



SIGERASTUR



PROYECTO 2º DAW 2022-2023

ENOL ALÁEZ RODRÍGUEZ

Índice

1. Descripción general	3
2. Plataforma y herramientas de desarrollo	4
3. Alcance del sistema.....	5
4. Tipología de usuarios.....	7
5. Consideraciones de seguridad.....	7
6. Vistas y resultados.....	8
7. Dependencias.....	28
8. Modelo de datos.....	29
9. Diagrama de clases.....	30
10. Casos de uso	46
11. Diagrama de clases.....	48

1. Descripción General

Mi proyecto desarrollado para la empresa familiar Sicerastur consiste en una aplicación web de gestión de fincas desarrollada con Java Spring Boot, HTML, JavaScript y CSS.

Es una aplicación con el foco principal en el uso fácil de todas sus funcionalidades para brindar a Sicerastur una experiencia de gestión optima.

▪ Funcionalidades generales:

La aplicación web cuenta con formas de registrar, modificar y eliminar fincas, variedades, árboles y recolecciones. Además de poder aplicar tratamientos y reposiciones a los árboles para que se tenga constancia de su estado

Se facilita la visualización de la estructura de las fincas proporcionando un plano compuesto por los diferentes árboles que hay en una finca. Para mayor claridad visual los diferentes arboles se representarán con diferentes colores dependiendo de la variedad a la que pertenezcan.

▪ Objetivos generales del proyecto:

- Un diseño simple y fácil de entender.
- Ofrecer una navegación intuitiva.
- Altos niveles de seguridad.
- Una experiencia de gestión ideal.
- Código sencillo para evitar errores y para facilitar la escalabilidad a futuro

En conclusión, la aplicación web proporciona todas las funcionalidades que se cabría esperar de una página web de gestión además de seguridad apropiada para evitar los errores y que la información sea expuesta a ataques.

2. Plataforma y Herramientas de Desarrollo

Este proyecto se ha llevado acabo utilizando las siguientes tecnologías:

▪ Lenguaje de servidor

La aplicación fue desarrollada en Java, haciendo uso de Spring Boot y Thymleaf para la creación de servlets y adición de dependencias. Thymleaf principalmente se utilizo para comunicarse con la parte de cliente e interfaz gráfica de la aplicación web.

▪ Lenguajes Cliente

Se empleó JavaScript de forma situacional para funcionalidades dinámicas, como controlar contraseñas.

▪ Lenguajes para la Interfaz Gráfica

Se utilizaron HTML y CSS para mostrar y maquetar los datos a enseñar respectivamente. Thymleaf se ha utilizado también en conjunto con HTML para comunicarse con la parte servidor de la aplicación web.

▪ Despliegue de la página web

Para desplegar la página eh utilizado La plataforma Heroku, que me permite ejecutar y operar mis aplicaciones en la web de forma gratuita. también fue necesario GitHub para subir los datos de mi página a la nube.

▪ Entorno de desarrollo integrado

La aplicación se desarrollo utilizando Eclipse como su IDE en su distribución más reciente.

▪ Realización de pruebas

Las pruebas se realizaron principalmente sobre el navegador Chrome, también de forma más situacional sobre el navegador Edge. En conjunto, también se utilizó MySQLWorkbench para probar la base de datos de forma local.

3. Alcance del sistema

Esta aplicación contará con dos tipos de usuarios, el usuario registrado y el usuario administrador, este último tiene permisos que los demás usuarios no tienen. Cualquiera que quiera visitar la página puede hacerlo sin necesidad de registrarse, pero solo tendrá acceso a la página de inicio de la web y la capacidad de registrarse.

▪ Página de inicio

Esta página contiene una pequeña presentación de la empresa Sicerastur, con su logo y una pequeña descripción. Desde esta página se podrán registrar los usuarios o iniciar sesión si ya tiene cuenta.

▪ Página de login

Desde esta página los usuarios podrán iniciar sesión, dependiendo de si son administradores o usuarios registrados se les redirigirá a una página u otra. Si no están registrados tendrán la opción de hacerlo desde aquí.

▪ Página de bienvenida

Una vez haya iniciado sesión se redirigirá al usuario a esta página, desde aquí podrá elegir cualquiera de las vistas principales: fincas, variedades, recolecciones y tratamientos.

▪ Fincas

Desde la página principal de fincas se verá un listado de todas las fincas registradas. Desde aquí se podrá acceder a los árboles de cada finca y al su plano correspondiente. Si se tiene los permisos necesarios se podrán añadir, modificar y eliminar fincas. También se podrá acceder a todas las otras vistas principales.

▪ Árboles

A esta página se accede desde la vista de fincas. Se podrán ver todos los árboles de la finca en la que se haya clicado. También se podrá acceder a los datos de los árboles, sus reposiciones y tratamientos. Si se tienen los permisos se permitirán añadir, modificar y eliminar árboles.

▪ Plano de árboles

A esta página se accede desde la vista de fincas, Se podrán ver todos los árboles registrados de la finca especificada en forma de cuadrícula. Desde aquí se podrán registrar nuevos árboles o ver la información de los árboles registrados si se clican en ellos.

■ Tratamientos de árboles

A esta pagina se accede desde la vista de árboles. Desde esta página se podrán ver los datos del árbol, sus tratamientos y recolecciones. Se podrán añadir nuevos tratamientos y recolecciones.

■ Variedades

Desde la pagina principal de variedades se pondrán a visualizar todas las variedades registradas. También, si se tiene los permisos, se podrán añadir, modificar o eliminar. Se podrá acceder a todas las otras vistas principales.

■ Tratamientos

Desde la página principal de tratamientos se pondrán a visualizar todos los tratamientos registrados. Los tratamientos registraos son los que se podrán elegir cuando se escoja un tratamiento a aplicar a diferentes árboles. También, si se tiene los permisos, se podrán añadir, modificar o eliminar. Se podrá acceder a todas las otras vistas principales.

■ Recolecciones

Desde la página principal de recolecciones se podrán ver todas las recolecciones hechas. Se podrán registrar nuevas recolecciones, si se tiene los permisos necesarios, también eliminar y modificar. Se podrá acceder a todas las otras vistas principales

Desde las paginas en las que se puede insertar o modificar objetos, a la hora de hacerlo se les redirigirá a un formulario donde introducirán o modificaran los datos.

Desde todas las vistas se tendrá acceso a cerrar sesión, lo que les redirigirá a la página de inicio, al igual que a todas las vistas principales: fincas, variedades, recolecciones y tratamientos.

4. Tipología de Usuarios

■ Usuario Anónimo o No registrado

- El usuario anónimo o usuario que no está registrado tendrá acceso únicamente a la página de presentación de la empresa.
- Tendrá acceso al login y al registro de usuarios para darse de alta.

■ Usuario Registrado

- El usuario registrado tendrá acceso a todas las vistas principales: fincas, variedades, recolecciones y tratamientos.
- Desde la vista de fincas podrá acceder al plano de la finca y todos los árboles de la finca seleccionada.
- En la vista de árboles podrá añadir un nuevo árbol, al igual que ver sus datos y añadirle al árbol seleccionado nuevos tratamiento y reposiciones.
- Desde la Vista recolecciones podrá solamente añadir una nueva recolección.
- Para el resto de las vistas solo tendrá acceso a visualizar los datos.

■ Usuario Administrador

- El usuario administrador tendrá acceso a todas las vistas.
- Podrá ver, añadir, modificar y eliminar objetos de cualquiera de las vistas.

5. Consideraciones de Seguridad

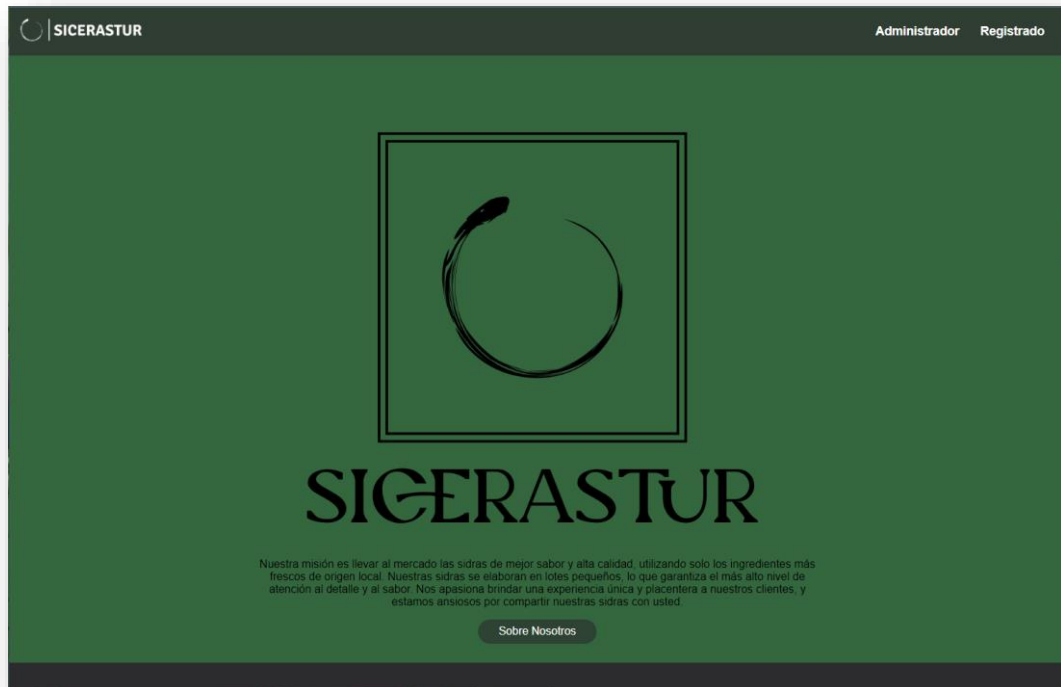
La principal restricción es garantizar que los usuarios registrados no puedan acceder a las vistas de los usuarios administradores de ninguna manera.

También a la hora de registrarse se tendrán que proporcionar dos instancias de contraseña, si no coinciden el registro no se llevará a cabo.

6. Vistas y Resultados

Vistas generales

Todo el mundo tiene acceso a estas vistas, ya sea este registrado o no.



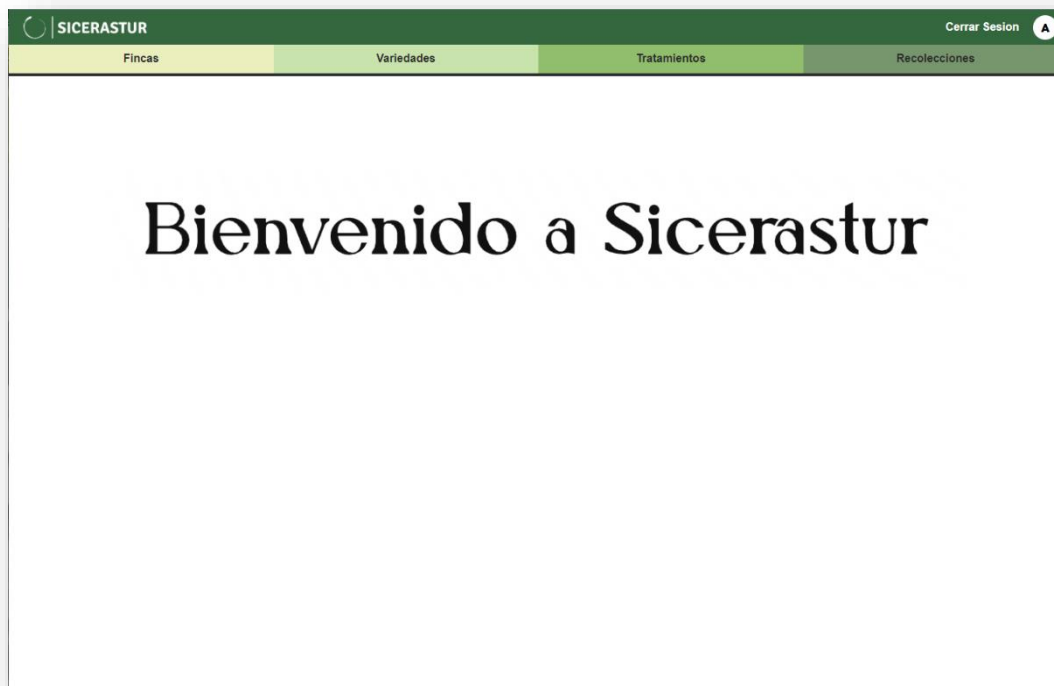
The screenshot shows the SICERASTUR registration interface. At the top, there is a dark green header with the SICERASTUR logo and name. Below this, a white box contains a dark green header labeled "Registrate". Inside this box, there are three input fields: "Usuario:", "Contraseña:", and "Confirmar Contraseña:". Below the input fields is a yellow "Registrarse" button and a purple "Volver" link.

Esta vista solo se muestra si las contraseñas no coinciden al intentar registrarse.

This screenshot shows the same registration interface as the previous one, but with the "Usuario:" field filled with "Enol" and both password fields filled with "*****". A red error message, "Las Contraseñas No Coinciden", is displayed above the yellow "Registrarse" button. The "Volver" link remains at the bottom.

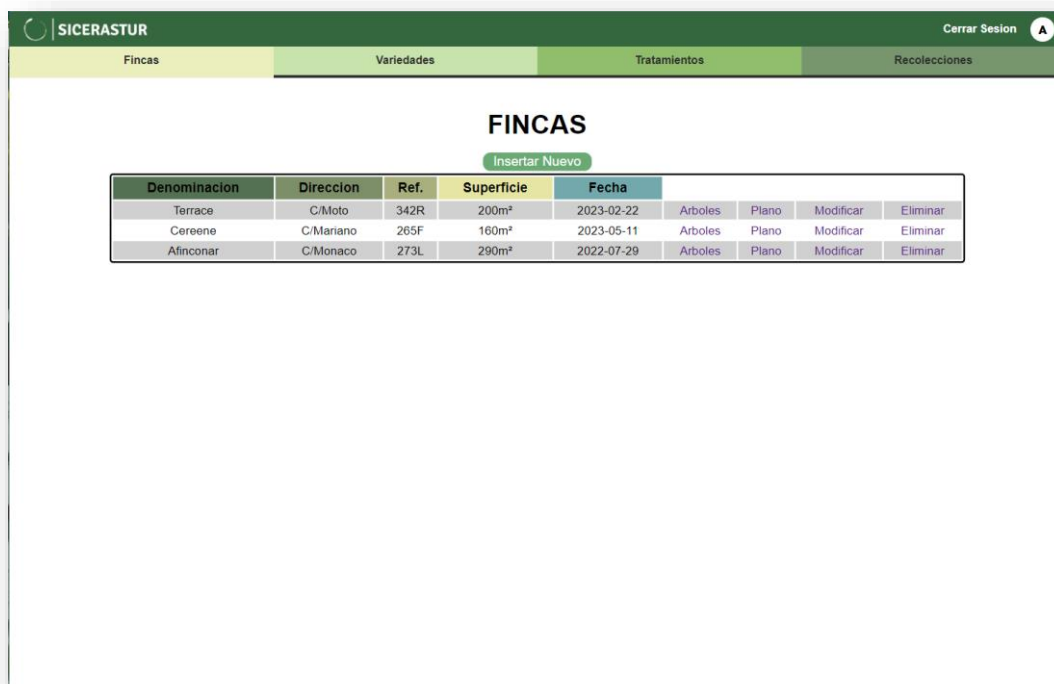
Vistas del usuario Administrador


Solo el usuario administrador puede ver estas vistas




- **Vistas de Fincas de Administrador**

Vistas de insertar, modificar y el plano de los arboles de la finca respectivamente.




SICERASTUR

Cerrar Sesión


Fincas
Variedades
Tratamientos
Recolecciones

Nueva Finca

Denominacion:


Direccion:


Ref:

Superficie:

Insertar

Volver


SICERASTUR

Cerrar Sesión


Fincas
Variedades
Tratamientos
Recolecciones

Modificar Finca Terrace

Nombre:

Direccion:

Ref:

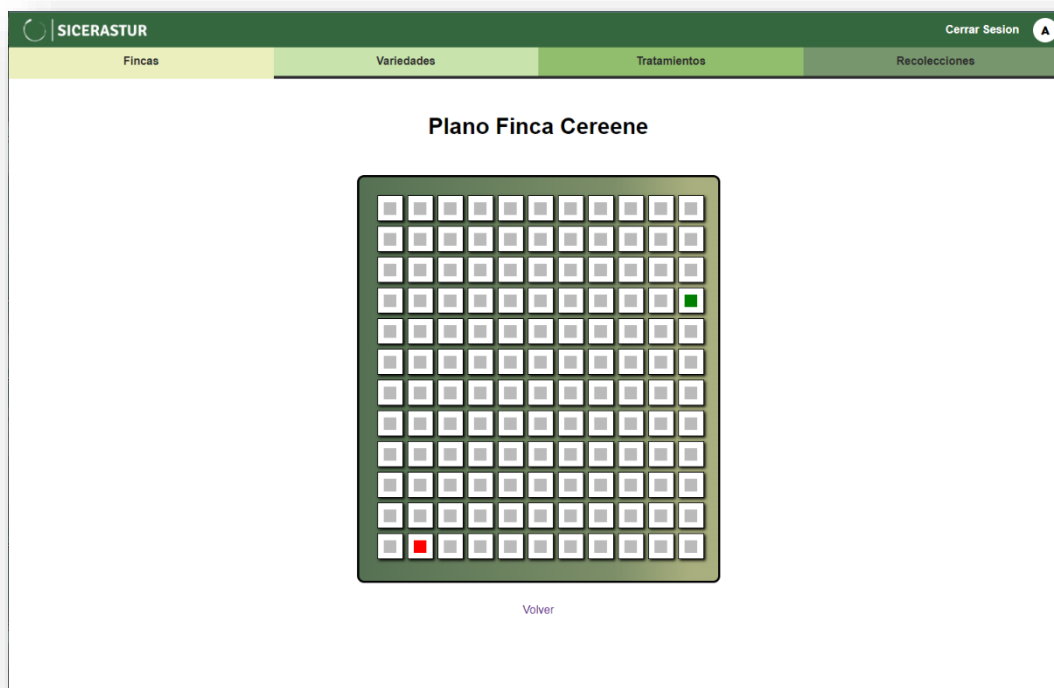
Superficie:

Fecha:

Fecha Anterior: 2023-02-22

Enviar

Volver



- **Vistas de Arboles de Administrador**

Vistas de mostrar, insertar, modificar, datos del árbol con reposiciones y tratamientos, insertar tratamiento relacionado con el árbol e insertar reposición del árbol y la vista de error que salta cuando no hay arboles respectivamente.



Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Nuevo Arbol

De Terrace

Fila:

Columna:

Estado:

Variedad:

Insertar

Volver

Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Arbol De Terrace

Fila: 4 / Columna: 4

Fila:

Columna:

Estado:

Variedad:

Finca:

Fecha:

dd/mm/aaaa

Enviar

Volver

Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Arbol De Terrace
Fila: 4 / Columna: 4

Fecha: 2023-09-01
Variedad: Meana
Estado: Sano

TRATAMIENTOS

REPOSICIONES

Nuevo Tratamiento

Nombre	Fecha
Fungicida	2023-06-06
Insecticida	2023-06-06
Fertilizante	2023-06-06

Nueva Reposición

Causa	Fecha
Muerte	2023-06-06
Mal posicionamiento	2023-06-06

Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Nuevo Tratamiento

Tratamiento:

fungicida

Insertar

Volver

SICERASTUR

Cerrar Sesión

Fincas

Variedades

Tratamientos

Recolecciones

Nueva Reposicion

Causa

Enviar

Volver

SICERASTUR

Cerrar Sesión

Fincas

Variedades

Tratamientos

Recolecciones

No Hay Arboles En La Finca Terrace

Nuevo Arbol

Volver

- **Vistas de Variedades de Administrador**

Vistas de mostrar, insertar y modificar variedades respectivamente.

VARIEDADES

Insertar Nuevo

Nombre	Color		
Meana	Red	Modificar	Eliminar
Clara	Green	Modificar	Eliminar

Nueva Variedad

Nombre:

Color:

Insertar

Volver

SICERASTUR Cerrar Sesión

Fincas Variedades Tratamientos Recolecciones

Modificar Variedad Meana

Nombre:

Color:

[Volver](#)

Modificar Variedad Meana

Nombre:

Color:

- Rojo
- verde
- rosa
- amarillo
- azul
- morado
- naranja
- marron
- cian
- beige
- khaki

- **Vistas de Tratamientos de Administrador**

Vistas de mostrar, insertar y modificar tratamientos respectivamente

The screenshot displays the 'TRATAMIENTOS' (Treatments) view. At the top, there is a navigation bar with the SICERASTUR logo and a 'Cerrar Sesión' (Logout) button. Below the navigation bar, there are four tabs: 'Fincas', 'Variedades', 'Tratamientos', and 'Recolecciones'. The 'Tratamientos' tab is currently selected. The main content area is titled 'TRATAMIENTOS' and features a green button labeled 'Insertar Nuevo' (Insert New). Below this, there is a table with the following data:

Nombre	Fecha		
Fungicida	2023-04-29	Modificar	Eliminar
Insecticida	2023-04-29	Modificar	Eliminar
Fertilizante	2023-05-16	Modificar	Eliminar

The screenshot displays the 'Nuevo Tratamiento' (New Treatment) form. The form has a title 'Nuevo Tratamiento' and a text input field labeled 'Nombre:'. Below the input field, there is a green button labeled 'insertar' and a link labeled 'Volver' (Return).

SICERASTUR

Cerrar Sesión

Fincas

Variedades

Tratamientos

Recolecciones

Modificar Tratamiento Fungicida

Nombre:

fungicida

Fecha

Fecha Anterior: 2023-04-29

dd/mm/aaaa

Enviar

Volver

- Vistas de Recolecciones de Administrador**
Vistas de mostrar, insertar y modificar recolecciones respectivamente

SICERASTUR

Cerrar Sesión

Fincas

Variedades


Tratamientos


Recolecciones

RECOLECCIONES

Insertar Nuevo

Finca	Variedad	Kilos	Fecha		
Terrace	Meana	105 Kg	2023-05-21	Modificar	Eliminar
Cereene	Clara	190 Kg	2023-05-25	Modificar	Eliminar


SICERASTUR

Cerrar Sesión


Fincas
Variedades
Tratamientos
Recolecciones

Nueva Recoleccion


Finca: terrace


Variedad: meana

Kilos: 0

Insertar

Volver


SICERASTUR

Cerrar Sesión


Fincas
Variedades
Tratamientos
Recolecciones

Modificar Recoleccion

Finca: terrace

Variedad: meana

Kilos: 105

Fecha: Fecha Anterior: 2023-05-21

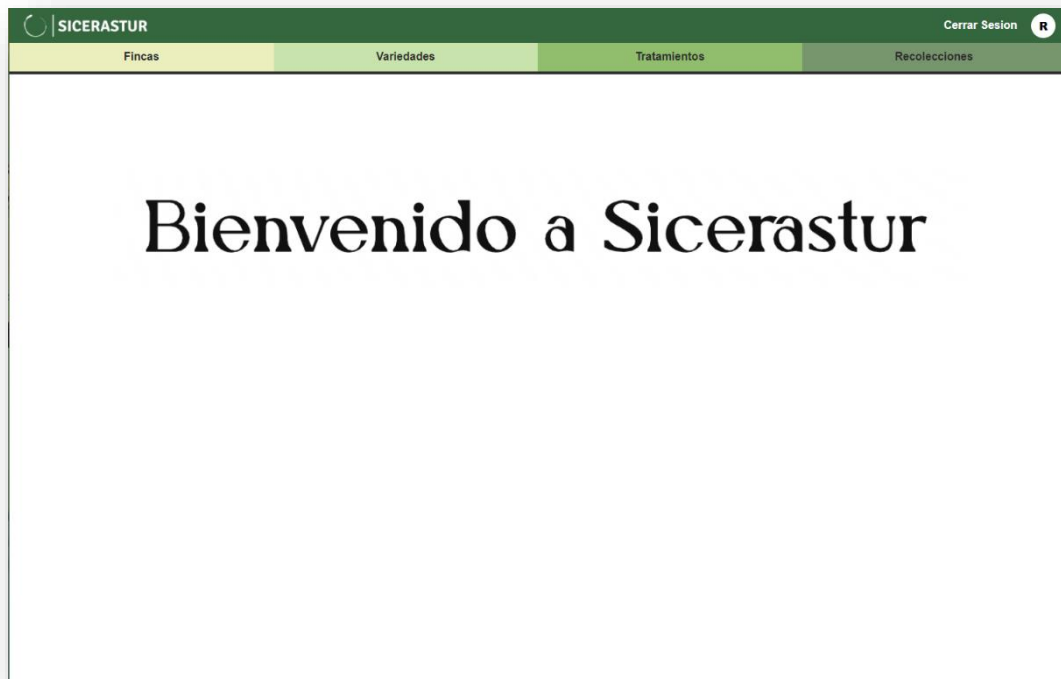
dd / mm / aaaa

Enviar

Volver

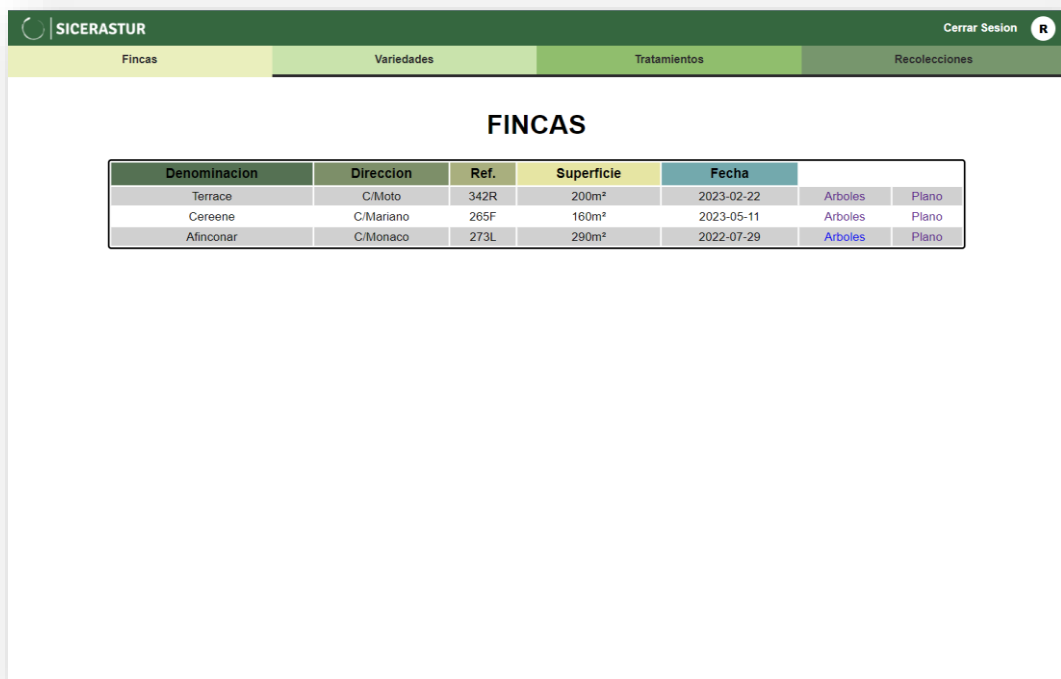
Vistas del usuario Registrado

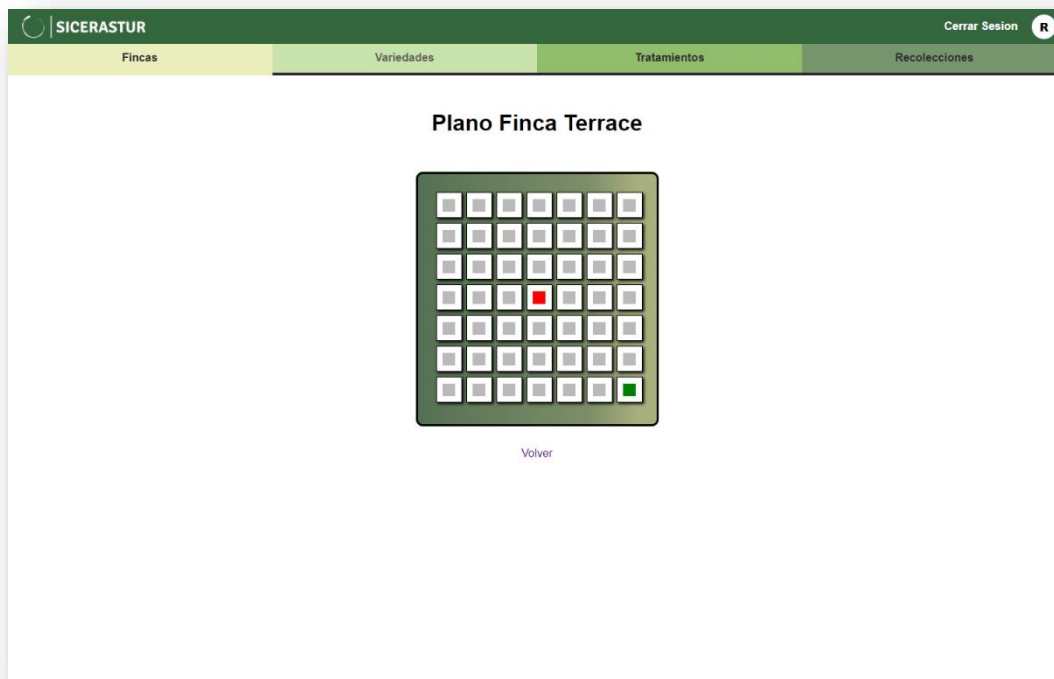
Tanto el usuario administrador como el usuario registrado pueden ver estas vistas. Se han reducido las funcionalidades a las que puede acceder el usuario registrado.



- **Vistas de Fincas del Usuario Registrado**

Vista de mostrar fincas y el plano de la finca respectivamente.





- **Vistas de Arboles del Usuario Registrado**

Vistas de mostrar, insertar, modificar, datos del árbol con reposiciones y tratamientos, insertar tratamiento relacionado con el árbol e insertar reposición del árbol respectivamente.



Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Nuevo Arbol

De Terrace

Fila:

Columna:

Estado:

Variedad:

Insertar

Volver

Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Arbol De Cereene

Fila: 12 / Columna: 2

Fila:

Columna:

Estado:

Variedad:

Finca:

Fecha:

dd/mm/aaaa

Enviar

Volver

Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Arbol De Cereene

Fila: 12 / Columna: 2

Fecha2023-09-01

VariedadMeana

EstadoSano

TRATAMIENTOS

Nuevo Tratamiento

Nombre	Fecha
Fertilizante	2023-06-06

REPOSICIONES

Nueva Reposicion

Causa	Fecha
Muerte	2023-06-06

Cerrar Sesión

Fincas
Variedades
Tratamientos
Recolecciones

Nuevo Tratamiento

Tratamiento:

fungicida

Insertar

Volver

The screenshot shows the 'Nueva Reposicion' form within the SICERASTUR application. The form is centered on a white background. It has a title bar 'Nueva Reposicion' in a green box. Below the title, there is a text input field labeled 'Causa'. At the bottom of the form, there is a green 'Enviar' button and a purple 'Volver' link. The application's header is green with the SICERASTUR logo on the left and 'Cerrar Sesión' with a user icon on the right. A navigation bar below the header has four tabs: 'Fincas' (highlighted in yellow), 'Variedades' (green), 'Tratamientos' (green), and 'Recolecciones' (green).

- **Vistas de Variedades del Usuario Registrado**
Vista de mostrar variedades.

The screenshot shows the 'VARIEDADES' view within the SICERASTUR application. The view displays a table with two columns: 'Nombre' and 'Color'. The table has two rows of data: 'Meana' with 'Red' color and 'Clara' with 'Green' color. The application's header is green with the SICERASTUR logo on the left and 'Cerrar Sesión' with a user icon on the right. A navigation bar below the header has four tabs: 'Fincas' (yellow), 'Variedades' (highlighted in green), 'Tratamientos' (green), and 'Recolecciones' (green).

Nombre	Color
Meana	Red
Clara	Green

- **Vistas de Tratamientos del Usuario Registrado**
Vista de mostrar tratamientos.

 SICERASTUR

Cerrar Sesión 

Fincas

Variedades

Tratamientos

Recolecciones

TRATAMIENTOS

Nombre	Fecha
Fungicida	2023-04-29
Insecticida	2023-04-29
Fertilizante	2023-05-16

- **Vistas de Recolectores del Usuario Registrado**
Vista de mostrar e insertar recolecciones.

SICERASTUR

Cerrar Sesión

R

Fincas

Variedades

Tratamientos

Recolecciones

RECOLECCIONES

Insertar Nuevo

Finca	Variedad	Kilos	Fecha
Terrace	Meana	105 Kg	2023-05-21
Cereene	Clara	190 Kg	2023-05-25

The screenshot shows the 'Nueva Recoleccion' (New Collection) form within the SICERASTUR application. The form is centered on a white background. It contains three dropdown menus: 'Finca' with 'terrace' selected, 'Variedad' with 'meana' selected, and 'Kilos' with '0' entered. Below these fields are two buttons: a green 'Insertar' button and a purple 'Volver' button. The application's header is visible at the top, showing the SICERASTUR logo, navigation tabs for 'Fincas', 'Variedades', 'Tratamientos', and 'Recolecciones', and a 'Cerrar Sesión' button with a user icon.

Vista de error de falta de permisos

Esta vista salta cuando se intenta acceder a una vista a la que no se tiene acceso.



7. Dependencias

La aplicación web se desplegará usando Heroku. Esto implica que si la pagina web se cae o esta inaccesible no se podrá acceder a Siceratur.

Heroku es una plataforma en la nube que permite a las empresas y usuarios crear, entregar, monitorear y escalar aplicaciones.

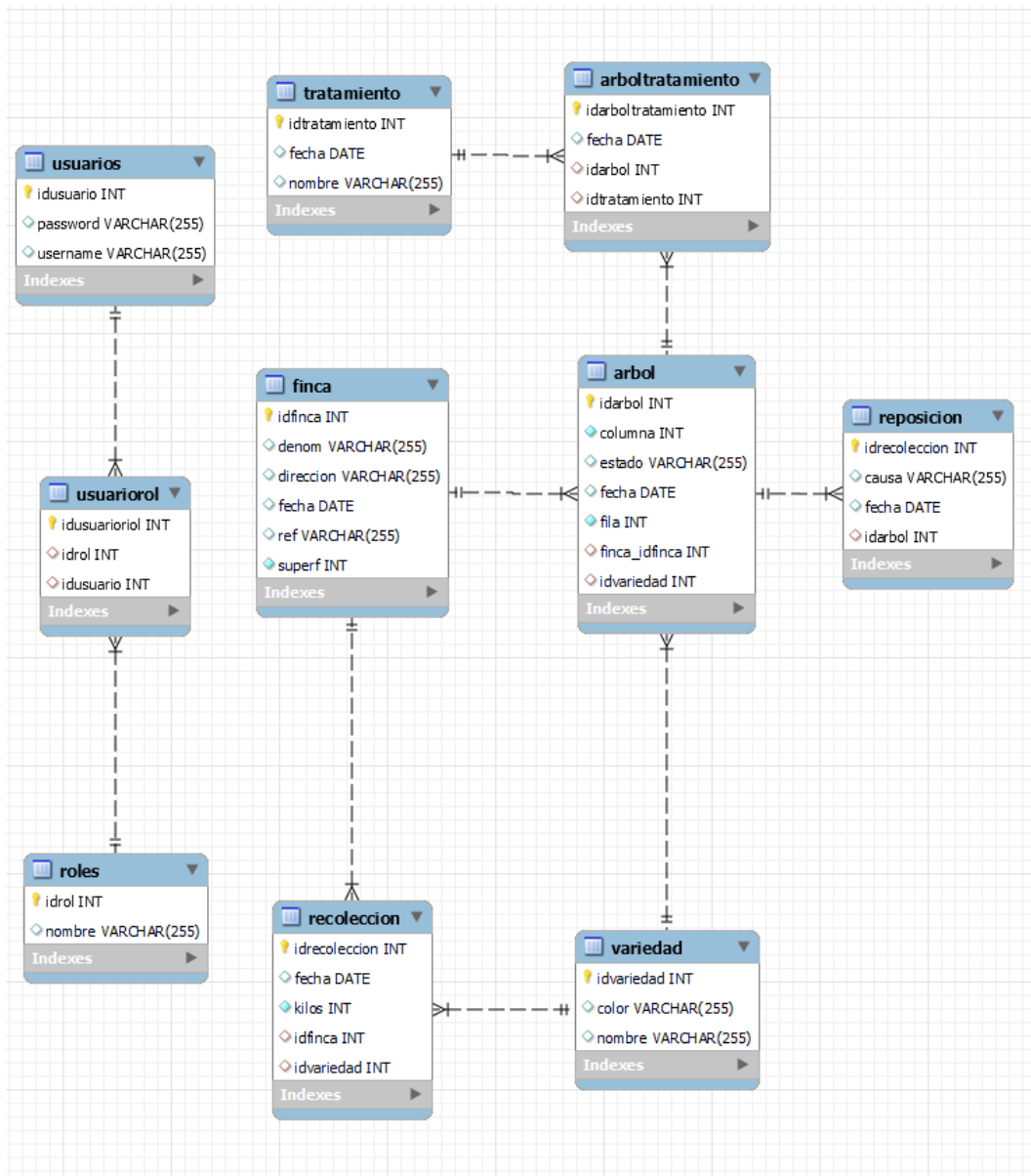
Heroku proporciona una base de datos en la nube de forma gratuita: ClearDB MySQL. Con el plan gratuito tendremos un máximo de queries por hora de 3600, pero por como es la naturaleza de la aplicación web no debería dar ningún problema.

Utilizando GitHub para subir allí los archivos de la aplicación web y mas tarde conectando el repositorio con Heroku se puede desplegar la página web sin problema.

url: <https://mysql-fincas-myproject.herokuapp.com/>



8. Modelo de Datos



9. Diagrama de Clases

■ Clases VO

```

RecoleccionVO
  ● getldrecoleccion() : int
  ● getKilos() : int
  ● getFecha() : LocalDate
  ● getVariedad() : VariedadVO
  ● getFinca() : FincaVO
  ● setldrecoleccion(int) : void
  ● setKilos(int) : void
  ● setFecha(LocalDate) : void
  ● setVariedad(VariedadVO) : void
  ● setFinca(FincaVO) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● RecoleccionVO()
  ● RecoleccionVO(int, int, LocalDate, VariedadVO, FincaVO)
  ■ idrecoleccion : int
  ■ kilos : int
  ■ fecha : LocalDate
  ■ variedad : VariedadVO
  ■ finca : FincaVO

```

```

ReposicionVO
  ● getldrecoleccion() : int
  ● getCausa() : String
  ● getFecha() : LocalDate
  ● getArbol() : ArbolVO
  ● setldrecoleccion(int) : void
  ● setCausa(String) : void
  ● setFecha(LocalDate) : void
  ● setArbol(ArbolVO) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● ReposicionVO()
  ● ReposicionVO(int, String, LocalDate, ArbolVO)
  ■ idrecoleccion : int
  ■ causa : String
  ■ fecha : LocalDate
  ■ arbol : ArbolVO

```

```

FincaVO
  ● getldfinca() : int
  ● getDenom() : String
  ● getDireccion() : String
  ● getRef() : String
  ● getSuperf() : int
  ● getFecha() : LocalDate
  ● getArbol() : List<ArbolVO>
  ● getRecoleccion() : List<RecoleccionVO>
  ● setldfinca(int) : void
  ● setDenom(String) : void
  ● setDireccion(String) : void
  ● setRef(String) : void
  ● setSuperf(int) : void
  ● setFecha(LocalDate) : void
  ● setArbol(List<ArbolVO>) : void
  ● setRecoleccion(List<RecoleccionVO>) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● FincaVO()
  ● FincaVO(int, String, String, String, int, LocalDate, List<ArbolVO>, List<RecoleccionVO>)
  ■ idfinca : int
  ■ denom : String
  ■ direccion : String
  ■ ref : String
  ■ superf : int
  ■ fecha : LocalDate
  ■ arbol : List<ArbolVO>
  ■ recolecciones : List<RecoleccionVO>
  ● FincaVO(String, String, String, int, LocalDate)

```

```

ArbolVO
  ● getIarbol() : int
  ● getFila() : int
  ● getColumna() : int
  ● getEstado() : String
  ● getFecha() : LocalDate
  ● getFinca() : FincaVO
  ● getVariedad() : VariedadVO
  ● getTratamientos() : List<ArbolTratamientoVO>
  ● getReposiciones() : List<ReposicionVO>
  ● setIarbol(int) : void
  ● setFila(int) : void
  ● setColumna(int) : void
  ● setEstado(String) : void
  ● setFecha(LocalDate) : void
  ● setFinca(FincaVO) : void
  ● setVariedad(VariedadVO) : void
  ● setTratamientos(List<ArbolTratamientoVO>) : void
  ● setReposiciones(List<ReposicionVO>) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ▲ hashCode() : int
  ▲ toString() : String
  ● ArbolVO()
  ● ArbolVO(int, int, int, String, LocalDate, FincaVO, VariedadVO, List<ArbolTratamientoVO>, List<ReposicionVO>)
  ■ idarbol : int
  ■ fila : int
  ■ columna : int
  ■ estado : String
  ■ fecha : LocalDate
  ■ finca : FincaVO
  ■ variedad : VariedadVO
  ■ tratamientos : List<ArbolTratamientoVO>
  ■ reposiciones : List<ReposicionVO>

```

```

TratamientoVO
  ● getIdtratamiento() : int
  ● getNombre() : String
  ● getFecha() : LocalDate
  ● getArboles() : List<ArbolTratamientoVO>
  ● setIdtratamiento(int) : void
  ● setNombre(String) : void
  ● setFecha(LocalDate) : void
  ● setArboles(List<ArbolTratamientoVO>) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ▲ hashCode() : int
  ▲ toString() : String
  ● TratamientoVO()
  ● TratamientoVO(int, String, LocalDate, List<ArbolTratamientoVO>)
  ■ idtratamiento : int
  ■ nombre : String
  ■ fecha : LocalDate
  ■ arboles : List<ArbolTratamientoVO>
  ● TratamientoVO(String, LocalDate)

```

```

ArbolDTO
  ● getIarbol() : int
  ● getFila() : int
  ● getColumna() : int
  ● getEstado() : String
  ● getIdfinca() : int
  ● getIdvariedad() : int
  ● setIarbol(int) : void
  ● setFila(int) : void
  ● setColumna(int) : void
  ● setEstado(String) : void
  ● setIdfinca(int) : void
  ● setIdvariedad(int) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ▲ hashCode() : int
  ▲ toString() : String
  ● ArbolDTO()
  ● ArbolDTO(int, int, int, String, int, int)
  ■ idarbol : int
  ■ fila : int
  ■ columna : int
  ■ estado : String
  ■ idfinca : int
  ■ idvariedad : int

```

```

ArbolTratamientoDTO
  ● getIIdtratamiento() : int
  ● getIIdarbol() : int
  ● setIIdtratamiento(int) : void
  ● setIIdarbol(int) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● C ArbolTratamientoDTO()
  ● C ArbolTratamientoDTO(int, int)
  ■ idtratamiento : int
  ■ idarbol : int

```

```

RecoleccionDTO
  ● getIIdrecoleccion() : int
  ● getKilos() : int
  ● getIIdfinca() : int
  ● getIIdvariedad() : int
  ● setIIdrecoleccion(int) : void
  ● setKilos(int) : void
  ● setIIdfinca(int) : void
  ● setIIdvariedad(int) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● C RecoleccionDTO()
  ● C RecoleccionDTO(int, int, int, int)
  ■ idrecoleccion : int
  ■ kilos : int
  ■ idfinca : int
  ■ idvariedad : int

```

```

VariedadVO
  ● getIIdvariedad() : int
  ● getNombre() : String
  ● getColor() : String
  ● getArbol() : List<ArbolVO>
  ● getRecoleccion() : List<RecoleccionVO>
  ● setIIdvariedad(int) : void
  ● setNombre(String) : void
  ● setColor(String) : void
  ● setArbol(List<ArbolVO>) : void
  ● setRecoleccion(List<RecoleccionVO>) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● C VariedadVO()
  ● C VariedadVO(int, String, String, List<ArbolVO>, List<RecoleccionVO>)
  ■ idvariedad : int
  ■ nombre : String
  ■ color : String
  ■ arbol : List<ArbolVO>
  ■ recoleccion : List<RecoleccionVO>
  ● C VariedadVO(String, String, List<ArbolVO>)

```


■ Controladores

```

▼ C ArbolController
  ▲ as : ArbolService
  ▲ fs : FincaService
  ▲ vs : VariedadService
  ▲ ats : ArbolTratamientoService
  ▲ ts : TratamientoService
  ▲ reps : ReposicionService
  ● mostrar(@RequestParam(value="idfinca") int, Model) : String
  ● forminsertar(@RequestParam(value="idfinca") int, Model) : String
  ● insertar(@RequestParam(value="idfinca") int, @ModelAttribute ArbolDTO, Model) : String
  ● formmodificar(@RequestParam(value="idarbol") int, Model) : String
  ● modificar(@ModelAttribute ArbolVO, Model) : String
  ● eliminar(@RequestParam(value="idarbol") int, Model) : String
  ● forminsertarTrata(@RequestParam(value="idarbol") int, Model) : String
  ● trata(@RequestParam(value="idarbol") int, Model) : String
  ● insertar(@RequestParam(value="idarbol") int, @ModelAttribute ArbolTratamientoDTO, Model) : String
  ● forminsertarrepo(@RequestParam(value="idarbol") int, Model) : String
  ● insertarrepos(@RequestParam(value="idarbol") int, @ModelAttribute ReposicionVO, Model) : String
  ● mostrarus(@RequestParam(value="idfinca") int, Model) : String
  ● forminsertarus(@RequestParam(value="idfinca") int, Model) : String
  ● insertarus(@RequestParam(value="idfinca") int, @ModelAttribute ArbolDTO, Model) : String
  ● formmodificarus(@RequestParam(value="idarbol") int, Model) : String
  ● modificarus(@ModelAttribute ArbolVO, Model) : String
  ● forminsertarTrataus(@RequestParam(value="idarbol") int, Model) : String
  ● trataus(@RequestParam(value="idarbol") int, Model) : String
  ● insertarus(@RequestParam(value="idarbol") int, @ModelAttribute ArbolTratamientoDTO, Model) : String
  ● forminsertarreposus(@RequestParam(value="idarbol") int, Model) : String
  ● insertarreposus(@RequestParam(value="idarbol") int, @ModelAttribute ReposicionVO, Model) : String

```

```

▼ C TratamientoController
  ▲ ts : TratamientoService
  ● mostrar(Model) : String
  ● mostrarus(Model) : String
  ● forminsertar(Model) : String
  ● insertar(@ModelAttribute TratamientoVO, Model) : String
  ● eliminar(@RequestParam(value="idtrata") int, Model) : String
  ● formmodificar(@RequestParam(value="idtrata") int, Model) : String
  ● modificar(@ModelAttribute TratamientoVO, Model) : String

```

```

▼ C FincaController
  ▲ fs : FincaService
  ● mostrar(Model) : String
  ● mostrarus(Model) : String
  ● forminsertar(Model) : String
  ● insertar(@ModelAttribute FincaVO, Model) : String
  ● formmodificar(@RequestParam(value="idfinca") int, Model) : String
  ● modificar(@ModelAttribute FincaVO, Model) : String
  ● eliminar(@RequestParam(value="idfinca") int, Model) : String
  ● plano(@RequestParam(value="idfinca") int, Model) : String
  ● planous(@RequestParam(value="idfinca") int, Model) : String

```

```

▼ C VariedadController
  ▲ vs : VariedadService
  ● mostrar(Model) : String
  ● mostrarus(Model) : String
  ● forminsertar(Model) : String
  ● insertar(@ModelAttribute VariedadVO, Model) : String
  ● formmodificar(@RequestParam(value="idvariedad") int, Model) : String
  ● modificar(@ModelAttribute VariedadVO, Model) : String
  ● eliminar(@RequestParam(value="idvariedad") int, Model) : String

```

```

▼ C RecoleccionController
  ▲ rs : RecoleccionService
  ▲ fs : FincaService
  ▲ vs : VariedadService
  ● mostrar(Model) : String
  ● forminsertar(Model) : String
  ● insertar(@ModelAttribute RecoleccionDTO, Model) : String
  ● eliminar(@RequestParam(value="idreco") int, Model) : String
  ● formmodificar(@RequestParam(value="idreco") int, Model) : String
  ● modificar(@ModelAttribute RecoleccionVO, Model) : String
  ● mostrarus(Model) : String
  ● forminsertarus(Model) : String
  ● insertarus(@ModelAttribute RecoleccionDTO, Model) : String

```

```

▼ C PrincipalController
  ▲ su : ServicioUsuario
  ▲ sur : ServicioUsuarioRol
  ▲ sr : ServicioRol
  ● Principal01() : String
  ● Principal() : String
  ● Admin() : String
  ● user() : String
  ● error403() : String
  ● loginr() : String
  ● logout() : String
  ● register() : String
  ● forminsertar(Model) : String
  ● insertar(Model, @ModelAttribute UsuarioVO) : String
  ● salir(HttpServletRequest) : String

```

■ Servicios y Servicios Implementados

```

▼ C ArbolServiceImpl
  ■ ar : ArbolRepository
  ● ▲ findAllByFincald(int) : List<ArbolVO>
  ● ▲ save(S) <S extends ArbolVO> : S
  ● ▲ saveAll(Iterable<S>) <S extends ArbolVO> : Iterable<S>
  ● ▲ findById(Integer) : Optional<ArbolVO>
  ● ▲ existsById(Integer) : boolean
  ● ▲ findAll() : Iterable<ArbolVO>
  ● ▲ findAllById(Iterable<Integer>) : Iterable<ArbolVO>
  ● ▲ count() : long
  ● ▲ deleteById(Integer) : void
  ● ▲ delete(ArbolVO) : void
  ● ▲ deleteAllById(Iterable<? extends Integer>) : void
  ● ▲ deleteAll(Iterable<? extends ArbolVO>) : void
  ● ▲ deleteAll() : void

```

```

▼ I ArbolService
  ● ▲ findAllByFincald(int) : List<ArbolVO>
  ● ▲ save(S) <S extends ArbolVO> : S
  ● ▲ saveAll(Iterable<S>) <S extends ArbolVO> : Iterable<S>
  ● ▲ findById(Integer) : Optional<ArbolVO>
  ● ▲ existsById(Integer) : boolean
  ● ▲ findAll() : Iterable<ArbolVO>
  ● ▲ findAllById(Iterable<Integer>) : Iterable<ArbolVO>
  ● ▲ count() : long
  ● ▲ deleteById(Integer) : void
  ● ▲ delete(ArbolVO) : void
  ● ▲ deleteAllById(Iterable<? extends Integer>) : void
  ● ▲ deleteAll(Iterable<? extends ArbolVO>) : void
  ● ▲ deleteAll() : void

```

```

▼ C FincaServiceImpl
  ■ fr : FincaRepository
  ● ▲ save(S) <S extends FincaVO> : S
  ● ▲ saveAll(Iterable<S>) <S extends FincaVO> : Iterable<S>
  ● ▲ findById(Integer) : Optional<FincaVO>
  ● ▲ existsById(Integer) : boolean
  ● ▲ findAll() : Iterable<FincaVO>
  ● ▲ findAllById(Iterable<Integer>) : Iterable<FincaVO>
  ● ▲ count() : long
  ● ▲ deleteById(Integer) : void
  ● ▲ delete(FincaVO) : void
  ● ▲ deleteAllById(Iterable<? extends Integer>) : void
  ● ▲ deleteAll(Iterable<? extends FincaVO>) : void
  ● ▲ deleteAll() : void

```

```

▼ ⓘ FincaService
  ▲ save(S) <S extends FincaVO> : S
  ▲ saveAll(Iterable<S>) <S extends FincaVO> : Iterable<S>
  ▲ findById(Integer) : Optional<FincaVO>
  ▲ existsById(Integer) : boolean
  ▲ findAll() : Iterable<FincaVO>
  ▲ findAllById(Iterable<Integer>) : Iterable<FincaVO>
  ▲ count() : long
  ▲ deleteById(Integer) : void
  ▲ delete(FincaVO) : void
  ▲ deleteAllById(Iterable<? extends Integer>) : void
  ▲ deleteAll(Iterable<? extends FincaVO>) : void
  ▲ deleteAll() : void

```

```

▼ ⓘ ArbolTratamientoServiceImpl
  ■ atr : ArbolTratamientoRepository
  ▲ save(S) <S extends ArbolTratamientoVO> : S
  ▲ saveAll(Iterable<S>) <S extends ArbolTratamientoVO> : Iterable<S>
  ▲ findById(Integer) : Optional<ArbolTratamientoVO>
  ▲ existsById(Integer) : boolean
  ▲ findAll() : Iterable<ArbolTratamientoVO>
  ▲ findAllById(Iterable<Integer>) : Iterable<ArbolTratamientoVO>
  ▲ count() : long
  ▲ deleteById(Integer) : void
  ▲ delete(ArbolTratamientoVO) : void
  ▲ deleteAllById(Iterable<? extends Integer>) : void
  ▲ deleteAll(Iterable<? extends ArbolTratamientoVO>) : void
  ▲ deleteAll() : void

```

```

▼ ⓘ TratamientoService
  ▲ save(S) <S extends TratamientoVO> : S
  ▲ saveAll(Iterable<S>) <S extends TratamientoVO> : Iterable<S>
  ▲ findById(Integer) : Optional<TratamientoVO>
  ▲ existsById(Integer) : boolean
  ▲ findAll() : Iterable<TratamientoVO>
  ▲ findAllById(Iterable<Integer>) : Iterable<TratamientoVO>
  ▲ count() : long
  ▲ deleteById(Integer) : void
  ▲ delete(TratamientoVO) : void
  ▲ deleteAllById(Iterable<? extends Integer>) : void
  ▲ deleteAll(Iterable<? extends TratamientoVO>) : void
  ▲ deleteAll() : void

```

```

▼ C RecoleccionServiceImpl
  rr : RecoleccionRepository
  save(S) <S extends RecoleccionVO> : S
  saveAll(Iterable<S>) <S extends RecoleccionVO> : Iterable<S>
  findById(Integer) : Optional<RecoleccionVO>
  existsById(Integer) : boolean
  findAll() : Iterable<RecoleccionVO>
  findAllById(Iterable<Integer>) : Iterable<RecoleccionVO>
  count() : long
  deleteById(Integer) : void
  delete(RecoleccionVO) : void
  deleteAllById(Iterable<? extends Integer>) : void
  deleteAll(Iterable<? extends RecoleccionVO>) : void
  deleteAll() : void

```

```

▼ I RecoleccionService
  save(S) <S extends RecoleccionVO> : S
  saveAll(Iterable<S>) <S extends RecoleccionVO> : Iterable<S>
  findById(Integer) : Optional<RecoleccionVO>
  existsById(Integer) : boolean
  findAll() : Iterable<RecoleccionVO>
  findAllById(Iterable<Integer>) : Iterable<RecoleccionVO>
  count() : long
  deleteById(Integer) : void
  delete(RecoleccionVO) : void
  deleteAllById(Iterable<? extends Integer>) : void
  deleteAll(Iterable<? extends RecoleccionVO>) : void
  deleteAll() : void

```

```

▼ C ReposicionServiceImpl
  repr : ReposicionRepository
  save(S) <S extends ReposicionVO> : S
  saveAll(Iterable<S>) <S extends ReposicionVO> : Iterable<S>
  findById(Integer) : Optional<ReposicionVO>
  existsById(Integer) : boolean
  findAll() : Iterable<ReposicionVO>
  findAllById(Iterable<Integer>) : Iterable<ReposicionVO>
  count() : long
  deleteById(Integer) : void
  delete(ReposicionVO) : void
  deleteAllById(Iterable<? extends Integer>) : void
  deleteAll(Iterable<? extends ReposicionVO>) : void
  deleteAll() : void

```

```

▼ ⓘ ReposicionService
  ● ^ save(S) <S extends ReposicionVO> : S
  ● ^ saveAll(Iterable<S>) <S extends ReposicionVO> : Iterable<S>
  ● ^ findById(Integer) : Optional<ReposicionVO>
  ● ^ existsById(Integer) : boolean
  ● ^ findAll() : Iterable<ReposicionVO>
  ● ^ findAllById(Iterable<Integer>) : Iterable<ReposicionVO>
  ● ^ count() : long
  ● ^ deleteById(Integer) : void
  ● ^ delete(ReposicionVO) : void
  ● ^ deleteAllById(Iterable<? extends Integer>) : void
  ● ^ deleteAll(Iterable<? extends ReposicionVO>) : void
  ● ^ deleteAll() : void

```

```

▼ ⓘ TratamientoServiceImpl
  ■ tr : TratamientoRepository
  ● ^ save(S) <S extends TratamientoVO> : S
  ● ^ saveAll(Iterable<S>) <S extends TratamientoVO> : Iterable<S>
  ● ^ findById(Integer) : Optional<TratamientoVO>
  ● ^ existsById(Integer) : boolean
  ● ^ findAll() : Iterable<TratamientoVO>
  ● ^ findAllById(Iterable<Integer>) : Iterable<TratamientoVO>
  ● ^ count() : long
  ● ^ deleteById(Integer) : void
  ● ^ delete(TratamientoVO) : void
  ● ^ deleteAllById(Iterable<? extends Integer>) : void
  ● ^ deleteAll(Iterable<? extends TratamientoVO>) : void
  ● ^ deleteAll() : void

```

```

▼ ⓘ TratamientoService
  ● ^ save(S) <S extends TratamientoVO> : S
  ● ^ saveAll(Iterable<S>) <S extends TratamientoVO> : Iterable<S>
  ● ^ findById(Integer) : Optional<TratamientoVO>
  ● ^ existsById(Integer) : boolean
  ● ^ findAll() : Iterable<TratamientoVO>
  ● ^ findAllById(Iterable<Integer>) : Iterable<TratamientoVO>
  ● ^ count() : long
  ● ^ deleteById(Integer) : void
  ● ^ delete(TratamientoVO) : void
  ● ^ deleteAllById(Iterable<? extends Integer>) : void
  ● ^ deleteAll(Iterable<? extends TratamientoVO>) : void
  ● ^ deleteAll() : void

```

```

▼ C VariedadServiceImpl
  ■ vr : VariedadRepository
  ● ▲ save(S) <S extends VariedadVO> : S
  ● ▲ saveAll(Iterable<S>) <S extends VariedadVO> : Iterable<S>
  ● ▲ findById(Integer) : Optional<VariedadVO>
  ● ▲ existsById(Integer) : boolean
  ● ▲ findAll() : Iterable<VariedadVO>
  ● ▲ findAllById(Iterable<Integer>) : Iterable<VariedadVO>
  ● ▲ count() : long
  ● ▲ deleteById(Integer) : void
  ● ▲ delete(VariedadVO) : void
  ● ▲ deleteAllById(Iterable<? extends Integer>) : void
  ● ▲ deleteAll(Iterable<? extends VariedadVO>) : void
  ● ▲ deleteAll() : void

```

```

▼ I VariedadService
  ● ▲ save(S) <S extends VariedadVO> : S
  ● ▲ saveAll(Iterable<S>) <S extends VariedadVO> : Iterable<S>
  ● ▲ findById(Integer) : Optional<VariedadVO>
  ● ▲ existsById(Integer) : boolean
  ● ▲ findAll() : Iterable<VariedadVO>
  ● ▲ findAllById(Iterable<Integer>) : Iterable<VariedadVO>
  ● ▲ count() : long
  ● ▲ deleteById(Integer) : void
  ● ▲ delete(VariedadVO) : void
  ● ▲ deleteAllById(Iterable<? extends Integer>) : void
  ● ▲ deleteAll(Iterable<? extends VariedadVO>) : void
  ● ▲ deleteAll() : void

```

■ Seguridad VO

```

v UsuarioVO
  ● getIdusuario() : int
  ● getUsername() : String
  ● getPassword() : String
  ● getRoles() : List<UsuarioRolVO>
  ● setIdusuario(int) : void
  ● setUsername(String) : void
  ● setPassword(String) : void
  ● setRoles(List<UsuarioRolVO>) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● UsuarioVO(int, String, String, List<UsuarioRolVO>)
  ● UsuarioVO()
  ● idusuario : int
  ● username : String
  ● password : String
  ● roles : List<UsuarioRolVO>
  ● getAuthorities() : Collection<? extends GrantedAuthority>
  ● isAccountNonExpired() : boolean
  ● isAccountNonLocked() : boolean
  ● isCredentialsNonExpired() : boolean
  ● isEnabled() : boolean

```

```

v UsuarioRolVO
  ● getIdusuariorol() : int
  ● getUsuario() : UsuarioVO
  ● getRol() : RolVO
  ● setIdusuariorol(int) : void
  ● setUsuario(UsuarioVO) : void
  ● setRol(RolVO) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● UsuarioRolVO(int, UsuarioVO, RolVO)
  ● UsuarioRolVO()
  ● idusuariorol : int
  ● usuario : UsuarioVO
  ● rol : RolVO

```

```

v RolVO
  ● getIdrol() : int
  ● getNombre() : String
  ● setIdrol(int) : void
  ● setNombre(String) : void
  ● equals(Object) : boolean
  ◆ canEqual(Object) : boolean
  ● hashCode() : int
  ● toString() : String
  ● RolVO(int, String)
  ● RolVO()
  ● idrol : int
  ● nombre : String

```


■ Seguridad Servicios y servicios Implementados

```

v C ServicioUsuarioImpl
  ▲ ur : UsuarioRepository
  ● loadUserByUsername(String) : UserDetails
  ● findByUsername(String) : UserDetails
  ● save(S) <S extends UsuarioVO> : S
  ● saveAll(Iterable<S>) <S extends UsuarioVO> : Iterable<S>
  ● findById(Integer) : Optional<UsuarioVO>
  ● existsById(Integer) : boolean
  ● findAll() : Iterable<UsuarioVO>
  ● findAllById(Iterable<Integer>) : Iterable<UsuarioVO>
  ● count() : long
  ● deleteById(Integer) : void
  ● delete(UsuarioVO) : void
  ● deleteAllById(Iterable<? extends Integer>) : void
  ● deleteAll(Iterable<? extends UsuarioVO>) : void
  ● deleteAll() : void

```

```

v I ServicioUsuario
  ▲ loadUserByUsername(String) : UserDetails
  ▲ findByUsername(String) : UserDetails
  ▲ save(S) <S extends UsuarioVO> : S
  ▲ saveAll(Iterable<S>) <S extends UsuarioVO> : Iterable<S>
  ▲ findById(Integer) : Optional<UsuarioVO>
  ▲ existsById(Integer) : boolean
  ▲ findAll() : Iterable<UsuarioVO>
  ▲ findAllById(Iterable<Integer>) : Iterable<UsuarioVO>
  ▲ count() : long
  ▲ deleteById(Integer) : void
  ▲ delete(UsuarioVO) : void
  ▲ deleteAllById(Iterable<? extends Integer>) : void
  ▲ deleteAll(Iterable<? extends UsuarioVO>) : void
  ▲ deleteAll() : void

```

```

v C ServicioUsuarioRolImpl
  ▲ urr : UsuarioRolRepository
  ● save(S) <S extends UsuarioRolVO> : S
  ● saveAll(Iterable<S>) <S extends UsuarioRolVO> : Iterable<S>
  ● findById(Integer) : Optional<UsuarioRolVO>
  ● existsById(Integer) : boolean
  ● findAll() : Iterable<UsuarioRolVO>
  ● findAllById(Iterable<Integer>) : Iterable<UsuarioRolVO>
  ● count() : long
  ● deleteById(Integer) : void
  ● delete(UsuarioRolVO) : void
  ● deleteAllById(Iterable<? extends Integer>) : void
  ● deleteAll(Iterable<? extends UsuarioRolVO>) : void
  ● deleteAll() : void

```

```

v I ServicioUsuarioRol
  ^ save(S) <S extends UsuarioRolVO> : S
  ^ saveAll(Iterable<S>) <S extends UsuarioRolVO> : Iterable<S>
  ^ findById(Integer) : Optional<UsuarioRolVO>
  ^ existsById(Integer) : boolean
  ^ findAll() : Iterable<UsuarioRolVO>
  ^ findAllById(Iterable<Integer>) : Iterable<UsuarioRolVO>
  ^ count() : long
  ^ deleteById(Integer) : void
  ^ delete(UsuarioRolVO) : void
  ^ deleteAllById(Iterable<? extends Integer>) : void
  ^ deleteAll(Iterable<? extends UsuarioRolVO>) : void
  ^ deleteAll() : void

```

```

v C ServicioRolImpl
  ▲ rr : RolRepository
  ▲ loadUserByUsername(String) : UserDetails
  ▲ findByNombre(String) : Optional<RolVO>
  ▲ save(S) <S extends RolVO> : S
  ▲ saveAll(Iterable<S>) <S extends RolVO> : Iterable<S>
  ▲ findById(Integer) : Optional<RolVO>
  ▲ existsById(Integer) : boolean
  ▲ findAll() : Iterable<RolVO>
  ▲ findAllById(Iterable<Integer>) : Iterable<RolVO>
  ▲ count() : long
  ▲ deleteById(Integer) : void
  ▲ delete(RolVO) : void
  ▲ deleteAllById(Iterable<? extends Integer>) : void
  ▲ deleteAll(Iterable<? extends RolVO>) : void
  ▲ deleteAll() : void

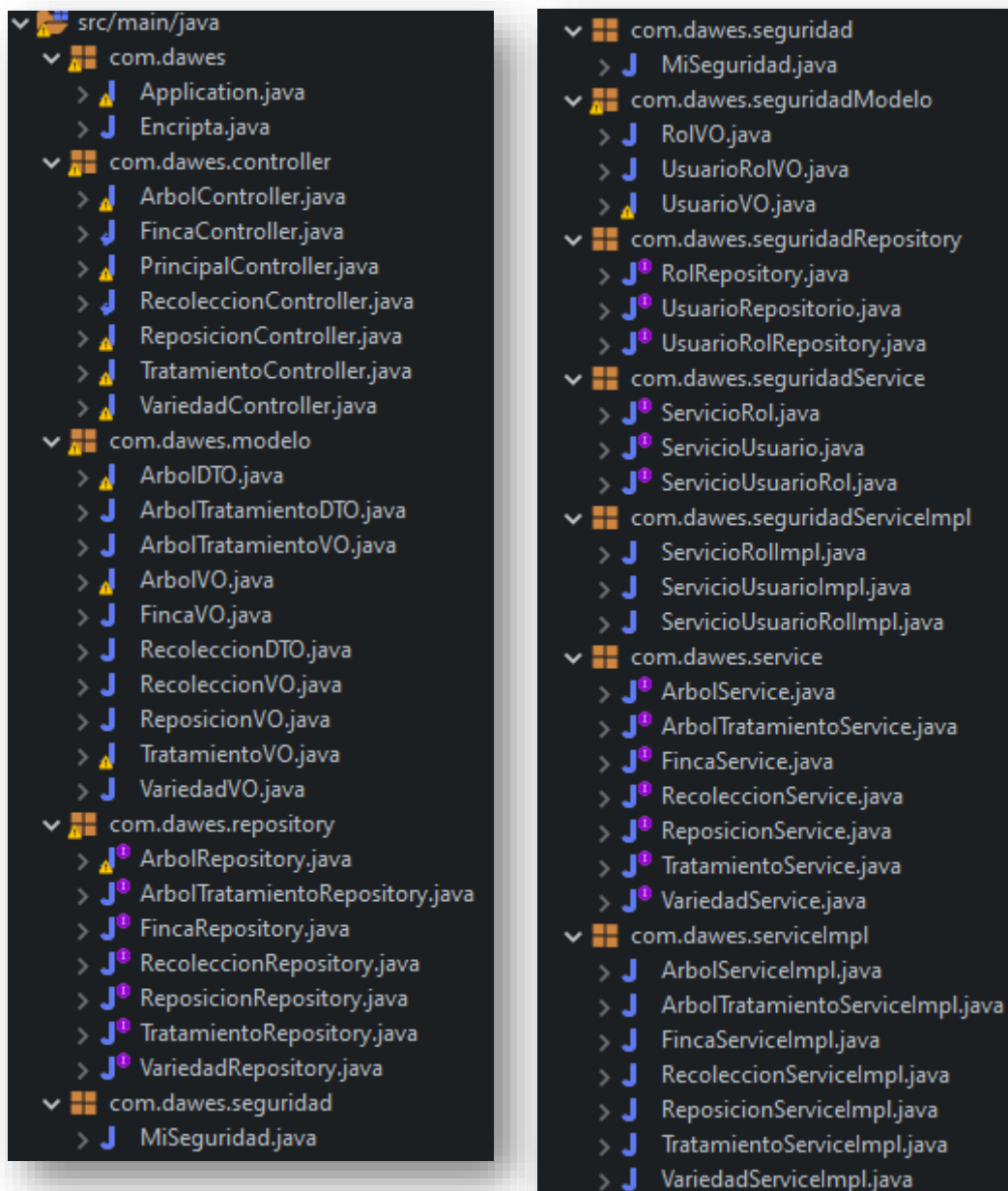
```

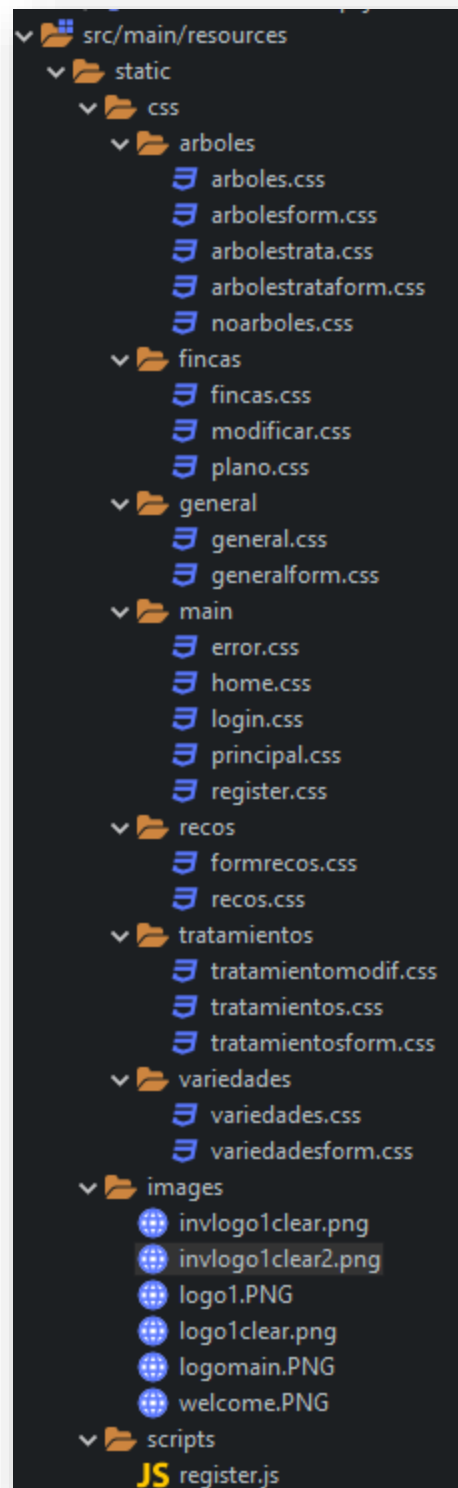
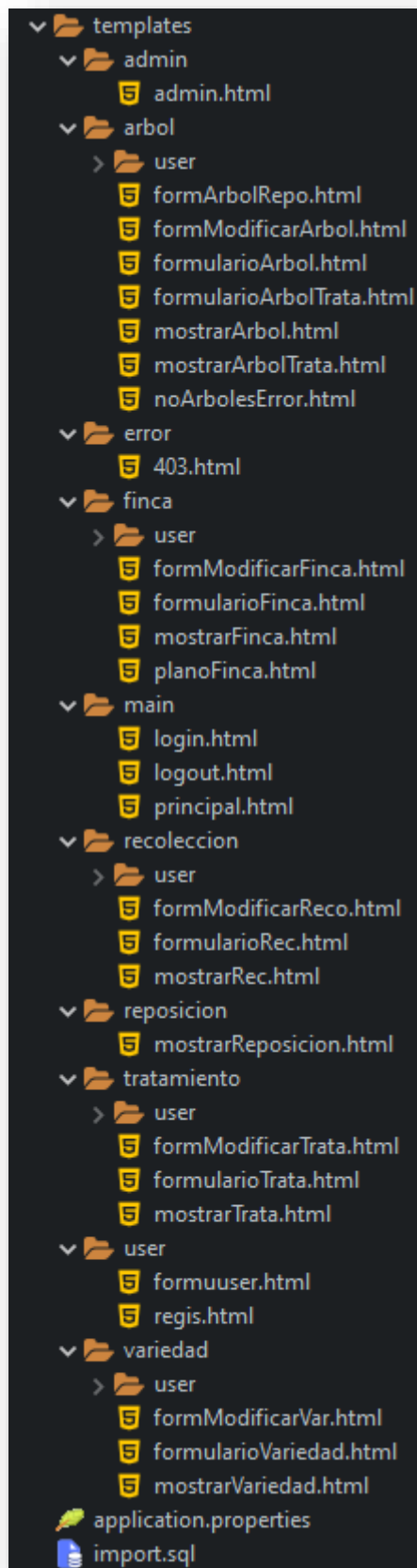
```

v I ServicioRol
  ▲ loadUserByUsername(String) : UserDetails
  ▲ findByNombre(String) : Optional<RolVO>
  ▲ save(S) <S extends RolVO> : S
  ▲ saveAll(Iterable<S>) <S extends RolVO> : Iterable<S>
  ▲ findById(Integer) : Optional<RolVO>
  ▲ existsById(Integer) : boolean
  ▲ findAll() : Iterable<RolVO>
  ▲ findAllById(Iterable<Integer>) : Iterable<RolVO>
  ▲ count() : long
  ▲ deleteById(Integer) : void
  ▲ delete(RolVO) : void
  ▲ deleteAllById(Iterable<? extends Integer>) : void
  ▲ deleteAll(Iterable<? extends RolVO>) : void
  ▲ deleteAll() : void

```

9. Estructura de carpetas en el IDE





10. Casos de uso

Imaginemos que un usuario tiene que añadir un tratamiento a un árbol de la finca Terrace de una variedad específica, en este caso Meana:

1. Accede a la página y se logea:

Login Form

Usuario
usuario1

Contraseña

Log in

Registrarse

Volver

2. Accede a Fincas:

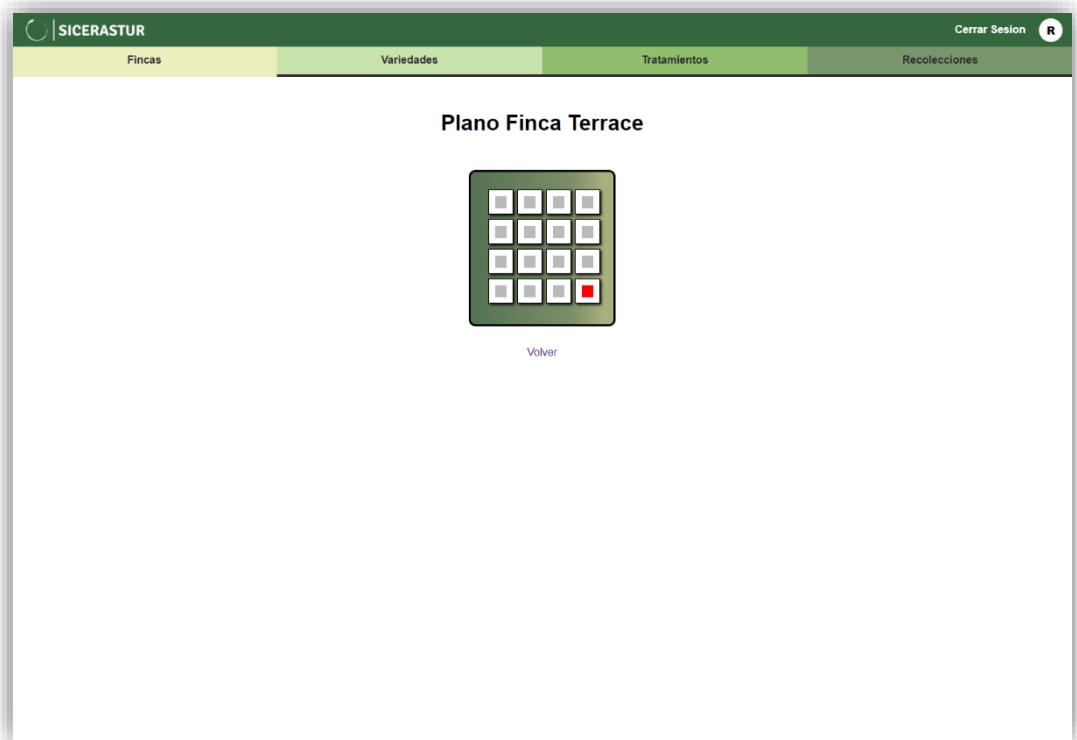
SICERASTUR Cerrar Sesión

Fincas Variedades Tratamientos Recolecciones

FINCAS

Denominacion	Direccion	Ref.	Superficie	Fecha	Arboles	Plano
Terrace	C/Molo	342R	200m²	2023-02-22	Arboles	Plano
Cereene	C/Mariano	265F	160m²	2023-05-11	Arboles	Plano
Afinonar	C/Monaco	273L	290m²	2022-07-29	Arboles	Plano

- Accede al palo de la finca para ver que árbol es que tiene la variedad Meana, en este caso la variedad meana esta representada por el cuadrado de color rojo en el plano.



- Clica en el cuadrado rojo para ver los datos del árbol:



- Clica en nuevo tratamiento para añadir un tratamiento al árbol:

The screenshot shows the 'Nuevo Tratamiento' (New Treatment) form within the SICERASTUR application. The form is centered on a white background. It has a green header bar with the SICERASTUR logo and a 'Cerrar Sesión' (Logout) button. Below the header, there are four tabs: 'Fincas', 'Variedades', 'Tratamientos', and 'Recolecciones'. The 'Tratamientos' tab is selected. The form itself has a green title bar 'Nuevo Tratamiento'. Inside, there is a dropdown menu labeled 'Tratamiento:' with 'fungicida' selected. Below the dropdown is a green 'Insertar' button and a purple 'Volver' link.

- Ya estaría añadido el tratamiento al árbol especificado:

The screenshot shows the 'Arbol De Terrace' (Terrace Tree) interface. At the top, there is a green header bar with the SICERASTUR logo and a 'Cerrar Sesión' button. Below the header, there are four tabs: 'Fincas', 'Variedades', 'Tratamientos', and 'Recolecciones'. The 'Tratamientos' tab is selected. The main content area shows a tree structure. The tree has a root node 'Arbol De Terrace' with a sub-node 'Fila: 4 / Columna: 4'. Below this, there are two tables: 'TRATAMIENTOS' and 'REPOSICIONES'. The 'TRATAMIENTOS' table has a header 'Nuevo Tratamiento' and contains one row with 'Nombre' 'Fungicida' and 'Fecha' '2023-06-06'. The 'REPOSICIONES' table has a header 'Nueva Reposición' and contains one row with 'Causa' and 'Fecha'.

11. Conclusión

En resumen, la aplicación web de Sicerastur cumple con todos los requisitos que se pedían en un principio.

- Sicerastur cuenta con la capacidad de ver, añadir, modificar y eliminar: fincas, variedades, tratamientos, arboles y recolecciones. Estas funcionalidades son la base del del proyecto.
- La seguridad implementada en Sicerastur permite distinguir entre un usuario registrado y administrador, cambiando las vistas y funcionalidades que se muestran dependiendo del usuario en activo en ese momento.
- La pagina web cuanta con un plano para poder visualizar todos los árboles de una finca de forma ordenada con estructura de cuadrícula, distinguiendo entre las diferentes variedades que hay en la finca por color.
- El despliegue de la aplicación fue de lo mas complicado, pero la aplicación debería ser accesible desde todos los dispositivos. Gracias a Heroku y GitHub el despliegue se realizó correctamente.



SICERASTUR

Proyecto DAWº2
2022-2023