ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

AUTOMATIZACIÓN DEL PROCESO DE CONVALIDACIÓN DE PRÁCTICAS PREPROFESIONALES

VALIDACIÓN DEL FORMULARIO FCP_001A MEDIANTE UNA APLICACIÓN WEB

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN TELECOMUNICACIONES

ERICK FRANCISCO LEMA CHANGOLUISA

erick.lema01@epn.edu.ec

DIRECTOR: JOSÉ ADRIÁN ZAMBRANO MIRANDA

jose.zambrano@epn.edu.ec

DMQ, Octubre 2022

CERTIFICACIONES

Yo, Erick Francisco Lema Changoluisa declaro que el trabajo de integración curricular 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

.

Erick Francisco Lema Changoluisa

Certifico que el presente trabajo de integración curricular fue desarrollado por Erick Francisco Lema Changoluisa, bajo mi supervisión.

José Adrián Zambrano Miranda
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Erick Francisco Lema Changoluisa

José Adrián Zambrano Miranda

DEDICATORIA

(Opcional)

AGRADECIMIENTO

(Opcional)

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

En el presente trabajo de titulación se expone el diseño e implementación del prototipo de aplicación web ConvalidaciónCPPP, el cual tiene el objetivo de validar y automatizar el llenado del formulario FCP_001A dentro del proceso de convalidación de prácticas preprofesionales de los estudiantes de Telecomunicaciones en la Facultad de Ingeniería Eléctrica y Electrónica de la Escuela Politécnica Nacional.

El presente trabajo se divide en tres capítulos principales con el objetivo de llevar un orden cronológico de las actividades desarrolladas en este proyecto.

En el primer capítulo se describen aspectos generales del proyecto: objetivo general, objetivos específicos, alcance del proyecto y marco teórico donde se describen los conceptos necesarios para el desarrollo del prototipo ConvalidaciónCPPP.

En el segundo capítulo de metodología se realiza un análisis del proceso de convalidación y del formulario FCP_001A con el fin de determinar los requerimientos para diseñar e implementar de la aplicación web.

En el último capítulo se presentan los resultados obtenidos de la aplicación web y las pruebas de funcionamiento realizadas con personas involucradas en el proceso de convalidación, además en este capítulo se presentan las conclusiones y recomendaciones del trabajo de titulación.

Adicional, como anexos del presente trabajo de titulación, se tienen: Instalación de requerimientos de Laravel, implementación de Api university-rest, implementación Api sri, código de proyectos en Laravel, implementación de prototipo ConvalidaciónCPPP, pruebas de funcionamiento realizadas por el desarrollador, encuestas hechas a estudiantes y enlaces de grabaciones de pruebas realizadas con algunos estudiantes de la EPN.

PALABRAS CLAVE: Prácticas preprofesionales, Convalidación, Laravel, API-REST, Frontend, Backend.

ABSTRACT

In this degree work, the design and implementation of the web application prototype ConvalidaciónCPPP is presented, which has the objective of validating and automating the filling of the form FCP_001A within the process of validation of pre-professional practices of Telecommunications students in the Faculty of Electrical and Electronic Engineering of the National Polytechnic School.

This work is divided into three main chapters to keep a chronological order of the activities developed in this project.

The first chapter describes general aspects of the project: general objective, specific objectives, scope of the project and theoretical framework where the concepts necessary for the development of the prototype ConvalidaciónCPPP are described.

In the second chapter of the methodology, an analysis of the validation process and the FCP_001A form is carried out to determine the requirements for the design and implementation of the web application.

The last chapter presents the results obtained from the web application and the functional tests performed with people involved in the validation process, and the conclusions and recommendations of the degree work are presented in this chapter.

Additionally, as annexes of this degree work, we have: Laravel requirements installation, university-rest Api implementation, sri Api implementation, Laravel project code, ConvalidaciónCPPP prototype implementation, functional tests performed by the developer, surveys made to students and links to test recordings made with some EPN students.

KEYWORDS: Pre-professional internship, Validation, Laravel, API-REST, Frontend, Backend.

1 INTRODUCCIÓN

La convalidación de prácticas preprofesionales en la Escuela Politécnica Nacional está disponible para aquellos estudiantes que están cursando los últimos niveles de las carreras de tercer nivel, que a causa de las restricciones de emergencia sanitaria y limitaciones de movilidad causadas por la pandemia por COVID19 no pudieron realizar sus prácticas preprofesionales [1].

Los estudiantes podrán solicitar la convalidación de actividades extracurriculares, las mismas que, sin ser parte del plan curricular de la carrera, tienen como objetivo complementar la formación integral del estudiante, contribuir a la aplicación de conocimientos y al desarrollo de habilidades profesionales. Estas actividades extracurriculares pueden estar relacionadas con aspectos académicos, sociales, deportivos, culturales o artísticos.

La Escuela Politécnica Nacional actualmente maneja un sistema manual para realizar la convalidación de práctica preprofesionales, el mismo que se describe en el documento “Procedimiento Convalidación Actividades Extracurriculares” [2] emitido por la dirección de docencia. En el proceso para la convalidación de actividades extracurriculares, existe la intervención de varios actores como son: el estudiante, el subdecano, el tutor, la comisión de prácticas preprofesionales y el decano. Cada uno de los actores intervienen en diferentes etapas de este proceso, invirtiendo una gran cantidad de tiempo en cada solicitud y, en muchas ocasiones, debido a errores cometidos en los formularios, hay que rehacer el proceso lo que puede tomar un tiempo demasiado largo e innecesario.

Es por esto que en la Institución surge la necesidad de automatizar este proceso mediante una aplicación web, para lo cual se deberán analizar los campos del formulario FCP_001A, considerar su variabilidad y las respectivas validaciones que se pueden implementar mediante el cruce de información tanto interna como externa a la EPN. Como solución a este problema, se propone en este proyecto el prototipo de aplicación web “ConvalidaciónCPPP”.

Los objetivos de este trabajo, su alcance y limitaciones se detallan a continuación. Además, se presentan los conceptos de las herramientas necesarias para desarrollar e implementar el prototipo de aplicación web.

1.1 OBJETIVO GENERAL

Implementar el prototipo de una aplicación web para validar y automatizar el llenado del formulario FCP_001A en el proceso de convalidación de prácticas preprofesionales de los estudiantes de Telecomunicaciones en la FIEE.

1.2 OBJETIVOS ESPECÍFICOS

1. Analizar el proceso de convalidación de prácticas preprofesionales descrito en el documento “Procedimiento Convalidación Actividades Extracurriculares” y determinar los requerimientos para el diseño de la aplicación web.
2. Analizar el formulario FCP_001A y determinar cómo se realizará el llenado de cada uno de sus campos mediante la aplicación web.
3. Diseñar un boceto de la interfaz de la aplicación web con base en los requerimientos establecidos para cada rol de usuario.
4. Diseñar una base de datos para almacenará la información de todos los formularios.
5. Desarrollar la aplicación web en el lenguaje de programación PHP usando el framework Laravel.
6. Verificar el funcionamiento de la aplicación web a través de pruebas juntamente con colaboradores de la institución.

1.3 ALCANCE

Como solución al problema antes mencionado, se propone la aplicación “ConvalidaciónCPPP” el cual es un prototipo que busca cumplir con dos diferentes propósitos que forman parte del mismo proyecto, el primero de ellos es validar y automatizar el llenado del formulario FCP_001A y el segundo es la elaboración de una base de datos con el fin de poder visualizar de una forma clara y organizada los registros de todos los formularios que han terminado con éxito el proceso de convalidación.

Los propósitos mencionados, fueron presentados en el plan aprobado previo al desarrollo de este proyecto, en el cual se establecen dos diferentes componentes. Por esta razón, la aplicación web “ConvalidaciónCPPP” será implementada por dos desarrolladores, pero cada uno implementará solamente los módulos necesarios para cumplir con su propósito. Bajo esta consideración, el presente trabajo se centrará en describir y desarrollar los módulos necesarios para cumplir con el primer propósito, que es, validar y automatizar el llenado del formulario FCP_001A en el proceso de convalidación de prácticas preprofesionales de los estudiantes de Telecomunicaciones en la FIEE.

Es importante además mencionar que, en el momento en el que se desarrolló el plan de este proyecto, se trabajó con el procedimiento y formulario vigentes en el periodo académico 2021-B; debido a esto, el prototipo que se presenta en este trabajo no considera los cambios realizados en el proceso a seguir para la convalidación de prácticas preprofesionales del periodo académico actual.

Existen varias consideraciones adicionales que hay que mencionar en el desarrollo del presente proyecto, siendo estas las siguientes:

- El proyecto será desarrollado en un ambiente local y controlado, es decir que no se hará uso de bases de datos externas para llenar de forma automática algunos campos del formulario.
- Se creará una API-REST llamada "*university-rest"* la cual contendrá información ficticia de estudiantes y profesores de la carrera de Telecomunicaciones. Con la creación de esta api-rest se buscará simular el cruce de información con la aplicación web, lo que permitirá implementar el llenado automático de los campos del formulario que requieren información tanto de estudiantes como de profesores.
- De igual forma se creará también una API-REST llamada "*sri"* con la cual se simula el cruce de información de empresas registradas en el SRI, esto permitirá llenar de forma automática los datos correspondientes a instituciones nacionales en el formulario FCP_001A.
- La aplicación web ConvalidaciónCPPP y las api-rest antes mencionadas serán desarrolladas en el lenguaje de programación PHP usando el framework Laravel.
- Se creará una base de datos propia de la aplicación web en MySQL, donde se almacenará la información de los estudiantes y profesores que harán uso de la aplicación, y también la información de cada uno de los formularios.
- Al tratarse de un prototipo es importante recalcar que el proyecto no será puesto en producción, es decir que únicamente será probado en un ambiente local y controlado.

1.4 MARCO TEÓRICO

En esta sección se presentarán los conceptos de las herramientas usadas para la implementación de la aplicación web “ConvalidaciónCPPP”, estos elementos son: el lenguaje de programación php, framework Laravel, Api-rest con Laravel y servidor Laragon; los cuales representan la base teórica para el desarrollo del prototipo.

1.4.1 LENGUAJE DE PROGRAMACIÓN PHP

1.4.1.1 Definición de PHP

PHP (Procesador de hipertexto) es un lenguaje de código abierto, utilizado a menudo en el desarrollo web debido a que puede ser incrustado en HTML [3]. De forma sencilla PHP se puede describir con las siguientes tres características:

- PHP es un preprocesador de hipertexto (HTML).
- PHP es un lenguaje de programación que se ejecuta en el servidor.
- PHP es un lenguaje de programación interpretado de secuencias de comandos.

Para entender de mejor manera como funciona PHP, se puede realizar una comparación con el lenguaje de programación JavaScript [4]. JavaScript al igual que PHP es un lenguaje de secuencias de comandos, pero la gran diferencia entre ellas es dónde se ejecuta el código. JavaScript recibe el código del lado del cliente y se ejecuta allí, por lo tanto, cualquiera usuario puede ver qué tipo de código está escrito.

PHP, por otro lado, ejecuta el código en el lado del servidor, es decir, que del lado del cliente solo se recibe el resultado y no es posible ver de qué tipo de código se derivó ese resultado. Mediante el uso de PHP, es posible una interacción eficiente y confiable con HTML[4].

1.4.1.2 Script en PHP

Un script es un programa que se ejecuta en un servidor dando respuesta a una solicitud hecha por un navegador. Un script PHP básicamente es un conjunto de expresiones, que le dicen al intérprete de PHP que realice una acción como por ejemplo: agregar dos números o mostrar información en la pantalla. En la estructura de un script hay que saber que cada expresión comienza en una nueva línea y termina con un punto y coma [3].

En el código 1.1 se presenta un ejemplo muy sencillo de un script en php que, al ejecutarse, mostrará una línea en la pantalla con el texto: "¡Hola mundo!”.

```php
<?php
    print("Hola mundo");
```

Código 1.1. Ejemplo de script en PHP

Es importante considerar que todos los scripts PHP deben comenzar con la siguiente línea: "*<?php"* de esta manera se indica al servidor web que a continuación se encuentra código PHP.

1.4.1.3 Funcionamiento de PHP

Como ya fue mencionado anteriormente PHP es un lenguaje de programación que ejecuta scripts en el servidor, esto significa que php estrictamente funciona en un servidor y no en el cliente como en el caso de HTML o JavaScript. PHP ejecuta scripts para atender las peticiones de los clientes, para entender como estas peticiones son atendidas, en la Figura 1.1 se presenta el funcionamiento de PHP.

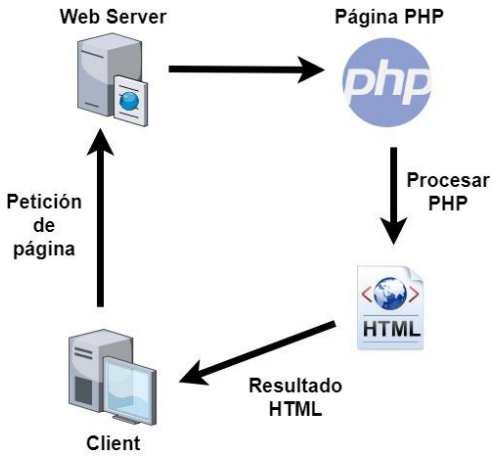

Figura 1.1. Funcionamiento de PHP

Funcionamiento de PHP [5]:

1. Desde una máquina cliente, el usuario envía una solicitud al servidor.
2. El servidor web recibe la solicitud del cliente y realiza la búsqueda de la página para acceder a ella.
3. Si la página contiene código PHP, el servidor procesa el código en un servidor y remite el resultado obtenido hacia el cliente en formato HTML.
4. El cliente recibe la página web en formato HTML y visualiza los resultados desde su navegador.

Conociendo ya el funcionamiento de PHP se puede entender por qué es llamado el preprocesador de hipertexto.

1.4.2 FRAMEWORK LARAVEL

1.4.2.1 ¿Qué es un framework?

Un framework de desarrollo web es una plataforma diseñada para implementar páginas y aplicaciones web facilitando el desarrollo e integración de diferentes componentes de un proyecto. Un framework solo contienen módulos básicos, y el desarrollador es quien debe implementar los componentes específicos del proyecto en función de sus necesidades. De esta manera, además de lograr una alta velocidad de desarrollo, también se obtiene un alto rendimiento y confiabilidad de soluciones [6].

1.4.2.2 Definición del framework Laravel

Laravel es un framework PHP gratuito de propósito general y uno de los motores PHP más populares; está escrito en PHP y crea código basado en el mismo lenguaje. Laravel ha sido creado para desarrollar sitios y aplicaciones web complejos, permitiendo simplificar aspectos como: el enrutamiento, la autenticación, las sesiones, el almacenamiento en caché, el trabajo con la base de datos, entre otros [7].

Las aplicaciones desarrolladas en Laravel proporcionan un mejor rendimiento en comparación de aquellas aplicaciones creadas con otros frameworks [6]. Algunas de las características más importantes de Laravel se presentan a continuación, con el fin de familiarizarse con este framework y su funcionamiento.

1.4.2.3 Estructura de las aplicaciones en Laravel

Laravel tiene una estructura ordenada para todas sus aplicaciones, esta estructura se presenta en la Figura 1.2 y el contenido de cada directorio será descrito a continuación [8].

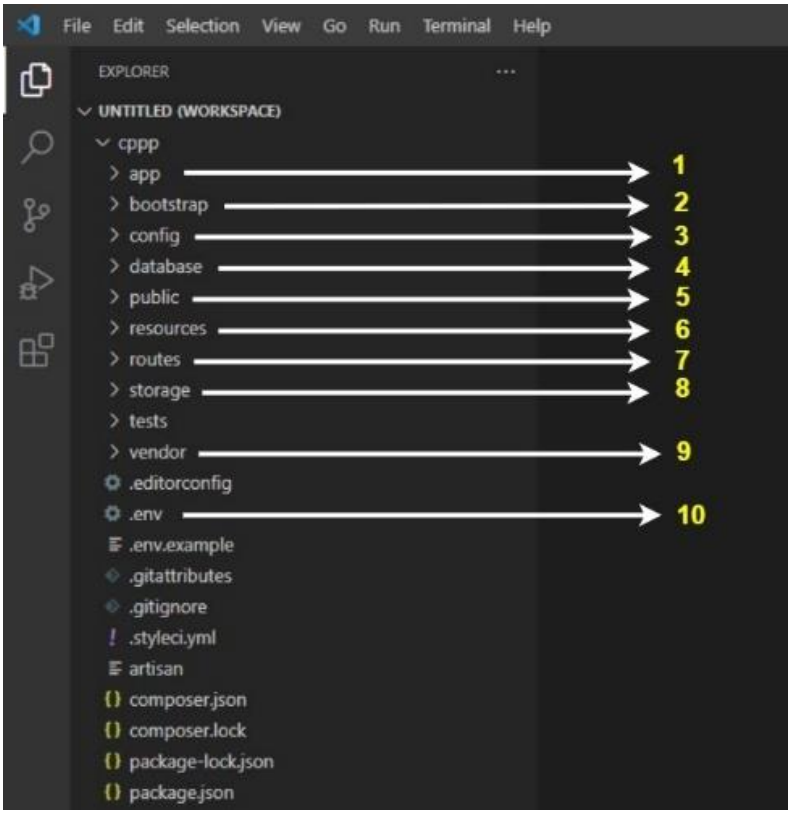

Figura 1.2. Estructura de los directorios de Laravel

1. **Carpeta de aplicaciones:** esta carpeta contiene el núcleo de todas las aplicaciones. Aquí se ubican las clases, funciones, modelos, controladores y, en general, la lógica de la aplicación [8].

2. **Carpeta de arranque:** aquí es donde se almacenan los archivos que cargan el framework y configuran la carga automática. También contiene una carpeta de caché en bootstrap el mismo que almacena archivos que se han generado por el Framework para optimizar el rendimiento[8].

3. **Carpeta de configuración:** en esta carpeta se encuentran varios archivos necesarios para configurar la base de datos, sesiones, correos electrónicos, aplicaciones, servicios, etc [8].

4. **Carpeta de base de datos:** contiene migraciones y clases para completar los datos iniciales de su base de datos. [8].

5. **Carpeta pública:** almacena el archivo index.php, el cual representa el punto de entrada de todas las solicitudes que llegan a las aplicaciones. Además, esta carpeta contiene recursos como imágenes, JavaScript, CSS [8].

6. **Carpeta de recursos:** la carpeta contiene todas las vistas creadas en la aplicación y recursos sin procesar como: CSS, SASS, JavaScript [8].

7. **Carpeta Rutas:** aquí se almacenan todas las rutas definidas en la aplicación. De forma predeterminada, Laravel tiene cuatro archivos de ruta integrados [8]:

- web.php: en este archivo se describen todas las rutas asociadas con la interfaz web.
- api.php: en este archivo se especifican todas las rutas asociadas con la API. Las rutas de este archivo son colocadas en el grupo middleware y las solicitudes realizadas a la aplicación a través de estas rutas están destinadas a autenticarse con tokens.
- console.php: en este archivo se pueden declarar nuevos comandos artisan, propios de cada desarrollador.
- channels.php: en este archivo se registran todos los canales de transmisión de eventos compatibles con su aplicación.

8. **Carpeta de almacenamiento:** contiene plantillas Blade, archivos de sesión, cachés de archivos y otros. Esta carpeta se divide en subcarpetas app, framework y logs [8].

- La **carpeta app** puede almacenar cualquier archivo generado por su aplicación.
- La **carpeta framework** contiene los archivos y el caché creados por el framework.
- La **carpeta de logs** almacena todos los archivos de registro de la aplicación.

La **carpeta storage/app/public** se puede usar para almacenar archivos, que deberían estar disponibles para toda la aplicación. Para poder hacer uso de esta carpeta se debe crear un acceso directo en la carpeta public/ que apunte a la carpeta storage/app/public. Este enlace se puede crear con el comando "php artisan storage:link" [8].

9. **Carpeta de proveedores:** la carpeta del proveedor contiene las dependencias de COMPOSER. Paquetes de Laravel que ayudan a ejecutar su aplicación [8].

10. **Archivo .env:** es el archivo de configuración de la aplicación, en la cual se describen parámetros como: conexión con la base de datos, duración de la sesión, datos para conectarse a las direcciones de correo electrónico[9]. Este archivo de configuración es usado cuando el proyecto está en fase de pruebas y desarrollo donde principalmente se configura la conexión con la base de datos.

1.4.2.4 Arquitectura MVC

Laravel permite crear aplicaciones y sitios web basados en MVC (modelo-vista-controlador) [6]. Esta es una variante de arquitectura, en la que los componentes del programa se dividen en tres partes como se muestra en la Figura 1.3:

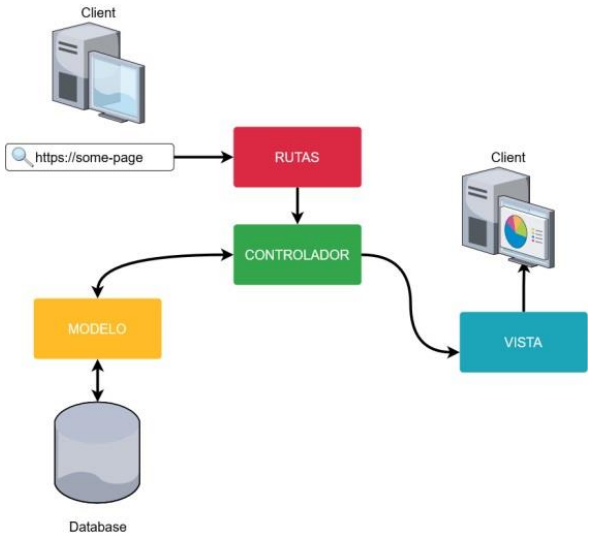

Figura 1.3. Arquitectura Modelo-Vista-Controlador

- **Modelo:** proporciona datos y métodos para realizar consultas a la base de datos. Responde a las peticiones del controlador devolviendo datos y/o cambiando su estado. Los modelos no interactúan directamente con el usuario [10].
- **Vista:** es la responsable de mostrar la información a los usuarios. Los mismos datos pueden representarse de diferentes maneras y formatos [10].
- **Controlador:** Proporciona el enlace entre usuario y sistema utilizando un modelo y una vista para dar respuesta a las peticiones de los usuarios. El controlador filtra los datos recibidos y realiza un proceso de autorización, es decir, comprueba los derechos del usuario para realizar acciones o recibir información [10].

Esta arquitectura ha demostrado ser una solución para estructurar eficazmente las aplicaciones y páginas web, permitiendo separar la lógica de la aplicación de su parte visual. MVC hace que el código sea más legible y el proceso de desarrollo sea más cómodo, separando el trabajo de los desarrolladores de front-end y back-end, los cuales se explicarán a continuación.

1.4.2.5 Frontend de Laravel

Frontend se define como la interfaz con la que interactúa el usuario de una aplicación. En el caso de un sitio web, es generado y mostrado por un navegador que funciona con HTML , CSS y JavaScript [11].

Laravel es principalmente un framework de desarrollo de backend; sin embargo, ofrece algunas funcionalidades de frontend bastante útiles en el desarrollo de aplicaciones web. Blade es un motor de plantillas que viene con Laravel y no impide usar código PHP en sus vistas [11]. Los archivos Blade View tienen una extensión y generalmente se almacenan en el formato *.blade.php* en el directorio resources/views. Los dos beneficios principales de usar Blade son la herencia y extensión de plantillas, las cuales se explicarán a continuación.

- **Herencia de plantilla de Blade**

Debido a que las aplicaciones web suelen mantener el mismo diseño en todas las páginas, es útil definir una sola vista Blade. De esta vista conocida como maestra se podrá heredar el diseño a todas las demás vistas de la aplicación, esto con la finalidad de no repetir el mismo código en cada ellas y solo centrarse en el contenido de cada vista [10]. A continuación, se muestra un ejemplo de una vista maestra de Blade.

```
<html>
    <head>
        <title> @yield('title')</title>
    </head>
    <body>
        <div class="container">
            @yield('content')
        </div>
    </body>
</html>
```

Código 1.2. Herencia de plantilla de blade

En el código 1.2, se puede observar la estructura típica de un archivo HTML. Sin embargo, presenta la directiva @yield que no es propia de html [10]. La directiva @yield es usada para especificar el lugar donde se insertará un contenido en la vista que hereda la plantilla.

- **Extensión de diseño**

En una vista secundaria, al usar la directiva de Blade @extends se especifica qué diseño se va a "heredar" a la vista secundaria. Las vistas que amplían el diseño de Blade pueden insertar nuevo contenido en secciones del diseño mediante la directiva @section [10]. En el código 1.3, se presenta un ejemplo de extensión de plantilla y de inserción de contenido.

```
@extends('app')
@section('content')
    <div class="container w-50 border p-4 mt-4">
<h1>Hola mundo!!</h1>
    </div>
@endsection
```

Código 1.3. Extensión de diseño

1.4.2.6 Backend de Laravel

Backend es todo aquello que se ejecuta en un servidor y hace uso de cualquier herramienta disponible en el mismo. Esto significa que se puede usar cualquier lenguaje de programación universal como: Ruby, PHP, Python, Java o bash. Además, se puede usar sistemas para administrar bases de datos como: PostgreSQL, MySQL, MongoDB, Cassandra, Redis o Memcached [11].

Laravel es un framework PHP del lado del servidor y proporciona todas estas características de backend, con las cuales se puede crear aplicaciones completas como, por ejemplo, cuentas de usuario, exportaciones, gestión de pedidos, etc. [7]. Las funciones más importantes del backend de Laravel que se han utilizado en el desarrollo de este proyecto, se presentan a continuación:

- **Modelos y ORM Eloquent**

ORM es una tecnología de programación para vincular un lenguaje de programación y una base de datos. El uso del ORM permite acelerar el desarrollo de aplicaciones web y conexiones con bases de datos. Existen muchas implementaciones de ORM para PHP, pero Laravel tiene la suya propia. Eloquent es el ORM de Laravel y funciona según el esquema ActiveRecord, según el cual cada una de las tablas de la base de datos corresponde a un Modelo [10]. Gracias a Eloquent ORM, se puede trabajar con bases de datos sin tener que escribir consultas SQL complejas.

- **Conexión con bases de datos**

Un componente importante de cualquier aplicación web es la base de datos. Laravel admite cuatro sistemas de bases de datos: SQL Server, SQLite, Postgresy MySQL [12].

Laravel se conecta a la base de datos de una forma sencilla mediante el archivo de configuración *database.php* el cual se encuentra en el directorio *config/*, en este, se pueden definir todas las conexiones a las bases de datos y también especificar la conexión predeterminada [12].

- **Migraciones**

Las migraciones en Laravel son un conjunto de características para el control de la base de datos de una aplicación. Permiten a los desarrolladores inicializar, destruir y recrear rápidamente la base de datos, ayudando a minimizar los problemas de inconsistencia que pueden ocurrir cuando varios desarrolladores trabajan en la misma aplicación. Todas las migraciones están almacenadas en el directorio *database/migrations* [13].

Para crear una migración, se lo hace mediante comando Artisan "make:migration" [13]

Las migraciones contienen métodos llamados "up" y "down". El primero de ellos es usado para agregar tablas, columnas e índices a la base de datos. Y la segunda cancela las operaciones realizadas por el primero. Ambos métodos admiten el uso del constructor de estructuras Laravel [13]. En el código 1.4, se presenta un ejemplo del uso de estos métodos mediante la creación de una tabla llamada "User":

```
public function up() {
    Schema::create('users', function (Blueprint $table) {
        $table->id();
        $table->string ('name')->nullable();
        $table->string ('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->rememberToken();
        $table->timestamps();
    });
}
public function down() {
    Schema::dropIfExists('users'); }
}
```

Código 1.4. Migraciones en Laravel

Para ejecutar todas las migraciones, se usa el comando Artisan: *php artisan migrate* [13].

- **Seeders**

Laravel permite llenar fácilmente la base de datos con información de prueba utilizando la clase seed, las cuales se almacenan en el directorio *database/seeders* [14]. Se puede nombrar la clase de forma arbitraria, pero se recomienda seguir una convención de nomenclatura similar a *UsersTableSeeder*. En todos los archivos seeder se encuentra la función *run,* donde se escriben las instrucciones para llenar una tabla [14], a continuación, se muestra un ejemplo del contenido de la función *run* del seeder *UsersTableSeeder*.

```
public function run()
{
    $id = DB::table('users')->insertGetId([
        'name' => 'Erick Lema',
        'username' => '123456789',
        'email' => 'erick@gmail.com',
        'password' => Hash::make('password'),
    ]);
}
```

Código 1.5. Seeders con Laravel

Para ejecutar todos los seeders, se hace uso del comando Artisan: *php artisan db:seed* [14]. De esta manera se llenará la tabla de acuerdo con las instrucciones establecidas en los seeders.

- **Controladores**

En Laravel, un controlador es una clase que se encuentra en el directorio *app/HTTP/Controllers* [15]. Los métodos de esta clase están asociados a rutas específicas y se encargan de procesarlas. Dentro de los controladores, se implementa la lógica de la aplicación, permitiendo realizar acciones como: mostrar, crear, actualizar y eliminar un registro [15]. Por ejemplo, la clase UserController manejará todas las solicitudes entrantes relacionadas con el usuario, como ver, crear, actualizar y eliminar un registro de usuarios.

Los controladores tienen tres funciones principales [16].

- Procese la información recibida de la ruta al modelo.
- Mostrar la información recibida del modelo en la vista.
- Validar los datos.

Específicamente, puede acceder a la base de datos desde el modelo, procesar los datos adquiridos con el controlador y pasarlos a la vista, o pasar los datos de la vista al modelo y guardarlos en la base de datos.

1.4.2.7 Seguridad de Laravel

La seguridad es una consideración importante al desarrollar aplicaciones web. Los desarrolladores deben asegurarse de que los datos de los usuarios estén protegidos. Por esta razón, Laravel proporcionan una variedad de mecanismos para asegurar un sitio web. Algunos de ellos se enumeran a continuación.

- Laravel proporciona un sistema de autenticación simple y eficiente listo para usar. Incluso proporciona control de acceso a los recursos. Esto le permite limitar el acceso de usuarios no autorizados a ciertas funcionalidades de la aplicación [17] .
- El código base del framework está protegido contra amenazas como la falsificación de solicitudes entre sitios (CSRF). Esto protege a los usuarios de la pérdida de datos importantes [17].
- Laravel tiene mecanismos incorporados para proteger contra la inyección SQL y los ataques XSS. La inyección de SQL se evita mediante su propio ORM el cual no permite procesar consultas de SQL extrañas. Y la capacidad de escapar de las etiquetas protege contra los ataques XSS [17].
- El middleware proporciona un mecanismo para filtrar las solicitudes HTTP enviadas a la aplicación. Laravel proporciona un middleware para verificar si el usuario de la aplicación está autenticado. Si el usuario no está autenticado, este middleware redirigirá al usuario a la página de inicio de sesión. Por el contrario, si el usuario está autenticado, permita que la solicitud continúe más allá de la aplicación [17].

1.4.3 API-REST CON LARAVEL

1.4.3.1 Definición de api-rest

Una API-REST, se basa en la transferencia de estado representacional (REST), que es un estilo arquitectónico y un enfoque para las comunicaciones que se usa a menudo en el desarrollo de servicios web, permitiendo recibir y modificar los datos y estados de aplicaciones remotas mediante la transmisión de llamadas HTTP a través de Internet o cualquier otra red [18].

En pocas palabras, la API REST es una aplicación de servidor que da acceso a sus datos a una aplicación cliente en una URL específica. La API REST permite utilizar el protocolo HTTP para la comunicación entre programas, con este se recibe y envía la mayor parte de la información en Internet [18].

1.4.3.2 Métodos HTTP

Para que el recurso que está solicitando realice las acciones deseadas, utiliza diferentes métodos para acceder a este. En un sistema API existen cuatro métodos clásicos de HTTP, los cuales se presentan a continuación [19]:

- GET: es un método para leer información. Las solicitudes GET únicamente devuelven datos del servidor y nunca los cambian ni los eliminan.
- POST: este método tiene la capacidad de crea nuevos registros.
- PUT: permite editar registros.
- DELETE: eliminación de registros.

Estas son los cuatro métodos que un programa puede usar al acceder a los datos de una API-REST.

1.4.3.3 Rutas de API en Laravel

En Laravel, todas las rutas API deben agregarse a un archivo en particular en el *routes/api.php* aunque se puede crear un archivo de ruta personalizado y vincularlo a la aplicación [18]. En el código 1.6, se presenta un ejemplo de rutas de una API, del archivo de ruta *routes/api.php*

```
Route::get('/ejemplo',[EjemploController::class, 'index']);
Route::get('/ejemplo/{id}',[ EjemploController::class, 'show']);
Route::post('/ejemplo',[ EjemploController::class, 'store']);
Route::put('/ejemplo/{id}',[ EjemploController::class, 'update']);
Route::delete('/ejemplo/{id}',[ EjemploController::class, 'destroy']);
```

Código 1.6. Rutas de una API-REST

Las rutas presentadas en el Código 1.6 están vinculadas a funciones específicas de un controlador las cuales son las encargadas de procesar la solicitud, estas se enumeran a continuación:

- GET /ejemplo, asignado al método index() del controlador EjemploController
- GET /ejemplo/{id}, asignado al método show() del controlador EjemploController
- POST /ejemplo, asignado al método store() del controlador EjemploController
- PUT /ejemplo/{id}, asignado al método update() del controlador EjemploController
- DELETE /ejemplo/{id}, asignado al método delete() del controlador EjemploController

1.4.4 SERVIDOR LARAGON

Laragon es un WAMP (Windows, Apache, MySQL y PHP) simple y compacto. Es un stack o conjunto de soluciones de software que, al instalar, se están instalando Apache, MySQL y PHP en un sistema operativo Windows; en muchos aspectos Laragon es similar a XAMPP, OpenServer, Denwer, etc. Laragon es una plataforma de desarrollo local robusta, este entorno de desarrollo tiene un conjunto de herramientas portátil, potente, aislado y rápido que presenta nuevas estrategias para los desarrolladores [20]. Laragon se utiliza como un espacio seguro para trabajar en un sitio web, sin necesidad de alojarlo de manera online. Las características más destacadas de Laragon con respecto a otros entornos de desarrollo son [20]:

- Crea Virtualhost de forma automática.
- Permite modificar la versión de PHP, Apache o MySQL/MariaDB.
- Permite compartir un proyecto local a través de Internet, para realizar pruebas antes de poner el proyecto en producción.
- Permite trabajar con Ngnix, además de Apache.
- Permite usar otros motores de base de datos como: MongoDB o PostgreSQL.

2 METODOLOGÍA

En este capítulo se describen las actividades realizadas en el desarrollo de este proyecto, divido en tres secciones. En la sección 2.1 se realiza un análisis del proceso de convalidación de prácticas preprofesionales y del formulario FCP_001A, posterior a este análisis se establecen los requerimientos para implementar la aplicación web. En la sección 2.2 se presenta el diseño de la aplicación web, el entorno de simulación, esquema de datos y boceto de la interfaz de usuario para cada rol. En la última sección 2.3 se explica cómo se desarrolló el prototipo de aplicación web “ConvalidaciónCPPP” y la creación de los módulos necesarios para cada rol de usuario.

2.1 ANÁLISIS DEL PROCESO DE CONVALIDACIÓN Y FORMULARIO FCP_001A

En esta sección se analiza el proceso de convalidación de prácticas preprofesionales y del formulario FCP_001A. La finalidad de este análisis es determinar los requerimientos para el diseño y desarrollo de la aplicación. Estos requerimientos involucran los roles de usuario que tendrá la aplicación, las actividades que desarrollarán cada uno de ellos y la forma en la que se llenarán los diferentes campos que contiene el formulario de convalidación. A continuación, se presentan estos análisis y posterior a ello los requerimientos para la aplicación web.

2.1.1 ANÁLISIS DEL PROCESO DE CONVALIDACIÓN

El primer punto de análisis es el proceso de convalidación de prácticas preprofesionales que se describe en el documento “Procedimiento Convalidación Actividades Extracurriculares” [2]. Aquí se da a conocer los pasos que se deben realizar para la convalidación y quienes intervienen en el proceso, siendo estos: el estudiante, el subdecano, el tutor, la Comisión de prácticas preprofesionales y el decano.

Adicionalmente a estos 5 actores, existe la intervención de un miembro de la CPP, el cual no está descrito en el documento antes mencionado, pero también desarrolla actividades dentro de este proceso. Por lo tanto, para la implementación de la aplicación web se considera la intervención de 6 actores en el proceso de convalidación.

El proceso de convalidación de prácticas preprofesionales se muestra en la Figura 2.1, donde se puede observar la intervención de los 6 actores antes mencionados y las actividades que cada uno desarrolla dentro de este proceso.

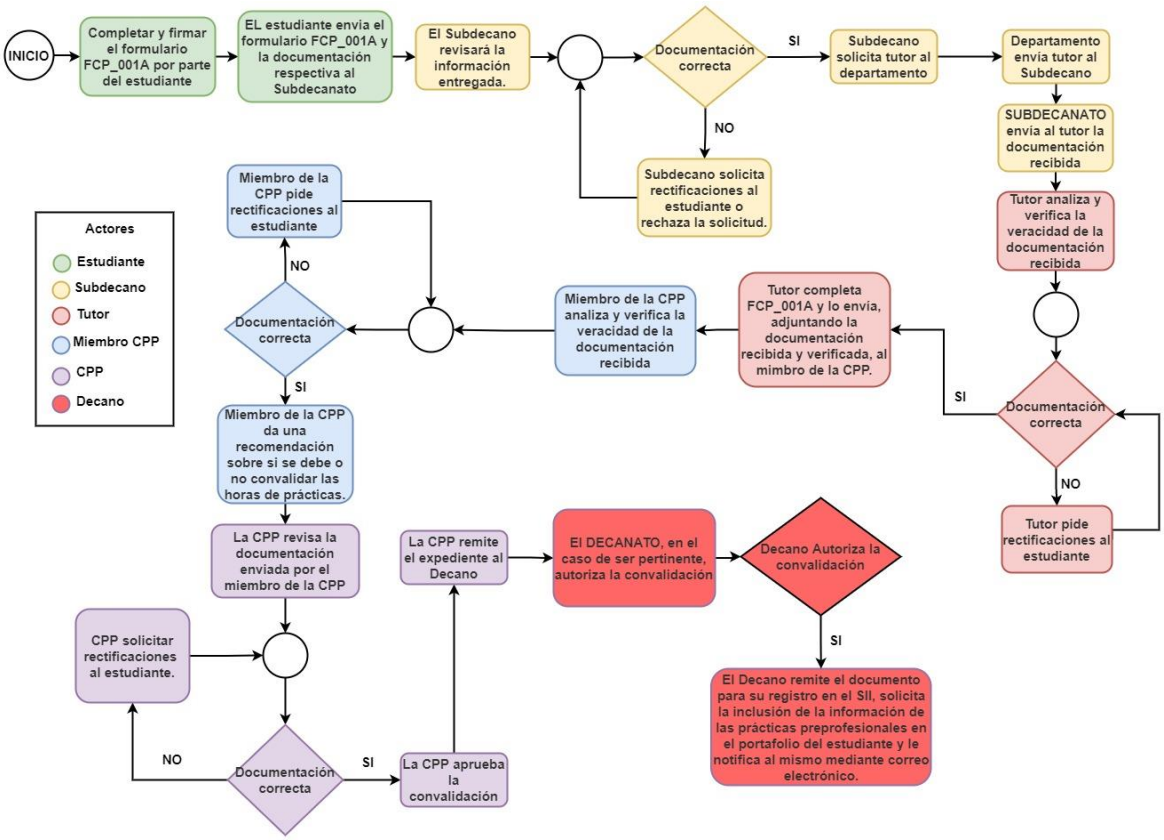

Figura 2.1. Proceso de convalidación de prácticas preprofesionales

En el documento “Procedimiento Convalidación Actividades Extracurriculares” [2]que estuvo vigente en el periodo académico 2021-B se establecía el siguiente proceso:

1. Cuando un estudiante requiere realizar un proceso de convalidación debe completar el formulario FCP_001A y firmarlo.
2. Los estudiantes deben adjuntar al formulario la documentación que respalde su petición y enviarla Subdecano o Subdirector de la ESFOT, según corresponda.
    a) El Subdecano o Subdirector de la ESFOT cuando recibe un formulario revisa la información, y si considera que la documentación está incompleta podrá negar el pedido.
3. En caso de que el Subdecano o Subdirector de la ESFOT considere que la información es correcta, podrá solicitar a los Departamentos afines a las carreras la asignación del tutor de prácticas preprofesionales.

4. El profesor que ha sido asignado como tutor de prácticas preprofesionales analizará y verificará la documentación y enviará el formulario FCP_001A completando las secciones que le corresponde y la documentación entregada por el estudiante a un miembro de la CPP.
5. El miembro de la CPP de igual manera analizará y verificará la documentación recibida y enviará el formulario adjuntando una recomendación sobre la convalidación a la CPP.
6. La Comisión de Prácticas Preprofesionales (CPP) analiza las recomendaciones emitidas por el tutor y el miembro de la CPP, y en caso de requerirse podrá pedir documentación adicional al estudiante; una vez que se haya revisado la documentación aprueba el formulario y remite el expediente al Decano o Director de la ESFOT para su autorización.
7. El Decano o Director de la ESFOT es el encargado de autorizar la convalidación.
8. El Decano o Director de la ESFOT, cuando se haya autorizado la convalidación, envía los documentos para el registro en el SII.

2.1.2 REQUERIMIENTOS BASADOS EN EL PROCESO DE CONVALIDACIÓN

Luego de conocer cuál es el proceso que se debe seguir para realizar la convalidación de prácticas preprofesionales, se pueden establecer los requerimientos para la aplicación web, para esto se ha elaborado un diagrama de contexto, el cual se presenta en la Figura 2.2. Este diagrama de contexto ayuda a representar el sistema como un solo proceso y en ella se describen las interacciones que existen entre el sistema y el resto de las entidades o actores [21].

Para la aplicación web ConvalidaciónCPPP como primer requisito se consideran siete roles de usuario como se muestra en la Figura 2.2, en este diagrama de contexto como ya fue mencionado se pueden observar las interacciones que existe entre los usuarios dependiendo su rol y la aplicación web, estas interacciones describen lo que cada usuario entrega a la aplicación, lo que recibe de ella y las funcionalidades que desarrolla cada rol. Para entender de mejor manera el diagrama de contexto y sus funcionalidades, a continuación, se realiza una explicación de algunos términos usados en este diagrama.

- Formularios enviados: corresponde a los formularios que un estudiante a enviado para realizar la convalidación y aún se encuentran en proceso, es decir, que no han sido rechazados, no se solicitan correcciones y tampoco han terminado con éxito el proceso.

- Formularios revisados: Corresponden a los formularios que ya han sido revisados por el subdecano, tutor, miembro de la cpp o coordinador de la CPP.
- Formularios por corregir: Corresponde a los formularios que han sido devueltos por alguna falla y que requieren ser corregidos por el estudiante.
- Formularios corregidos: Son aquellos formularios que ya han sido corregidos por el estudiante y enviados nuevamente a la persona que solicito la corrección.
- Formularios aceptados: Son aquellos formularios que ya han terminado su proceso con éxito.
- Formularios rechazados: Son aquellos formularios que por alguna razón fueron rechazados por alguno de los actores que presentan esta funcionalidad.
- CRUD: es un acrónimo que hace referencia a las cuatro funciones que se consideran necesarias para implementar una aplicación de almacenamiento las cuales son: crear, leer, actualizar y borrar un registro [22]. En este caso estos registros corresponden a estudiantes y profesores.

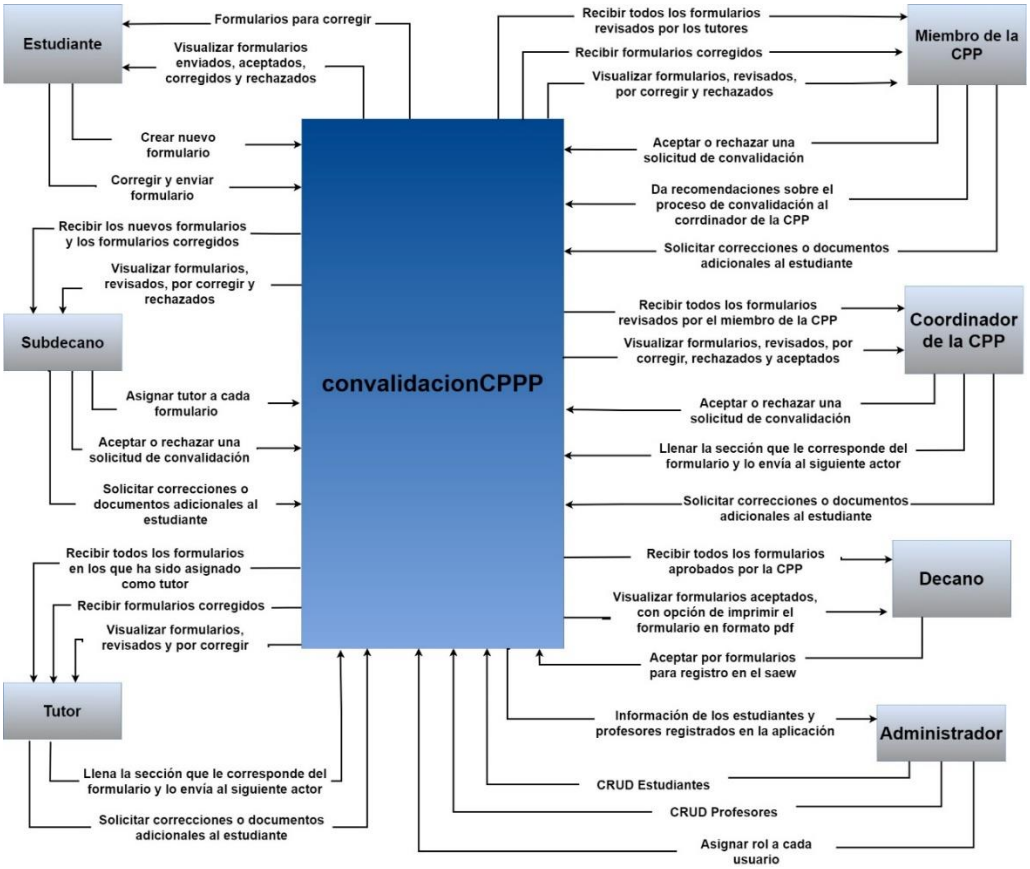

Figura 2.2. Diagrama de contexto del prototipo ConvalidaciónCPPP

En la Tabla 2.1 se muestra un resumen de las funcionalidades para cada rol de usuario, los mismos forman parte de los requerimientos para diseñar e implementar la aplicación web.

Tabla 2.1. Funcionalidades para cada rol de usuario

Rol de usuario	Funcionalidad
Estudiante	• Crear un nuevo formulario • Corregir los formularios que han sido devueltos por alguno de los revisores solicitando correcciones. • Visualizar los formularios enviados, rechazados, aceptados y corregidos.
Subdecano	• Revisar los formularios y la documentación adjunta en cada proceso • Asignar tutor a cada proceso de convalidación si la documentación es correcta • Solicitar correcciones al estudiante • Rechazar peticiones de convalidación • Visualizar los formularios revisados, rechazados, y por corregir.
Tutor	• Revisar los formularios y la documentación adjunta en cada proceso • Llenar la sección del formulario correspondiente a su rol • Solicitar correcciones al estudiante • Visualizar los formularios revisados, y por corregir.
Miembro de la comisión de prácticas preprofesionales	• Revisar los formularios y la documentación adjunta en cada proceso • Dar una recomendación al coordinador sobre la convalidación • Solicitar correcciones al estudiante • Rechazar peticiones de convalidación • Puede visualizar los formularios revisados, rechazados, y por corregir.
Comisión de prácticas preprofesionales	• Revisar los formularios y la documentación adjunta en cada proceso • Aprobar o rechazar el proceso de convalidación • Llenar la sección correspondiente a su rol • Solicitar correcciones al estudiante • Visualizar los formularios revisados, rechazados, por corregir y aceptados.
Decano	• Firmar el documento aceptando su registro en el saew • Visualizar los formularios que s**e** han autorizado para el registro • Convierte el formulario en pdf con el formato original del formulario FCP_001A

Administrador	• CRUD de usuarios • Puede visualizar todos los usuarios registrados en la aplicación web • Puede editar el rol de los usuarios

El sistema solo permitirá que el Administrador de la aplicación web cree nuevos usuarios.

2.1.3 ANÁLISIS DEL FORMULARIO FCP_001A

El formulario FCP_001A debe ser llenado cuando un estudiante requiere realizar la convalidación de prácticas preprofesionales. El formulario que estuvo vigente al momento de realizar el plan de este trabajo corresponde a la versión 2 del mismo, este se muestra en Figura 2.3. El formulario está compuesto por 10 secciones, los mismos que son llenados por diferentes actores dentro de este proceso. A continuación, se presenta una descripción de estas secciones, lo cual más adelante servirá para establecer requerimientos adicionales para el diseño y desarrollo de la aplicación web ConvalidaciónCPPP.

1. **Actividades para las que solicita la convalidación:** en esta sección se debe especificar el tipo de actividad para el que se solicita la convalidación, existen 11 opciones de las cuales solo se puede seleccionar una de ellas.
2. **Datos del estudiante:** contiene información personal del estudiante que solicita la convalidación, nombres y apellidos, cédula de identidad, correo electrónico, celular y teléfono.
3. **Documentación de soporte adjunta:** el estudiante tiene la obligación de presentar documentos que soporten la solicitud, como por ejemplo certificados de cursos realizados que contribuyen al perfil profesional del estudiante.
4. **Información de la institución en la que realizó las actividades:** en esta sección se debe presentar información de la institución en la que el estudiante realizó las actividades, esta información incluye, razón social, ruc, dirección, ciudad, país, teléfono, celular, correo y tipo de institución. Además de existir un convenio entre la EPN y la institución se debe especificar el código y nombre del convenio.
5. **Información de las actividades realizadas:** el estudiante debe responder las preguntas que se le plantean en esta sección describiendo las actividades realizadas.
6. **Información adicional:** en esta sección se especifican las fechas de inicio y fin de prácticas, así como el total de horas solicitadas para convalidación. Y de requerirse se puede incluir archivos adicionales como registros de asistencias.

7. **Declaración:** incluye el nombre y firma del estudiante, aceptando la veracidad de los documentos entregados para la convalidación.
8. **Informe del tutor:** aquí el tutor debe responder las preguntas que se le platean en esta sección, y da su recomendación sobre si se debe o no convalidar las horas solicitadas por el estudiante.
9. **Comisión de prácticas preprofesionales:** de haber comprobado que la documentación es correcta, el coordinador de la comisión determinar el total de horas convalidadas, esto se describe en esta sección.
10. **Certificaciones:** incluye fechas de revisión, fechas de recepción, nombres y firmas del tutor, coordinadora de prácticas preprofesionales y decano.

FORMULARIO FCP_001A
Versión 2

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

CARRERA: (RRA20) Telecomunicaciones

CONVALIDACIÓN DE PRÁCTICAS PREPROFESIONALES

SECCIONES A SER LLENADAS POR CADA RESPONSABLE:
Estudiante (resaltadas en verde)
Tutor (resaltada en amarillo)
CPP (resaltada en anaranjado)

1. ACTIVIDADES PARA LAS QUE SOLICITA LA CONVALIDACIÓN

	Cursos y Seminarios Profesionales		Idiomas diferentes al Inglés y Lengua Materna
	Participación Estudiantil en Actividades Académicas, de Gestión, de Investigación y de Colaboración en Eventos Académicos **		Dirección de ramas de organizaciones estudiantiles académicas
	Representación Estudiantil		Representación de la Institución en competencias académicas
	Estudiantes mentores		Coro y Grupo de Cámara
	Representación de la Institución en competencias deportivas		Participación en la dirección de asociaciones de estudiantes
	Actividades solidarias y de cooperación		Participación en juntas receptoras del voto
	Experiencia Laboral		

2. DATOS DEL ESTUDIANTE

Nombres y Apellidos:					
Cédula de Identidad:					
Correo electrónico:		Teléfono:		Celular:	

3. DOCUMENTACIÓN DE SOPORTE ADJUNTA

Listar los documentos que se adjuntan

4. INFORMACIÓN DE LA INSTITUCIÓN EN LA QUE REALIZÓ LAS ACTIVIDADES

Razón Social *:					
RUC *:					
Dirección *:					
Ciudad/País:		Teléfono *:		Celular* :	
Correo electrónico:					
Tipo de Institución:					
Campo Amplio:	CIENCIAS NATURALES, MATEMÁTICAS Y ESTADÍSTICA				
Campo Específico:	Matemáticas y estadística				
Código de Proyecto/Convenio **:		Nombre del Proyecto/Convenio:			

** En el caso de que la Razón Social corresponda a un organismo internacional (Coursera, Edx u otras plataformas) colocar N/A (No Aplica).*

*** En caso de que las actividades sean bajo Convenios o Proyectos, indicar el código y nombre del convenio o proyecto.*

5. INFORMACIÓN DE LAS ACTIVIDADES REALIZADAS

Breve resumen de las actividades realizadas:
¿De qué manera las actividades realizadas contribuyeron al perfil de egreso de su carrera?
¿A qué resultados de aprendizaje del perfil de egreso considera que aportaron las actividades realizadas?
¿Cuáles son las asignaturas de la malla curricular y las temáticas de mayor utilidad para el desarrollo de las actividades?

6. INFORMACIÓN ADICIONAL

Información de las fechas en las que realizó las actividades			
Fecha inicio:		**Fecha fin:**	
Horas solicitadas *:**			

**** En el caso de actividades con horarios flexibles, detallar los horarios de trabajo y adjuntar el registro de asistencia y actividades.*

7. DECLARACIÓN

Yo, NOMBRE_ESTUDIANTE, declaro que la información presentada para la convalidación de prácticas preprofesionales es verídica.
Fecha: dd/mm/aaaa **Firma:**

8. INFORME DEL TUTOR EPN

Nombre:		**Departamento:**			
¿Considera que las actividades reportadas contribuyeron a la aplicación de conocimientos o al desarrollo de competencias en la formación del estudiante?	**SI:**		**NO:**		
¿Considera que las actividades reportadas contribuyeron a la consecución de los resultados del aprendizaje del perfil de egreso?	**SI:**		**NO:**		
¿Validó las actividades reportadas por el estudiante?	**SI:**		**NO:**		
Análisis y Recomendaciones respecto de la información presentada:					
Horas validadas y sugeridas de convalidación:					

9. COMISIÓN DE PRÁCTICAS PREPROFESIONALES

Horas convalidadas:			
Prácticas Laborales:		**Servicio Comunitario:**	
Observaciones de la CPP:			

10. CERTIFICACIONES

TUTOR DE PRÁCTICAS PREPROFESIONALES	**COMISIÓN DE PRÁCTICAS PREPROFESIONALES**
Fecha de Recepción: DD/MM/AAAA	Fecha de Recepción: DD/MM/AAAA
Fecha de Revisión: DD/MM/AAAA	Fecha de Aprobación: DD/MM/AAAA
f. _____ Tutor	f. _____ Presidente
Nombre:	Nombre:

DECANO(A) DE LA FACULTAD / DIRECTOR(A) DE LA ESFOT

Fecha de Recepción: DD/MM/AAAA
Fecha de Autorización: DD/MM/AAAA

f. _____
Decano (a) / Director (a)

Nombre:

Fecha de Registro en SAEw: DD/MM/AAAA Responsable Registro SAEw: Nombre

Figura 2.3. Formulario FCP_001A versión 2

2.1.4 REQUERIMIENTOS BASADOS EN EL ANÁLISIS DEL FORMULARIO FCP_001A

Luego de haber realizado el análisis del formulario FCP_001A se pueden establecer requerimientos adicionales para la aplicación web, estos determinarán la forma en la que se llenarán los diferentes campos que contiene el formulario.

Antes de comenzar a detallar estos requerimientos, es importante mencionar como se logrará que algunos de los campos sean llenados de forma automática. Para esto, como ya fue mencionado en el alcance de este proyecto, se deberán crear dos API-REST, esto con la finalidad de simular el cruce de información con entidades internas o externas a la EPN, las cuales contienen información requerida por el formulario.

- La API *university-rest* contendrá información ficticia de estudiantes y profesores, y de ella se importarán datos a la aplicación web ConvalidaciónCPPP mediante consultas HTTP. Los datos obtenidos a través de estas consultas para el caso de estudiantes son: carrera, nombres, apellidos, número único, cédula, teléfono, celular y correo. Para el caso de profesores los datos que se obtienen al realizar la consulta son: departamento, nombres, apellidos, número único, cédula, teléfono, celular y correo.
- La API *sri* tiene la misma funcionalidad que la api *university-rest,* con la diferencia que esta contiene información de empresas registradas en el SRI. Al tratarse de una simulación la base de datos en este caso está limitada a 7 empresas. Las consultas realizadas a esta api retornan la siguiente información: razón social, ruc, dirección, ciudad, país, teléfono, celular, correo y tipo de institución.

Una vez aclarada esta funcionalidad, se presentan los requerimientos para la implementación de la aplicación web basados en el análisis del formulario:

a) Carrera

Este campo debe ser llenado de forma automática y no debe permitir su edición, la carrera a la que pertenece un estudiante se la obtiene a través de la consulta realizada a la API university-rest.

b) Actividades para las que solicita la convalidación

Se debe permitir seleccionar únicamente una de las once opciones que se presentan en el formulario. Para esto se usará un radio button que presente las once opciones.

c) Datos del estudiante:

Todos los campos de esta sección deben ser llenado de forma automática. Como ya se explicó anteriormente esta información se obtiene mediante la consulta a la API *university-rest*, los campos nombres y cédula no serán editables, mientras que los campos de correo, celular y teléfono a pesar de que ya son llenados deben permitir ser editados, esto considerando que estos datos pueden ser actualizados.

d) Documentación de soporte adjunta

En esta sección se deben cargar los documentos que sustentan la solicitud, con el objetivo de optimizar el almacenamiento se permitirá cargar únicamente archivos en formato pdf.

e) Información de la institución en la que realizó las actividades

En esta sección, en primer lugar, se deberá indicar si la institución en la que realizó las actividades es Nacional o Internacional, esto debido a que los campos correspondientes a información de la institución se llenarán de diferente manera en cada uno de los casos. A continuación, se presenta como se llenarán los campos correspondientes a esta sección para cada caso.

- **Institución internacional**

  De tratarse de una institución internacional como Coursera Edx u otras plataformas, el llenado es el siguiente:

  - Razón social: N/A
  - RUC: N/A
  - Dirección: N/A
  - Teléfono: N/A
  - Celular: N/A
  - Ciudad/País: N/A
  - Correo electrónico: N/A
  - Tipo de institución: Se podrá seleccionar entre las opciones que muestra el formulario, pública, privada, organismos internacionales, tercer sector u otras.
  - Campo amplio y campo específico: Deberán ser llenado de forma automática sin permitir su edición, para esto se deberá validar la carrera de cada estudiante y de acuerdo con la misma se establecerá el campo amplio y especifico.

- Código de proyecto/convenio y Nombre de proyecto/convenio: Los campos correspondientes a convenio serán llenados de forma manual debido a que no existe una base de datos, con la cual validar esta información.

➢ **Institución Nacional**

Si se trata de una institución nacional los campos correspondientes a esta sección serán llenados de la siguiente manera:

- Se debe permitir únicamente ingresar el ruc de la empresa, con este dato, la aplicación realizará una consulta en los datos entregados por la api-rest sri obteniendo así la información de la empresa correspondiente. De esta manera el ruc, razón social, dirección, país/ciudad, teléfono, celular, correo y tipo de institución serán completados de forma automática.
- Campo amplio y campo específico: deberán ser llenado de forma automática sin permitir su edición, para esto se deberá validar la carrera de cada estudiante y de acuerdo con la misma se establecerá el campo amplio y específico.
- Código de proyecto/convenio y Nombre de proyecto/convenio: Los campos correspondientes a convenio serán llenados de forma manual debido a que no existe una base de datos, con la cual validar esta información.

f) Información de las actividades realizadas

Todos estos campos deben ser llenados de forma manual mediante un cuadro de texto, en este campo no se establece un ningún límite de caracteres.

g) Información Adicional

- Fecha de inicio y fecha de fin: Se debe permitir seleccionar las fechas mediante un calendario, además, es de suma importancia validar que la fecha de fin no sea antes que la fecha de inicio y que la fecha de fin no sea después de la fecha actual.
- Horas solicitadas: estas serán ingresadas de forma manual, validando que se ingrese un número entero positivo.
- Archivos de información adicional: en esta sección, se debe permitir cargar archivos tipo pdf que respalden las horas solicitadas, por ejemplo, un registro de asistencia de ser necesario.

h) Declaración

- El nombre del estudiante debe ser llenado se forma automática
- Fecha de declaración: debe ser llenada de forma automática y corresponderá a la fecha en la que el estudiante crear un nuevo formulario.
- Firma: permitirá cargar únicamente un archivo de imagen con la firma del estudiante.

i) Informe del tutor:

- Nombre del tutor: El subdecano en el caso de aceptar el formulario, deberá hacer la asignación de tutor al estudiante, de esta forma el campo nombre y departamento de esta sección serán llenados de forma automática.
- Preguntas al tutor: las tres preguntas de selección, sí o no, solo permitirá seleccionar una de ellas, mientras que las dos preguntas restantes sobre recomendaciones que da el tutor serán llenadas mediante un cuadro de texto.

j) Comisión de prácticas preprofesionales:

- El campo horas convalidadas: debe permitir el ingreso manual de un número, validando que este sea entero positivo.
- De igual forma los campos, Prácticas Laborales y Servicio Comunitario: deben permitir el ingreso manual de un número y validar que este sea entero positivo.
- El campo Observaciones de la CPP: debe permitir redactar las observaciones mediante un cuadro de texto.

k) Certificaciones

- **Firmas y fechas de la sección "Tutor de prácticas preprofesionales"**

    - La fecha de recepción debe ser establecida de forma automática y ésta corresponde a la fecha en la se hace la asignación de tutor.
    - La fecha de revisión de igual manera debe ser establecida de forma automática y corresponde a la fecha en la que el tutor revisa el formulario y lo envía al siguiente actor.
    - La firma debe ser únicamente cargada como un archivo de imagen.
    - El nombre debe ser llenado automáticamente.

- **Firmas y fechas de la sección "Comisión prácticas preprofesionales"**

  - La fecha de recepción debe ser establecida de forma automática y ésta corresponde a la fecha en la que el miembro de la CPP envía el formulario al coordinador de la CPP.
  - La fecha de revisión de igual manera debe ser establecida de forma automática y corresponde a la fecha en la que el coordinador de la CPP revisa el formulario y lo envía al siguiente actor.
  - La firma debe ser únicamente cargada como un archivo de imagen.
  - El nombre debe ser llenado automáticamente.

- **Firmas y fechas de la sección "Decano(a) de la facultad"**

  - La fecha de recepción debe ser establecida de forma automática y ésta corresponde a la fecha en la que el coordinador de la CPP envía el formulario al decano.
  - La fecha de revisión, de igual manera, debe ser establecida de forma automática y corresponde a la fecha en la que el decano(a) revisa el formulario y lo aprueba.
  - La firma debe ser únicamente cargada como un archivo de imagen.
  - El nombre debe ser llenado automáticamente.

l) Descargar formularios en el formato original

El decano cuando finaliza el proceso de convalidación puede convertir el formulario, en un archivo PDF con su formato original y descargarlo. Este documento contendrá toda la información que ha sido ingresada durante el proceso. Adicional a ello, este formulario será almacenado en un espacio de memoria.

m) Manejo de firmas en el formulario

Un punto importante que mencionar, como parte de los requerimientos para la implementación de la aplicación web, es el manejo de las firmas dentro del formulario. El presente proyecto se limita únicamente al uso de firmas mediante un archivo de imagen, la cual debe contener la firma de cada uno de los actores de forma clara.

n) Manejo de archivos

Los documentos de respaldo que son cargados a la aplicación web, así como el formulario correspondiente a cada proceso de convalidación, deben ser almacenados en un espacio de memoria, manteniendo una estructura ordenada de los archivos. Para esto, cada estudiante contará con un directorio principal el cual tendrá como nombre su número único. Este directorio almacenará subdirectorios relacionados a cada proceso y en estos se tendrán dos últimos subdirectorios donde se almacenan los documentos de soporte e información adicional. En la Figura 2.4 se muestra la estructura de almacenamiento de los documentos en la aplicación web.

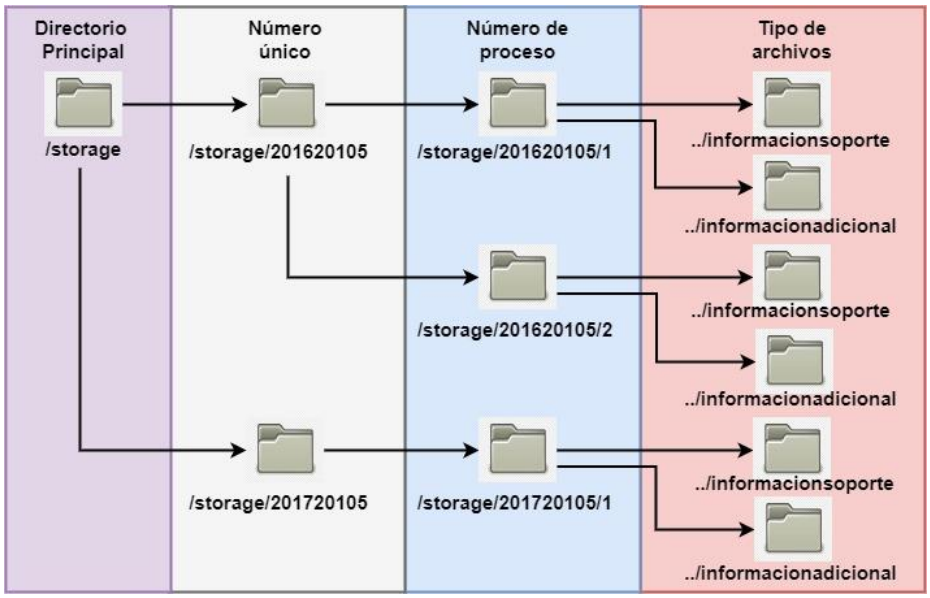

Figura 2.4. Esquema de almacenamiento de archivos

o) Validaciones del formulario

Además de presentar la forma en la que se deben completar cada uno de los campos del formulario, es importante determinar, que campos son obligatorios de llenar y sus respectivas validaciones como, por ejemplo, longitud máxima de caracteres, tipo de datos (texto o números), tipos de archivos aceptados, etc. Esto se presenta en la siguiente tabla “Validaciones en el formulario”.

Tabla 2.2. Validaciones del Formulario mediante la aplicación web

Sección del formulario	Campos del formulario	Validaciones
Sección 1	Actividades para las que solicita la convalidación	• Obligatorio
Sección 2	Nombres y Apellidos	• Obligatorio
	Cédula de Identidad	• Obligatorio
	Correo electrónico	• Obligatorio
	Teléfono	• Obligatorio • Máximo 10 dígitos
	Celular	• Obligatorio
Sección 3	Documentación de soporte adjunta	• Obligatorio • Solo archivos pdf • Máximo 2MB
Sección 4	Razón social	• Obligatorio
	RUC	• Obligatorio
	Dirección	• Obligatorio
	Ciudad/País	• Obligatorio
	Correo electrónico	• Obligatorio
	Teléfono	• Obligatorio
	Celular	• Obligatorio
	Tipo de Institución	• Obligatorio
	Campo Amplio	• Obligatorio
	Campo Especifico	• Obligatorio
	Código de Proyecto/Convenio	• No obligatorio
	Nombre de Proyecto/Convenio	• No obligatorio
Sección 5	Pregunta 1	• Obligatorio
	Pregunta 2	• Obligatorio
	Pregunta 3	• Obligatorio
	Pregunta 4	• Obligatorio
Sección 6	Fecha de inicio	• Obligatorio • No puede después de la fecha de fin
	Fecha de fin	• Obligatorio • No puede ser posterior a la fecha actual
	Horas solicitadas	• Obligatorio • Número entero positivo
Sección 7	Fecha	• Obligatorio

	Firma	• Obligatorio • Solo archivo de imagen
Sección 8	Nombre	• Obligatorio
	Departamento	• Obligatorio
	Pregunta 1	• Obligatorio
	Pregunta 2	• Obligatorio
	Pregunta 3	• Obligatorio
	Análisis y recomendación	• Obligatorio
	Horas validadas y sugeridas de convalidación	• Obligatorio
Sección 9	Horas convalidas	• Obligatorio Numero entero positivo
	Prácticas Laborales	• Obligatorio • Numero entero positivo • La suma de las horas de servicio comunitario y prácticas laborales no puede ser mayor a las horas convalidadas
	Servicio comunitario	• Obligatorio • Numero entero positivo • La suma de las horas de servicio comunitario y prácticas laborales no puede ser mayor a las horas convalidadas
	Observaciones de la CPP	• Obligatorio
Sección 10	Nombres	• Obligatorio
	Fechas de recepción	• Obligatorio
	Fechas de revisión	• Obligatorio
	Firmas de tutor, decano y cpp	• Obligatorio • Solo archivo de imagen

2.2 DISEÑO DE LA APLICACIÓN WEB

Una vez que se han establecido los roles de usuario, las actividades que estos desarrollan y también los requerimientos que debe tener la aplicación web, se procede con el diseño de esta. El diseño involucra, establecer el entorno de simulación donde interactúan: la aplicación web ConvalidaciónCPPP y las api-rest; además, se presenta el esquema de datos con el que trabajará la aplicación web y el diseño de la interfaz gráfica

para cada rol de usuario considerando las actividades que se desarrollan en cada uno respectivamente.

2.2.1 ENTORNO DE SIMULACIÓN

El prototipo de aplicación web propuesto en este trabajo tiene el objetivo validar y automatizar el llenado del formulario FCP_001A. En la sección 2.1 se presentó el análisis del proceso de convalidación y del formulario, donde se determinaron los requerimientos para implementar la aplicación web. En este análisis se estableció que se requieren desarrollar dos api-rest con el fin de simular el cruce de información con entidades externas o internas a la EPN, y con esto, conseguir el llenado automático de algunos campos del formulario.

La necesidad de crear estas api-rest surge debido a complicaciones presentadas para realizar consultas a las bases de datos de la EPN y también la negación de permisos para el uso del servicio web disponible en la institución. Por esta razón, al no poder realizar consultas a bases de datos reales se crean estas dos api-rest que simulan a estas entidades (EPN y SRI).

La aplicación ConvalidaciónCPPP requiere usar datos externos a ella para completar de forma automática cierta información del formulario. Debido a que la aplicación no será puesta en producción ya que solo se trabaja en un ambiente local, tanto las api-rest como la aplicación web se ejecutarán en la misma máquina local, pero se lo hará en diferentes puertos. Esto es necesario para lograr la comunicación entre ellas sin generar conflictos entre los servicios. En la Figura 2.5. se presenta el esquema del entorno de simulación que se usará en el desarrollo del proyecto, aquí se puede observar la interacción que existe entre las api-rest y la aplicación web.

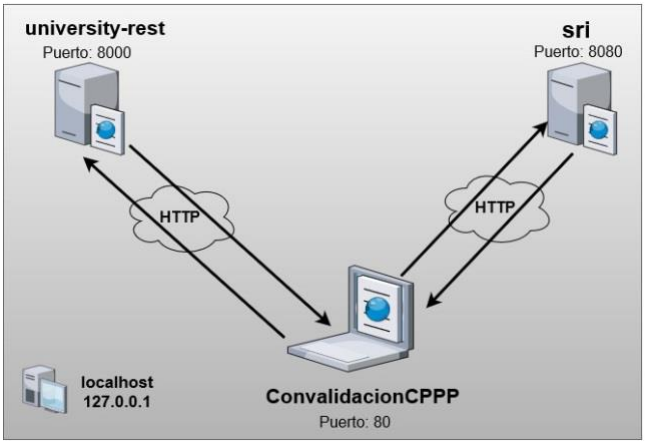

Figura 2.5. Esquema de simulación del prototipo de aplicación web

Laravel permite ejecutar los servicios web en diferentes puertos de una máquina local, esto se logra de la siguiente forma [23]:

- Desde una terminal ingresar a la carpeta del proyecto con el comando "*cd*"
- Una vez en la carpeta se debe ejecutar el comando: *php artisan serve –port:8000 (Donde la opción –port especifica el puerto en el cual se levanta el servicio)*

2.2.2 ESQUEMA DE BASE DE DATOS

Un esquema de base de datos es un diseño abstracto que representa el almacenamiento de sus datos en una base de datos. Describe tanto la organización de los datos como las relaciones entre tablas en una base de datos determinada. Los desarrolladores planifican el esquema de la base de datos con anticipación para saber qué componentes se necesitan y cómo se conectarán entre sí [24]. El esquema de la base de datos ayuda a garantizar que:

- Exista un formato consistente de registros de datos.
- Todos los registros tengan una clave primaria única.
- No falte ningún dato importante en las tablas o entidades.

El proyecto ConvalidaciónCPPP presenta el esquema de base de datos de la Figura 2.6.

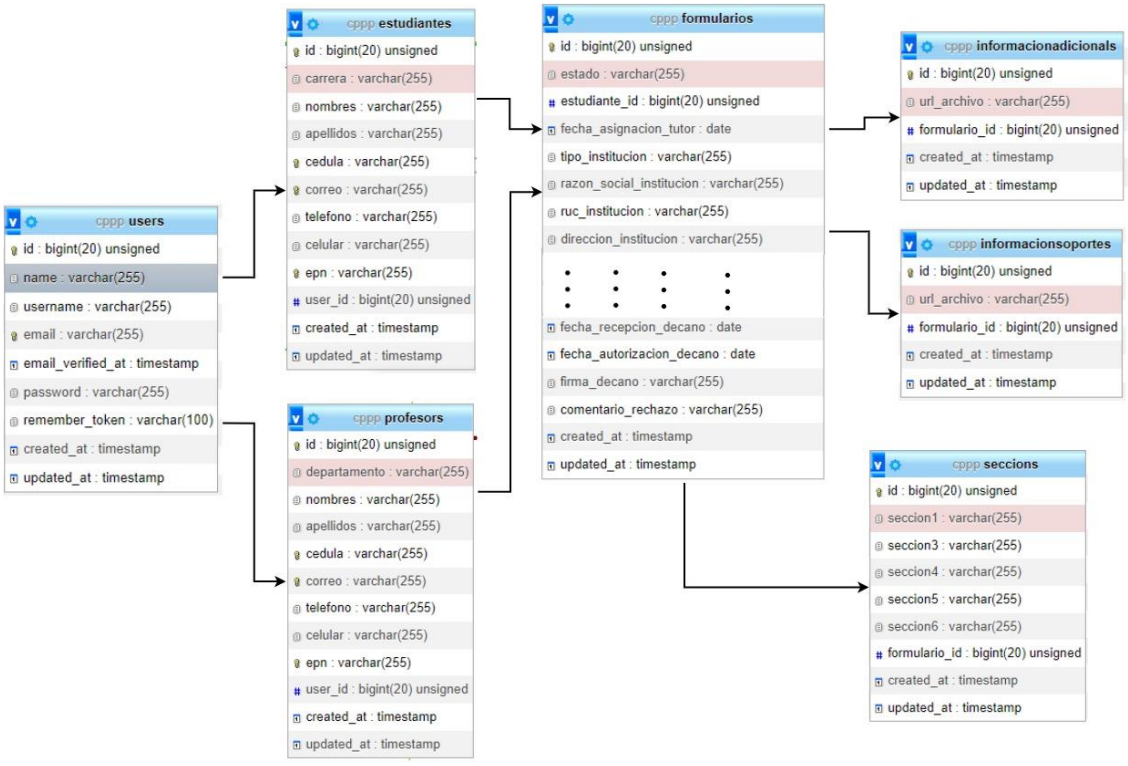

Figura 2.6. Esquema de base de datos de la aplicación ConvalidaciónCPPP

Como se puede observar en la Figura 2.6, la base de datos requiere implementar 7 tablas para el funcionamiento de esta aplicación, siendo estas las siguientes:

- Tabla users: contiene los registros de los usuarios de la aplicación web.
- Tabla estudiantes: contiene los registros de estudiantes que tiene acceso a la aplicación web.
- Tabla profesors: contiene los registros de profesores que tiene acceso a la aplicación web.
- Tabla formularios: contiene toda la información del formulario de convalidación.
- Tabla informacionsoportes: contiene los archivos que sustentan la solicitud.
- Tabla informacionadicionals: contiene los archivos adicionales para la solicitud.
- Tabla seccions: Lleva el registro de las secciones que se deben corregir en cada formulario cuando un actor lo solicita.

2.2.3 DISEÑO DE LA INTERFAZ GRÁFICA

Antes del desarrollo de la aplicación web, es importante diseñar también la interfaz gráfica de la aplicación web cumpliendo con los requerimientos para cada uno de los roles de usuarios. Para elaborar los bocetos de las pantallas para cada usuario se usó uizard, el cual es una aplicación web que permite crear de forma rápida prototipos y bocetos de aplicaciones y páginas web [25].

En la Figura 2.7, se presenta el diseño del boceto de la interfaz gráfica para el rol Administrador y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

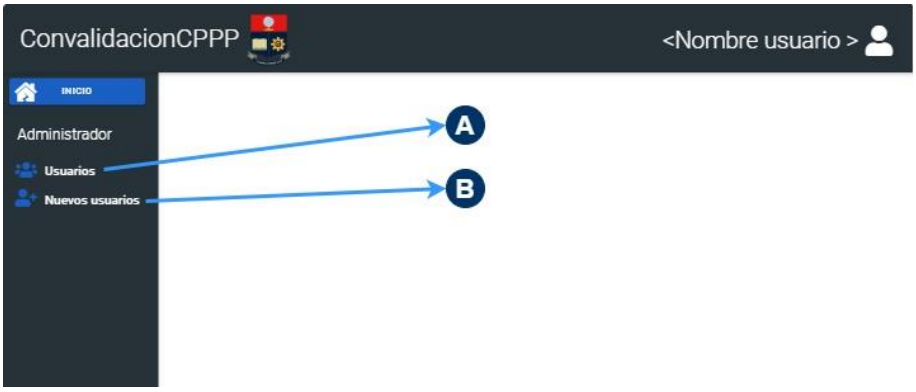

Figura 2.7. Diseño de la interfaz principal del rol Administrador

En la Tabla 2.3 se da un detalle de los rótulos de la Figura 2.7.

Tabla 2.3. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Administrador

Figura	Rótulo	Funcionalidad
2.7	A	Este icono listará todos los usuarios de la aplicación web, permitiendo eliminar o editar características de los usuarios
	B	Este icono mostrará las 3 opciones para hacer un nuevo registro de usuarios, ya sea mediante consultas al api-rest o un registro manual de los datos de los usuarios

En la Figura 2.8, se presenta el diseño del boceto de la interfaz gráfica para el rol Estudiante y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

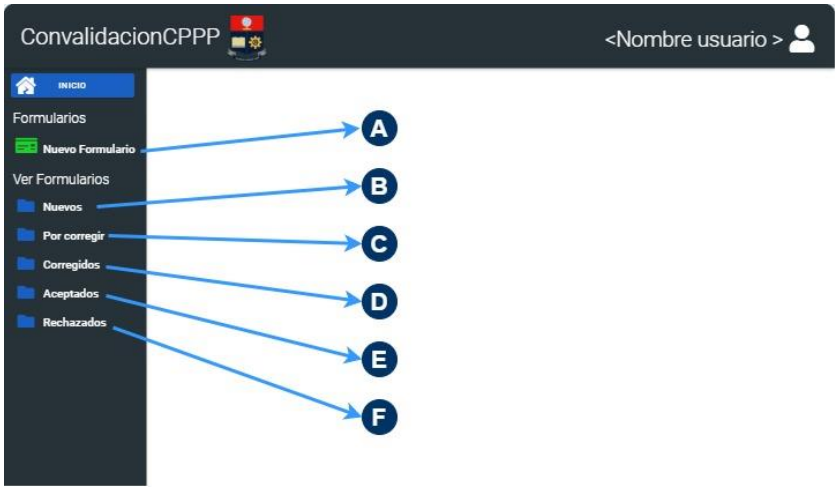

Figura 2.8. Diseño de la interfaz principal del rol Estudiante

En la Tabla 2.4 se da un detalle de los rótulos de la Figura 2.8.

Tabla 2.4. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Estudiante

Figura	Rótulo	Funcionalidad
2.8	A	Mediante icono el estudiante podrá crear un nuevo formulario
	B	Este icono muestra los formularios enviados para su revisión
	C	Este icono muestra los formularios que se deben corregir
	D	Este icono muestra los formularios que ya han sido corregidos
	E	Este icono muestra los formularios que han terminado con éxito el proceso de convalidación
	F	Este icono muestra los formularios que han sido rechazados

En la Figura 2.9, se presenta el diseño del boceto de la interfaz gráfica para el rol Subdecano y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

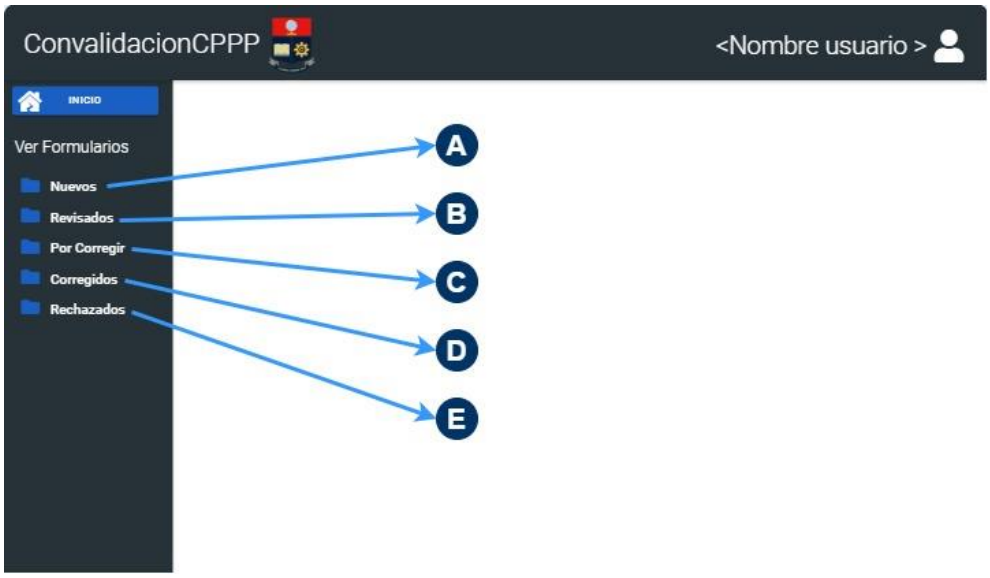

Figura 2.9. Diseño de la interfaz principal del rol Subdecano

En la Tabla 2.5 se da un detalle de los rótulos de la Figura 2.9.

Tabla 2.5. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Subdecano

Figura	Rótulo	Funcionalidad
2.9	A	Este icono muestra los formularios enviados por los estudiantes para su revisión y asignación de tutor
	B	Este icono muestra los formularios que ya han sido revisados por el subdecano y ya tienen un tutor asignado
	C	Este icono muestra los formularios en los que el subdecano solicita correcciones
	D	Este icono muestra los formularios que ya han sido corregidos
	E	Este icono muestra los formularios que han sido rechazados

En la Figura 2.10, se presenta el diseño del boceto de la interfaz gráfica para el rol Tutor y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

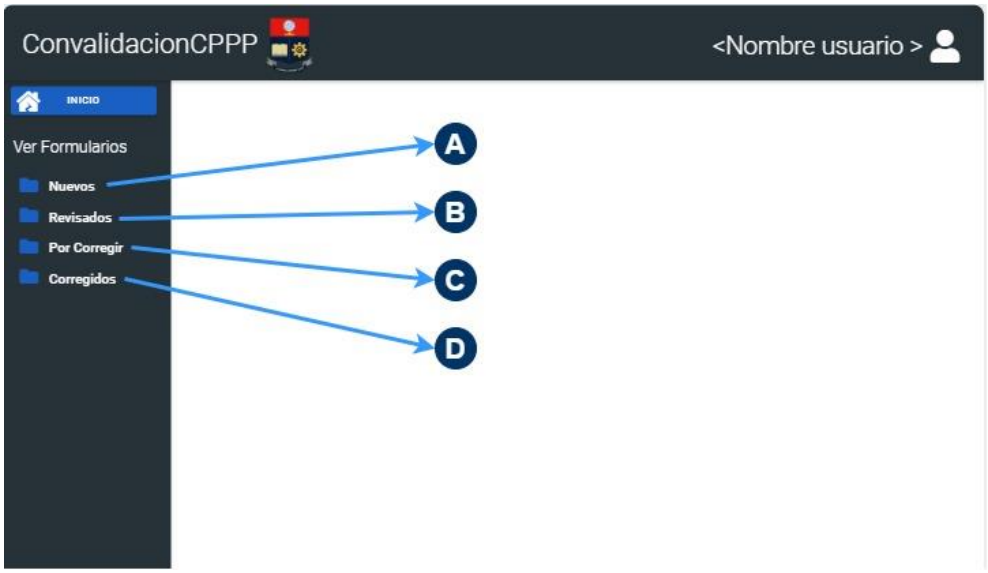

Figura 2.10. Diseño de la interfaz principal del rol Tutor

En la Tabla 2.6 se da un detalle de los rótulos de la Figura 2.10.

Tabla 2.6. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Tutor

Figura	Rótulo	Funcionalidad
2.10	A	Este icono muestra los nuevos formularios en los que se le ha asignado como tutor
	B	Este icono muestra los formularios que ya han sido revisados por el tutor
	C	Este icono muestra los formularios en los que el tutor solicita correcciones
	D	Este icono muestra los formularios que ya han sido corregidos

En la Figura 2.11, se presenta el diseño del boceto de la interfaz gráfica para el rol Miembro de la CPP y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

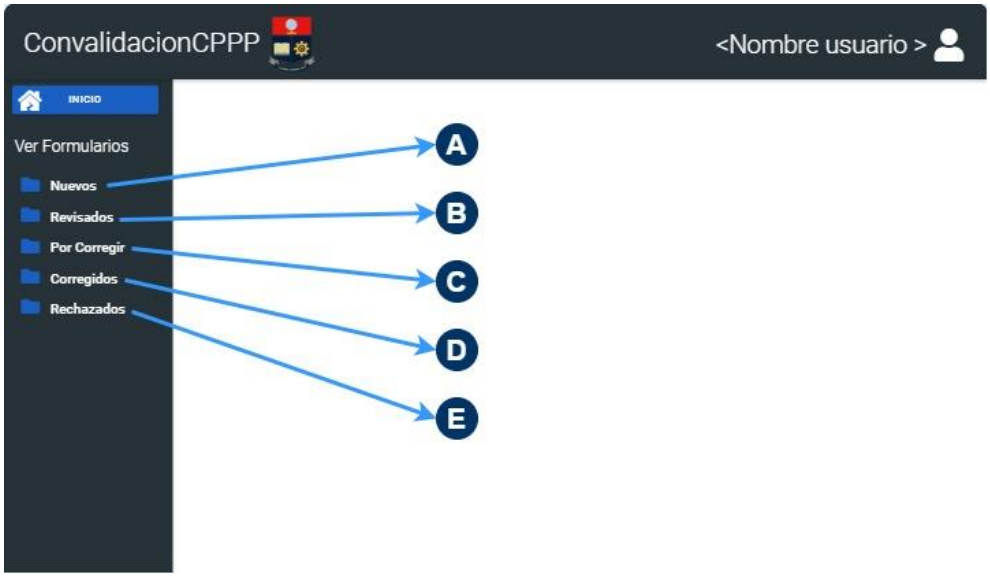

Figura 2.11. Diseño de la interfaz principal del rol Miembro de la CPP

En la Tabla 2.7 se da un detalle de los rótulos de la Figura 2.11.

Tabla 2.7. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Miembro de la CPP

Figura	Rótulo	Funcionalidad
2.11	A	Este icono muestra los formularios que ya han sido revisados por el tutor
	B	Este icono muestra los formularios que ya han sido revisados por el miembro de la CPP
	C	Este icono muestra los formularios en los que el miembro de la CPP solicita correcciones
	D	Este icono muestra los formularios que ya han sido corregidos
	E	Este icono muestra los formularios que han sido rechazados

En la Figura 2.12, se presenta el diseño del boceto de la interfaz gráfica para el rol Coordinador CPP y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

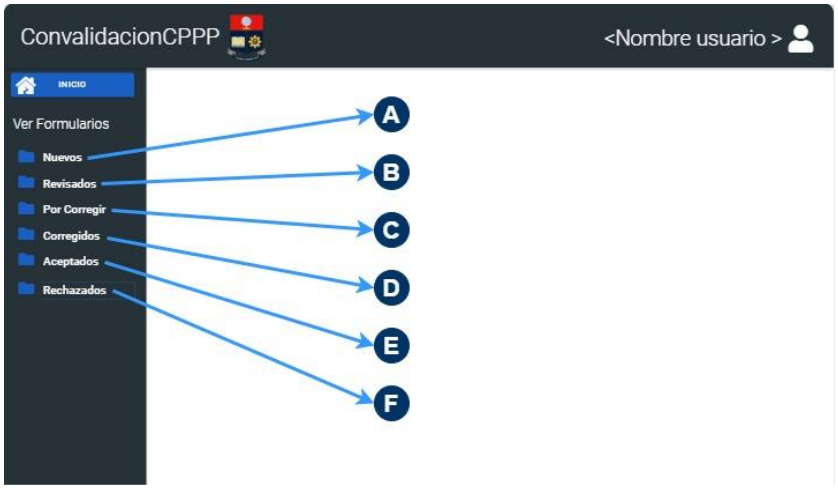

Figura 2.12. Diseño de la interfaz principal del rol Coordinador CPP

En la Tabla 2.8 se da un detalle de los rótulos de la Figura 2.12.

Tabla 2.8. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Coordinador CPP

Figura	Rótulo	Funcionalidad
2.12	A	Este icono muestra los formularios que ya han sido revisados por el miembro de la CPP
	B	Este incono muestra los formularios que ya han sido revisados por el coordinador de la CPP
	C	Este incono muestra los formularios en los que el coordinador de la CPP solicita correcciones
	D	Este icono muestra los formularios que ya han sido corregidos
	E	Este icono muestra los formularios que han terminado con éxito el proceso de convalidación
	F	Este icono muestra los formularios que han sido rechazados

En la Figura 2.13, se presenta el diseño del boceto de la interfaz gráfica para el rol Decano y en ella se muestran rótulos con letras que permiten identificar las funcionalidades para este rol.

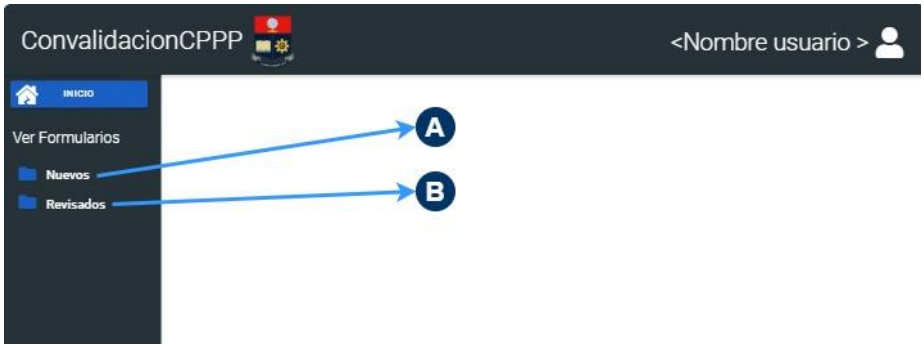

Figura 2.13. Diseño de la interfaz principal del rol Decano

En la Tabla 2.9 se da un detalle de los rótulos de la Figura 2.13.

Tabla 2.9. Descripción de las funcionalidades del boceto de la interfaz gráfica del rol Decano

Figura	Rótulo	Funcionalidad
2.13	A	Este icono muestra los formularios que ya han sido revisados por el coordinador de la CPP
	B	Este icono muestra los formularios que han terminado con éxito el proceso de convalidación

Los bocetos presentados en esta sección corresponden a las interfaces principales de cada uno de los roles de usuario de la aplicación web. Los iconos presentados en cada boceto cumplen con las funcionalidades que debe desarrollar cada rol. En la siguiente sección se presenta la implementación y desarrollo de la aplicación web ConvalidaciónCPPP.

2.3 IMPLEMENTACIÓN DE LA APLICACIÓN WEB

En este apartado se presenta el desarrollo de la aplicación web y las Api-rest. Para entender de mejor manera el desarrollo y llevar un orden cronológico de las actividades desarrolladas, este apartado se divide en cuatro secciones. En la sección 2.3.1 se presentan las instalaciones necesarias para configurar el entorno de trabajo. En la sección 2.3.2 se describe el desarrollo de la API university-resy. En la sección 2.3.3 se presenta el desarrollo de la API sri y en la última sección 2.3.4 se describe el desarrollo de la aplicación web ConvalidaciónCPPP.

2.3.1 ENTORNO DE TRABAJO.

Antes de comenzar con la implementación de la aplicación se debe preparar el entorno de trabajo. Como ya fue mencionado en el alcance del proyecto, se trabajará únicamente en un ambiente local, es por esto que los servicios tanto de las api-rest como de la aplicación web, se ejecutarán en una máquina local con las siguientes características: Sistema Operativo Windows 10, Almacenamiento SSD 260GB, memoria RAM de 8GB, procesador core i3-4010.

En esta máquina se deben instalar todos los requisitos que Laravel necesita para su funcionamiento. A continuación, se describen de una forma general cuáles son estos requisitos y en el Anexo I se presenta de una forma detallada y paso a paso como preparar el entorno de trabajo antes de crear un proyecto en Laravel.

2.3.1.1 Requerimientos de Laravel.

Antes de crear un proyecto en Laravel, se requieren cumplir con varios requisitos [26]:

- Servidor web Apache o Nginx
- PHP 7.3 o superior
- Base de datos: Postgresql, MySQL, Sqlite o sqlserver
- Extensiones php: BCMath, Ctype, JSON, Mbstring, OpenSSL, PDO, Tokenizer y XML

Todos estos son necesarios para el funcionamiento de Laravel en un servidor. Pero debido a que el proyecto está limitado a trabajar en un ambiente local, este trabajo no se centrará en la instalación de cada uno de estos requisitos en un servidor.

Al trabajar en una máquina con sistema operativo Windows, se puede instalar de manera sencilla un entorno de desarrollo que cumple con todos estos requisitos. El entorno de desarrollo con el que se trabaja este proyecto es Laragon, el cual es una suite de desarrollo para PHP que funciona sobre Windows diseñada especialmente para Laravel. Si se trabaja con otro entorno es importante verificar que cumpla con todos los requisitos antes mencionados.

2.3.1.2 Gestor de dependencias COMPOSER

Composer es una herramienta que permite gestionar dependencias en PHP. Declarar bibliotecas de las cuales depende un proyecto y las administrará (instalar/actualizar). Este gestor de dependencias realiza las siguientes acciones [27]:

- Declara bibliotecas de las cuales depende el proyecto.

- Averigua las versiones de los paquetes que pueden y necesitan instalarse y los instala dentro del proyecto.
- Puede actualizar todas las dependencias en un solo comando.

Composer se encarga de administrar paquetes o bibliotecas proyecto por proyecto, instalándolos en un directorio (por ejemplo, vendor) dentro del proyecto. Para administrar las dependencias usa un archivo dentro de un proyecto de Laravel, este archivo se llama: composer.json [27].

De igual manera en el Anexo I se presenta la instalación paso a paso del gestor de dependencias composer.

2.3.2 IMPLEMENTACIÓN DE LA API-REST UNIVERSITY-REST

Una vez instalados todos los requisitos de Laravel, incluyendo PHP y composer, se crea el primer proyecto en Laravel, el cual es la API university-rest. Esta api tiene el objetivo de simular el cruce de información de estudiantes y profesores de la EPN con la aplicación web, como se muestra en la Figura 2.5. En el Anexo II se describe paso a paso la creación y configuración de esta API. Sin embargo, a continuación, se describen las principales características de ésta para conocer su funcionamiento e importancia dentro de este proyecto.

2.3.2.1 Datos que contiene la API

La API university-rest tiene una base de datos con el mismo nombre del proyecto. En esta base de datos se han creado dos tablas mediante el uso de migraciones, estas tablas son: estudiantes y profesors. Estas contienen registros ficticios de estudiantes y profesores, los mismos que fueron creados mediante el uso de seeders. En la Figura 2.14 se muestra la tabla de estudiantes y en la Figura 2.15 la tabla de profesores de la API.

id	carrera	nombres	apellidos	cedula	correo	telefono	celular	epn
1	(RRA20) Telecomunicaciones	Avis	Willms	7916733640	jakob83@ritchie.com	4228364	908143837	884375756
2	(RRA20) Telecomunicaciones	Junius	Price	6818351150	smiller@nitzsche.com	5991244	930582256	685755424
3	(RRA20) Telecomunicaciones	Shanel	Weber	4454746567	ujenkins@gmail.com	2634928	962039718	620709861
4	(RRA20) Telecomunicaciones	Aditya	Haag	3709219265	brock.will@bruen.net	7345447	999680644	621339738
5	(RRA20) Telecomunicaciones	Marisa	Kuvalis	1477749394	icrooks@pollich.com	6027726	998982653	621063976

Figura 2.14. Tabla estudiantes de la Api university-rest

id	departamento	nombres	apellidos	cedula	correo	telefono	celular	epn
1	DETRI	Rodrigo	Pfannerstill	9723568038	sabrina.bruen@ledner.com	2836183	910305263	320183445
2	DETRI	Donato	Quigley	5543454598	baylee00@gmail.com	7072470	933869915	207810464
3	DETRI	Linda	Zemlak	2867766388	ykohler@marks.com	6107480	959848274	654954940
4	DETRI	Wilfredo	Harvey	4240159312	kuhn.franz@gulgowski.com	2012026	961001641	741015133
5	DETRI	Carmella	Predovic	9951073614	gerhold.jaime@yahoo.com	4615312	961570756	755960830

Figura 2.15. Tabla profesors de la Api university-rest

La aplicación ConvalidaciónCPPP realizará consultas a esta Api para obtener información de estudiantes y profesores presentada en las Figuras 2.14 y 2.15.

2.3.2.2 Configuración de rutas la API

Para que la api puede atender estas consultas, requiere de la configuración de una ruta, esta ruta se la configura en el archivo api.php el cual se encuentra en el directorio /routes. Las rutas creadas se presentan en el Código 2.1.

```
//Estudiantes
Route::post('estudiantesEPN', [EstudiantesController::class, 'indexByEPN']);
//Profesores
Route::post('profesoresEPN', [ProfesorController::class, 'indexByEPN']);
```

Código 2.1. Rutas para atender consultas desde la Api university-rest

En el Código 2.1 se puede ver que se han creado dos rutas con el método POST para procesar las consultas realizadas a las tablas estudiantes y profesores respectivamente. La creación de estas dos únicas rutas con el método POST se debe a que solo se requiere realizar consultas de información a la Api; de requerirse otras funcionalidades como crear, actualizar o eliminar registros, se deberán crear rutas adicionales para realizar estas acciones.

2.3.2.3 Controlador para procesar las consultas a la API university-rest

A las rutas presentadas en el Código 2.1 se les ha asignado un controlador y una función, las cuales son las encargadas de procesar y dar respuesta a la petición hecha por un cliente.

La función que da respuesta a las consultas realizadas a la tabla profesors se la debe escribir en el controlador *EstudiantesController* que se encuentra en el directorio *app/HTTP/Controllers.* En el Código 2.2 se muestra esta función.

```php
public function indexByEPN(Request $request)
{
    // Función para atender consultas de estudiantes
    $result = DB::table('estudiantes')
                ->select('*')
                ->where('cedula', $request->busqueda)
                ->first();
    return response()->json($result, 201);
}
```

Código 2.2. Función encargada de atender las consultas de estudiantes en la Api university-rest

Mientras que la función que da respuesta a las consultas realizadas a la tabla profesors se la debe escribir en el controlador *ProfesorController* que se encuentra en el directorio *app/HTTP/Controllers.* En el Código 2.3 se muestra esta función.

```php
public function indexByEPN(Request $request)
{
// Función para atender consultas de estudiantes
$result = DB::table('profesors')
                ->select('*')
                ->where('cedula', $request->busqueda)
                ->first();
    return response()->json($result, 201);
}
```

Código 2.3. Función encargada de atender las consultas de profesores en la Api university-rest

2.3.2.4 Resultado de consulta a la API university-rest

Para realizar las consultas a la Api se usa el protocolo HTTP, y como repuesta se tiene un archivo .json con la información solicitada, a continuación en las Figuras 2.16 y 2.17 se presenta la información obtenida de una consulta realizada a la tabla estudiantes y profesores respectivamente. Esta información servirá más adelante para llenar algunos campos del formulario de manera automática.

```
// 20220805173614
// http://cppp.test/estudiantesEPN

{
  "id": 13,
  "carrera": "(RRA20) Telecomunicaciones",
  "nombres": "Franz",
  "apellidos": "Kirlin",
  "cedula": "3024885839",
  "correo": "zboncak.katarina@hotmail.com",
  "telefono": "4806321",
  "celular": "923323572",
  "epn": "772174698",
  "created_at": "2022-08-05 22:34:14",
  "updated_at": "2022-08-05 22:34:14"
}
```


Figura 2.16. Respuesta de la Api university-rest (estudiante)


```
// 20220805173840
// http://cppp.test/profesoresEPN

{
  "id": 5,
  "departamento": "DETRI",
  "nombres": "Rosalyn",
  "apellidos": "Leannon",
  "cedula": "8026588360",
  "correo": "bayer.rhea@sporer.biz",
  "telefono": "2926484",
  "celular": "962139857",
  "epn": "895076354",
  "created_at": "2022-08-05 22:34:14",
  "updated_at": "2022-08-05 22:34:14"
}
```


Figura 2.17. Respuesta de la Api university-rest (profesores)

2.3.3 IMPLEMENTACIÓN DE API-REST SRI

EL segundo proyecto creado en Laravel es la API sri. Esta Api tiene el objetivo de simular el cruce de información de empresas registradas en SRI con la aplicación, como se muestra en la Figura 2.5. En el Anexo III se describe paso a paso la creación y configuración de esta API. Sin embargo, a continuación, se describen las principales características de esta Api para conocer su funcionamiento e importancia dentro de este proyecto.

2.3.3.1 Datos que contiene la API

La API sri tiene una base de datos con el mismo nombre del proyecto. En esta base de datos se ha creado una tabla llamada sris mediante el uso de migraciones. Esta tabla contiene registros de empresas reales registradas en el SRI, estos registros se los realizó

mediante el uso de seeders. Hay que considerar que, al tratarse de una simulación, la base de datos está limitada a 7 empresas. En la Figura 2.18 se muestra la tabla sris de la API.

id	razon_social	direccion	ciudad	pais	correo	telefono	celular	ruc	tipo_institucion
1	EMPRESA ELECTRICA QUITO S.A. E.E.Q.	Av. 10 de Agosto E1-24 y, Quito 170519	Quito	Ecuador	rosalia.dare@willms.com	7858517	980071459	1790053881001	PÚBLICA
2	EMPRESA PUBLICA METROPOLITANA DE AGUA POTABLE Y SA...	Italia, Quito 170515	Quito	Ecuador	alda.stanton@yahoo.com	4023056	946723632	1768154260001	PÚBLICA
3	Produbanco	Av. Simon Bolivar s/n Via A Nayon	Quito	Ecuador	streich.erik@yahoo.com	6452285	917283352	1790368718001	PRIVADA

Figura 2.18. Tabla sris de la Api sri

2.3.3.2 Configuración de rutas la API

Para que la api puede procesar las consultas, requiere de la configuración de una ruta, esta ruta se la configura en el archivo api.php el cual se encuentra en el directorio /routes. La ruta creada se presenta a continuación:

```
Route::post('srisRUC', [SrisController::class, 'indexByRUC']);
```

Código 2.4. Rutas para atender consultas desde la Api sri

En el Código 2.4 se puede observar que se ha creado una sola ruta con el método POST para atender las consultas a la tabla sris. La creación de esta única rutas con el método POST se debe a que solo se requiere realizar consultas de información a la API. De requerirse otras funcionalidades como crear, actualizar o eliminar registros, se deberán rutas adicionales para realizar estas acciones.

2.3.3.3 Controlador para procesar las consultas a la API sri

A la ruta presentada en el Código 2.4 se le ha asignado un controlador y una función, las cuales son las encargadas de procesar y dar respuesta a la petición hecha por un cliente.

La función que da respuesta a las consultas realizadas a la tabla sris se la debe escribir en el controlador *SrisController* que se encuentra en el directorio *app/HTTP/Controllers.* En el Código 2.5 se muestra esta función.

```php
public function indexByRUC(Request $request)
{
  $result = Sri::all();
// Enviar archivo .json con información de    las empresas
  return response()->json($result, 201);
}
```

Código 2.5. Función encargada de atender las consultas de empresas en la Api university-rest

2.3.3.4 Resultado de consulta a la API sri

Para realizar las consultas a la Api sri se usa el protocolo HTTP y como repuesta se tiene un archivo .json con la información solicitada, a continuación en la Figura 2.19 se presenta la información obtenida de la consulta realizada a la API.


```
// 20220805181615
// http://cppp.test/formularios/nuevo-formulario

[
  {
    "id": 1,
    "razon_social": "EMPRESA ELECTRICA QUITO S.A. E.E.Q.",
    "direccion": "Av. 10 de Agosto E1-24 y, Quito 170519",
    "ciudad": "Quito",
    "pais": "Ecuador",
    "correo": "rosalia.dare@willms.com",
    "telefono": "7858517",
    "celular": "980071459",
    "ruc": "1790053881001",
    "tipo_institucion": "PÚBLICA",
    "created_at": "2022-08-05T22:34:41.000000Z",
    "updated_at": "2022-08-05T22:34:41.000000Z"
  },
```


Figura 2.19. Respuesta de la Api sri (empresas)

2.3.4 IMPLEMENTACIÓN DE PROTOTIPO ConvalidaciónCPPP

La aplicación principal de este proyecto es el prototipo ConvalidaciónCPPP. Esta aplicación hará uso de las dos APIs presentadas en los puntos anteriores, esto con la finalidad de que algunos de los campos del formulario se llenen de forma automática, evitando así cometer muchos errores y haciendo que este proceso sea mucho más rápido. En esta parte del proyecto se presenta el desarrollo de la aplicación ConvalidaciónCPPP el cual es el producto final de este proyecto y se adjunta como Anexo IV. Y de igual manera los pasos para implementar el prototipo se describen en el Anexo V. Sin embargo, a continuación, se presentan las interfaces finales de cada rol de usuario.

2.3.4.1 Interfaces finales de cada rol de usuario

En la Figura 2.20 se presenta la interfaz final del rol Administrador, el cual cumple con el diseño presentado anteriormente en la Figura 2.7.

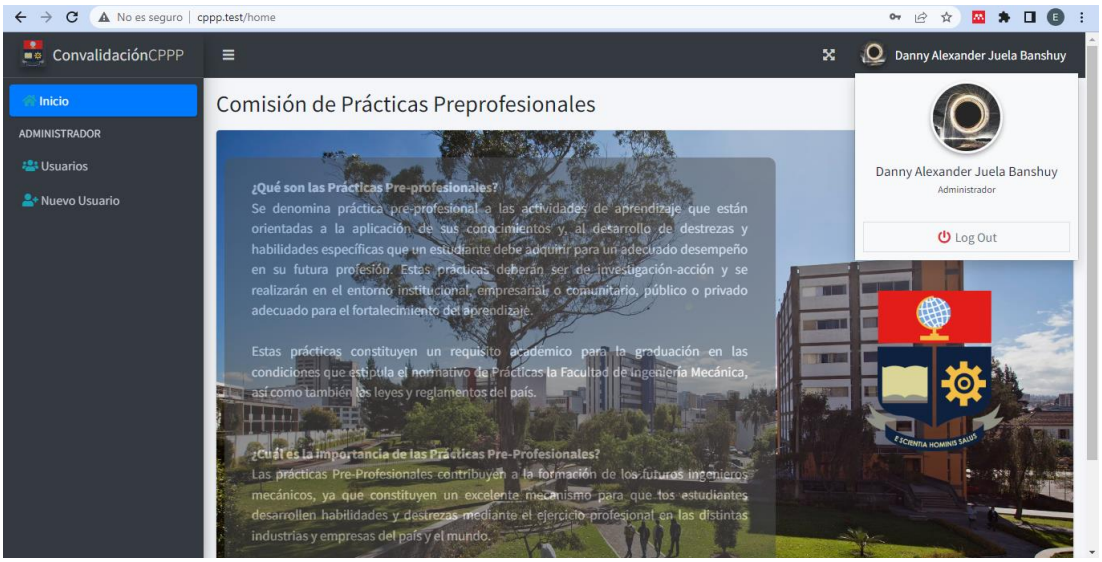

Figura 2.20. Interfaz final del rol Administrador

En la Figura 2.21 se presenta la interfaz final del rol Estudiante, el cual cumple con el diseño presentado anteriormente en la Figura 2.8.

Figura 2.21. Interfaz final del rol Estudiante

En la Figura 2.22 se presenta la interfaz final del rol Subdecano, el cual cumple con el diseño presentado anteriormente en la Figura 2.9.

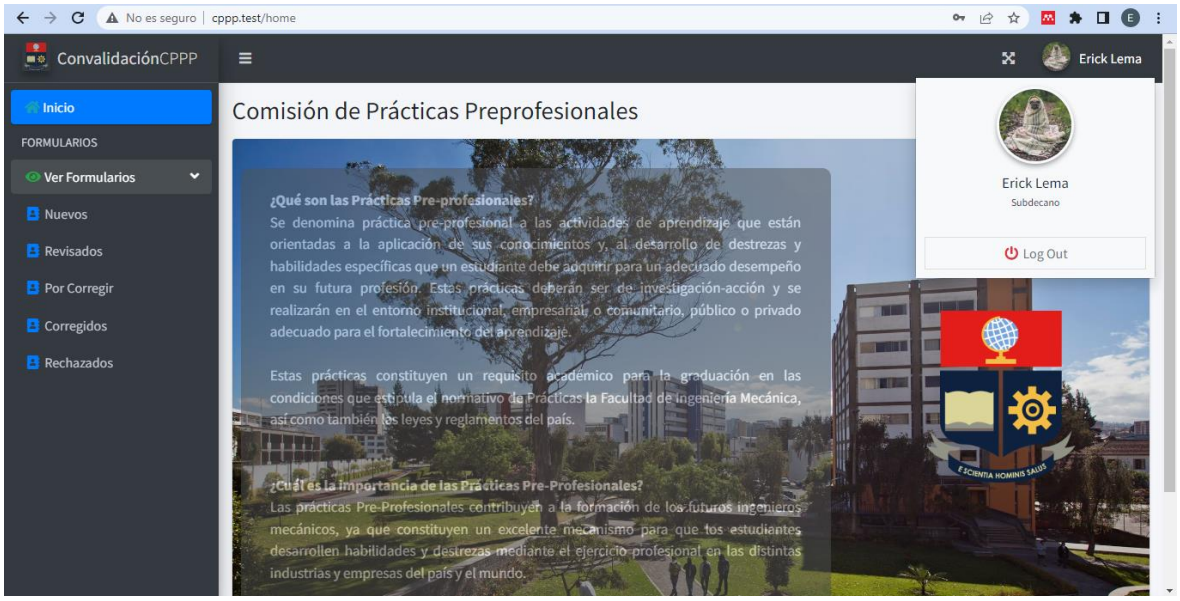

Figura 2.22. Interfaz final del rol Subdecano

En la Figura 2.23 se presenta la interfaz final del rol Tutor, el cual cumple con el diseño presentado anteriormente en la Figura 2.10.

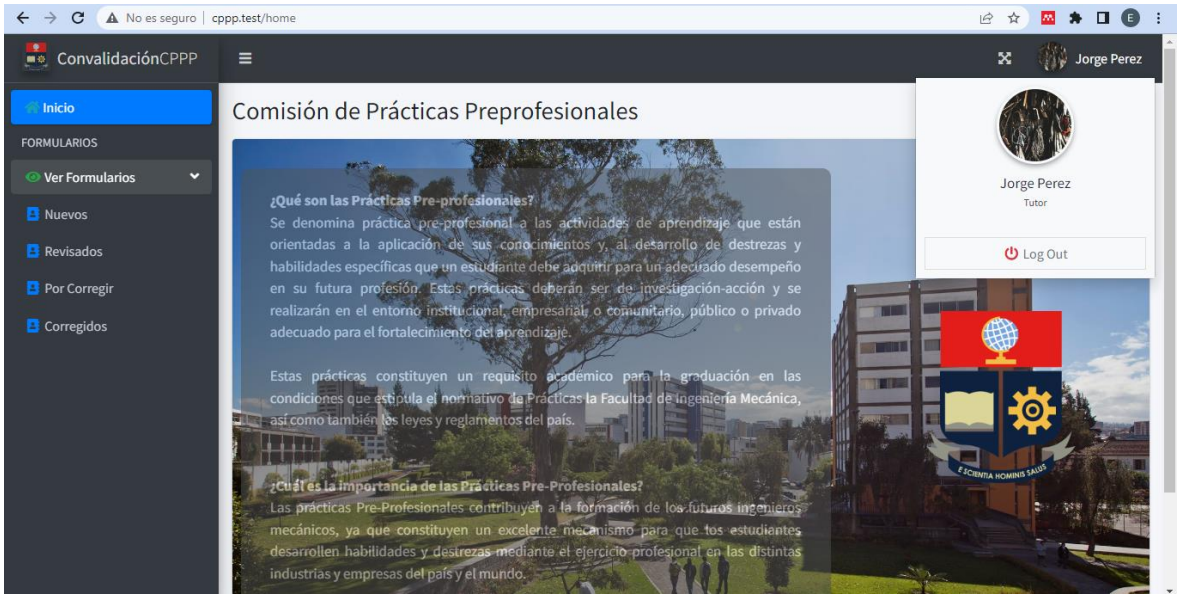

Figura 2.23. Interfaz final del rol Tutor

En la Figura 2.24 se presenta la interfaz final del rol Miembro de la CPP, el cual cumple con el diseño presentado anteriormente en la Figura 2.11.

Figura 2.24. Interfaz final del rol Miembro de la CPP

En la Figura 2.25 se presenta la interfaz final del rol Coordinador de la CPP, el cual cumple con el diseño presentado anteriormente en la Figura 2.12.

Figura 2.25. Interfaz final del rol Coordinador de la CPP

En la Figura 2.26 se presenta la interfaz final del rol Decano, el cual cumple con el diseño presentado anteriormente en la Figura 2.13.

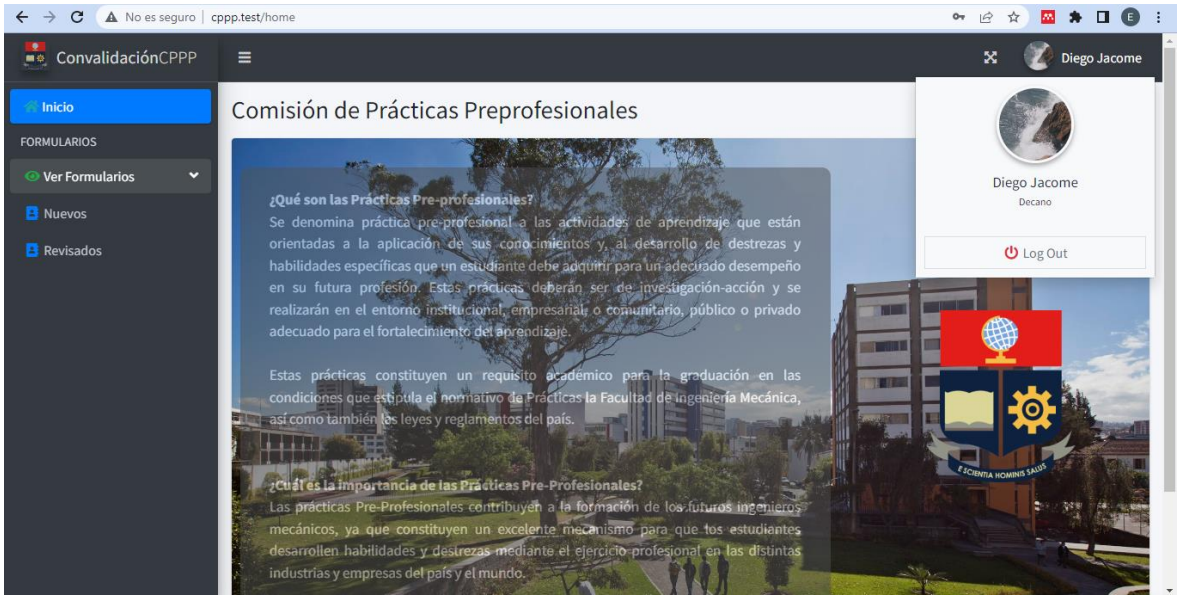

Figura 2.26. Interfaz final del rol Decano

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 RESULTADOS

Para validar el correcto funcionamiento se han desarrollado pruebas en dos entornos diferentes, el primero corresponde a pruebas realizadas por el desarrollador del proyecto y el segundo a pruebas realizadas con colaborados e involucrados en el proceso.

3.1.1 PRUEBAS DE FUNCIONAMIENTO REALIZADAS POR EL DESARROLLADOR

La primera prueba que se requieren cumplir corresponde a las realizadas por el desarrollador del proyecto, esto con la finalidad de confirmar el correcto desempeño de las funcionalidades más importantes de cada rol antes de compartir la aplicación con algunos involucrados en el proceso. En esta fase, se simulará el proceso de convalidación de prácticas preprofesionales, desde la creación de nuevos usuarios y asignación de roles, creación de nuevo formulario, asignación de tutor, revisión por parte del tutor, revisión por parte del miembro de la CPP, revisión por parte del coordinador de la CPP y decano. Las pruebas realizadas se presentan en el Anexo VI.

3.1.1.1 PRUEBAS CON COLABORADORES DEL PROCESO

Una vez realizadas las pruebas por parte del desarrollador y comprobando que la aplicación funciona de manera correcta, se realizan pruebas con algunos colaboradores e involucrados en el proceso, de los cuales se obtuvo los resultados que se exponen en esta sección. Para analizar estos resultados y la satisfacción de los usuarios se ha creado un cuestionario, con el cual se pretende conocer el impacto de la aplicación en el proceso de convalidación, los beneficios que esta puede aportar al proceso y también las mejoras que se pueden incluir en la aplicación. Las respuestas a la encuesta realizada se las encuentran en el Anexo VII. Además, en el anexo VIII se añaden los enlaces de las grabaciones de las pruebas realizadas con algunos estudiantes de la carrera de Telecomunicaciones de la EPN.

En la Tabla 3.1 se muestran las respuestas obtenidas de las encuestas realizadas. Se puede ver que las respuestas obtenidas son positivas lo que implica que el uso de la aplicación web dentro del proceso de convalidación tendría alta aceptación por parte de los estudiantes.

Tabla 3.1. Resultado de las encuestas realizadas a estudiantes de la carrera

#	Preguntas	Respuestas		Comentario
		SI	NO	
1	¿Le parece fácil entender la interfaz de usuario?	100%	0%	Los usuarios opinaron que la interfaz es sencilla y fácil de entender.
2	¿Los módulos implementados cumplen con su función?	100%	0%	Los usuarios determinaron que todos los módulos implementados cumplen con su función.
3	¿Le resulta fácil llenar el formulario o existió alguna confusión?	80%	0%	La mayoría de los usuarios opinaron que el formulario es fácil de llenar.
4	¿El sistema de correcciones es comprensible?	100%	0%	Los usuarios opinaron que el sistema de solicitud de correcciones es muy claro.
5	¿Le parece útil la aplicación para			Los usuarios creen que la aplicación contribuiría en gran

	realizar el proceso de convalidación de prácticas preprofesionales?	100%	0%	manera al proceso de convalidación de prácticas preprofesionales
6	¿Tiene alguna recomendación sobre mejoras que se puedan implementar?	N/A	N/A	N/A

Los resultados mostrados en la Tabla 3.1 reflejan la satisfacción de los usuarios con la implementación de esta aplicación en el proceso de convalidación de prácticas preprofesionales. Todos los estudiantes opinaron que esta contribuiría de gran manera al proceso, reduciendo tiempo y cometiendo menos errores en el llenado del formulario. Además, se dieron algunas sugerencias para mejorar la aplicación, las cuales serán descritas en las recomendaciones de este trabajo.

3.2 CONCLUSIONES

El producto final de este trabajo es el prototipo ConvalidaciónCPPP, mediante el cual se realiza la validación y automatización del proceso de convalidación de prácticas preprofesionales en FIEE. Para desarrollar este proyecto se adquirió conocimientos en bases de datos, lenguaje de programación php, y el desarrollo de frontend y backend mediante el framework Laravel. Las conclusiones a las que se llegó una vez culminado el desarrollo de este trabajo de titulación, se presentan a continuación.

- Para determinar los requerimientos del prototipo de aplicación web ConvalidaciónCPPP, se realizó un análisis del proceso de convalidación de prácticas preprofesionales y también del formulario FCP_001A. Del primer análisis realizado se determinó que se requieren 7 roles de usuarios en la aplicación web: Administrador, Estudiante, Subdecano, Tutor, Miembro de la CPP, Coordinador de la CPP y Decano. Todos los roles implementados participan de forma directa en proceso de convalidación a excepción del rol administrador, sin embargo, resulta un rol importante debido a que es el encargado de realizar el CRUD de usuarios y la asignación de roles.

- Mediante el análisis del proceso de convalidación de prácticas preprofesionales, también se determinó el flujo de trabajo y las actividades que desarrollan cada uno de los involucrados en el proceso de convalidación. De esta forma, se establecieron las funcionalidades para cada rol de usuario, las mismas que fueron consideradas para el diseño e implementación de la aplicación web.

- Del segundo análisis realizado correspondiente al formulario FCP_001A, se determinaron las validaciones que se deben hacer para el llenado del mismo. Además, se estableció que algunos campos del formulario deben ser llenados de forma automática, estos corresponden a la información de estudiantes y empresas nacionales registradas en el SRI. Debido a dificultades presentadas para acceder a la información real de estudiantes de la carrera de Telecomunicaciones se implementó una Api llamada university-rest la cual contiene información ficticia de estudiantes de la EPN. De igual manera debido a que no se permitió el uso del web service disponible en la institución para realizar consultas de empresas registradas en el SRI, se creó una Api llamada sri, la cual contiene una base de datos limitada de 7 empresas reales. Con la implementación de estas dos Api se simula el cruce de información con entidades internas y externas a la EPN, con la finalidad de conseguir llenar de forma automática los campos antes mencionados.

- La aplicación web ConvalidaciónCPPP, la Api university-rest y la Api sri fueron desarrolladas con el framework Laravel. En la implementación se consideraron todos los requerimientos establecidos en la fase de análisis y de igual manera se implementaron los módulos necesarios para que cada rol de usuario pueda cumplir con sus actividades dentro del proceso de convalidación.

- Antes de que la aplicación sea probada por algunos estudiantes de la carrera de Telecomunicaciones, se realizaron pruebas de funcionamiento por parte del desarrollador de la aplicación. En estas pruebas se simuló el proceso de convalidación de inicio a fin, el cual involucra las siguientes actividades: creación de usuarios, creación de nuevo formulario, asignación de tutor, revisión por parte del tutor, revisión por parte del miembro de la CPP, revisión por parte del coordinador de la CPP y decano. En estas pruebas participaron todos los roles cumpliendo correctamente con las funcionalidades asignados a cada uno, además se verificó que todo la información se guarda de forma correcta en la base de datos.

- La segunda fase de pruebas se las realizó con algunos estudiantes de la carrera de Telecomunicaciones, los cuales probaron la aplicación web de manera remota mediante la aplicación zoom. Ninguno de los estudiantes tuvo dificultad en usar la aplicación pues se dijo que su interfaz es sencilla y fácil de entender. Esta fase de pruebas culminó de forma exitosa, no se presentaron errores en de los módulos implementados y los estudiantes realizaron con éxito el llenado del formulario de forma rápida y sencilla.

- La implementación de la aplicación ConvalidaciónCPPP dentro del proceso de convalidación de prácticas preprofesionales proporciona grandes ventajas para todos aquellos actores que intervienen en este proceso. Los estudiantes que requieran realizar la convalidación pueden hacerlo de forma rápida y sencilla, minimizando los errores que se pueden cometer al momento de llenar el formulario principalmente en aquellos campos que involucran información de los estudiantes y empresas nacionales registradas en el SRI. Los estudiantes pueden realizar un seguimiento y conocer el estado en el que se encuentra el formulario, además, en caso de que un actor le solicite corregir algún campo del formulario, esto lo puede realizar de forma rápida. En general la aplicación permite reducir el tiempo que lleva terminar el proceso y minimizar los errores que se cometen en el mismo.

3.3 RECOMENDACIONES

La aplicación ConvalidaciónCPPP permite reducir el tiempo que toma realizar el proceso de convalidación. Está cumple con su propósito de validar y automatizar el llenado del formulario FCP_001A considerando las limitaciones y el alcance establecido para este proyecto; sin embargo, en caso de que la aplicación pueda ser implementada en la FIEE se presentan a continuación recomendaciones que se pueden considerar para implementaciones futuras.

- Si la Institución considera en implementar la aplicación web en el proceso de convalidación de prácticas preprofesionales, se debería realizar una reestructuración de este proceso. Al analizar el proceso de convalidación que estuvo vigente en el periodo académico 2021-B se puede observar que cuatro de los seis actores involucrados: el subdecano, el tutor, el miembro de la CPPP y el coordinador, revisan el formulario y la información llenada por el estudiante. Al realizar la misma actividad por estos cuatro actores el proceso se vuelve más largo y tarda más tiempo en finalizar. Por esta razón se debería realizar una

reestructuración y optimizar la revisión sin tener que repetir la misma actividad por tantos actores.

- Para que los campos correspondientes a la información del estudiantes y tutor sean llenado de forma automática con información real de estudiantes legítimos de la carrera de telecomunicaciones, se recomienda implementar un sistema que permita hacer consultas a las bases de datos de la DGIP de la cual se podría importar a esta aplicación la información de los estudiantes y profesores de la EPN.

- De igual manera para que los campos correspondientes a información de empresas nacionales sean llenados de forma automática con información real, se recomienda implementar un sistema de validación haciendo uso del servicio web que posee la DGIP de la EPN.

- Se recomienda implementar un sistema de notificaciones para la aplicación web, este sistema trabajará para cada uno de los roles, por ejemplo, cuando un estudiante crea un nuevo formulario, la aplicación debe informar al subdecano de la creación del mismo. De igual manera cuando se le asigna a un tutor un formulario, la aplicación debe informar al tutor que se le ha asignado un nuevo formulario. En el caso del estudiante, cuando un actor le solicite realizar una corrección, la aplicación deberá notificar al estudiante que debe realizar esta corrección. Además, beneficiaria de gran manera si las notificaciones además de realizarse en la aplicación se envíen por correo electrónico.

- La asignación de tutores se podría realizar de forma automática, es decir, que la aplicación web al momento en el que un estudiante cree un nuevo formulario podrá realizar la asignación automática de un tutor. Esta asignación no debe ser aleatoria, sino que, se la debe llevar de una manera organizada, realizando la asignación de formularios de forma equitativa para todos los tutores registrados en la aplicación web.

- Se recomienda también realizar copias de seguridad de la base de datos, a fin de tener respaldos de la información en caso de que exista algún problema en la base de datos actual.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] V. de Docencia, “Guía para Estudiantes 2021-B,” Oct. 2021, Accessed: Jan. 20, 2022. [Online]. Available: https://atenea.epn.edu.ec/handle/25000/727

[2] C. de Docencia, “Procedimiento Convalidación Actividades ExtraCurriculares,” Dec. 2021, Accessed: Jan. 20, 2022. [Online]. Available: https://atenea.epn.edu.ec/handle/25000/596

[3] D. Sklar, *Learning PHP : a gentle introduction to the Web’s most popular language*. Boston, 2016.

[4] R. Nixon, *Learning PHP, MySQL & JavaScript with Jquery, CSS & HTML5*. 2014.

[5] H. Beati, *PHP : creación de páginas web dinámicas*, vol. 2. 2015.

[6] F. Sierra, J. Acosta, J. Ariza, and M. Salas, “Estudio y análisis de los framework en php basados en el modelo vista controlador para el desarrollo de software orientado a la web | Investigación y desarrollo en TIC,” vol. 4. Universidad Simón Bolívar, Barranquilla, 2013. Accessed: Jul. 22, 2022. [Online]. Available: http://revistas.unisimon.edu.co/index.php/identic/article/view/2480

[7] M. Stauffer, *Laravel : up & running : a framework for building modern PHP apps*. 2019.

[8] “Directory Structure - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/9.x/structure (accessed Jul. 22, 2022).

[9] “Configuration - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/9.x/configuration#environment-configuration (accessed Jul. 22, 2022).

[10] S. Sinha, “Beginning Laravel A beginner’s guide to application development with Laravel 5.3,” *Beginning Laravel*, 2019.

[11] “What Is Front End and Back End Development? - Code Institute Global.” https://codeinstitute.net/global/blog/frontend-backend-development/ (accessed Jul. 22, 2022).

[12] “Database: Getting Started - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/9.x/database (accessed Jul. 22, 2022).

[13] “Database: Migrations - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/9.x/migrations (accessed Jul. 22, 2022).

[14] “Database: Seeding - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/9.x/seeding (accessed Jul. 22, 2022).

[15] “Controllers - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/9.x/controllers#main-content (accessed Jul. 22, 2022).

[16] “Simple Laravel CRUD with Resource Controllers | DigitalOcean.” https://www.digitalocean.com/community/tutorials/simple-laravel-crud-with-resource-controllers (accessed Jul. 22, 2022).

[17] I. Vanderlei, J. Araujo, R. Rocha, G. Silva, F. Pacheco, and J. Dantas, “Analysis of Laravel Framework Security Techniques against Web Application Attacks,” *Iberian Conference on Information Systems and Technologies, CISTI*, Jun. 2021, doi: 10.23919/CISTI52073.2021.9476475.

[18] X. Chen, Z. Ji, Y. Fan, and Y. Zhan, “Restful API Architecture Based on Laravel Framework,” *J Phys Conf Ser*, vol. 910, no. 1, p. 012016, Oct. 2017, doi: 10.1088/1742-6596/910/1/012016.

[19] C. Muyón and F. Montaluisa, “Métodos de seguridad de la información para proteger la comunicación y los datos de servicios web REST en peticiones HTTP utilizando JSON Web Token y Keycloak Red Hat Single Sign On,” *Ibérica de Sistemas e Tecnologias de Informação*, 2020.

[20] “Documentation | Laragon - portable, isolated, fast & powerful universal development environment for PHP, Node.js, Python, Java, Go, Ruby.” https://laragon.org/docs/ (accessed Jul. 22, 2022).

[21] J. L. Comesaña, “DESARROLLO WEB ENTORNO SERVIDOR Desarrollo de Aplicaciones Web”, Accessed: Aug. 15, 2022. [Online]. Available: https://www.sitiolibre.com/curso/pdf/DWES01.pdf

[22] T. Chusho and J. Li, “CONCEPTUAL MODELING FOR WEB APPLICATIONS AND DEFINITIONS OF BUSINESS LOGIC FOR END-USER-INITIATIVE DEVELOPMENT,” *Meiji University*, 2014, Accessed: Aug. 15, 2022. [Online]. Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.639.3984&rep=rep1&type=pdf

[23] “How To Run Laravel Development Server On A Different Port - Devtonight.” https://devtonight.com/articles/how-to-run-laravel-development-server-on-a-different-port (accessed Aug. 15, 2022).

[24] V. Nevado Cabello, “Introduccion a Las Bases de Datos Relacionales,” *Editorial Visión Libros*, p. 105, 2010, Accessed: Aug. 15, 2022. [Online]. Available: https://books.google.es/books?hl=es&lr=&id=0lUpB1lNUdIC&oi=fnd&pg=PA11&dq=base+de+datos+relacional&ots=sLXTG1rXRI&sig=x0uNe7BrHlpGrzDXIYtstM1QO84#v=onepage&q=base de datos relacional&f=false%0Ahttps://books.google.es/books?hl=es&lr=&id=0lUpB1lNUdIC&oi=fnd

[25] “Digital Product Design | Design Apps & Webpages | Uizard.” https://uizard.io/es/product/ (accessed Aug. 15, 2022).

[26] “Deployment - Laravel - The PHP Framework For Web Artisans.” https://laravel.com/docs/8.x/deployment#server-requirements (accessed Aug. 15, 2022).

[27] “Introduction - Composer.” https://getcomposer.org/doc/00-intro.md (accessed Aug. 15, 2022).

5 ANEXOS

ANEXO I. Instalación de requisitos de Laravel en Windows.

ANEXO II. Implementación de Api university-rest.

ANEXO III. Implementación de Api sri.

ANEXO IV. Códigos del prototipo de aplicación web ConvalidaciónCPPP, Api university-rest y Api sri.

ANEXO V. Implementación de prototipo de aplicación web ConvalidaciónCPPP.

ANEXO VI. Pruebas de funcionamiento por parte del desarrollador.

ANEXO VII. Encuestas realizadas a estudiantes de la carrera de Telecomunicaciones.

ANEXO VIII. Enlaces de grabaciones de pruebas realizadas con estudiantes de la carrera de Telecomunicaciones.

ANEXO I

En este anexo se describe el procedimiento paso a paso para preparar el entorno de trabajo antes de crear proyectos con Laravel. Las siguientes instalaciones y configuraciones, se las realizará en una máquina con sistema operativo Windows 10 con las siguientes características: Almacenamiento SSD 260GB, memoria RAM de 8GB, procesador core i3-4010U.

Los requisitos que tiene Laravel para su funcionamiento son los siguientes:

- Servidor web Apache o Nginx
- PHP 7.3 o superior
- Base de datos: Postgresql, MySQL, Sqlite o sqlserver
- Extensiones php: BCMath, Ctype, JSON, Mbstring, OpenSSL, PDO, Tokenizer y XML

Al trabajar en una máquina con sistema operativo Windows, se puede instalar de manera sencilla un entorno de desarrollo que cumple con todos estos requisitos. El entorno de desarrollo con el que se trabaja este proyecto es Laragon, el cual es una suite de desarrollo para PHP que funciona sobre Windows y fue diseñada especialmente para trabajar con Laravel.

1. Instalación de Laragon en Windows

Laragon se utiliza como un espacio seguro para trabajar en un sitio web, sin necesidad de alojarlo de manera online. Las características más destacadas de Laragon con respecto a otros entornos de desarrollo son:

- Crea Virtualhost de forma automática.
- Permite modificar la versión de PHP, Apache o MySQL/MariaDB.
- Permite compartir un proyecto local a través de Internet, para realizar pruebas antes de poner el proyecto en producción.
- Permite trabajar con Ngnix, además de Apache.
- Permite usar otros motores de base de datos como: MongoDB o PostgreSQL.

Los pasos para instalar y configurar Laragon son los siguientes:

1.1 Descargar Laragon

El archivo de instalación de Laragon se encuentra en el siguiente enlace: https://laragon.org/download/index.html, la versión que se recomienda descarga es la versión Full, puesto a que tiene mayores características que permiten trabajar sin problemas.

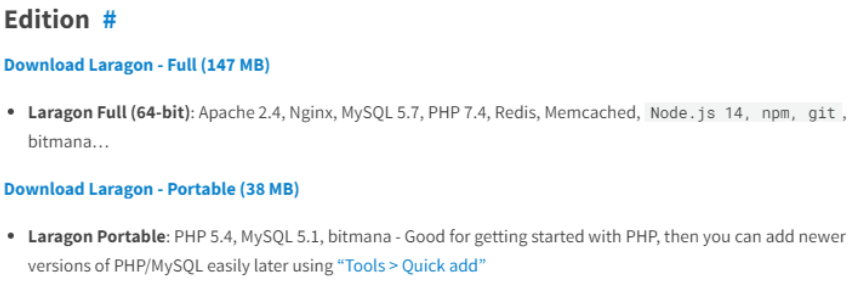

Figura 1. Descargar Laragon de la página oficial

1.2 Instalación de Laragon en Windows

Una vez descargado el instalador, se debe ejecutar el mismo y proceder con la instalación como se muestra en la Figura 2. En el momento de la instalación se deben mantener las configuraciones recomendadas por Laragon.

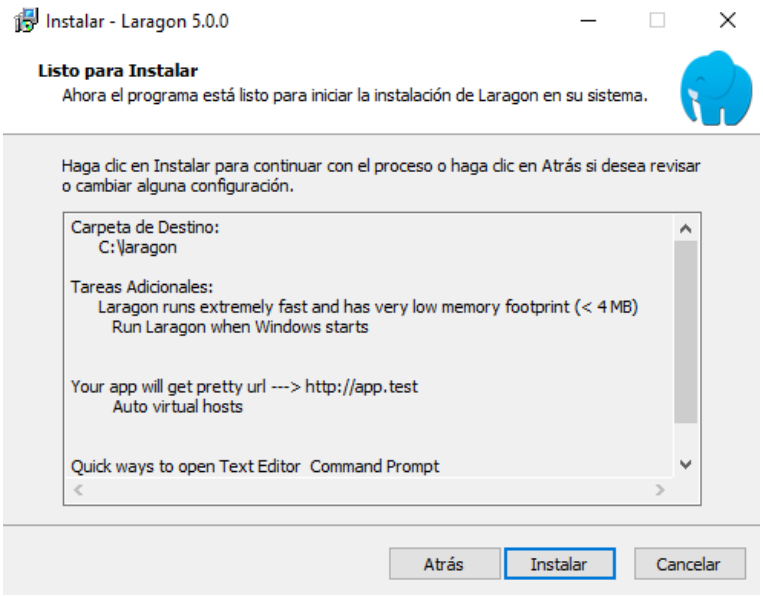

Figura 2. Instalación de Laragon en Windows

1.3 Verificación de instalación de Laragon

Una vez terminada la instalación, se puede ejecutar la aplicación y comprobar que en esta se encuentra funcionando de manera correcta el servidor web Apache y la base de datos MySQL. Esto se presenta en la Figura 3.

Figura 3. Verificación de instalación de Laragon

2. Instalación de composer en Windows

Una vez instalado el Servidor web con Laragon, se necesita instalar composer, el cuál es el gestor de dependencias que forma parte de los requisitos para trabajar con Laravel.

Los pasos para instalar composer son los siguientes:

2.1 Descargar composer

La forma de instalar Composer en una máquina con sistema operativo Windows es sencilla, consiste en descargar y ejecutar el archivo Composer-Setup.exe, este archivo se lo puede obtener de la página oficial de composer en el siguiente en enlace: https://getcomposer.org/download/

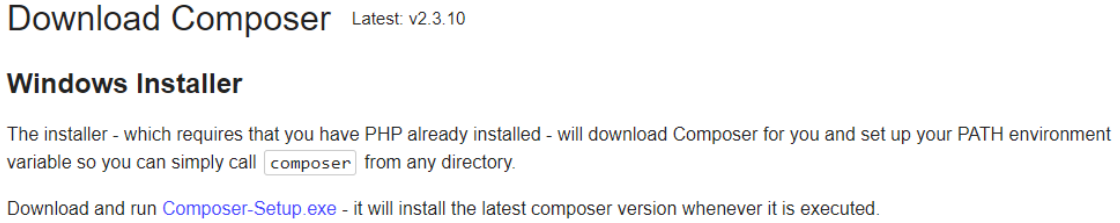

Figura 4. Descargar composer de la página oficial

2.2 Instalación de composer en Windows

El archivo Composer-Setup.exe instala la última versión de Composer. Al ejecutar el archivo de instalación se debe realizar una configuración importante que es añadir el path donde se encuentra instalado php, para este caso al estar usando laragon como servidor web, esta configuración se presenta en la Figura 5.

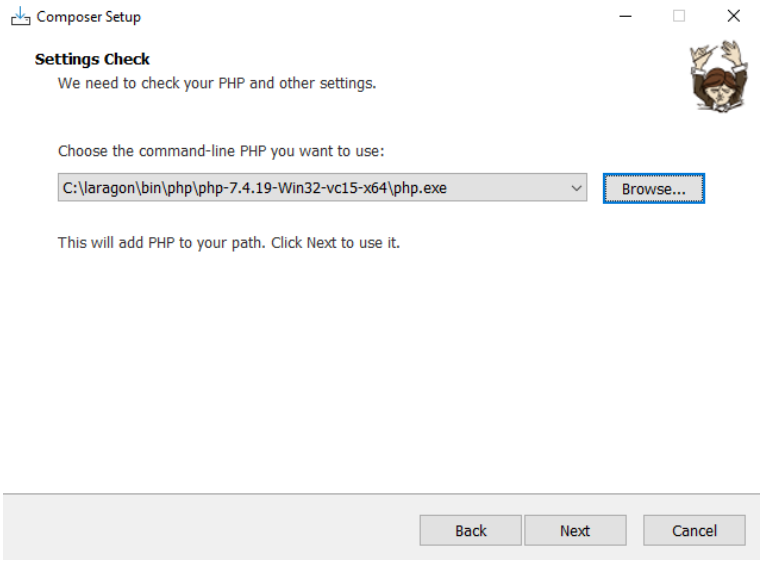

Figura 5. Configuración de path de php en composer

Luego de realizar la configuracion del path se debe continuar con la instalación, como se muestra en el Figura 6.

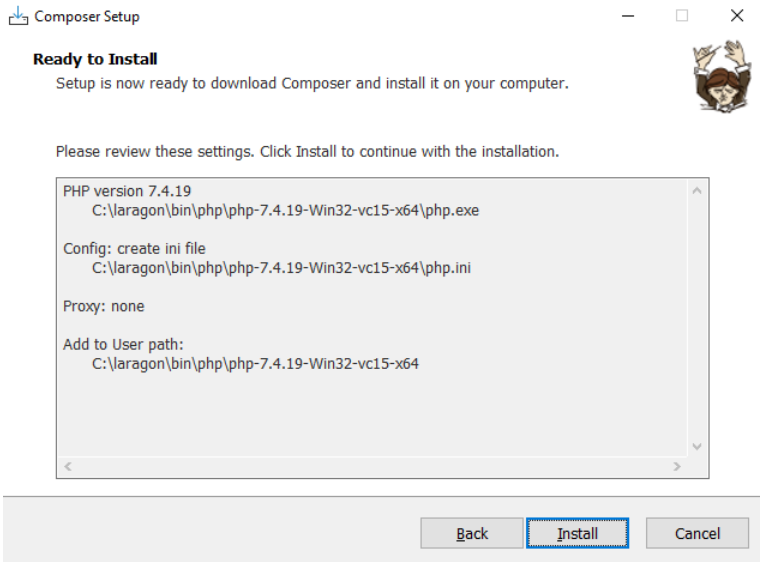

Figura 6. Instalación de composer en Windows

2.3 Verificación de instalación de composer

Terminada la instalacion de composer, se debe verificar si esta se encuentra funcionando correctamente en nuestra máquina, para esto desde una terminal de windows y se ejecuta el comando *composer,* en la Figura 7 se muestra que el gestor de dependencias se encuentra funcionando en nuestra máquina local.

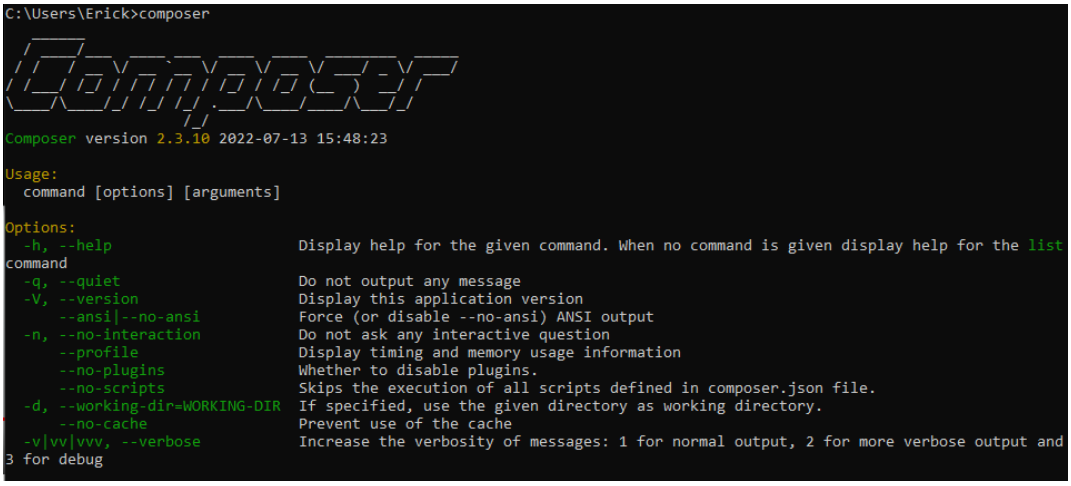

Figura 7. Verificación de instalación de composer

ANEXO II

Una vez preparado el entorno de trabajo. Se pueden empezar a crear proyectos en Laravel a través del comando *create-project* de composer.

El primer proyecto creado en Laravel es la API university-rest. Esta Api tiene el objetivo de simular el cruce de información de estudiantes y profesores de la EPN con la aplicación web. En este Anexo se describe paso a paso la creación, configuración y funcionamiento de esta API.

1. Creación de proyecto en Laravel

Como primer paso se tiene que crear un proyecto en Laravel con el nombre *university-rest*. Todos los proyectos son importantes crearlos en la carpeta raíz del servidor web de Laragon, para esto desde la terminar de Windows se debe ingresar a la carpeta raíz con el siguiente comando:

cd C: \laragon\www

Una vez en la carpeta raíz del servidor web, hay que ejecutar el siguiente comando para crear el proyecto *university-rest:*

composer create-project laravel/laravel university-rest

En la Figura 1 se presenta la creación del proyecto *university-rest* con composer.

```
C:\laragon\www>composer create-project laravel/laravel university-rest
Creating a "laravel/laravel" project at "./university-rest"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v8.6.12)
  - Downloading laravel/laravel (v8.6.12)
  - Installing laravel/laravel (v8.6.12): Extracting archive
Created project in C:\laragon\www\university-rest
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
Updating dependencies
```

Figura 1. Creación del proyecto university-rest en laravel

Después de haber creado el proyecto, se puede comprobar su funcionamiento usando el comando *php artisan serve* de Laravel. Para esto se debe ingresar al directorio del proyecto con el comando *cd university-rest,* estando en la carpeta del proyecto se ejecuta el comando antes mencionado para levantar el servicio, esto se presenta en la Figura 2.

```
C:\laragon\www>cd university-rest

C:\laragon\www\university-rest>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sun Jul 24 01:31:00 2022] PHP 7.4.19 Development Server (http://127.0.0.1:8000) started
```

Figura 2. Levantamiento del servicio del proyecto university-rest

Una vez levantado el servidor de desarrollo de Artisan, se puede comprobar su funcionamiento desde un navegador con la dirección: *http://127.0.0.1:8000.* El resultado se muestra en la Figura 3, donde se observa que el proyecto creado funciona correctamente.

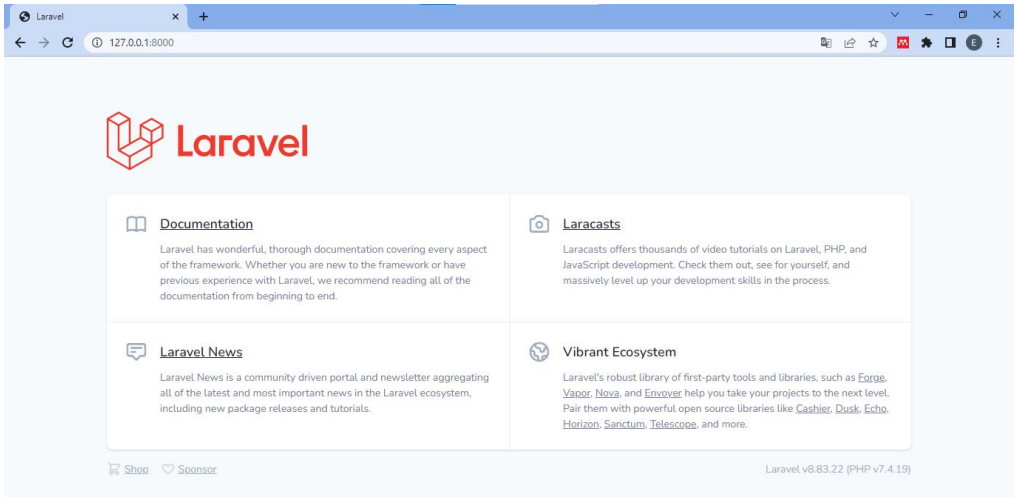

Figura 3. Visualización de proyecto university-rest desde navegador web

Debido a que este proyecto solo tiene la finalidad de procesar consultas, no se requiere desarrollar funcionalidades de frontend. Por esta razón las configuraciones realizadas se centran únicamente en el desarrollo de la API, las mismas que se detallan a continuación.

2. Conexión con la Base de datos

La primera configuración realizada es la conexión con base de datos. Como fue presentado en el Anexo I, Laragon nos provee del motor MySQL, el cual funciona mediante el puerto 3306. Esto es importante conocer para la configuración de conexión con la base de datos.

Antes de configurar la conexión, se debe crear una base de datos en MySQL. Para esto, mediante un administrador de bases de datos como phpMyadmin se puede acceder al motor de base de datos y crear una base llamada *university-rest.* La creación de la misma se presenta en la Figura 4.

Figura 4. Creación de base de datos university-rest desde phpMyadmin

Una vez creada la base de datos university-rest en MySQL, se debe configurar la conexión con la misma desde la API. Esta conexión se la puede realizar de dos maneras, desde el archivo de configuración *config/database.php* o desde el archivo *.env,* para este caso se usa la configuración mediante el archivo *.env.* La configuración realizada se presenta en el Código 1, donde es importante menciona que no se muestra todo el contenido del archivo, si no únicamente los campos que fueron configurados.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=university-rest
DB_USERNAME=root
DB_PASSWORD=
```

Código 1. Conexión con base de datos university-rest

3. Migración de tablas estudiantes y profesors

Se deben crear las tablas que contendrán la información ficticia de estudiantes y profesores. Laravel permite crear tablas mediante una funcionalidad llamada migraciones, con esta se pueden crear tablas de forma dinámica sin la necesidad de usar código SQL. Para crear los archivos encargados de realizar las migraciones tanto de la tabla profesors y la tabla estudiantes se usan los siguientes comandos:

php artisan make:migration create_estudiantes_table

php artisan make:migration create_profesors_table

Estos archivos se encuentran en el directorio *database/migrations.* Y en estos se crean las funciones encargadas de generar o eliminar las tablas en la base de datos, estas corresponden a las funciones up y down respectivamente. En el Código 2 se encuentra el código encargado de la migración de la tabla estudiantes, mientras que el Código 3 se encuentra el código encargado de la migración de la tabla profesors.

```php
public function up()
{
    Schema::create('estudiantes', function (Blueprint $table) {
        $table->id();
        $table->string('carrera')->nullable();
        $table->string('nombres')->nullable();
        $table->string('apellidos')->nullable();
        $table->string('cedula')->unique();
        $table->string('correo')->unique();
        $table->string('telefono')->nullable();
        $table->string('celular')->nullable();
        $table->string('epn')->unique();
        $table->timestamps();
    });
}

public function down()
{
    Schema::dropIfExists('estudiantes');
}
```

Código 2. Función para crear y eliminar la tabla estudiante

```php
public function up()
{
    Schema::create('profesors', function (Blueprint $table) {
        $table->id();
        $table->string('departamento')->nullable();
        $table->string('nombres')->nullable();
        $table->string('apellidos')->nullable();
        $table->string('cedula')->unique();
        $table->string('correo')->unique();
        $table->string('telefono')->nullable();
        $table->string('celular')->nullable();
        $table->string('epn')->unique();
        $table->timestamps();
    });
}

public function down()
{
    Schema::dropIfExists('profesors');
}
```

Código 3. Función para crear y eliminar la tabla profesors

Estando creadas las funciones que permiten generar y eliminar las tablas dentro de la base de datos, se deben ejecutar las migraciones con el comando *php artisan migrate,* de esta manera se generan las tablas en la base de datos. En la Figura 5 se presenta la ejecución del comando mencionado y el resultado obtenido.

```
C:\laragon\www\university-rest>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (137.13ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (103.75ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (70.16ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (124.22ms)
Migrating: 2022_05_01_022247_create_estudiantes_table
Migrated:  2022_05_01_022247_create_estudiantes_table (200.09ms)
Migrating: 2022_05_06_055345_create_profesors_table
Migrated:  2022_05_06_055345_create_profesors_table (199.77ms)
```

Figura 5. Ejecución de migraciones

Adicional al resultado presentado en la Figura 5 donde se muestra que las migraciones fueron exitosas, también se puede verificar desde el administrador de bases de datos phpMyadmin. Aquí se pude observar que las tablas estudiantes y profesors ya se encuentran creadas en la base de datos, como se muestra en la Figura 6.

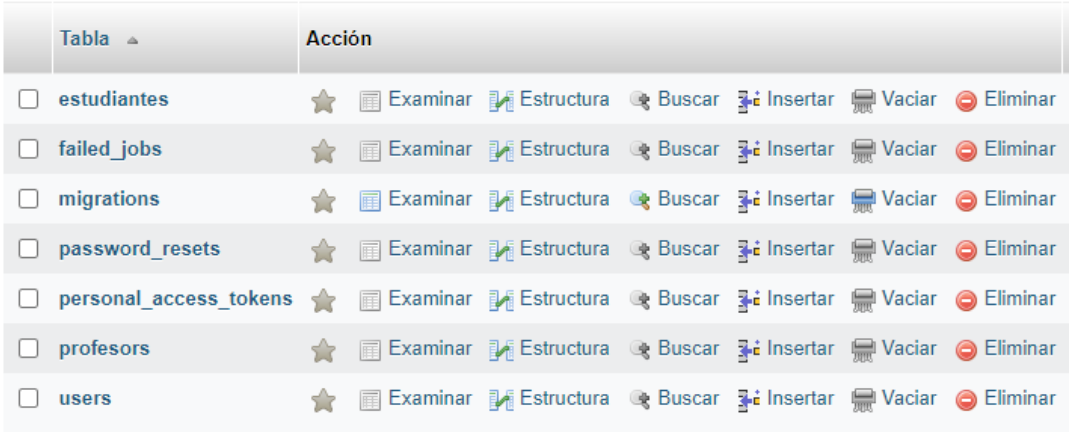

Figura 6. Tablas estudiantes y profesors en la base de datos university-rest

4. Generación de datos de prueba

Laravel proporciona la funcionalidad de crear datos de prueba en una base de datos, esto se lo realiza mediante seeders. Para este proyecto como ya fue mencionado, al tratarse de una simulación se trabajan con datos de profesores y estudiantes ficticios. Estos datos son generados mediante seeders alojados en el directorio *database/seeders,* en este directorio se encuentra un archivo llamado *DatabaseSeeder.php* donde se escriben las líneas de código que permiten llenar las tablas con datos aleatorio. El código utilizado para este propósito se presenta en el Código 4.

```php
public function run(){
    $faker = \Faker\Factory::create();
    DB::table('users')->insert([
        'name' => Str::random(10),
        'email' => $faker->email,
        'password' => Hash::make('password'), ]);
    $arrayCarrera = ['(RRA20) Telecomunicaciones'];
    $arrayDepartamento = ['DETRI'];
    for ($i = 0; $i < 20; $i++) {
        //Para crear los estudiantes en la Base de Datos
        \App\Models\Estudiante::create([
            'cedula' => $faker->numberBetween(1000000000, 9999999999),
            'correo' => $faker->unique()->email,
            'epn'=> $faker->numberBetween(100000000, 999999999),
            'carrera' => $arrayCarrera[0],
            'nombres' => $faker->firstname,
            'apellidos' => $faker->lastName,
            'telefono' => $faker->numberBetween(2000000, 7999999),
            'celular' => $faker->numberBetween(900000000, 999999999),
        ]);
        //Para crear los profesores en la Base de datos
        \App\Models\Profesor::create([
            'cedula' => $faker->numberBetween(1000000000, 9999999999),
            'correo' => $faker->unique()->email,
            'epn'=> $faker->numberBetween(100000000, 999999999),
            'departamento' => $arrayDepartamento[0],
            'nombres' => $faker->firstname,
            'apellidos' => $faker->lastName,
            'telefono' => $faker->numberBetween(2000000, 7999999),
            'celular' => $faker->numberBetween(900000000, 999999999),
        ]);
    }
}
```

Código 4. Función para generar datos aleatorios de estudiantes y profesores

Para ejecutar los seeders que han sido configurados en el proyecto, se lo hace mediante el comando *php artisan db:seed* como se muestra en la Figura 7.

```
C:\laragon\www\university-rest>php artisan db:seed
Database seeding completed successfully.
```

Figura 7. Ejecución de seeders

Para comprobar que los datos de prueba han sido generados correctamente, se debe acceder a la base de datos mediante el administrador phpMyadmin. Aquí se encontrarán los siguientes resultados:

- La tabla de estudiantes contiene la información de: carrera, nombres, apellidos, número único, cédula, teléfono, celular y correo, como se puede observar en la Figura 8.

id	carrera	nombres	apellidos	cedula	correo	telefono	celular	epn
1	(RRA20) Telecomunicaciones	Avis	Willms	7916733640	jakob83@ritchie.com	4228364	908143837	884375756
2	(RRA20) Telecomunicaciones	Junius	Price	6818351150	smiller@nitzsche.com	5991244	930582256	685755424
3	(RRA20) Telecomunicaciones	Shanel	Weber	4454746567	ujenkins@gmail.com	2634928	962039718	620709861
4	(RRA20) Telecomunicaciones	Aditya	Haag	3709219265	brock.will@bruen.net	7345447	999680644	621339738
5	(RRA20) Telecomunicaciones	Marisa	Kuvalis	1477749394	icrooks@pollich.com	6027726	998982653	621063976

Figura 8. Tabla estudiantes con datos de prueba

- La tabla profesores por otro lado contiene la información de: departamento, nombres, apellidos, número único, cédula, teléfono, celular y correo, como se puede observar en la Figura 9.

id	departamento	nombres	apellidos	cedula	correo	telefono	celular	epn
1	DETRI	Rodrigo	Pfannerstill	9723568038	sabrina.bruen@ledner.com	2836183	910305263	320183445
2	DETRI	Donato	Quigley	5543454598	baylee00@gmail.com	7072470	933869915	207810464
3	DETRI	Linda	Zemlak	2867766388	ykohler@marks.com	6107480	959848274	654954940
4	DETRI	Wilfredo	Harvey	4240159312	kuhn.franz@gulgowski.com	2012026	961001641	741015133
5	DETRI	Carmella	Predovic	9951073614	gerhold.jaime@yahoo.com	4615312	961570756	755960830

Figura 9. Tabla profesors con datos de prueba

5. Configuración de rutas en la API

El objetivo de esta Api es procesar las consultas realizadas por la aplicación ConvalidaciónCPPP, estas consultas se las realiza mediante el número de cédula, es decir que el cliente en su petición enviará un número cédula y la Api realizará una búsqueda en su base de datos. De encontrar un registro con este número de cédula la Api responderá con un archivo .json que contendrá toda la información del registro correspondiente al número de cédula.

Para que la api puede procesar estas consultas, requiere de la configuración de una ruta, esta ruta se la configura en el archivo api.php el cual se encuentra en el directorio /routes. Las rutas creadas se presentan en el Código 5.

```
//Estudiantes
Route::post('estudiantesEPN', [EstudiantesController::class, 'indexByEPN']);
//Profesores
Route::post('profesoresEPN', [ProfesorController::class, 'indexByEPN']);
```

Código 5. Rutas para procesar consultas desde la Api

En el Código 5 se puede ver que se han creado dos rutas con el método POST para procesar las consultas realizadas a las tablas estudiantes y profesores respectivamente. La creación de estas dos únicas rutas con el método POST se debe a que solo se requiere realizar consultas de información a la Api, de requerirse otras funcionalidades como crear, actualizar o eliminar registros, se deberán crear rutas adicionales para realizar estas acciones.

6. Controlador para procesar las consultas a la API university-rest

A las rutas presentadas en el Código 5 se les ha asignado un controlador y una función, las cuales son las encargadas de procesar y dar respuesta a la petición hecha por un cliente.

Para crear el controlador que se encarga de procesar las consultas a la tabla estudiantes se usa el siguiente comando:

php artisan make:controller EstudiantesController

Mientras que, para crear el controlador que se encarga de procesar las consultas a la tabla profesors se usa el siguiente comando:

php artisan make:controller ProfesorController

La función que da respuesta a las consultas realizadas a la tabla profesors se la debe escribir en el controlador *EstudiantesController* que se encuentra en el directorio *app/HTTP/Controllers. Esta función* se muestra en el Código 6.

```php
public function indexByEPN(Request $request)
{
    // Función para procesar consultas de estudiantes
    $result = DB::table('estudiantes')
                ->select('*')
                ->where('cedula', $request->busqueda)
                ->first();
    return response()->json($result, 201);
}
```

Código 6. Función encargada de procesar las consultas de estudiantes

Mientras que la función que da respuesta a las consultas realizadas a la tabla profesors se la debe escribir en el controlador *ProfesorController* que se encuentra en el directorio *app/HTTP/Controllers. Esta función* se muestra en el Código 7.

```php
public function indexByEPN(Request $request)
{
//Función para procesar consultas de estudiantes
$result = DB::table('profesors')
                ->select('*')
                ->where('cedula', $request->busqueda)
                ->first();
    return response()->json($result, 201);
}
```

Código 7. Función encargada de procesar las consultas de profesores

ANEXO III

El segundo proyecto creado en Laravel es la API sri, esta tiene el objetivo de simular el cruce de información de empresas registradas en el Servicio de Rentas Internas (SRI) con la aplicación web. En este Anexo se describe paso a paso la creación, configuración y funcionamiento de esta API.

1. Creación de proyecto en Laravel

Como primer paso se tiene que crear un Proyecto en Laravel con el nombre nombre *sri*. Todos los proyectos son importantes crearlos en la carpeta raíz del servidor web de Laragon, para esto desde la terminar de Windows se debe ingresar a la carpeta raíz con el siguiente comando:

cd C: \laragon\www

Una vez en la carpeta raíz del servidor web, hay que ejecutar el siguiente comando para crear el proyecto *sri:*

composer create-project laravel/laravel sri

En la Figura 1 se presenta la creación del proyecto *sri* con composer.

```
C:\laragon\www>composer create-project laravel/laravel sri
Creating a "laravel/laravel" project at "./sri"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v8.6.12)
  - Installing laravel/laravel (v8.6.12): Extracting archive
Created project in C:\laragon\www\sri
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
```

Figura 1. Creación del proyecto sri en laravel

Después de haber creado el proyecto, se puede comprobar su funcionamiento usando el comando *php artisan serve* de Laravel. Para esto se debe ingresar al directorio del proyecto con el comando *cd sri,* estando en la carpeta del proyecto se ejecuta el comando antes mencionado para levantar el servicio, esto se presenta en la Figura 2.

```
C:\laragon\www>cd sri

C:\laragon\www\sri>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sun Jul 24 01:59:44 2022] PHP 7.4.19 Development Server (http://127.0.0.1:8000) started
```

Figura 2. Levantamiento del servicio del proyecto university-rest

Una vez levantado el servidor de desarrollo de Artisan, se puede comprobar su funcionamiento desde un navegador web accediendo a la dirección: *http://127.0.0.1:8000.* El resultado se muestra en la Figura 3, donde se puede observar que el proyecto creado funciona correctamente.

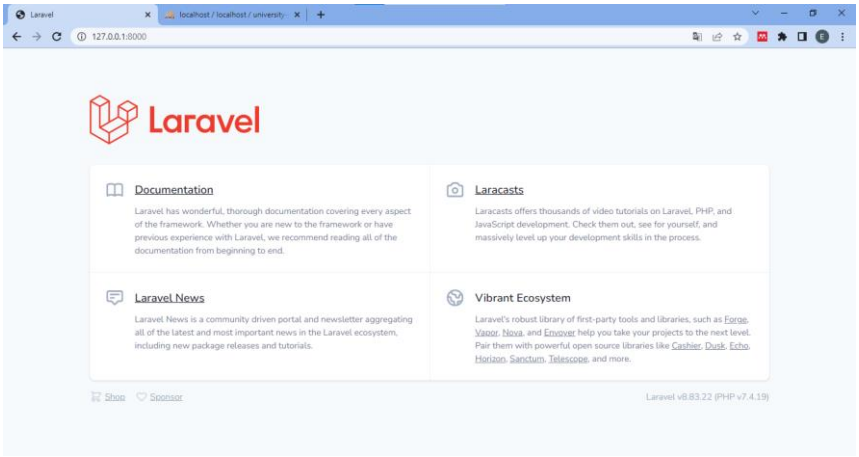

Figura 3. Visualización de proyecto sri desde navegador web

Debido a que este proyecto solo tiene la finalidad de procesar consultas, no se requiere desarrollar funcionalidades de frontend. Por esta razón las configuraciones realizadas se centran únicamente en el desarrollo de la API, la misma que se irán detallando a continuación.

2. Conexión con la Base de datos

La primera configuración realizada es la conexión con base de datos. Como fue presentado en el Anexo I, Laragon nos provee del motor de base de datos MySQL, el cual funciona mediante el puerto 3306. Esto es importante conocer para la configuración de conexión con la base de datos.

Antes de configurar la conexión, se debe crear una base de datos en MySQL. Para esto mediante un administrador de bases de datos como phpMyadmin se puede acceder al motor de base de datos y crear una base llamada *sri.* La creación de esta base de la misma se presenta en la Figura 4.

Figura 4. Creación de base de datos university-rest desde phpMyadmin

Una vez creada la base de datos sri en MySQL, se debe configurar la conexión con la misma desde la API. Esta conexión se la puede realizar de dos maneras, desde el archivo de configuración *config/database.php* o desde el archivo *.env,* para este caso se usa la configuración mediante el archivo *.env.* La configuración realizada se presenta en el Código 1, es importante menciona que no se muestra todo el contenido del archivo, si no únicamente los campos que fueron configurados.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=SRI
DB_USERNAME=root
DB_PASSWORD=
```

Código 1. Conexión con base de datos SRI

3. Migración de tabla sris

Después de establecer la conexión con la base de datos, se debe crear la tabla que contendrá la información de empresas registradas en el SRI. Laravel permite crear las tablas mediante una funcionalidad llamada migraciones, con esta se pueden crear tablas de forma dinámica sin la necesidad de usar código SQL. Para crear el archivo encargado de realizar la migración de latabla sris se usa el siguiente comando:

php artisan make:migration create_Sri_table

Este archivo se encuentra en el directorio *database/migrations.* Y en este se crea las funciones encargadas de generar o eliminar la tabla, estas corresponden a las funciones up y down respectivamente. En el Código 2 se encuentra el código encargado de la migración de la tabla sris.

```php
public function up()
{
    Schema::create('sris', function (Blueprint $table) {
        $table->id();
        $table->string('razon_social')->nullable();
        $table->string('direccion')->nullable();
        $table->string('ciudad')->nullable();
        $table->string('pais')->nullable();
        $table->string('correo')->unique();
        $table->string('telefono')->nullable();
        $table->string('celular')->nullable();
        $table->string('ruc')->unique();
        $table->string('tipo_institucion')->nullable();
        $table->timestamps();
    });
}
public function down()
{
    Schema::dropIfExists('sris');
}
```

Código 2. Función para crear o eliminar tabla sris

Estando creadas las funciones que permiten generar y eliminar la tabla sris en la base de datos, se deben ejecutar la migración con el comando *php artisan migrate,* de esta manera se crean las tablas en la base de datos. En la Figura 5 se presenta la ejecución del comando mencionado y el resultado obtenido.

```
C:\laragon\www\sri>php artisan migrate
Migration table created successfully.
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (146.97ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (175.68ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (123.35ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (134.33ms)
Migrating: 2022_05_01_022247_create_Sri_table
Migrated:  2022_05_01_022247_create_Sri_table (116.93ms)
```

Figura 5. Ejecución de migraciones

Adicional al resultado presentado en la Figura 5, donde se muestra que las migraciones fueron exitosas, también se puede verificar desde el administrador de bases de datos phpMyadmin. Aquí se puede verificar que la tabla sris, ya se encuentran en la base de datos, como se muestra en la Figura 6.

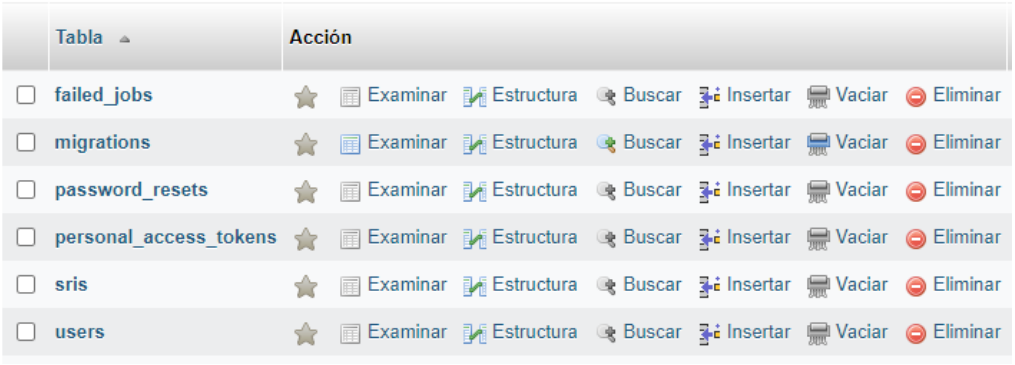

Figura 6. Tabla sris en la base de datos SRI

4. Generación de datos de prueba

Para este proyecto como ya fue mencionado al tratarse de una simulación, se deben ingresar de forma manual datos de algunas empresas reales, en este caso se ingresarán 7 empresas. Estos datos pueden ser cargados a la base mediante seeders, los cuales están alojados en el directorio *database/seeders,* en este directorio se encuentra un archivo llamado *DatabaseSeeder.php* donde se escriben las líneas de código que permiten llenar las tablas con estos datos. El código utilizado para este propósito se presenta en el Código 3.

```php
public function run(){
    $faker = \Faker\Factory::create();

    DB::table('users')->insert([
        'name' => Str::random(10),
        'email' => $faker->email,
        'password' => Hash::make('password'),
    ]);

$arrayRazonsocial = ['EMPRESA ELECTRICA QUITO S.A. E.E.Q.', 'EMPRESA PUBLICA METROPOLITANA DE AGUA POTABLE Y SANEAMIEN
$arrayDireccion = ['Av. 10 de Agosto E1-24 y, Quito 170519', 'Italia, Quito 170515', 'Av. Simon Bolivar s/n Via A Nayo
$arrayRuc = ['1790053881001', '1768154260001', '1790368718001', '0991327371001', '1792457157001', '1790896269001', '17
$arrayTipoinstitucion = ['PÚBLICA', 'PÚBLICA', 'PRIVADA', 'PRIVADA', 'PRIVADA', 'PRIVADA', 'PRIVADA'];
    $arrayCiudad = ['Quito'];
    $arrayPais = ['Ecuador'];

    //Para ingresar los datos en la base de datos
    for ($i = 0; $i < 7; $i++) {

        \App\Models\Sri::create([
            'razon_social' => $arrayRazonsocial[$i],
            'direccion' => $arrayDireccion[$i],
            'ciudad' => $arrayCiudad[0],
            'pais' => $arrayPais[0],
            'correo' => $faker->unique()->email,
            'telefono' => $faker->numberBetween(2000000, 7999999),
            'celular' => $faker->numberBetween(900000000, 999999999),
            'ruc'=> $arrayRuc[$i],
            'tipo_institucion' => $arrayTipoinstitucion[$i],
        ]);
    }
}
```

Código 3. Función para llenar la tabla sris con datos de empresas

Para ejecutar los seeders que han sido configurados en el proyecto, se lo hace mediante el comando *php artisan db:seed* como se muestra en la Figura 7

```
C:\laragon\www\sri>php artisan db:seed
Database seeding completed successfully.
```

Figura 7. Ejecución de seeders

Para comprobar que los datos de prueba han sido generados correctamente, se debe acceder a la base de datos mediante en administrador phpMyadmin. Aquí se encontrarán los siguientes resultados:

- La tabla de sris contiene la información de: ruc, razón social, dirección, país/ciudad, teléfono, celular, correo y tipo de institución, como se puede observar en la Figura 8.

id	razon_social	direccion	ciudad	pais	correo	telefono	celular	ruc	tipo_institucion
1	EMPRESA ELECTRICA QUITO S.A. E.E.Q.	Av. 10 de Agosto E1-24 y, Quito 170519	Quito	Ecuador	imogene09@casper.com	5755812	934363709	1790053881001	PÚBLICA
2	EMPRESA PUBLICA METROPOLITANA DE AGUA POTABLE Y SA...	Italia, Quito 170515	Quito	Ecuador	ruecker.alycia@heathcote.com	4375277	997825121	1768154260001	PÚBLICA
3	Produbanco	Av. Simon Bolivar s/n Via A Nayon	Quito	Ecuador	chris.auer@gmail.com	2517765	921688487	1790368718001	PRIVADA

Figura 8. Tabla sris con datos de empresas del SRI

5. Configuración de rutas en la API

El objetivo de esta Api es procesar consultas de empresas realizadas por la aplicación ConvalidaciónCPPP, esta consulta es diferente a la Api university-rest, en este caso en las consultas realizadas a la Api, esta envía un archivo .json con toda la información de la tabla sris, para que la búsqueda por ruc se la realiza en la aplicación ConvalidaciónCPPP.

Para que la api puede procesar estas consultas, requiere de la configuración de una ruta, esta ruta se la configura en el archivo api.php el cual se encuentra en el directorio /routes. La ruta creada se presenta en el Código 4.

```
Route::post('srisRUC', [SrisController::class, 'indexByRUC']);
```

Código 4. Rutas para procesar consultas desde la Api

En el Código 4 se puede ver que se ha creado una sola ruta con el método POST para procesar las consultas realizadas a la tabla sris. La creación de esta única rutas con el método POST se debe a que solo se requiere realizar consultas de información a la API. De requerirse otras funcionalidades como crear, actualizar o eliminar registros, se deberán rutas adicionales para realizar estas acciones.

6. Controlador para procesar las consultas a la API sri

A la ruta presentada en el Código 4 se le ha asignado un controlador y una función, las cuales son las encargadas de procesar y dar respuesta a la petición hecha por un cliente.

Para crear el controlador que se encarga de procesar las consultas se usa el siguiente comando:

php artisan make:controller SrisController

La función que da respuesta a las consultas realizadas a la tabla sris se la debe escribir en el controlador *SrisController* que se encuentra en el directorio *app/HTTP/Controllers. Esta función* se muestra en el Código 5.

```
public function indexByRUC(Request $request)
{
  $result = Sri::all();
// Enviar archivo .json con información de    las empresas
  return response()->json($result, 201);
}
```

Código 5. Función encargada de procesar las consultas de empresas

ANEXO IV

En este Anexo se encuentran los códigos implementados en este proyecto los cuales incluye, Api university-rest, Api sri y prototipo de aplicación web ConvalidaciónCPPP. Los tres proyectos se encuentran comprimidos en un archivo .zip el cual se adjunta a este trabajo de titulación.

ANEXO V

La aplicación principal de este proyecto se la ha nombrado ConvalidaciónCPPP, esta tiene el objetivo de validar y automatizar el llenado del formulario FCP_001A. En el presente Anexo se describe la creación y configuración de esta aplicación web. Debido a la extensión y gran cantidad de códigos realizados, se describirán los aspectos más importantes para la creación del proyecto. Este Anexo se divide en 8 secciones de la siguiente manera: Sección 1 Creación del proyecto, Sección 2 Conexión con la base de datos, Sección 3 Migraciones, Sección 4 Sistema de login, Sección 5 Instalación de plantilla adminLTE, Sección 6 Creación de rutas, Sección 7 Frontend de acuerdo con el rol de usuario y Sección 8 Backend

De requerir un análisis más profundo de los códigos implementados, estos se encuentran en el Anexo IV de este trabajo.

1. Creación de proyecto en Laravel

El primer paso es crear un Proyecto ConvalidaciónCPPP en Laravel con el nombre *cppp*. Es importante crear el proyecto en la carpeta raíz del servidor web de Laragon, para esto desde la terminar de Windows se debe ingresar a la carpeta raíz con el siguiente comando:

cd C: \laragon\www

Estando en la carpeta raíz del servidor web se debe ejecutar el siguiente comando para la crear el proyecto *cppp:*

composer create-project laravel/laravel cppp

En la Figura 1 se presenta la creación del proyecto *cppp*

```
C:\laragon\www>composer create-project laravel/laravel cppp
Creating a "laravel/laravel" project at "./cppp"
Info from https://repo.packagist.org: #StandWithUkraine
Installing laravel/laravel (v8.6.12)
  - Installing laravel/laravel (v8.6.12): Extracting archive
Created project in C:\laragon\www\cppp
> @php -r "file_exists('.env') || copy('.env.example', '.env');"
Loading composer repositories with package information
```

Figura 1. Creación del proyecto cppp en laravel

Después de haber creado el proyecto, se puede comprobar su funcionamiento usando el comando *php artisan serve* de Laravel. Para esto se debe ingresar al directorio del proyecto con el comando *cd cppp,* estando en la carpeta del proyecto se ejecuta el comando antes mencionado para levantar el servicio, esto se presenta en la Figura 2.

```
C:\laragon\www\cppp>php artisan serve
Starting Laravel development server: http://127.0.0.1:8000
[Sun Jul 24 02:07:15 2022] PHP 7.4.19 Development Server (http://127.0.0.1:8000) started
```

Figura 2. Levantamiento del servicio del proyecto cppp

Una vez levantado el servidor de desarrollo de Artisan, desde el navegador web con la dirección: *http://127.0.0.1:800*, se puede acceder a esta aplicación. El resultado se muestra en la Figura 3, donde se observar que el proyecto creado funciona correctamente.

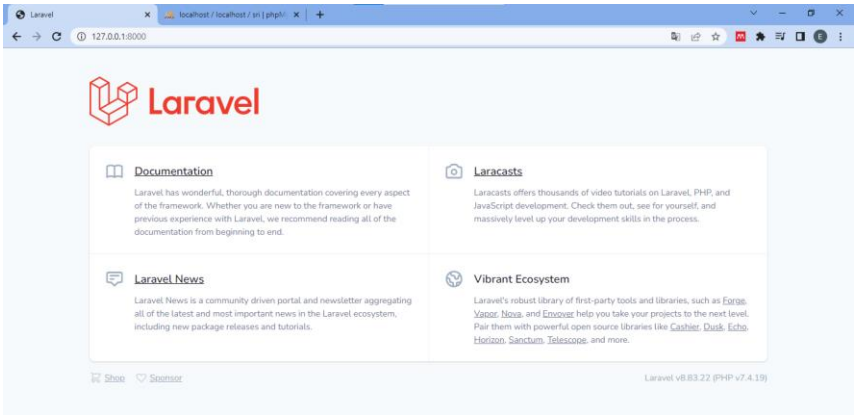

Figura 3. Visualización de proyecto cppp desde navegador web

2. Conexión con la Base de datos

Antes de configurar esta conexión desde la aplicación web, se debe crear una base de datos en MySQL. Para esto mediante un administrador de bases de datos como phpMyadmin se accede al motor de base de datos para crear una base llamada *cppp.* La creación de esta base de datos se presenta en la Figura 4.

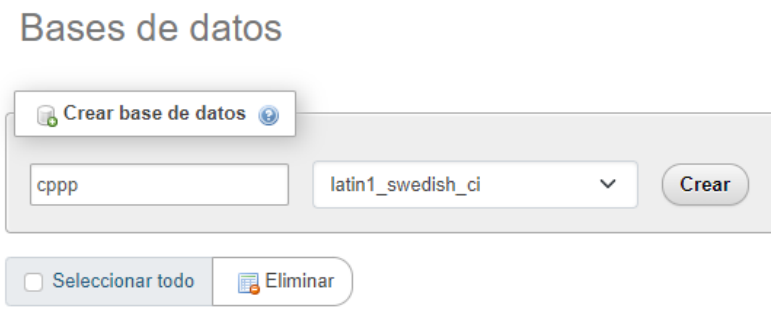

Figura 4. Creación de base de datos cppp desde phpMyadmin

Una vez creada la base de datos *cppp* en MySQL, se debe configurar la conexión con la misma desde la aplicación web. Esta conexión se la puede realizar de dos maneras, desde el archivo de configuración *config/database.php* o desde el archivo *.env,* para este

caso se usa la configuración mediante el archivo *.env.* La configuración realizada se presenta en el Código 1, es importante menciona que no se muestra todo el contenido del archivo, si no únicamente los campos que fueron configurados.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=cppp
DB_USERNAME=root
DB_PASSWORD=
```

Código 1. Conexión con base de datos cppp

3. Migraciones

Después de realizar la conexión con la base de datos, se deben crear las tablas que fueron diseñadas en la sección 2.2.2 de este proyecto. Laravel permite crear las tablas mediante migraciones, con esta funcionalidad se pueden crear tablas de forma dinámica sin la necesidad de usar código SQL. Para crear los archivos encargados de realizar la migración se usan los siguientes comandos:

php artisan make:migration create_estudiantes_table

php artisan make:migration create_profesors_table

php artisan make:migration create_formularios_table

php artisan make:migration create_informacionsoportes_table

php artisan make:migration create_informacionadicional_table

php artisan make:migration create_seccions_table

Estos archivos se encuentran en el directorio *database/migrations.* En estos se crean las funciones encargadas de generar o eliminar las tablas, estas corresponden a las funciones up y down respectivamente. A continuación, se presentan los códigos para cada uno de los archivos de migraciones que se han creado.

.

```php
public function up() {
    Schema::create('estudiantes', function (Blueprint $table) {
            $table->id();
            $table->string('carrera')->nullable();
            $table->string('nombres')->nullable();
            $table->string('apellidos')->nullable();
            $table->string('cedula')->unique();
            $table->string('correo')->unique();
            $table->string('telefono')->nullable();
            $table->string('celular')->nullable();
            $table->string('epn')->unique();
            $table->foreignIdFor(User::class)->nullable();
            $table->timestamps();
        });
    }
 public function down() {
        Schema::dropIfExists('estudiantes');
    }
```

Código 2. Código para crear o eliminar tabla estudiantes

```php
public function up() {
  Schema::create('profesors', function (Blueprint $table) {
            $table->id();
            $table->string('departamento')->nullable();
            $table->string('nombres')->nullable();
            $table->string('apellidos')->nullable();
            $table->string('cedula')->unique();
            $table->string('correo')->unique();
            $table->string('telefono')->nullable();
            $table->string('celular')->nullable();
            $table->string('epn')->unique();
            $table->foreignIdFor(User::class)->nullable();
            $table->timestamps();
        });
    }
public function down() {
        Schema::dropIfExists('profesors');
}
```

Código 3. Código para crear o eliminar tabla profesors

```php
public function up()
{
    Schema::create('formularios', function (Blueprint $table) {
        // Datos propios del formulario
        $table->id();
        $table->string('estado');
        //1.- ACTIVIDADES PARA LAS QUE SE SOLICITA CONVALIDACIÓN
        $table->enum('actividades', [
            'Cursos y Seminarios Profesionales',
            'Participación Estudiantil en Actividades Académicas, de Gestión,
            de Investigación y de Colaboración en Eventos Académicos **',
            'Representación Estudiantil',
            'Estudiantes mentores',
            'Representación de la Institución en competencias deportivas',
            'Actividades solidarias y de cooperación',
            'Experiencia Laboral',
            'Idiomas diferentes al Inglés y Lengua Materna',
            'Dirección de ramas de organizaciones estudiantiles académicas',
            'Representación de la Institución en competencias académicas ',
            'Coro y Grupo de Cámara',
            'Participación en la dirección de asociaciones de estudiantes',
            'Participación en juntas receptoras del voto'
            ])->nullable();
        //2.- DATOS DEL ESTUDIANTE
        $table->unsignedBigInteger('estudiante_id');
        $table->foreign('estudiante_id')
            ->references('id')
            ->on('estudiantes')
            ->onDelete('cascade');
        $table->date('fecha_asignacion_tutor')->nullable();
        //3.- DOCUMENTACIÓN DE SOPORTE ADJUNTO
            // la documentación no se almacena en esta Base de datos
        //4.- INFORMACIÓN DE LAS ACTIVIDADES DONDE SE REALIZO LAS PRÁCTICAS
        $table->string('tipo_institucion')->nullable();  //saber si es nacional o internacional
        $table->string('razon_social_institucion')->nullable();
        $table->string('ruc_institucion')->nullable();
        $table->string('direccion_institucion')->nullable();
        $table->string('telefono_institucion')->nullable();
        $table->string('celular_institucion')->nullable();
        $table->string('ciudad_pais_institucion')->nullable();
        $table->string('correo_institucion')->nullable();
        $table->string('tipo_institucion2')->nullable();
        $table->string('campo_amplio_institucion')->nullable();
        $table->string('campo_especifico_institucion')->nullable();
        $table->string('codigo_proyecto_convenio')->nullable();
        $table->string('nombre_proyecto_convenio')->nullable();
        //5.- INFORMACIÓN DE LAS ACTIVIDADES REALIZADAS
        $table->string('resumen_actividades')->nullable();
        $table->string('actividades_realizadas')->nullable();
        $table->string('aprendizaje_perfil')->nullable();
        $table->string('malla_curricular')->nullable();
        //6.- INFORMACIÓN ADICIONAL
        $table->date('fecha_inicio_actividades')->nullable();
        $table->date('fecha_fin_actividades')->nullable();
        $table->integer('horas_solicitadas')->nullable();
       //7.- DECLARACIÓN
        $table->date('fecha_declaracion')->nullable();
        $table->string('firma_declaracion')->nullable();
```

```
            //8.- INFORME DEL TUTOR
            $table->string('devolucion_subdecano')->nullable();
            $table->unsignedBigInteger('profesors_id')->nullable();
            $table->foreign('profesors_id')
                ->references('id')
                ->on('profesors')
                ->onDelete('cascade');
            $table->string('inf_tutor_Q1')->nullable();
            $table->string('inf_tutor_Q2')->nullable();
            $table->string('inf_tutor_Q3')->nullable();
            $table->string('recomendaciones_tutor')->nullable();
            $table->string('horas_sugeridas')->nullable();
            $table->date('fecha_recepcion_tutor')->nullable();
            $table->date('fecha_revision_tutor')->nullable();
            $table->string('firma_tutor')->nullable();
            $table->string('devolucion_tutor')->nullable();
            //9.- MIEMBRO DE LA CPP
            $table->date('fecha_recepcion_miembrocpp')->nullable();
            $table->date('fecha_revision_miembrocpp')->nullable();
            $table->string('recomendacion_miembro_cpp')->nullable();
            $table->string('devolucion_miembro_cpp')->nullable();
            //9.- COMISIÓN DE PRÁCTICAS PREPROFESIONALES
            $table->integer('horas_convalidades')->nullable();
            $table->integer('horas_convalidades_practicas')->nullable();
            $table->integer('horas_convalidades_comunitario')->nullable();
            $table->string('observaciones_cpp')->nullable();
            $table->date('fecha_recepcion_cpp')->nullable();
            $table->date('fecha_revision_cpp')->nullable();
            $table->string('firma_cpp')->nullable();
            $table->string('devolucion_cpp')->nullable();
            //10.- DECANO
            $table->date('fecha_recepcion_decano')->nullable();
            $table->date('fecha_autorizacion_decano')->nullable();
            $table->string('firma_decano')->nullable();

            // COMENTARIO DE RECHAZO
            $table->string('comentario_rechazo')->nullable();
            //INFORMACIÓN ADICONAL QUE ES INGRESADA POR COMPONENTE 2
            //DE ESTE PROYECTO

            //TIME STAMPS
            $table->timestamps();

        });
    }


    public function down()
    {
        Schema::dropIfExists('formularios');
    }
}
```

Código 4. Código para crear o eliminar tabla formularios

```php
public function up() {
    Schema::create('informacionsoportes', function (Blueprint $table) {
            $table->id();
            $table->string('url_archivo')->nullable();
            $table->unsignedBigInteger('formulario_id')->nullable();
            $table->foreign('formulario_id')
                ->references('id')
                ->on('formularios')
                ->onDelete('cascade');
            $table->timestamps();
        });
    }

  public function down() {
        Schema::dropIfExists('informacionsoportes');
    }
```

Código 5. Código para crear o eliminar tabla informacionsoportes

```php
public function up() {
  Schema::create('informacionadicionals', function (Blueprint $table) {
            $table->id();
            $table->string('url_archivo')->nullable();
            $table->unsignedBigInteger('formulario_id')->nullable();
            $table->foreign('formulario_id')
                ->references('id')
                ->on('formularios')
                ->onDelete('cascade');
            $table->timestamps();
        });
    }

public function down() {
        Schema::dropIfExists('informacionadicionals');
}
```

Código 6. Código para crear o eliminar tabla informacionadicionals

```php
public function up() {
    Schema::create('seccions', function (Blueprint $table) {

        $table->id();
        $table->string('seccion1')->nullable();
        $table->string('seccion3')->nullable();
        $table->string('seccion4')->nullable();
        $table->string('seccion5')->nullable();
        $table->string('seccion6')->nullable();
        $table->unsignedBigInteger('formulario_id')->nullable();
        $table->foreign('formulario_id')
            ->references('id')
            ->on('formularios')
            ->onDelete('cascade');
        $table->timestamps();
    });
}

public function down() {
    Schema::dropIfExists('seccions');
}
```

Código 7. Código para crear o eliminar tabla seccions

Una vez creadas las funciones que permiten generar y eliminar las migraciones de las tablas en la base de datos, se deben ejecutar esta migración con el comando *php artisan migrate,* de esta manera se crean las tablas en la base de datos. En la Figura 5 se presenta la ejecución del comando mencionado y el resultado obtenido.

```
C:\laragon\www\cppp>php artisan migrate
Migrating: 2014_10_12_000000_create_users_table
Migrated:  2014_10_12_000000_create_users_table (74.12ms)
Migrating: 2014_10_12_100000_create_password_resets_table
Migrated:  2014_10_12_100000_create_password_resets_table (111.28ms)
Migrating: 2019_08_19_000000_create_failed_jobs_table
Migrated:  2019_08_19_000000_create_failed_jobs_table (67.60ms)
Migrating: 2019_12_14_000001_create_personal_access_tokens_table
Migrated:  2019_12_14_000001_create_personal_access_tokens_table (95.90ms)
Migrating: 2022_08_02_004618_create_estudiantes_table
Migrated:  2022_08_02_004618_create_estudiantes_table (128.89ms)
Migrating: 2022_08_02_004707_create_profesors_table
Migrated:  2022_08_02_004707_create_profesors_table (127.27ms)
Migrating: 2022_08_02_004942_create_formularios_table
Migrated:  2022_08_02_004942_create_formularios_table (202.35ms)
Migrating: 2022_08_02_005022_create_informacionsoportes_table
Migrated:  2022_08_02_005022_create_informacionsoportes_table (95.89ms)
Migrating: 2022_08_02_005052_create_informacionadicionals_table
Migrated:  2022_08_02_005052_create_informacionadicionals_table (128.64ms)
Migrating: 2022_08_02_005137_create_seccions_table
Migrated:  2022_08_02_005137_create_seccions_table (125.08ms)
```

Figura 5. Ejecución de migraciones

4. Sistema de login

La autenticación es el proceso de registro e inicio de sesión de los usuarios. Laravel permite crear un sistema de autenticación de una forma sencilla mediante los siguientes comandos:

composer require laravel/ui

php artisan ui boostrap –auth

En la Figura 6 se puede ver que el sistema de autenticación se ha creado de manera exitosa, pero en la Figura 7 se puede ver que el sistema de autenticación no tiene un estilo que usa css o jss.

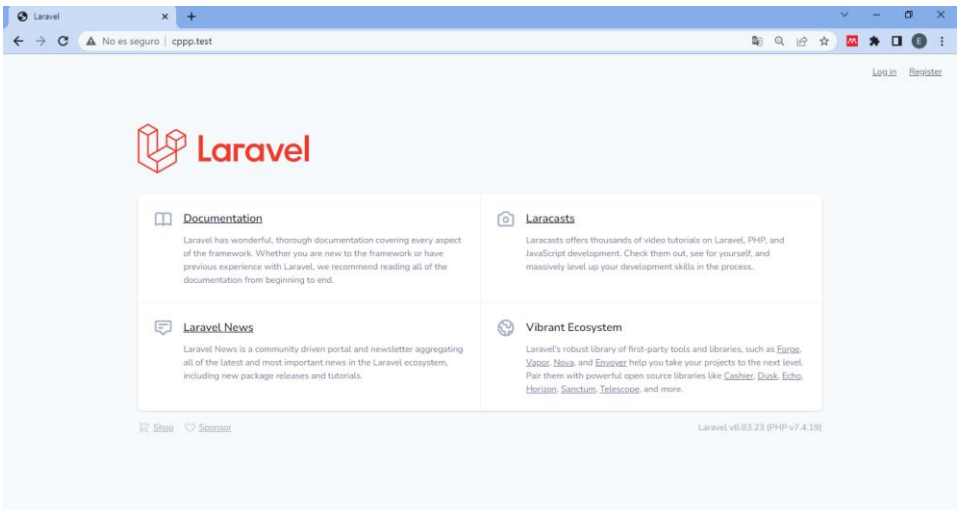

Figura 6. Sistema de autenticación creado exitosamente

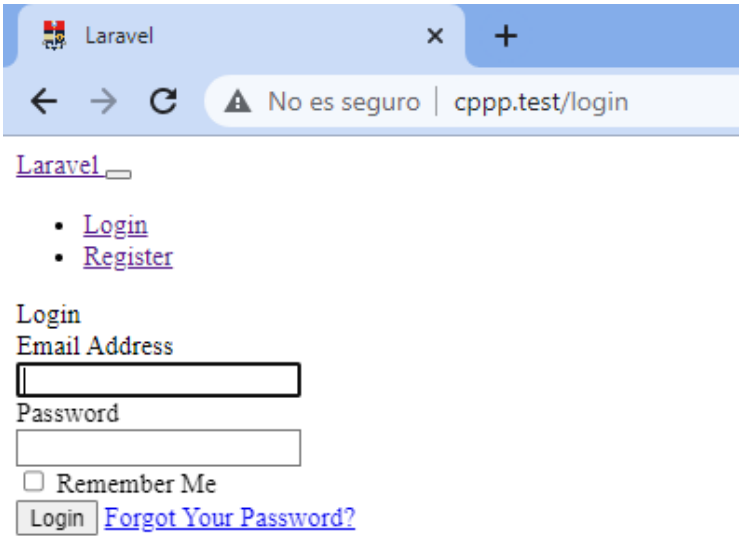

Figura 7. Sistema de autenticación sin css o jss

Luego se requiere ejecutar los comandos *npm install y npm run dev* los cuales generan los archivos CSS y JSS para el sistema de autenticación, pero para ello primero se debe realizar la instalación de node.js como se muestra en la Figura 8.

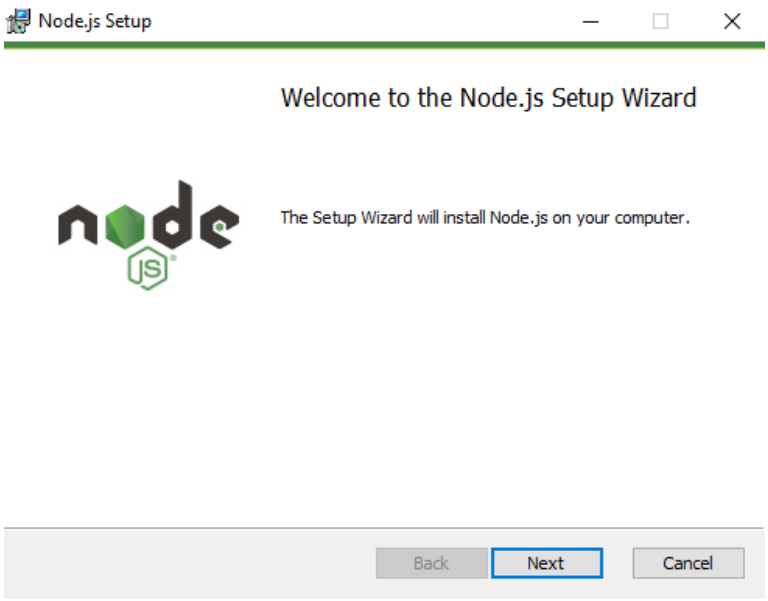

Figura 8. Instalación de node.js

Una vez instalado node.js el sistema de autenticación de tiene una mejor apariencia como se muestra en la Figura 9.

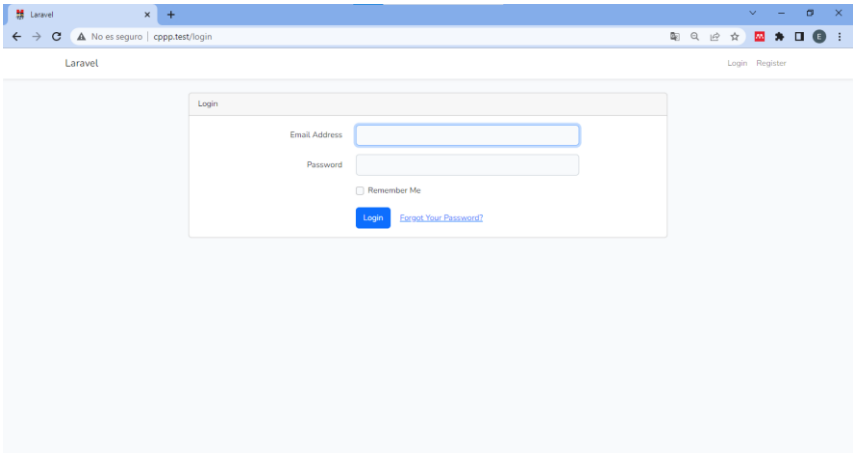

Figura 9. Sistema de autenticación de Laravel

5. Instalación de plantilla adminLTE

Laravel ofrece algunas funcionalidades de frontend bastante útiles en el desarrollo de aplicaciones web. Para este caso en el proyecto se incorpora una plantilla de Blade llamada adminLTE, el proceso para la instalación de esta plantilla en la aplicación ConvalidaciónCPPP se presenta a continuación:

En la carpeta raíz del proyecto Laravel, se instala el paquete usando el siguiente comando de composer

composer require jeroennoten/laravel-adminlte

Luego se deben instalar los recursos del paquete necesarios con el siguiente comando:

php artisan adminlte:install

Con estos sencillos la se habrá instalado la plantilla de adminLTE en el proyecto de laravel, para hacer uso de esta se utilizan las directivas @extends y @section como se muestra en el código 8.

```
@extends('adminlte::page')

@section('title', 'Dashboard')

@section('content_header')
    <h1>Dashboard</h1>
@stop

@section('content')
    <p>Welcome to this beautiful admin panel.</p>
@stop

@section('css')
    <link rel="stylesheet" href="/css/admin_custom.css">
@stop

@section('js')
    <script> console.log('Hi!'); </script>
@stop
```

Código 8. Uso de la plantilla adminLTE

El resultado obtenido se presenta en la Figura 10.

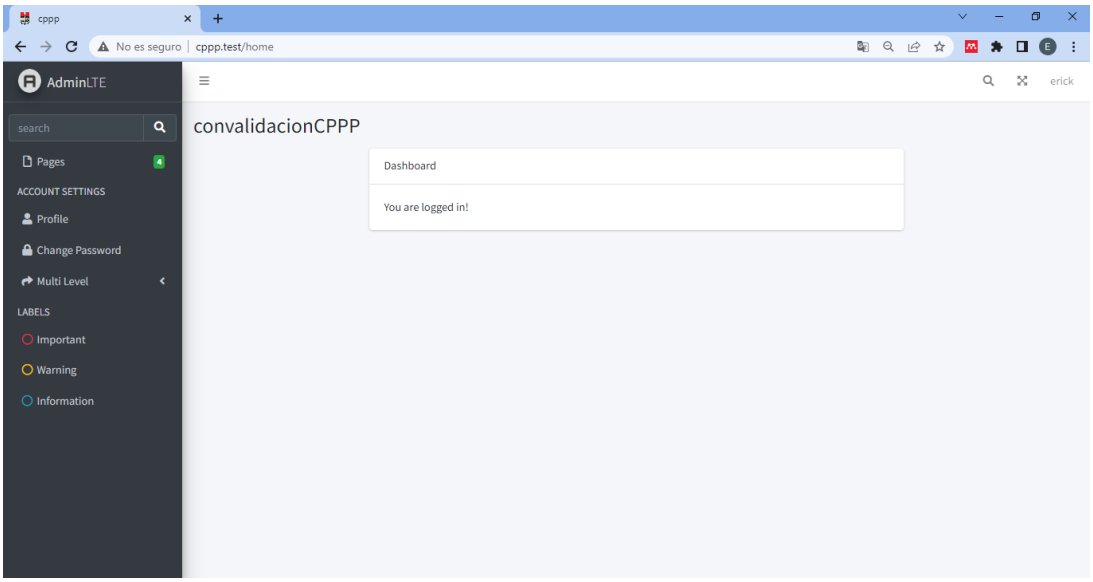

Figura 10. Uso de la plantilla adminLTE en el proyecto cppp

6. Creación de rutas

En el archivo *web.php* del directorio /routes se describen todas las rutas asociadas con la interfaz web. En el código 9 se presentan todas las rutas creadas para la aplicación web ConvalidaciónCPPP.

```
//Para consultar API de Estudiantes
Route::post('/estudiantesEPN', [EstudiantesController::class, 'indexByEPN'])-
>name('estudiantesEPN');
//Para consultar API de Profesores
Route::post('/profesoresEPN', [ProfesorController::class, 'indexByEPN'])-
>name('profesoresEPN');

// Metodos de ESTUDIANTES
Route::post('/estudiantes', [EstudiantesController::class, 'store'])-
>name('estudiantes.insert');
//Metodos de PROFESORES
Route::post('/profesores', [ProfesorController::class, 'store'])-
>name('profesores.insert');

// //Metodos de Formulario
Route::get('formularios/nuevo-formulario',
[App\Http\Controllers\FormularioController::class, 'nuevoFormulario'])-
>name('nuevo-formulario');
Route::get('/formulario', [FormularioController::class, 'index'])-
>name('formulario');
Route::post('/formulario', [FormularioController::class, 'store'])-
>name('formulario.insert');
Route::put('/formulario-tutor', [FormularioController::class, 'storeSubdecano'])-
>name('formulario.insert.tutor');
Route::put('/formulario-llenar-tutor', [FormularioController::class,
'storeTutor'])->name('formulario.insert.datos.tutor');
Route::put('/formulario-miembrocomision', [FormularioController::class,
'storeMiembrocomision'])->name('formulario.insert.miembrocomision');
Route::put('/formulario-comision', [FormularioController::class,
'storeComision'])->name('formulario.insert.comision');
Route::put('/formulario-decano', [FormularioController::class, 'storeDecano'])-
>name('formulario.insert.decano');
Route::put('/formulario/rechazar', [FormularioController::class, 'destroy'])-
>name('formulario.destroy');
Route::get('/aceptar-formulario/{id}', [FormularioController::class,
'aceptarFormulario'])->name('formulario.aceptar');
Route::get('/aceptar-formulario-tutor/{id}', [FormularioController::class,
'aceptarFormularioTutor'])->name('formulario.aceptar.tutor');
Route::get('/aceptar-formulario-miembrocpp/{id}', [FormularioController::class,
'aceptarFormularioMiembrocpp'])->name('formulario.aceptar.miembrocpp');
```

```
Route::get('/aceptar-formulario-decano/{id}', [FormularioController::class,
'aceptarFormularioDecano'])->name('formulario.aceptar.decano');
// // Ruta para revisar los formularios en el formato original
Route::put('/revisar-formulario/{id}', [FormularioController::class,
'revisarFormulario'])->name('formulario.revisar');

// Metodo para solicitar correciones
Route::put('/formulario-corregir', [FormularioController::class,
'storeCorregir'])->name('formulario.corregir');// para comentarios de correcion
Route::get('/corregir-formulario/{id}', [FormularioController::class,
'aceptarCorregir'])->name('formulario.aceptarcorreccion'); // para aceptar la
correcion
Route::put('/formulario-corregir-estudiante', [FormularioController::class,
'updateFormulario'])->name('formulario.updateformulario'); // para setear nuevas
correcciones
// // Para formularios rechazados
Route::get('/formulario-rechazado', [FormularioController::class,
'indexRechazado'])->name('formulario.rechazado');
// // Para ver los formularios que ya tiene un tutor asignado
Route::get('/formulario-aceptado', [FormularioController::class,
'indexAceptado'])->name('formulario.aceptado');
// Formularios terminados Estudiante
Route::get('/formulario-completados', [FormularioController::class,
'indexCompletados'])->name('formulario.completados');
// Formularios devueltos para revision
Route::get('/formulario-devueltos', [FormularioController::class,
'indexDevueltos'])->name('formulario.devueltos');
// Formularios formularios corregidos
Route::get('/formulario-corregidos', [FormularioController::class,
'indexCorregidos'])->name('formulario.corregidos');
// //Generar PDF
Route::get('/generate-pdf/{id}', [PDFController::class, 'generatePDF'])-
>name('generar.pdf');
```

Código 9. Rutas para atender consultas desde la Api

7. Frontend de acuerdo con el rol de usuario

Una vez instalada la plantilla adminLTE, esta puede ser modificada de acuerdo al diseño realizado en la sección 2.2.3. Para cada rol de usuarios se requieren módulos para que cumplan con las funcionalidades establecidas para su rol y adminLTE permite incorporar a su plantilla módulos de una manera sencilla, estos deben ser incluidos en el archivo *adminlte.php* el cual se encuentra en el directorio */config*. Dentro del archivo *adminlte.php* se debe identificar la sección “Menu Items” en el cual se escriben las líneas de código presentada en el Código 10.

```
    'menu' => [
        // Navbar items:
        [
            'type'         => 'fullscreen-widget',
            'topnav_right' => true,
        ],

        // Sidebar items:
        [
            'text'        => 'Inicio',
            'url'         => 'home',
            'icon'        => 'fa fa-home',
            'icon_color' => 'cyan',
        ],
        [
            'header' => 'ADMINISTRADOR',
            'can' => 'usuarios',
        ],
        [
            'text' => 'Usuarios',
            'route'  => 'usuarios.index',
            'icon' => 'fas fa-fw fa-users',
            'icon_color' => 'cyan',
            'can' => 'usuarios',
        ],
        [
            'text' => 'Nuevo Usuario',
            'route'  => 'usuarios.create',
            'icon' => 'fas fa-fw fa-user-plus',
            'icon_color' => 'cyan',
            'can' => 'usuarios',
        ],
        ['header' => 'FORMULARIOS',
         'can' => 'Ver Nuevos'],
        [
            'text' => 'Nuevo Formulario',
            'route'  => 'nuevo-formulario',
            'icon' => 'fas fa-fw fa-address-card',
            'icon_color' => 'green',
            'can' => 'Nuevo Formulario',
        ],
        [
            'text' => 'Ver Formularios',
            'icon' => 'fas fa-fw fa-eye',
            'icon_color' => 'green',
            'can' => 'Ver Nuevos',
            'submenu' => [
```

```
                [
                    'text' => 'Nuevos',
                    'route'  => 'formulario',
                    'icon' => 'fas fa-fw fa-address-book',
                    'icon_color' => 'blue',
                    'can' => 'Ver Nuevos',
                ],
                [
                    'text' => 'Revisados',
                    'route'  => 'formulario.aceptado',
                    'icon' => 'fas fa-fw fa-address-book',
                    'icon_color' => 'blue',
                    'can' => 'Ver Revisados',
                ],
                [
                    'text' => 'Por Corregir',
                    'route'  => 'formulario.devueltos',
                    'icon' => 'fas fa-fw fa-address-book',
                    'icon_color' => 'blue',
                    'can' => 'Ver Devueltos',
                ],
                [
                    'text' => 'Corregidos',
                    'route'  => 'formulario.corregidos',
                    'icon' => 'fas fa-fw fa-address-book',
                    'icon_color' => 'blue',
                    'can' => 'Ver Corregidos',
                ],
                [
                    'text' => 'Aceptados',
                    'route'  => 'formulario.completados',
                    'icon' => 'fas fa-fw fa-address-book',
                    'icon_color' => 'blue',
                    'can' => 'Ver Aceptados',
                ],
                [
                    'text' => 'Rechazados',
                    'route'  => 'formulario.rechazado',
                    'icon' => 'fas fa-fw fa-address-book',
                    'icon_color' => 'blue',
                    'can' => 'Ver Rechazados',
                ],
            ]
        ],
    ],
```

Código 10. Creación de botones para funcionalidades de cada rol

En el código 10 se presentó el código para la creación de los módulos necesarios para cumplir con las funcionalidades de cada rol establecidos para este proyecto. En esta sección es importante mencionar sobre la asignación de permisos para cada rol de usuarios, como se puede observar en el código 10 para cada módulo existe una directiva llamada *"can"*, esta es la encargada de determinar los permisos para cada rol de usuario. A continuación, se presentan las configuraciones realizadas para la generación de estos permisos.

Primero se debe instalar la librería que permite crear roles y permisos y asociar estos a roles o usuarios de la aplicación, además permite usar la directiva de Blade *"can"* antes mencionada. Para instalar esta librería se usa el siguiente comando:

composer require spatie/laravel-permission

Como segundo paso se debe implementar la librería en el modelo user como se muestra en el código 11.

```
class User extends Authenticatable
{
    use HasRoles;
    // ...
}
```

Código 11. Implementación de librería de permisos en modelo user

El siguiente es agregar el proveedor de servicios en el archivo *config/app.cong* como se muestra en el código 12.

```
'providers' => [
   // ...
Spatie\Permission\PermissionServiceProvider::class,];
```

Código 12. Agregar el proveedor de servicios de spati

El siguiente paso es publicar las migraciones generadas por spati y el archivo de configuración */config/permission.php* mediante el siguiente comando:

php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"

Como último paso se debe generar un seeder en el cual se crean los roles y los permisos, el seeder se crea con el siguiente comando:

php artisan make:model RoleSeeder

Dentro de este archive se crean los roles y permisos como se muestra en el código 13.

```
    public function run()
    {
        // Administrador Técnico:
        $role1 = Role::create(['name' => 'Administrador']);
        // -------------------------------------------------
        $role2 = Role::create(['name' => 'Subdecano']);
        $role3 = Role::create(['name' => 'Decano']);
        $role4 = Role::create(['name' => 'Coordinador CPPP']);
        $role5 = Role::create(['name' => 'Tutor']);
        $role6 = Role::create(['name' => 'Miembro CPPP']);
        $role7 = Role::create(['name' => 'Secretaria de Coordinación']);
        $role8 = Role::create(['name' => 'Usuario General']);
        $role9 = Role::create(['name' => 'Estudiante']);

        // Vista general:
        Permission::create(['name' => 'home'])->syncRoles([$role1, $role2,
       $role3, $role4, $role5, $role6, $role7, $role8, $role9]);

        // Administrador:
        Permission::create(['name' => 'usuarios'])->syncRoles([$role1]);
        Permission::create(['name' => 'usuarios.edit'])->syncRoles([$role1]);
        Permission::create(['name' => 'usuarios.index'])->syncRoles([$role1]);
        Permission::create(['name' => 'usuarios.create'])->syncRoles([$role1]);

        // Usuarios:
        Permission::create(['name' => 'registros'])->syncRoles([$role1, $role4,
       $role6, $role7]);
        Permission::create(['name' => 'registros.index'])->syncRoles([$role1,
       $role4, $role6, $role7]);
        Permission::create(['name' => 'registros.edit'])->syncRoles([$role1,
       $role6, $role4, $role7]);

        // Permisos unicos:
        Permission::create(['name' => 'Fechas_Detalle'])->syncRoles([$role1]);
        Permission::create(['name' => 'Fecha_Ingreso'])->syncRoles([$role6]);
        Permission::create(['name' => 'Fecha_Certificado'])-
       >syncRoles([$role4]);
        Permission::create(['name' => 'Fecha_Registro'])->syncRoles([$role7]);
```

```
        // Permisos Estudiantes:
        Permission::create(['name' => 'Ver Nuevos'])-
    >syncRoles([$role9,$role2,$role3, $role4, $role5, $role6]);
        Permission::create(['name' => 'Ver Revisados'])->syncRoles([$role2,
    $role3, $role4, $role5, $role6]);
        Permission::create(['name' => 'Ver Rechazados'])->syncRoles([$role9,
    $role2, $role4, $role6]);
        Permission::create(['name' => 'Nuevo Formulario'])->syncRoles([$role9]);
        Permission::create(['name' => 'Ver Devueltos'])->syncRoles([$role9,
    $role2, $role4, $role5, $role6]);
        Permission::create(['name' => 'Ver Corregidos'])->syncRoles([$role9,
    $role2, $role4, $role5, $role6]);
        Permission::create(['name' => 'Ver Aceptados'])->syncRoles([$role9,
    $role4]);
    }
```

Código 13. Creación de roles y permisos

Una vez determinado los permisos para cada rol, el resultados que se obtiene se presentan en las siguientes Figuras de la 11 a la 17. Donde se puede observar que cada rol de usuario tiene únicamente acceso a los módulos necesarios para cumplir con sus funcionalidades.

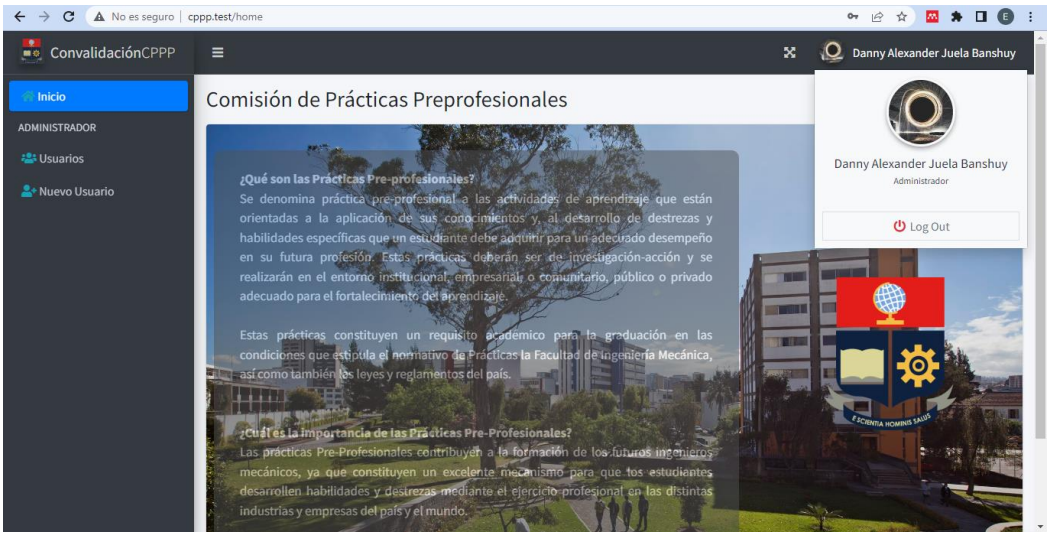

Figura 11. Interfaz del rol Administrador

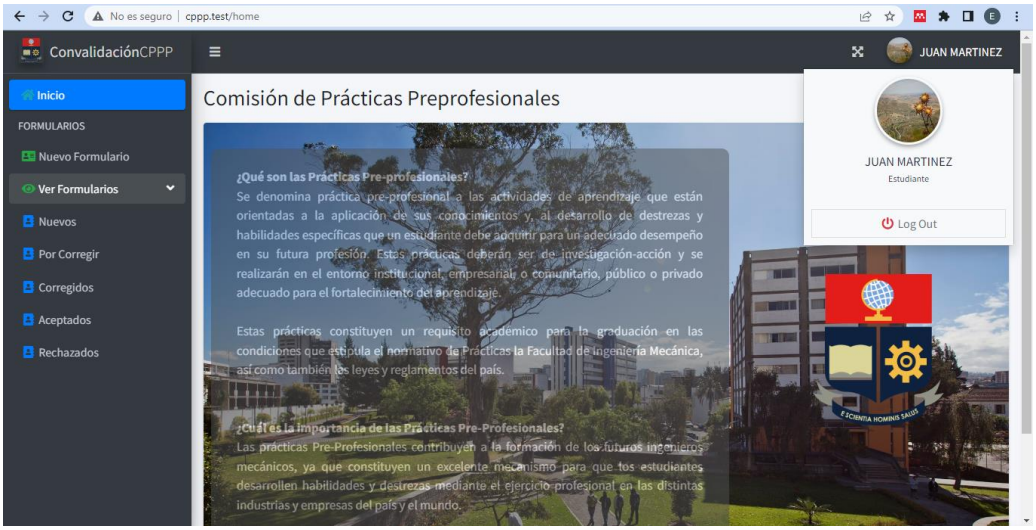

Figura 12. Interfaz del rol Estudiante

Figura 13. Interfaz del rol Subdecano

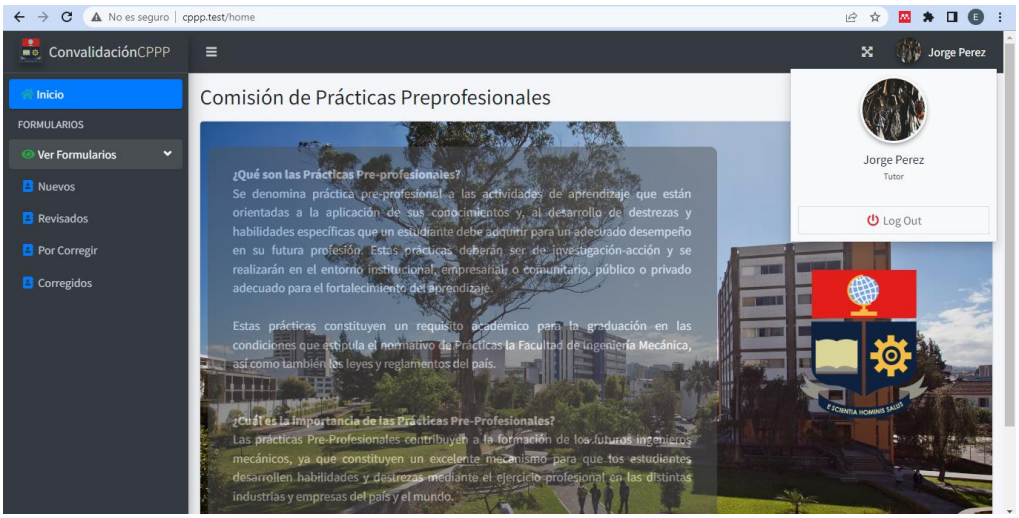

Figura 14. Interfaz del rol Tutor

Figura 15. Interfaz del rol Miembro de la CPP

Figura 16. Interfaz del rol Coordinador de la CPP

Figura 17. Interfaz del rol Decano

Una vez creados los módulos se deben implementar las vistas asociadas a cada uno de estos, a continuación, se especifican cuáles son estas vistas. En esta sección no se presentan los códigos de cada una de las vistas debido a su gran extensión, pero estas pueden ser revisadas en el Anexo IV.

a) *users.create.blade.php*

Esta vista muestras las opciones para crear un nuevo usuario, esta acción solo la puede realizar el usuario administrador.

b) *estudiante.nuevo-formulario.blade.php*

Esta vista permite la creación de un nuevo formulario, esta acción solo la puede realizar el estudiante.

c) *estudiante.insert.blade.php*

Esta vista permite la creación de un nuevo usuario con rol estudiante, esta acción solo la puede realizar Administrador de la aplicación.

d) *profesor.insert.blade.php*

Esta vista permite la creación de un nuevo usuario con rol tutor, esta acción solo la puede realizar Administrador de la aplicación.

e) *formulario.index.blade.php*

Esta vista permite listar los formularios, ya sean nuevos, aceptados, rechazados, corregidos y por corregir.

f) *formulario. aceptar-formulario.blade.php*

Esta vista permite revisar los formularios por parte del subdecano y se presentan las opciones de asignar tutor, solicitar correcciones y rechazar formulario.

g) *formulario. aceptar-formulario-tutor.blade.php*

Esta vista permite revisar los formularios por parte del tutor y se presentan también las secciones que el tutor debe completar y adicional una opción para solicitar correcciones.

h) *formulario. aceptar-formulario-miembrocomision.blade.php*

Esta vista permite revisar los formularios por parte del miembro de la comision y se presentan también una sección para que el miembro de la comisión de su

recomendación, adicional tiene la opción para solicitar correcciones o rechazar el formulario.

i) *formulario. aceptar-formulario-comision.blade.php*

Esta vista permite revisar los formularios por parte del coordinador de la comisión y se presentan también las secciones que el coordinador debe completar, adicional tiene la opción de solicitar correcciones o rechazar el formulario.

j) *formulario. aceptar-formulario-decano.blade.php*

Esta vista permite revisar los formularios por parte del decano y se presentan también la sección que el decano debe completar.

k) *formulario. aceptar-correcion-formulario.blade.php*

Esta vista permite revisar los formularios devueltos y que deben ser corregidos.

l) *formulario. revisar-formulario.blade.php*

Esta vista permite revisar los formularios que han sido enviados en modo de resumen.

m) *formulario. myPDF.blade.php*

Esta vista permite la generación del formulario en su formato original e imprimirlo y guardarlo en formato PDF.

8. Backend

En esta sección básicamente se explicarán las funciones que han sido implementadas, todas estas se encuentran en diferentes controladores creados en este proyecto. Debido a la gran extensión de código implementado en los controladores, estos no serán presentados en esta sección, pero están disponibles en el Anexo E.

La aplicación ConvalidaciónCPPP hace uso de cuatro controladores para cumplir con el objetivo de este proyecto, estos controladores son los siguientes:

- EstudiantesController.php
- ProfesorController.php
- FormularioController.php
- PDFController.php

Estos se los crea con los siguientes comandos desde la terminal:

php artisan make:controller EstudiantesController

php artisan make:controller ProfesorController

php artisan make:controller FormularioController

php artisan make:controller PDFController

A continuación, en la tabla 1 se presentan las funciones implementadas en los controladores y una descripción de lo que estos realizan en la aplicación.

Tabla 1. Funciones de la aplicación ConvalidaciónCPPP

Controlador	Función	Descripción
EstudiantesController.php	indexByEPN	Esta función es la encargada de realizar consultas de estudiantes a la Api university-rest.
	store	Se encarga de agregar un nuevo registro en la tabla usuarios y estudiantes
ProfesorController.php	indexByEPN	Esta función es la encargada de realizar consultas de profesores a la Api university-rest.
	store	Se encarga de agregar un nuevo registro en la tabla usuarios y profesors
	nuevoFormulario	Esta se encarga de pasar datos que se llenan de forma automática a la vista nuevo-formulario
	Index	Se encarga de filtrar los formularios y enviar a la vista formulario.index.blade.php únicamente los nuevos formularios
	indexRechazado	Se encarga de filtrar los formularios y enviar a la vista formulario.index.blade.php únicamente los formularios rechazados

FormularioController.php	indexAceptado	Se encarga de filtrar los formularios y enviar a la vista formulario.index.blade.php únicamente los formularios que ya lleno cada actor
	indexCompletados	Se encarga de filtrar los formularios y enviar a la vista formulario.index.blade.php únicamente los formularios que terminaron con éxito el proceso
	indexDevueltos	Se encarga de filtrar los formularios y enviar a la vista formulario.index.blade.php únicamente los formularios que se deben corregir
	indexCorregidos	Se encarga de filtrar los formularios y enviar a la vista formulario.index.blade.php únicamente los formularios que ya fueron corregidos
	store	Se encarga de agregar el nuevo formulario a la base de datos
	storeSubdecano	Se encarga añadir los campos llenados por el subdecano en la base de datos
	storeTutor	Se encarga añadir los campos llenados por el tutor en la base de datos
	storeMiembrocomision	Se encarga añadir los campos llenados por el miembro de la comisión en la base de datos
	storeComision	Se encarga añadir los campos llenados por el coordinador de la cpp en la base de datos
		Se encarga añadir los campos

	storeDecano	llenados por el decano en la base de datos
	storeCorregir	Se encarga añadir el mensaje y las secciones que se deben corregir
	destroy	Se encarga rechazar un formulario
	aceptarFormulario	Se encarga de pasar al vista aceptar-formulario.blade.php la información llenada por el estudiante
	aceptarFormularioTutor	Se encarga de pasar al vista aceptar-formulario-tutor.blade.php la información llenada por los actores anteriores.
	aceptarFormularioMiembrocpp	Se encarga de pasar al vista aceptar-formulario-miembrocomision.blade.php la información llenada por los actores anteriores.
	aceptarFormularioComision	Se encarga de pasar al vista aceptar-formulario-comision.blade.php la información llenada por los actores anteriores.
	aceptarFormularioDecano	Se encarga de pasar al vista aceptar-formulario-decano.blade.php la información llenada por los actores anteriores.
	aceptarCorregir	Se encarga de pasar al vista aceptar-correcion-formulario.blade.php la información sobre las correcciones que debe hacer el estudiante
		Se encarga de actualizar las

	updateFormulario	correcciones hechas por el estudiante
	revisarFormulario	Se encarga de pasar al vista revisar-formulario.blade.php toda la información que ya ha sido llenada.
PDFController.php	generatePDF	Se encarga de pasar al vista myPDF.blade.php toda la información del formulario para convertirlo en pdf

ANEXO VI

En este Anexo se simulará el proceso de convalidación de prácticas preprofesionales, las actividades que se realizarán son: creación de usuarios, creación de nuevo formulario, asignación de tutor, revisión por parte del tutor, revisión por parte del miembro de la CPP, revisión por parte del coordinador de la CPP y decano. Las pruebas realizadas se presentan a continuación.

1. Levantar servicios de las API

La Api university-rest debe ser levanta en el puerto 8000 como se muestra en la Figura 1 de este Anexo.

Figura 1. Levantamiento de la api university-rest en el puerto 8000

Mientras que la Api sri debe ser levantada en el puerto 8080 como se muestra en la Figura 2 de este Anexo.

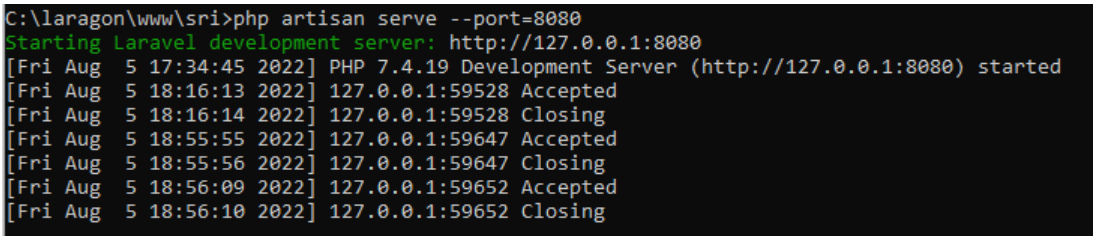

Figura 2. Levantamiento de la api sri en el puerto 8080

2. Creación de un nuevo usuario

El administrador es el encargo de realizar el CRUD de usuarios, por lo tanto, la creación de un nuevo usuarios esta entre sus funcionalidades. Como primer paso para realizar las pruebas de funcionamiento es la creación de un nuevo usuario y la aplicación permite realizarlo de 2 diferentes maneras, ya sea consultado a la Api university-rest o mediante el ingreso manual de los datos de un usuario, esto se puede observar en la Figura 3.

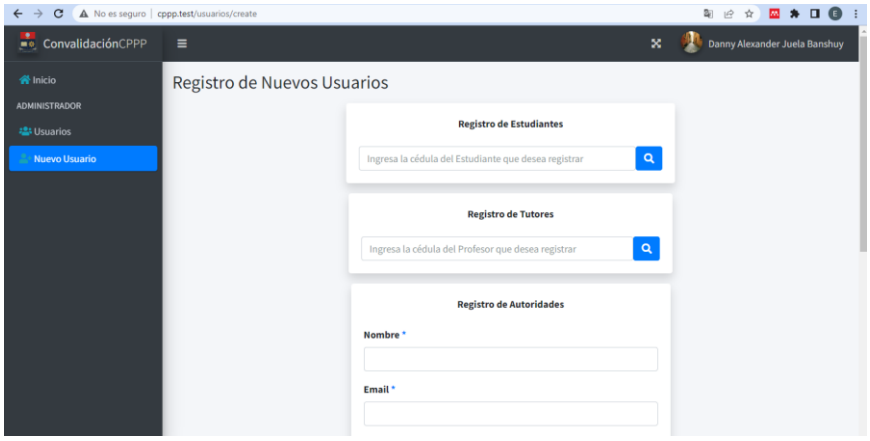

Figura 3. Opciones para crear un nuevo usuario

Para este caso se realiza la prueba creando un usuario mediante la consulta a la apu university-rest. Lo que se debe hacer es ingresar el número de cédula del estudinate y la app realizara una busqueda en la Api y traera los datos encontrados, la busqueda se muestra en la Figura 4.

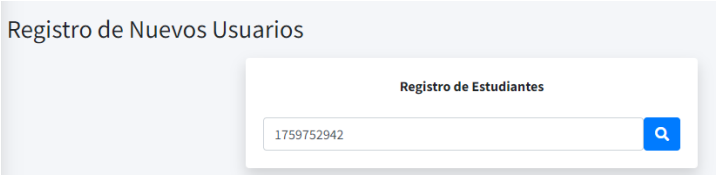

Figura 4. Consulta de un estudiante a la Api university-rest

Si el número de cédula si correpsonde a un estudinate en la pantalla se desplega la información para su registro, sin opciones de edición como se muestra en la Figura 5.

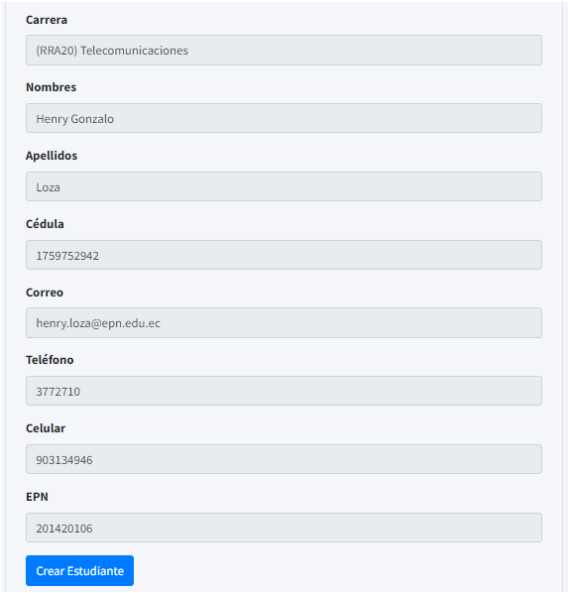

Figura 5. Datos obtenidos de la Api para el registro

Una vez creado el estudiante, el administrador puede observar que este ha sido creado exitosamente en la opción usuarios como se muestra en la Figura 6

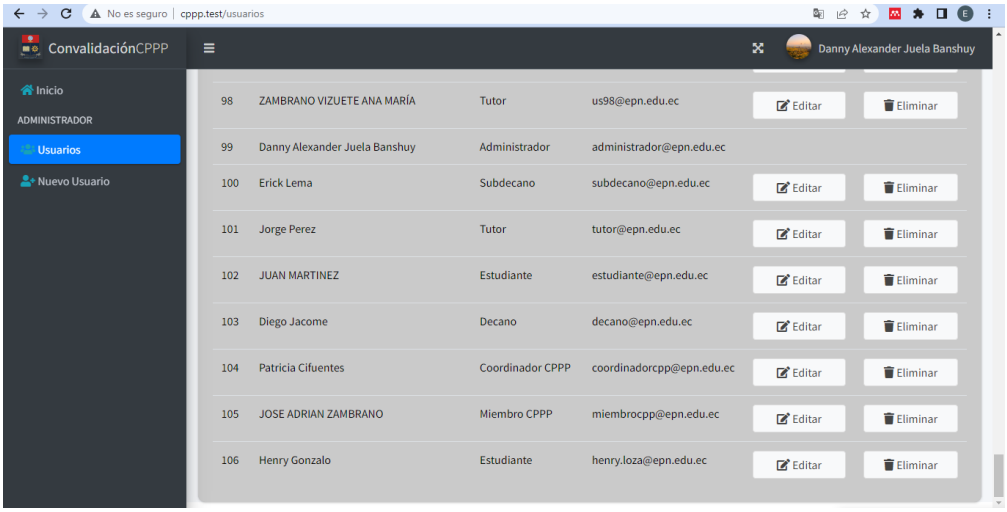

Figura 6. Creación exitosa de usuario

3. Creación de nuevo formulario

El estudiante es el único que puede crear un nuevo formulario, para esto primero ingresa a la aplicación con su correo como se muestra en la Figura 7.

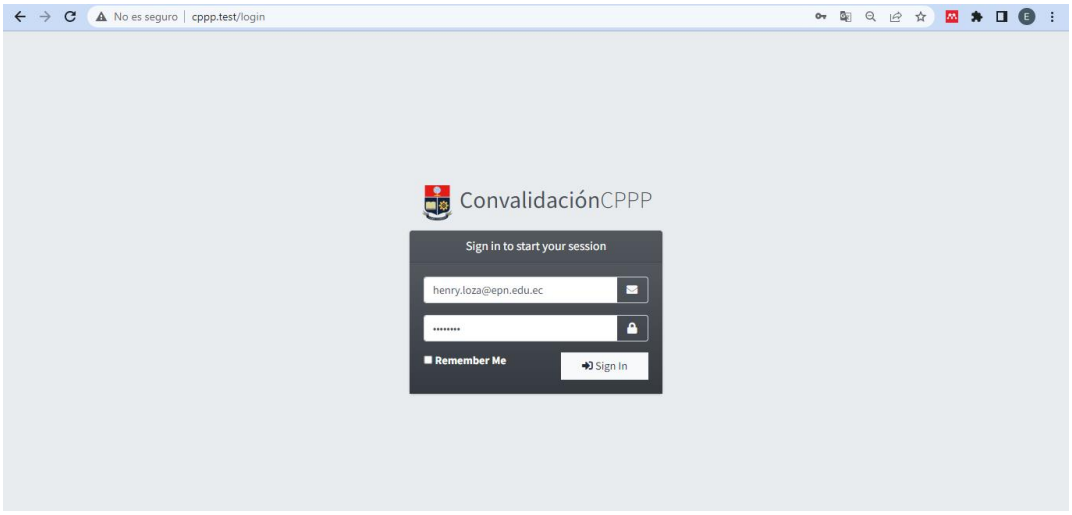

Figura 7. Login del estudiante

Una vez que el estuidnate ingresa a la aplicacación, el sistema reconce su rol de usurio de forma automatica. Para crear un nuevo formulario, se debe dar click en la opción “Nuevo Formulario”. El llenado del formulario se presenta desde la Figura 8 a la 14.

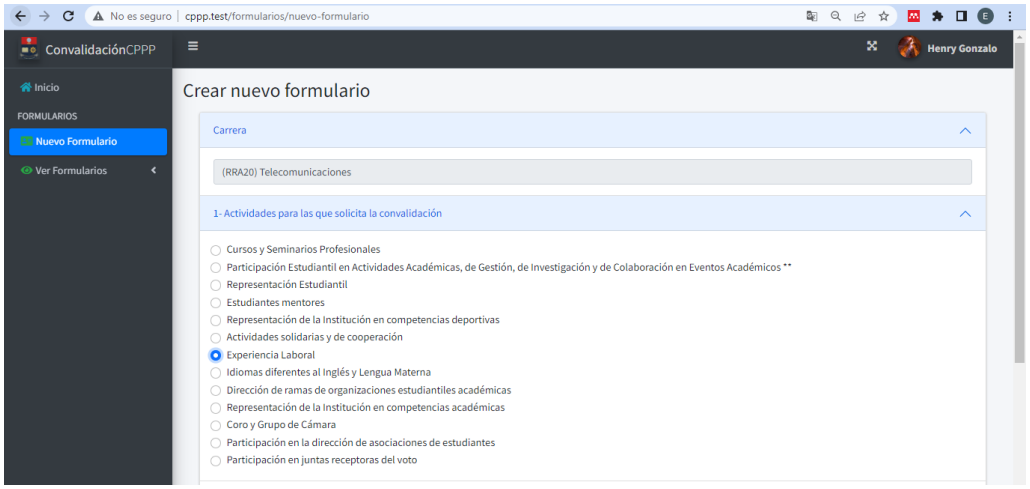

Figura 8. Llenado de formulario por el estudiante

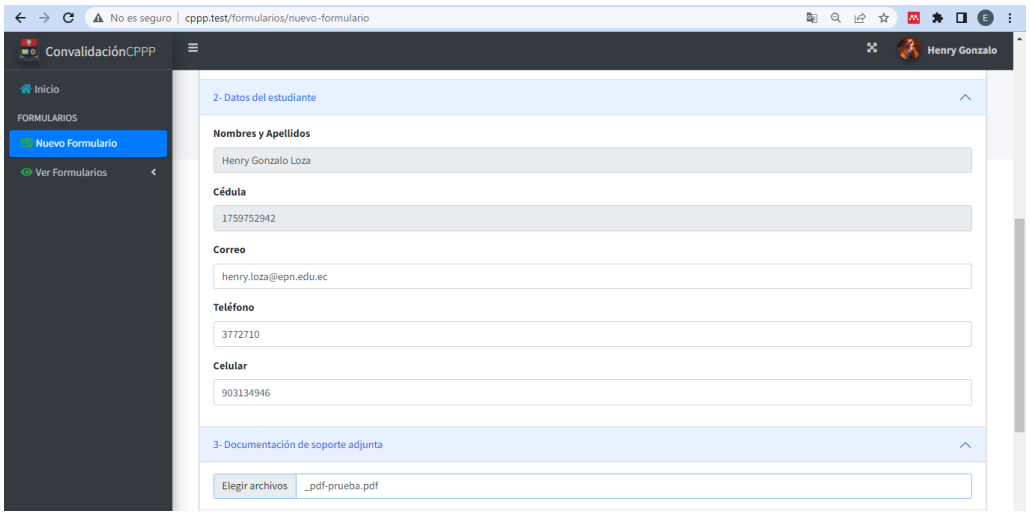

Figura 9. Llenado de formulario por el estudiante

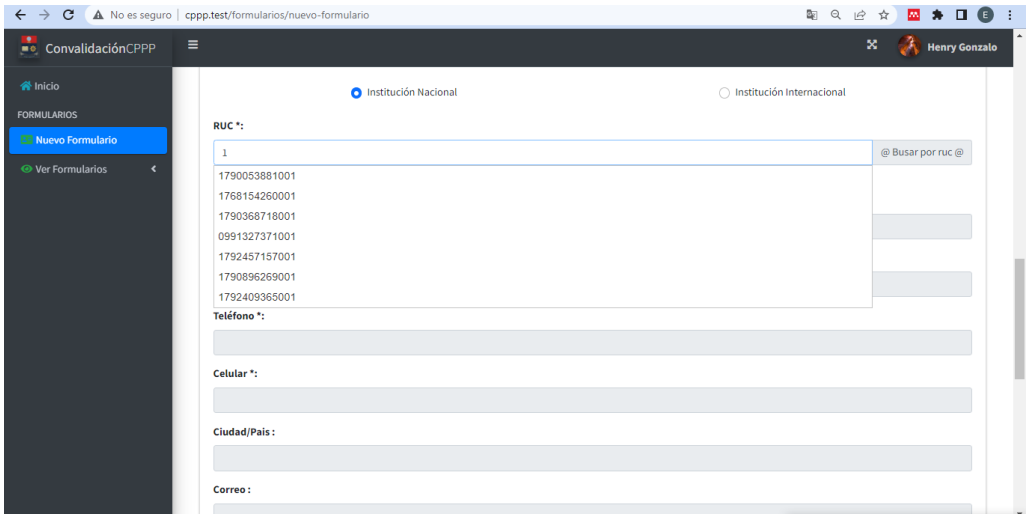

Figura 10. Llenado de formulario por el estudiante

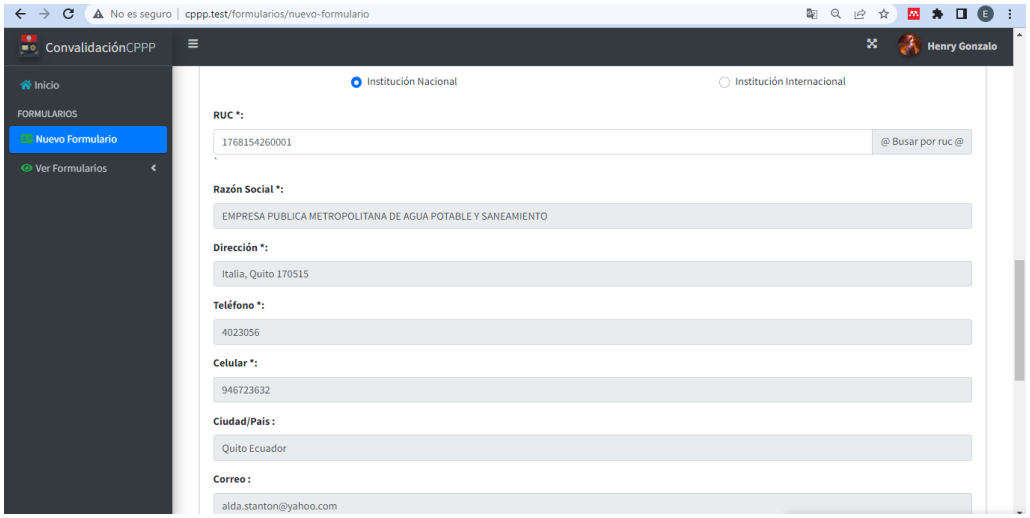

Figura 11. Llenado de formulario por el estudiante

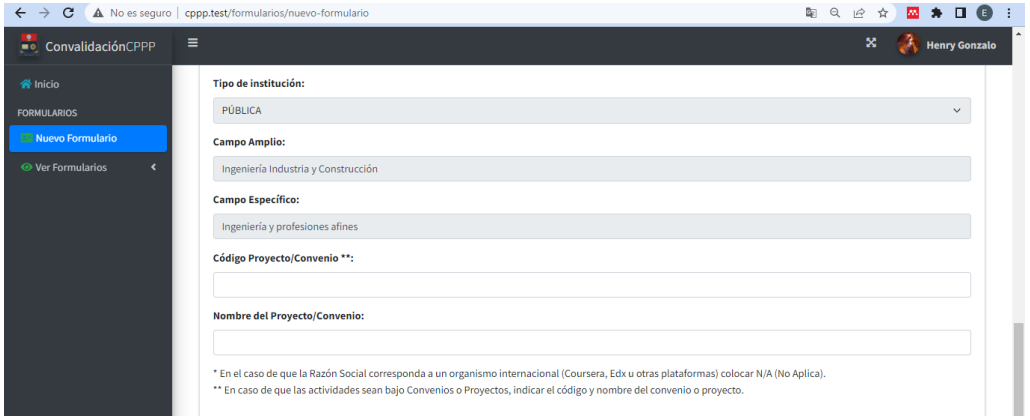

Figura 12. Llenado de formulario por el estudiante

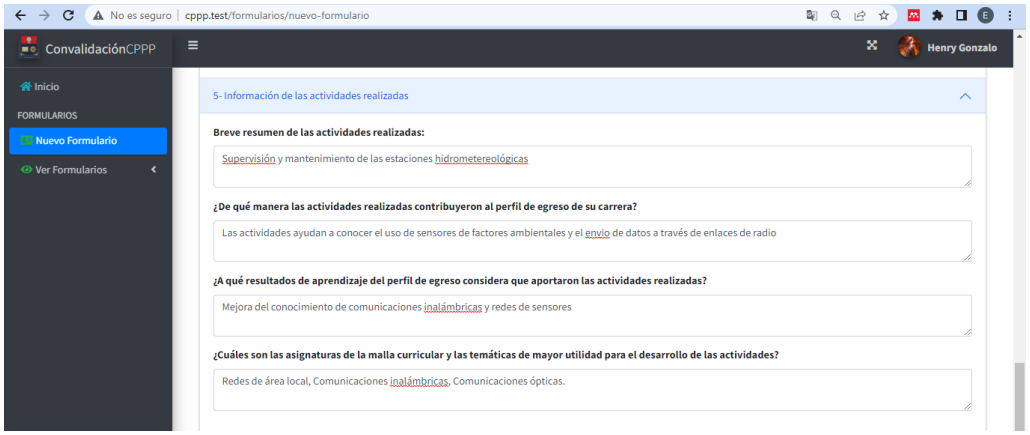

Figura 13. Llenado de formulario por el estudiante

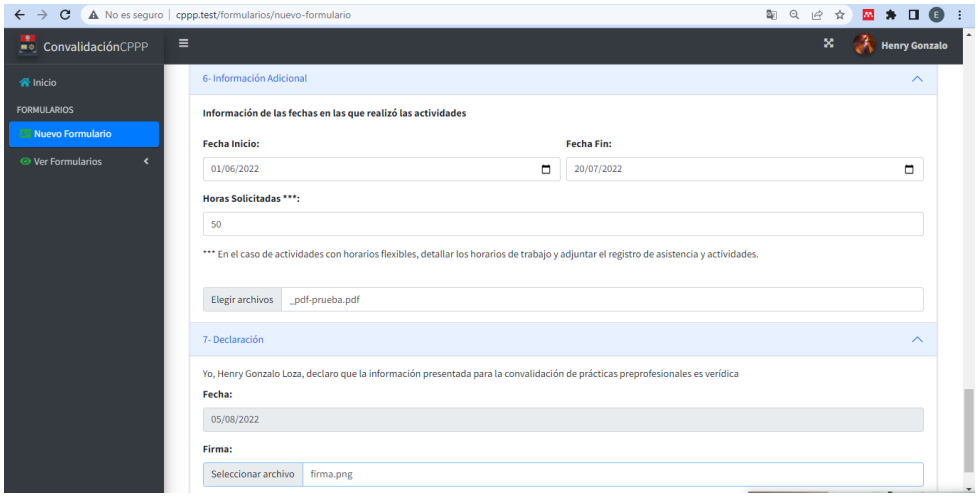

Figura 14. Llenado de formulario por el estudiante

Una vez llenado el formulario el estuidnate lo envia, obteniendo el mensaje que se muestra en la Figura 15.

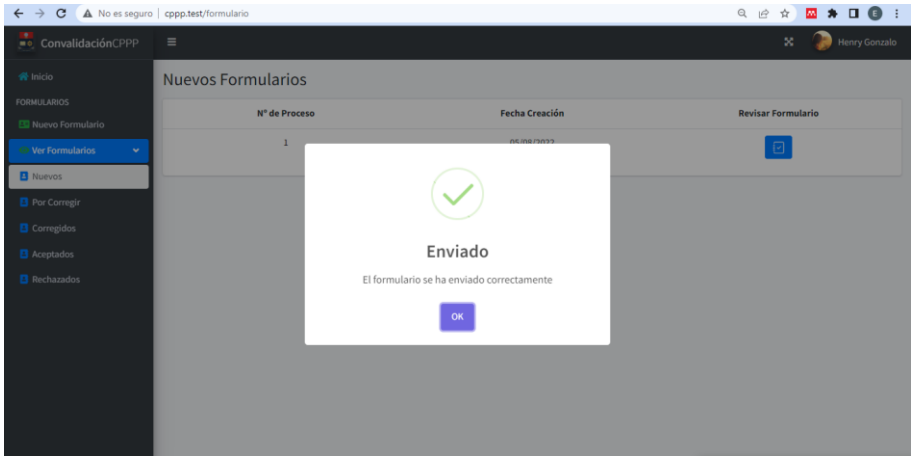

Figura 15. Mensaje de envió de formulario exitoso

En la Opción Nuevos el estuidnate puede revisar el formulario que acaba de enviar.

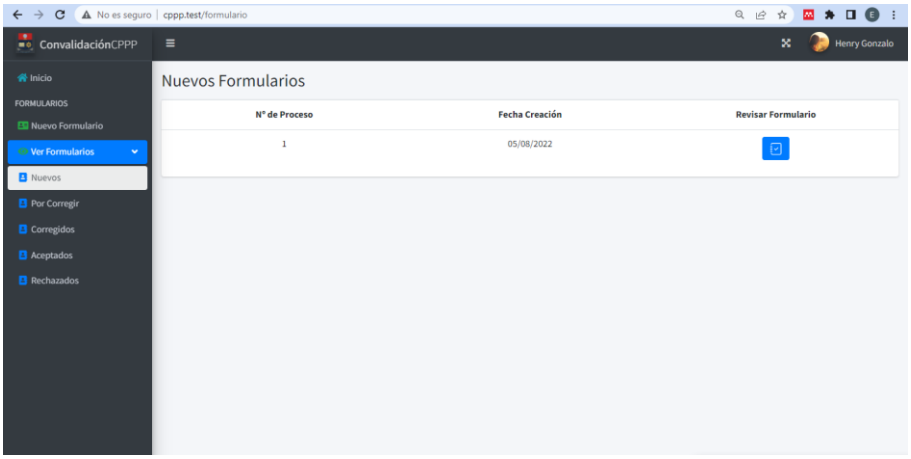

Figura 16. Ver el nuevo formulario enviado

4. Asignación de tutor por parte del subdecano

El segundo actor es el subdecano y es el encargado de revisar toda la documentación enviada por el estudiante y asignarle un tutor. Para esto primero debe acceder a la aplicación con su correo, esto se muestra en la Figura 17.

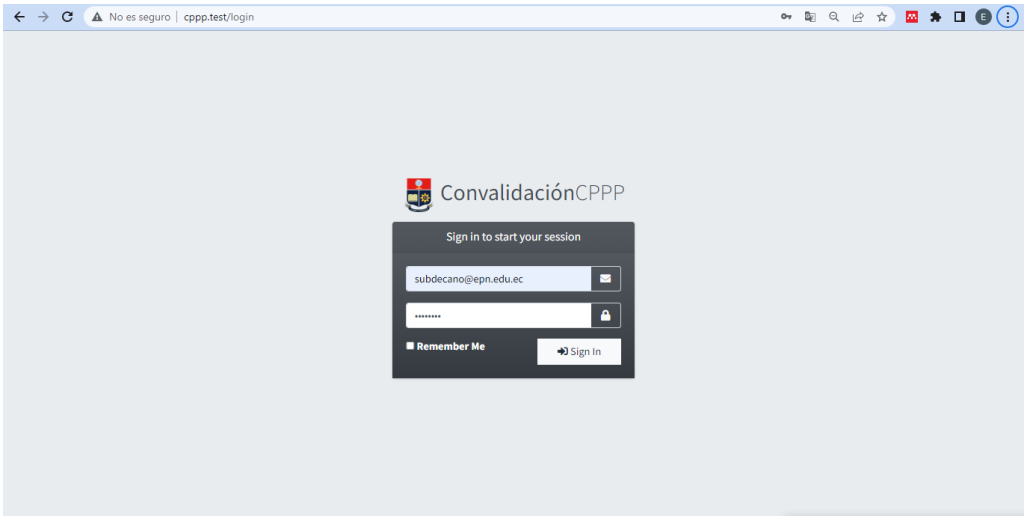

Figura 17. Login Sudecano

Una vez ingresado al sistema en la opción Nuevos el subdecano podrá ver el formulario que fue enviado por el estudiante y podrá revisarlo, dando click en el botón revisar.

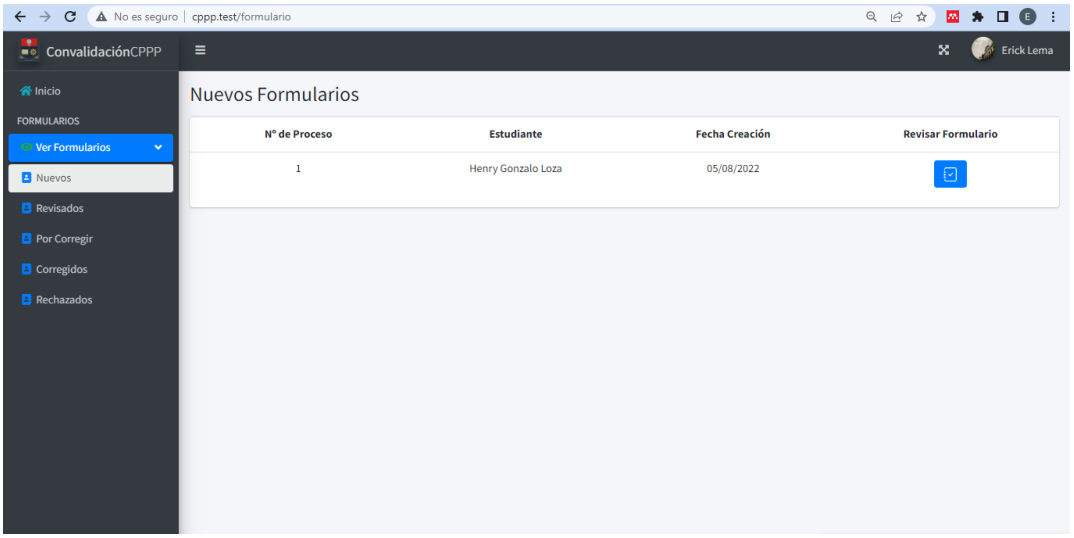

Figura 18. Nuevo formulario enviado al subdecano

Cuando el subdecano de click en revisar, se le despliega la vetana presentada en la Figura 19, donde se pueden observar todos los campos llenado por el estudiante y se le presentan nuevas opciones llamandas asignar tutor, solicitar correcciones o rechazar formulario.

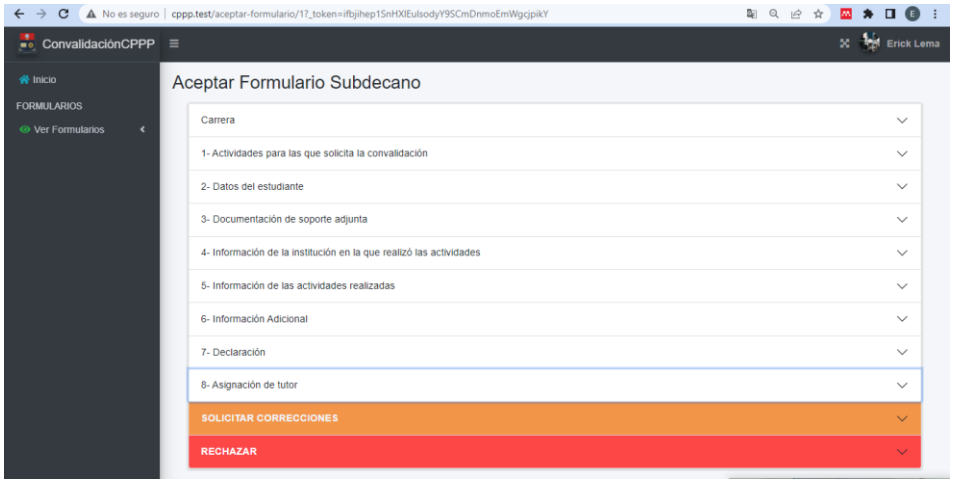

Figura 19. Revisar formulario por parte del subdecano.

El subdecano podrá revisar toda la información llenada por el estudiante sin opción a editarla, esto se muestra desde la Figura 20 a la 24

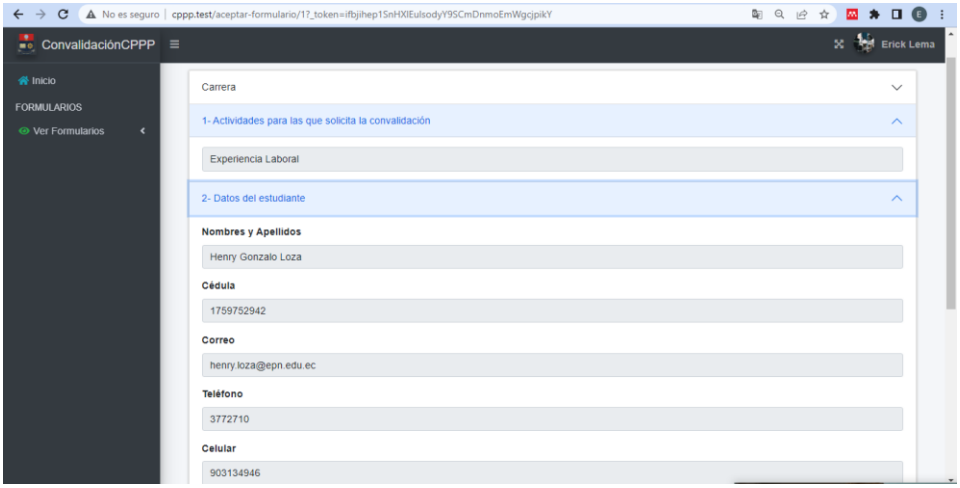

Figura 20. Revisión de formulario por parte del subdecano

Figura 21. Revisión de formulario por parte del subdecano

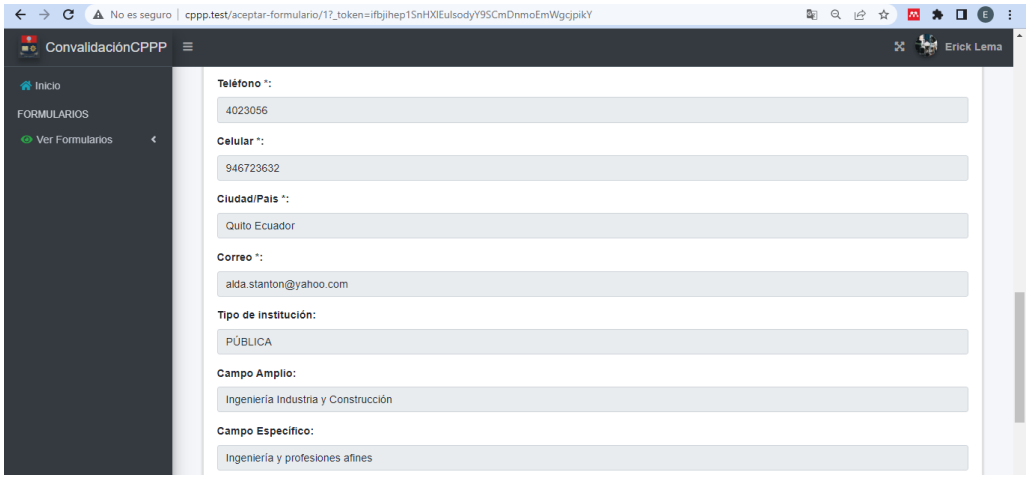

Figura 22. Revisión de formulario por parte del subdecano

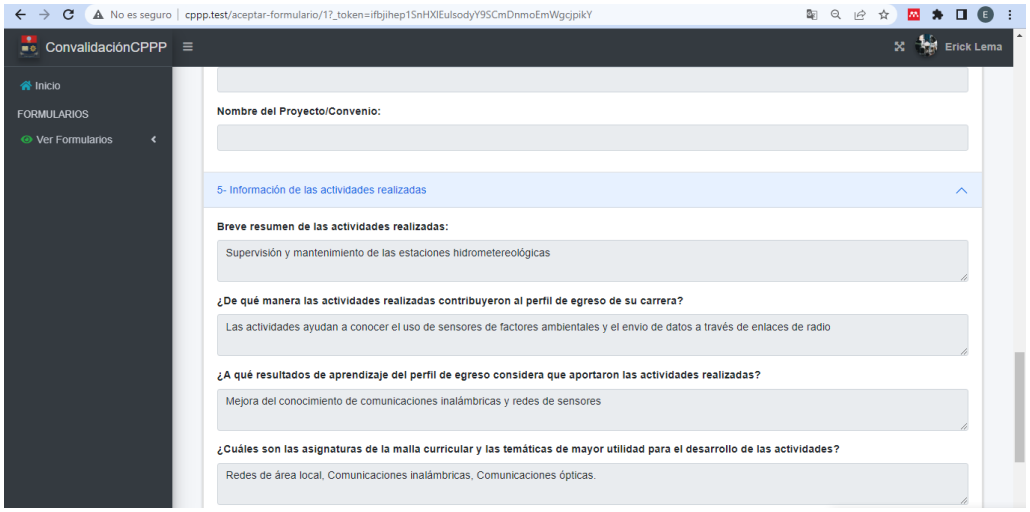

Figura 23. Revisión de formulario por parte del subdecano

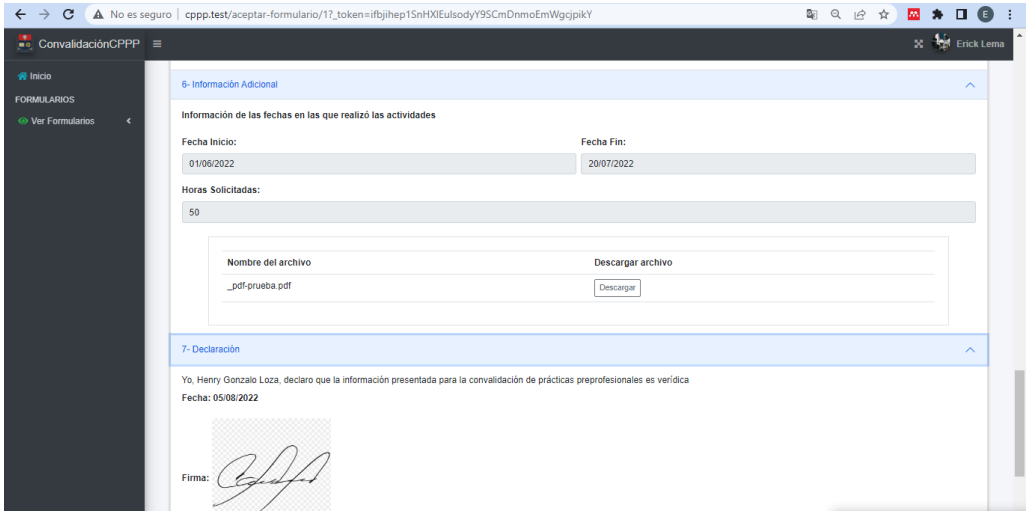

Figura 24. Revisión de formulario por parte del subdecano

Si el subdecano considera que todo está correcto, realiza la asignación de tutor, de no ser el caso puede solicitar correcciones o rechazar la solicitud, en este caso realiza la asignación de tutor como se muestra en la Figura 25.

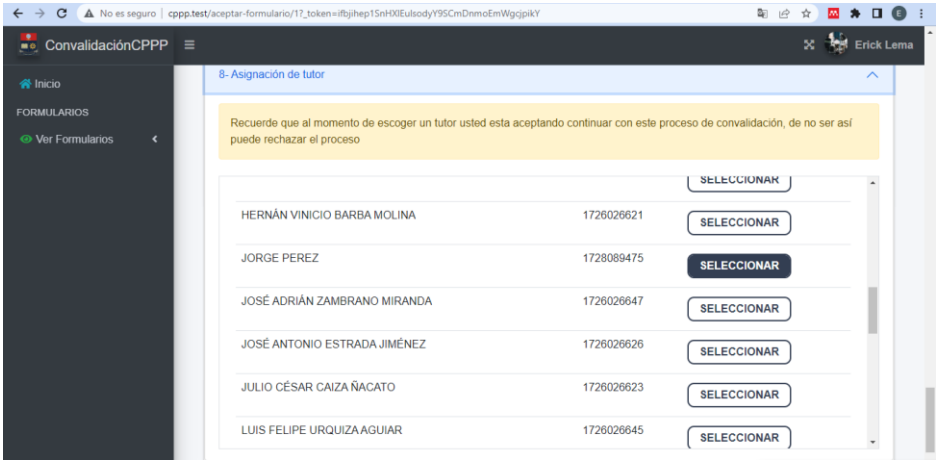

Figura 25. Asignación de tutor

El sistema le preguntará si está seguro de que quiere asignar un tutor

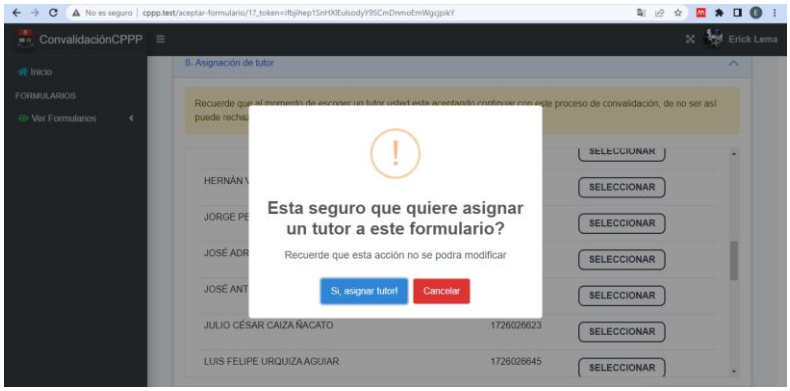

Figura 26. Confirmación se asignación de tutor

El sistema dará una respuesta de asignación correcta.

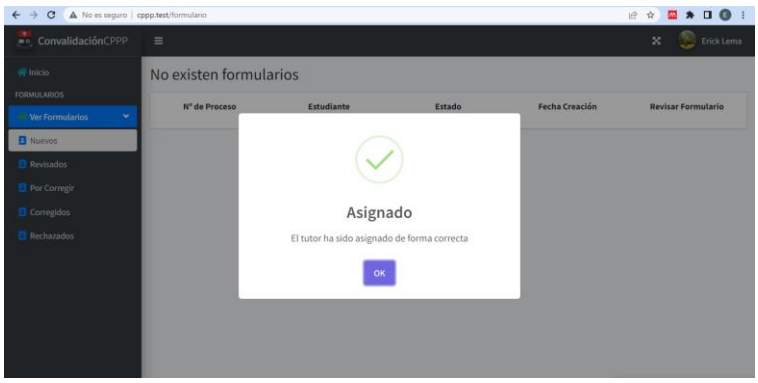

Figura 27. Asignación de tutor correcta

5. Revisión por parte del tutor

El tercer actor que interviene en el proceso es el tutor y es el encargado de revisar con mayor cautela la información llenada por el estudinate y dar su ecomnedación sobre si se debe o no convalidar las prácticas. Para esto primero se debera ingresar como tutor, como se muestra en la Figura 28

Figura 28. Login tutor

Estando ya en el sistema como tutor, en la opción nuevos, se encuentra el formulario al que el subdecano lo asigno como tutor, para revisarlo se debe dar en la opción revisar como se muestra en la Figura 29.

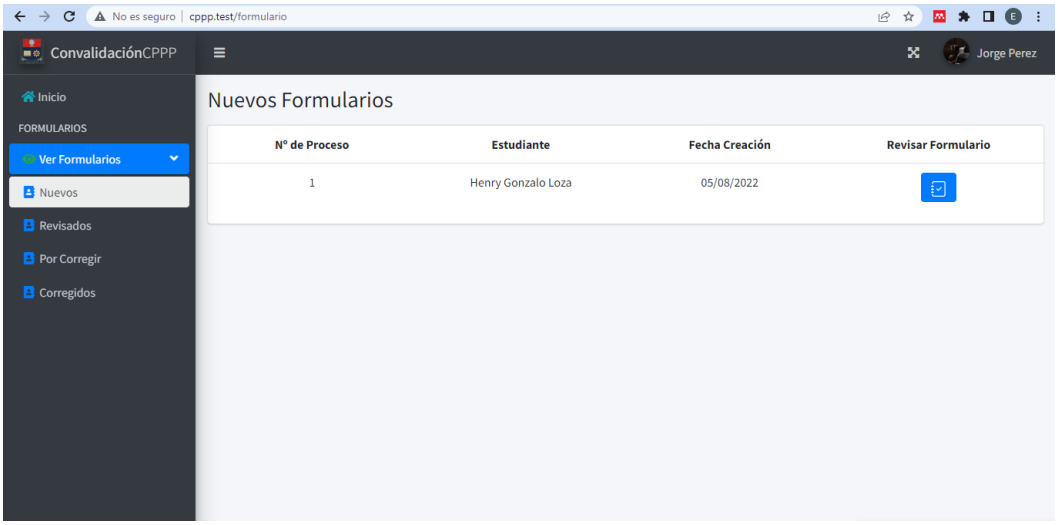

Figura 29. Revisión por parte del tutor

EL tutor revisara los campos llenado por el estudiante como se mostró en las Figuras de la 20 a la 24. Y adicional responderá las preguntas que se le plantean en el formulario, esto se presenta en la Figuras de 30 a la 33.

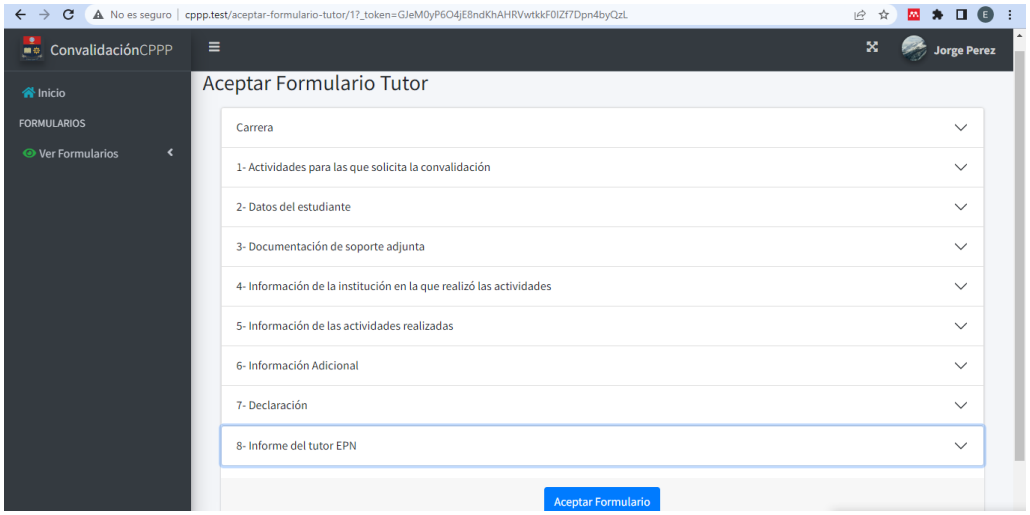

Figura 30. Revisión de formulario por parte del tutor

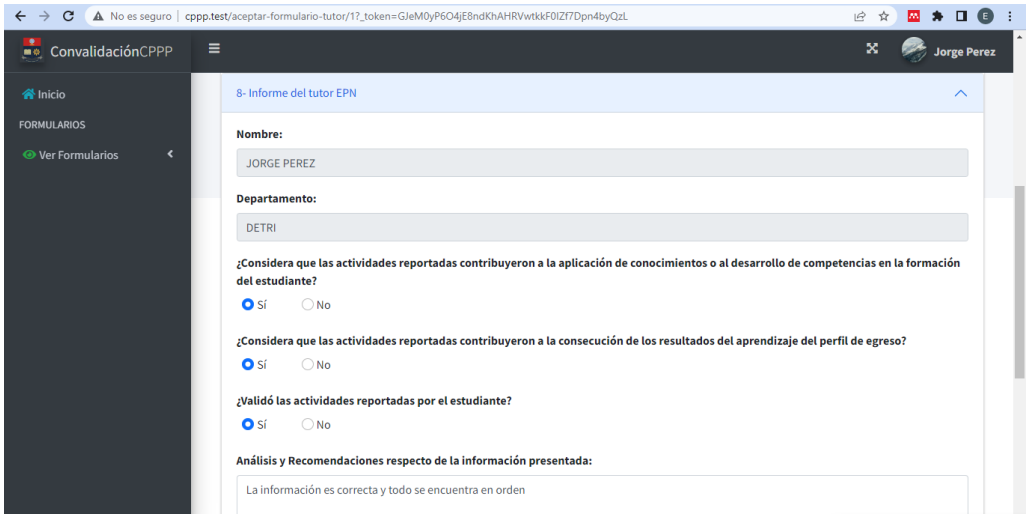

Figura 31. Llenado de campos correspondientes al tutor

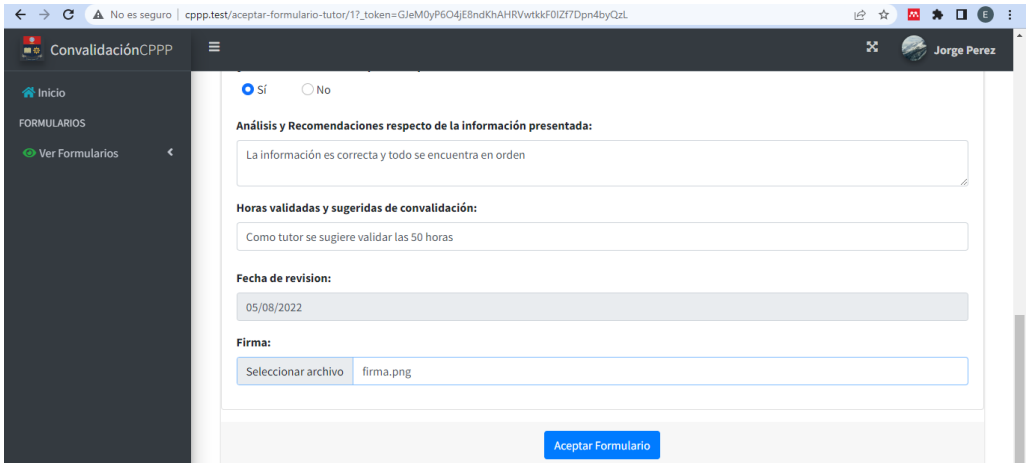

Figura 32. Llenado de campos correspondientes al tutor

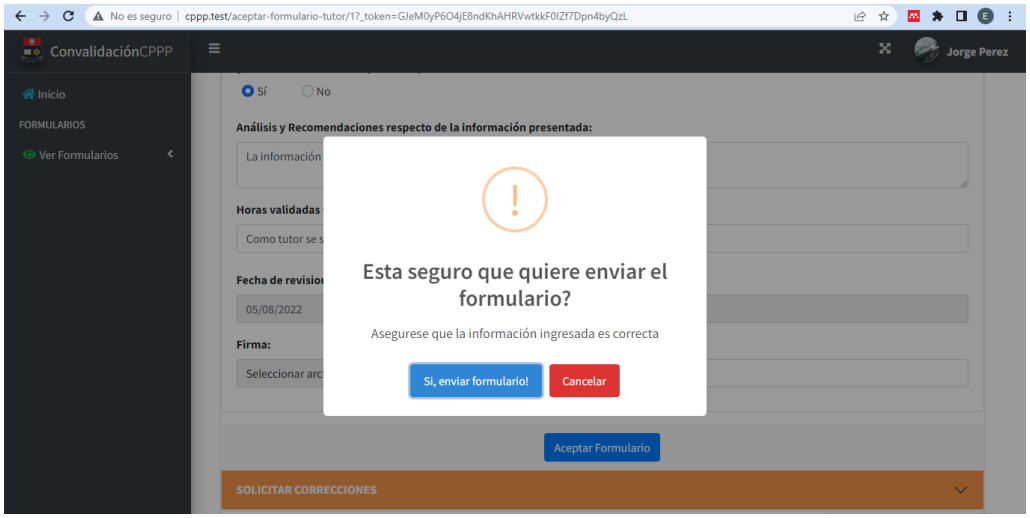

Figura 33. Confirmación de envío de formulario por parte del tutor

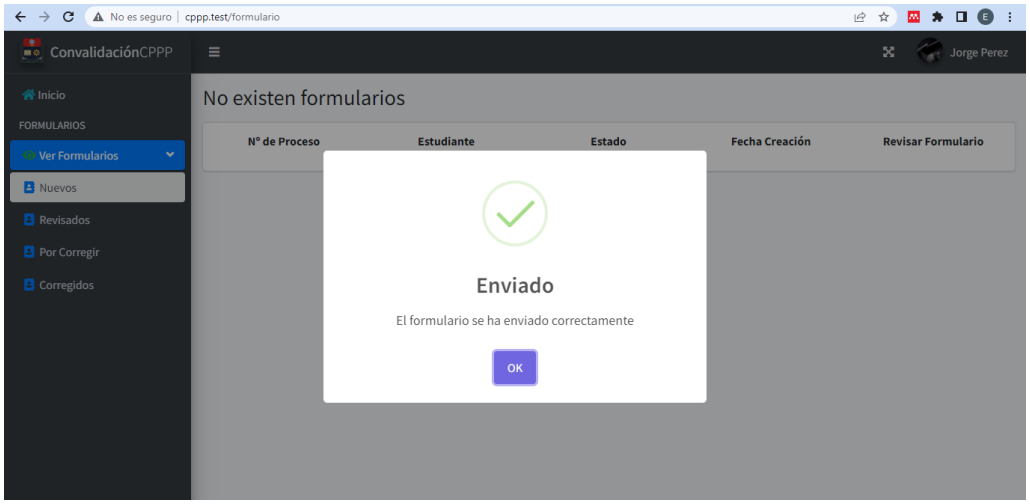

Figura 34. Envío exitoso del formulario por parte del tutor

6. Revisión por parte del miembro de la CPP

El siguiente actor que interviene en el proceso de convalidación es un miembro de la comisión de prácticas preprofesionales, este actor realiza las mismas actividades que el tutor, revisa la documentación y puede solicitar correcciones si esque se lo requiere o puede rechazar la solicitud. Para esto primero debe ingresar al sistema como se muestra en la Figura 35

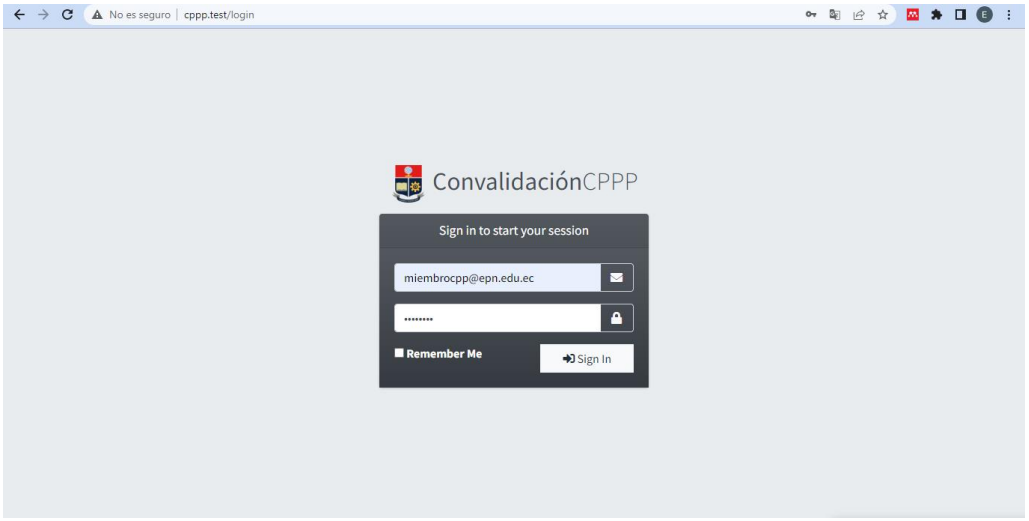

Figura 35. Login miembro de la CPP

Una vez en el sistema en la opción Nuevos, el miembro de la CPP encontrara el formulario que ya fue revisado por el tutor, esto se muestra enla Figura 36.

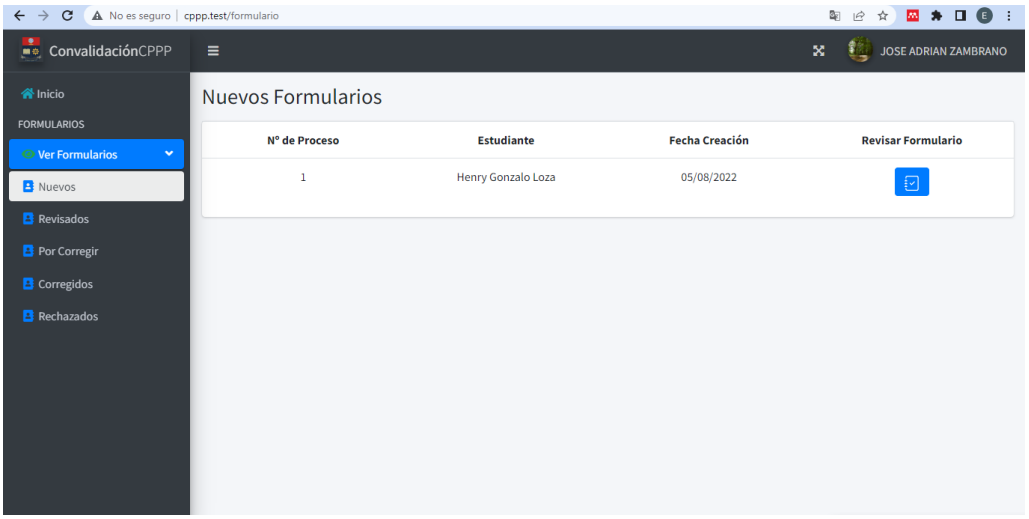

Figura 36. Revisión de formulario por miembro CPP

De igual manera podra revisar toda la información llenada por los anteriores actores sin la opción de editar. En este caso para comprobar la peticion de correcciones el Miembro de la CPP solicitará la correccion de la fecha de fin de prácticas en la sección 6 del formulario. Esto se muestra en la Figura 37. Y el formulario en este caso pasará el repositorio de formularios Por corregir, esperando que sea corregido.

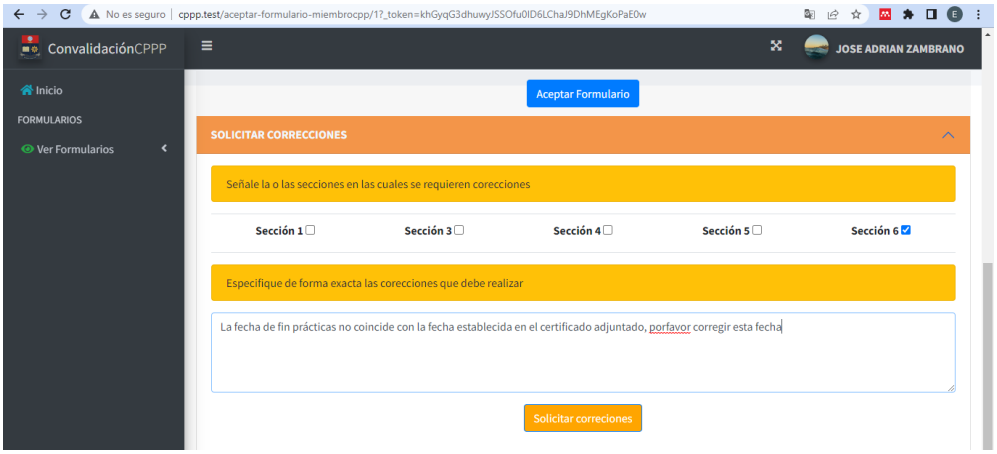

Figura 37. Solicitud de corrección

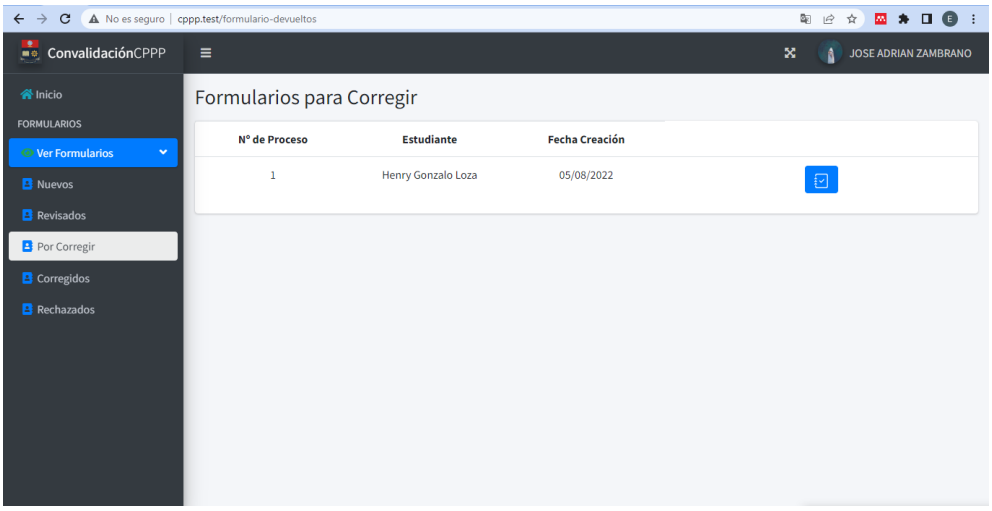

Figura 38. Formulario en el repositorio Por Corregir

En el estudiante el formulario cambia de repositorio, este se encuentra en el repositorio Por corregir, dando a entender que se requieren realizar correcciones en el mismo.

Figura 39. Formulario que debe ser corregido

Cuando el estudinate va a corregir el formulario, en este se muestran los mensajes de la Figura 40 donde se describen las correcciones que debe hacer y en que sección.

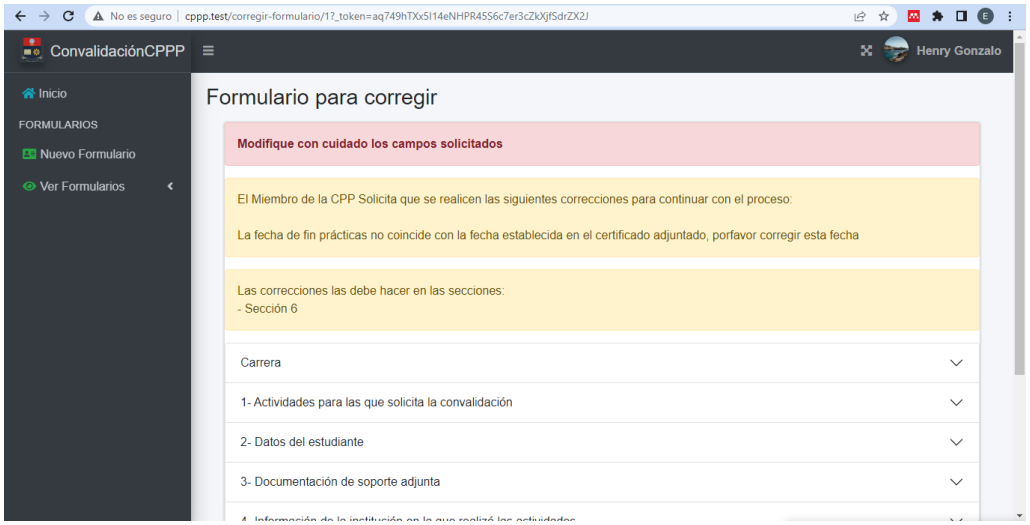

Figura 40. Mensajes de corrección

Como es la sección 6 donde se debe realizar corrección, solo esta se encuentra para editar, todos los demas campos se encuentran bloqueados. Aquí el estuidnate realiza la corrección como se muestra en la Figura 41.

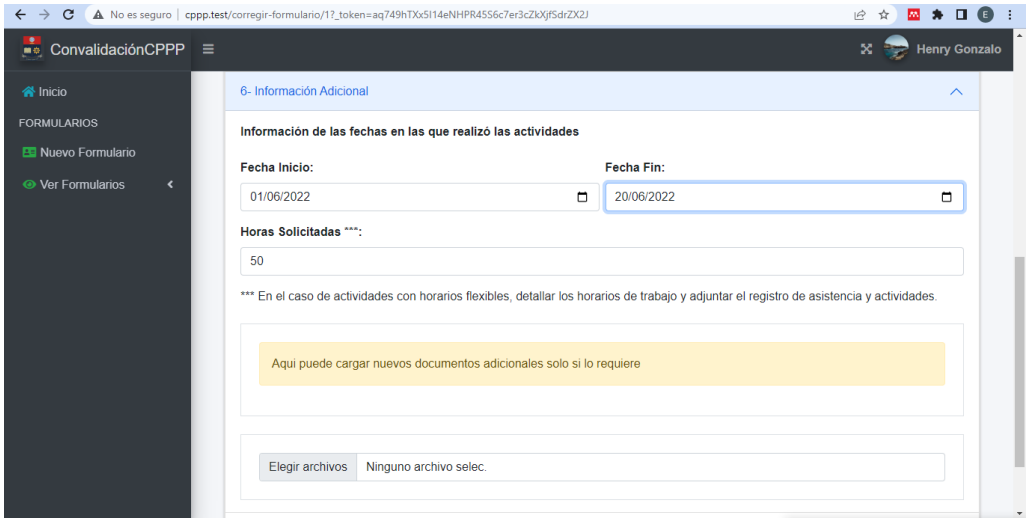

Figura 41. Corrección del campo solicitado

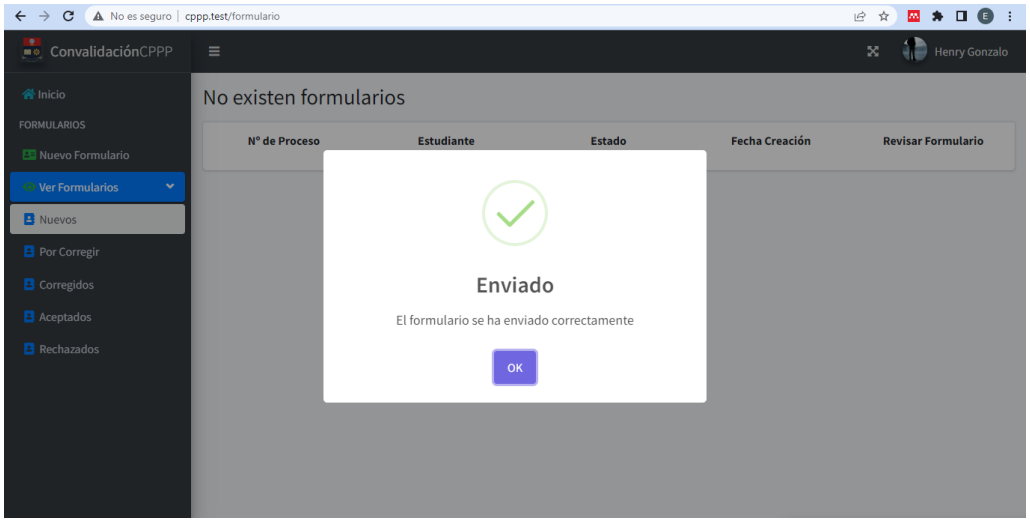

Figura 42. Envío exitoso del formulario corregido

Luego de que el estudinate realiaza la corrección, el miembro de la CPP quien fue el que lo solicito, podra encontrar este formulario en el repositorio, Corregidos, lo que indica que el estudiante ya corrigio lo colicitado, esto se muestra en la Figura 43

Figura 43. Formulario corregido por el estudiante

El miembro de la CPP podrá revisar la corrección realiza como y también un mensaje de cuáles fueron las correcciones que se debían hacer como se muestra en la Figura 44 y 45.

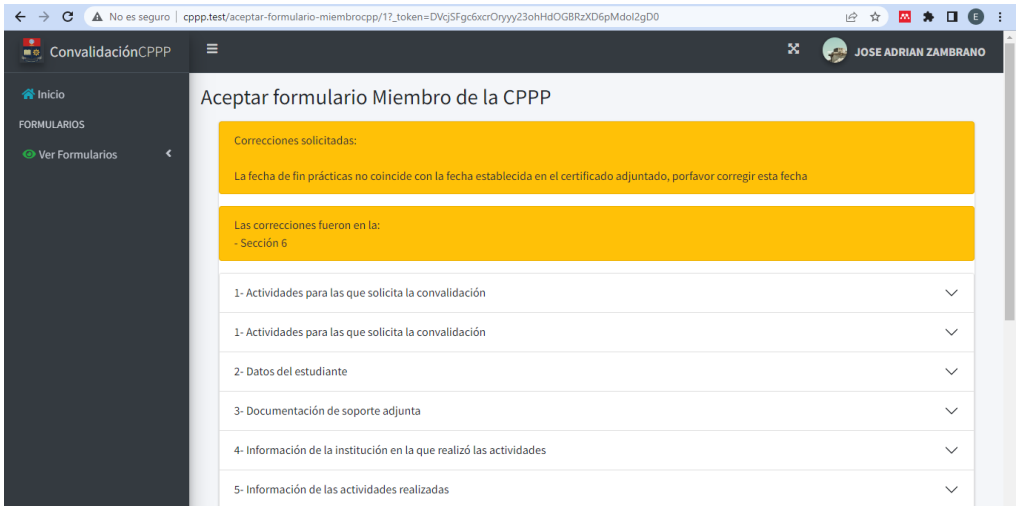

Figura 44. Mensaje de las correcciones que fueron solicitadas

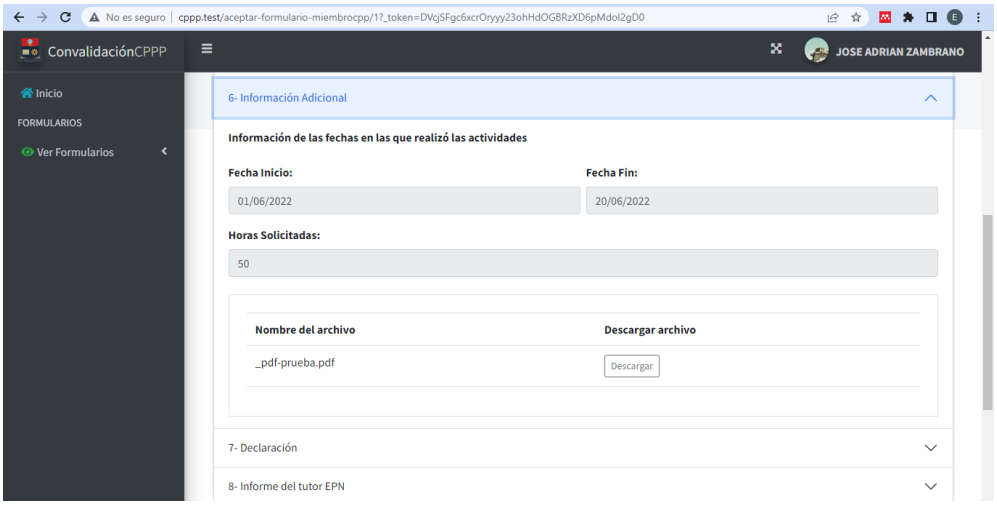

Figura 45. Verificación de las correcciones realizadas

Después de revisar y que todo este correcto el miembro de a CPP da su recomendación sobre la convalidación y envía el formulario.

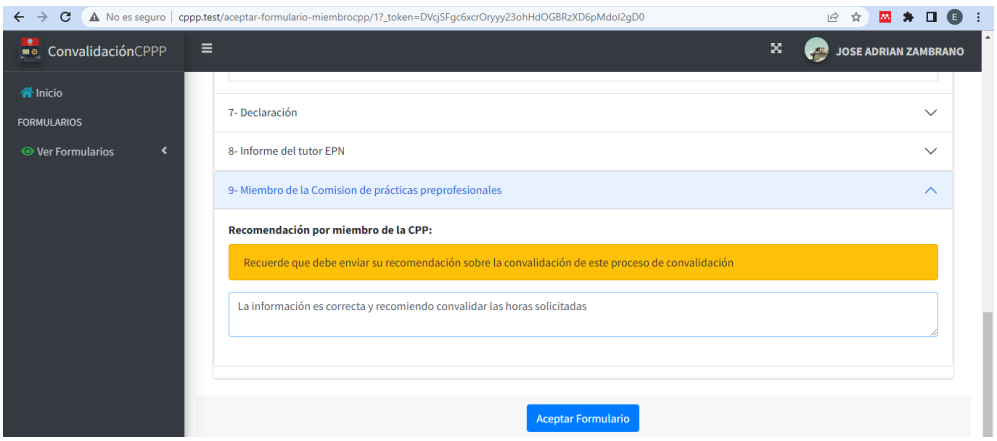

Figura 46. Recomendación del miembro de la CPP

7. Revisión por parte del coordinador de la CPP

El siguiente actor es el coordinador de la CPP quien es el encargado de aceptar o no la convalidación en base a las revisiones hechas por el tutor y el miembro de la CPP, para esto primero se debe ingresar al sistema como se muestra en la Figura 47.

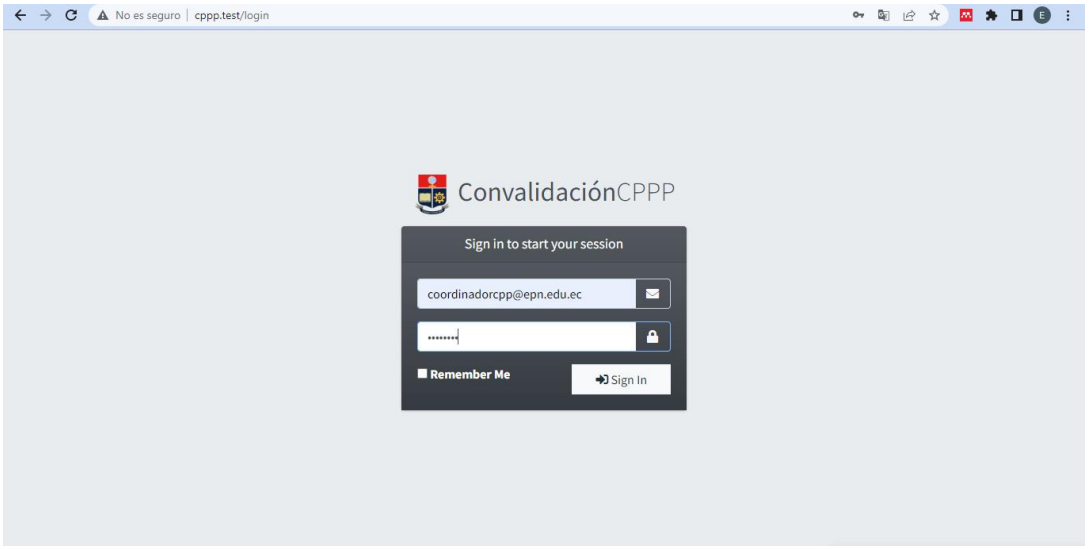

Figura 47. Login coordinadorCPP

Estando ya en el sistema, el coordinador de la CPP encontrará el formulario en la Opción Nuevos, donde podrá revisar el formulario. Esto se muestra en la Figura 48.

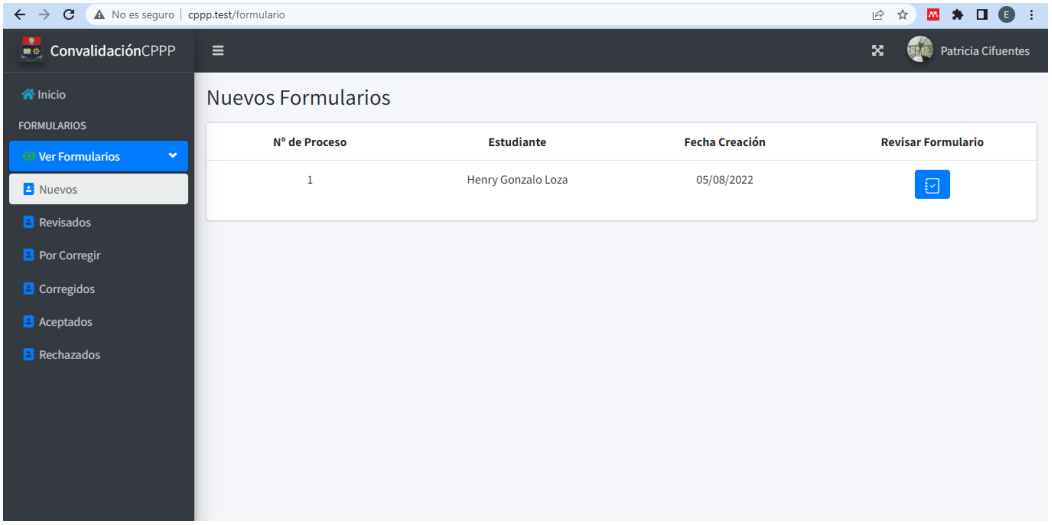

Figura 48. Nuevo formulario para revisar por el coordinador CPP

El coordinador de la CPP de igual manera podrá revisar toda la información llenada por los anteriores actores sin opción a editar y adicional a ello, llenara los campos que le corresponden como se muestra en las Figuras 49.

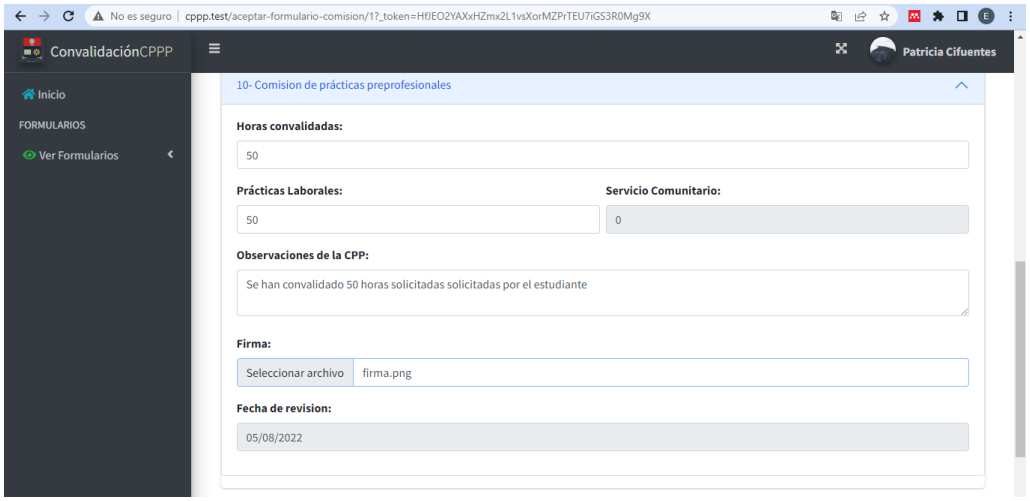

Figura 49. Llenado de los campos correspondiente al miembro de la CPP

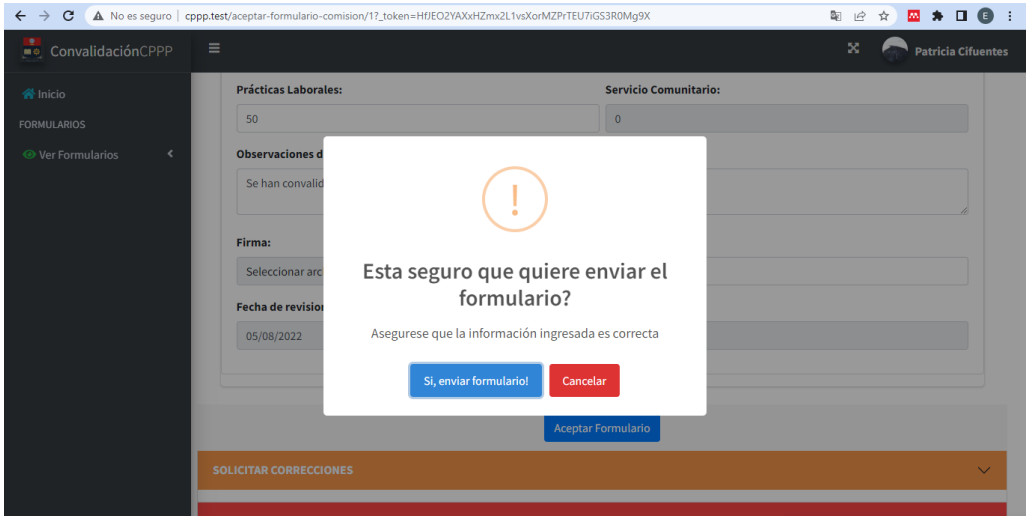

Figura 50. Confirmación de envío del formulario

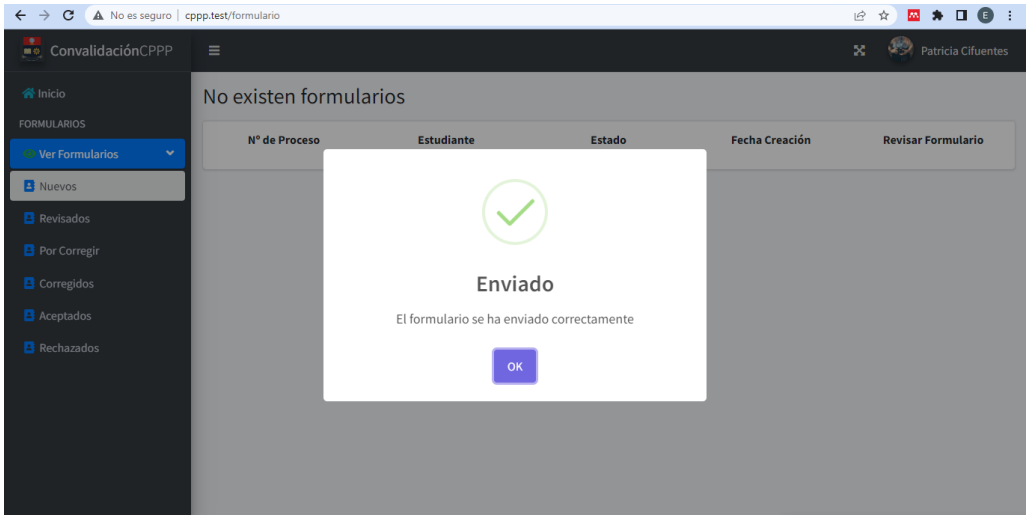

Figura 51. Envío exitoso

8. Autorización por parte del Decano

El último actor que interviene en el proceso es el Decano, y es quien autoriza que las horas sean registradas en el sistema saew. En esta parte el decano únicamente carga una forma lo cual le permite descargar el formulario en el formato original. Para ello lo primero es ingresar al sistema como se muestra en la Figura 52

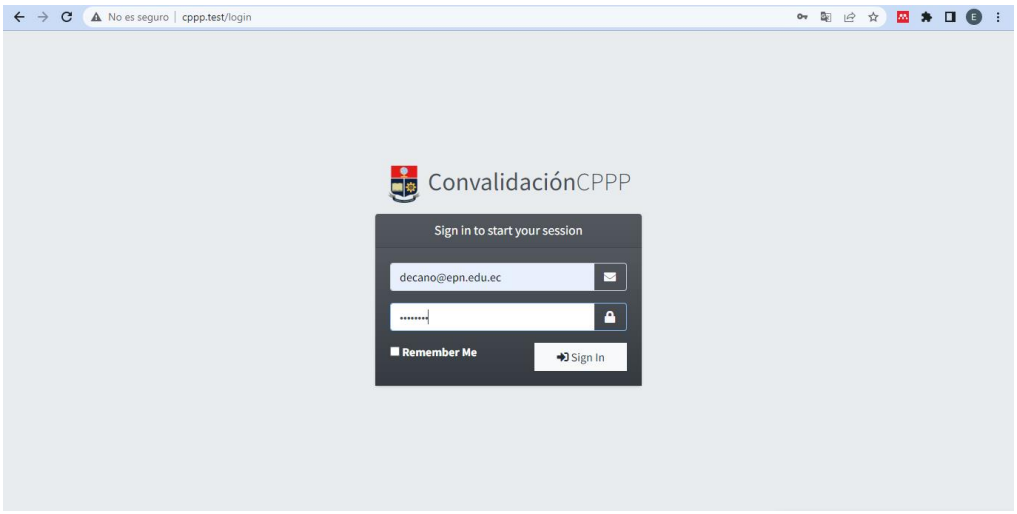

Figura 52. Login decano

En la opción Nuevos el Decano encontrará los formularios aceptados por el coordinador de la CPP, la opción revisar, le permitirá cargar su firma como se muestra en la Figura 54.

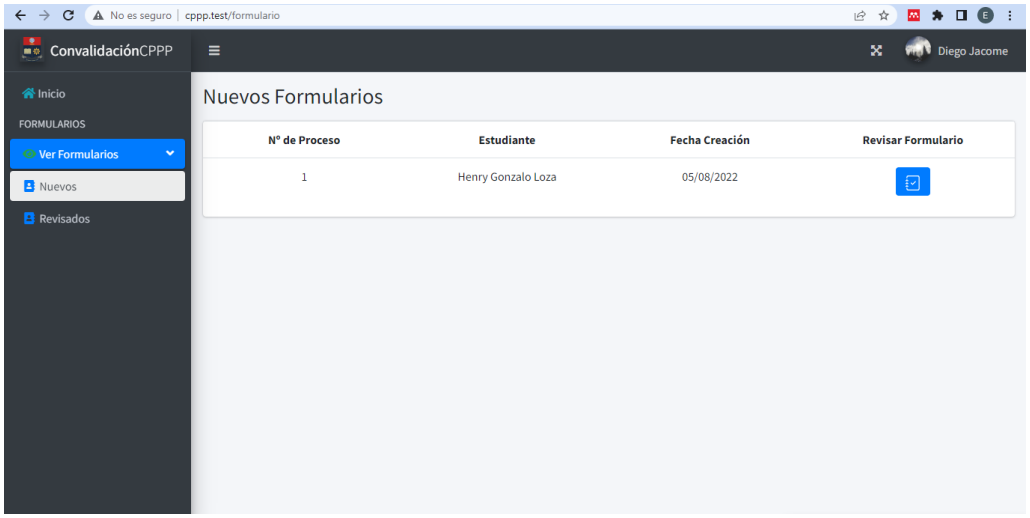

Figura 53. Revisión por parte del decano

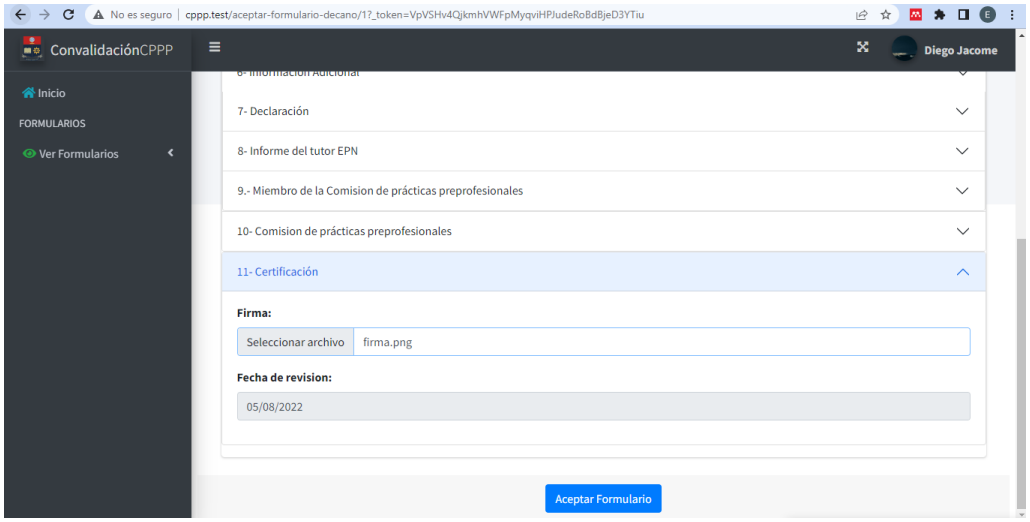

Figura 54. Carga de firma del decano

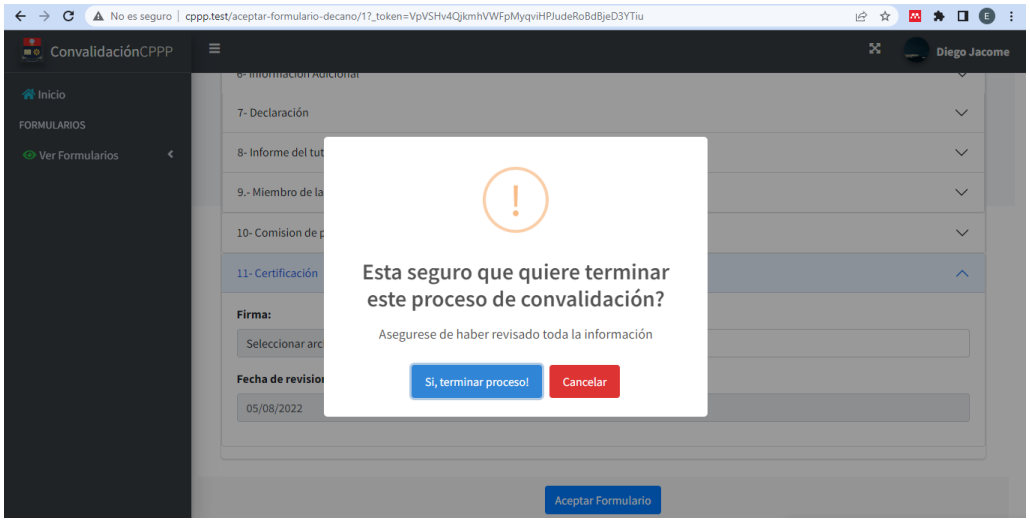

Figura 55. Confirmación de envío de firma

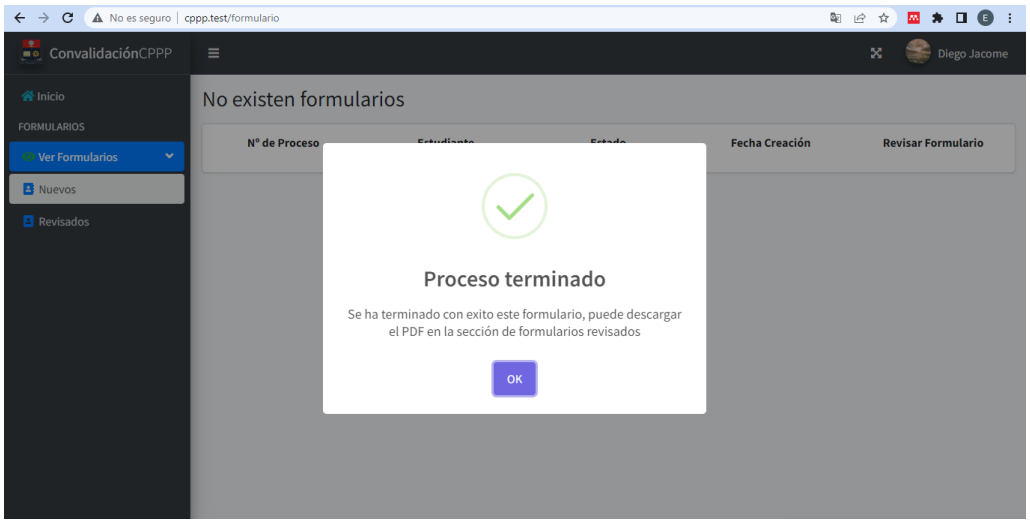

Figura 56. Proceso terminado con éxito

Una vez aceptado el formulario, este se dirigirá al repositorio aceptados, donde se permite descargar el formulario en el formato original como se muestra en la Figura 57

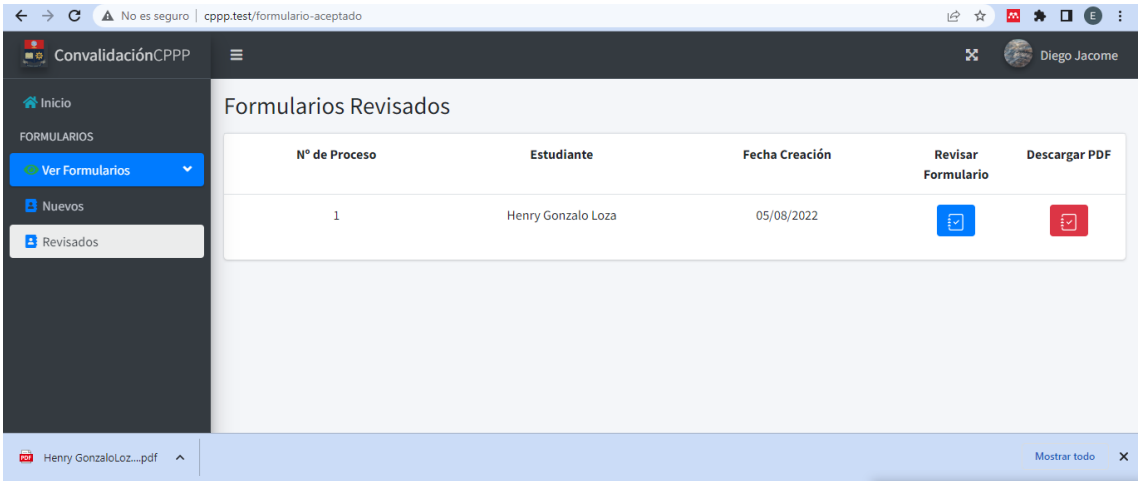

Figura 57. Descarga de archivo en el formato original

A continuación, se presenta el PDF final con el formato original del formulario.

ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE ELÉCTRICA Y ELECTRÓNICA

FORMULARIO FCP_001A
Versión 2

CARRERA: **(RRA20) Telecomunicaciones**

CONVALIDACIÓN DE PRÁCTICAS PREPROFESIONALES

SECCIONES A SER LLENADAS POR CADA RESPONSABLE:
Estudiante (resaltadas en verde)
Tutor (resaltada en amarillo)
CPP (resaltada en anaranjado)

1. ACTIVIDADES PARA LAS QUE SOLICITA LA CONVALIDACIÓN

	Cursos y Seminarios Profesionales		Idiomas diferentes al Inglés y Lengua Materna
	Participación Estudiantil en Actividades Académicas, de Gestión, de Investigación y de Colaboración en Eventos Académicos **		Dirección de ramas de organizaciones estudiantiles académicas
	Representación Estudiantil		Representación de la Institución en competencias académicas
	Estudiantes mentores		Coro y Grupo de Cámara
	Representación de la Institución en competencias deportivas		Participación en la dirección de asociaciones de estudiantes
	Actividades solidarias y de cooperación		Participación en juntas receptoras del voto
X	Experiencia Laboral		

2. DATOS DEL ESTUDIANTE

Nombres y Apellidos:	Henry GonzaloLoza				
Cédula de Identidad:	1759752942				
Correo electrónico:	henry.loza@epn.edu.ec	Teléfono:	3772710	Celular:	903134946

3. DOCUMENTACIÓN DE SOPORTE ADJUNTA

Lista de Documentos subidos:	_pdf-prueba.pdf

4. INFORMACIÓN DE LA INSTITUCIÓN EN LA QUE REALIZÓ LAS ACTIVIDADES

Razón Social *:	EMPRESA PUBLICA METROPOLITANA DE AGUA POTABLE Y SANEAMIENTO				
RUC *:	1768154260001				
Dirección *:	Italia, Quito 170515				
Ciudad/País:	Quito Ecuador	Teléfono:	4023056	Celular:	946723632
Correo electrónico:	alda.stanton@yahoo.com				
Tipo de Institución:	PÚBLICA				
Campo Amplio:	Ingeniería Industria y Construcción				
Campo Específico:	Ingeniería y profesiones afines				
Código de Proyecto/Convenio **:		Nombre del Proyecto/Convenio:			

** En el caso de que la Razón Social corresponda a un organismo internacional (Coursera, Edx u otras plataformas) colocar N/A (No Aplica).*
*** En caso de que las actividades sean bajo Convenios o Proyectos, indicar el código y nombre del convenio o proyecto.*

5. INFORMACIÓN DE LAS ACTIVIDADES REALIZADAS

Breve resumen de las actividades realizadas:
Supervisión y mantenimiento de las estaciones hidrometereológicas
¿De qué manera las actividades realizadas contribuyeron al perfil de egreso de su carrera?
Las actividades ayudan a conocer el uso de sensores de factores ambientales y el envio de datos a través de enlaces de radio
¿A qué resultados de aprendizaje del perfil de egreso considera que aportaron las actividades realizadas?
Mejora del conocimiento de comunicaciones inalámbricas y redes de sensores
¿Cuáles son las asignaturas de la malla curricular y las temáticas de mayor utilidad para el desarrollo de las actividades?
Redes de área local, Comunicaciones inalámbricas, Comunicaciones ópticas.

6. INFORMACIÓN ADICIONAL

Información de las fechas en las que realizó las actividades			
Fecha inicio:	**2022-06-01**	**Fecha fin:**	**2022-06-20**
Horas solicitadas *:50**			

*** En el caso de actividades con horarios flexibles, detallar los horarios de trabajo y adjuntar el registro de asistencia y actividades.*

7. DECLARACIÓN

Yo, Henry GonzaloLoza, declaro que la información presentada para la convalidación de prácticas preprofesionales es verídica.			
Fecha:	**2022-08-05**	**Firma:**	

8. INFORME DEL TUTOR EPN

Nombre: JORGE PEREZ			**Departamento:**	**DETRI**
¿Considera que las actividades reportadas contribuyeron a la aplicación de conocimientos o al desarrollo de competencias en la formación del estudiante?	**SI:**	**X**	**NO:**	
¿Considera que las actividades reportadas contribuyeron a la consecución de los resultados del aprendizaje del perfil de egreso?	**SI:**	**X**	**NO:**	
¿Validó las actividades reportadas por el estudiante?	**SI:**	**X**	**NO:**	
Análisis y Recomendaciones respecto de la información presentada:				
La información es correcta y todo se encuentra en orden				
Horas validadas y sugeridas de convalidación:	**Como tutor se sugiere validar las 50 horas**			

9. COMISIÓN DE PRÁCTICAS PREPROFESIONALES

Horas convalidadas:	**50**		
Prácticas Laborales:	**50**	**Servicio Comunitario:**	**0**
Observaciones de la CPP:	**Se han convalidado 50 horas solicitadas solicitadas por el estudiante**		

10. CERTIFICACIONES

TUTOR DE PRÁCTICAS PREPROFESIONALES
Fecha de Recepción: 2022-08-05
Fecha de Revisión: 2022-08-05

f. _____
Tutor
Nombre: JORGE PEREZ

COMISIÓN DE PRÁCTICAS PREPROFESIONALES
Fecha de Recepción:
Fecha de Revisión: 2022-08-05

f. _____
Presidente
Nombre: Patricia Cifuentes

DECANO(A) DE LA FACULTAD / DIRECTOR(A) DE LA ESFOT
Fecha de Recepción: 2022-08-05
Fecha de Revisión: 2022-08-05

f. _____
Decano (a) / Director (a)
Nombre: Diego Jacome

Fecha de Registro en SAEw:_____ Responsable Registro SAEw:_____

9. Almacenamiento de archivos adjuntos

Otro punto importante de la revisión del almacenamiento de los archivos, esto se presenta en las Figuras de la 58 a la 60. Donde se puede observar que la estructura de los directorios de almacenamiento, corresponden a los presentados en la fase de diseño.

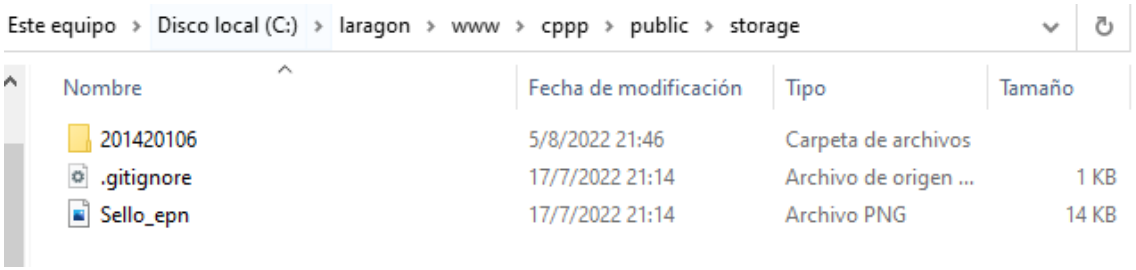

Figura 58. Directorio principal con el número único del estudiante

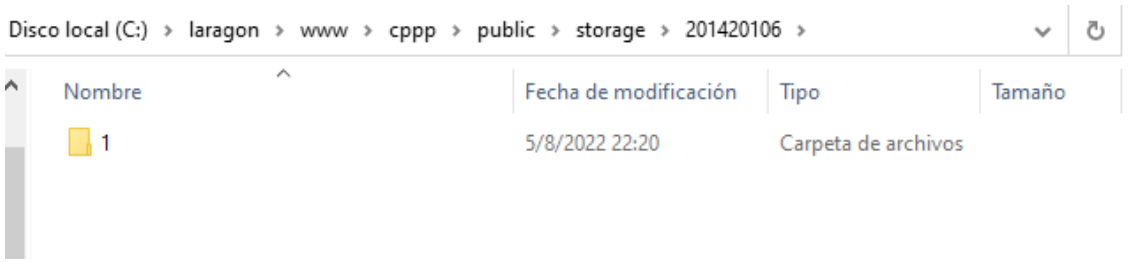

Figura 59. Subdirectorio con el Id del formulario

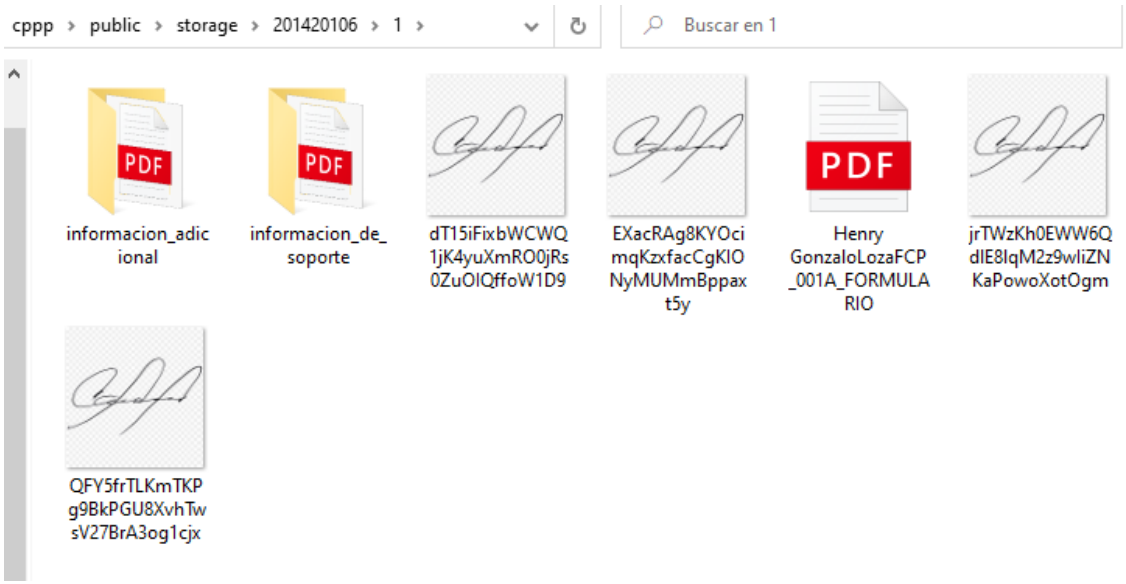

Figura 60. Directorios para archivos de soporte e información adicional

ANEXO VII

En este anexo se presentan las encuestas realizadas a algunos estudiantes de la carrera de Telecomunicaciones de la Escuela Politécnica Nacional, con quienes se realizaron pruebas de funcionamiento de la aplicación web. Las respuestas obtenidas en esta encuesta reflejan el resultado de la implementación de la aplicación web ConvalidaciónCPPP. En las Tablas 1, 2 y 3 se presentan las respuestas obtenidas por parte de 3 estudiantes diferentes.

Tabla 1. Encuesta realizada al primer estudiante.

#	PREGUNTAS	RESPUESTAS
1	¿Le parece fácil entender la interfaz de usuario?	La interfaz resulto muy fácil de entender, ya que está muy ordenado y muy adecuada.
2	¿Los módulos implementados cumplen con su función?	En general muy didáctica puesto que despliegan muchas opciones que tiene la aplicación y se puede usar para varios procedimientos.
3	¿Le resulta fácil llenar el formulario o existió alguna confusión?	Si, incluso en campos que son difíciles de saber por ejemplo al ingresar el RUC se automatiza los campos relacionados a esa organización facilitando el trabajo, además cuando se debía ingresar archivos estaba claramente especificado que tipo de archivo subir.
4	¿El sistema de correcciones es comprensible?	Si, resulta fácil realizar las correcciones y la aplicación muestra de forma clara que correcciones hacer y en que sección. Esto ayuda a reducir tiempos en el proceso.
5	¿Le parece útil la aplicación para realizar el proceso de convalidación de prácticas preprofesionales?	Si, ya que en realizar un formulario no se demora más de 5 minutos, cosa que actualmente se puede demorar incluso horas llenando formularios de forma manual y tediosa.
6	¿Tiene alguna recomendación sobre mejoras que se puedan implementar?	Tal vez si hacer un poco más entendible ciertas preguntas puesto que presente un poco de problemas ya que no entendía a que se refería la pregunta

Tabla 2. Encuesta realizada al segundo estudiante.

#	PREGUNTAS	RESPUESTAS
1	¿Le parece fácil entender la interfaz de usuario?	Si, las opciones son entendibles y sencillas de manejar.
2	¿Los módulos implementados cumplen con su función?	Si, todo cumple la función específica.
3	¿Le resulta fácil llenar el formulario o existió alguna confusión?	Es fácil llenar, todo está en orden
4	¿El sistema de correcciones es comprensible?	Si, se puede verificar todo sin complicar otros campos.
5	¿Le parece útil la aplicación para realizar el proceso de convalidación de prácticas preprofesionales?	Si, es una forma más sencilla y rápida de poder verificar nuestro proceso.
6	¿Tiene alguna recomendación sobre mejoras que se puedan implementar?	Para el proceso de notificaciones se debería implementar que estas sean por correo electrónico.

Tabla 3. Encuesta realizada al tercer estudiante.

#	PREGUNTAS	RESPUESTAS
1	¿Le parece fácil entender la interfaz de usuario?	Si, la interfaz resulta fácil de entender y amigable para el usuario
2	¿Los módulos implementados cumplen con su función?	Si, todos los módulos implementados cumplen con su función de forma correcta.
3	¿Le resulta fácil llenar el formulario o existió alguna confusión?	Si, llenar el formulario mediante la aplicación web resulta más sencilla que llenarla de forma manual como se la hace actualmente.
4	¿El sistema de correcciones es comprensible?	Si, la aplicación muestra de una forma clara las correcciones que se deben hacer y en que sección del formulario.
5	¿Le parece útil la aplicación para realizar el proceso de convalidación de prácticas preprofesionales?	Es una herramienta bastante útil y agiliza el proceso de convalidación. Permite que el proceso sea más rápido y sencillo.
6	¿Tiene alguna recomendación sobre mejoras que se puedan implementar?	La plataforma está muy bien desarrollada.

ANEXO VIII

En este Anexo se presentan tres enlaces de las grabaciones realizadas en la pruebas de funcionamiento junto a tres estudiantes de la carrera de Telecomunicaciones. En estas grabaciones se puede observar el funcionamiento de la aplicación y cómo reaccionan los estudiantes con el uso de la aplicación.

Enlace 1:

https://www.youtube.com/watch?v=nzmf-xaYAhs

Enlace 2:

https://www.youtube.com/watch?v=ZavKwk_GeXs

Enlace 3:

https://www.youtube.com/watch?v=uMLfvxQzCTc