estudiante a lo largo del tiempo va a generar varios documentos académicos. El prototipo le da la posibilidad de visualizar la lista de los últimos documentos académicos solicitados.	5
	Total	60

El requerimiento RF004 es la parte más importante y compleja del prototipo. Se debe codificar las plantillas necesarias para la creación de los documentos académicos. Para obtener los datos necesarios para las plantillas se deben realizar las consultas necesarias a la base de datos, y así poder generar el documento académico. Una vez listo el documento se debe enviar un correo electrónico al estudiante. Por consiguiente se concluye que para la implementación de este requerimiento, se estima que es necesario 65 horas. En la Tabla 2.4 se muestra estimación de tiempo que tomará la realización del Módulo de generación de documentos académicos.

Tabla 2.4: Estimación de tiempo del Módulo de generación de documentos académicos

RF	**Descripción**	**Estimación en horas**
RF004	El prototipo enviará un email al usuario tipo estudiante notificando la generación exitosa del documento académico y el código hash asociado al mismo.	65

El Módulo de descarga al ser implementado, el sistema permitirá descargar el documento académico tanto al usuario estudiante como público por ende, se estima que para la realización de este módulo es necesario 40 horas. La estimación de tiempo para la realización de cada requerimiento de este Módulo se muestra en la Tabla 2.5.

Tabla 2.5: Estimación de tiempo del Módulo de descarga de documentos académicos

RF	Descripción	Estimación en horas
RF005	El usuario tipo estudiante puede descargar el documento académico entregable, para esto debe iniciar sesión e ingresar el código hash asociado al documento académico.	20
RF006	El usuario de tipo público puede descargar un documento académico sin necesidad de hacer login simplemente usando el código asociado al documento académico.	20
	Total	40

2.1.5. DISEÑO DE LA CAPA DE DATOS

En esta sección se diseñará la base de datos del sistema prototipo. Dado que el sistema prototipo usará una base de datos no relacional; se realizará en esta sección el diseño del diagrama de colecciones.

Diagrama de colecciones

El diagrama de colecciones consta de dos colecciones una llamada *'users'* y la otra llamada *'documents'* como se ve en la Figura 2.10.

La colección *'users'* va a almacenar toda la información relacionada al estudiante y la información necesaria para la generación de los documentos académicos. Esta colección va a tener documentos embebidos en el campo 'Materias' es decir un array de documentos. La colección va a a tener dos índices de tipo *unique* el campo '_id' y el campo 'email'. En la Tabla 2.7 se detalla el tipo y la descripción de cada campo de la colección *'users'*.

La colección *'documents'* va a generar un nuevo *documento* cada vez que el sistema genere un nuevo documento académico. Esta colección almacenará toda la información relacionada con el documento académico generado es decir su tipo, ubicación y su respectivo código hash. Además va a tener un índice de tipo *unique* el campo '_id'. En la Tabla 2.6 se detalla el tipo y la descripción de cada campo de la colección *'documents'*.

Para la realización de este diagrama de colecciones se utilizó el software *DbSchema* [27], el cual es un gestor y diseñador de base de datos, entre una de sus características es la ingeniería inversa que consiste en cargar la base de datos y como resultado devuelve el diagrama de colecciones.

2.1.6. DISEÑO DE LA CAPA DE NEGOCIO

La presente sección mostrará el diseño de casos de uso y el diagrama de clases del prototipo.

a) **Diagrama de casos de uso**

El prototipo a desarrollar posee dos tipos de usuarios: estudiante y público. A conti-

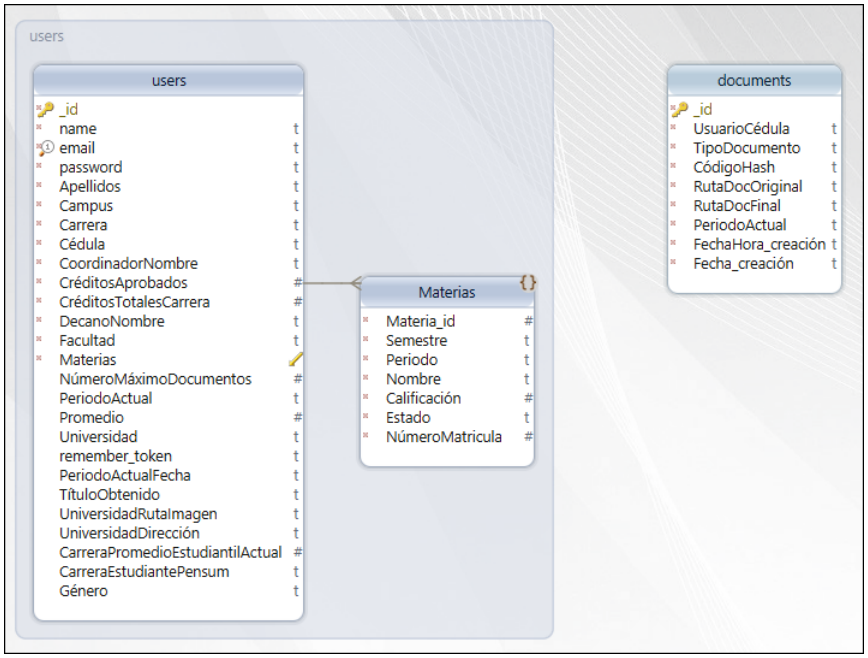


Figura 2.10: Diagrama de colecciones para la base de datos

Tabla 2.6: Colección ”documents”

Colección documents		
Campo	**Tipo**	**Descripción**
_id	oid	Identificador único de cada documento de la colección.
UsuarioCédula	string	Cédula del estudiante.
TipoDocumento	string	Tipo de documento académico.
CódigoHash	string	Código hash del documento académico.
RutaDocOriginal	string	Ruta del documento académico.
FechaHora_creación	string	Fecha y hora de creación del documento.
Fecha_creación	string	Fecha de creación del documento.

nuación, se detalla la definición de cada tipo de usuario y la manera en como opera con el prototipo.

- **Usuario tipo estudiante:** Este tipo de usuario son todos los estudiantes en general, los cuales tendrán un usuario y contraseña para poder iniciar sesión en el prototipo. Además, son los únicos usuarios que podrán generar los documentos académicos los mismos que tendrán su respectivo código hash como leyenda en la parte inferior del documento.
- **Usuario tipo público:** Es todo aquel usuario que posee en sus manos algún documento académico generado por el prototipo, por ende, posee también el código hash correspondiente al documento académico ya que este viene en el pie de página del documento académico entregable. Este tipo de usuario puede ser el receptor del documento, es decir, a la persona que el estudiante entrega el documento académico o el mismo estudiante. Este tipo de usuario puede des-

Tabla 2.7: Colección “users”

Campo	Tipo	Descripción
_id	oid	Identificador único de un estudiante.
name	string	Nombres del estudiante.
email	string	Correo electrónico del estudiante.
password	string	Contraseña del sistema.
Apellidos	string	Apellidos del estudiante.
Campus	string	Dirección del campus.
Carrera	string	Carrera del estudiante.
Cédula	string	Cédula del estudiante.
CoordinadorNombre	string	Nombre del coordinador de la carrera.
CréditosAprobados	double	Créditos aprobados por el estudinate.
CréditosTotalesCarrera	double	Créditos totales de la carrera.
DecanoNombre	string	Nombre del Decano.
Facultad	string	Nombre de la Facultad.
Materias	list	Almacenará documentos embebidos.
Materias.Materia_id	double	Identificador único de una materia.
Materias.Semestre	string	Semestre al que pertenece la materia.
Materias.Periodo	string	Periodo al que pertenece la materia.
Materias.Nombre	string	Nombre de la materia.
Materias.Calificación	double	Calificación de la materia
Materias.Estado	string	Estado de la materia
Materias.NúmeroMatricula	double	Número de matrícula de la materia.
NúmeroMáximoDocumentos	double	Número máximo de documentos que se puede generar por periodo.
PeriodoActual	string	Periodo actual en el que se encuentra el estudiante.
Promedio	double	Promedio en la carrera del estudiante.
Universidad	string	Nombre de la Universidad.
remember_token	string	Token que sirve para recuperar la contraseña.
PeriodoActualFecha	string	La fecha de inicio y fin del periodo.
TítuloObtenido	string	Título obtenido en la carrera.
UniversidadRutaImagen	string	Imagen de la universidad.
UniversidadDirección	string	Dirección de la universidad.
CarreraPromedioEstudiantilActual	double	Promedio general de los estudiantes en la carrera.
CarreraEstudiantePensum	string	Nombre del pensum académico.
Género	string	Género del estudiante.

cargarse el documento académico original desde el prototipo sin la necesidad de iniciar sesión, de modo que ingresa a la plataforma web y digita el código hash del documento a descargar.

La Figura 2.11 muestra los casos de uso que son necesarios para el funcionamiento total del Prototipo para la Emisión y Verificación de la Integridad de Documentos Académicos. El diagrama se lo puede resumir de la siguiente manera:

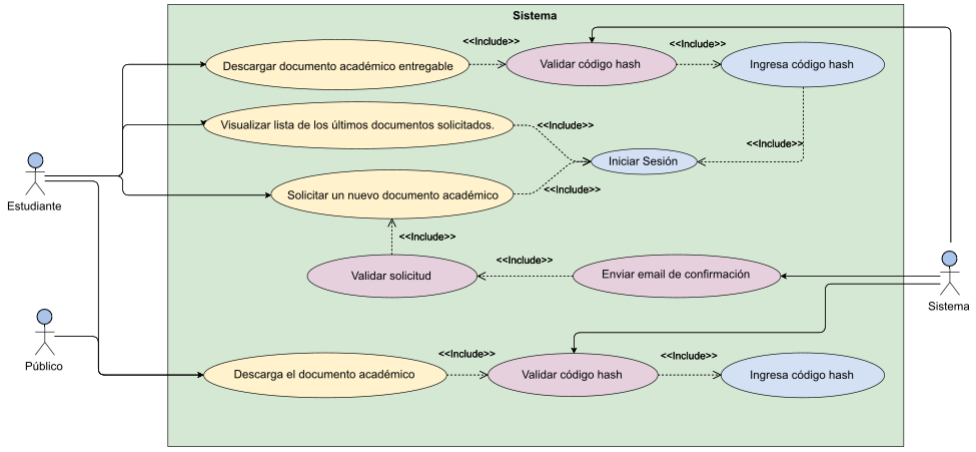


Figura 2.11: Casos de usos del sistema.

- El usuario de tipo estudiante es el único que puede hacer la solicitud de un documento, previamente debe iniciar sesión con su respectivo nombre de usuario y contraseña. Una vez procesada la solicitud el prototipo enviará un correo electrónico al usuario para notificarle que el documento se generó exitosamente. Seguidamente, hará login en el prototipo, ingresará el código hash asociado al documento, el sistema validará el mismo y finalmente el estudiante podrá descargar el documento académico entregable.

- El usuario de tipo público que puede ser el receptor del documento académico o el estudiante puede descargarlo simplemente ingresando a la url del prototipo y digitar el código hash asociado al documento académico a descargar.

- Además, el usuario de tipo estudiante luego de iniciar sesión puede visualizar la lista de los últimos documentos académicos generados.

Casos de uso de cada requerimiento funcional.

Para iniciar sesión es necesario que el sistema previamente valide el usuario y contraseña. Los casos de uso necesarios para realizar esta acción se muestra en la Figura 2.12.
Para que el usuario tipo estudiante solicite un documento académico debe iniciar sesión caso contrario no va a poder realizar la solicitud. La Figura 2.13 muestra los casos de uso que son necesarios para solicitar un nuevo documento académico.

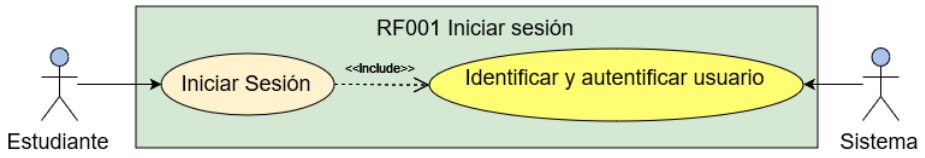


Figura 2.12: Caso de uso para iniciar sesión.

Figura 2.13: Caso de uso para solicitar un documento.

Un usuario tipo estudiante solamente puede visualizar los últimos documentos solicitados una vez que ingrese al sistema, como se muestra en la Figura 2.14.

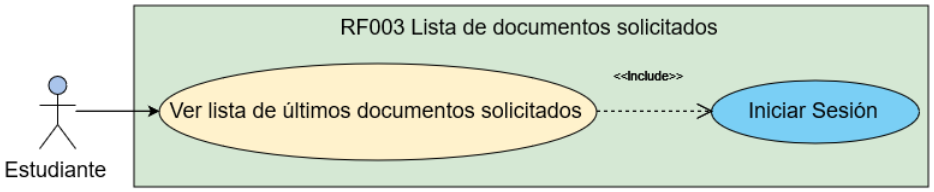


Figura 2.14: Caso de uso para visualizar la lista de últimos documentos generados.

Para que el sistema envíe un email de confirmación, el usuario tipo estudiante debe haber iniciado sesión en el sistema y haber solicitado un documento para que el sistema pueda validar la solicitud y ver si debe o no enviar el email. Los casos de uso asociados al requerimiento RF004 se muestra en la Figura 2.15.

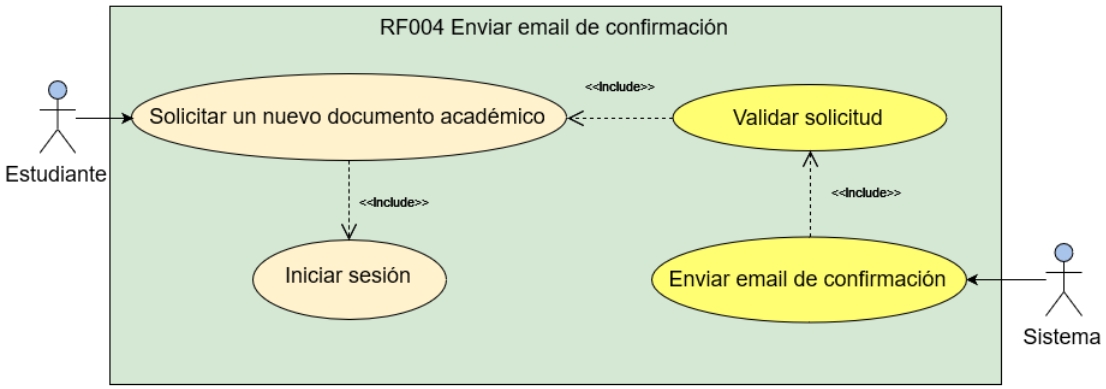


Figura 2.15: Caso de uso para enviar email de confirmación.

Para que el estudiante pueda descargar el documento académico entregable debe iniciar sesión en el sistema, luego debe ingresar el código hash para que el sistema valide y apruebe la descarga del documento. La Figura 2.16 muestra los casos de uso que son necesarios para poder descargar el documento académico entregable.

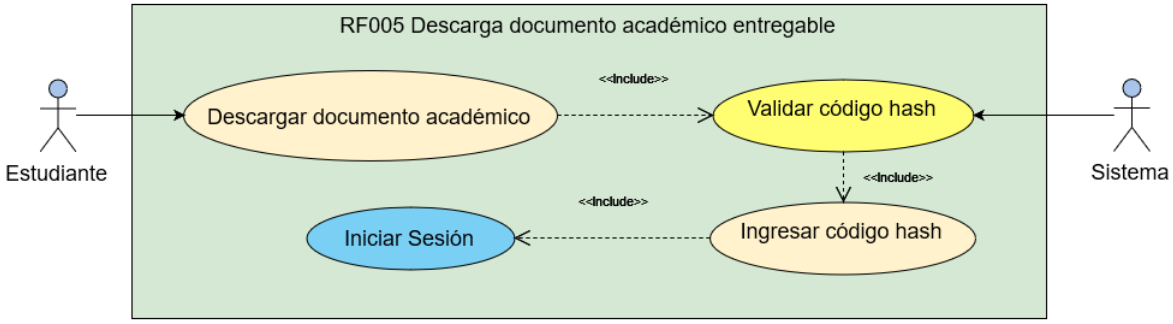


Figura 2.16: Caso de uso para descargar un documento entregable.

Para que un usuario tipo público pueda descargar un documento académico solo debe ingresar el código hash asociado a un documento, para que el sistema pueda validarlo y de ser correcto permitir la descarga. La Figura 2.17 muestra los casos de uso que son necesarios para el requerimiento RF006.

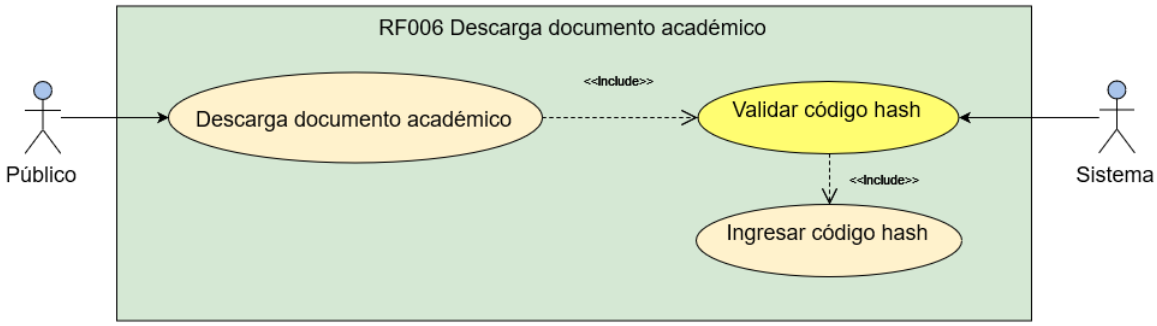


Figura 2.17: Caso de uso para descargar un documento.

b) **Diagrama de clases**

Como se ve en la Sección 1.4.5.2 para la realización de este prototipo se va a utilizar Laravel, el cual es un *framework* que se basa en una arquitectura modelo vista controlador. Una aplicación hecha en Laravel a grandes rasgos se divide en tres secciones:

- **Modelo:** Se encarga de las peticiones a la base de datos, es decir la que realiza la comunicación con la base de datos.
- **Vista:** Muestra visualmente mediante interfaces gráficas la información del prototipo. Generalmente son archivos html.
- **Controlador:** Es el intermediario entre el modelo y la vista, se encarga de la parte lógica de la aplicación recibe las órdenes del usuario, se encarga de solicitar los

datos al modelo, procesa dicha información y la envía a la vista.

De lo expresado anteriormente se desprende que para el diseño de clases se debe crear tanto modelos para la comunicación con la base de datos y controladores que van a llevar la lógica de la aplicación.

Controladores: Para el prototipo se van a crear dos controladores o clases los cuales se menciona a continuación:

- ***'UsuarioEstudiante':*** En esta clase estará toda la lógica referente al usuario tipo estudiante. Va a tener tres métodos:
  - *ListarDocumentos()*: tiene la lógica para mostrar una pantalla en donde el usuario elija el documento académico que desea solicitar. Básicamente lo que hace este método es redireccionar a la vista que contiene el listado de documentos que se puede solicitar.
  - *GenerarPDF()*: tiene la lógica para la generación de documentos académicos. Tiene el atributo *request* que es donde se almacena el tipo de documento que el usuario desea generar. Este método primero valida si el usuario seleccionó algún documento, luego crea una carpeta con el nombre de usuario en el servidor. Después valida si el estudiante posee datos suficientes para generar el documento académico. Luego de pasar todas las validaciones se procede a generar el pdf.
  - *UltimosDocumentos()*: realiza una consulta a la base de datos, para averiguar los últimos documentos académicos generados por el usuario, dicha información es enviada a una vista para que el usuario pueda visualizarla.
- ***'UsuarioPublico':*** En esta clase esta toda la lógica relacionada al usuario tipo público. Tiene el método *Descargarpdf()* dicho método tiene toda la lógica referente a la descarga de un documento académico. Este método tiene el atributo *request* que es donde se almacena el tipo de documento que el usuario desea generar.

Modelos: El prototipo va a tener dos modelos o clases que van a heredar los métodos y clases de la clase *'Moloquent'* que es una clase del Paquete Laravel MongoDB el cual permite la comunicación con MongoDB, por ende esta clase es muy extensa y el diagrama solo se mostrará una síntesis de la misma. Los modelos en Laravel basta que tengan el mismo nombre de la colección para que Laravel haga la comunicación con dicha colección. Los modelos a crear son los siguientes:

- ***'User':*** Encargado de la comunicación con la colección ***'users'***.
- ***'Document'*** Encargado de la comunicación con la colección ***'documents'***.

Para el presente prototipo se va a usar el módulo de autenticación [28] de este *framework* el cual proporciona todas las clases y componentes necesarios para que el sistema tenga un módulo de autenticación completo. Cabe mencionar que Laravel al ser un framework de php cuenta con una extensa cantidad de clases y componentes por consiguiente este diagrama de clases solamente se limita a mostrar las clases principales. Las clases que son parte del módulo de autenticación son *'ForgotPasswordController', 'LoginController', 'ResetPasswordControlles', 'VerificationController'* .

En la Figura 2.18 se visualiza el diagrama de clases de la parte lógica de la capa de negocios. El diagrama se realizó con el software *'Día'* que sirve para dibujar diagramas de estructura. [29]

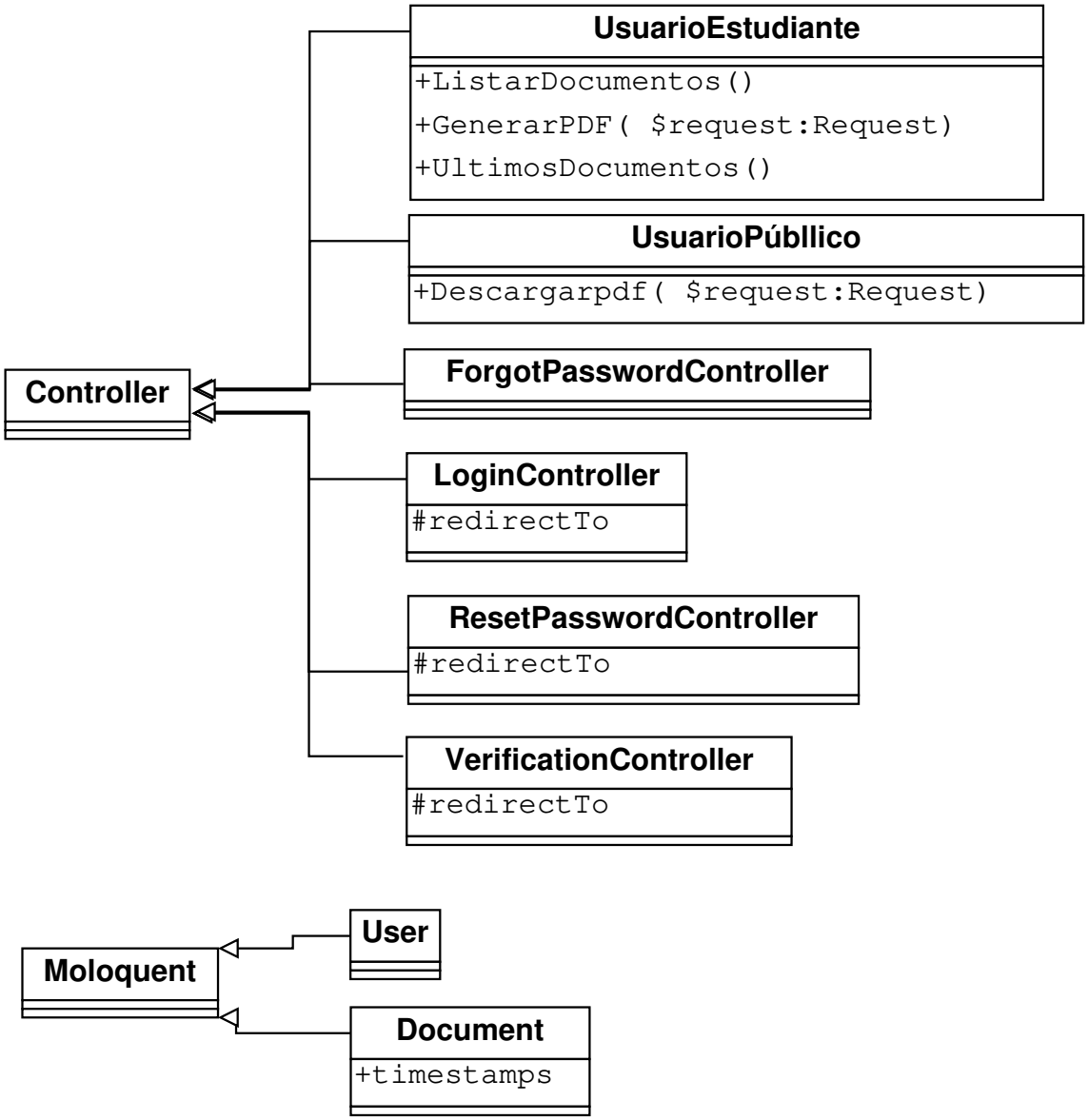


Figura 2.18: Diagrama de clases del prototipo.

2.1.7. DISEÑO DE LA CAPA DE PRESENTACIÓN

En esta sección se diseñan cada interfaz gráfica del sistema prototipo las cuales tienen una interacción directa con el usuario final. Todas estas interfaces se diseñaron usando la herramienta en línea Moqups, la cual sirve para la creación de maquetas, diagramas y prototipos para el diseño web. [30]

Diseño de las vistas del sistema prototipo

A continuación se presenta el diseño de cada pantalla que conformará el sistema prototipo para la emisión y verificación de documentos académicos.

La Figura 2.19 muestra la pantalla de inicio del prototipo esta pantalla visualizará el usuario cada que ingrese al aplicativo web. Está confomada por:

- **Botón de login:** Permite iniciar sesión en el sistema.
- **Caja de texto:** Es donde el usuario ingresa el código hash asociado a un documento académico.
- **Botón descargar:** Si se ingresa un código en la caja de texto permite la descarga de un docuemnto académico.

Figura 2.19: Interfaz de la página de incio.

La Figura 2.20 muestra la pantalla de inicio de sesión en donde el usuario debe ingresar sus respectivas credenciales para poder ingresar al sistema. Esta pantalla tiene la opción de recuperar contraseña.

La Figura 2.21 muestra la pantalla que se le despliega al usuario luego de iniciar sesión, esta pantalla muestra en un grid los últimos documentos académicos solicitados por el usuario. También tiene una barra de navegación esta va a estar presente en todas las pantallas menos el la pantalla de inicio (Ver Figura 2.19) y tiene tres botones:

Figura 2.20: Interfaz para el inicio de sesión.

- **Solicitar documentos:** Permite abrir la pantalla para la solicitud de documentos académicos.
- **Listado de documentos:** Abre la pantalla en donde se visualizan los últimos documentos académicos generados.
- **Nombre de Usuario:** El texto de este botón va acorde al usuario que inició sesión, permitiendo al usuario cerrar sesión.

Figura 2.21: Interfaz del listado de documentos generados.

La pantalla que permite solicitar documentos académicos, contiene la lista de todos los documentos que el sistema prototipo puede generar. En esta pantalla el usuario selecciona el documento que desea solicitar. La implementación de la misma se muestra en la Figura 2.22.

Figura 2.22: Interfaz para la petición de documentos académicos.

Luego que el usuario solicita el documento se muestra una pantalla mostrando el mensaje de que la solicitud fue generada exitosamente. La implementación de esta pantalla se muestra en la Figura 2.23.

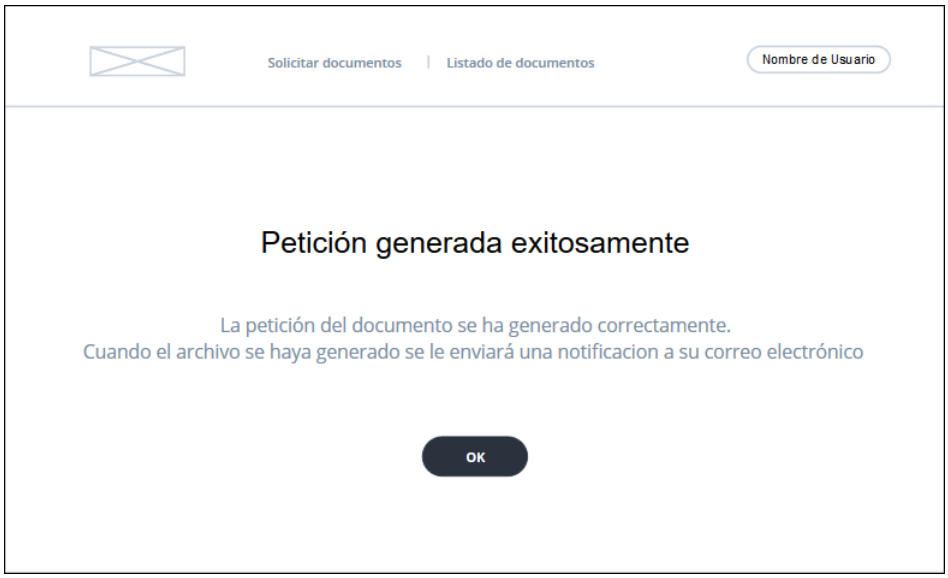


Figura 2.23: Interfaz para el mensaje de confirmación.

2.2. IMPLEMENTACIÓN

En esta sección se muestra todos los procesos y configuraciones que se deben realizar para la implementación del sistema prototipo. Primero se muestra la actualización del tablero Kanban luego se procede con la instalación de las herramientas y tecnologías necesarias para la implementación del sistema prototipo. Para finalmente presentar la codificación del

mismo.

2.2.1. ACTUALIZACIÓN DEL TABLERO KANBAN

A continuación se presenta la actualización del tablero Kanban donde, las tareas relacionadas a la etapa de diseño pasan a la columna “Tareas finalizadas” y a la columna de “Tareas en curso” pasan todas las actividades relacionadas a la etapa de implementación, como se muestra en la Figura 2.24.

Figura 2.24: Tablero Kanban etapa de implementación.

2.2.2. INSTALACIÓN DE LAS HERRAMIENTAS NECESARIAS

En esta sección se detalla todo el software y servicios necesarios en la implementación del sistema prototipo para la emisión y verificación de la integridad de documentos académicos.

2.2.2.1. Creación y configuración de una instancia EC2 en AWS

Para la creación de nuestro servidor se escogió el servicio web Amazon EC2 o Amazon Elastic Compute Cloud de Amazon Web Service el cual permite la creación de un servidor de una manera rápida y sencilla. Dicho servicio web ofrece dos planes uno gratuito y otro de pago. El plan de pago ofrece una gran cantidad de características de hardware por ejemplo: Para el procesador se puede elegir desde el más sencillo hasta un Intel Xeon con GPU de NVIDIA. El cobro de este plan se hace en función del tiempo que esta activa la instancia y

de las características del servidor como son: tamaño de memoria *ram*, espacio de almacenamiento entre otros [19].
Para el presente prototipo se escogió el plan gratuito, el cual nos ofrece una instancia del tipo *t2.micro* por un año; las características se muestran en la Tabla 2.8.

Tabla 2.8: Características de la instancia *t2.micro* [31].

Característica	**Capacidad**
Horas de uso por mes	750 horas.
Almacenamiento	30 GB.
CPU virtual	Intel AVX†, Intel Turbo†, 1 núcleo.
Memoria ramm	1 GB.
Sistema operativo	Linux o Windows.

La menor versión de *Windows Server* que ofrece *EC2* es *Windows server 2012 R2*, los requerimientos de hardware para dicho sistema operativo según documentación oficial de *Windows* es 2GB de memoria *ramm* y 60GB de espacio en disco para el sistema [32], por otro lado *Linux* se caracteriza por rendir de mejor manera en equipos de bajos recursos en consecuencia para este prototipo se escogió Ubuntu Server 18.04 LTS.
Para la creación la instancia *t2.micro* se lo hace desde la consola de Amazon Web Service, se escoje el servicio EC2 y se selecciona *Launch Instance*, luego se procede a realizar las siguientes configuraciones:

1. **Sistema Operativo:** Se configura el sistema operativo *Ubuntu Server 18.04 LTS* para la instancia del prototipo. (ver Figura 2.25)

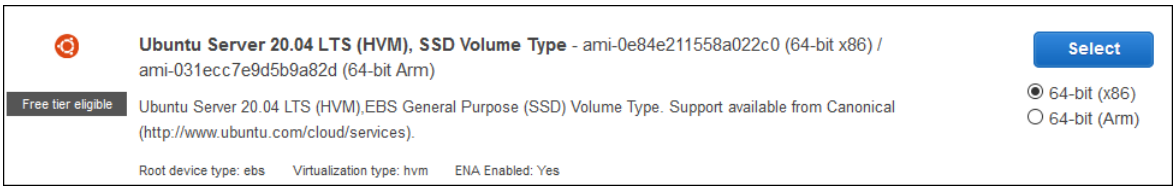
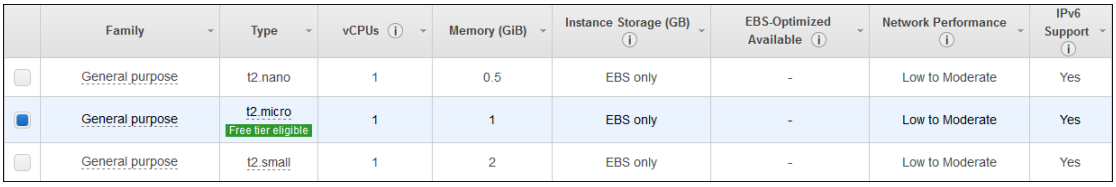


Figura 2.25: Configuración del sistema operativo de la instancia.

2. **Tipo de instancia:** Como se mencionó anteriormente el prototipo usará una instancia tipo *t2.micro* en su versión gratis.(ver Figura 2.26)



	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
☐	General purpose	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
☑	General purpose	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
☐	General purpose	t2.small	1	2	EBS only	-	Low to Moderate	Yes

Figura 2.26: Configuración del tipo de instancia.

3. **Almacenamiento:** La instancia t2.micro en su versión gratis nos proporciona 30GB de almacenamiento, si se requiere un valor mayor a este tendrá ya un costo. Por consiguiente se configura con 30GB de almacenamiento.(ver Figura 2.27)

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-059723d81654148d6	30	General Purpose SSD (gp2)	100 / 3000	N/A	☑	Not Encrypte

Figura 2.27: Configuración del almacenamiento de la instancia.

4. **Grupo de seguridad:** El grupo de seguridad son un conjunto de reglas del *firewall* las cuales permiten controlar el tráfico de la instancia. Para este prototipo se va a permitir la conexión de cualquier dirección IP para los protocolos HTTP y HTTPS ya que la comunicación entre el prototipo y el usuario va a manejar dichos protocolos. Además se restringe la conexión entrante del protocolo SSH a una única dirección la cual es del administrador para así evitar ataques al servidor.(ver Figura 2.28)

Type	Protocol	Port Range	Source	
SSH	TCP	22	My IP	[illegible]
HTTP	TCP	80	Anywhere	0.0.0.0/0, ::/0
HTTPS	TCP	443	Anywhere	0.0.0.0/0, ::/0

Figura 2.28: Configuración del Grupo de Seguridad de la instancia .

Realizadas todas las configuraciones, *EC2* genera un archivo de llave privada que permite conectarse a la instancia creada mediante el protocolo SSH para esto se usará el terminal de GIT [23]. Como configuración final se le asignó el nombre 'Sisvda' a la intancia creada.

Configuración del dominio

El dominio utilizado en el sistema prototipo para la emisión y verificación de documentos académicos es 'sisvda.ml', mismo que se adquirió de manera gratuita en la plataforma web *Freenom* [33] como observa en la Figura 2.29, esta plataforma ofrece a su vez dominios de pago.

Los dominios gratuitos que ofrece la plataforma terminan en .tk, .ml, .ga, .cf, .gq.

Dominio	Fecha de Registro	Fecha de caducidad	Estado
sisvda.ml	2020-05-25	2021-05-25	ACTIVE

Figura 2.29: Dominio del sistema prototipo.

Luego para asociar este dominio con la instancia 'Sisvda' se debe realizar el siguiente proceso:

- Desde la consola de AWS seleccionar el servicio Route53 y crear una zona hospedada pública, esto va a permitir direccionar el tráfico del dominio del prototipo como se desee, esto se indica en la Figura 2.30. Al crear una zona hospedada se va a proporcionar los servidores de nombres, los cuales van a permitir resolver el nombre del dominio del prototipo. Estos deben ser ingresados en la plataforma de Freenom como se ve en la Figura 2.31.

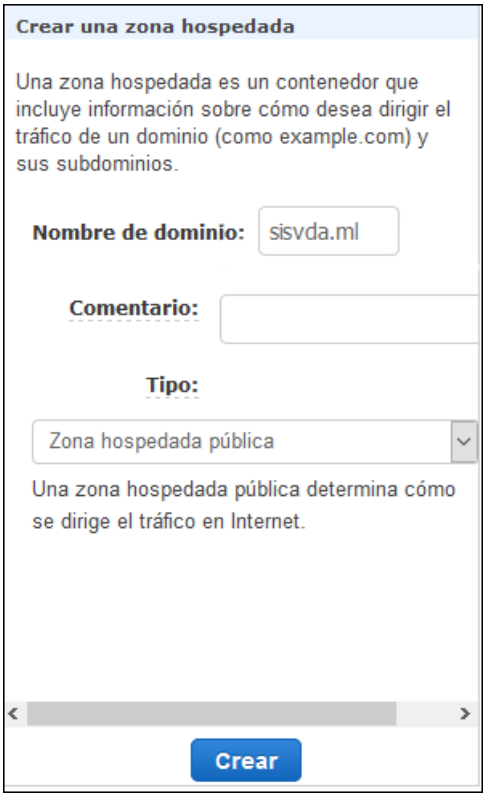


Figura 2.30: Crear zona hospedada pública.

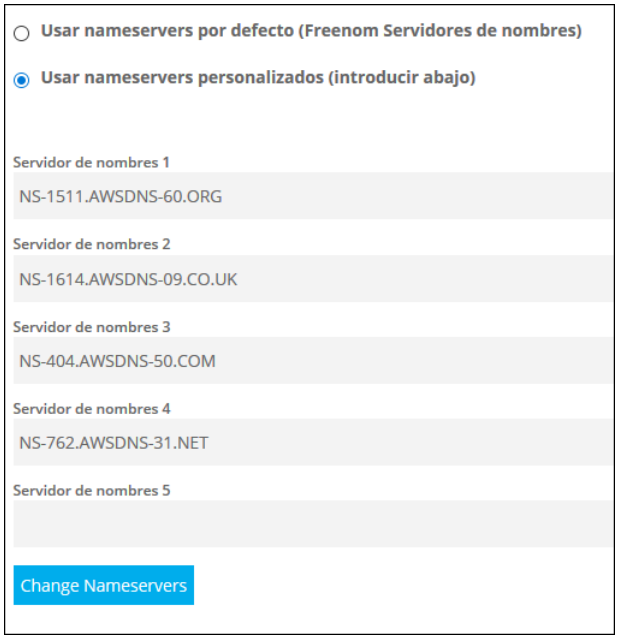


Figura 2.31: Configuración de los servidores de nombres en el dominio.

- En este paso se procede con la generación de un certificado SSL para el prototipo. Para lo cual se va a usar el servicio de AWS Certificate Manager. Para la solicitud de un certificado SSL se debe ingresar a la consola de AWS y seleccionar el servicio *Certificate Manager* y el botón 'solicitar un certificado'. Es importante señalar que

este servicio permite la solicitud de un certificado SSL público de manera gratuita. Lo siguiente es configurar los nombres de dominio del sitio al que se desea proteger con un certificado SSL como muestra la Figura 2.32. Para que AWS Certificate Manager valide la solicitud de certificado necesita verificar que el solicitante es el propietario de los dominios. La validación se hace modificando la configuración DNS de los dominios, si el dominio esta en Amazon Route 53 la configuración se la puede hacer simplemente seleccionando el botón de 'Crear registro en Route53' como se ve en la Figura 2.33. Finalizada la validación se tendrá un certificado SSL activo.

Figura 2.32: Configuración de dominios en AWS Certificate Manager.

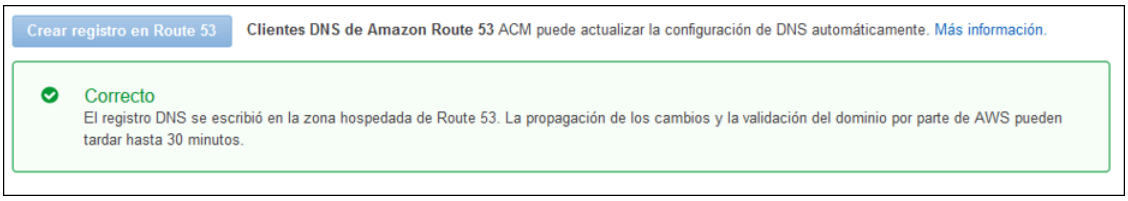


Figura 2.33: Validación de AWS Certificate Manager.

- Para finalizar, dentro de la configuración de la instancia 'Sisvda' se debe crear un balanceador de cargas para vincular el dominio con la instancia. Al crear este balanceador de cargas se debe seleccionar el certificado que se solicitó en el paso anterior y la instancia a la que se quiere asociar, estas configuraciones se muestra en la Figura 2.34. Para direccionar del dominio 'sisvda.ml' al balanceador de cargas se debe crear el conjunto de registros en Route53 mismo que se muestra en la Figura 2.35. Por lo tanto el direccionamiento en Route53 quedaría de la siguiente manera: el dominio 'sisvda.ml' va a direccionar al balanceador de cargas y este a su vez va apuntar a la instancia 'sisvda'.

2.2.2.2. Instalación del servidor Apache

El servidor web HTTP usado para la implementación de este prototipo es Apache. En la instancia creada en la Sección 2.2.2.1 se procede a instalar Apache como se observa en el segmento de código 2.1. En la línea 1 se hace una búsqueda de los paquetes disponibles y la línea 2 se los instala para así tener los paquetes del sistema operativo actualizados. En la línea 3 se instala el servidor HTTP Apache.

Segmento de código 2.1: Comandos para instalar Apache.

```
1 sudo apt-get update
2 sudo apt-get upgrade
```

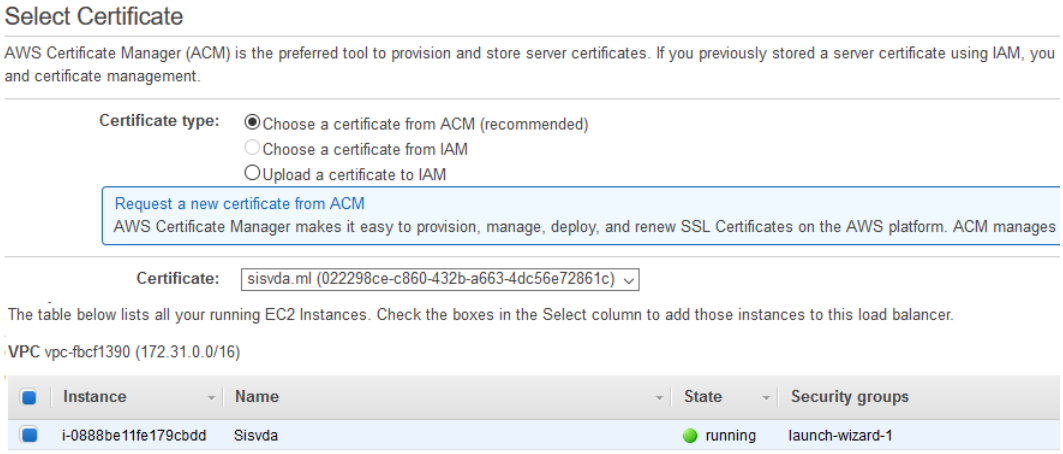


Figura 2.34: Configuraciones del balanceador de carga.

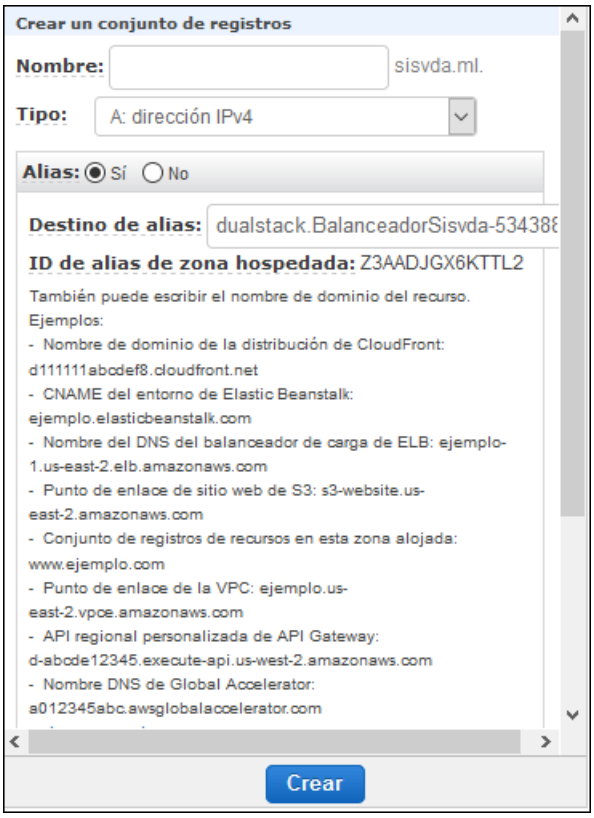


Figura 2.35: Direccionamiento de el dominio al balanceador de carga.

```
sudo apt-get install apache2
```

Adicionalmente se configura el modo *rewrite* en Apache, el que permite convertir las *urls* a *urls* amigables, este módulo es requerido por Laravel. En consecuencia se procede con su activación como muestra en el segmento de código 2.2.

Segmento de código 2.2: Comandos para activar del modo rewrite en Apache.

```
sudo a2enmod rewrite
```

```
2 sudo service apache2 restart
```

2.2.2.3. Instalación de Laravel

Para que funcione Laravel previamente se debe instalar php y algunos paquetes adicionales de php. El código para la instalación de php se muestra en el segmento de código 2.3; en la línea 2 se reinicia el servidor, en la línea 3 se verifica el estado del servidor Apache con el fin de verificar que no existe ningún error a causa de la instalación y de verificar que el servidor esté activo.

Segmento de código 2.3: Comandos para instalar PHP.

```
1 sudo apt-get install php7.2
2 sudo service apache2 restart
```

Luego se procede con la instalación de algunos paquetes necesarios para el correcto funcionamiento de la aplicación de Laravel, como se muestra en el segmento de código 2.4.

Segmento de código 2.4: Comandos para instalar paquetes PHP.

```
1 sudo apt-get install libapache2-mod-php
2 sudo apt-get install php7.2-xml
3 sudo apt-get install unzip
4 sudo apt-get install php7.2-zip
5 sudo apt-get install php7.2-mysql
6 sudo apt-get install php7.2-gd
7 sudo apt-get install php7.2-bcmath
8 sudo apt-get install php7.2-mbstring
9 sudo apt-get install php7.2-curl
10 sudo apt-get install php7.2-dev
11 sudo service apache2 restart
12 sudo service apache2 status
```

En la línea 10 del segmento de código 2.4 se reinicia el servidor para efectuar los cambios realizados y la línea 11 verifica que el servidor se encuentre activo.
Una vez instalado php y sus paquetes se procede a instalar el administrador de dependencias para php Composer [22] ejecutando las líneas de código mostradas en el segmento de código 2.5. Este administrador de dependencias es imprescindible para la creación de una aplicación de Laravel.

Segmento de código 2.5: Comandos para instalar Composer.

```
1 php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
2 php -r "if (hash_file('sha384', 'composer-setup.php') === ...
    'e0012edf3e80b6978849f5eff0d4b4e4c79ff160
```

```
3 9dd1e613307e16318854d24ae64f26d17af3ef0bf7cfb710ca74755a') { echo 'Installer ...
    verified'; } else { echo 'Installer corrupt'; ...
    unlink('composer-setup.php'); } echo PHP_EOL;"
4 php composer-setup.php
5 php -r "unlink('composer-setup.php');"
```

2.2.2.4. Instalación de Git

Para pasar el código fuente desde la máquina de desarrollo, al servidor de producción se va a usar *Git* [23], juntamente con *Github* de ahí que se procede a la instalación de *Git* con los comandos que se muestran en el segmento de código 2.6.

Segmento de código 2.6: Comandos para instalar Git.

```
1 add-apt-repository ppa:git-core/ppa
2 apt update; apt install git
```

2.2.2.5. Instalación de MongoDB

El presente prototipo usa como base de datos MongoDB. Para instalar en nuestra instancia de *EC2* esta base de datos se debe ejecutar la línea 1 del segmento de código 2.7, para activar el proceso de MongoDB se debe ejecutar la línea 2 respectivamente.

Segmento de código 2.7: Comandos para instalar de MongoDB.

```
1 sudo apt-get install mongodb
2 sudo service mongodb start
```

Una vez instalado MongoDB se procede a instalar el controlador MongoDB PHP Driver [18] necesario para que Laravel pueda trabajar junto con MongoDB. Para esto primero se ejecutan los comandos del segmento de código 2.8, donde en la línea 1 se habilita el repositorio PECL el cuál es necesario para la instalación de este controlador y en la línea 2 se instala el driver.

Segmento de código 2.8: Comandos para instalar MongoDB PHP Driver.

```
1 sudo apt install php-pear -y
2 sudo pecl install -f mongodb-1.5.3
```

Finalmente para terminar la instalación se debe añadir en el archivo php.ini del servidor la línea del segmento de código 2.9.

Segmento de código 2.9: Edición del archivo php.ini.

```
1 extension=mongodb.so
```

2.2.2.6. Creación de correo electrónico

El sistema prototipo va ha enviar notificaciones a los estudiantes vía correo electrónico cuando el documento académico esté listo, para lo cual el sistema prototipo necesita de un servicio de envío de correos electrónicos. Con este fin el sistema prototipo va a usar *Gmail*, de ahí que se creó la cuenta 'soportesistemadeverificacion@gmail.com'.
A fin de que el prototipo pueda usar esta cuenta para el envío de correos electrónicos se debe crear una contraseña para la aplicación desde las configuraciones de la cuenta de *Google* como muestra la Figura 2.36.

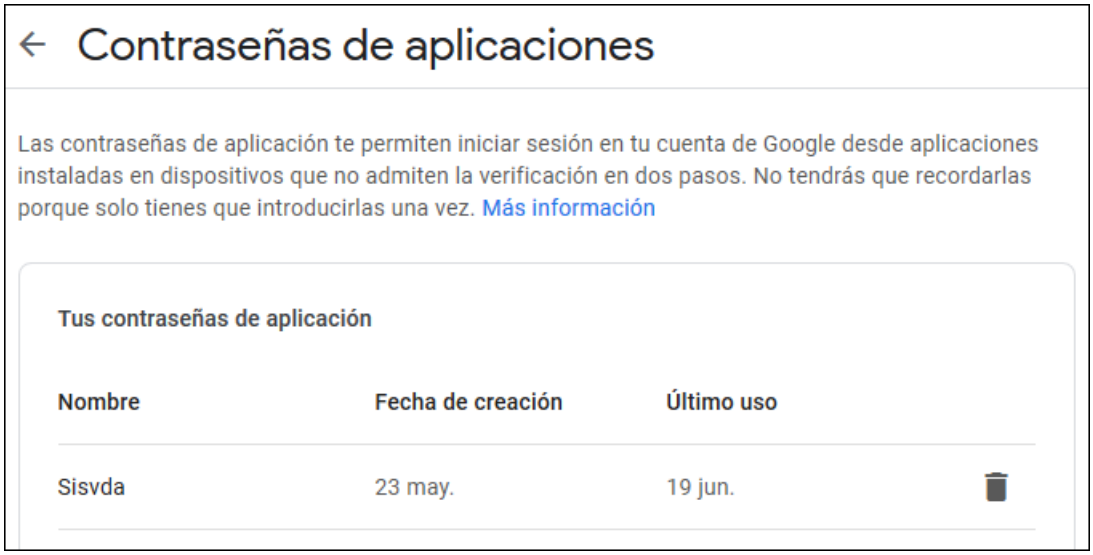


Figura 2.36: Contraseña de la cuenta de Gmail para el prototipo.

2.2.2.7. Obtención de las claves de Google reCAPTCHA

Para que el sistema prototipo pueda hacer uso de *Google reCAPTCHA* necesita tener dos claves, por consiguiente se procede desde la consola de *Google reCAPTCHA* [34] a registrar un nuevo sitio, donde se coloca el nombre del sitio y sus dominios como se muestra en la Figura 2.37. Esto genera dos llaves que se indican en la Figura 2.38.

2.2.2.8. Instalación de Visual Studio Code

El editor de código a usar para la codificación del prototipo es Visual Studio Code. Se procede a instalar el editor de código en el ambiente de desarrollo, el instalador se encuentra en la página oficial de Visual Studio Code [35]. Con el fin de usar buenas prácticas, antes de subir la aplicación web o un cambio de la misma a producción se realizarán pruebas localmente para luego mediante *Git* y *Github* poner en producción la aplicación.

2.2.3. IMPLEMENTACIÓN DE LA CAPA DE DATOS

En esta sección, se mostrará el proceso que se llevó a cabo para la codificación de la Capa de Datos basado en el diseño de colecciones mostrado en la Sección 2.1.5.

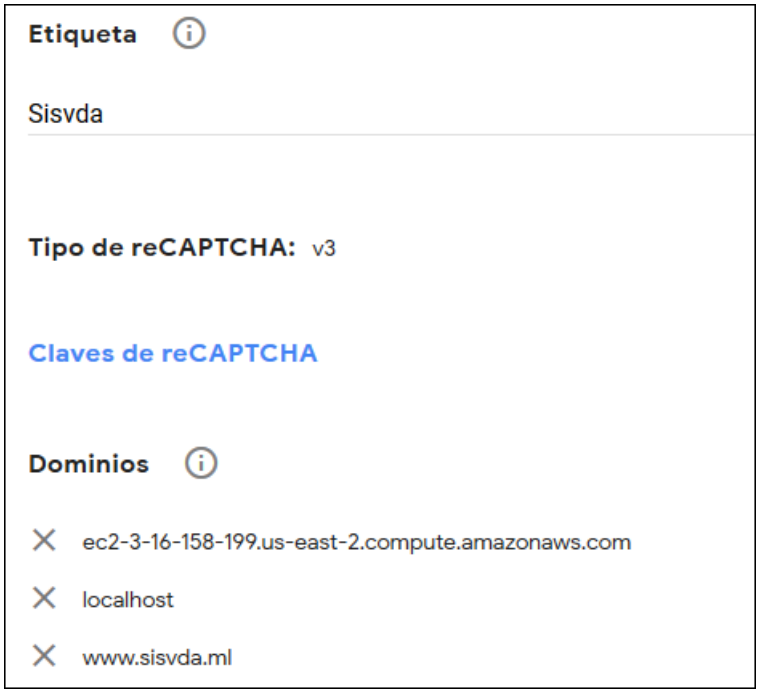


Figura 2.37: Configuración de Google reCAPTCHA.

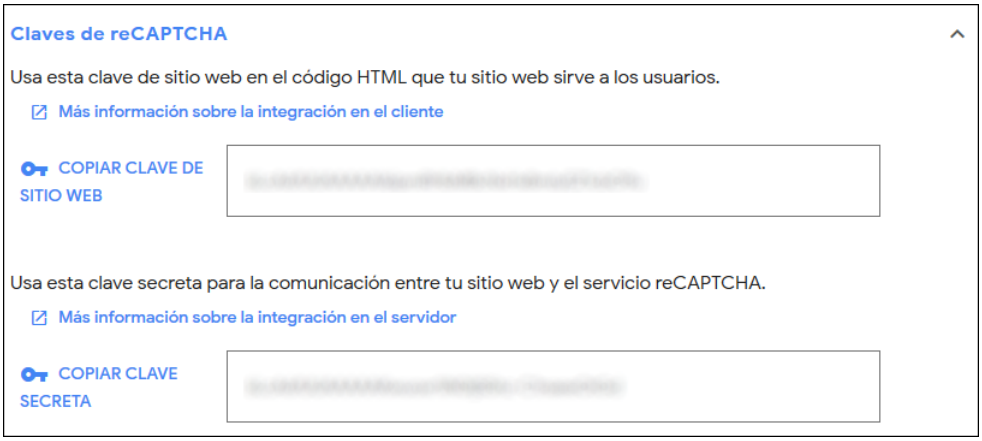


Figura 2.38: Llaves de Google reCAPTCHA.

Como se especificó en la Sección 1.3 el prototipo se alimentará de una base de datos con datos ficticios con el fin de simular una integración con la base de datos de alguna universidad. De lo antes dicho se desprende que se debe crear una base de datos con datos ficticios. Primero se crea la base de datos Sisvda, para esto se ejecuta los comandos en la terminal descritos en el segmento de código 2.10, donde, la línea 1 abre el *shell* de MongoDB y la línea 2 crea la base de datos.

Segmento de código 2.10: Código para la creación de la base de datos.

```
1 mongo
2 use Sisvda
```

Luego se debe crear la colección “users” para esto se procede a crear un *documento* en esta colección lo cual creará la colección y el *documento* a la vez. El documento creado servirá como plantilla para la creación de más usuarios. Debido a que existe gran cantidad de campos en los documentos de la colección “users” para ejemplificar la creación de un documento en esta colección solo se tomará en cuenta 15 campos. Las líneas de código del segmento de código 2.11 se debe ejecutar en el *shell* de MongoDB para la creación del documento.

Segmento de código 2.11: Código para la creación de un documento de la colección users.

```
1  item1 = {
2    name: "Pablo",
3    Apellidos: "Martinez",
4    email: "pablomartinez@prueba.com",
5    password: "123456",
6    Universidad: "Escuela Poltécnica Nacional",
7    Campus: "Politécnico J. Rubén Orellana R.",
8    Facultad: "INGENIERÍA ELÉCTRICA Y ELECTRÓNICA",
9    Carrera: "Ingeniería en Sistemas",
10   DecanoNombre: "MSc. Carlos Herrera",
11   CoordinadorNombre: "MSc. Jorge Carvajal",
12   CreditosTotalesCarrera: 245,
13   CreditosAprobados: 50,
14   Promedio: 7,
15   PeriodoActual: "2020A",
16   NumeroMaximoDocumentos: 5
17 }
18 db.users.insert(item1)
```

Una vez creado el usuario base se procede crear un conjunto de usuarios ejecutando el *script* del segmento de código 2.12, en el *shell* de MongoDB. Lo que hace este *script* es generar datos ficticios, creando 13 documentos en la colección “users” copiando todos los campos del usuario base, menos los que son únicos para cada usuario, los cuales son generados por el *script*.

Segmento de código 2.12: Código para la creación de un grupo de usuarios.

```
1 var copy = db.users.findOne("email":"pablomartinez@prueba.com");
2 for (var i = 1; i< 14; i++){
3     copy._id = new ObjectId();
4     copy.name = "usr" + i;
5     copy.email = "usr"+ i + "@mail.com"
6     copy.Cédula = "050284334" + i
7     db.users.insert(copy);
8 }
```

2.2.4. IMPLEMENTACIÓN DE LA CAPA DE NEGOCIO

En la siguiente sección se muestra la codificación del sistema prototipo basado en el diseño de clases de la Sección 2.1.6. Para esto se utilizará el *framework* Laravel en su versión 5.8 descrita en la Sección 1.4.5.2.
Para crear un nuevo proyecto en Laravel se hace uso del manejador de dependencias Composer por lo que se procede a ejecutar en la terminal el comando del segmento de código 2.13, donde Sisvda es el nombre del proyecto.

Segmento de código 2.13: Creación del proyecto.

```
1 composer create-project --prefer-dist laravel/laravel Sisvda "5.8.*"
```

Para que el proyecto pueda usar *eloquent* es necesario la instalación del Paquete Laravel MongoDB por ende para instalar este paquete desde la terminal se debe dirigir al *path* de la carpeta del proyecto y ejecutar el comando descrito en el segmento de código 2.14 [17].

Segmento de código 2.14: Instalar el Paquete Laravel MongoDB.

```
1 composer require jenssegers/mongodb
```

Siguiendo la documentación oficial del Paquete Laravel MongoDB como paso final para que la aplicación se integre con MongoDB se debe añadir las líneas del segmento de código 2.15 al archivo database.php del proyecto creado anteriormente [17]. Estas líneas añaden una conexión a una base de datos a la aplicación que en este caso es MongoDB.

Segmento de código 2.15: Configuración del Paquete Laravel MOngoDB.

```
1  'mongodb' => [
2              'driver'   => 'mongodb',
3              'host'     => env('DB_HOST', 'localhost'),
4              'port'     => env('DB_PORT', 27017),
5              'database' => env('DB_DATABASE'),
6              'username' => env('DB_USERNAME'),
7              'password' => env('DB_PASSWORD'),
8              'options'  => [
9                  'database' => 'admin' // sets the authentication database ...
                       required by mongo 3
10             ]
11         ],
```

El funcionamiento de una aplicación hecha en Laravel en síntesis es el siguiente:

- Las rutas asocian a cada *url* de la aplicación con una acción o una lógica enlazándola con un controlador. Los controladores son las clases que llevan la lógica de la aplicación.

- El controlador se comunica con el modelo el cual a su vez es el encargado de recibir o enviar información a la base de datos.
- Una vez que el controlador obtiene los datos los procesa y los envía a la vista. La vista es la interfaz gráfica que interactúa con el usuario final.

En las siguientes líneas se muestra la codificación de las rutas, controladores y modelos del presente prototipo.

2.2.4.1. Codificación del sistema de autenticación

Laravel trae consigo un sistema de autenticación que dota a las aplicaciones de un control de acceso de usuarios además permite al usuario recuperar su contraseña mediante el envío de un correo electrónico. Para poder integrarlo al presente prototipo se debe ejecutar en el terminal el segmento de código 2.16. Esto generará todo el código necesario para el sistema de autenticación es decir genera todos los controladores, vistas, rutas y modelos necesarios (ver Tabla 2.9). Entre los archivos que genera está el modelo *'user'*.

Tabla 2.9: Archivos generados por el sistema de autenticación de Laravel

Tipo	Nombre
Controlador	ForgotPasswordController.php
Controlador	LoginController.php
Controlador	ResetPasswordController.php
Controlador	VerificationController.php
Modelo	user
Vista	email.blade.php
Vista	reset.blade.php
Vista	login.blade.php
Vista	verify.blade.php
Vista	home.blade.php

Segmento de código 2.16: Instalación del sistema de autenticación.

```
1 php artisan make:auth
```

2.2.4.2. Codificación de los Modelos

Cabe destacar que para la creación de modelos Laravel recomienda colocar el nombre del modelo en el singular del nombre de la colección para que así Laravel asocie automáticamente ese modelo a esa colección. En el paso anterior se realizó la codificación del modelo *'user'*, en esta sección se presenta la codificación del modelo *'document'* misma que se detalla en el segmento de código 2.17. En dicho segmento de código se importa la clase *'Model'* la cual permite utilizar *'eloquent'* al modelo. Cabe recalcar que Laravel por defecto cuando crea un nuevo documento en la base de datos crea dos campos en donde se colocan hora

de creación y fecha de modificación. Para desactivar esta característica según la documentación oficial del paquete Laravel MongoDB [17] se debe poner el atributo *timestamps* en falso como se ve en la línea 7.

Segmento de código 2.17: Código del modelo document.

```
1 <?php
2 namespace App;
3 use Jenssegers\Mongodb\Eloquent\Model as Moloquent;
4
5 class Document extends Moloquent
6 {
7     public $timestamps = FALSE;
8 }
```

2.2.4.3. Codificación de Rutas

La codificación de rutas definen las diferentes direcciones o *urls* que va a tener la aplicación, asociando a cada ruta una acción. Esta codificación en Laravel se las realiza en el archivo 'web.php' la misma que se muestra en el segmento de código 2.19, donde, en la línea 3 muestra la ruta de la página principal la cual devuelve una vista llamada *welcome*. Las rutas generadas por el sistema de autenticación se muestra en la línea 8. De la línea 10 a la línea 15 se muestra las rutas asociadas al usuario tipo estudiante y público estas rutas siguen el formato descrito en el segmento de código 2.18. Donde especifica que para poner un nombre a la ruta se usa el método *'name'*.

Segmento de código 2.18: Formato de rutas en Laravel.

```
1 Route::<métodoHTTP>('nombre de la ...
      ruta','NombreControlador'@'metodo')->name('Nombre de la ruta');
```

Segmento de código 2.19: Código de las rutas.

```
1 <?php
2 // ruta de la pagina principal
3 Route::get('/', function () {
4 return view('welcome');
5 });
6
7 //rutas del sistema de autenticación
8 Auth::routes();
9
10 // Rutas para el usuario tipo estudiante
11 Route::get('/home', 'UsuarioEstudiante@UltimosDocumentos')->name('home');
```

```
12 Route::get('documentlist', ...
       'UsuarioEstudiante@ListarDocumentos')->name('documentlist.form');
13 Route::post('generate', ...
       'UsuarioEstudiante@GenerarPDF')->name('generate.form');
14
15 // Rutas para el usuario tipo publico y estudiante
16 Route::post('descargarpdf', ...
       'UsuarioPublico@Descargarpdf')->name('descargarpdf');
```

2.2.4.4. Codificación de los controladores

Los controladores van a llevar toda la lógica de la aplicación, en esta sección se muestra la codificación de los controladores *'UsuarioEstudiante'* y *'UsuarioPublico'*.
En el segmento de código 2.20 se muestra una parte de la implementación del controlador *'UsuarioPublico'* el cual recibe una petición para descargar un documento académico. Mediante la *api* de *Google ReCAPTCHA* se verifica que la petición no la haya realizado un *bot*. La *api* devuelve un puntaje de 0 a 1 donde, 1.0 es una interacción confiable, al contrario que un puntaje más cercano a 0.0 indicará que el tráfico muy probablemente fue generado por *bots*. Este puntaje se evalúa mediante un condicional, como se ve en la línea 1.
Si el puntaje es bajo se manda un mensaje de error caso contrario se procede a buscar si en la colección *'documents'* existe algún *documento* que tenga el código hash ingresado. Esto se lo hace mediante el modelo *'document'* el cual va a permitir hacer la búsqueda dentro de la colección *'documents'*, como se puede apreciar en la línea 5. El código completo del controlador esta disponible en el ANEXO A.
Posteriormente si la búsqueda es exitosa se procede a la descarga del documento académico.

Segmento de código 2.20: Parte del código del controlador UsuarioPublico.

```
1  if ($resultJson->score ≤ 0.3) {
2              return back()->with('mensaje', 'ReCaptcha Error: Alerta de bot');
3      } else {
4          //Validación exitosa de ReCAPTCHA
5          $documentos =Document::where('CódigoHash', ...
               $codigohash)->take(1)->get();
6
7          if($documentos->isEmpty()){
8              return back()->with('mensaje','El código no es valido');
9          }
10         else
11         {
12             //Validación y descarga del documento
```

Por otro lado, el controlador *'UsuarioEstudiante'* va a encargarse de toda la lógica que engloba al usuario tipo estudiante. En el segmento de código 2.21 se puede observar una parte

de la implementación del controlador, específicamente el código de los métodos *'UltimosDocumentos()'* y *'ListarDocumentos()'*. El método *'UltimosDocumentos()'* permite la obtención del listado de los documentos generados, de ahí que se hace una búsqueda en la colección *'document'* haciendo uso del modelo *'Document'* como se muestra en la línea 4. El método *'ListarDoscumentos()'* direcciona a una vista en la cual el usuario debe seleccionar el documento académico que desea solicitar como muestra la línea 11.
El controlador también cuenta con el método *'GenerarPDF()'* que en pocas palabras recibe la petición de generar un documento esta es almacenada en la variable 'optionselected'. Luego hace las validaciones necesarias para saber si el usuario puede solicitar el documento académico, si pasa las validaciones genera el documento PDF como se muestra en la línea 3 del segmento de código 2.22. Posteriormente se calcula el código hash del documento y se genera el documento entregable en las líneas 8 y 14 respectivamente. El código completo del presente controlador esta disponible en el ANEXO B.

Segmento de código 2.21: Código de los métodos UltimosDocumentos y ListarDocumentos.

```
1  public function UltimosDocumentos()
2  {
3      $cedula = \Auth::user()->Cédula;
4      $posts = Document::where('UsuarioCédula',$cedula)->get();
5
6      return view('home',compact('posts'));
7  }
8
9  // Direcciona a la panatalla donde se elige el documento a solicitar
10 public function ListarDocumentos(){
11     return view('Pages.documentlist');
12 }
```

Segmento de código 2.22: Parte del código del método GenerarPDF.

```
1  // SI PASA VALIDACIONES GENERA PDF
2  // Crea PDF
3  $pdf = ...
       PDF::loadView('Documents.'.$optionselected,compact('DataPDF'))->save($PathCompletoDocume
4
5
6  //Genero codigo hash del PDF
7  $DataEmail = array(
8      'codigohash' => hash_file('sha256', $PathCompletoDocumento),
9      'documento' => $optionselected,
10     'PathCompletoDocumento' => $request->root()
11 );
12
```

```
13
14 $DataPDF['CódigoHash'] = $DataEmail['codigohash'];
15
16
17 // Crea PDF con hash
18 $pdf = PDF::loadView('Documents.'.$optionselected,compact('DataPDF'));
19 $pdf->save($PathCompletoDocumentoFinal);
```

2.2.5. IMPLEMENTACIÓN DE LA CAPA DE PRESENTACIÓN

En la presente sección se muestra la implementación de las interfaces gráficas que se diseñaron en la Sección 2.1.7. Es preciso señalar que en esta sección solo se detallará una parte del código para ejemplificar el funcionamiento del prototipo, ya que el código completo de las vistas principales del sistema *frontend* estará en el Anexo C.
Para la codificación de cada interfaz gráfica del prototipo se usó el sistema de plantillas *Blade* de Laravel.

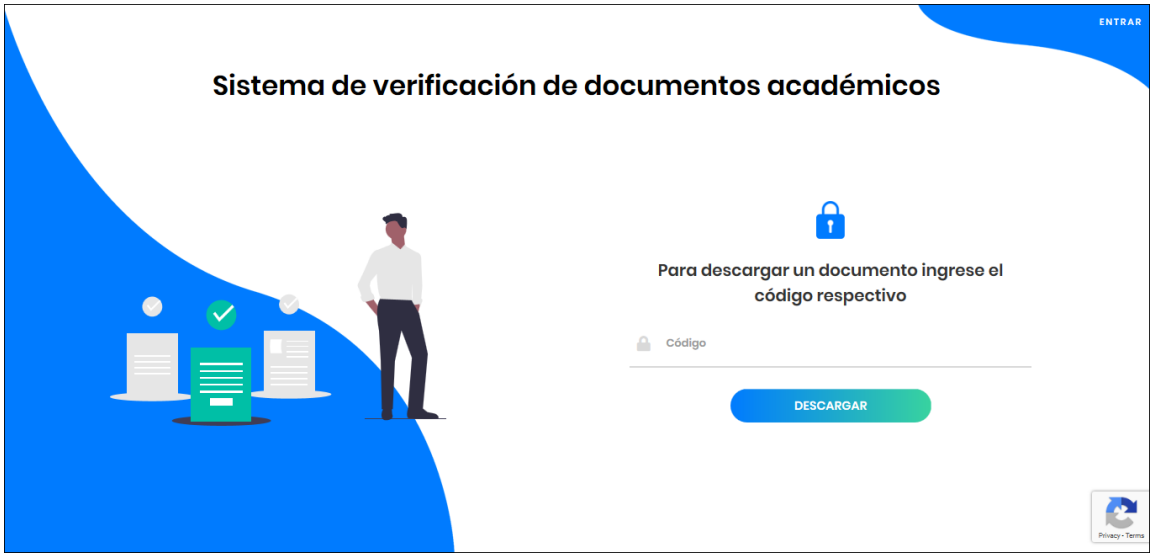


Figura 2.39: Implementación de la interfaz de la página de inicio.

Cuando el usuario ingresa al sistema la primera página que se visualiza es la página de inicio. En la figura 2.39 se visualiza la implementación de la interfaz de la página de inicio del prototipo, dicha implementación se encuentra en la vista *'welcome'*. Esta interfaz consta de una caja de texto por medio del cual el usuario ingresa el código hash asociado a un documento académico y dos botones uno que permite la descarga del documento académico y otro permite al usuario iniciar sesión.
El segmento de código 2.23 presenta parte del contenido de la vista *'welcome'* la cual tiene un formulario *HTML*, en donde el código que se ingresa en la caja de texto se manda mediante el método POST de HTTP a la ruta *'descargarpdf'* como se muestra en la línea 1. Esta ruta a su vez direcciona la información al método *'Descargarpdf'* del controlador *'UsuarioPublico'* el mismo que tiene la lógica para la descarga del documento académico, como se detalló en la Sección 2.2.4.3.

Segmento de código 2.23: Parte del código de la vista ‘welcome’.

```
<form target="_blank" action="{{route('descargarpdf')}}" method="post">
    @csrf
    @error('codigohash')
        <div class="alert">
            Ingrese un código, no deje vacío el campo
        </div>
    @enderror
    <img src="{{ asset('img/Candado.png') }}">
    <h2>Para descargar un documento ingrese el código respectivo</h2>
    <div class="input-div one">
        <div class="i">
            <i class="fas fa-lock"></i>
        </div>
        <div class="div">
                <h5>Código</h5>
                <input type="text" class="input" name="codigohash">
        </div>
    </div>
    <input type="hidden" name="recaptcha" id="recaptcha">
    <p align="center"><button class="btn">Descargar</button></p>
</form>
```

Una vez que el usuario inicia sesión aparece en pantalla la interfaz del listado de documentos solicitados. Esta implementación se encuentra en la vista *‘home’*, conformada por una tabla que muestra los últimos documentos académicos solicitados y el botón ‘solicitar documentos’ el cual permite al usuario pedir un documento académico. La implementación de esta pantalla se muestra en la Figura 2.40.

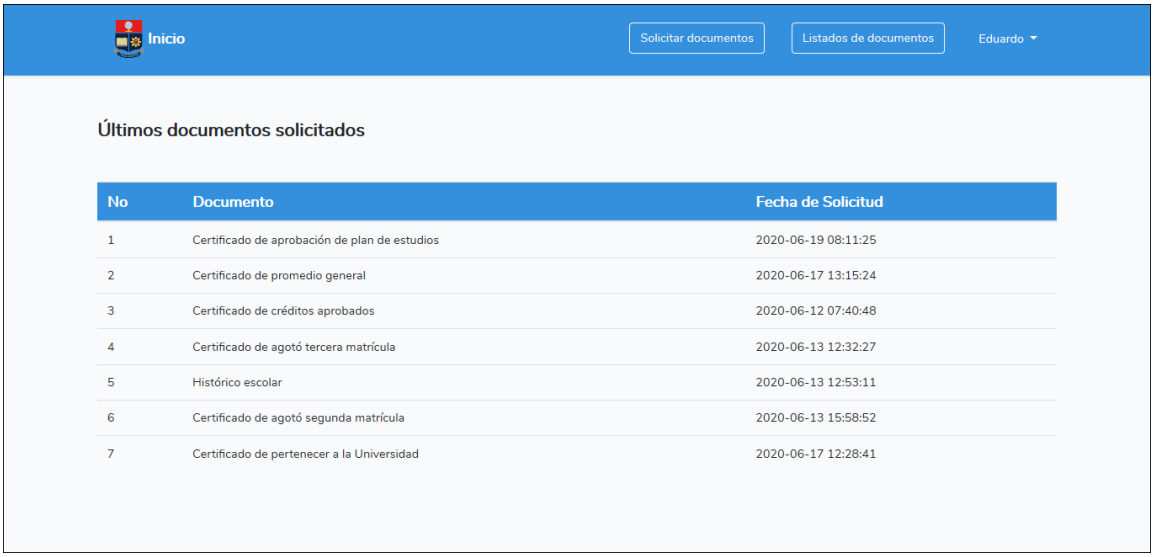


Figura 2.40: Implementación de la interfaz de el listado de documentos solicitados.

Cuando el usuario selecciona el botón de ‘solicitar documentos’ se abre la vista ‘documentlist’ en donde se encuentra la implementación de la interfaz para la petición de do-

cumentos académicos, dicha implementación se muestra en la Figura 2.41. La vista 'documentlist' consta de un formulario *HTML* el cual en su interior tiene 7 botones radio, que permiten al usuario seleccionar el documento académico a solicitar. Esta solicitud es enviada a la ruta *'generate.form'* como se muestra en la línea 1 del segmento de código 2.24. Esta ruta a su vez direcciona la solicitud al método *'GenerarPDF()'* del controlador *'UsuarioEstudiante'* el cual tiene la lógica necesaria para la generación de un documento PDF, como se detalló en la secciones 2.2.4.3 y 2.2.4.4.

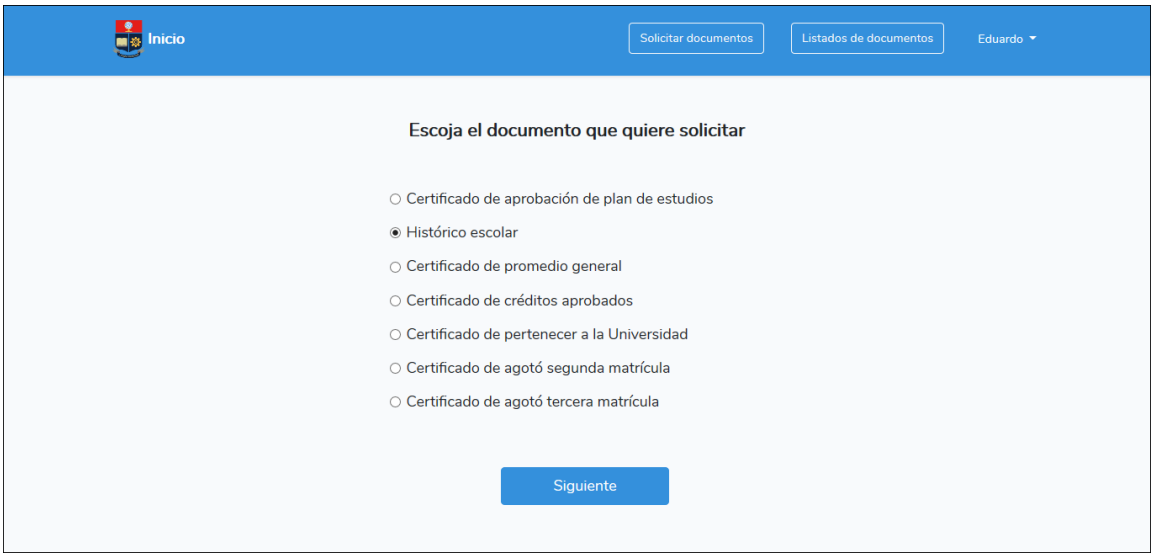


Figura 2.41: Implementación de la interfaz para la petición de documentos.

Segmento de código 2.24: Parte del código de la vista 'documentlist'.

```
1  <form action="{{route('generate.form')}}" method="POST">
2      @csrf
3      <div class="form-check d-flex flex-column justify-content-around" ...
           style="height: 60vh">
4          <div>
5              <input class="form-check-input" type="radio" ...
                   name="exampleRadios" value="Certificado de aprobación de ...
                   plan de estudios" checked>
6              <label class="form-check-label" for="exampleRadios1">
7                  Certificado de aprobación de plan de estudios
8              </label>
9          </div>
10         <div>
11             <input class="form-check-input" type="radio" ...
                   name="exampleRadios" value="Histórico escolar" checked>
12             <label class="form-check-label" for="exampleRadios1">
13                 Histórico escolar
14             </label>
15         </div>
```

Es importante destacar que todas las vistas fueron codificadas de tal manera que la aplicación es totalmente responsiva.

2.2.6. PUESTA EN PRODUCCIÓN

En esta sección se detalla el proceso a realizar para pasar del ambiente de desarrollo a la instancia creada en la Sección 2.2.2.1.
Para este propósito se va a usar *GIT* junto con *GitHub*. Primero desde la página web de *Github* se crea un repositorio nuevo, ciertamente este proceso es sencillo, basta con seleccionar el botón *new* de la página principal de *Github* y colocar el nombre de repositorio.
Por otro lado en el ambiente local desde la terminal, se debe ubicar en el directorio del proyecto y ejecutar los comandos del segmento de código 2.25. Esto crea un repositorio local, pasa todos los archivos del proyecto al área de preparación y genera una versión del repositorio local como se muestra en las lineas 1, 2, 3 respectivamente. Los comandos de las líneas 4 y 5 envían el proyecto al repositorio de *Github*.

Segmento de código 2.25: Comandos para actualizar el repositorio en Github.

```
1 git init
2 git add .
3 git commit -m "Paso a producción"
4 git remote add origin https://github.com/Brakith/Sisvda.git
5 git push -u origin master
```

Por último se debe pasar la información del repositorio de *github* a la instancia creada en la sección 2.2.2.1, por ende se debe clonar el repositorio de *Github* la línea que se debe ejecutar para este propósito se muestra en el segmento de código 2.26. Así ya se tendrá todo el código del proyecto en producción. Es importante destacar que al copiar la información del repositorio local al repositorio en *Github* el archivo '.env' no se copia. Este archivo contiene todas las credenciales de los servicios que usa la aplicación, entre los cuales están el de correo electrónico, base de datos y *recaptchakeys*. Este archivo debe ser creado manualmente en la instancia con el formato del segmento de código 2.27.

Segmento de código 2.26: Comando para clonar el repositorio de la nube.

```
1 git clone https://github.com/Brakith/Sisvda.git
```

Segmento de código 2.27: Formato del archivo .env.

```
1 APP_NAME=SISVDA
2 APP_ENV=production
3 APP_KEY=base64:kxq6WFK0NgWXLZX/Uah1SQvlJko=
```

```
4  APP_DEBUG=false
5  APP_URL=www.sisvda.ml
6
7  LOG_CHANNEL=stack
8
9  DB_CONNECTION=mongodb
10 DB_HOST= localhost
11 DB_PORT=27017
12 DB_DATABASE=laravelmongodb
13 DB_USERNAME= admin
14 DB_PASSWORD= admin
15
16 RECAPTCHA_SECRET=6LcA6fUUAAAAAKwuoxr1i
17 RECAPTCHA_SITEKEY=6LcA6fUUAAAAAI6ipmRH
18
19 MAIL_DRIVER=smtp
20 MAIL_HOST=smtp.gmail.com
21 MAIL_PORT=587
22 MAIL_USERNAME=soportesistemadeverificacion@gmail.com
23 MAIL_PASSWORD=uzeigchx
24 MAIL_ENCRYPTION=tls
25 MAIL_FROM_NAME = 'Soporte SISVDA'
```

Cabe señalar que todo el código del proyecto está disponible para el público en el repositorio de Github antes mencionado.

Una vez el proyecto está en producción se procede a realizar pruebas con un grupo de usuarios finales, con el fin de realizar captura de errores del sistema y poder posteriormente realizar a los usuarios una encuesta de satisfacción.

3. RESULTADOS Y DISCUSIÓN

Este capítulo muestra las diferentes pruebas realizadas al prototipo con el fin de validar su correcto funcionamiento. Primero en la Sección 3.2 se muestran los resultados de las pruebas que realizaron un grupo de usuarios finales, luego en las Secciones 3.3 y 3.4 se valida que el prototipo cumpla con los requerimientos definidos en las Secciones 2.1.4 y 2.1.3.

3.1. ACTUALIZACIÓN DEL TABLERO KANBAN

Las tareas a realizar en el presente capítulo son las relacionadas a la etapa de pruebas, de ahí que se actualiza el tablero Kanban como muestra la Figura 3.1.

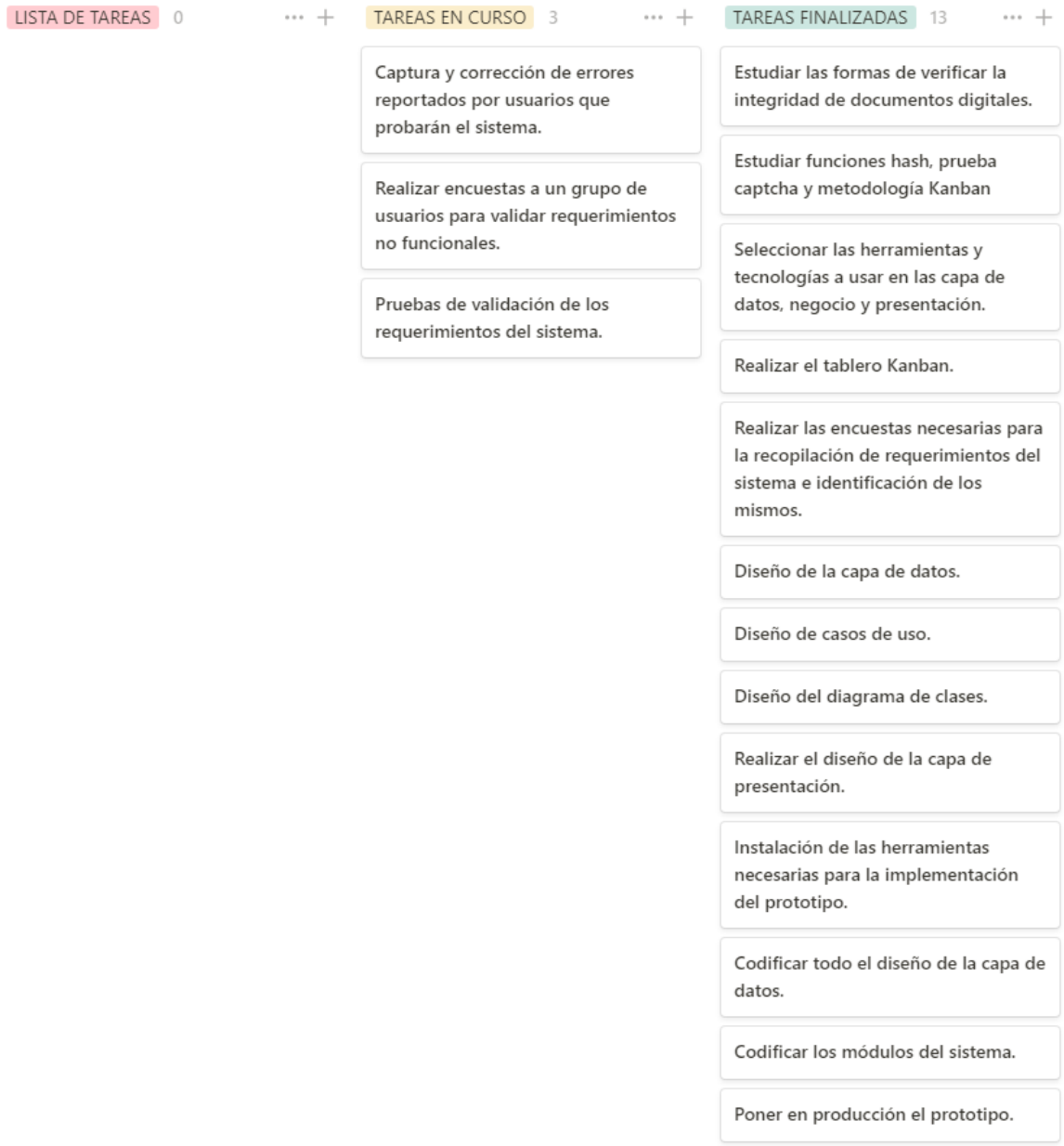


Figura 3.1: Tablero Kanban etapa de pruebas.

3.2. RESULTADOS DE PRUEBAS DE USUARIOS

Con la finalidad de obtener un producto final de calidad y que satisfaga los requerimientos definidos en las Secciones 2.1.3 y 2.1.4, se realizó una serie de pruebas de funcionalidad al sistema prototipo. Estas pruebas las realizaron un grupo de 7 voluntarios, los mismos que son estudiantes universitarios y potenciales usuarios del prototipo.
El proceso para la ejecución de las pruebas comenzó con la realización de una vídeo conferencia con cada uno de los voluntarios, para explicar el funcionamiento y objetivos del prototipo. Se les explicó todo lo que debían probar en el sistema incluyendo validaciones, lo que se detalla a continuación:

- Verificar que cada tipo de documento académico se pueda generar una sola vez por día y 5 veces por semestre.
- Validar que los documentos académicos se generen sin ningún error.
- Constatar que el sistema envíe un correo notificando al usuario la generación exitosa del documento.
- Probar en general el sistema.

Finalizadas las pruebas los usuarios detectaron algunos errores en el sistema, en su mayoría fueron errores de UI y otros fueron falsos positivos, todos los errores encontrados por los usuarios se detallan a continuación:

Falsos positivos:
Algunos usuarios reportaron que no habían recibido el correo electrónico. Esto fue un falso positivo, debido a que el correo si llego, solo que fue detectado como *spam*, esto debido a que el sistema prototipo para el envío de correos electrónicos usa una cuenta personal de Gmail. Para sistemas reales se usan servicios de pago como Amazon Simple Notification Service, los cuales tiene todas las configuraciones necesarias para que los correos enviados no sean detectados como *spam*.

Errores de UI:

- El encabezado de los documentos generados por el prototipo, tenían un error de escritura, el nombre de la Universidad, estaba impreso 'POLTÉCNICA' en vez de ´POLITÉCNICA'. Esto fue por causa de un error en la generación de los datos en MongoDB; se corrigió con una actualización en la misma, ejecutando el comando del Segmento de código 3.28 en el Shell de MongoDB.

Segmento de código 3.28: Actualizar campo Universidad.

```
1    db.users.update({Universidad:"POLTÉCNICA"},
2    {$set:{Universidad:"POLITÉCNICA"}})
```

- Una usuaria reportó que los documentos generados contenían el texto ‘señor Lorena’. Este error es a causa de que el campo *´Género’* del *documento* perteneciente a la usuaria estaba asignado con el valor “Masculino”. Para corregir este error se actualizó dicho campo, ejecutando el comando del Segmento de código 3.29 en el Shell de MongoDB.

Segmento de código 3.29: Actualizar campo Género.

```
db.users.update({"email":"lorena.tepud@epn.edu.ec"},
{$set:{Género:"Femenino"}})
```

- Se reportó que en la pantalla donde se selecciona el documento académico, cuando se genera un mensaje de alerta, se selecciona un botón de radio diferente al seleccionado por el usuario. Esta falla es a causa de un error en la codificación de los botones de radio. En la vista *‘documentlist’* cada etiqueta *input* tenía el atributo *checked*, este atributo se usa para que por defecto se seleccione ese botón de radio. Al tener todos los botones de radio este atributo, el sistema no sabia cual seleccionar por defecto. Para corregir este error se eliminó el atributo *checked* de todos los botones de radio, logrando así que cuando se produzca un mensaje de error la interfaz se reinicie, es decir que ningún botón de radio esté seleccionado y así evitar confusión en los usuarios. El resultado de esta corrección se muestra en la figura 3.2.

Figura 3.2: Corrección en la vista ‘documentlist’.

3.3. PRUEBAS DE VALIDACIÓN DE REQUERIMIENTOS FUNCIONALES

En esta Sección se valida el correcto funcionamiento de cada módulo del sistema prototipo detallados en la Sección 2.1.4. El proceso a seguir es separar los requerimientos funcionales

por módulo y posteriormente se procede a validar cada requerimiento funcional.

3.3.1. PRUEBA DE FUNCIONAMIENTO AL MÓDULO DE PETICIÓN DE DOCUMENTO

Este Módulo permite al usuario iniciar sesión y validar las solicitudes de documentos emitidas por los estudiantes. La Tabla 3.1 muestra los requerimientos funcionales por cumplir en el presente módulo.

Tabla 3.1: Requerimientos funcionales del Módulo de petición de documentos académicos.

RF	Descripción
RF001	El prototipo permitirá al usuario tipo estudiante hacer login en la plataforma
RF002	El usuario del tipo estudiante, puede solicitar un documento académico bajo las siguientes restricciones: Se puede generar cada tipo documento una vez al día y como máximo 5 veces por semestre.
RF003	El usuario tipo estudiante a lo largo del tiempo va a generar varios documentos académicos. El prototipo le da la posibilidad de visualizar la lista de los últimos documentos académicos solicitados.

La pantalla de inicio de sesión permite ingresar a la plataforma digitando el correo electrónico y la contraseña respectiva, si las credenciales no son correctas se muestra un mensaje de error como se observa en la Figura 3.3. Si las credenciales son correctas se ingresa al sistema desplegándose la pantalla de últimos documentos generados, como se ve en la Figura 3.4. De lo dicho anteriormente se demuestra el cumplimiento del RF001 y RF003.

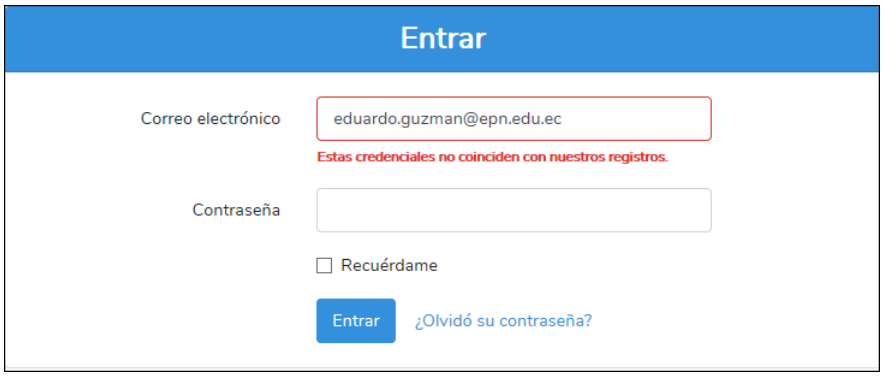


Figura 3.3: Inicio de sesión fallido.

En la pantalla para la solicitud de documentos académicos, si el estudiante quiere solicitar un documento que ya ha sido generado ese mismo día, se muestra un mensaje de alerta como se muestra en la Figura 3.5. Si el estudiante quiere generar mas de 5 documentos académicos en el semestre del mismo tipo, se despliega un mensaje de alerta como se muestra en la Figura 3.6. De esta manera se demuestra el cumplimiento del RF002.

Últimos documentos solicitados

No	Documento	Fecha de Solicitud
1	Certificado de aprobación de plan de estudios	2020-06-28 11:52:43
2	Certificado de promedio general	2020-06-17 13:15:24
3	Certificado de créditos aprobados	2020-06-12 07:40:48
4	Certificado de agotó tercera matrícula	2020-06-29 13:14:09
5	Histórico escolar	2020-06-28 11:42:58
6	Certificado de agotó segunda matrícula	2020-06-13 15:58:52
7	Certificado de pertenecer a la Universidad	2020-06-17 12:28:41

Figura 3.4: Lista de últimos documentos generados.

El documento 'Certificado de agotó tercera matrícula' ya se genero el día de hoy. Recuerde que sólo se puede generar una vez al día cada tipo de documento.

Escoja el documento que quiere solicitar

- Certificado de aprobación de plan de estudios
- Histórico escolar
- Certificado de promedio general
- Certificado de créditos aprobados
- Certificado de pertenecer a la Universidad
- Certificado de agotó segunda matrícula
- Certificado de agotó tercera matrícula

Siguiente

Figura 3.5: Validación por día de la generación documental.

3.3.2. PRUEBA DE FUNCIONAMIENTO AL MÓDULO DE GENERACIÓN DE DOCUMENTOS

Este módulo se encarga de la generación de los documentos y notificar al estudiante por medio de correo electrónico cuando el documento esté listo, el requerimiento funcional para este módulo se muestra en la Tabla 3.2

Tabla 3.2: Requerimiento funcional del Módulo de generación de documentos académicos

RF	Descripción
RF004	El prototipo enviará un email al usuario tipo estudiante notificando la generación exitosa del documento académico y el código hash asociado al mismo.

Una vez solicitado el documento llega al correo electrónico un correo similar al de la Figura 3.7, por ende se comprueba el cumplimiento del RF004.

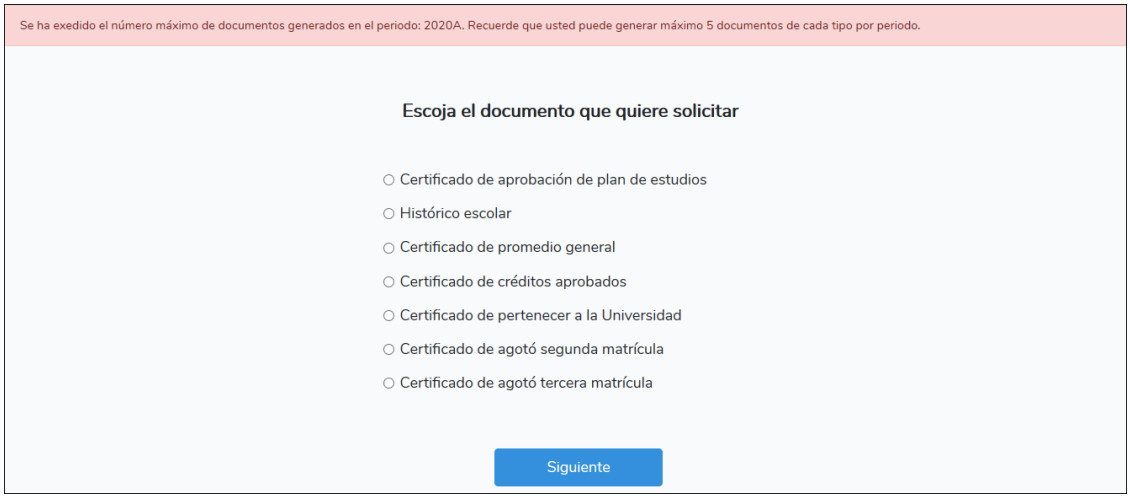


Figura 3.6: Validación por periodo de la generación documental.

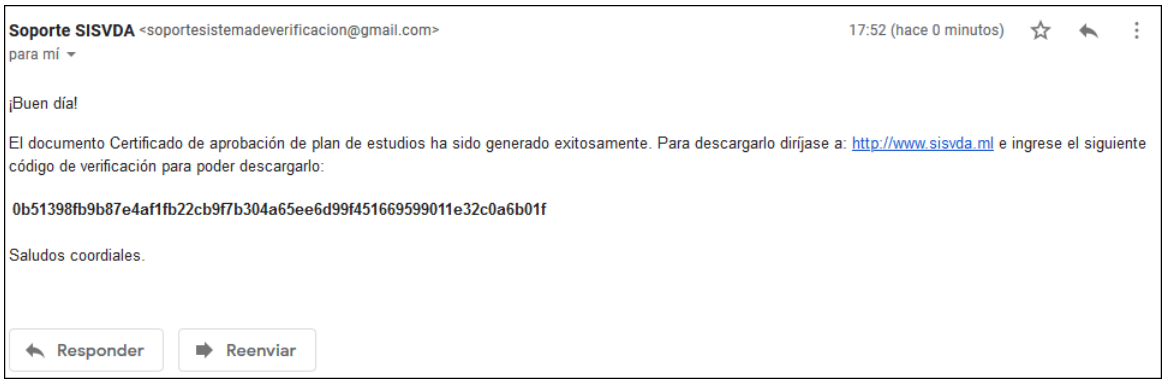


Figura 3.7: Correo de notificación “generación exitosa del documento”.

3.3.3. PRUEBA DE FUNCIONAMIENTO AL MÓDULO DESCARGA DE DOCUMENTOS

Este módulo permite descargar los documentos académicos ingresando el código hash asociado, los requerimientos funcionales de este módulo se detallan en la Tabla 3.3.

Tabla 3.3: Requerimientos funcionales del Módulo de descarga de documentos académicos

RF	Descripción
RF005	El usuario tipo estudiante puede descargar el documento académico entregable, para esto debe iniciar sesión e ingresar el código hash asociado al documento académico.
RF006	El usuario de tipo público puede descargar un documento académico sin necesidad de hacer login simplemente usando el código asociado al documento académico.

Si el usuario tipo estudiante, después de iniciar sesión ingresa el código hash en la página de descarga (ver Figura 3.8) y presiona el botón descargar, se abre una nueva pestaña con el documento académico entregable como se muestra en la Figura 3.9.
Por otro lado si el usuario tipo público ingresa el código hash y presiona el botón descargar,

Figura 3.8: Pantalla de descarga.

ESCUELA POLITÉCNICA NACIONAL

Campus Politécnico J. Rubén Orellana R.

FACULTAD DE INGENIERÍA EN ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

2020-06-30

CERTIFICO

Que el señor Eduardo Guzmán, aprobó el Plan de Asignaturas de la Carrera de Ingeniería en Electrónica y Telecomunicaciones. Adicionalmente indico que el mencionado estudiante debe realizar su Trabajo de Titulación previo a la obtención del título de **INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**. El señor Eduardo Guzmán, puede hacer uso de esta certificación para los fines que considere conveniente.

MSc. Jorge Carvajal
COORDINADOR

CÓDIGO DE AUTENTICIDAD

Verifique la autenticidad de este documento ingresando a la pagina **http://www.sisvda.ml** e ingresado el siguiente codigo:

0b51398fb9b87e4af1fb22cb9f7b304a65ee6d99f451669599011e32c0a6b01f

SISVDA
Escuela Politécnica Nacional
Av. Ladrón de Guevara 253, Quito 170517
http://www.sisvda.ml

Figura 3.9: Documento académico entregable.

se abre en otra pestaña el documento académico como se ve en la Figura 3.10.
Concluidas las pruebas a cada requerimiento funcional, se concluye que el sistema prototipo para la emisión y verificación de la integridad de documentos académicos cumple de manera correcta con las especificaciones de diseño y objetivos para el cual fue desarrollado. El cumplimiento de los RF se detalla en la Tabla 3.4.

ESCUELA POLITÉCNICA NACIONAL

Campus Politécnico J. Rubén Orellana R.

FACULTAD DE INGENIERÍA EN ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

2020-06-30

CERTIFICO

Que el señor Eduardo Guzmán, aprobó el Plan de Asignaturas de la Carrera de Ingeniería en Electrónica y Telecomunicaciones. Adicionalmente indico que el mencionado estudiante debe realizar su Trabajo de Titulación previo a la obtención del título de **INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**. El señor Eduardo Guzmán, puede hacer uso de esta certificación para los fines que considere conveniente.

MSc. Jorge Carvajal
COORDINADOR

SISVDA
Escuela Politécnica Nacional
Av. Ladrón de Guevara 253, Quito 170517
http://www.sisvda.ml

Figura 3.10: Documento académico.

Tabla 3.4: Cumplimiento de los requerimientos funcionales

Módulo	RF	Descripción	Cumple
Petición	RF001	El prototipo permitirá al usuario tipo estudiante hacer login en la plataforma.	SI
	RF002	El usuario del tipo estudiante, puede solicitar un documento académico bajo las siguientes restricciones: Se puede generar cada tipo documento una vez al día y como máximo 5 veces por semestre.	SI
	RF003	El usuario tipo estudiante a lo largo del tiempo va a generar varios documentos académicos. El prototipo le da la posibilidad de visualizar la lista de los últimos documentos académicos solicitados.	SI
Generación	RF004	El prototipo enviará un email al usuario tipo estudiante notificando la generación exitosa del documento académico y el código hash asociado al mismo.	SI
Descarga	RF005	El usuario tipo estudiante puede descargar el documento académico entregable, para esto debe iniciar sesión e ingresar el código hash asociado al documento académico.	SI
	RF006	El usuario de tipo público puede descargar un documento académico sin necesidad de hacer login simplemente usando el código asociado al documento académico.	SI

3.4. PRUEBAS DE VALIDACIÓN DE REQUERIMIENTOS NO FUNCIONALES

En esta sección se valida cada requerimiento no funcional planteado en la Sección 2.1.3. Los requerimientos no funcionales a validar se listan en la Tabla 3.5.

Tabla 3.5: Requerimientos no funcionales

	RF	Descripción
Presentación	RNF001	La interfaz deberá ser rápida, amigable e intuitiva.
Autenticación	RNF002	El acceso al módulo de solicitud de documentos académicos debe estar sólo accesible a usuarios registrados.
Rendimiento	RNF003	La entrega de documentos académicos debe ser menor a un día.
Seguridad	RNF004	Las credenciales de los usuarios y los documentos académicos no deben ser enviados en texto plano.
Escalabilidad	RNF005	El prototipo debe ser implementado en un servidor de tal manera que sea posible el mejoramiento de su hardware como son: la memoria RAM, procesador y espacio de almacenamiento.

3.4.1. Presentación y Rendimiento

A los 7 voluntarios que realizaron las pruebas al sistema, mismas que se detallan en la Sección 3.2, se les realizó una encuesta haciendo uso de la herramienta web Google Forms [36], con la finalidad de validar requerimientos no funcionales correspondientes a “Presentación” y “Rendimiento”.
Las preguntas y los resultados de las encuestas fueron los siguientes:

- **¿El sistema es amigable e intuitivo, 1 es nada amigable y 5 muy amigable?**
  La totalidad de los encuestados calificó a la usabilidad de la plataforma con una valoración mayor a 4 sobre 5, por consiguiente se concluye que la plataforma resulto fácil de usar al usuario final. Los resultados de la pregunta se muestra en la Figura 3.11.

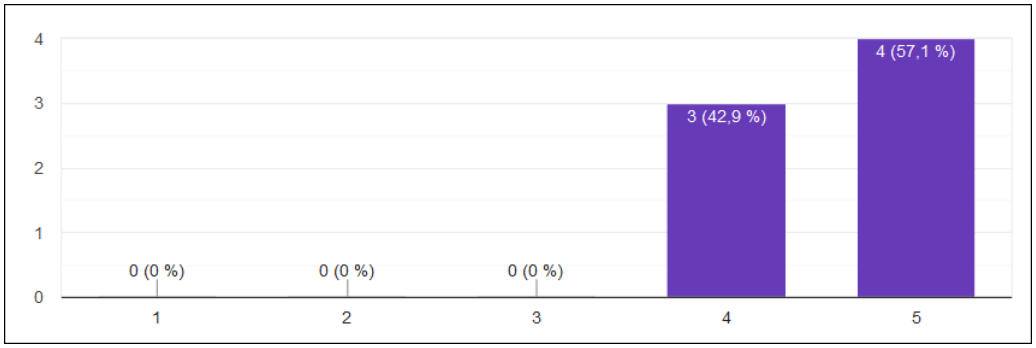


Figura 3.11: Usabilidad del sistema.

- **¿Considera que el rendimiento del sistema es rápido, 1 es lento y 5 excelente?**
  Casi la totalidad de los encuestados encuentran que el rendimiento de la plataforma en

cuanto a velocidad es excelente, por ende se verifica el cumplimiento del requerimiento RNF001 correspondiente a “Presentación”. Los resultados de la pregunta se muestra en la Figura 3.12.

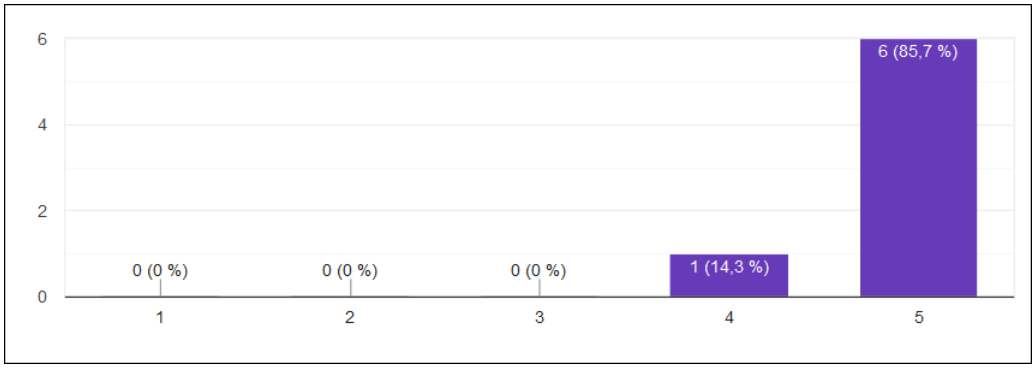


Figura 3.12: Rendimiento del prototipo.

- **¿Cuánto demoró el sistema para la entrega del documento académico solicitado?**
  Ninguno de los encuestados afirmo que la entrega del documento académico demoró mas de 30 minutos, por consiguiente se verifica el cumplimiento del requerimiento RNF003 correspondiente a “Rendimiento”. Los resultados de la pregunta se muestra en la Figura 3.13.

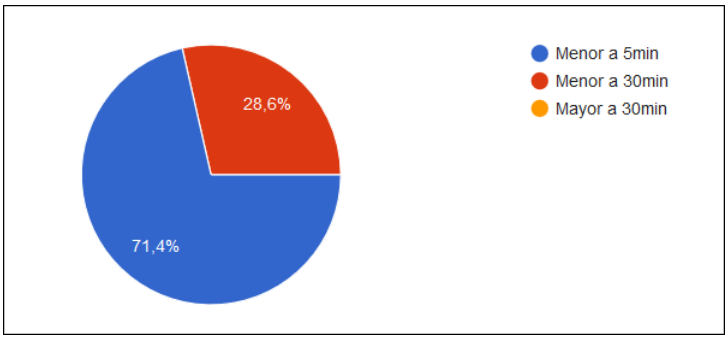


Figura 3.13: Tiempo de entrega de los documentos.

3.4.2. Autenticación

El acceso al módulo de solicitud de documentos académicos sólo debe ser accesible a los usuarios que inicien sesión en el sistema. Para comprobar si el módulo esta protegido, se debe ingresar al sistema y digitar la ruta de la pantalla que permite solicitar documentos académicos como se muestra en la Figura 3.14. Al querer ingresar a esta ruta se abre la pantalla de iniciar sesión como se muestra en la Figura 3.15, en la cual si no se tiene las credenciales correctas no se puede ingresar al sistema. Como resultado, se verifica el cumplimiento del RFN002 correspondiente a “Autenticación”.

Figura 3.14: Ruta de la vista “documentlist”.

Figura 3.15: Pantalla de inicio de sesión.

3.4.3. Seguridad

Las credenciales de los usuarios y los documentos académicos no deben ser enviados en texto plano. El prototipo para el intercambio de información entre el usuario y aplicación implementa el protocolo HTTPS como se ve en la Figura 3.16, es decir que el intercambio de información se va a realizar en un canal cifrado. Como conclusión se cumple con el RFN004. Es importante destacar, que el prototipo se codificó de tal manera que si intentamos acceder a la aplicación por medio de HTTP se a redireccionar al protocolo HTTPS.

Figura 3.16: URL del sistema prototipo.

3.4.4. Escalabilidad

El presente prototipo ha sido implementado en el servicio web Elastic compute cloud de Amazon [19]. Este servicio provee instancias escalables, es decir que se puede ir incrementando los recursos del servidor tales como: almacenamiento, memoria, y velocidad de red, por consiguiente el requerimiento “Escalabilidad” o RFN005 se da por cumplido.

Para finalizar se presenta la lista de cumplimiento de los requerimientos no funcionales (ver Tabla 3.6).

3.5. TABLERO KANBAN FINAL

Concluidas las tareas de la etapa de resultados, se completa todas las actividades que se estipularon en un inicio en el tablero Kanban, como lo muestra la Figura 3.17, por ende se da por terminado el presente prototipo.

Tabla 3.6: Cumplimiento de requerimientos no funcionales

	RF	Descripción	Cumple
Presentación	RNF001	La interfaz deberá ser rápida, amigable e intuitiva.	SI
Autenticación	RNF002	El acceso al módulo de solicitud de documentos académicos debe estar sólo accesible a usuarios registrados.	SI
Rendimiento	RNF003	La entrega de documentos académicos debe ser menor a un día.	SI
Seguridad	RNF004	Las credenciales de los usuarios y los documentos académicos no deben ser enviados en texto plano.	SI
Escalabilidad	RNF005	El prototipo debe ser implementado en un servidor de tal manera que sea posible el mejoramiento de su hardware como son: la memoria RAM, procesador y espacio de almacenamiento.	SI

Figura 3.17: Actualización final del Tablero Kanban.

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

- Finalizado este Trabajo de Titulación, se ha cumplido el objetivo general: diseñar un sistema prototipo para generar y verificar la integridad de documentos académicos. Este prototipo permite a los estudiantes generar diferentes tipos de documentos académicos en línea, permitiendo al receptor del documento verificar la integridad del mismo haciendo uso del código hash; de esta manera se lograría optimizar tiempo y recursos humanos en las universidades y estudiantes.

- El principal resultado de las encuestas para la obtención de los requerimientos funcionales se expresó en la necesidad de automatizar el proceso de emisión de documentos académicos. Actualmente los tiempos de entrega de los documentos académicos son mayores a dos días en el mejor de los casos y en el peor de los casos puede llegar a los diez días, según datos de la encuesta. En el contexto de la pandemia del COVID-19, se hacen indispensables sistemas que automaticen la emisión de documentos, logrando así evitar aglomeraciones innecesarias; formando parte de un protocolo de salud a nivel del país cuyo fin sea evitar posibles contagios de virus.

- En la etapa de diseño del prototipo se encontró un problema: Cómo poder imprimir el código hash en el documento que emite el prototipo, ya que si se modifica el documento académico se cambia el código hash. La solución que presenta este prototipo es generar dos tipos de documentos académicos, el original y el entregable. El documento académico entregable va a ser el mismo documento original pero en su pié de página va a tener el hash del documento original. El receptor del documento entregable va a ingresar al prototipo, digitar el código que viene impreso en el documento y descargarlo, este documento que se descarga es el original, si a este se le saca el código hash va a coincidir con el que viene impreso en el pié de pagina del documento entregable.

- La elección de MongoDB frente a una base de datos relacional trajo varias ventajas. Primero hizo posible que el modelado de datos sea rápido y efectivo, ya que el modelado con MongoDB es mas natural al no estar ligado a una estructura fija y no se debe estar analizando relaciones de una tabla con otra. Segundo MongoDB permite a los datos no tener una estructura fija, y esta característica es la necesaria para el prototipo, debido a que si el sistema se integra en un futuro con algunas universidades cada una de estas va a tener su propia estructura. Finalmente MongoDB realiza las consultas a la base de datos de una manera más rápida. [37]

- El uso de Laravel en la codificación de este prototipo permitió un desarrollo ágil, ordenado y eficiente, debido a que este *framework* incluye diversas funcionalidades entre las que destacan sus características de seguridad proveyendo a sus aplicaciones protección a ataques del tipo *cross-site request forgery*, protección de rutas mediante

middlewares y protección a ataques de inyección SQL. Esto permite al desarrollador enfocarse netamente en la lógica de la aplicación logrando desarrollar aplicaciones robustas en menos tiempo.

- ReCAPTCHAv3 contribuyó a tener un control total de las peticiones mal intencionadas de *bots* sin la intervención del usuario, ya que si ReCAPTCHAv3 detecta que una solicitud proviene de un *bot* notifica al sistema, este a su vez toma la acción más conveniente, por ejemplo bloquear la solicitud. Desde el punto de usuario la aplicación que maneje ReCAPTCHAv3 generará una mejor experiencia de usuario versus las versiones anteriores, ya que el usuario no debe estar armando rompecabezas o identificando letras, por ende será la aplicación fácil de usar y más atractiva al usuario final.

- Basado en la encuesta de satisfacción , se concluye que la aplicación le resulta amigable al usuario, ya que la totalidad de los encuestados dieron una calificación mayor a 4 sobre 5 en usabilidad. El rendimiento de la aplicación es excelente según el 85,7 % de los encuestados además el tiempo de entrega de los documentos para el 71.4 % de los encuestados fue menor a 5 minutos lo que significa que el sistema tiene un rendimiento positivo.

4.2. RECOMENDACIONES

- Se recomienda tomar el aplicativo web como punto de partida para la elaboración de un sistema final el cual esté integrado completamente con las universidades para generación y verificación de la integridad de los documentos académicos. Para que este prototipo se pueda implementar en alguna universidad bastaría solo con consumir la información de la base de datos de la universidad ya sea mediante web service o generando un acceso limitado a la misma. Estos datos alimentarían a la base de datos MongoDB. Por consiguiente, el sistema ya entregará información real.

- En base a mi tesis, se puede resolver el mismo problema haciendo uso de la tecnología *blockchain* en la cual la información se almacena en bloques, cada bloque además de tener su hash tiene el hash del bloque que le precede. Si un intruso edita esta información es fácilmente detectada por el sistema ya que la información la almacena de una forma descentralizada lo que ayuda a verificar la integridad de los datos. El nivel de seguridad de esta tecnología es tan alto que actualmente uno de los principales usos es la transferencia de dinero. Si los documentos se generan usando esta Tecnología también habría la certeza de que la integridad de los documentos no se alteraría.

- Es de vital importancia la investigación previa, identificar cada requerimiento, cada necesidad del usuario ya que así se podrá diseñar y dimensionar de manera correcta la aplicación, siempre pensando en futuro y permitiendo a la aplicación escalar con facilidad. Esto evita problemas de rendimiento en el sistema y genera una excelente experiencia de usuario.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] Y. Amghar and R. Chbeir, “Advanced management of documents integrity in an intranet environment, acs,” in *IEEE Conference: International Conference on Computer Systems and Applications, Tunisia*, 2003.

[2] B. Donohue, “¿Qué Es Un Hash Y Cómo Funciona? | Blog oficial de Kaspersky,” Apr. 2014. [Online]. Disponible en: https://latam.kaspersky.com/blog/que-es-un-hash-y-como-funciona/2806/

[3] S. Bakhtiari, R. Safavi-Naini, J. Pieprzyk *et al.*, “Cryptographic hash functions: A survey,” Citeseer, Tech. Rep., 1995.

[4] C. Lynch, “Authenticity and integrity in the digital environment: An exploratory analysis of the central role of trust,” *Museums in a digital age*, pp. 319–320, 2000.

[5] A. Sarasa, “Introducción a las bases de datos nosql usando mongodb,” *Introducción a las bases de datos NoSQL usando MongoDB*, pp. 17–66, 2016.

[6] L. Von Ahn, M. Blum, N. J. Hopper, and J. Langford, “Captcha: Using hard ai problems for security,” in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2003, pp. 294–311.

[7] G. Shaw, “Digital document integrity,” 2000.

[8] S. B. Escalona and L. V. Inclán, “Funciones resúmenes o hash,” *Revista Telemática*, vol. 10, no. 1, 2012.

[9] B. Preneel, “El estado de las funciones hash,” in *Criptología y seguridad de la Información: Actas de la VI Reunión Española sobre Criptología y Seguridad de la Información, VI RECSI, Tenerife, Islas Canarias, 14-16 septiembre de 2000*. RA-MA, 2000, pp. 3–38.

[10] R. Palacios and V. Delgado, “Introducción a la criptografía: tipos de algoritmos,” in *anales de mecánica y electricidad*, 2006.

[11] I. Akrout, A. Feriani, and M. Akrout, “Hacking google recaptcha v3 using reinforcement learning,” *arXiv preprint arXiv:1903.01003*, 2019.

[12] “reCAPTCHA v3,” library Catalog: developers.google.com. [Online]. Disponible en: https://developers.google.com/recaptcha/docs/v3?hl=es-419

[13] K. Calvo, J. Durán, E. Quirós, and E. Malinowski, “Mongodb: alternativas de implementar y consultar documentos,” in *IX Congreso Internacional de Computación y Telecomunicaciones, COMTEL, Lima*, 2017, pp. 48–49.

[14] “Introduction to MongoDB — MongoDB Manual,” library Catalog: docs.mongodb.com. [Online]. Disponible en: https://docs.mongodb.com/manual/core/data-modeling-introduction

[15] “Indexes — MongoDB Manual,” library Catalog: docs.mongodb.com. [Online]. Disponible en: https://docs.mongodb.com/manual/indexes

[16] “Unique Indexes — MongoDB Manual,” library Catalog: docs.mongodb.com. [Online]. Disponible en: https://docs.mongodb.com/manual/core/index-unique

[17] J. Segers, “jenssegers/laravel-mongodb,” Jun. 2020, original-date: 2013-03-31T14:31:04Z. [Online]. Disponible en: https://github.com/jenssegers/laravel-mongodb

[18] “PHP: Installing the MongoDB PHP Driver with PECL - Manual.” [Online]. Disponible en: https://www.php.net/manual/en/mongodb.installation.pecl.php

[19] “AWS | Elastic compute cloud (EC2) de capacidad modificable en la nube,” library Catalog: aws.amazon.com. [Online]. Disponible en: https://aws.amazon.com/es/ec2/

[20] Y. D. González and Y. F. Romero, “Patrón modelo-vista-controlador.” *Revista Telemática*, vol. 11, no. 1, pp. 49–50, 2012.

[21] “Documentation - Laravel - The PHP Framework For Web Artisans.” [Online]. Disponible en: https://laravel.com/docs/7.x

[22] “Composer.” [Online]. Disponible en: https://getcomposer.org/

[23] “Git is a free and open source distributed version control system.” [Online]. Disponible en: https://git-scm.com/

[24] “Build software better, together,” library Catalog: github.com. [Online]. Disponible en: https://github.com

[25] “Blade Templates - Laravel - The PHP Framework For Web Artisans.” [Online]. Disponible en: https://laravel.com/docs/5.8/blade

[26] E. Guzmán, “Requerimientos Funcionales del Software - Formularios de Google.” [Online]. Disponible en: https://docs.google.com/forms/d/1dow83cgsdrrc0g58Qhuv_hFU2bqOjbbeHdQfrISAMNI/edit#responses

[27] admin@dbschema.com, “DbSchema: The Best Visual Database Designer & GUI Tool,” library Catalog: dbschema.com. [Online]. Disponible en: https://dbschema.com/

[28] “Authentication - Laravel - The PHP Framework For Web Artisans.” [Online]. Disponible en: https://laravel.com/docs/5.8/authentication

[29] “Día dibuja los diagramas estructurados: Libre de Windows, Mac OS X y Linux versión del popular programa de código abierto.” [Online]. Disponible en: http://dia-installer.de/index.html.es

[30] “Online Mockup, Wireframe & UI Prototyping Tool · Moqups,” library Catalog: moqups.com. [Online]. Disponible en: https://moqups.com

[31] “Tipos de instancias de Amazon EC2 - Amazon Web Services,” library Catalog: aws.amazon.com. [Online]. Disponible en: https://aws.amazon.com/es/ec2/instance-types/

[32] nnamuhcs, “Requisitos de hardware para el equipo de destino,” library Catalog: docs.microsoft.com. [Online]. Disponible en: https://docs.microsoft.com/es-es/windows-server-essentials/install/hardware-requirements-for-the-target-computer

[33] “Área del Cliente - Freenom.” [Online]. Disponible en: https://my.freenom.com/clientarea.php

[34] “reCAPTCHA console.” [Online]. Disponible en: https://www.google.com/u/1/recaptcha/admin/site/351660288

[35] “Visual Studio Code - Code Editing. Redefined,” library Catalog: code.visualstudio.com. [Online]. Disponible en: https://code.visualstudio.com/

[36] E. Guzmán, “Encuesta de satisfacción del sistema prototipo para la emisión y verificación de documentos académicos.” library Catalog: docs.google.com. [Online]. Disponible en: https://docs.google.com/forms/d/e/1FAIpQLSeYBDe8kLdP1Z3TNbnQfepGgD5I3HOUE5dKtyBI9n00LRXlJQ/viewform?usp=embed_facebook

[37] S. H. Aboutorabiª, M. Rezapour, M. Moradi, and N. Ghadiri, “Performance evaluation of sql and mongodb databases for big e-commerce data,” in *2015 International Symposium on Computer Science and Software Engineering (CSSE)*. IEEE, 2015, pp. 4–7.

5. ANEXOS

Los siguientes anexos se encuentran en formato digital:

ANEXO A. UsuarioPublico.php

ANEXO B. UsuarioEstudiante.php

ANEXO C. documentlist.blade.php, generate.blade.php, home.blade.php, welcome.blade.php

5. ORDEN DE EMPASTADO