



DIRECCIÓN ACADÉMICA
VICERRECTORADO ACADÉMICO

Facultad de Ingeniería

Carrera de Ingeniería en Tecnologías de la Información

Informe de Actividad de Investigación Formativa

Periodo Académico
2024-1S

Contenido

1.	Autores.....	3
2.	Personal Académico.....	3
3.	Resultados de Aprendizaje de la asignatura.....	3
4.	Tema de la Actividad de la Investigación Formativa	3
5.	Objetivos de las actividades:	3
6.	Fecha de la ejecución: 12 de JUNIO DEL 2024	3
7.	Desarrollo del Informe	4
7.1	Introducción	4
7.2	Descripción de la metodología	5
7.3	Descripción de las acciones realizadas	8
7.4	Resultados	23
7.5	Bibliografía	24
8.	ANEXO	25



1. Autores

- Leidy Dayana Tene Caiza

2. PERSONAL ACADÉMICO

- Director de Carrera: Jorge Delgado
- Profesor de Asignatura: Ing. Milton Paul Lopez

3. RESULTADOS DE APRENDIZAJE DE LA ASIGNATURA:

Durante la actividad "Desarrollo de una Aplicación Web Interoperable utilizando el Framework Laravel", el estudiante ha aprendido a integrar sistemas de software mediante estándares y protocolos de comunicación. Ha diseñado arquitecturas interoperables, desarrollado y consumido APIs, y asegurado la coherencia y seguridad de los datos. Además, ha mejorado sus habilidades en pruebas de software, documentación, y trabajo en equipo, utilizando una metodología ágil para gestionar el proyecto de manera efectiva.

4. TEMA DE LA ACTIVIDAD DE LA INVESTIGACIÓN FORMATIVA:

Desarrollo de una Aplicación Web Interoperable utilizando el Framework Laravel consumo API con Spring.

5. OBJETIVOS DE LAS ACTIVIDADES:

- Implementar una arquitectura de software interoperable que permita la integración eficiente de diferentes sistemas y plataformas utilizando el framework Laravel.
- Desarrollar y consumir APIs y servicios web asegurando la coherencia, seguridad y correcta comunicación de datos entre sistemas heterogéneos.
- Aplicar una metodología ágil para gestionar el proyecto, mejorando las habilidades de trabajo en equipo, pruebas de software y documentación técnica.

6. FECHA DE LA EJECUCIÓN: 12 DE JUNIO DEL 2024

7. DESARROLLO DEL INFORME

7.1 Introducción.

El desarrollo de aplicaciones web que integren múltiples sistemas y plataformas es un desafío creciente en el campo de la ingeniería de software. En este contexto, la asignatura "Interoperabilidad de plataformas" se centra en enseñar las técnicas y herramientas necesarias para lograr dicha integración de manera eficiente y segura. Este informe presenta el proyecto final de la asignatura, cuyo objetivo fue desarrollar una aplicación web interoperable utilizando el framework Laravel.

Para llevar a cabo este proyecto, se utilizó XAMPP como entorno de desarrollo local, permitiendo la gestión del servidor Apache y la base de datos MySQL de manera eficiente. MySQL fue elegido por su robustez y capacidad de manejo de grandes volúmenes de datos, mientras que Visual Studio Code se empleó como el entorno de desarrollo integrado (IDE) debido a sus potentes características de edición de código y extensiones útiles para el desarrollo en Laravel.

Laravel, un framework PHP moderno, fue el núcleo del desarrollo de la aplicación. Laravel se seleccionó por su elegancia en la sintaxis, su potente sistema de ORM (Eloquent), y su robusto soporte para la creación de APIs y servicios web, esenciales para asegurar la interoperabilidad entre sistemas heterogéneos. El uso de Laravel permitió diseñar una arquitectura de software modular y escalable, facilitando la integración de diversas funcionalidades de manera coherente y segura.

La metodología ágil fue adoptada para gestionar el proyecto, lo que permitió una planificación flexible y adaptativa. El trabajo se organizó en sprints cortos, con reuniones diarias de seguimiento que aseguraron un progreso constante y la rápida resolución de problemas. Este enfoque también promovió la colaboración efectiva en el equipo, mejorando las habilidades de comunicación y trabajo en equipo del estudiante.

Durante el desarrollo del proyecto, se realizaron pruebas unitarias y de integración para garantizar la calidad del código y la funcionalidad correcta de la aplicación. Se prestó especial atención a la documentación técnica y de usuario, asegurando que el proyecto sea mantenible y comprensible para futuros desarrolladores.

En resumen, este proyecto proporcionó una experiencia integral en el desarrollo de aplicaciones web interoperables, combinando conocimientos teóricos y prácticos. El estudiante no solo adquirió habilidades técnicas en el uso de Laravel, MySQL, y Visual Studio Code, sino que también mejoró en áreas clave como la gestión de proyectos, pruebas de software, y documentación. Este informe detalla cada una de las fases del proyecto, los desafíos enfrentados, y los resultados obtenidos, ofreciendo una visión completa del proceso de desarrollo y los aprendizajes alcanzados.

7.2 Descripción de la metodología

Para desarrollar la aplicación web interoperable con el framework Laravel, se empleó una metodología ágil que facilitó una gestión eficiente del proyecto y una adaptación flexible a los cambios. La metodología se dividió en varias fases esenciales, cada una con objetivos específicos y tareas definidas, asegurando un progreso ordenado y efectivo. A continuación, se describen las principales fases de esta metodología

Planificación y Definición de Requisitos

Identificación de Requisitos: Se llevaron a cabo reuniones con los interesados para comprender y documentar las necesidades del proyecto. Se definieron los requisitos funcionales y no funcionales, estableciendo un alcance claro y realista.

Establecimiento de Objetivos: Se fijaron objetivos específicos y medibles para guiar el desarrollo, asegurando que todos los miembros del equipo tuvieran una visión común y clara del proyecto

Diseño de la Arquitectura del Sistema

Diseño de la Arquitectura: Se diseñó una arquitectura modular y escalable para la aplicación, considerando la estructura de la base de datos, la organización del código y la integración con otros sistemas. Este diseño facilitó la interoperabilidad y el mantenimiento a largo plazo.

Selección de Tecnologías: Se eligieron las herramientas y tecnologías adecuadas para el proyecto, incluyendo Laravel para el backend, Blade para el frontend, MySQL para la base de datos, y Visual Studio Code como entorno de desarrollo integrado (IDE).

Desarrollo Iterativo

Sprints de Desarrollo: El trabajo se organizó en sprints cortos y manejables, cada uno con un conjunto de tareas específicas. Se llevaron a cabo reuniones diarias (daily stand-ups) para monitorear el progreso, resolver problemas y ajustar el plan según fuera necesario.

Implementación de Funcionalidades: Se desarrollaron las funcionalidades de la aplicación siguiendo los requisitos definidos, aplicando las mejores prácticas de codificación y asegurando una integración fluida entre los componentes del sistema.

Integración y Pruebas:

Pruebas Unitarias y Funcionales: Se implementaron pruebas unitarias y funcionales para validar que cada componente de la aplicación funcionara correctamente. Estas pruebas ayudaron a detectar y corregir errores en etapas tempranas del desarrollo.

Pruebas de Integración: Se llevaron a cabo pruebas de integración para asegurar que los distintos componentes de la aplicación interactuaran de manera efectiva y coherente. Se utilizaron herramientas de automatización de pruebas para incrementar la cobertura y la eficiencia del proceso.

Despliegue y Documentación

Despliegue en Entornos de Prueba: La aplicación se desplegó en entornos de prueba para realizar pruebas de aceptación por parte de los usuarios finales. Esto permitió recoger feedback valioso y realizar ajustes antes del despliegue final.

Despliegue en Producción: Se realizó un despliegue planificado y controlado de la aplicación en el entorno de producción, minimizando los riesgos y asegurando una transición suave.

Documentación Completa: Se creó documentación técnica y de usuario detallada, incluyendo guías de instalación, configuración y uso. Esta documentación es esencial para el mantenimiento futuro y para la incorporación de nuevos desarrolladores al proyecto.

Mantenimiento y Mejora Continua

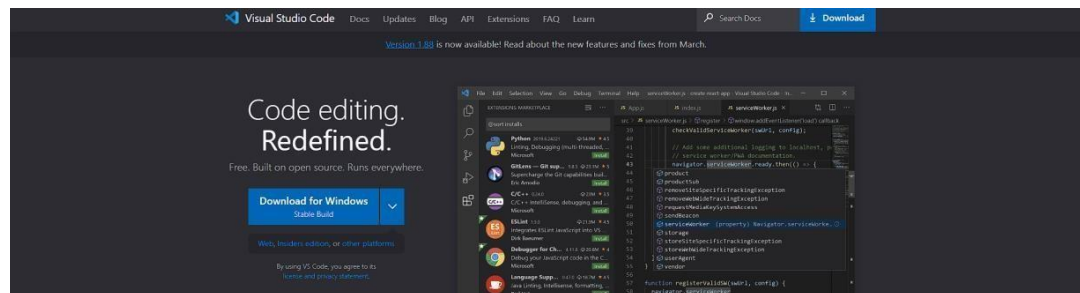
Monitoreo y Soporte: Una vez en producción, la aplicación fue monitoreada de manera continua para detectar y resolver cualquier problema rápidamente. Se proporcionó soporte técnico a los usuarios para asegurar una experiencia de uso satisfactoria.

Feedback y Mejoras: Se recopiló feedback de los usuarios y se analizaron métricas de rendimiento para identificar áreas de mejora. Este proceso de mejora continua permitió planificar y ejecutar actualizaciones y mejoras periódicas, asegurando la evolución constante de la aplicación.

Esta metodología ágil, con su enfoque iterativo y colaborativo, garantizó el desarrollo eficiente de una aplicación web interoperable, capaz de satisfacer las necesidades de los usuarios y de integrarse adecuadamente con otros sistemas

7.3 Descripción de las acciones realizadas:

Para instalar visual code, se debe ir a la página oficial, descargar el respectivo archivo y proceder a instalarlo



De la misma manera para instalar xampp server, se descarga de la página oficial y se procede a instalar, cabe recalcar que ambas instalaciones no son nada fuera de lo normal, a todo se le debe dar siguiente, siguiente y finish.



XAMPP es una distribución de Apache fácil de instalar que contiene MariaDB, PHP y Perl. Simplemente descarga y ejecuta el instalador. ¡Es así de fácil!

XAMPP para Windows 8.0.30, 8.1.25 & 8.2.12

Versión	Suma de comprobación	Tamaño
8.0.30 / PHP 8.0.30 ¿Qué está incluido?	md5 sha1	Descargar (64 bit) 144 Mb
8.1.25 / PHP 8.1.25 ¿Qué está incluido?	md5 sha1	Descargar (64 bit) 148 Mb
8.2.12 / PHP 8.2.12 ¿Qué está incluido?	md5 sha1	Descargar (64 bit) 149 Mb

[Requisitos](#) [Más Descargas »](#)

Documentación/FAQs

No hay un manual para XAMPP. Escribimos la documentación en forma de preguntas frecuentes (FAQs). ¿Tienes una pregunta que no está respondida? Prueba los Foros o Stack Overflow.

- [Linux Preguntas frecuentes](#)
- [Windows Preguntas frecuentes](#)
- [OS X Preguntas frecuentes](#)

Además, se debe descargar el composer para unas dependencias que el laravel requerirá para los proyectos

[Home](#) | [Getting Started](#) | [Download](#) | [Documentation](#) | [Browse Packages](#)

Download Composer Latest: v2.7.2

Windows Installer

The installer - which requires that you have PHP already installed - will download Composer for you and set up your PATH environment variable so you can simply call `composer` from any directory.

Download and run `Composer-Setup.exe` - it will install the latest composer version whenever it is executed.

Y de igual manera para su instalación es todo siguiente. Se puede comprobar la versión que se instaló con el `composer -v`

```
Símbolo del sistema
Microsoft Windows [Versión 10.0.22631.3296]
(c) Microsoft Corporation. Todos los derechos reservados.

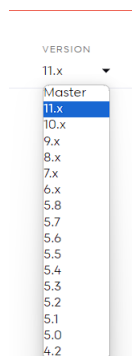
C:\Users\UserPRO>composer -v

Composer version 2.7.1 2024-02-09 15:26:28

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display help for the given command. When no command is given display help for the list
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi|--no-ansi         Force (or disable --no-ansi) ANSI output
  -n, --no-interaction      Do not ask any interactive question
  --profile                Display timing and memory usage information
  --no-plugins              Whether to disable plugins.
  --no-scripts             Skips the execution of all scripts defined in composer.json file.
  -d, --working-dir=WORKING-DIR If specified, use the given directory as working directory.
  --no-cache               Prevent use of the cache
  -v|vv|vvv, --verbose     Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and
                             3 for debug
```

Para la instalación de laravel se debe dirigirse a la página oficial de laravel y



seleccionar la versión que se va utilizar.

Una vez escogida la versión se debe proceder con su respectiva instalación, esto se lo realizara mediante la consola de comandos (CMD) con el comando que la página nos proporciona

Installation Via Composer

If your computer already has PHP and Composer installed, you may create a new Laravel project by using Composer directly. After the application has been created, you may start Laravel's local development server using the Artisan CLI's `serve` command:

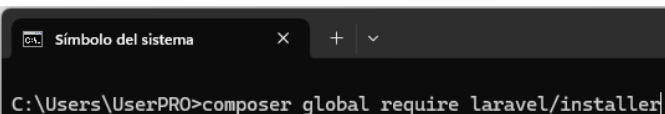
```
composer create-project laravel/laravel:^8.0 example-app  
  
cd example-app  
  
php artisan serve
```

The Laravel Installer

Or, you may install the Laravel Installer as a global Composer dependency:

```
composer global require laravel/installer  
  
laravel new example-app  
  
cd example-app  
  
php artisan serve
```

Make sure to place Composer's system-wide vendor bin directory in your `$PATH` so the `laravel` executable can be located by your system. This directory exists in different locations based on your operating system; however, some common locations include:



```
C:\Users\UserPRO>composer global require laravel/installer
```

Podemos comprobar la correcta instalación y la versión de la misma con el siguiente comando; laravel -v

```
Simbolo del sistema
C:\Users\UserPRO>laravel -v
Laravel Installer 5.7.1

Usage:
  command [options] [arguments]

Options:
  -h, --help            Display help for the given command. When no command is given display help for the list command
  -q, --quiet            Do not output any message
  -V, --version          Display this application version
  --ansi|--no-ansi      Force (or disable --no-ansi) ANSI output
  -n, --no-interaction  Do not ask any interactive question
  -vv|vvv, --verbose    Increase the verbosity of messages: 1 for normal output, 2 for more verbose output and 3 for debug

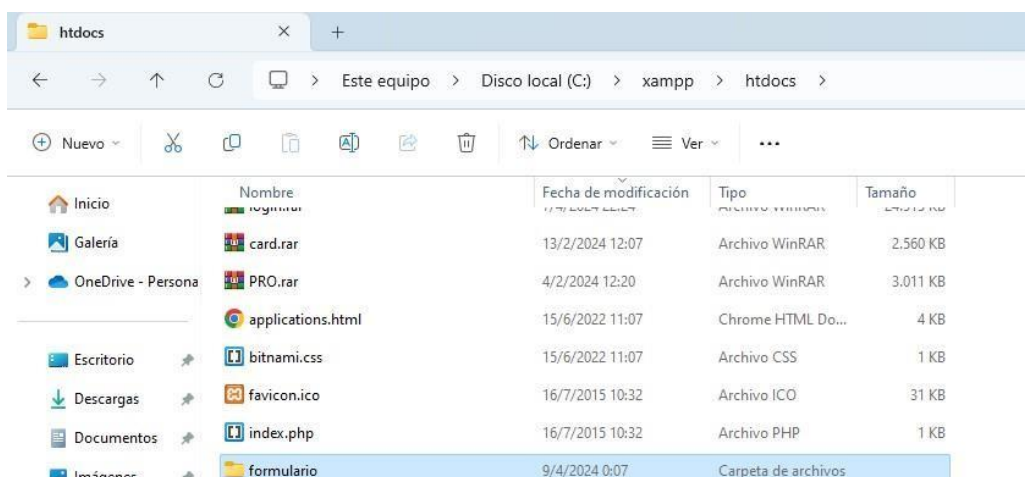
Available commands:
  completion  Dump the shell completion script
  help        Display help for a command
  list        List commands
  new         Create a new Laravel application

C:\Users\UserPRO>
```

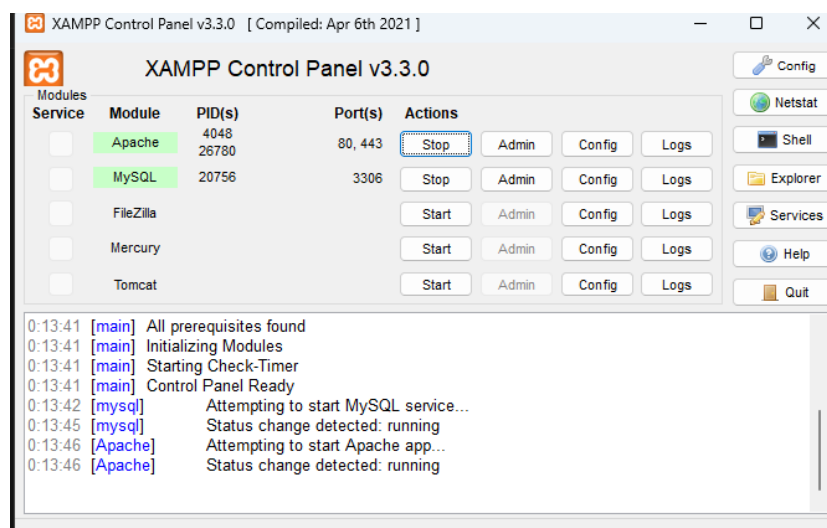
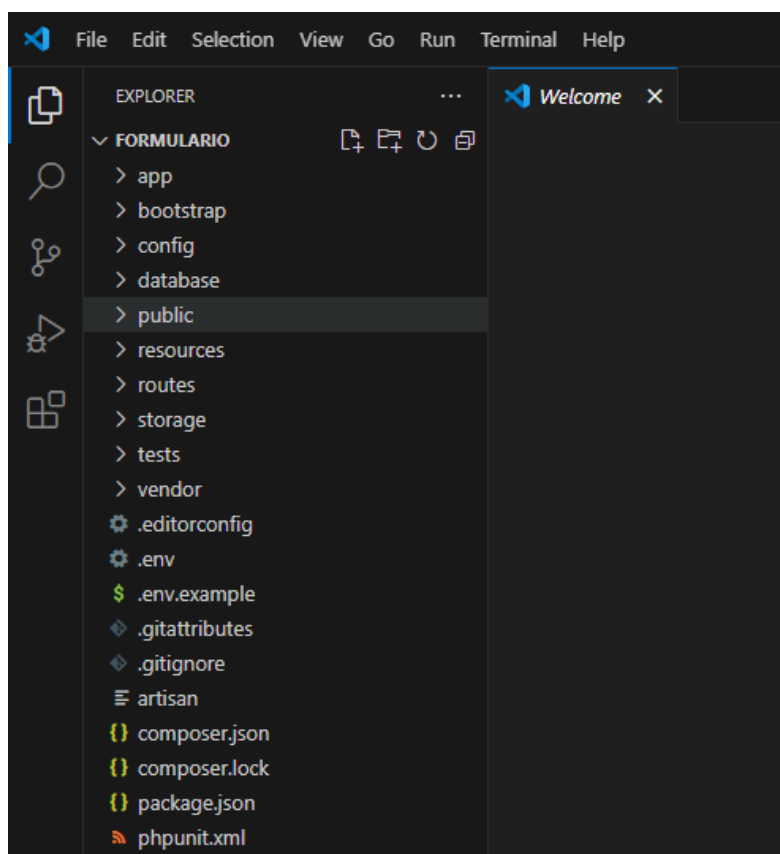
Una vez instalado el laravel y todas sus dependencias se procedera a crear un proyecto para proseguir a realizar un formulario, cabe recalcar que para crear un proyecto se debe estar en la ruta de htdocs desde el cmd.

```
C:\xampp\htdocs>laravel new formulario
```

Y como podemos ver se nos crea una carpeta con todo lo correspondiente para el proyecto



En primer lugar, se debe abrir la carpeta que se creó al iniciar el proyecto con el visual studio code, xampp



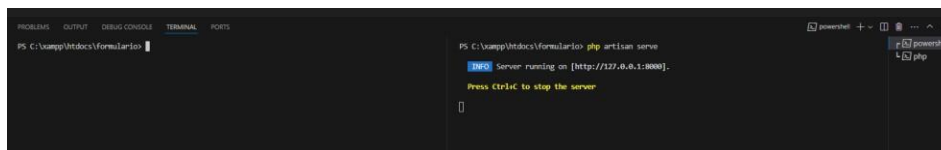
desplegar el servidor local con el siguiente comando en una terminal de vs; php artisan serve

```
PS C:\xampp\htdocs\formulario> php artisan serve
```

Abrir otra terminal para poder trabajar



Abrir otra terminal para poder trabajar



Instalar las dependencias para usar Bootstrap

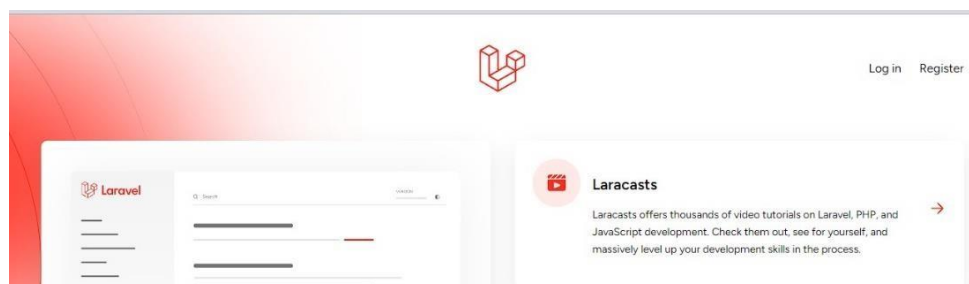
```
PS C:\xampp\htdocs\formulario> composer require laravel/ui  
/composer.json has been updated
```

```
PS C:\xampp\htdocs\formulario> php artisan ui bootstrap --auth
```

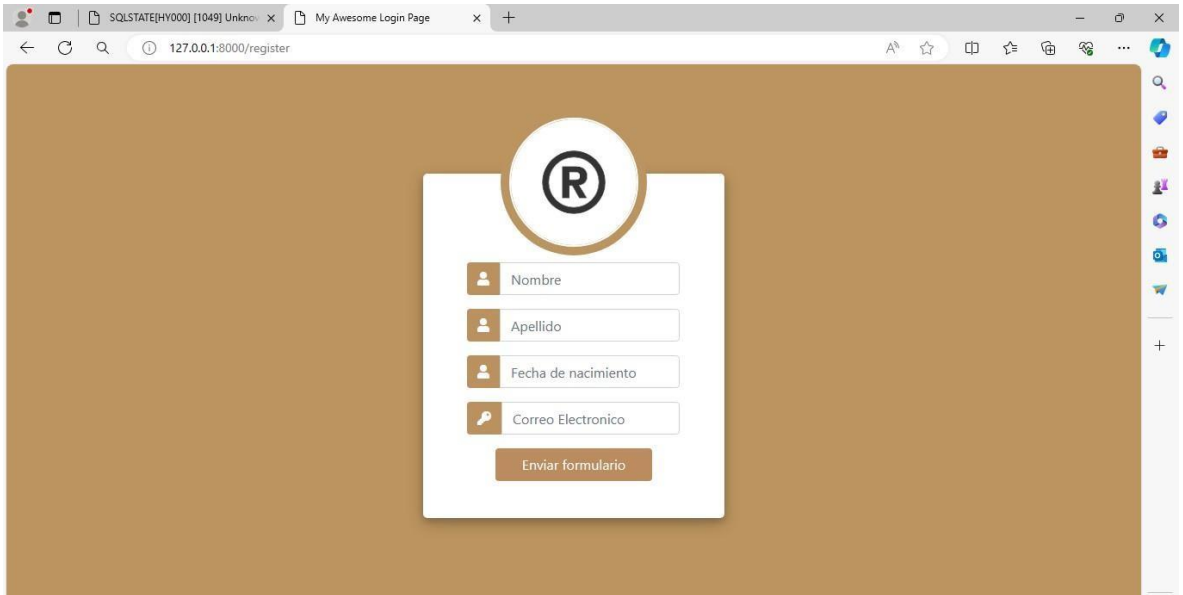
```
PS C:\xampp\htdocs\formulario> php artisan ui bootstrap --auth
```

```
PS C:\xampp\htdocs\formulario> php artisan migrate
```

Una vez realizado lo anterior comprobamos que se agregue el botón de registro en nuestra página principal



Realizamos los cambios necesarios en la estructura del register y sus css correspondientes para obtener un formulario con lo pedido anteriormente.



Una vez que contamos con la interfaz de inicio, planeamos reforzar la seguridad en el proceso de inicio de sesión al requerir la verificación del correo electrónico para confirmar el acceso. Para implementar esto, vamos a introducir las siguientes líneas de código en nuestro proyecto.

```
55 MAIL_MAILER=smtp
56 MAIL_HOST=sandbox.smtp.mailtrap.io
57 MAIL_PORT=2525
58 MAIL_USERNAME=41e40091833dcb
59 MAIL_PASSWORD=67505f3daddddb
60 MAIL_FROM_ADDRESS="leidy.tene@unach.edu.ec"
61 MAIL_FROM_NAME="${APP_NAME}"
```

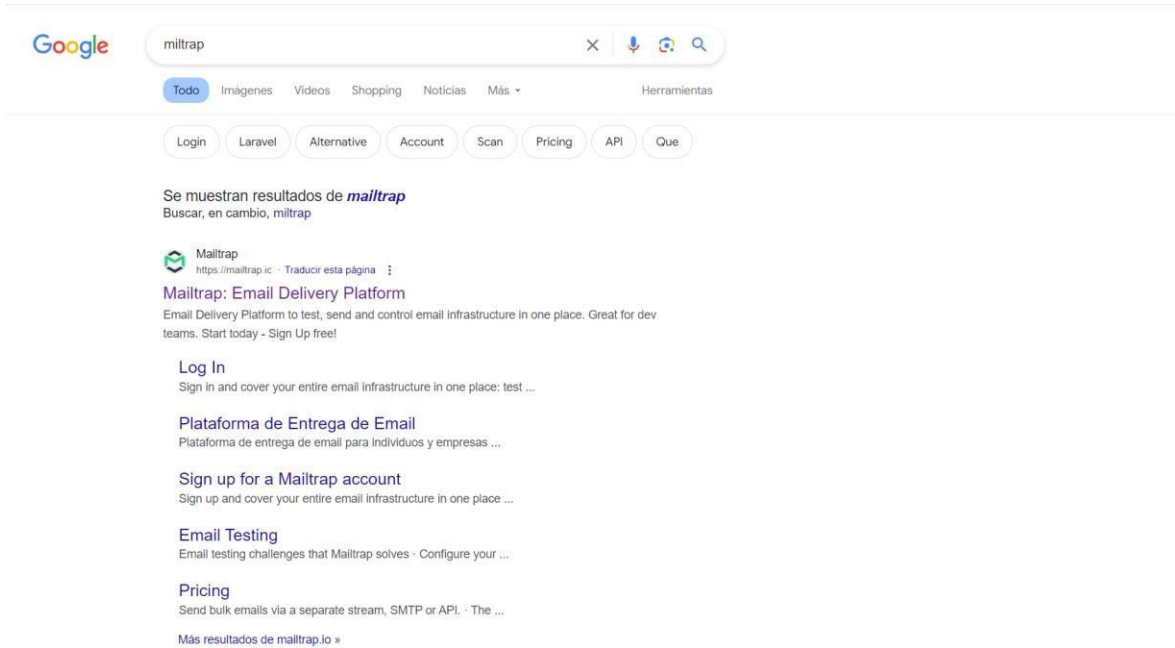
Incorporamos la validación del correo electrónico al perfil del usuario.

```
app > Models > User.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Contracts\Auth\MustVerifyEmail;
6  use Illuminate\Database\Eloquent\Factories\HasFactory;
7  use Illuminate\Foundation\Auth\User as Authenticatable;
8  use Illuminate\Notifications\Notifiable;
9  use Laravel\Sanctum\HasApiTokens;
10
11 class User extends Authenticatable implements MustVerifyEmail
```

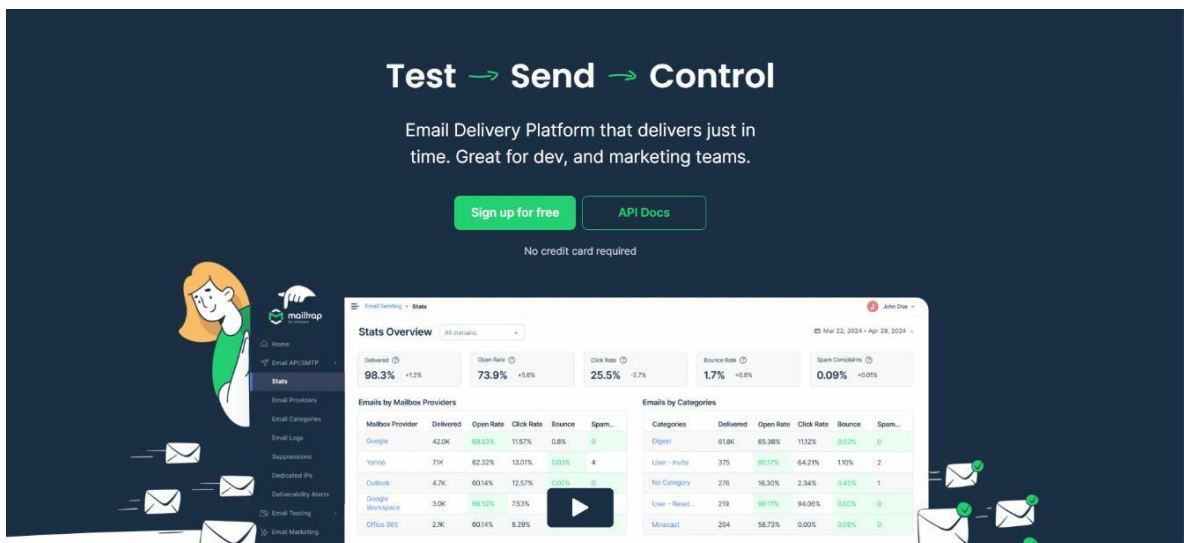
Además, desmarcamos la línea de código específica.

```
145
146     'features' => [
147         Features::registration(),
148         Features::resetPasswords(),
149         Features::emailVerification(),
150         Features::updateProfileInformation(),
151         Features::updatePasswords(),
152         Features::twoFactorAuthentication([
153             'confirm' => true,
154             'confirmPassword' => true,
155             // 'window' => 0,
156         ]),
157     ],
158
159 ];
```


Ahora, vamos a configurar nuestro servidor de correo electrónico.

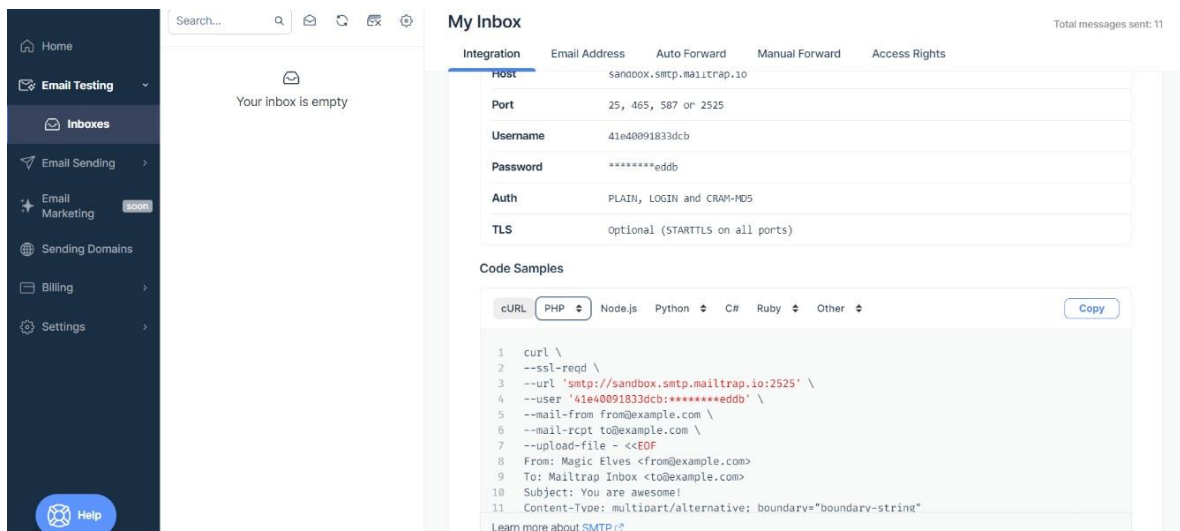


Nos registramos de forma gratuita



Ahora configuremos nuestro servidor de emails

Y procedemos a copiar las credenciales para enlazar nuestro proyecto al servidor.



The screenshot shows the Mailtrap web interface. On the left is a sidebar with navigation links: Home, Email Testing, Inboxes, Email Sending, Email Marketing, Sending Domains, Billing, and Settings. The main area is titled 'My Inbox' and shows an empty inbox. To the right, the 'Integration' tab is active, displaying SMTP configuration details:

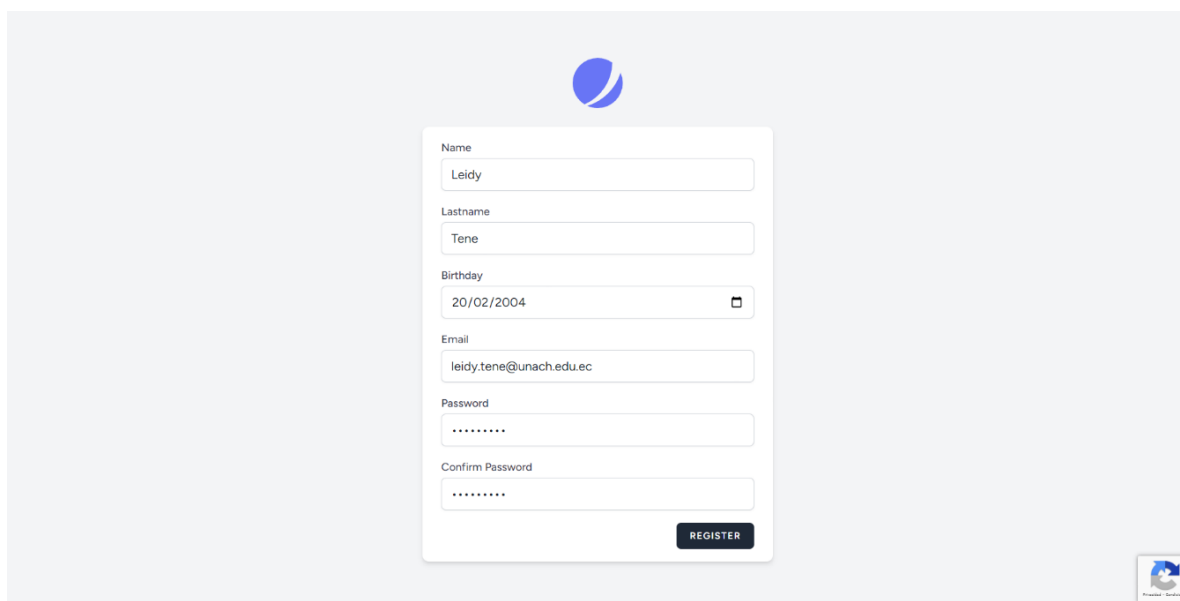
Integration	Email Address	Auto Forward	Manual Forward	Access Rights
Host	sandbox.smtp.mailtrap.io			
Port	25, 465, 587 or 2525			
Username	41e40891833dcb			
Password	*****eddb			
Auth	PLAIN, LOGIN and CRAM-MD5			
TLS	Optional (STARTTLS on all ports)			

Below the settings, the 'Code Samples' section shows a cURL command for sending an email:

```
1 curl \
2 --ssl-reqd \
3 --url 'smtp://sandbox.smtp.mailtrap.io:2525' \
4 --user '41e40891833dcb:*****eddb' \
5 --mail-from from@example.com \
6 --mail-rcpt to@example.com \
7 --upload-file - <<EOF
8 From: Magic Elves <from@example.com>
9 To: Mailtrap Inbox <to@example.com>
10 Subject: You are awesome!
11 Content-Type: multipart/alternative; boundary="boundary-string"
```

A 'Copy' button is available next to the code sample.

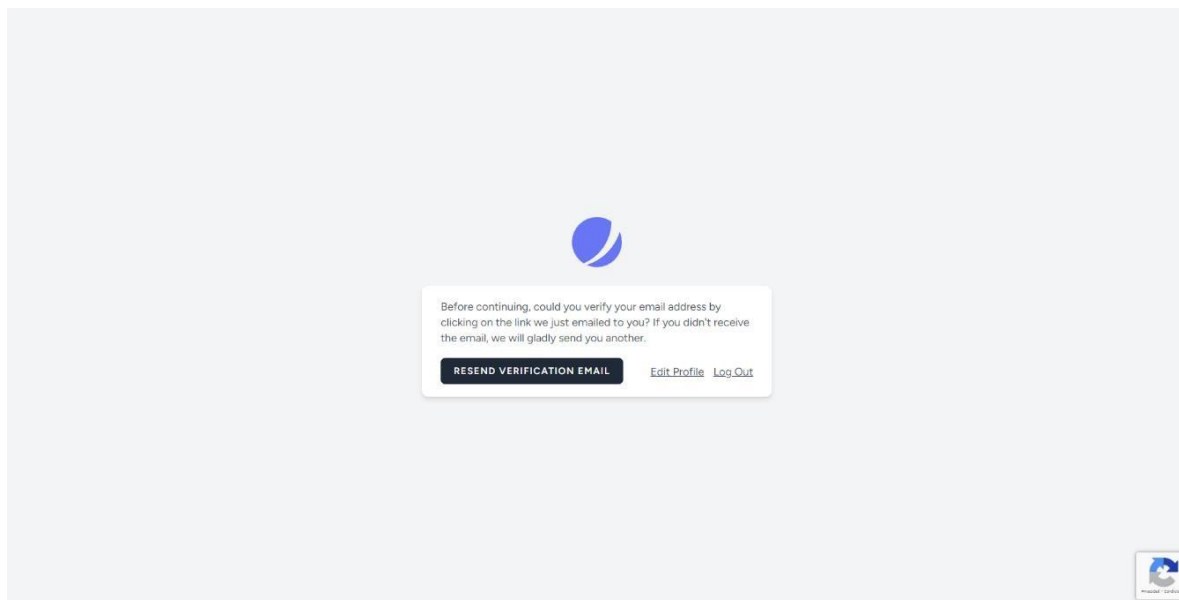
Ahora nos registraremos para verificar la funcionalidad



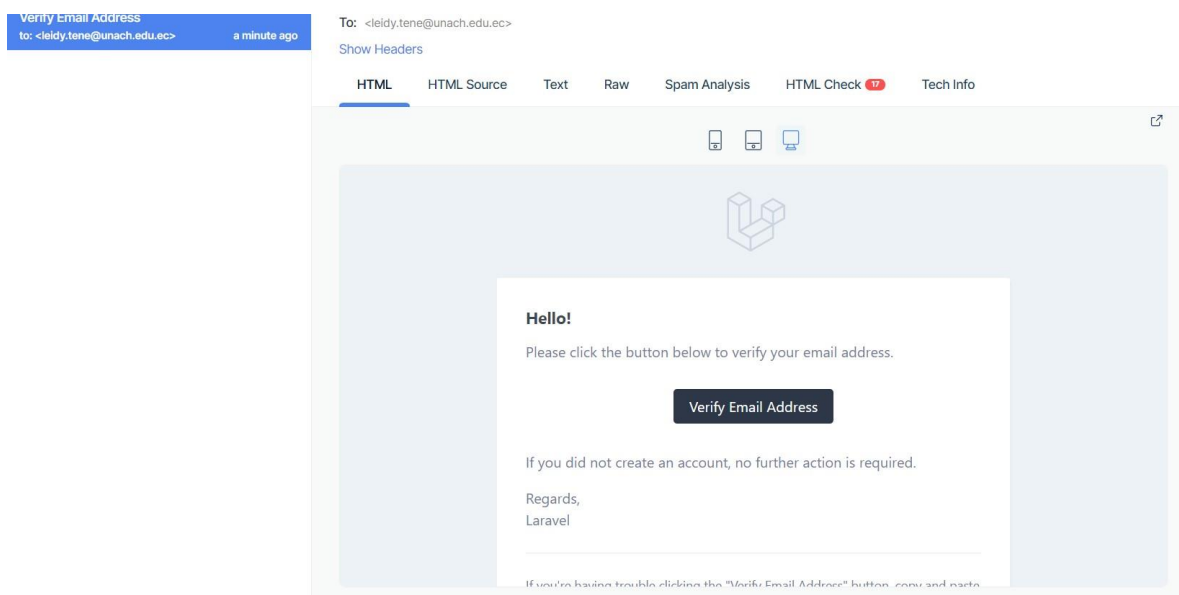
The screenshot shows a registration form for Mailtrap. The form is titled 'Name' and contains the following fields:

- Name: Leidy
- Lastname: Tene
- Birthday: 20/02/2004
- Email: leidy.tene@unach.edu.ec
- Password:
- Confirm Password:

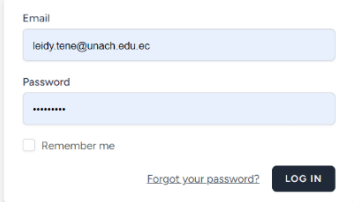
A 'REGISTER' button is located at the bottom right of the form.



Verificamos que al correo llegara el mensaje de confirmación



Ahora, ingresamos a nuestra página sin ningún problema



Logo


Email


leidy.tene@unach.edu.ec

Password

☐ Remember me


[Forgot your password?](#) [LOG IN](#)



 Dashboard


Leidy ▾

Dashboard

 **Laravel Jetstream**


Welcome to your Jetstream application!

Laravel Jetstream provides a beautiful, robust starting point for your next Laravel application. Laravel is designed to help you build your application using a development environment that is simple, powerful, and enjoyable. We believe you should love expressing your creativity through programming, so we have spent time carefully crafting the Laravel ecosystem to be a breath of fresh air. We hope you love it.

 **Documentation**


Laravel has wonderful documentation covering every aspect of the framework. Whether you're new to the framework or have previous experience, we recommend reading all of the documentation from beginning to end.

[Explore the documentation →](#)


 **Laracasts**

Laracasts offers thousands of video tutorials on Laravel, PHP, and JavaScript development. Check them out, see for yourself, and massively level up your development skills in the process.

[Start watching Laracasts →](#)

 **Tailwind**

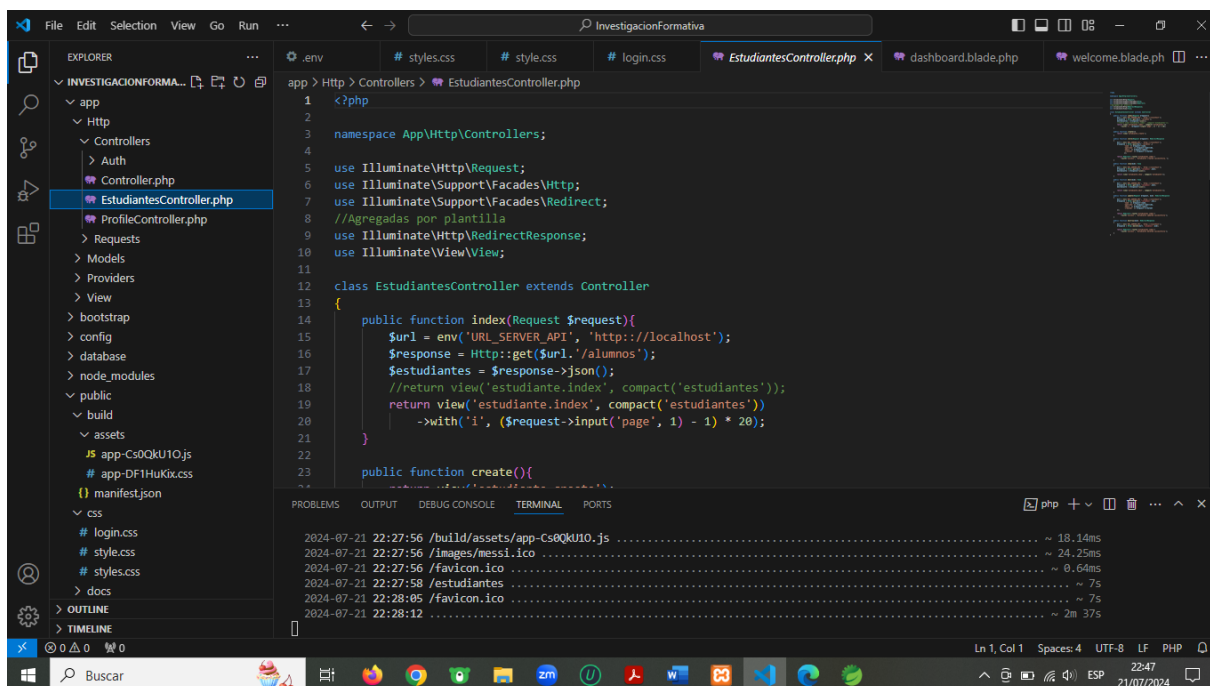
Laravel Jetstream is built with Tailwind, an amazing utility first CSS framework that doesn't get in your way. You'll be amazed how easily you can build and maintain fresh,

 **Authentication**

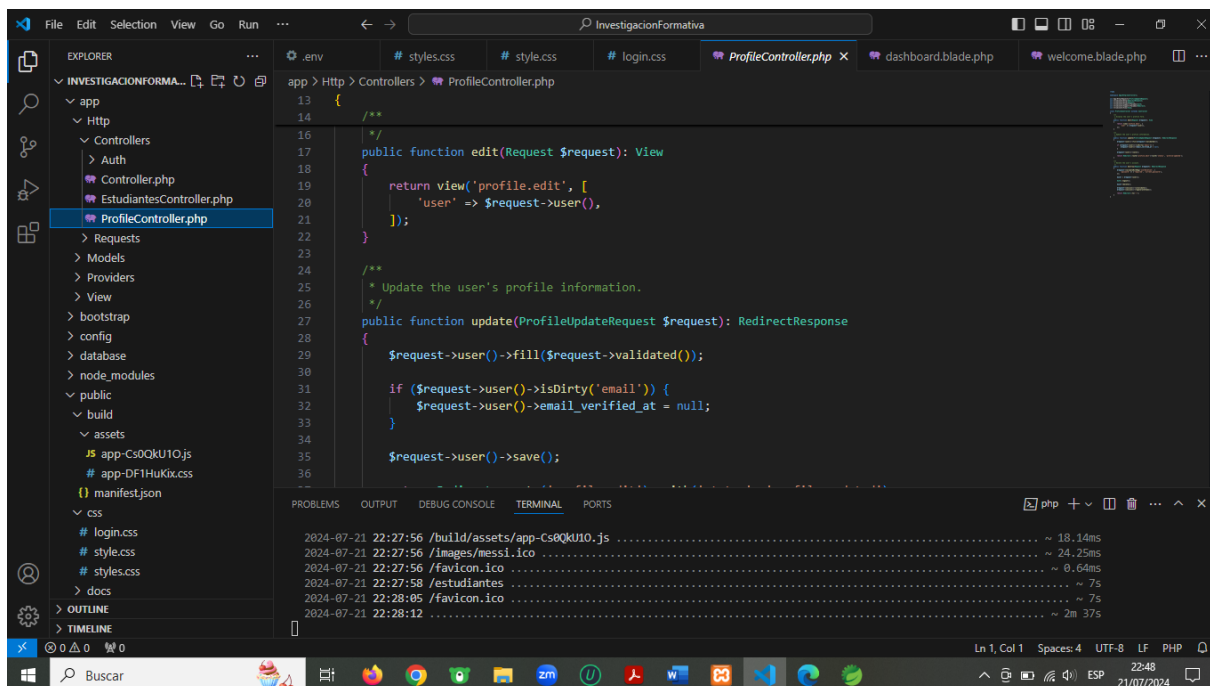
Authentication and registration views are included with Laravel Jetstream, as well as support for user email verification and resetting forgotten passwords. So, you're free to

Crear una API en Spring

Estudiante controller

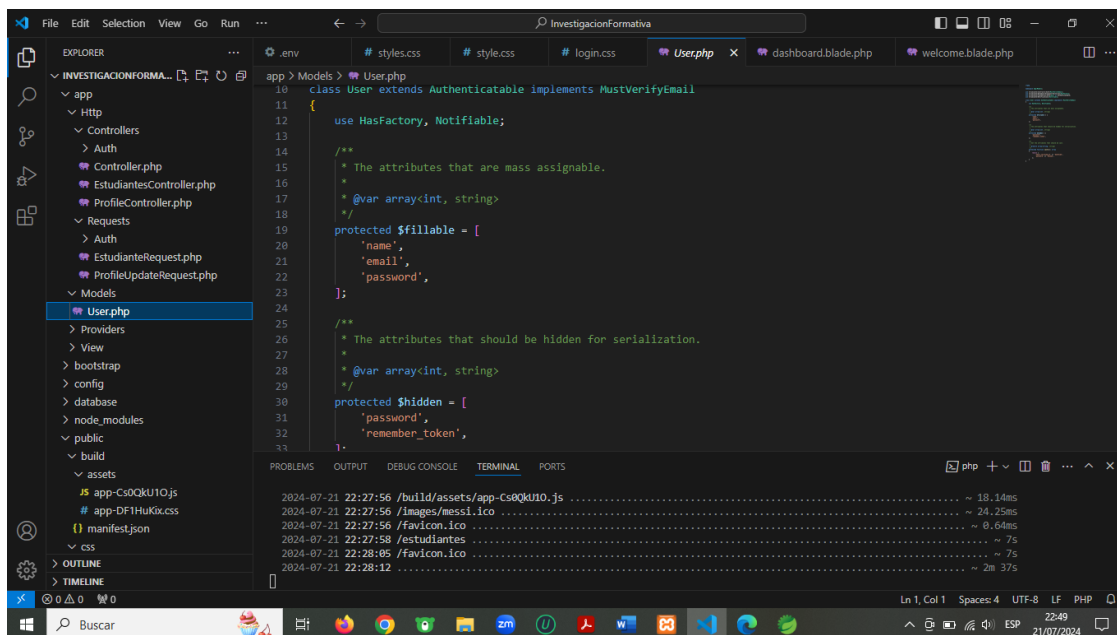


```
<?php
1
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Http;
7 use Illuminate\Support\Facades\Redirect;
8 //Agregadas por plantilla
9 use Illuminate\Http\RedirectResponse;
10 use Illuminate\View\View;
11
12 class EstudiantesController extends Controller
13 {
14     public function index(Request $request){
15         $url = env('URL_SERVER_API', 'http://localhost');
16         $response = Http::get($url, '/alumnos');
17         $estudiantes = $response->json();
18         //return view('estudiante.index', compact('estudiantes'));
19         return view('estudiante.index', compact('estudiantes'))
20             ->with('1', ($request->input('page', 1) - 1) * 20);
21     }
22
23     public function create(){
24
25     }
26 }
```



```
13 {
14     /**
15      *
16      */
17     public function edit(Request $request): View
18     {
19         return view('profile.edit', [
20             'user' => $request->user(),
21         ]);
22     }
23
24     /**
25      * Update the user's profile information.
26      */
27     public function update(ProfileUpdateRequest $request): RedirectResponse
28     {
29         $request->user()->fill($request->validated());
30
31         if ($request->user()->isDirty('email')) {
32             $request->user()->email_verified_at = null;
33         }
34
35         $request->user()->save();
36     }
37 }
```

Models users



```

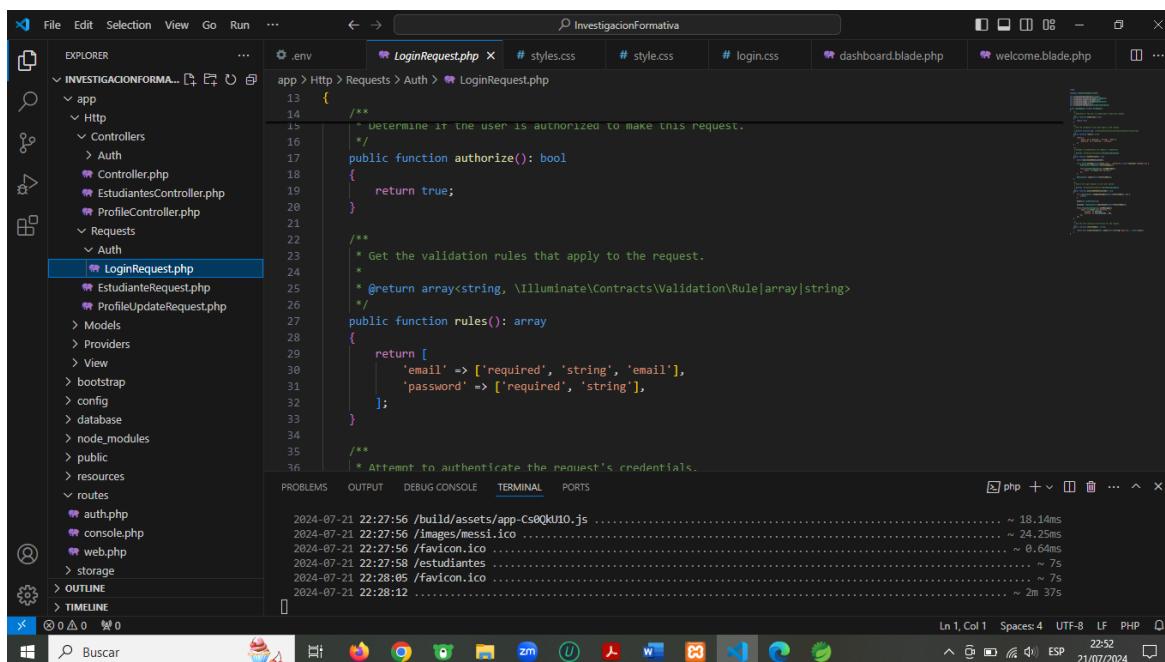
class User extends Authenticatable implements MustVerifyEmail
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array<int, string>
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];

    /**
     * The attributes that should be hidden for serialization.
     *
     * @var array<int, string>
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];
}

```

Login request



```

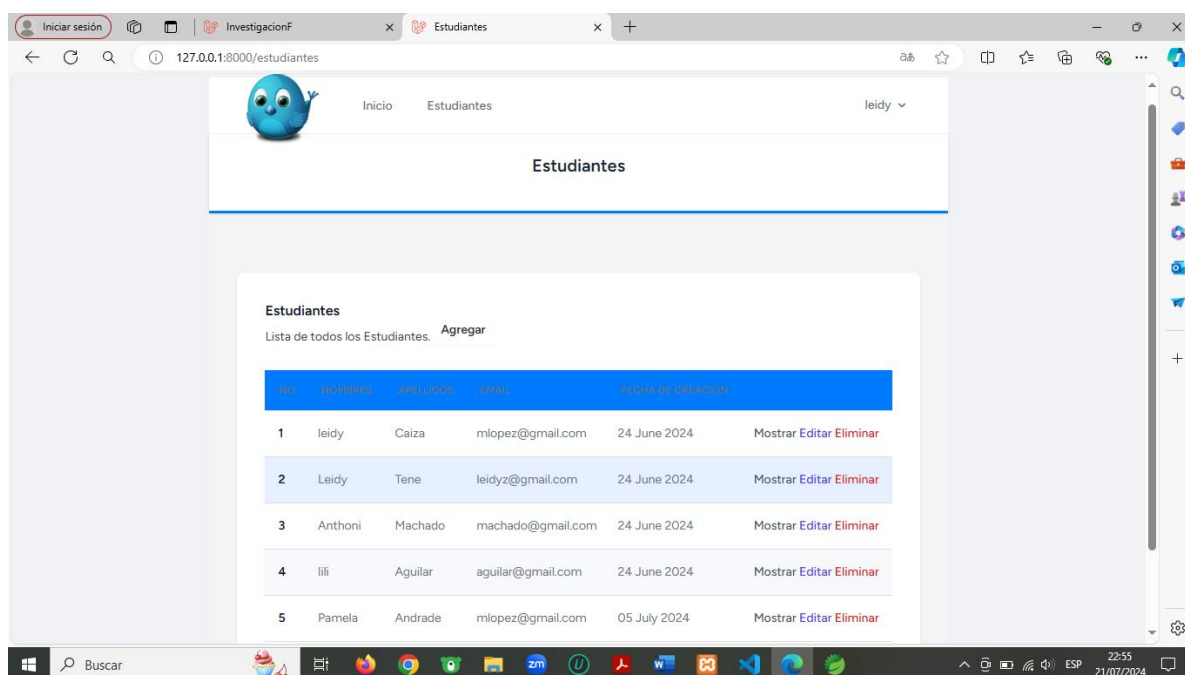
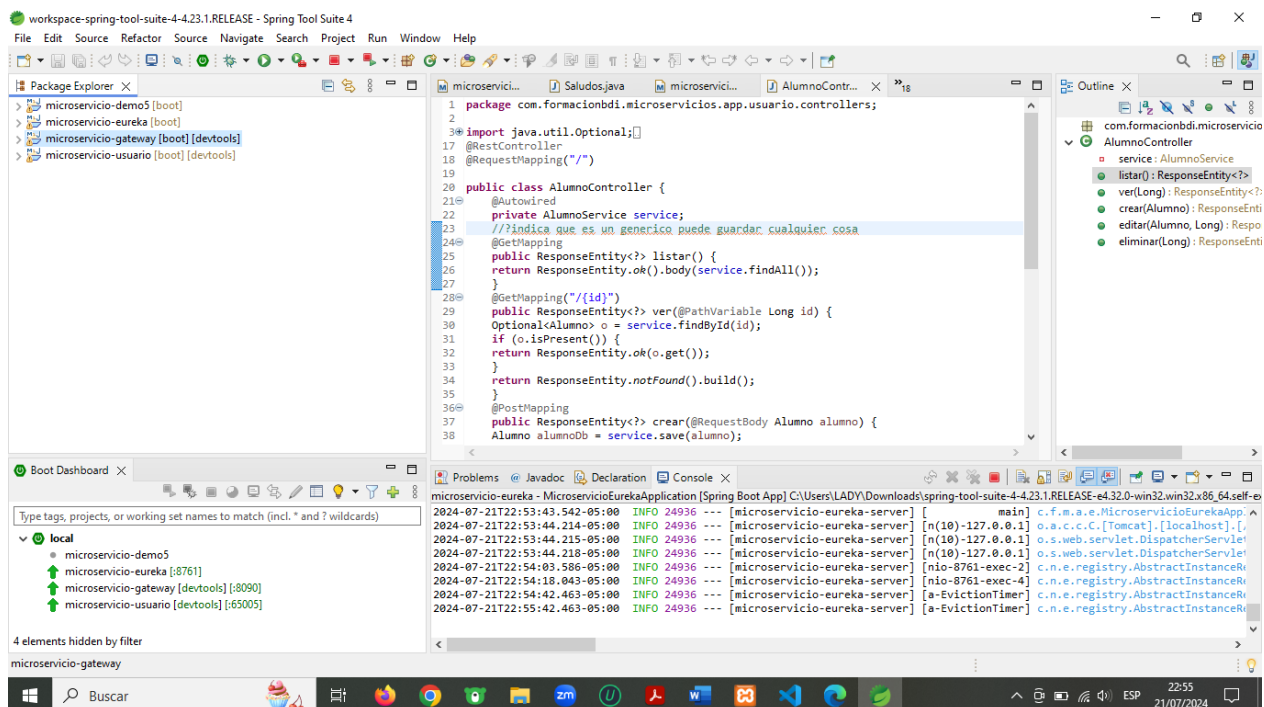
class LoginRequest
{
    /**
     * Determine if the user is authorized to make this request.
     */
    public function authorize(): bool
    {
        return true;
    }

    /**
     * Get the validation rules that apply to the request.
     *
     * @return array<string, Illuminate\Contracts\Validation\Rule|array|string>
     */
    public function rules(): array
    {
        return [
            'email' => ['required', 'string', 'email'],
            'password' => ['required', 'string'],
        ];
    }

    /**
     * Attempt to authenticate the request's credentials.
     */
}

```

Ejecución de los servicios



7.3.1 MARCO TEÓRICO

La interoperabilidad en sistemas de software se refiere a la capacidad de diferentes sistemas y aplicaciones para intercambiar información y utilizarla de manera efectiva. Esta capacidad es esencial para garantizar que los sistemas diversos puedan trabajar juntos sin problemas, lo cual es crucial en entornos empresariales complejos [1]. La interoperabilidad permite que las organizaciones aprovechen sus sistemas existentes y los integren con nuevas aplicaciones, facilitando la colaboración y el intercambio de información.

Framework Laravel

Laravel es un framework de PHP diseñado para el desarrollo de aplicaciones web que sigue el patrón de arquitectura Modelo-Vista-Controlador (MVC). Es conocido por su sintaxis elegante, su potente ORM (Eloquent), y su capacidad para simplificar tareas comunes como la autenticación, la gestión de sesiones y la creación de APIs [2].

Características Clave de Laravel

Eloquent ORM: Eloquent proporciona una forma sencilla y expresiva de interactuar con la base de datos, permitiendo la implementación de relaciones complejas y consultas avanzadas sin necesidad de escribir SQL explícito [2].

Blade Templating Engine: Blade permite la creación de vistas reutilizables y la separación lógica del diseño HTML, facilitando el desarrollo y mantenimiento del frontend [2].

Middleware: Middleware en Laravel ofrece una manera conveniente de filtrar HTTP requests y añadir funcionalidades como autenticación y manejo de sesiones [2].

Herramientas de Desarrollo

XAMPP

XAMPP es una distribución de Apache que incluye MySQL, PHP y Perl. Es una solución completa para el desarrollo local, permitiendo a los desarrolladores crear y probar sus aplicaciones en un entorno controlado antes de su despliegue [3].

MySQL

MySQL es un sistema de gestión de bases de datos relacional ampliamente utilizado en el

desarrollo web. Es conocido por su eficiencia, robustez y capacidad para manejar grandes volúmenes de datos, siendo una elección común para aplicaciones desarrolladas con PHP y Laravel [4].

Visual Studio Code

Visual Studio Code (VS Code) es un entorno de desarrollo integrado (IDE) popular entre los desarrolladores web debido a su flexibilidad, extensibilidad y soporte para múltiples lenguajes de programación. VS Code facilita el desarrollo con Laravel mediante extensiones específicas que mejoran la productividad y eficiencia [5].

Metodología Ágil

La metodología ágil es un enfoque iterativo e incremental para la gestión de proyectos de software que se centra en la colaboración, la flexibilidad y la entrega continua de valor al cliente [6]. Este enfoque se basa en ciclos de desarrollo cortos y reuniones frecuentes para revisar y ajustar los planes de acuerdo con el progreso y los cambios en los requisitos.

Fases de la Metodología Ágil

Planificación y Recolección de Requisitos: Esta fase inicial involucra la identificación y documentación de las necesidades del usuario, estableciendo un alcance claro y objetivos específicos para el proyecto. Esto asegura que todos los involucrados comprendan los objetivos y expectativas del proyecto.

Diseño de la Arquitectura: Se crea una arquitectura modular y escalable para la aplicación, considerando la estructura de la base de datos, la organización del código y la integración con otros sistemas. Este diseño facilita la interoperabilidad y el mantenimiento a largo plazo.

Desarrollo Iterativo: El proyecto se divide en sprints cortos y manejables, cada uno con un conjunto de tareas específicas. Se llevan a cabo reuniones diarias (daily stand-ups) para monitorear el progreso, resolver problemas y ajustar el plan según sea necesario.

Pruebas e Integración: Se realizan pruebas unitarias, funcionales e integrales para validar que cada componente de la aplicación funcione correctamente y que los diferentes componentes interactúen de manera efectiva. Se utilizan herramientas de automatización de pruebas para incrementar la cobertura y la eficiencia del proceso.

Despliegue y Documentación: La aplicación se despliega en entornos de prueba para

realizar pruebas de aceptación por parte de los usuarios finales. Una vez validadas las pruebas, se realiza el despliegue controlado en el entorno de producción. Se crea documentación técnica y de usuario detallada, incluyendo guías de instalación, configuración y uso.

Mantenimiento y Mejora Continua: Una vez en producción, la aplicación es monitoreada de manera continua para detectar y resolver problemas rápidamente. Se proporciona soporte técnico a los usuarios y se recopila feedback para identificar áreas de mejora. Este proceso de mejora continua permite planificar y ejecutar actualizaciones y mejoras periódicas, asegurando la evolución constante de la aplicación.

El uso de una metodología ágil permitió gestionar de manera efectiva el desarrollo de la aplicación web interoperable, asegurando una entrega continua de valor y una respuesta rápida a los cambios.

Spring Framework

Historia y Evolución de Spring

Spring Framework fue creado por Rod Johnson en 2003. Su objetivo principal era simplificar el desarrollo de aplicaciones Java y proporcionar una alternativa ligera a los modelos Enterprise JavaBeans (EJB). Desde su creación, Spring ha evolucionado para incluir una amplia gama de funcionalidades que abarcan desde la inyección de dependencias hasta el desarrollo de microservicios y aplicaciones en la nube. [1]

Componentes Principales de Spring

- Spring Framework se compone de varios módulos que pueden ser utilizados de forma independiente o conjunta para desarrollar aplicaciones robustas y escalables.
- Core Container: El núcleo de Spring que proporciona funcionalidades básicas como la Inversión de Control (IoC) y la Inyección de Dependencias (DI).
- Spring AOP (Aspect-Oriented Programming): Permite la implementación de preocupaciones transversales como la gestión de transacciones, la seguridad y el registro de logs.



- Spring Data: Facilita el acceso y la manipulación de datos a través de tecnologías como JDBC, JPA, MongoDB, y más.
- Spring MVC: Proporciona un framework para el desarrollo de aplicaciones web siguiendo el patrón Modelo-Vista-Controlador.
- Spring Security: Proporciona autenticación y autorización robustas para aplicaciones Java.
- Spring Boot: Simplifica la configuración y el despliegue de aplicaciones Spring, permitiendo un desarrollo más rápido y una configuración mínima.
- Spring Cloud: Proporciona herramientas para el desarrollo de aplicaciones distribuidas y basadas en microservicios.
- Spring Integration: Facilita la integración de aplicaciones mediante patrones de integración empresarial.

¿Qué es Spring?

Spring es un framework de código abierto para la plataforma Java, creado inicialmente por Rod Johnson. Es conocido por su enfoque modular y su capacidad para facilitar el desarrollo de aplicaciones empresariales robustas. Uno de los componentes más utilizados de Spring es Spring Boot, que simplifica la configuración y el despliegue de aplicaciones Spring. [2]

Características de Spring

- **Inyección de Dependencias (DI):** Facilita la gestión de dependencias y aumenta la modularidad.
- **Aspect-Oriented Programming (AOP):** Permite separar preocupaciones transversales como la seguridad y el registro.

- **Data Access:** Herramientas para interactuar con bases de datos, incluyendo JDBC y JPA.
- **MVC Framework:** Facilita la creación de aplicaciones web siguiendo el patrón MVC.
- **Spring Security:** Proporciona autenticación y autorización robustas.

Características de Spring Boot:

- **Configuración Automática:** Detecta las dependencias presentes en el proyecto y configura automáticamente los componentes necesarios.
- **Servidor Integrado:** Permite ejecutar aplicaciones como aplicaciones independientes sin necesidad de servidores externos.
- **Starter POMs:** Proporciona un conjunto de dependencias agrupadas que simplifican la configuración del proyecto.

Ventajas de Spring

- **Actuadores:** Proporciona endpoints para monitorear y gestionar la aplicación.
- **Modularidad:** Permite construir aplicaciones de manera modular y escalable.
- **Flexibilidad:** Soporta diversas configuraciones y arquitecturas.
- **Ecosistema Ampliado:** Amplia gama de proyectos complementarios como Spring Cloud y Spring Data.

Interoperabilidad entre Laravel y Spring

La interoperabilidad entre Laravel y Spring se logra mediante el uso de APIs RESTful. Las APIs permiten que diferentes sistemas intercambien datos y funciones de manera estándar y comprensible. Aquí se presentan algunos conceptos clave para lograr esta interoperabilidad:

APIs RESTful

REST (Representational State Transfer): Un estilo de arquitectura que utiliza métodos HTTP estándar (GET, POST, PUT, DELETE) para la comunicación entre sistemas.

JSON (JavaScript Object Notation): Un formato de intercambio de datos ligero y fácil de

leer para humanos y máquinas.

Endpoints: Puntos de acceso definidos en una API para realizar operaciones específicas.

Consumo de APIs con Laravel

Laravel proporciona herramientas como Guzzle HTTP Client para consumir APIs externas.

Con Guzzle, se pueden enviar solicitudes HTTP y manejar las respuestas de manera eficiente. [3]

Creación de APIs con Spring

Spring Boot facilita la creación de APIs RESTful mediante anotaciones como `@RestController` y `@RequestMapping`. Estas anotaciones permiten definir endpoints y manejar solicitudes HTTP de manera intuitiva.

7.4 Resultados

Mejora en la Eficiencia del Desarrollo: La adopción de metodologías ágiles y el uso de herramientas como Laravel y Visual Studio Code tienden a acelerar el proceso de desarrollo. Los ciclos cortos de desarrollo, las reuniones diarias para revisar el progreso y la automatización de tareas repetitivas contribuyen a una mayor eficiencia en la producción de software.

Mayor Calidad del Producto: La aplicación de pruebas unitarias y funcionales durante el desarrollo, junto con la integración continua, tiende a mejorar la calidad del producto final. Esto reduce la cantidad de errores y bugs en el software, lo que se traduce en una mejor experiencia para el usuario y una reducción en los costos asociados a la corrección de errores después del despliegue.

Facilitación de la Interoperabilidad: Al diseñar la arquitectura de la aplicación con interoperabilidad en mente y utilizando un framework como Laravel, se facilita la integración con otros sistemas. Esto permite que la aplicación se comuniquen de manera eficiente con sistemas externos, lo que puede ser crucial en entornos empresariales donde la interoperabilidad es fundamental.

Retroalimentación Continua y Mejora Iterativa: La adopción de metodologías ágiles fomenta la retroalimentación continua por parte de los usuarios y el equipo de desarrollo.

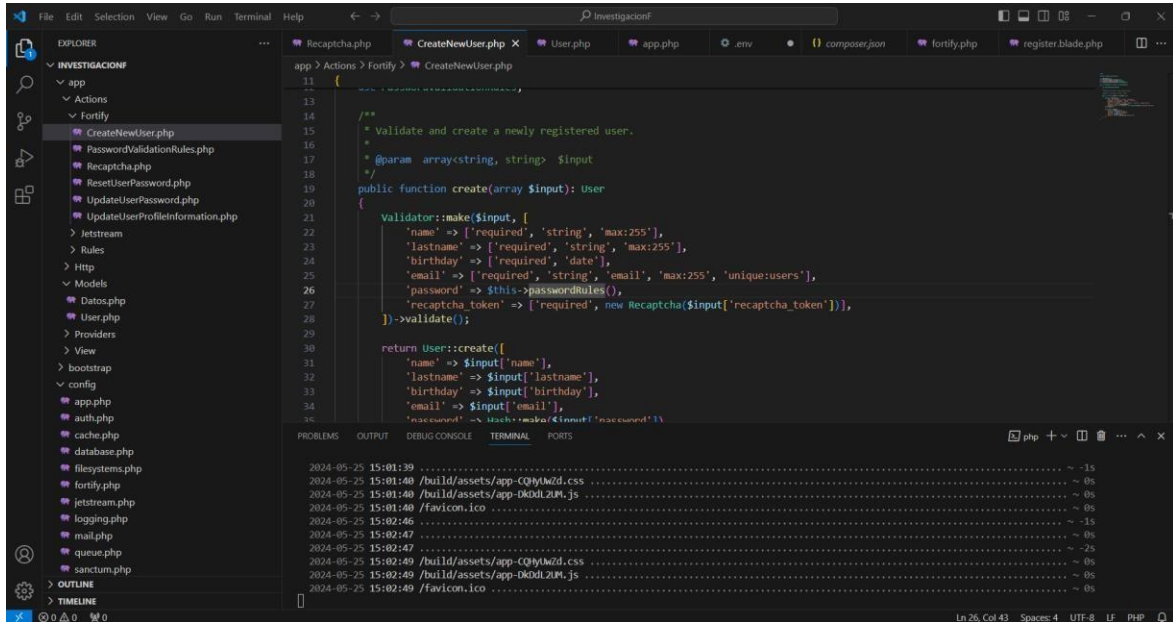
Esto permite identificar áreas de mejora y realizar ajustes en el producto de manera iterativa. La capacidad de respuesta a los cambios y la mejora continua son aspectos clave en el desarrollo exitoso de software.

Despliegue y Mantenimiento Simplificados: La documentación detallada y la planificación cuidadosa del despliegue permiten una transición suave del desarrollo a la producción. Además, el monitoreo continuo y el soporte técnico garantizan un funcionamiento óptimo de la aplicación una vez en producción, minimizando el tiempo de inactividad y los problemas de rendimiento.

7.5 Bibliografía

- [1] W. Guédria, "Interoperability assessment: a systematic literature review," *Computers in Industry*, vol. 65, no. 7, pp. 874-885, 2014. doi: 10.1016/j.compind.2014.04.002.
- [2] M. Stauffer, *Laravel: Up & Running: A Framework for Building Modern PHP Apps*. O'Reilly Media, 2019.
- [3] A. Tariq, *Mastering PHP 7*. Packt Publishing Ltd, 2015.
- [4] M. Widenius and D. Axmark, *MySQL Reference Manual*. O'Reilly Media, Inc, 2002.
- [5] J. Chen, *Visual Studio Code: End-to-End Editing and Debugging Tools for Web Developers*. Packt Publishing Ltd, 2019.
- [6] K. Beck et al., "Manifesto for Agile Software Development," Recuperado de <http://agilemanifesto.org/>, 2001.
- [7] 2. Spring.io, «Spring Framework, "Spring Framework Reference Documentation,"» [En línea]. Available: <https://docs.spring.io/spring-framework/docs/current/reference/html/>. [Último acceso: 20 julio 2024].
- [8] 2. Spring.io, «Spring Boot, "Spring Boot Reference Documentation,"» [En línea]. Available: <https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/>. [Último acceso: 20 julio 2024].
- [9] 2. Baeldung, «Baeldung, "Spring Tutorials and Guides,"» [En línea]. Available: <https://www.baeldung.com/>. [Último acceso: 19 julio 2024].

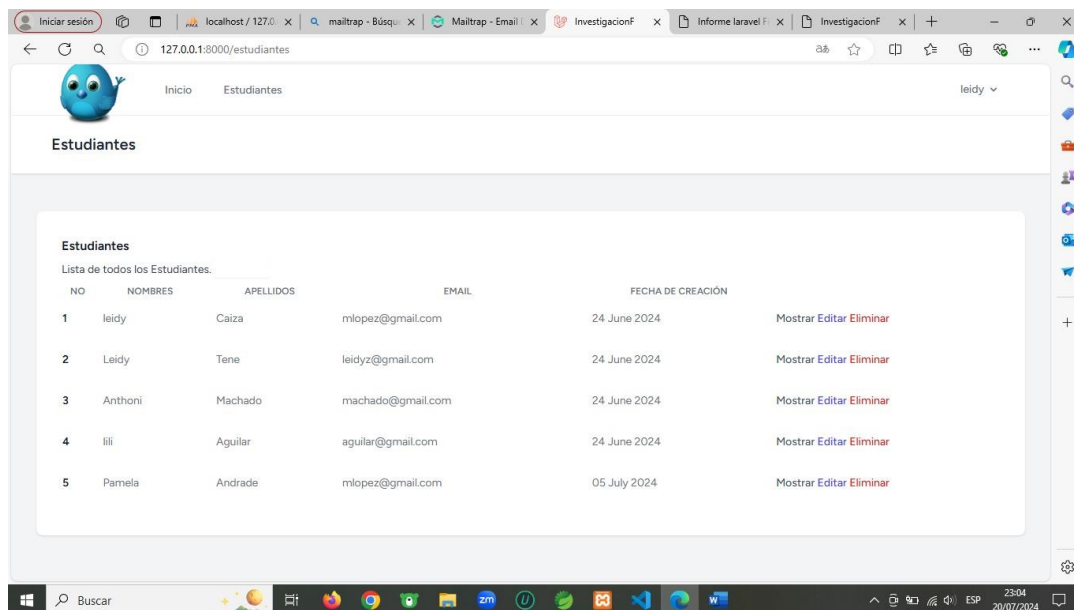
8. ANEXO



```

11 {
12     /**
13      * Validate and create a newly registered user.
14      *
15      * @param array<string, string> $input
16      */
17     public function create(array $input): User
18     {
19         $validator = Validator::make($input, [
20             'name' => ['required', 'string', 'max:255'],
21             'lastname' => ['required', 'string', 'max:255'],
22             'birthday' => ['required', 'date'],
23             'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
24             'password' => $this->passwordRules(),
25             'recaptcha_token' => ['required', new Recaptcha($input['recaptcha_token'])],
26         ]);
27         if ($validator->fails()) {
28             return $this->redirectTo('recaptcha_token');
29         }
30
31         return User::create([
32             'name' => $input['name'],
33             'lastname' => $input['lastname'],
34             'birthday' => $input['birthday'],
35             'email' => $input['email'],
36             'password' => Hash::make($input['password']),
37         ]);
38     }
39 }

```



NO	NOMBRES	APELLIDOS	EMAIL	FECHA DE CREACIÓN	
1	leidy	Caiza	mlopez@gmail.com	24 June 2024	Mostrar Editar Eliminar
2	Leidy	Tene	leidy@gmail.com	24 June 2024	Mostrar Editar Eliminar
3	Anthoni	Machado	machado@gmail.com	24 June 2024	Mostrar Editar Eliminar
4	lili	Aguilar	aguilar@gmail.com	24 June 2024	Mostrar Editar Eliminar
5	Pamela	Andrade	mlopez@gmail.com	05 July 2024	Mostrar Editar Eliminar

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:

New Spring Starter Project Dependencies

Spring Boot Version:

Frequently Used:

☐ MySQL Driver ☐ Spring Boot DevTools ☐ Spring Data JPA

☐ Spring Web

Available: Selected:

☒ Eureka Server

▼ AI

☐ Transformers (ONNX) Embeddings

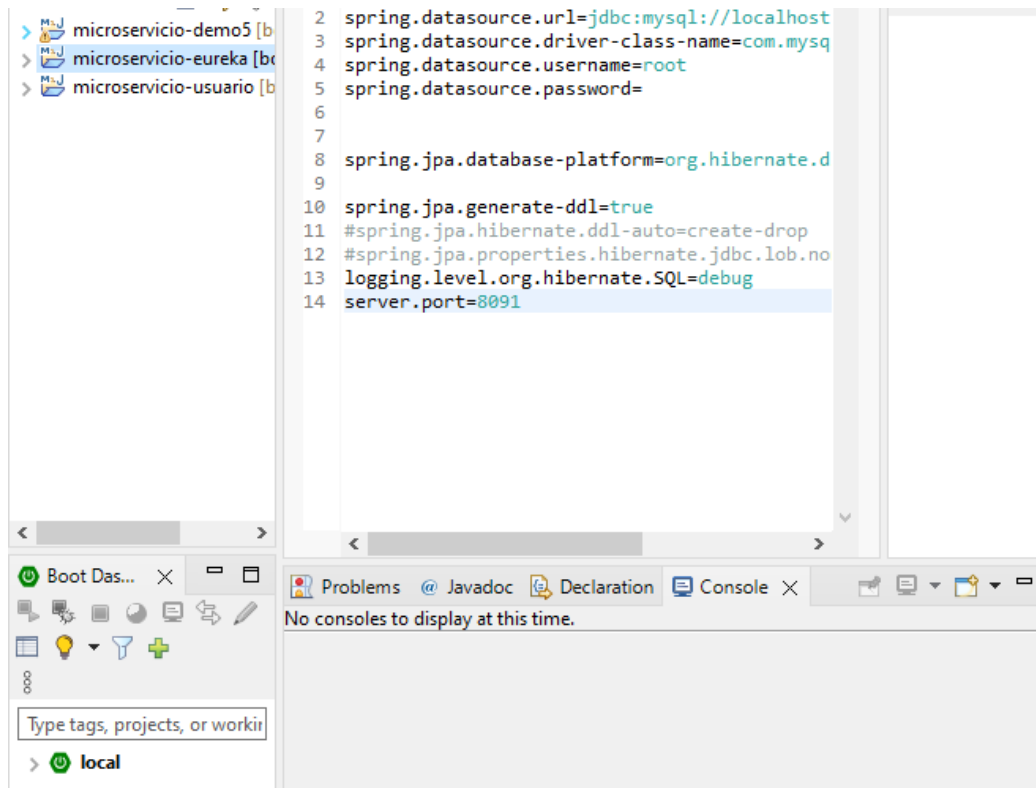
▼ Spring Cloud Discovery

☐ Eureka Discovery Client

☒ Eureka Server

▼ VMware Tanzu Application Service

☐ Service Registry (TAS)



```
2 spring.datasource.url=jdbc:mysql://localhost
3 spring.datasource.driver-class-name=com.mysql
4 spring.datasource.username=root
5 spring.datasource.password=
6
7
8 spring.jpa.database-platform=org.hibernate.d
9
10 spring.jpa.generate-ddl=true
11 #spring.jpa.hibernate.ddl-auto=create-drop
12 #spring.jpa.properties.hibernate.jdbc.lob.no
13 logging.level.org.hibernate.SQL=debug
14 server.port=8091
```

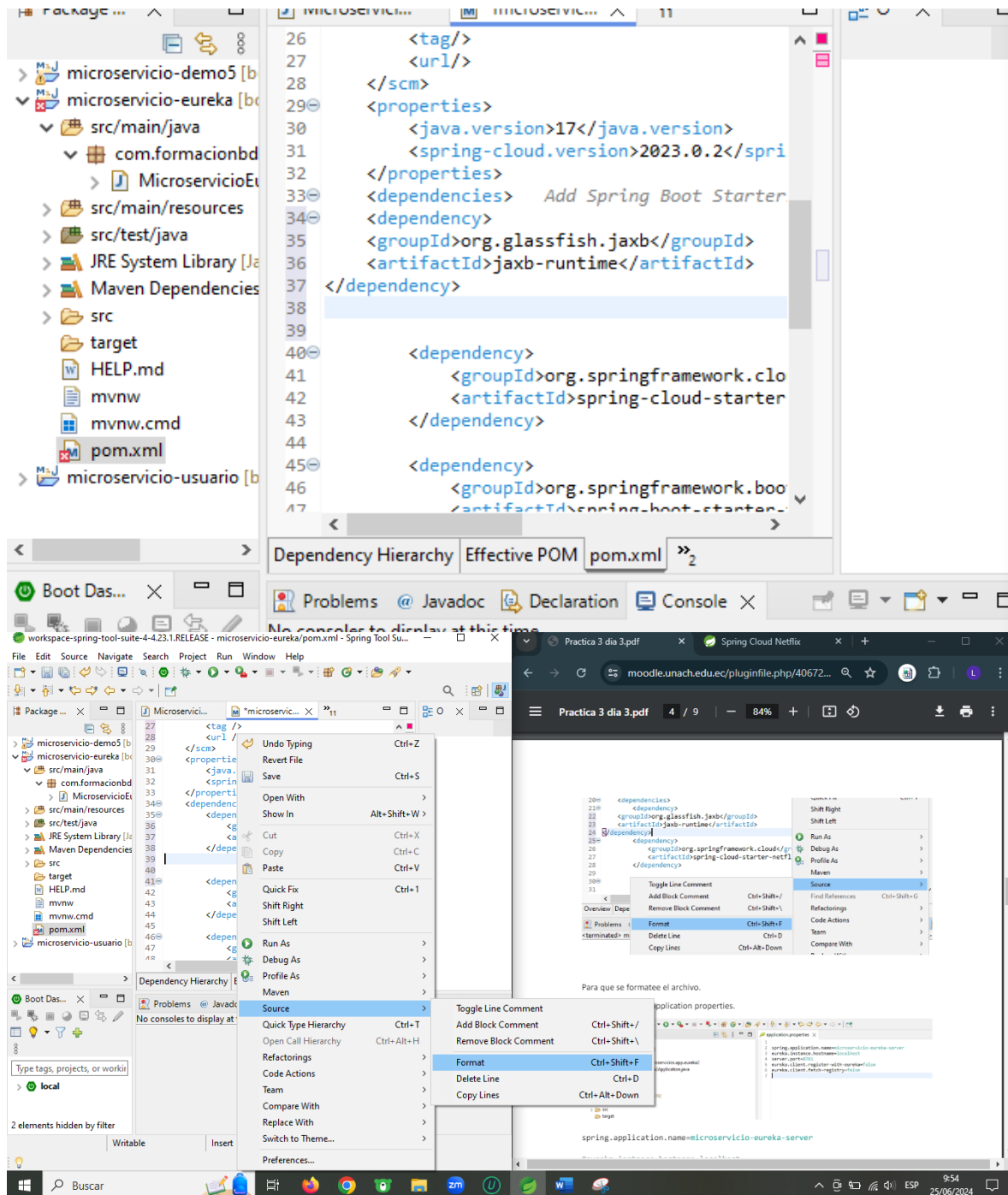
2.9. JDK 11 Support

The JAXB modules which the Eureka server depends upon were removed in JDK 11. If you intend to use JDK 11 when running a Eureka server you must include these dependencies in your POM or Gradle file.

```
<dependency>
  <groupId>org.glassfish.jaxb</groupId>
  <artifactId>jaxb-runtime</artifactId>
</dependency>
```

2.10. AOT and Native Image Support

Spring Cloud Netflix Eureka Server does not support Spring AOT transformations or native images.



The screenshot displays an IDE environment with a Maven project structure. The main editor shows the `pom.xml` file with the following content:

```

26     <tag/>
27     <url/>
28   </scm>
29   <properties>
30     <java.version>17</java.version>
31     <spring-cloud.version>2023.0.2</spring-cloud.version>
32   </properties>
33   <dependencies>
34     <dependency>
35       <groupId>org.glassfish.jaxb</groupId>
36       <artifactId>jaxb-runtime</artifactId>
37     </dependency>
38
39     <dependency>
40       <groupId>org.springframework.cloud</groupId>
41       <artifactId>spring-cloud-starter</artifactId>
42     </dependency>
43
44     <dependency>
45       <groupId>org.springframework.boot</groupId>
46       <artifactId>spring-boot-starter</artifactId>
47     </dependency>

```

The IDE interface includes a Project Explorer on the left showing the project structure, a Console window at the bottom, and a menu bar with options like File, Edit, Source, Navigate, Search, Project, Run, Window, and Help. The status bar at the bottom indicates the current file is `pom.xml` and the project is named `microservicio-eureka`.





The screenshot displays a Spring Eureka web interface in a browser window at localhost:8761. The interface shows the 'System Status' with environment 'test', data center 'default', and current time '2024-06-25T09:57:40 -0500'. It also shows 'DS Replicas' and 'Instances currently registered with Eureka' with a table indicating 'No instances available'. The 'General Info' section is partially visible.

Overlaid on the bottom is an IDE window showing a code editor with the following code:

```
th-eureka=false  
try=false
```

The IDE also shows a console window with the following output:

```
INFO 18100 --- [microservicio-eureka] [ ... ]  
INFO 18100 --- [microservicio-eureka] [ ... ]  
INFO 18100 --- [microservicio-eureka] [ ... ]
```

The IDE's 'Run' menu is open, showing options like 'Run As', 'Debug As', 'Profile As', 'Restore from Local History...', 'Maven', 'Team', 'Compare With', 'Configure', and 'Spring'. The 'Spring' option is selected.

In the background, a PDF document titled 'Practica 3 dia 3.pdf' is visible, showing a section titled '7. Ir a microservicio usuarios y editar las dependencias con click o'.

ckage

Service URL:

Spring Boot Version:

Frequently Used:

☐ Eureka Server ☐ MySQL Driver ☐ Spring Boot DevTools

☐ Spring Data JPA ☐ Spring Web

Available: X

Selected:

X Eureka Discovery Client

▼ AI

☐ Transformers (ONNX) Embeddings

▼ Spring Cloud Discovery

☒ Eureka Discovery Client

☐ Eureka Server

▼ VMware Tanzu Application Service

☐ Service Registry (TAS)

ot Da

tags,

loc

ADY

reka

reka

reka

reka

reka

reka

Compare local project with generated project from Spring Initializr

Differences detected between local project and project from Initializr Service

Structure Compare

- ☐ HELP.md
- ☐ pom.xml
- ☒ src
 - ☒ main
 - ☒ resources
 - ☐ application.properties

Maven POM Compare No Difference

Spring Initializr: starter.zip	Local project: microservicio-usuario
31 <spring-cloud.version>202	31 <spring-cloud.version>
32 </properties>	32 </properties>
33 <dependencies>	33 <dependencies>
34 <dependency>	34 <dependency>
35 <groupId>org.springframework	35 <groupId>org.springframework
36 <artifactId>spring-cl	36 <artifactId>spring
37 </dependency>	37 </dependency>
38 <dependency>	38 <dependency>
39 <groupId>org.springframework	39 <groupId>org.springframework
40 <artifactId>spring-bc	40 <artifactId>spring
41 <scope>test</scope>	41 </dependency>
42 </dependency>	42 <dependency>
43 </dependencies>	43 <groupId>org.springframework
44 <dependencyManagement>	44 <artifactId>spring
45 <dependencies>	45 <scope>runtime</s
46 <dependency>	46 <optional>true</c
47 <groupId>org.springframework	47 </dependency>
48 <artifactId>spring	48 <dependency>
49	49

Navigation: < Back Next > Finish Cancel



```
vigate Search Project Run Window Help
microservi... *application... X Microservi... microservi... application...
vicio-demo5 [boot]
vicio-eureka [boot]
vicio-usuario [boot] [devtools]
ain/java
ain/resources
tic
nplates
application.properties
st/java
stem Library [JavaSE-17]
n Dependencies

md
r
cmd
uml

bard X
ects, or working set names to match (incl. * an

pservicio-demo5
pservicio-eureka [8761]
pservicio-usuario [devtools]

microservicio-eureka - MicroservicioEurekaApplication [Spring Boot App] C:\Users\LADY\Downloads\spring-tool-suite-4-4.23.1.RELEASE-e4.32.0-win32.win32.x86_64.self-ext
2024-06-25T09:57:05.767-05:00 INFO 18100 --- [microservicio-eureka] [n(10)-127.0.0.1] o.s.web.servlet.DispatcherServlet :
2024-06-25T09:57:05.768-05:00 INFO 18100 --- [microservicio-eureka] [n(10)-127.0.0.1] o.s.web.servlet.DispatcherServlet :
2024-06-25T09:58:04.327-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :
2024-06-25T09:59:04.328-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :
2024-06-25T10:00:04.341-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :
2024-06-25T10:01:04.340-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :
2024-06-25T10:02:04.351-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :
2024-06-25T10:03:04.351-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :
2024-06-25T10:04:04.362-05:00 INFO 18100 --- [microservicio-eureka] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry :

1 spring.application.name=microservicio-usuario
2 server.port=${PORT:80}
3
4 eureka.instance.instance-id=${spring.application.name}:${random.value}
5 eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka
6 spring.cloud.discovery.enabled=true
7 eureka.client.register-with-eureka=true
8 spring.datasource.url=jdbc:mysql://localhost:3306/academico
9 spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
10 spring.datasource.username=root
11 spring.datasource.password=
12
13
14 spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
15
16 spring.jpa.generate-ddl=true
17 #spring.jpa.hibernate.ddl-auto=create-drop
18 #spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
19 logging.level.org.hibernate.SQL=debug
20 #server.port=8091
```

New Spring Starter Project

Service URL:

Name:

☒ Use default location

Location:

Type: Packaging:

Java Version: Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:



New Spring Starter Project Dependencies

Spring Boot Version: 3.3.1

Frequently Used:

- ☒ Eureka Discovery Client
- ☐ Eureka Server
- ☐ MySQL Driver
- ☒ Spring Boot DevTools
- ☐ Spring Data JPA
- ☐ Spring Web

Available:

gatewa

- ▼ Spring Cloud Routing
 - ☒ Gateway
 - ☐ Reactive Gateway

Selected:

- X Spring Boot DevTools
- X Eureka Discovery Client
- X Gateway

```

23<scm>
24<connection/>
25<developerConnection/>
26<tag/>
27<url/>
28</scm>
29<properties>
30<java.version>17</java.version>
31<spring-cloud.version>2023.0.2</spring-cloud.version>
32</properties>
33<dependencies> Add Spring Boot Starters...
34<dependency>
35<groupId>org.springframework.cloud</groupId>
36<artifactId>spring-cloud-starter-gateway-mvc</artifactId>
37</dependency>
38<dependency>
39<groupId>org.springframework.cloud</groupId>
40<artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
  
```

view Dependencies Dependency Hierarchy Effective POM pom.xml

Problems @ Javadoc Declaration Console

consoles to display at this time.



workspace-spring-tool-suite-4-4.23.1.RELEASE - Spring Tool Suite 4

File Edit Source Refactor Source Navigate Search Project Run Window Help

Package Explorer

- microservicio-demo5 [boot]
- microservicio-eureka [boot]
- microservicio-gateway [boot] [devtools]
- microservicio-usuario [boot] [devtools]

microservicio-gateway

```
1 package com.formacionbdi.microservicios.app.usuario.controllers;
2
3 import java.util.Optional;
4
5 @RestController
6 @RequestMapping("/")
7
8 public class AlumnoController {
9     @Autowired
10     private AlumnoService service;
11
12     //Indica que es un generico puede guardar cualquier cosa
13     @GetMapping
14     public ResponseEntity<> listar() {
15         return ResponseEntity.ok().body(service.findAll());
16     }
17
18     @GetMapping("/{id}")
19     public ResponseEntity<> ver(@PathVariable Long id) {
20         Optional<Alumno> o = service.findById(id);
21         if (o.isPresent()) {
22             return ResponseEntity.ok(o.get());
23         }
24         return ResponseEntity.notFound().build();
25     }
26
27     @PostMapping
28     public ResponseEntity<> crear(@RequestBody Alumno alumno) {
29         Alumno alumnoDb = service.save(alumno);
30     }
31 }
```

Outline

- com.formacionbdi.microservicio
- AlumnoController
- service: AlumnoService
- listar(): ResponseEntity<>
- ver(Long): ResponseEntity<>
- crear(Alumno): ResponseEntity<>
- editar(Alumno, Long): Response
- eliminar(Long): ResponseEnti

Boot Dashboard

Type tags, projects, or working set names to match (incl. * and ? wildcards)

- local
- microservicio-demo5
- microservicio-eureka [8761]
- microservicio-gateway [devtools] [8090]
- microservicio-usuario [devtools] [65005]

4 elements hidden by filter

microservicio-gateway

Problems

Console

```
2024-07-21T22:54:42.463-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T22:55:42.463-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T22:56:42.464-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T22:57:42.470-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T22:58:42.471-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T22:59:42.479-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T23:00:42.479-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
2024-07-21T23:01:42.480-05:00 INFO 24936 --- [microservicio-eureka-server] [a-EvictionTimer] c.n.e.registry.AbstractInstanceRe
```

23:02
21/07/2024