

- Gestión de una empresa agrícola -

Alexis Gil Cabrera

Abril de 2024

Proyecto fin de ciclo
Desarrollo de Aplicaciones Web





Fase de análisis.....	2
Nombre del proyecto.....	2
Resumen.....	2
Objetivos del proyecto.....	3
Análisis de requisitos previos.....	3
Diagrama de casos de uso.....	5
Fase de diseño.....	7
Diagrama de clases.....	7
Diagrama entidad-relación.....	8
Estructura de base de datos.....	8
Pantallas de diseño de nuestra aplicación.....	13
Codificación.....	16
Preparación de un entorno de Laravel desde 0.....	16
Creación del proyecto Agrolanza.....	17
Implementación de sistema de login.....	18
Instalación de Bootstrap y Bootstrap Icons.....	20
Crear panel administrativo con AdminLTE.....	22
Implementación de roles y permisos.....	23
Estructura de modelos.....	29
Migraciones.....	34
Implementación de mapa interactivo.....	39
Controladores.....	40
Enrutamiento.....	45
Implementación de generador PDF.....	46
Pruebas unitarias.....	47
Montaje del proyecto mediante servidor web.....	52
Conclusiones.....	56
Repositorio.....	57
Anexo. Definiciones.....	57



Fase de análisis

Nombre del proyecto

El nombre que he querido asignar a este proyecto es “Agrolanza”. Este nombre nace de la fusión del sector agrícola de Lanzarote, el cual podemos considerar único a nivel mundial por muchos factores, con la propia palabra “Lanzarote”.

Resumen

Desde hace varios siglos, la principal fuente de alimentación y comercio en la isla de Lanzarote ha sido el sector agrario (tanto ganadería como agricultura). Esto se debe a que, hasta hace relativamente poco, nos situábamos en un entorno aislado por su propia naturaleza, y las importaciones desde el exterior no eran tan abundantes como lo son en la actualidad.

Podemos considerar entonces que, el trabajo agrícola ha sido fundamental para subsistir. Como hemos mencionado en el apartado anterior, el sector agrícola isleño se caracteriza por ser, en muchos aspectos, único en el mundo. Esto se debe al clima seco y tan complicado en el que nos encontramos, donde la única forma de obtener agua potable era obteniéndose de la propia lluvia, mediante el uso de estructuras como aljibes y maretas. El viento constante todo el año, la presencia de rofe o picón en terrenos de cultivo (también característico a nivel mundial) y el difícil acceso a ciertas áreas por su orografía conllevan una serie de técnicas y costumbres que se han ido desarrollando a lo largo del tiempo para conseguir mejores resultados en este sector.

Actualmente, podemos apreciar que muchos de los terrenos o parcelas destinados a la explotación agrícola se encuentran en estado de abandono. Esto se puede deber por dos principales factores:

- Economía: Actualmente, el trabajo agrícola está muy mal valorado, por lo que los agricultores sienten que pierden su tiempo y sacrificio para obtener cosechas que tendrán que mal-vender.
- Auge del sector terciario. Esto va muy vinculado con la presencia del turismo en la isla. Este comenzó en la década de 1920, y se consolidó en la década de los 80. Desde este punto, ha ido creciendo de manera exponencial hasta la actualidad. Por tanto, un gran porcentaje de los habitantes isleños se encuentran trabajando de cara al turismo.

Sin embargo, gracias a una iniciativa del Cabildo de Lanzarote, se está tratando de recuperar poco a poco este sector que en un pasado nos daba de comer, y que sin éste no hubiera sido posible sobrevivir. Esto se puede apreciar en el proyecto de recuperación de terrenos en estado de abandono, en el que el Cabildo se compromete a ofrecer ayuda para esta tarea, a cambio del compromiso del propietario para mantener la parcela en un estado óptimo para su explotación. También se ofrecen ayudas y subvenciones a aquellos que se dediquen a este sector, para poder seguir adelante con el proyecto.



Existen varias empresas en la isla que se especializan en el sector agrícola, y algunas de ellas siguen llevando a cabo procedimientos de gestión de manera rudimentaria, ya sea a base de papel y bolígrafo, o como mucho mediante uso de ofimática.

En el siguiente apartado haremos mención del objetivo de este proyecto, cómo podemos involucrarnos en este sector, y las ventajas que puede ofrecer frente a otras alternativas y metodologías.

Objetivos del proyecto

El principal objetivo de este proyecto es ayudar a ciertas empresas agrícolas con el proceso de informatizarse, con el fin de lograr mayor eficiencia en la gestión de la misma. Nuestro sistema se basará principalmente en la gestión de **empleados**, **clientes** y sus **parcelas**, herramientas y **maquinarias** para labores de labranza, **productos fitosanitarios**, y por último pero no menos importante, los **servicios** que se deben llevar a cabo, y la asignación de personal y elementos como maquinarias o fitosanitarios que se requieran.

Además, se implementarán las siguientes opciones:

- La exportación de información en formato PDF, ya que presenta comodidad a los usuarios para imprimir o compartir por plataformas como correo electrónico o aplicaciones de mensajería instantánea.
- Calculadora para fitosanitarios, que ayudará a hacer todos aquellos cálculos en base a las necesidades para preparar el **caldo**.
- Mapa interactivo que será utilizado para localizar coordenadas de parcelas en tiempo real.

Cabe mencionar que los empleados podrán ser dados de alta en el sistema por el administrador, y por el rol que se les asigne tendrán acceso a determinados recursos accesibles en la plataforma. Esto puede permitir tareas más exhaustivas para aquellos empleados que deban realizar gestiones, y permitir o denegar el acceso a áreas a las que no estén autorizados.

A continuación, llevaremos a cabo un análisis de requisitos para poder llevar a cabo el proyecto.

Análisis de requisitos previos

Para llevar adelante el desarrollo de este sistema de gestión, debemos tener en cuenta qué herramientas necesitamos:



- **Gestor de base de datos.** Para el almacenamiento de datos necesitaremos un gestor de base de datos que sea eficiente y práctico de usar. Entre otros gestores, predominan MySQL y PostgreSQL. Este último es un buen gestor de base de datos, pero se enfoca más a sistemas de datos algo más masivos, y no ofrece la misma velocidad de rendimiento de lectura frente a **MySQL**. MySQL ofrece gran estabilidad y fluidez en el manejo de datos, además de ser muy intuitivo durante su uso, mediante lenguaje SQL. Por tanto, con MySQL podremos desarrollar nuestra aplicación web sin problema. En el hipotético caso de que en el futuro podamos observar ciertas limitaciones con este gestor de base de datos (por la estructura del diagrama que desarrollaremos sería poco probable), podremos migrar a Postgres fácilmente, ya que éste admite todos los tipos de datos de MySQL.
- **Entorno de trabajo.** Debemos determinar cómo desarrollar el proyecto. En mi caso, he decidido utilizar un framework o entorno de trabajo, de manera que podamos tener un manejo de datos de manera que sea seguro y eficiente. Concretamente, he decidido enfocarme en el entorno de trabajo Laravel (basado principalmente en el lenguaje de programación PHP), ya que para los objetivos a tener en cuenta para la plataforma a desarrollar es ideal. En la fecha que se desarrolla este trabajo, ha salido recientemente **Laravel 12** de manera oficial. Esta versión ofrece ciertas ventajas frente a Laravel 11, tales como ejecución más rápida, código más limpio y herramientas más inteligentes. Por lo demás, no presenta muchas más diferencias frente a su predecesor. Estas dos versiones sí presentan diversas diferencias frente a las versiones anteriores, pero siguen siendo compatibles con un gran número de herramientas y plugins.
- **Manejo de perfiles de usuario / empleado.** Para el sistema que vamos a desarrollar, es imprescindible la adecuada gestión de perfiles de usuario, o en este caso, de los empleados. En Laravel 12, podemos utilizar **Laravel Breeze**, un paquete de autenticación muy cómodo de manejar y práctico. La recomendación para utilizar dicho paquete, es que se recomienda instalar justo después de la creación del proyecto, ya que puede modificar ciertos ficheros, y si los hemos editado, se puede perder información.
- **Creación de roles.** En todas las plataformas informáticas, es importante cumplir con los 3 pilares fundamentales de la información:
 - **Disponibilidad.** La información debe ser accesible a los usuarios autorizados
 - **Integridad.** Los datos y metadatos no deben estar expuestos a amenazas que puedan corromperlos.
 - **Confidencialidad.** La información no debe estar al alcance de entidades no autorizadas

Es por ello que, al igual que se debe implantar un sistema de inicio de sesión seguro, se deben implantar permisos y roles para que los usuarios tengan acceso a aquellos recursos a los que estén autorizados. En Laravel, podemos instalar el paquete **Spatie Laravel Permission**. Este creará tablas de permisos y roles en la base de datos, y podremos asociarlas a los usuarios registrados para asignarles los



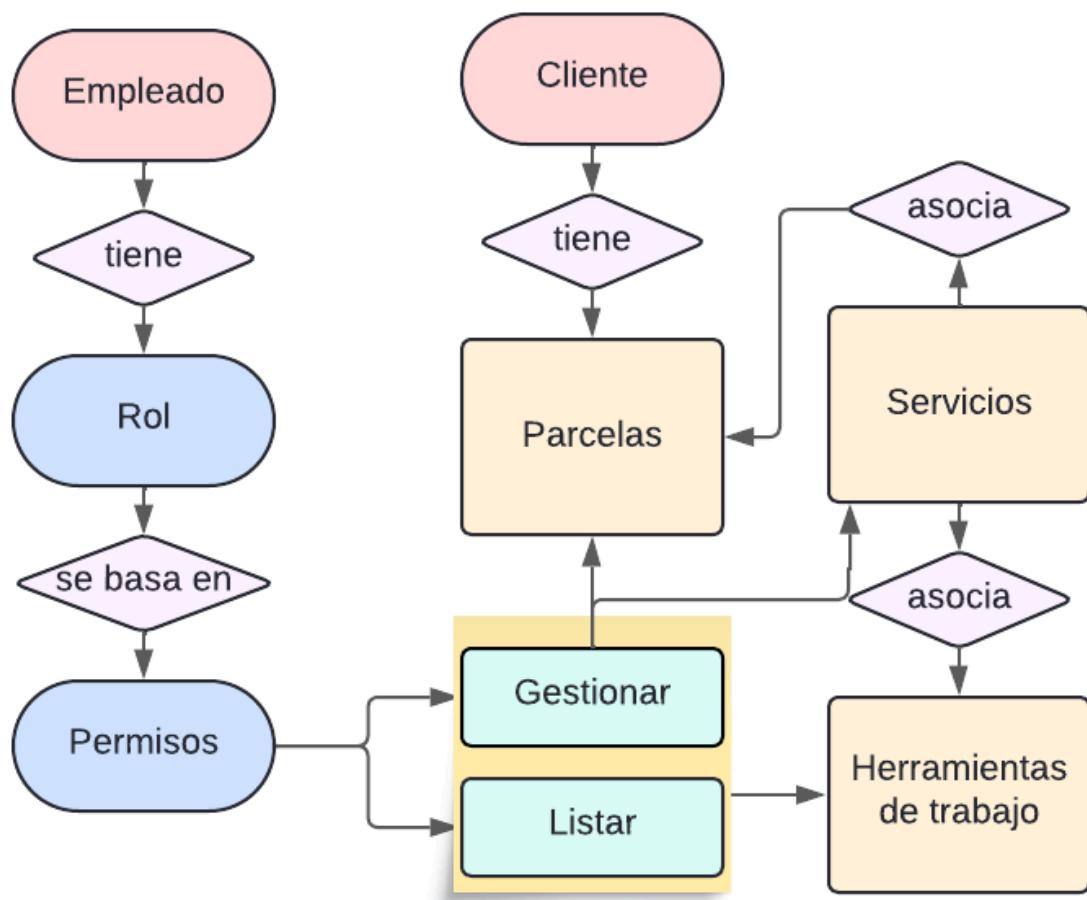
permisos apropiados. En este caso, se van a asignar mediante los roles que hemos mencionado en los objetivos del proyecto.

- **Uso de estilos en la aplicación.** No solo nos tenemos que ceñir a la seguridad de la plataforma, sino que ésta sea intuitiva y atractiva de cara al usuario. Por tanto, es ideal utilizar en todo momento estilos que concuerden con la usabilidad y accesibilidad web, logrando que nuestra aplicación web se adapte a un abanico amplio de personas. Para que nos resulte más fácil este paso, implementaremos **Bootstrap** y **Bootstrap Icons** en nuestro entorno de trabajo.
- **Generador de PDF.** Es importante que los empleados puedan exportar la información que se muestra en la propia plataforma, ya sea para poder descargar y almacenar como registro en sus equipos personales, o directamente tenerlos impresos. **DomPDF** es una herramienta que se implementa en Laravel. Concretamente, en las últimas versiones, el código se vuelve mucho más simple para su manejo, en comparación con versiones anteriores a Laravel 10.
- **Visualización y manejo de coordenadas en tiempo real.** Muchas veces, se anotan las coordenadas de una parcela o terreno, ya que no se encuentra en una calle, o directamente se debe acceder por camino de tierra a las afueras. Es por ello que es indispensable que el sistema cuente con un visualizador de coordenadas en tiempo real que sea fácil de manejar. Una opción que personalmente me llamó la atención es **Leafletjs**. Este se basa en lenguaje JavaScript, lo que permite su interactividad desde el navegador del usuario en tiempo real. Además, refleja muchos datos reales de las áreas que se visualizan, es muy ligero, eficiente y además, es gratuito. Otras APIs requieren de un token de usuario, ya que presentan precios por su uso o por mensualidades. En nuestro caso, para visualizar una coordenada y obtener los mismos haciendo click en un punto del mapa, Leafletjs es muy viable. Además, para su implementación simplemente debemos apuntar a los recursos desde la propia vista, al igual que normalmente hacemos si estamos utilizando JQuery.
- **Barra de navegación.** Para los usuarios es muy intuitivo el manejo de una plataforma cuando tiene una barra de navegación. Podemos crearla, o incluso implementar alguna creada. Nos basta con la que incorpora **AdminLTE 4**.

En el siguiente paso, se presentará el diagrama de casos de uso.

Diagrama de casos de uso

A continuación, se mostrará un diagrama que refleja la ideología del proyecto en cuanto al uso de la aplicación web:



Como podemos observar, la idea principal es clasificar por roles (basados en permisos) para que los empleados puedan hacer determinadas tareas sobre servicios, herramientas de trabajo (en las que se incluyen tanto la gestión de productos fitosanitarios como de las maquinarias agrícolas), las parcelas (asociadas a los clientes o representantes legales de las mismas) y los servicios, que involucran parcelas y herramientas de trabajo.

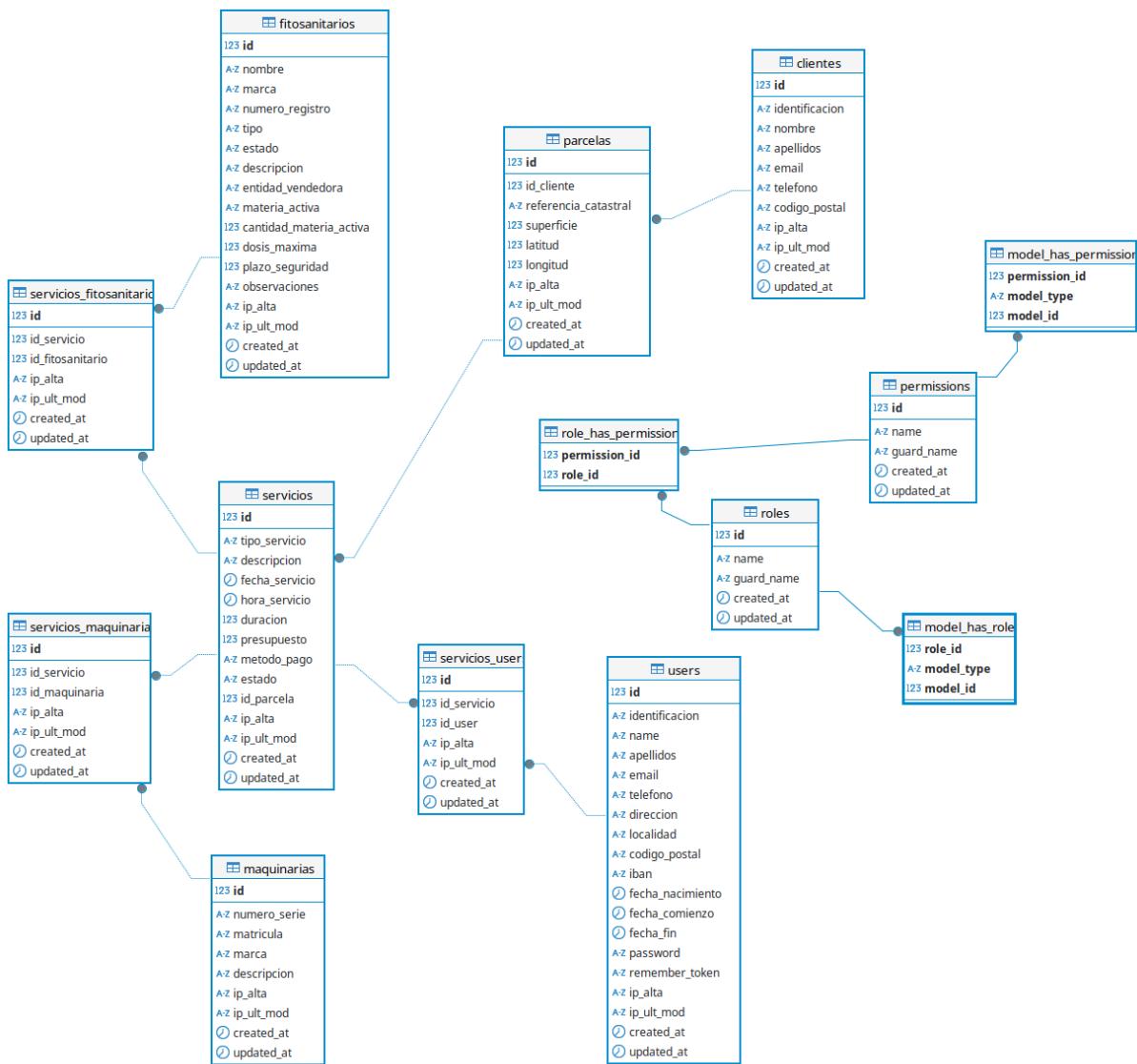


Fase de diseño

Para crear cualquier aplicación web, es de vital importancia planificar y diseñar la estructura, y posteriormente pasar al desarrollo en sí. De esta forma, evitaremos muchos imprevistos e inconsistencias. De este modo, seremos más previsores y evitaremos a la larga complicaciones durante el trabajo.

Diagrama de clases

Se mostrará a continuación, el diagrama de clases del proyecto a diseñar:

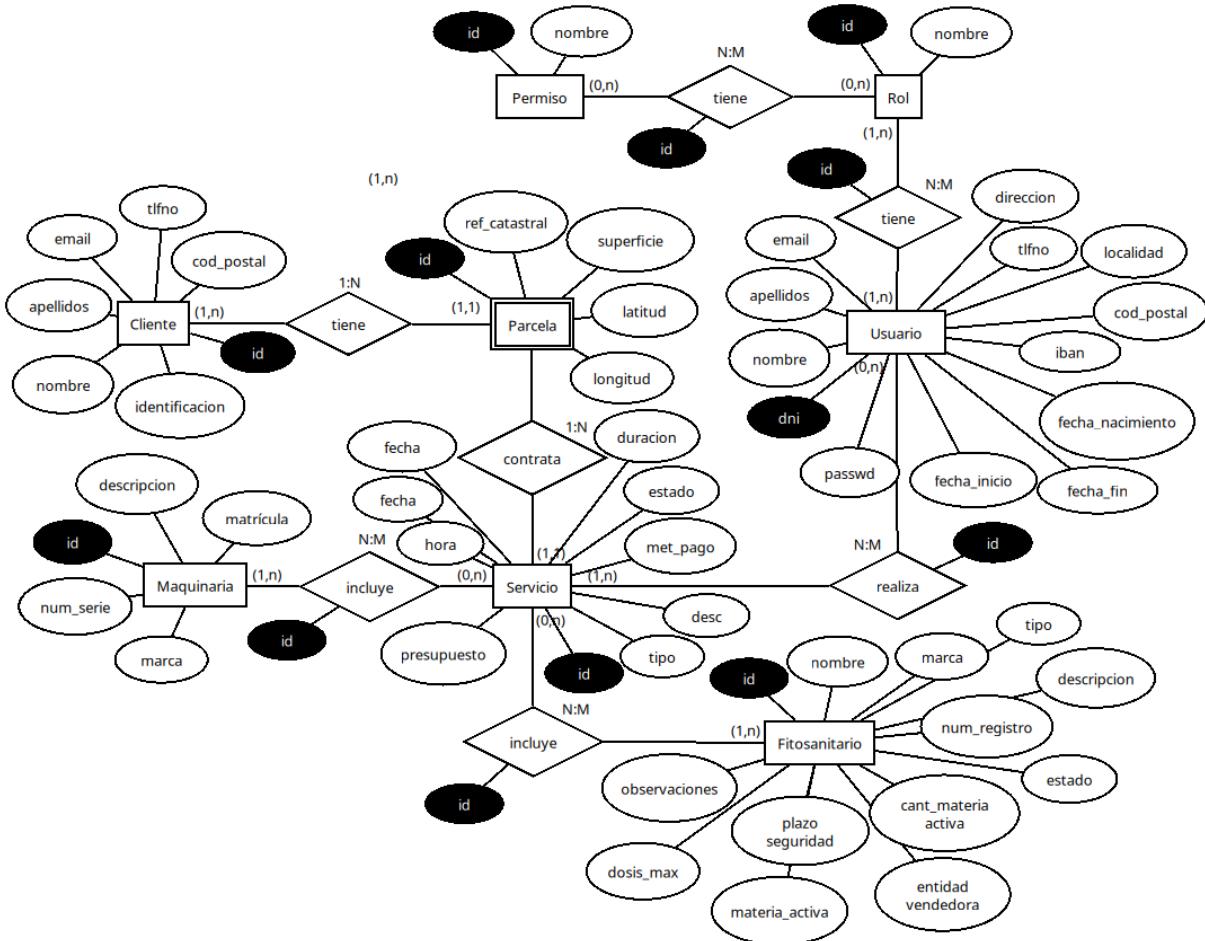


* La tabla **model_has_permission** enlaza a la tabla **users**.



Diagrama entidad-relación

Acerca del diagrama presentado anteriormente, ahora se presenta el diagrama Entidad-Relación:



Estructura de base de datos

Sobre los esquemas anteriores, se incluye a continuación la creación de las tablas en lenguaje SQL que, en un futuro, serán adaptadas a los ficheros de migraciones de Laravel:

➤ users

```
CREATE TABLE users (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    identificacion CHAR(10) UNIQUE,
    name VARCHAR(30),
    apellido VARCHAR(60),
    email VARCHAR(40) UNIQUE,
```



```
telefono CHAR(12),
direccion VARCHAR(150),
localidad VARCHAR(50),
codigo_postal CHAR(5),
iban CHAR(24),
fecha_nacimiento DATE,
fecha_comienzo DATE,
fecha_fin DATE NULL,
password VARCHAR(255),
remember_token VARCHAR(100) NULL,
ip_alta VARCHAR(15) NULL,
ip_ult_mod VARCHAR(15) NULL,
created_at TIMESTAMP NULL,
updated_at TIMESTAMP NULL
);
```

➤ clientes

```
CREATE TABLE clientes (
id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
identificacion CHAR(10) UNIQUE,
nombre VARCHAR(30),
apellidos VARCHAR(60),
email VARCHAR(40) NULL,
telefono CHAR(12),
codigo_postal CHAR(5),
ip_alta VARCHAR(15) NULL,
ip_ult_mod VARCHAR(15) NULL,
created_at TIMESTAMP NULL,
updated_at TIMESTAMP NULL
);
```

➤ parcelas

```
CREATE TABLE parcelas (
id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
id_cliente BIGINT UNSIGNED,
```



```
referencia_catastral CHAR(20) UNIQUE,  
superficie INT UNSIGNED,  
latitud DECIMAL(10, 6),  
longitud DECIMAL(10, 6),  
ip_alta VARCHAR(15) NULL,  
ip_ult_mod VARCHAR(15) NULL,  
created_at TIMESTAMP NULL,  
updated_at TIMESTAMP NULL,  
FOREIGN KEY (id_cliente) REFERENCES clientes(id)  
);
```

➤ maquinarias

```
CREATE TABLE maquinarias (  
id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
numero_serie VARCHAR(20),  
matricula CHAR(10),  
marca VARCHAR(20),  
descripcion VARCHAR(200),  
ip_alta VARCHAR(45) NULL,  
ip_ult_mod VARCHAR(45) NULL,  
created_at TIMESTAMP NULL,  
updated_at TIMESTAMP NULL  
);
```

➤ fitosanitarios

```
CREATE TABLE fitosanitarios (  
id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
nombre VARCHAR(29),  
marca VARCHAR(29),  
numero_registro VARCHAR(29),  
tipo CHAR(2),  
estado CHAR(2),  
descripcion VARCHAR(200),  
entidad_vendedora VARCHAR(40),  
materia_activa VARCHAR(30),
```



```
cantidad_materia_activa DECIMAL(5, 2),
dosis_maxima DECIMAL(5, 2),
plazo_seguridad INT,
observaciones VARCHAR(200) NULL,
ip_alta VARCHAR(15) NULL,
ip_ult_mod VARCHAR(15) NULL,
created_at TIMESTAMP NULL,
updated_at TIMESTAMP NULL
);
```

➤ servicios

```
CREATE TABLE servicios (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    tipo_servicio ENUM('CP', 'CM', 'SC', 'SS'),
    descripcion VARCHAR(60),
    fecha_servicio DATE,
    hora_servicio TIME,
    duracion INT,
    presupuesto DECIMAL(7, 2),
    metodo_pago ENUM('EF', 'TA', 'TR'),
    estado ENUM('P', 'N'),
    id_parcela BIGINT UNSIGNED,
    ip_alta VARCHAR(15) NULL,
    ip_ult_mod VARCHAR(15) NULL,
    created_at TIMESTAMP NULL,
    updated_at TIMESTAMP NULL,
    FOREIGN KEY (id_parcela) REFERENCES parcelas(id)
);
```

➤ servicios_users

```
CREATE TABLE servicios_users (
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,
    id_servicio BIGINT UNSIGNED,
    id_user BIGINT UNSIGNED,
    ip_alta VARCHAR(15) NULL,
    ip_ult_mod VARCHAR(15) NULL,
    created_at TIMESTAMP NULL,
```



```
updated_at TIMESTAMP NULL,  
FOREIGN KEY (id_servicio) REFERENCES servicios(id) ON DELETE CASCADE,  
FOREIGN KEY (id_user) REFERENCES users(id) ON DELETE CASCADE  
);
```

➤ servicios_maquinarias

```
CREATE TABLE servicios_maquinarias (  
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    id_servicio BIGINT UNSIGNED,  
    id_maquinaria BIGINT UNSIGNED,  
    ip_alta VARCHAR(15) NULL,  
    ip_ult_mod VARCHAR(15) NULL,  
    created_at TIMESTAMP NULL,  
    updated_at TIMESTAMP NULL,  
    FOREIGN KEY (id_servicio) REFERENCES servicios(id) ON DELETE CASCADE,  
    FOREIGN KEY (id_maquinaria) REFERENCES maquinarias(id) ON DELETE CASCADE  
);
```

➤ servicios_fitosanitarios

```
CREATE TABLE servicios_fitosanitarios (  
    id BIGINT UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
    id_servicio BIGINT UNSIGNED,  
    id_fitosanitario BIGINT UNSIGNED,  
    ip_alta VARCHAR(15) NULL,  
    ip_ult_mod VARCHAR(15) NULL,  
    created_at TIMESTAMP NULL,  
    updated_at TIMESTAMP NULL,  
    FOREIGN KEY (id_servicio) REFERENCES servicios(id) ON DELETE CASCADE,  
    FOREIGN KEY (id_fitosanitario) REFERENCES fitosanitarios(id) ON DELETE CASCADE  
);
```

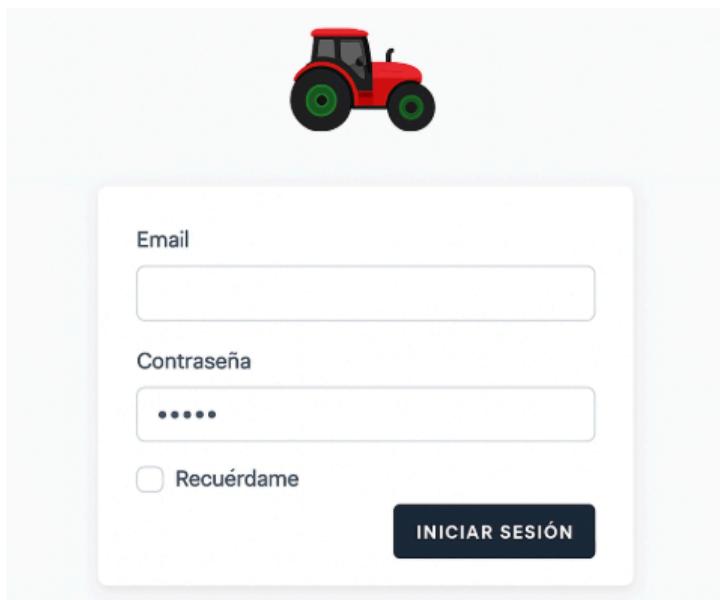


Pantallas de diseño de nuestra aplicación

- Página de bienvenida



- Inicio de sesión





➤ Página principal

Bienvenido, Alexis!

Sistema centralizado de Agrolanza S.L.

Consulta de servicios agrarios
Visualización y exportación de información: Maquinarias, productos fitosanitarios, servicios a parcelas.

Gestión de datos centralizada
Integridad, disponibilidad y confidencialidad de datos para personal autorizado.

Consulta de productos fitosanitarios
Sección con información detallada, y herramienta de cálculo de dosis y otros datos de interés en ámbito fitosanitario.

➤ Vista general de gestión

Servicios

Fecha	Hora	Duración	Parcela	Representante	Tipo de servicio	Presupuesto	Estado	Acciones
05-12-2025	10:00	5 horas	35011A015003380000WH	Isabel	Sembrado por surcos	230.75 €	No pagado	[Actions]
25-11-2025	07:30	4 horas	35034A007006290000QF	Marta	Sembrado por cazolejas	250.00 €	Pagado	[Actions]
18-10-2025	09:00	3 horas	35028A017001970000SH	Raúl	Control de maleza	140.50 €	No pagado	[Actions]
10-09-2025	08:00	2 horas	35028A001000580000SA	Lucía	Control de plagas	210.00 €	Pagado	[Actions]
14-08-2025	12:00	5 horas	35024A018003460000LF	Antonio	Sembrado por surcos	180.00 €	No pagado	[Actions]
22-07-2025	09:00	4 horas	35019A001000880000AH	Carmen	Sembrado por cazolejas	195.00 €	Pagado	[Actions]
08-06-2025	07:30	3 horas	35024A021010250000LP	Manuel	Control de maleza	130.00 €	No pagado	[Actions]
12-05-2025	11:00	2 horas	35024A002000100000LQ	Patricia	Control de plagas	220.00 €	Pagado	[Actions]
20-04-2025	10:00	5 horas	35024A007001890000LR	Alejandro	Sembrado por surcos	175.75 €	No pagado	[Actions]
05-03-2025	07:00	4 horas	35024A008009240000LL	Isabel	Sembrado por cazolejas	200.00 €	Pagado	[Actions]

Mostrando 1 a 10 de 11 resultados

Volver + Nuevo servicio Generar PDF



➤ Vista de edición de datos

Screenshot of the 'Gestión de parcela' (Parcel Management) section in the Agrolanza application.

The interface includes:

- Left sidebar with navigation links: Buscar, Empleados, Clientes, Parcelas, Servicios, Maquinarias, Fitosanitarios, Configuración de cuenta, Perfil.
- Top right corner shows the user's name: Alexis.
- Main area:
 - Titular de la parcela:** Fernando Gómez Serrano
 - Referencia catastral:** 35029A030003350000HO
 - Superficie:** 2511
 - Latitud:** 29,037688
 - Longitud:** -13,688792
 - Guardar** button
- Map view showing the parcel boundaries and elevation points (Montañeta Caldereta at 268m, Montañeta de Uga at 275m, Calera Pintorada at 311m).

➤ Calculadora

Screenshot of the 'Calculadora de dosis' (Dose Calculator) section in the Agrolanza application.

The interface includes:

- Left sidebar with navigation links: Buscar, Empleados, Clientes, Parcelas, Servicios, Maquinarias, Fitosanitarios, Configuración de cuenta, Perfil.
- Top right corner shows the user's name: Alexis.
- Main area:
 - Dosis recomendada:** 1,05
 - Unidad de dosis:** cc/L (selected)
 - Volumen de caldo (L) por Ha (1.000 m):** 15,5
 - Volumen del depósito o mochila (L):** 15
 - Cantidad de producto:** [empty input field]
 - Buttons:** Volver, Calcular



Codificación

Preparación de un entorno de Laravel desde 0

A continuación, se mostrarán los pasos a llevar a cabo para la instalación y creación de un entorno de trabajo con Laravel. Primero instalamos las siguientes herramientas indispensables:

```
sudo apt install npm
sudo apt install php-mysql
sudo apt install composer
```

Procedemos con la instalación de Laravel:

```
composer global require laravel/installer
```

Para que los comandos propios de Laravel funcionen en la terminal, se debe introducir en `~/.bashrc` la siguiente directiva al final:

```
export PATH=".config/composer/vendor/bin:$PATH"
```

Ejecutar el siguiente comando para cargar la configuración:

```
source ~/.bashrc
```

Instalar la siguiente herramienta, necesaria para Pest (orientado a pruebas unitarias):

```
sudo apt install php-xml
```

Comprobar que funciona:

```
alexis@E16-03:~/Documentos/dsw/Laravel$ laravel
Laravel Installer 5.11.0

Usage:
  command [options] [arguments]

Options:
  -h, --help           Display help for the given command. When no command is given display help for the list command
  --silent            Do not output any message
  -q, --quiet          Only errors are displayed. All other output is suppressed
  -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
  -v|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
alexis@E16-03:~/Documentos/dsw/Laravel$ []
```

Crearemos la base de datos en MySQL, que posteriormente será utilizada por Laravel, en el momento de llevar a cabo las migraciones:



```
mysql> create database agrolanza;
Query OK, 1 row affected (0,01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| agrolanza |
| dew |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
6 rows in set (0,00 sec)

mysql> |
```

Ya está todo listo. Podemos proseguir con la creación del proyecto.

Creación del proyecto Agrolanza

Procedemos con la creación del proyecto en cuestión. Utilizaremos MySQL como gestor de base de datos.

```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza$ laravel new

[|] [||] [||]
[|] [||] [||]
[|] [||] [||]
[|] [||] [||]
[|] [||] [||]
[|] [||] [||]

What is the name of your project? ——————
agrolanza

Would you like to install a starter kit? ——————
No starter kit

Which testing framework do you prefer? ——————
Pest

Creating a "laravel/laravel" project at "./agrolanza"
Installing laravel/laravel (v12.0.1)
- Installing laravel/laravel (v12.0.1): Extracting archive
Created project in /home/alexisg/Documentos/2daw/proyecto_agrolanza/agrolanza
Loading composer repositories with package information
Updating dependencies

INFO Application key set successfully.

Which database will your application use? ——————
MySQL
```



```
INFO Application ready in [agrolanza]. You can start your local development using:  
→ cd agrolanza  
→ npm install && npm run build  
→ composer run dev  
  
New to Laravel? Check out our bootcamp and documentation. Build something amazing!
```

Aún no se han procesado las migraciones. Primero instalaremos el sistema de login de usuarios del sistema, y a continuación modificaremos el fichero `.env`, el cual tiene las credenciales para conectar a la base de datos (en este caso de MySQL).

Implementación de sistema de login

```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza$ cd agrolanza/  
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ composer require laravel/breeze --dev  
.composer.json has been updated  
Running composer update laravel/breeze  
Loading composer repositories with package information  
Updating dependencies  
Lock file operations: 1 install, 0 updates, 0 removals  
- Locking laravel/breeze (v2.3.5)  
Writing lock file  
Installing dependencies from lock file (including require-dev)  
Package operations: 1 install, 0 updates, 0 removals  
  
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ php artisan breeze:install  
  
Which Breeze stack would you like to install?  
Blade with Alpine  
  
Would you like dark mode support?  
No  
  
Which testing framework do you prefer?  
-> ● Pest  
○ PHPUnit  
  
found 0 vulnerabilities  
  
> build  
> vite build  
  
vite v6.2.0 building for production...  
✓ 54 modules transformed.  
public/build/manifest.json      0.27 kB | gzip:  0.14 kB  
public/build/assets/app-DbTWgVAB.css 38.35 kB | gzip:  6.91 kB  
public/build/assets/app-DQN0lDuK.js  79.55 kB | gzip: 29.62 kB  
✓ built in 2.88s  
  
INFO Breeze scaffolding installed successfully.
```

El siguiente paso será configurar el fichero `.env` de Laravel, e introduciremos los parámetros pertinentes para conectar a la base de datos:



```
> tests                                         22
> vendor                                       23  DB_CONNECTION=mysql
⚙️ .editorconfig                                24  DB_HOST=127.0.0.1
⚙️ .env                                         25  DB_PORT=3306
$ .env.example                                  26  DB_DATABASE=agrolanza
◆ .gitattributes                                27  DB_USERNAME=alexis
◆ .gitignore                                    28  DB_PASSWORD=[REDACTED]
Ξ artisan                                       29
30  SESSION_DRIVER=database
```

Podemos llevar a cabo las migraciones:

```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ php artisan migrate
INFO: Preparing database.
creating migration table ..... 88.39ms DONE
INFO: Running migrations.
0001_01_01_000000_create_users_table ..... 566.83ms DONE
0001_01_01_000001_create_cache_table ..... 179.71ms DONE
0001_01_01_000002_create_jobs_table ..... 395.68ms DONE
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ mysql -u alexis -p
Enter password:
```

Si accedemos a la base de datos desde MySQL, podremos observar que se han generado las tablas pertinentes.

```
mysql> use agrolanza;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_agrolanza |
+-----+
| cache
| cache_locks
| failed_jobs
| job_batches
| jobs
| migrations
| password_reset_tokens
| sessions
| users
+-----+
9 rows in set (0,00 sec)
```

*Cabe mencionar que todas estas tablas son generadas por defecto. En un futuro, se irán creando los ficheros para las migraciones de las tablas que hemos incluido en el diagrama Entidad - Relación.

Inicializamos el proyecto:



El servicio arranca sin problemas.

Instalación de Bootstrap y Bootstrap Icons

El siguiente paso es incluir Bootstrap y Bootstrap Icons en nuestro proyecto, para así poder hacer uso de los mismos en las vistas.

```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ npm install bootstrap @popperjs/core
added 2 packages, and audited 199 packages in 2s
49 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ npm i bootstrap-icons
added 1 package, and audited 200 packages in 2s
50 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
```

Modificamos el fichero js/app.js. El fichero css/app.css no debemos eliminarlo, puesto que en Laravel 12 sí requiere de dicho fichero. Si lo eliminamos, nos puede devolver un error.



The screenshot shows a file tree on the left with the following structure:

- AGROLANZA
- app
- bootstrap
- config
- database
- node_modules
- public
- resources
 - css
 - js
- app.js
- bootstrap.js

The right pane displays the content of `app.js`:

```
1 import 'bootstrap';
2 import 'bootstrap/dist/css/bootstrap.min.css';
3 import 'bootstrap-icons/font/bootstrap-icons.css';
4
5 //import './bootstrap';
6
7 import Alpine from 'alpinejs';
8
9 window.Alpine = Alpine;
10
11 Alpine.start();
```

Ejecutamos el siguiente comando para comprobar que todo funciona correctamente:

npm run build && npm run dev --watch

```
VITE v6.2.0  ready in 510 ms

→ Local: http://localhost:5173/
→ Network: use --host to expose
→ press h + enter to show help

LARAVEL v12.0.1  plugin v1.2.0

→ APP_URL: http://localhost
```

Probamos si funciona bootstrap correctamente, agregando la siguiente línea en la vista dashboard:

```
<button type="button" class="btn btn-primary">Primary <i class="bi bi-arrow-return-left"></i>
```

The screenshot shows a web application interface. At the top, there is a navigation bar with a logo, a "Dashboard" link, and a user profile dropdown labeled "Alexis". Below the navigation bar, the word "Dashboard" is displayed in bold. A modal window is open, containing the text "You're logged in!" and a blue button with the text "Primary" and a left arrow icon.



Crear panel administrativo con AdminLTE

AdminLTE es un panel de administración de código abierto, el cual puede ser utilizado y modificado por los usuarios. Nosotros haremos uso de la barra lateral izquierda para poder navegar por las diversas secciones de nuestra plataforma.

```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ composer require jeroennoten/laravel-adminlte
./composer.json has been updated
Running composer update jeroennoten/laravel-adminlte
Loading composer repositories with package information
Updating dependencies
Lock file operations: 2 installs, 0 updates, 0 removals
- Locking almasaeed2010/adminlte (v3.2.0)
- Locking jeroennoten/laravel-adminlte (v3.15.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 2 installs, 0 updates, 0 removals
- Downloading almasaeed2010/adminlte (v3.2.0)
- Downloading jeroennoten/laravel-adminlte (v3.15.0)
  1/2 [=====>-----] 50%
No security vulnerability advisories found.
Using version ^3.15 for jeroennoten/laravel-adminlte
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ php artisan adminlte:install
AdminLTE assets files published successfully
Configuration file published successfully
Translation files published successfully
The installation is complete.
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ |
```

Agregamos el siguiente contenido en **dashboard.blade.php**:

```
@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')
    {{-- Add here extra stylesheets --}}
    {{-- <link rel="stylesheet" href="/css/admin_custom.css"> --}}
@stop

@section('js')
    <script> console.log("Hi, I'm using the Laravel-AdminLTE package!"); </script>
@stop
```

Dicha plantilla la podemos visualizar cuando accedemos a Dashboard:



Podremos ir personalizando esta barra lateral a nuestras necesidades y gusto personal.

Implementación de roles y permisos

El uso de roles y permisos es fundamental, como previamente hemos comentado anteriormente. Procedemos a instalar la herramienta **Spatie Laravel Permission**:

```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ composer require spatie/laravel-permission
./composer.json has been updated
Running composer update spatie/laravel-permission
Loading composer repositories with package information
Updating dependencies
Lock file operations: 1 install, 0 updates, 0 removals
  - Locking spatie/laravel-permission (6.15.0)
Writing lock file
Installing dependencies from lock file (including require-dev)
Package operations: 1 install, 0 updates, 0 removals
  - Installing spatie/laravel-permission (6.15.0): Extracting archive
Generating optimized autoload files
 Illuminate\Foundation\ComposerScripts::postAutoloadDump
No security vulnerability advisories found.
Using version ^6.15 for spatie/laravel-permission
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ php artisan vendor:publish --provider="Spatie\Permission\PermissionServiceProvider"
    INFO Publishing assets.

    Copying file [vendor/spatie/laravel-permission/config/permission.php] to [config/permission.php] .....
..... DONE
    Copying file [vendor/spatie/laravel-permission/database/migrations/create_permission_tables.php.stub]
to [database/migrations/2025_02_27_223250_create_permission_tables.php]  DONE
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ |
```



```
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ php artisan migrate:fresh
Dropping all tables ..... 321.21ms DONE
INFO Preparing database.
creating migration table ..... 101.46ms DONE
INFO Running migrations.
0001_01_01_000000_create_users_table ..... 305.82ms DONE
0001_01_01_000001_create_cache_table ..... 100.54ms DONE
0001_01_01_000002_create_jobs_table ..... 217.86ms DONE
2025_02_27_223250_create_permission_tables ..... 868.06ms DONE
alexisg@samsung:~/Documentos/2daw/proyecto_agrolanza/agrolanza$ |
```

En app/Providers/AppServiceProvider.php se debe agregar "use Illuminate\Routing\Router;"

```
EXPLORADOR ... .env JS app.js AppServiceProvider.php # app.css 3 dashboard.blade.php
└─ AGROLANZA
    └─ app
        ├─ Http
        ├─ Models
        └─ Providers
            └─ AppServiceProvider.php
                1  <?php
                2
                3  namespace App\Providers;
                4  use Illuminate\Support\ServiceProvider;
                5  use Illuminate\Routing\Router;
                6  class AppServiceProvider extends ServiceProvider
                {
                /**
                 *
                 * @return void
                 */
                public function register()
                {
                    // Registrar servicios
                }
                /**
                 * Realiza las operaciones de arranque para la aplicación.
                 *
                 * @return void
                 */
                public function boot(Router $router)
                {
                    $router->aliasMiddleware('role', \Spatie\Permission\Middleware\RoleMiddleware::class);
                }
            }

```

En el modelo User, debemos agregar la siguiente directiva:

```
use Spatie\Permission\Traits\HasRoles;

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



```
EXPLORADOR ... .env app.js User.php X AppServiceProvider.php
└─ AGROLANZA
    └─ app
        ├─ Http
        └─ Models
            └─ User.php
                └─ Providers
                    └─ AppServiceProvider...
                > View
                > bootstrap
                > config
                > database
                > lang
                > node_modules
                > public
                └─ resources
User.php
<?php
namespace App\Models;
// use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Spatie\Permission\Traits\HasRoles;
class User extends Authenticatable
{
    /** @use HasFactory<\Database\Factories\UserFactory>
     * use HasFactory, Notifiable;
    use HasRoles;
}
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
```

De esta forma, podremos configurar los permisos y roles, y asignarlos a los usuarios registrados en el sistema.

A continuación, se muestra un ejemplo de crear rol y permisos, y asociarlos a un usuario registrado previamente:

```
$ php artisan tinker
Psy Shell v0.12.7 (PHP 8.3.6 — cli) by Justin Hileman
> use Spatie\Permission\Models\Role;
> use Spatie\Permission\Models\Permission;
> $admin = Role::create(['name' => 'admin']);
= Spatie\Permission\Models\Role {#5344
    guard_name: "web",
    name: "admin",
    updated_at: "2025-02-27 22:53:53",
    created_at: "2025-02-27 22:53:53",
    id: 1,
}

> $crearUsuario = Permission::create(['name' => 'listar usuarios']);
= Spatie\Permission\Models\Permission {#5359
    guard_name: "web",
    name: "listar usuarios",
    updated_at: "2025-02-27 22:56:41",
    created_at: "2025-02-27 22:56:41",
    id: 1,
}

> $editarUsuario = Permission::create(['name' => 'editar usuarios']);
= Spatie\Permission\Models\Permission {#5320
    guard_name: "web",
```



```
name: "editar usuarios",
updated_at: "2025-02-27 22:56:59",
created_at: "2025-02-27 22:56:59",
id: 2,
}

> $eliminarUsuario = Permission::create(['name' => eliminar usuarios']);
= Spatie\Permission\Models\Permission {#5485
    guard_name: "web",
    name: "eliminar usuarios",
    updated_at: "2025-02-27 22:57:31",
    created_at: "2025-02-27 22:57:31",
    id: 3,
}

> $admin->givePermissionTo($crearUsuario, $editarUsuario, $eliminarUsuario);
= Spatie\Permission\Models\Role {#5344
    guard_name: "web",
    name: "admin",
    updated_at: "2025-02-27 22:53:53",
    created_at: "2025-02-27 22:53:53",
    id: 1,
}
```

Ahora es momento de asignar a un usuario existente el rol de admin:

```
> $user= User::where('email', 'alexisgilcabrera@gmail.com')->first();
[!] Aliasing 'User' to 'App\Models\User' for this Tinker session.
= App\Models\User {#6310
    id: 1,
    name: "Alexis",
    email: "alexisgilcabrera@gmail.com",
    email_verified_at: null,
    #password: "xxx",
    #remember_token: null,
    created_at: "2025-02-27 23:05:11",
    updated_at: "2025-02-27 23:05:11",
}

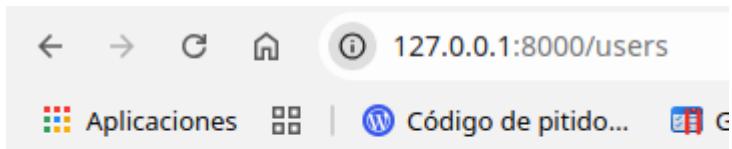
> if ($user) {
.   $user->assignRole('admin');
. } else {
.   echo "Usuario no encontrado.";
. }
= App\Models\User {#6310
    id: 1,
    name: "Alexis",
    email: "alexisgilcabrera@gmail.com",
```



```
email_verified_at: null,  
#password: "xxx",  
#remember_token: null,  
created_at: "2025-02-27 23:05:11",  
updated_at: "2025-02-27 23:05:11",  
}
```

Agregamos las siguientes líneas en web.php, para comprobar si funciona el acceso por roles:

```
Route::middleware(['auth', 'role:admin'])->group(function () {  
    Route::get('/users', function () {  
        return view('acceso');  
    })->middleware(['auth', 'verified'])->name('users.listado');  
});
```



Si a role cambiamos por otro valor, por ejemplo ‘administrador’, no nos debería dejar acceder a dicho sitio:

```
Route::middleware(['auth', 'role:administrador'])->group(function () {  
    Route::get('/users', function () {  
        return view('acceso');  
    })->middleware(['auth', 'verified'])->name('users.listado');  
});
```



127.0.0.1:8000/users

Aplicaciones | Código de pitido... | Glosario Informa... | Reparar un chip... | Z370M DS3H (rev... | ULPGC - Titulacio...

403 | USER DOES NOT HAVE THE RIGHT ROLES.

De manera muy eficiente, hemos implementado el rol de admin a un usuario. Para esta plataforma, se aplicará desde el fichero **RolesPermissionsSeeder** un modelo de permisos y roles como el que se mostrará en la siguiente tabla:

		Roles			
		admin	auxiliar	conductor	aplicador
Servicios	listar servicios				
	crear servicios				
	editar servicios				
Clientes	listar clientes				
	crear clientes				
	editar clientes				
Parcelas	listar parcelas				
	crear parcelas				
	editar parcelas				
Maquinarias	listar maquinarias				
	crear maquinarias				
	editar maquinarias				
Fitosanitarios	listar fitosanitarios				
	crear fitosanitarios				
	editar fitosanitarios				



Usuarios	listar usuarios	green	red	red	red
	crear usuarios	green	red	red	red
	editar usuarios	green	red	red	red

Estructura de modelos

El proyecto se compone de 6 modelos, diseñados para las 6 entidades relacionadas en base de datos. En cada una de ellas se especifica el tipo de relación entre las mismas.

➤ User

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Spatie\Permission\Traits\HasRoles;

class User extends Authenticatable
{
    /**
     * @use HasFactory<Database\Factories\UserFactory>
     */
    use HasFactory, Notifiable;
    use HasRoles;

    /**
     * The attributes that are mass assignable.
     *
     * @var list<string>
     */

    protected $fillable = [
        'name',
        'email',
        'password',
    ];
    /**
     * The attributes that should be hidden for serialization.
     *
     * @var list<string>
     */

    protected $hidden = [

```



```
'password',
'remember_token',
];
/**
 * Get the attributes that should be cast.
 *
 * @return array<string, string>
 */
protected function casts(): array
{
    return [
        'email_verified_at' => 'datetime',
        'password' => 'hashed',
    ];
}

public function servicios()
{
    return $this->belongsToMany(Servicio::class, 'servicios_users', 'id_user', 'id_servicio');
}
}
```

➤ Cliente

```
<?php
namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Cliente extends Model
{
    protected $guarded = ['id'];
    public function parcelas(): HasMany
    {
        return $this->hasMany(Parcela::class);
    }
}
```



➤ Parcela

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Model;
use Illuminate\Database\Eloquent\Relations\HasMany;

class Parcela extends Model
{
    protected $guarded = ['id'];

    public function cliente()
    {
        return $this->belongsTo(Cliente::class, 'id_cliente');
    }

    public function servicios(): HasMany
    {
        return $this->hasMany(Servicio::class);
    }
}
```

➤ Servicio

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;

class Servicio extends Model
{
    protected $guarded = ['id'];
    const TIPOS_SERVICIO = [
        'CP' => 'Control de plagas',
        'CM' => 'Control de maleza',
        'SC' => 'Sembrado por cazolejas',
    ];
}
```



```
'SS' => 'Sembrado por surcos',
'CF' => 'Cosecha de frutos rastreros',
'RF' => 'Recolecta de frutos arbóreos',
];
const METODOS_PAGO = [
    'EF' => 'Efectivo',
    'TA' => 'Tarjeta',
    'TR' => 'Transferencia'
];
const ESTADOS = [
    'P' => 'Pagado',
    'N' => 'No pagado',
];
public function parcela()
{
    return $this->belongsTo(Parcela::class, 'id_parcela');
}
public function fitosanitarios()
{
    return $this->hasMany(Fitosanitario::class, 'servicios_fitosanitarios', 'id_servicio',
'id_fitosanitario');
}
public function maquinarias()
{
    return $this->hasMany(Maquinaria::class, 'servicios_maquinarias', 'id_servicio',
'id_maquinaria');
}
public function users()
{
    return $this->hasMany(User::class, 'servicios_users', 'id_servicio', 'id_user');
}
```



➤ Maquinaria

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Maquinaria extends Model
{
    protected $guarded = ['id'];
    public function servicios()
    {
        return $this->belongsToMany(Servicio::class, 'servicios_maquinarias', 'id_maquinaria',
'id_servicio');
    }
}
```

➤ Fitosanitario

```
<?php
namespace App\Models;
use Illuminate\Database\Eloquent\Model;
class Fitosanitario extends Model
{
    protected $guarded = ['id'];
    const TIPOS = [
        'HE' => 'Herbicida',
        'IN' => 'Insecticida',
        'FU' => 'Fungicida',
        'BA' => 'Bactericida',
        'NE' => 'Nematicida',
        'AC' => 'Acaricida'
    ];
    const ESTADOS = [
        'PV' => 'Sólido - Líquido',
        'PP' => 'Sólido - Sólido',
        'VV' => 'Líquido - Líquido'
    ];
    public function servicios()
    {
        return $this->belongsToMany(Servicio::class, 'servicios_fitosanitarios', 'id_fitosanitario',
'id_servicio');
    }
}
```



Migraciones

A continuación, se muestra el contenido de los ficheros de migraciones de Laravel, respetando las tablas en SQL que se mostraron previamente en la estructura de base de datos.

- Usuarios (equivalente a empleados de la organización)

```
Schema::create('users', function (Blueprint $table) {  
    $table->id();  
    $table->char('identificacion', 10)->unique();  
    $table->string('name', 30);  
    $table->string('apellidos', 60);  
    $table->string('email', 40)->unique();  
    $table->char('telefono', 12);  
    $table->string('direccion', 150);  
    $table->string('localidad', 50);  
    $table->char('codigo_postal', 5);  
    $table->char('iban', 24);  
    $table->date('fecha_nacimiento');  
    $table->date('fecha_comienzo');  
    $table->date('fecha_fin')->nullable();  
    $table->string('password');  
    $table->rememberToken();  
  
    $table->string('ip_alta', 15)->nullable();  
    $table->string('ip_ult_mod', 15)->nullable();  
    $table->timestamps();  
});
```

- Clientes

```
Schema::create('clientes', function (Blueprint $table) {  
    $table->id();  
    $table->char('identificacion', 10)->unique();  
    $table->string('nombre', 30);  
    $table->string('apellidos', 60);  
    $table->string('email', 40)->nullable();
```



```
$table->char('telefono', 12);
$table->char('codigo_postal', 5);

$table->string('ip_alta', 15)->nullable();
$table->string('ip_ult_mod', 15)->nullable();
$table->timestamps();
});
```

➤ Parcelas

```
Schema::create('parcelas', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('id_cliente');
    $table->char('referencia_catastral', 20)->unique();
    $table->unsignedInteger('superficie');
    $table->decimal('latitud', 10, 6);
    $table->decimal('longitud', 10, 6);

    $table->string('ip_alta', 15)->nullable();
    $table->string('ip_ult_mod', 15)->nullable();
    $table->timestamps();

    $table->foreign('id_cliente')->references('id')->on('clientes');
});
```

➤ Maquinarias

```
Schema::create('maquinarias', function (Blueprint $table) {
    $table->id();
    $table->string('numero_serie', 20);
    $table->char('matricula', 10);
    $table->string('marca', 20);
    $table->string('descripcion', 200);

    $table->ipAddress('ip_alta')->nullable();
```



```
$table->ipAddress('ip_ult_mod')->nullable();  
  
$table->timestamps();  
});
```

➤ Fitosanitarios

```
Schema::create('fitosanitarios', function (Blueprint $table) {  
    $table->id();  
    $table->string('nombre', 29);  
    $table->string('marca', 29);  
    $table->string('numero_registro', 29);  
    $table->char('tipo', 2);  
    $table->char('estado', 2);  
    $table->string('descripcion', 200);  
    $table->string('entidad_vendedora', 40);  
    $table->string('materia_activa', 30);  
    $table->decimal('cantidad_materia_activa', 5, 2);  
    $table->decimal('dosis_maxima', 5, 2);  
    $table->integer('plazo_seguridad');  
    $table->string('observaciones', 200)->nullable();  
  
    $table->string('ip_alta', 15)->nullable();  
    $table->string('ip_ult_mod', 15)->nullable();  
  
    $table->timestamps();  
});
```

➤ Servicios

```
Schema::create('servicios', function (Blueprint $table) {  
    $table->id();  
    $table->enum('tipo_servicio', ['CP', 'CM', 'SC', 'SS']);  
    $table->string('descripcion', 60);  
    $table->date('fecha_servicio');
```



```
$table->time('hora_servicio');
$table->integer('duracion');
$table->decimal('presupuesto', 7, 2);
$table->enum('metodo_pago', ['EF', 'TA', 'TR']);
$table->enum('estado', ['P', 'N']);
$table->unsignedBigInteger('id_parcela');

$table->string('ip_alta', 15)->nullable();
$table->string('ip_ult_mod', 15)->nullable();

$table->timestamps();

$table->foreign('id_parcela')->references('id')->on('parcelas');
});
```

➤ Servicios Usuarios

```
Schema::create('servicios_users', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('id_servicio');
    $table->unsignedBigInteger('id_user');

    $table->string('ip_alta', 15)->nullable();
    $table->string('ip_ult_mod', 15)->nullable();

    $table->timestamps();

    $table->foreign('id_servicio')->references('id')->on('servicios')->onDelete('cascade');
    $table->foreign('id_user')->references('id')->on('users')->onDelete('cascade');
});
```

➤ Servicios Maquinarias

```
Schema::create('servicios_maquinarias', function (Blueprint $table) {
    $table->id();
```



```
$table->unsignedBigInteger('id_servicio');
$table->unsignedBigInteger('id_maquinaria');

$table->string('ip_alta', 15)->nullable();
$table->string('ip_ult_mod', 15)->nullable();

$table->timestamps();

$table->foreign('id_servicio')->references('id')->on('servicios')->onDelete('cascade');
$table->foreign('id_maquinaria')->references('id')->on('maquinarias')->onDelete('cascade');
});
```

➤ ServiciosFitosanitarios

```
Schema::create('servicios_fitosanitarios', function (Blueprint $table) {
    $table->id();
    $table->unsignedBigInteger('id_servicio');
    $table->unsignedBigInteger('id_fitosanitario');

    $table->string('ip_alta', 15)->nullable();
    $table->string('ip_ult_mod', 15)->nullable();

    $table->timestamps();

    $table->foreign('id_servicio')->references('id')->on('servicios')->onDelete('cascade');
    $table->foreign('id_fitosanitario')->references('id')->on('fitosanitarios')->onDelete('cascade');
});
```

*Las tablas de roles y permisos son las que se generan por defecto al instalar el paquete [Spatie Laravel Permission](#).



Implementación de mapa interactivo

En un proyecto sobre gestión de parcelas u otro tipo de proyectos similares, es fundamental la incorporación de un buen mapa, el cual sea capaz de cargar datos rápidamente y sin ocupar mucha memoria temporal. Existen diversas APIs enfocadas en esta tarea, pero muchas de ellas funcionan mediante pagos mensuales. Entonces, una gran opción que cumpla todos estos requerimientos y sea gratuita es [Leafletjs](#). Esta API es un mapa interactivo que funciona en lenguaje Javascript, y aplica “Lazy Loading”.

Lazy Loading es una técnica de programación la cual retrasa la carga de recursos hasta que sean necesarios, lo cual hace que en el momento de llamar la API no colapse al cliente. Únicamente solicitará al servidor aquellos recursos que se necesiten en el momento, y a medida que el usuario se vaya moviendo por el mapa, se solicitará al servidor los recursos de la zona seleccionada en cuestión.

Para importarlo en nuestro proyecto y hacer uso del mapa interactivo, simplemente debemos crear un fichero html (o, en este caso, una vista) y un fichero Javascript, y agregar el enlace a los recursos (también se debe hacer lo mismo con JQuery, ya que Leafletjs lo necesita). Si queremos consultar el mapa por coordenadas, podemos crear dos inputs, y un script:

```
<div class="mb-3 row">
  <div class="col-md-6">
    <label for="latitud" class="form-label">Latitud</label>
    <input type="number" step="0.000001" name="latitud" class="form-control" id="latitud"
value="{{ old('latitud', $parcela->latitud ?? '') }}" placeholder="Latitud">
    @error('latitud') <p style="color: red;">{{ $message }}</p> @enderror
  </div>
  <div class="col-md-6">
    <label for="longitud" class="form-label">Longitud</label>
    <input type="number" step="0.000001" name="longitud" class="form-control" id="longitud"
value="{{ old('longitud', $parcela->longitud ?? '') }}" placeholder="Longitud">
    @error('longitud') <p style="color: red;">{{ $message }}</p> @enderror
  </div>
</div>

<div id="show_map" style="height: 700px; margin-top: 50px;"></div>
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css" />
<script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js"></script>
<script src="scripts.js"></script>
<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.3/dist/leaflet.css" />
<script src="https://unpkg.com/leaflet@1.9.3/dist/leaflet.js"></script>
<script src="scripts.js"></script> #Cargar scripts que se verán a continuación
```



```
document.addEventListener("DOMContentLoaded", function () {
    var latitud = parseFloat(document.getElementById("latitud").value) || 29.015526;
    var longitud = parseFloat(document.getElementById("longitud").value) || -13.614990;
    var map = L.map('show_map').setView([latitud, longitud], 15);
    L.tileLayer('https://s.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        maxZoom: 18,
    }).addTo(map);
    var marker = L.marker([latitud, longitud], {draggable: true}).addTo(map)
        .bindPopup("Arrastra el marcador o haz clic en el mapa")
        .openPopup();
    marker.on("dragend", function (event) {
        var position = marker.getLatLng();
        document.getElementById("latitud").value = position.lat.toFixed(6);
        document.getElementById("longitud").value = position.lng.toFixed(6);
    });
    map.on("click", function (event) {
        var lat = event.latlng.lat.toFixed(6);
        var lng = event.latlng.lng.toFixed(6);
        document.getElementById("latitud").value = lat;
        document.getElementById("longitud").value = lng;
        marker.setLatLng([lat, lng]);
    });
    document.getElementById("latitud").addEventListener("change", updateMarker);
    document.getElementById("longitud").addEventListener("change", updateMarker);
    function updateMarker() {
        var newLat = parseFloat(document.getElementById("latitud").value) || latitud;
        var newLon = parseFloat(document.getElementById("longitud").value) || longitud;
        marker.setLatLng([newLat, newLon]);
        map.setView([newLat, newLon], 15);
    }
});
```

Controladores

En Laravel, un **controlador** es una clase que maneja la lógica para las peticiones HTTP dirigidas a tu aplicación. Actúa como intermediario entre el modelo y la vista, gestionando cómo se deben procesar los datos y qué respuestas devolver al usuario.

En este proyecto, existen los siguientes controladores:

- ClienteController. Se gestionan los clientes.
- FitosanitarioController. Se gestionan los productos fitosanitarios.



- **GestionServicioController.** Se incluye la gestión de usuarios, fitosanitarios y maquinarias en un servicio.
- **MaquinariaController.** Se gestionan las maquinarias.
- **ParcelaController.** Se gestionan las parcelas de los clientes.
- **PDFController.** Procesa la petición, redirigiendo a la vista con el contenido a generar en formato PDF
- **ProfileController.** Se genera por defecto al implementar la gestión de perfiles en la plataforma
- **ServicioController.** Se gestionan los servicios.
- **UserController.** Se gestionan los usuarios del sistema.

A continuación, se muestra el código del controlador **ServicioController**, como ejemplo de la estructura del mismo:

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Models\Parcela;
use App\Models\Servicio;
use App\Models\Cliente;

class ServicioController extends Controller
{
    #Función para listar los servicios
    function listado()
    {
        $servicios = Servicio::with('parcela')->paginate(10);
        #Se cargan las traducciones mediante el modelo Servicio
        $TIPOS_SERVICIO = Servicio::TIPOS_SERVICIO;
        $METODOS_PAGO = Servicio::METODOS_PAGO;
        $ESTADOS = Servicio::ESTADOS;
        return view('servicios.servicio', compact('servicios', 'TIPOS_SERVICIO',
        'METODOS_PAGO', 'ESTADOS'));
    }
    #Función para cargar el formulario de servicios
    #El parámetro $oper indica la operación a realizar (alta, modificación, consulta o
    eliminación)
    #El parámetro $id indica el id del servicio a modificar, consultar o eliminar
    #Si no se indica el id, se carga un formulario en blanco para dar de alta un nuevo servicio
    function formulario($oper="", $id="")
    {
        $servicio = empty($id) ? new Servicio() : Servicio::find($id);
```



```
$parcelas = Parcela::all();
$TIPOS_SERVICIO = Servicio::TIPOS_SERVICIO;
$METODOS_PAGO = Servicio::METODOS_PAGO;
$ESTADOS = Servicio::ESTADOS;

return view('servicios.formulario',compact('TIPOS_SERVICIO', 'METODOS_PAGO',
'ESTADOS','servicio', 'oper', 'parcelas'));
}

#Función para cargar el formulario en modo consulta
function mostrar($id)
{
    return $this->formulario('cons', $id);
}

#Función para cargar el formulario en modo edición
function actualizar($id)
{
    return $this->formulario('modi', $id);
}

#Función para cargar el formulario en modo de eliminación
function eliminar($id)
{
    return $this->formulario('supr', $id);
}

#Función para cargar el formulario en modo alta
function alta()
{
    return $this->formulario();
}

#Función para almacenar o eliminar el servicio en la base de datos
function almacenar(Request $request)
{
    #Si oper es igual a 'supr', se eliminará el servicio
if ($request->oper == 'supr')
{
    $servicio = Servicio::find($request->id);
    $servicio->delete();
    $salida = redirect()->route('servicios.listado');
}
else
    #Si oper es diferente de 'supr', se almacenará el servicio
{
```



```
#Se valida que el tipo de servicio sea válido
$validacion_tipo_servicio = "";
foreach(Servicio::TIPOS_SERVICIO as $codigo_tipo_servicio => $texto_tipo_servicio)
{
    $validacion_tipo_servicio .= $codigo_tipo_servicio .';
}
$validacion_tipo_servicio = substr($validacion_tipo_servicio,0,-1);
#Se valida que el método de pago sea válido
$validacion_metodo_pago = "";
foreach(Servicio::METODOS_PAGO as $codigo_metodo_pago =>
$texto_metodo_pago)
{
    $validacion_metodo_pago .= $codigo_metodo_pago .';
}
$validacion_metodo_pago = substr($validacion_metodo_pago,0,-1);
#Se valida que el estado sea válido
$validacion_estado = "";
foreach(Servicio::ESTADOS as $codigo_estado => $texto_estado)
{
    $validacion_estado .= $codigo_estado .';
}
$validacion_estado = substr($validacion_estado,0,-1);
#Se validan los campos del formulario
$validatedData = $request->validate([
    #Se valida que el tipo de servicio sea válido
    'tipo_servicio' => 'required|in:'.$validacion_tipo_servicio,
    #Se valida que la descripción sea obligatoria y de tipo cadena de texto, que no
    exceda de 60 caracteres
    'descripcion' => 'required|string|max:60',
    #Se valida que la fecha del servicio sea obligatoria y de tipo fecha
    'fecha_servicio'=> 'required|date',
    #Se valida que la hora del servicio sea obligatoria
    'hora_servicio' => 'required',
    'duracion'      => 'required|integer',
    #Se valida que el presupuesto sea obligatorio y de tipo numérico, que no exceda de
    5 dígitos y que tenga 1 o 2 decimales opcionales
    'presupuesto'  => 'required|numeric|regex:/^[\d]{0,5}(\.[\d]{1,2})?$/',
    #Se valida que el método de pago sea válido
    'metodo_pago'  => 'required|in:'.$validacion_metodo_pago,
    #Se valida que el estado sea válido
    'estado'        => 'required|in:'.$validacion_estado,
```



```
#Se valida que la parcela sea obligatoria y que exista en la base de datos
'id_parcela' => 'required|exists:parcelas,id',
], [
    'tipo_servicio.required' => 'El tipo de servicio es obligatorio.',
    'descripcion.required' => 'La descripción es obligatoria.',
    'descripcion.string' => 'Debe ser de tipo cadena de texto.',
    'descripcion.max' => 'Máximo 255 caracteres',
    'fecha_servicio.required' => 'La fecha es obligatoria.',
    'fecha_servicio.date' => 'Debe ser de tipo fecha.',
    'hora_servicio.required' => 'La hora es obligatoria.',
    'duracion.required' => 'La duración es obligatoria.',
    'duracion.integer' => 'Debe ser de tipo numérico entero.',
    'presupuesto.required' => 'El importe es obligatorio.',
    'presupuesto.numeric' => 'Debe ser de tipo numérico.',
    'presupuesto.regex' => 'El formato admitido es un numero positivo entero de
maximo 5 digitos con decimales opcionales 1 o 2 como máximo',
    'metodo_pago.required' => 'El método de pago es obligatorio.',
    'estado.required' => 'El estado es obligatorio.',
    'id_parcela.required' => 'La parcela es obligatoria.',
    'id_parcela.exists' => 'La parcela seleccionada no existe.',
]);
#Se valida que el id de la parcela exista en la base de datos
$servicio = empty($request->id)? new Servicio() : Servicio::find($request->id);
#Se asignan los valores del formulario a los atributos del servicio
$servicio->tipo_servicio = $request->tipo_servicio;
$servicio->descripcion = $request->descripcion;
$servicio->fecha_servicio = $request->fecha_servicio;
$servicio->hora_servicio = $request->hora_servicio;
$servicio->duracion = $request->duracion;
$servicio->presupuesto = $request->presupuesto;
$servicio->metodo_pago = $request->metodo_pago;
$servicio->estado = $request->estado;
$servicio->id_parcela = $request->id_parcela;
$servicio->ip_alta = $request->ip();
#Se almacena en la base de datos
$servicio->save();
#Se redirige al formulario con un mensaje de éxito
$salida = redirect()->route('servicios.alta')->with([
    'success' => 'Servicio insertado correctamente.'
    , 'formData' => $servicio
])
]
```



```
    );
}

return $salida;
}

}
```

Los demás controladores presentan una estructura similar. Aplicando el **enrutamiento** (en routes/web.php) podemos asociar los controladores a las respectivas vistas.

Enrutamiento

El fichero “routes/web.php” es el archivo donde defines las rutas web de tu aplicación, es decir, las URLs que el navegador puede visitar. Cada ruta indica qué controlador y método deben manejar la solicitud. A continuación, un ejemplo sobre la entidad Fitosanitarios, aplicando el acceso a las rutas mediante roles:

```
Route::middleware(['auth', 'role:admin|auxiliar|aplicador|conductor'])->group(function () {
    Route::get('/fitosanitarios' , [FitosanitarioController::class, 'listado'])
        ->middleware(['auth', 'verified'])
        ->name('fitosanitarios.listado');

    Route::get('/fitosanitario/{id}' , [FitosanitarioController::class, 'mostrar'])
        ->middleware(['auth', 'verified'])
        ->name('fitosanitarios.mostrar');

    Route::get('fitosanitarios/pdf' , [PDFController::class, 'pdf_fitosanitarios'])
        ->name('fitosanitarios.pdf_fitosanitarios');

    Route::get('/fitosanitario/pdf/{id}' , [PDFController::class, 'pdf_fitosanitario'])
        ->name('fitosanitarios.pdf_fitosanitario');
});

Route::middleware(['auth', 'role:admin|auxiliar'])->group(function () {
    Route::get('/fitosanitario/actualizar/{id}' , [FitosanitarioController::class, 'actualizar'])
        ->middleware(['auth', 'verified'])
        ->name('fitosanitarios.actualizar');

    Route::get('/fitosanitario/eliminar/{id}' , [FitosanitarioController::class, 'eliminar'])
        ->middleware(['auth', 'verified'])
        ->name('fitosanitarios.eliminar');

    Route::get('/fitosanitarios/nuevo' , [FitosanitarioController::class, 'alta'])
        ->middleware(['auth', 'verified'])
        ->name('fitosanitarios.alta');

    Route::post('/fitosanitarios/nuevo' , [FitosanitarioController::class, 'almacenar'])
        ->middleware(['auth', 'verified'])
        ->name('fitosanitarios.almacenar');
});
```



Implementación de generador PDF

Muchas veces utilizamos documentos en formato PDF, ya que son muy útiles para compartir información con otras personas, para poder almacenarlo en cualquier memoria y/o almacenamiento remoto (nube), o inclusive para poder imprimir en un tamaño adecuado al papel. Es por tanto, que puede ser muy interesante agregar esta opción al proyecto.

Una herramienta de generación de PDF en Laravel muy popular es [DomPDF](#), gratuito y de código abierto a los usuarios. La forma de instalar esta herramienta es la siguiente:

```
composer require barryvdh/laravel-dompdf
```

```
php artisan make:controller PDFController
```

Introducir el siguiente código en el controlador:

```
<?php  
namespace App\Http\Controllers;  
use Illuminate\Http\Request;  
use App\Models\User;  
use PDF;  
  
class PDFController extends Controller  
{  
    /**  
     * Display a listing of the resource.  
     *  
     * @return \Illuminate\Http\Response  
     */  
    public function generatePDF()  
    {  
        $users = User::get();  
  
        $data = [  
            'title' => 'Welcome to ItSolutionStuff.com',  
            'date' => date('m/d/Y'),  
            'users' => $users  
        ];  
  
        $pdf = PDF::loadView('myPDF', $data);  
        return $pdf->download('itsolutionstuff.pdf');  
    }  
}
```



Pruebas unitarias

Una prueba unitaria es un bloque de código que verifica la precisión de un bloque más pequeño y aislado de código de aplicación, normalmente una función o un método. La prueba unitaria está diseñada para verificar que el bloque de código se ejecuta según lo esperado, de acuerdo con la lógica teórica del desarrollador.

En Laravel, se pueden hacer dichas pruebas utilizando PHPUnit o Pest. La principal diferencia entre los dos frameworks es que PHPUnit se centra en garantizar que el código esté libre de errores, mientras que Pest desempeña esta tarea, pero con un código más conciso y elegante.

Al comienzo del proyecto, hemos elegido Pest como framework de pruebas, por lo que ya se encuentra listo para ser utilizado. Vamos a crear un fichero de pruebas para la creación de un usuario en el sistema:

```
php artisan pest:test UserTest
```

Se creará el fichero “tests/Feature/UserTest”. En dicho fichero, vamos a definir los siguientes aspectos:

- Validación exitosa con datos válidos
- Falla con identificación inválido
- Falla si el usuario es menor de edad (fecha_nacimiento)
- Falla si confirm_password no coincide con password
- Falla si name excede los 50 caracteres
- Falla si apellidos está vacío
- Falla si email tiene formato inválido
- Falla si telefono no tiene 9 dígitos
- Falla si dirección supera los 255 caracteres
- Falla si localidad supera los 100 caracteres
- Falla si codigo_postal no tiene 5 dígitos
- Falla si el IBAN no tiene el formato correcto
- Validación exitosa al actualizar sin password ni confirm_password

A continuación se muestra el código con las pruebas unitarias:

```
<?php

use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Str;
use Carbon\Carbon;

function getValidationRules($request = [])
{
```



```
return [
    'identificacion' => ['required',
'regex:/^([X-Z])?\d{8}[A-Z]$/', 'unique:users,identificacion', .
($request['id'] ?? '')],
    'name' => 'required|string|max:50',
    'apellidos' => 'required|string|max:50',
    'email' => 'required|string|email|max:255|unique:users,email,' .
($request['id'] ?? ''),
    'telefono' => ['required', 'regex:/^\d{9}$/'],
    'direccion' => 'required|string|max:255',
    'localidad' => 'required|string|max:100',
    'codigo_postal' => 'required|regex:/^\d{5}$/,
    'iban' => 'required|regex:/^ES[0-9]{22}$/,
    'fecha_nacimiento' => 'required|date|before:' .
Carbon::now()->subYears(18)->toDateString(),
    'password' => empty($request['id']) ? 'required' :
(!empty($request['password']) ? 'string' : ''),
    'confirm_password' => !empty($request['password']) ?
'required|same:password' : '',
];
}

it('passes validation with valid data', function () {
$validData = [
    'identificacion' => '12345678Z',
    'name' => 'Juan',
    'apellidos' => 'Pérez',
    'email' => 'juan@example.com',
    'telefono' => '612345678',
    'direccion' => 'Calle Falsa 123',
    'localidad' => 'Madrid',
    'codigo_postal' => '28080',
    'iban' => 'ES7620770024003102575766',
    'fecha_nacimiento' =>
Carbon::now()->subYears(20)->toDateString(),
    'password' => 'secret123',
    'confirm_password' => 'secret123',
];
}

$validator = Validator::make($validData, getValidationRules());
expect($validator->passes())->toBeTrue();
});
```



```
it('fails validation with invalid identificacion', function () {
    $data = ['identificacion' => '1234A'];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('identificacion'))->toBeTrue();
});

it('fails when user is underage', function () {
    $data = ['fecha_nacimiento' =>
Carbon::now()->subYears(17)->toDateString()];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('fecha_nacimiento'))->toBeTrue();
});

it('fails if confirm_password does not match', function () {
    $data = [
        'password' => 'secret123',
        'confirm_password' => 'notmatching',
    ];
    $validator = Validator::make($data, getValidationRules($data));
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('confirm_password'))->toBeTrue();
});

it('fails validation with invalid name (too long)', function () {
    $data = ['name' => str_repeat('a', 51)];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('name'))->toBeTrue();
});

it('fails validation with invalid apellidos (empty)', function () {
    $data = ['apellidos' => ''];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('apellidos'))->toBeTrue();
});

it('fails validation with invalid email format', function () {
    $data = ['email' => 'invalid-email'];
})
```



```
validator = Validator::make($data, getValidationRules());
expect($validator->fails())->toBeTrue();
expect($validator->errors()->has('email'))->toBeTrue();
}) ;

it('fails validation with invalid telefono', function () {
    $data = ['telefono' => '12345'];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('telefono'))->toBeTrue();
}) ;

it('fails validation with invalid direccion (too long)', function () {
    $data = ['direccion' => str_repeat('b', 256)];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('direccion'))->toBeTrue();
}) ;

it('fails validation with invalid localidad (too long)', function () {
    $data = ['localidad' => str_repeat('c', 101)];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('localidad'))->toBeTrue();
}) ;

it('fails validation with invalid codigo_postal', function () {
    $data = ['codigo_postal' => '123'];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('codigo_postal'))->toBeTrue();
}) ;

it('fails validation with invalid IBAN', function () {
    $data = ['iban' => 'ES123456'];
    $validator = Validator::make($data, getValidationRules());
    expect($validator->fails())->toBeTrue();
    expect($validator->errors()->has('iban'))->toBeTrue();
}) ;
```



```
it('passes validation without password when updating and password not set', function () {
    $data = [
        'id' => 1,
        'identificacion' => '12345678Z',
        'name' => 'Ana',
        'apellidos' => 'López',
        'email' => 'ana@example.com',
        'telefono' => '612345678',
        'direccion' => 'Calle Nueva 45',
        'localidad' => 'Barcelona',
        'codigo_postal' => '08001',
        'iban' => 'ES7620770024003102575766',
        'fecha_nacimiento' =>
Carbon::now()->subYears(25)->toDateString(),
    ];
    $validator = Validator::make($data, getValidationRules($data));
    expect($validator->passes())->toBeTrue();
});
```

Si el código es correcto, debe pasar las 13 pruebas. Ejecutamos la siguiente sentencia:

```
./vendor/bin/pest tests/Feature/UserTest.php
```

```
PASS Tests\Feature\UserTest
✓ it passes validation with valid data 5.01s
✓ it fails validation with invalid identificacion 0.02s
✓ it fails when user is underage 0.02s
✓ it fails if confirm_password does not match 0.02s
✓ it fails validation with invalid name (too long) 0.02s
✓ it fails validation with invalid apellidos (empty) 0.02s
✓ it fails validation with invalid email format 0.06s
✓ it fails validation with invalid telefono 0.04s
✓ it fails validation with invalid direccion (too long) 0.02s
✓ it fails validation with invalid localidad (too long) 0.02s
✓ it fails validation with invalid codigo_postal 0.02s
✓ it fails validation with invalid IBAN 0.01s
✓ it passes validation without password when updating and password not set 0.02s
```

Tests: **13 passed** (24 assertions)

Duration: **5.42s**

El resto de baterías de pruebas se puede encontrar en la ruta **tests/Features**.



Montaje del proyecto mediante servidor web

Hasta ahora siempre se ha utilizado el comando “`php artisan serve`”. Dicho comando lanza un servidor embebido de PHP (por defecto en el puerto 8000), pero para hacerlo funcionar en red de forma más estable y segura, es necesario utilizar un servidor web. Junto a este, es indispensable incluir un **certificado SSL**.

Como servidor web vamos a utilizar Apache 2, servidor web muy popular y estandarizado en el mercado. Citamos algunas de sus características que lo hacen destacar entre otros programas dedicados al servicio web:

- Gratuito y de código abierto
- Modular y configurable
- Soporte para múltiples lenguajes y tecnologías
- Reescritura de URLs amigables
- Soporte SSL/TLS
- Posibilidad de alojar múltiples sitios web en la máquina física
- Alta compatibilidad
- Amplio registro de logs

Para proceder con la instalación y la vinculación del proyecto con el servidor web, llevaremos a cabo una serie de pasos. En primer lugar, procedemos a instalarlo:

```
sudo apt install apache2 libapache2-mod-php -y
```

Es muy interesante hacer funcionar el cortafuegos o firewall, abriendo únicamente aquellos puertos que necesitamos. En este caso debemos abrir el puerto 443, destinado al protocolo HTTPS (HTTP con cifrado SSL):

```
sudo ufw enable && sudo ufw allow 443
```

```
alexisg@samsung:~$ sudo ufw enable
El cortafuegos está activo y habilitado en el arranque del sistema
alexisg@samsung:~$ sudo ufw allow 80
Regla añadida
Regla añadida (v6)
alexisg@samsung:~$ sudo ufw allow 443
Regla añadida
Regla añadida (v6)
alexisg@samsung:~$ sudo ufw status
Estado: activo

Hasta          Acción      Desde
----          ----      -----
80            ALLOW      Anywhere
443           ALLOW      Anywhere
80 (v6)       ALLOW      Anywhere (v6)
443 (v6)     ALLOW      Anywhere (v6)

alexisg@samsung:~$ 1~|
```

Movemos el proyecto a la ruta `/var/www/htmlagrolanza/`, desde la cual Apache tendrá permisos para acceder a los recursos que se encuentran alojados.



```
alexisg@samsung:/var/www/html$ sudo chown -R www-data:www-data agrolanza/
alexisg@samsung:/var/www/html$ ls -la
total 12
drwxrwxrwx  3 root      root      4096 abr 24 23:07 .
drwxr-xr-x  3 root      root      4096 ene 21 22:52 ..
drwxrwxr-x 14 www-data www-data 4096 abr 24 23:41 agrolanza
alexisg@samsung:/var/www/html$ ls -la agrolanza/
total 596
drwxrwxr-x 14 www-data www-data 4096 abr 24 23:41 .
drwxrwxrwx  3 root      root      4096 abr 24 23:07 .
drwxrwxr-x  6 www-data www-data 4096 feb 27 08:51 app
-rwxr-xr-x  1 www-data www-data 425 feb 24 15:40 artisan
drwxrwxr-x  3 www-data www-data 4096 feb 24 15:40 bootstrap
-rw-rw-r--  1 www-data www-data 2545 abr 13 23:23 composer.json
-rw-rw-r--  1 www-data www-data 354192 abr 13 23:23 composer.lock
drwxrwxr-x  2 www-data www-data 4096 abr 10 17:18 config
drwxrwxr-x  5 www-data www-data 4096 feb 24 15:40 database
-rw-rw-r--  1 www-data www-data 258 feb 24 15:40 .editorconfig
-rw-rw-r--  1 www-data www-data 1140 abr 24 23:52 .env
-rw-rw-r--  1 www-data www-data 1075 feb 27 08:48 .env.example
-rw-rw-r--  1 www-data www-data 186 feb 24 15:40 .gitattributes
-rw-rw-r--  1 www-data www-data 306 mar  6 10:02 .gitignore
drwxrwxr-x  5 www-data www-data 4096 abr 10 17:18 lang
drwxrwxr-x 167 www-data www-data 4096 feb 27 08:54 node_modules
-rw-rw-r--  1 www-data www-data 624 feb 27 08:52 package.json
-rw-rw-r--  1 www-data www-data 142392 feb 27 08:52 package-lock.json
-rw-rw-r--  1 www-data www-data 1191 feb 24 15:40 phpununit.xml
         93 feb 27 08:51 postcss.config.js
drwxrwxr-x  6 www-data www-data 4096 abr 10 17:18 public
-rw-rw-r--  1 www-data www-data 3285 abr 13 23:26 README.md
drwxrwxr-x  5 www-data www-data 4096 feb 24 15:40 resources
drwxrwxr-x  2 www-data www-data 4096 abr 10 17:18 routes
drwxrwxr-x  5 www-data www-data 4096 feb 24 15:40 storage
-rw-rw-r--  1 www-data www-data 541 feb 27 08:51 tailwind.config.js
drwxrwxr-x  4 www-data www-data 4096 feb 27 08:48 tests
drwxrwxr-x  54 www-data www-data 4096 abr  6 17:02 vendor
-rw-rw-r--  1 www-data www-data 263 feb 27 08:51 vite.config.js
alexisg@samsung:/var/www/html$ |
```

Es momento de configurar el virtualhost en /etc/apache2/. Nos aseguramos que Apache escuche por el puerto 443 cuando haya presente SSL:

```
alexisg@samsung:/etc/apache2$ cat ports.conf
# If you just change the port or add more ports here, you will likely also
# have to change the VirtualHost statement in
# /etc/apache2/sites-enabled/000-default.conf

Listen 80

<IfModule ssl_module>
    Listen 443
</IfModule>

<IfModule mod_gnutls.c>
    Listen 443
</IfModule>
alexisg@samsung:/etc/apache2$ |
```

Configuramos el fichero **agrolanza.conf** en /etc/apache2/sites-available con la configuración correspondiente para el virtualhost:



```
<VirtualHost *:443>
    ServerName agrolanza.local
    DocumentRoot /var/www/html/agrolanza/public
    <Directory /var/www/html/agrolanza>
        AllowOverride All
        Require all granted
    </Directory>

    SSLEngine on
    SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
    SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

    ErrorLog ${APACHE_LOG_DIR}/agrolanza-ssl-error.log
    CustomLog ${APACHE_LOG_DIR}/agrolanza-ssl-access.log combined
</VirtualHost>

<VirtualHost *:80>
    ServerName agrolanza.local
    Redirect permanent / https://agrolanza.local/
</VirtualHost>
```

Ejecutamos los siguientes comandos en la terminal para habilitar los módulos para SSL y rewrite (para aplicar cambios de configuración desde un fichero .htaccess), dar de alta el virtualhost con fichero de configuración agrolanza.conf y reiniciamos el servicio:

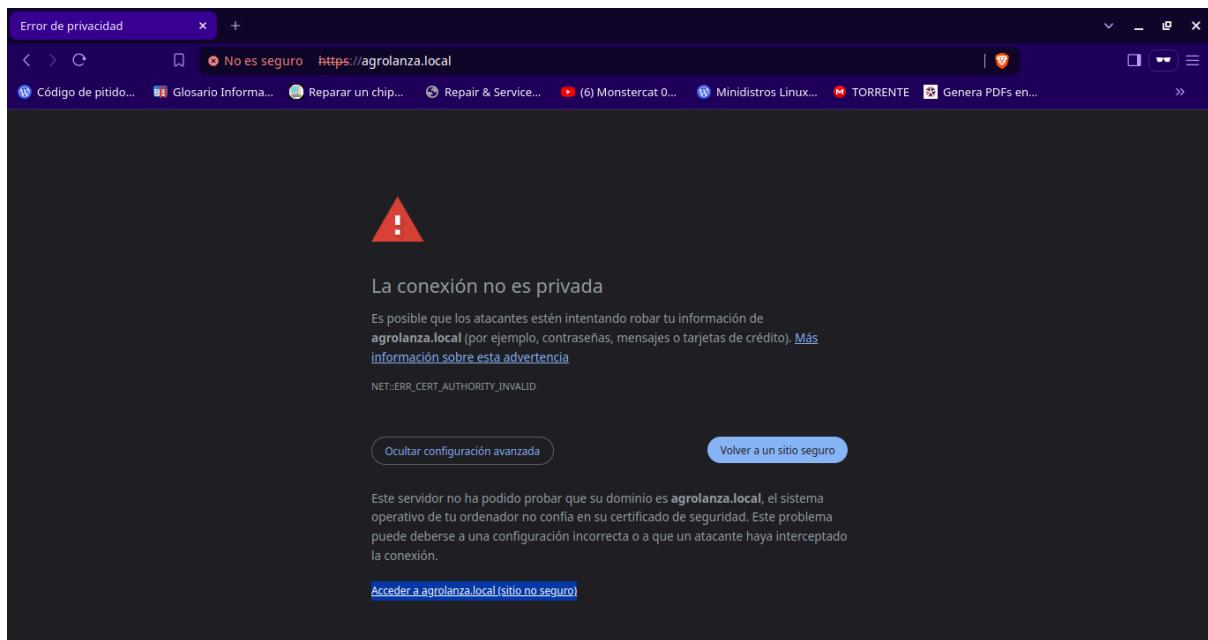
```
sudo a2ensite agrolanza.conf
sudo a2enmod rewrite
sudo a2enmod ssl
sudo systemctl restart apache2
```

Comprobamos el estado del servicio:

```
alexisg@samsung:/etc/apache2/sites-available$ sudo systemctl status apache2.service
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
  Active: active (running) since Thu 2025-04-24 23:30:50 WEST; 22s ago
    Docs: https://httpd.apache.org/docs/2.4/
   Process: 11623 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
 Main PID: 11626 (apache2)
    Tasks: 6 (limit: 18991)
   Memory: 14.5M (peak: 14.9M)
      CPU: 94ms
     CGroup: /system.slice/apache2.service
             └─11626 /usr/sbin/apache2 -k start
```

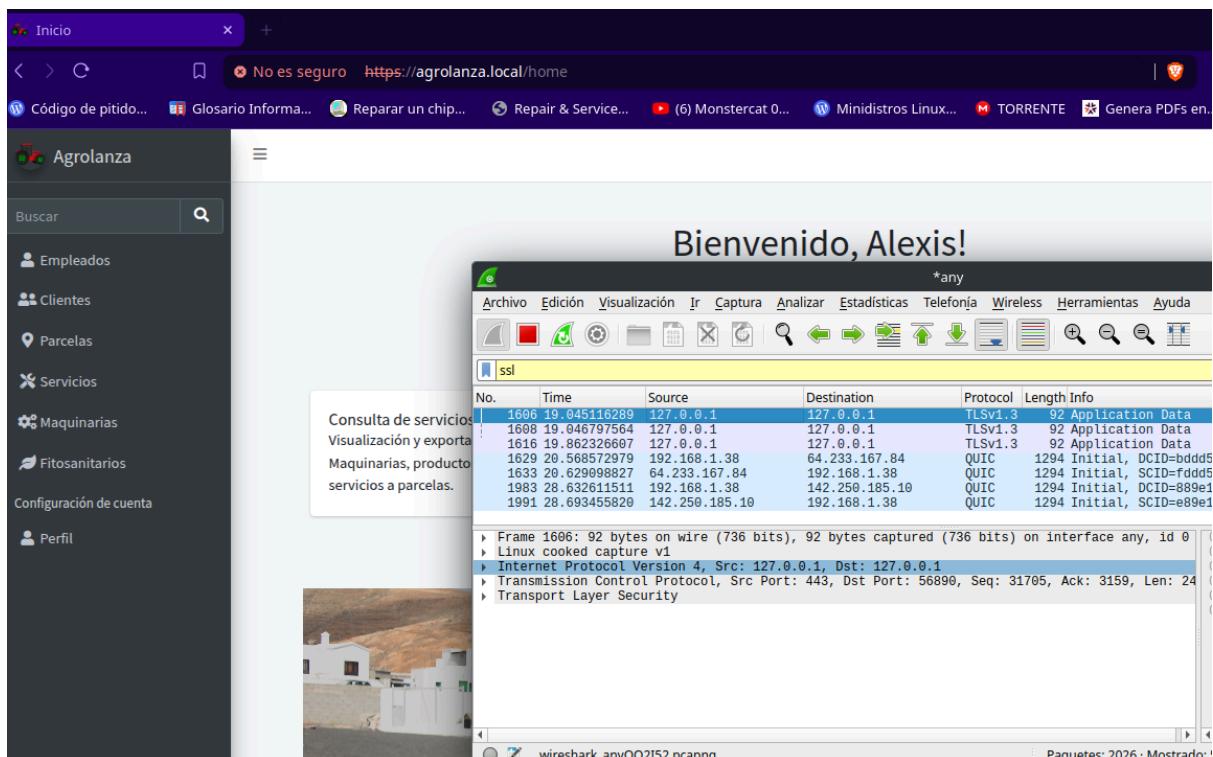
Agregamos la siguiente línea en **/etc/hosts**: 127.0.0.1 agrolanza.local.

Veamos si al tratar de acceder a agrolanza.local funciona correctamente:



Esta alerta se debe a que el certificado SSL utilizado no ha sido emitido ni verificado por una entidad certificadora reconocida, por lo que el navegador no puede garantizar la autenticidad del sitio. Esto puede solucionarse obteniendo un certificado verificado.

Sin embargo, nos sirve para que la plataforma esté funcionando por protocolo HTTPS y puerto 443, lo cual permite el cifrado de los paquetes que circulan en la comunicación entre el cliente y el servidor, aumentando significativamente la seguridad en red.





Conclusiones

A lo largo de este trabajo se ha evidenciado la necesidad real de modernizar el sector agrícola en la isla de Lanzarote, no sólo por su valor histórico y cultural, sino también por su potencial para generar empleo y reactivar terrenos actualmente en desuso. La informatización de los procesos dentro de las pequeñas y medianas empresas agrícolas no solo contribuirá a una mejor gestión de recursos y tiempo, sino que además permitirá una mayor trazabilidad, organización y control sobre cada aspecto de la actividad agrícola.

El proyecto desarrollado en Laravel 12 responde a esta necesidad mediante la creación de una plataforma web robusta, segura y fácilmente escalable. Laravel ha demostrado ser el entorno de trabajo adecuado para abordar los objetivos del proyecto, gracias a su estructura organizada, su integración nativa con herramientas modernas como Breeze y Spatie Permission, y su compatibilidad con soluciones de terceros como DomPDF o LeafletJS.

Durante el desarrollo se han abordado múltiples dimensiones del sistema: desde la gestión de empleados y clientes, hasta la administración de maquinaria, productos fitosanitarios y servicios, así como funcionalidades complementarias que aportan un alto valor añadido, como la generación de informes en PDF y la localización geográfica de parcelas.

La elección de tecnologías como MySQL para el almacenamiento de datos, Bootstrap para la interfaz de usuario y LeafletJS para la visualización de coordenadas, se ha basado en criterios de rendimiento, accesibilidad y facilidad de implementación. Además, se ha asegurado la aplicación de buenas prácticas en términos de seguridad, usabilidad y diseño centrado en el usuario.

En definitiva, este sistema de gestión agrícola no solo representa una herramienta tecnológica útil para los profesionales del sector, sino que además puede ser el primer paso hacia una transformación digital más profunda y necesaria en el entorno rural de la isla. Gracias a la estructura modular y la flexibilidad, el sistema queda preparado para futuras mejoras, integraciones o adaptaciones según las necesidades cambiantes del sector.





Repository

El repositorio de este proyecto se encuentra publicado en GitHub. Para acceder, basta con pinchar [aquí](#).

Anexo. Definiciones

Caldo: Cantidad de agua + producto fitosanitario, con sus respectivas proporciones, medido normalmente en litros o kilogramos.

Fitosanitario: Producto diseñado para el control de plagas y enfermedades, o malas hierbas. Es de ser llevado a cabo por profesionales.

Sector agrario: El sector agrario es el conjunto de actividades económicas relacionadas con la producción de alimentos, fibras, semillas y otros productos agrícolas.

Certificado SSL: Un certificado SSL (Secure Sockets Layer) es un certificado digital que autentica la identidad de un sitio web y establece una conexión cifrada entre el navegador y el servidor web, lo que permite la comunicación segura a través de HTTPS.

Firewall: Un cortafuegos, o firewall, es un sistema de seguridad que funciona como una barrera entre una red interna y una externa. Su función principal es proteger un sistema informático de accesos no autorizados, ciberataques y malware, permitiendo o bloqueando el tráfico de red según una serie de reglas predefinidas

Virtualhost: Un virtual host, o anfitrión virtual, es una configuración que permite a un servidor web (en este caso Apache) alojar múltiples sitios web en una misma máquina física. Esto se logra mediante la identificación de cada sitio web a través de su nombre de dominio, dirección IP, puerto, o una combinación de estos