

Cuaderno de campo

Daniel Baldazo Sánchez, Mauricio Sanchez Peña y Daniel Michel Cámara

Tutor del proyecto: Sergio Sánchez-Crespo Fernández



ÍNDICE

Introducción.....	4
Título del proyecto.....	4
Contexto y justificación del trabajo.....	4
Justificación/Antecedentes.....	5
Motivación del proyecto.....	5
Planificación del proyecto.....	6
Objetivos a conseguir.....	6
Objetivos extraordinarios.....	6
Objetivos extraordinarios imprevistos.....	7
API propia.....	7
API del SIGPAC.....	7
Análisis de riesgos.....	8
Diagramas de Gantt.....	9
Diagrama inicial.....	9
Diagrama final.....	9
DAFO.....	9
Debilidades.....	9
Amenazas.....	10
Fortalezas.....	10
Oportunidades.....	10
Matriz de DAFO.....	10
Análisis.....	11
Software desarrollado.....	11
Requisitos funcionales.....	11
Agricultores.....	11
Asesores.....	12
Explotaciones.....	12
Equipos de tratamiento.....	13
Parcelas.....	14
Cultivos.....	16
Tratamientos.....	18
Actualizar JSON productos.....	22
Requisitos no funcionales.....	23
Rendimiento.....	23
Usabilidad.....	23
Seguridad.....	23
Mantenibilidad.....	23
Compatibilidad y portabilidad.....	24
Fiabilidad.....	24
Escalabilidad.....	24
Diagrama de casos de uso.....	24
Diseño.....	25
Base de datos.....	25
Diagrama entidad-relación.....	25
Relaciones.....	25
Triggers.....	27
Borrados y actualizaciones.....	27

Diagrama de componentes/Diagrama de despliegue.....	28
Introducción.....	28
Funcionamiento.....	28
Localhost.....	28
Diseño de la interfaz del usuario.....	29
Página de inicio.....	29
Login.....	30
Menú del administrador.....	30
Diseño responsive.....	34
Desarrollo.....	36
Librería jsPDF para Generación de Informes.....	36
Descripción general.....	36
Integración en el proyecto.....	36
Envío del Informe por Correo Electrónico.....	37
Generación del PDF.....	37
Envío del PDF desde el Servidor.....	38
Inserción del JSON de productos y usos en la base de datos.....	39
Inserción Manual.....	39
Inserción Automática.....	40
Repositorio en GitHub.....	42
API.....	43
App.js.....	44
Config.js.....	45
Db.js.....	46
Server.js.....	47
Rutas y Endpoints.....	48
Hosting.....	50
Conexión con la base de datos.....	50
Servir Frontend desde Express.....	50
Sistema de login y control de sesiones.....	51
Detección del entorno de ejecución.....	53
Repositorio en GitHub.....	53
Diagrama de flujo.....	54
Carga del JSON.....	54
Flujo desde el principio.....	54
Pruebas realizadas.....	55
Control de bugs general.....	55
Validación de las dosis de un tratamiento.....	55
Comprobar si existe una parcela en SIGPAC.....	56
Inyecciones SQL.....	56
Producto final.....	57
Objetivos conseguidos.....	57
Mejoras a realizar.....	57
Conclusiones.....	58
Opinión general.....	58
Elementos aprendidos.....	58
Dificultades y fallos.....	58
Agradecimientos.....	59
Bibliografía.....	60

Introducción

Título del proyecto

Cuaderno de campo

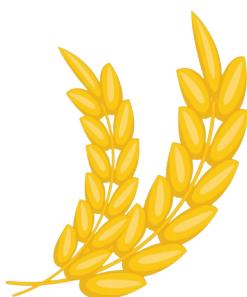
Contexto y justificación del trabajo

El cuaderno de campo es un registro obligatorio en España desde 2012, donde el titular de una explotación debe tener constancia de las actividades que realiza en sus explotaciones, de manera especial se deben registrar los tratamientos realizados en las parcelas donde se usan productos fitosanitarios. Debiendo llevar el control de la cantidad de producto que se emplea al tratarse de productos destinados al consumo humano, para vigilar la posible contaminación que pueda causarse por usar una dosis elevada de un producto fitosanitario en proporción al tamaño de las hectáreas de una parcela.

Hasta este momento, se podía tener un registro del cuaderno de campo mediante un excel o incluso en hojas impresas, donde el ministerio de agricultura proporcionaba las plantillas necesarias para llenar la información de las actividades de una explotación.

A partir del 1 de enero de 2023, se estableció que ya no se podría tener registro en los excel o las hojas impresas, sólo se podría tener constancia de las actividades de una explotación a través del formato digital, pero esta medida fue aplazada. El cuaderno de campo en formato digital sustituye a los excel o a las hojas impresas que se han estado usando hasta este momento, su uso será un requisito obligatorio para solicitar cualquier tipo de ayuda, indemnización o subvención como por ejemplo las PAC.

Inicialmente se esperaba que fuera obligatorio a partir de 2023 como se mencionó anteriormente, sin embargo ha sido prorrogado hasta del 1 de enero de 2026 que será obligatorio llevar el control de los productos fitosanitarios en toda la Unión Europea. Como aún no es obligatorio, se ha establecido una voluntariedad temporal para el uso del cuaderno digital. [Reglamento de Ejecución \(UE\) 2023/564](#)



Justificación/Antecedentes

Motivación del proyecto

La agricultura forma parte del sector primario, dónde se producen productos imprescindibles para la población, los alimentos. A su vez conlleva una gran responsabilidad en la salud de los consumidores, y por ello requieren gran cantidad de cuidados sanitarios y su correspondiente documentación.

Desde que se obligó a documentar todos los cuidados que se llevaban a cabo, esta documentación se ha hecho en papel adoptando el nombre vulgar de “papeleo”. A diferencia de otros sectores que se han digitalizado como el turismo o muchos restaurantes, el sector agrícola ha mantenido los mismos medios. Es por ello que surgen con gran necesidad herramientas como la web desarrollada.

La principal motivación es facilitar a los agricultores a cumplir la normativa del cuaderno digital de campo, ofreciendo un servicio web para que los ingenieros agrónomos (los encargados de los registros) hagan su trabajo con mayor facilidad, y que en un futuro los propios agricultores puedan abstenerse de un ingeniero.

La idea fue propuesta por nuestros profesores, y tras un tiempo de meditación ya que había en mente gran variedad de ideas a desarrollar, esta fue la propuesta dónde más tecnologías podíamos aprender e implementar todos los conocimientos aprendidos durante el ciclo. Por lo que decidimos aceptar el reto.



Planificación del proyecto

Objetivos a conseguir

Los objetivos mínimos que la aplicación web tendrá son:

- Dar de alta/baja a agricultores, asesores, explotaciones, parcelas, equipos de tratamiento, cultivos y tratamientos. Se incluye la gestión de los tratamientos y poder añadir detalles e información que le pertenecen a un agricultor con su parcela.¹
- Filtrado de productos según el tipo de cultivo, plaga y máximo valor permitido de la dosis mediante consultas.²
- Actualizar archivo JSON de productos de forma manual.
- Generación de Informes requeridos.³
- Enviar informes mediante email.

¹ La gestión y poder añadir detalles e información, se refiere a llevar a cabo el seguimiento y control mediante un formulario disponible en nuestra aplicación, la utilizará el ingeniero agrónomo para el correcto uso de los tratamientos utilizados para controlar las plagas de manera que se cumpla la normativa vigente.

² En este apartado pretendemos que los campos del formulario sean dinámicos, es decir que se rellenen automáticamente con los datos correspondientes y avise al ingeniero agrónomo si se ha pasado de la cantidad máxima de la dosis establecida de un producto, y que el mismo se pueda usar en una determinada plaga y tipo de cultivo permitido.

³ Si el ingeniero agrónomo lo desea puede exportar los datos de un tratamiento en formato PDF para poder imprimirlo.

Objetivos extraordinarios

En base a lo propuesto en los objetivos mínimos, tenemos como objetivos extraordinarios:

- Crear la aplicación multiusuario para que varias personas puedan interactuar con ella a la vez mediante hilos.
- Adaptar la aplicación para teléfonos móviles y tablets ya que en un principio está pensada para pantalla completa de ordenador.
- Crear mapas a partir de la localización que se pueda interactuar con ellos o al menos visualizarlos.
- Adaptar la aplicación para que la puedan utilizar también los agricultores, somos conscientes de que un gran sector de los agricultores pertenece a la población mayor por lo que habría que aplicar usabilidad a la aplicación web.
- Actualizar archivo JSON de manera automática.
- Seguridad para la protección de Datos (SSL/TLS)*

* Se hace referencia a que la página web utilice el protocolo HTTPS para asegurar los datos en internet, esto se consigue mediante un certificado digital.

Objetivos extraordinarios imprevistos

Tras proponer los objetivos mínimos y los extraordinarios pensábamos que serían objetivos suficientes para la aplicación, sin embargo a medida de que fuimos avanzando en el proyecto nos dimos cuenta de tener una API propia ya que nos ha permitido separar al backend del frontend y de esa manera se han desarrollado ambas partes de forma separada, permitiendo una mayor escalabilidad de la aplicación debido a las rutas que hemos desarrollado dentro de nuestra API.

Por otro lado, a finales de Abril descubrimos que el SIGPAC contaba con una API propia para realizar consultas, antes de esto las parcelas y los recintos se creaban de forma independiente, sin verificar su existencia. Gracias a la implementación de la API del SIGPAC en nuestra aplicación hemos podido automatizar las parcelas y los recintos. Ambas APIs se han incorporado de manera exitosa a nuestra aplicación.

API propia

Usamos la API que hemos desarrollado para realizar consultas a nuestra base de datos y llenar los campos del frontend de forma automática, ha sido desarrollada con Node.js y el framework Express, además dentro de la API hay instalados módulos que nos han facilitado el trabajo o nos han permitido conseguir más funcionalidades. También hemos conseguido aumentar la seguridad ya que contamos con un archivo .env donde se guardan datos confidenciales. En el apartado de desarrollo se verá detalladamente como funciona la API.

API del SIGPAC

La implementación de la API del SIGPAC nos ha permitido automatizar todos los datos de las parcelas, las referencias catastrales, las provincias, municipios y todos los recintos de una parcela con sus respectivos datos. Además nos ha permitido verificar la existencia de las parcelas ya que gracias a la incorporación de la API podemos verificar si una parcela existe en el SIGPAC.



Análisis de riesgos

Hemos analizado los posibles riesgos que puedan suceder durante el TFG hasta su entrega, los hemos categorizado con el nombre del riesgo, la probabilidad que ocurra, el impacto en el proyecto en caso de que ocurra y la solución planteada ante el problema.

RIESGO	PROBABILIDAD	IMPACTO EN EL PROYECTO	SOLUCIÓN
Pérdida de datos	MEDIA	MUY ALTA	Tener copias de seguridad en diferentes sitios aplicando el principio 3-2-1
Falta de tiempo para completar el proyecto	MEDIA-ALTA	MUY ALTA	Dedicar el tiempo completo al proyecto evitando otras tareas secundarias
Ataques informáticos o caídas en las tecnologías que usamos en el proyecto	MUY BAJA	MUY ALTA	Al tener los datos en nuestra base de datos, la aplicación se podría seguir usando sin las funcionalidades que requieren de servicios externos
Problemas al desarrollar el software con tecnologías que no hemos tenido experiencia	MEDIA	ALTA	Consultar libros, tutoriales, documentación oficial y en el peor de los casos cambiar de tecnología
Ausencia por enfermedad de uno o dos de los miembros del grupo	BAJA	BAJA	Hacer compañerismo y ayudarnos entre nosotros para poner el esfuerzo de la persona que no puede colaborar en ese momento
Problemas al entender tecnicismos del sector agrícola o leyes	MEDIA-ALTA	MUY ALTA	Contactar con personas de nuestro entorno que tengan conocimientos sobre el tema que desconocemos o usar la IA para obtener ayuda (siempre corroborando los datos)

Diagramas de Gantt

Diagrama inicial

Escaneando el siguiente código QR se puede visualizar el diagrama de Gantt con las tareas repartidas de los miembros del grupo en su versión inicial que se planteó al principio del proyecto.



https://docs.google.com/spreadsheets/d/1xbTCe73S4P7xaz68yvSxXNJUHInGpwDxz1vIYNQ9h44/edit?usp=drive_link

Diagrama final

En el siguiente diagrama se pueden visualizar los días que hemos tardado en completar las tareas y la planificación de las mismas en el tiempo real que se han conseguido completar.



https://docs.google.com/spreadsheets/d/1vjg_YHVZovWSHzuh5l8b99_JFoFi_a0mrbJ1yEhuw/edit?usp=drive_link

DAFO

Esta parte se encarga de analizar las Debilidades, Amenazas, Fortaleza, Oportunidades del proyecto con el fin de identificar las carencias y los objetivos principales que realizar.

Debilidades

- Las debilidades internas son:
- La poca experiencia en el sector al que va dirigida la aplicación.

- La falta de recursos tanto financieros como personales.
- Desconocimiento del movimiento del mercado.

Amenazas

- La fuerte competencia que hay en el mercado.
- El cambio de leyes o decretos que nos hagan replantear la lógica del proyecto.
- La barrera de entrada en un sector nuevo.
- Una estrategia de marketing mal ejecutada.

Fortalezas

- La motivación del equipo por sacar un producto completo.
- Las tecnologías que se utilizaran para el proyecto.
- La ayuda de personas experimentadas en el sector.
- La abundancia de información en internet proveniente de personas especializadas o del ministerio de agricultura, pesca y alimentación.

Oportunidades

- El crecimiento de este servicio digital ya que entra en vigor en toda la Unión Europea la obligación de un cuaderno de campo digital desde 2026.
- Las ayudas provenientes del gobierno.

Matriz de DAFO

DAFO	
Debilidades Falta de recursos y experiencia. Dependencia del equipo de trabajo.	Amenazas Competencia profesional. Cambio de leyes que afecte el proyecto.
Fortalezas Conocimientos técnicos y de herramientas. Acceso a profesores y tutores que nos guían. Motivación y creatividad del equipo.	Oportunidades Sector en crecimiento y oportunidades. Contactos iniciales en un sector nuevo.

Análisis

Software desarrollado

Hemos desarrollado una aplicación web que permite gestionar el cuaderno de campo de forma digital y facilitar la gestión de los datos a los ingenieros de los agricultores junto a sus explotaciones, equipos de tratamiento, parcelas, recintos, tratamientos y actualizar el JSON de productos. La aplicación permite generar informes de los tratamientos según los registros guardados de cada parcela cumpliendo con la ley, también se pueden visualizar las parcelas de los agricultores en el SIGPAC con un enlace directo desde nuestra propia aplicación.

Requisitos funcionales

Nuestra aplicación se divide en 8 secciones, cada una de ellas tiene varias funcionalidades relacionadas entre sí que nos permiten tener organizada la aplicación.

Agricultores

Esta sección consta de 3 funcionalidades para gestionar la información de los agricultores en nuestro sistema.

Alta de agricultores

Permite dar de alta agricultores en nuestra aplicación, se requieren todos los datos solicitados en la funcionalidad para que se pueda dar de alta un agricultor, se controlan errores en los campos que se introducen datos para asegurar que la información sea correcta, estas pruebas serán explicadas en el apartado de pruebas.

Baja de agricultores

La funcionalidad de dar de baja agricultores nos permite eliminar de la base de datos un agricultor, en la interfaz hay un select que al principio se cargan todos los agricultores pero permitimos filtrar tanto por nombre, apellido y DNI. Una vez seleccionado el agricultor se muestran sus datos en la interfaz y si se pulsa el botón de eliminar preguntará si estamos seguros, si aceptamos se eliminarán todos los datos del agricultor incluidas sus explotaciones, equipos, parcelas, recintos, tratamientos y la relación con sus asesores.

Editar datos agricultores

Permite editar los datos de un agricultor, al principio hay un select donde se cargan todos los agricultores pero permitimos filtrar tanto por nombre, apellido y DNI. Una vez seleccionado un agricultor se cargarán sus datos en la interfaz, se podrán editar: nombre, primer apellido, segundo apellido y el número de carnet del agricultor. De manera opcional

se podrá cambiar la contraseña si se pulsa el checkbox para restablecer la contraseña, en caso contrario su antigua contraseña se mantendrá.

Asesores

En esta sección se gestiona toda la información relativa a los asesores, cuenta con 5 funcionalidades.

Alta de asesores

Permite dar de alta asesores en nuestra aplicación, se requieren todos los datos solicitados en la funcionalidad para que se pueda dar de alta un asesor.

Baja de asesores

La funcionalidad de dar de baja asesores nos permite eliminar de la base de datos un asesor, en la interfaz hay un select que al principio se cargan todos los asesores pero permitimos filtrar tanto por nombre, apellido y DNI. Una vez seleccionado el asesor se muestran sus datos en la interfaz y si se pulsa el botón de eliminar preguntará si estamos seguros, si aceptamos se eliminarán todos los datos del asesor, incluidas las relaciones con sus agricultores.

Editar datos asesores

Permite editar los datos de un asesor, al principio hay un select donde se cargan todos los agricultores pero permitimos filtrar tanto por nombre, apellido y DNI. Se podrá cambiar: nombre, primer apellido, segundo apellido y su número de carnet.

Asignar asesores

Esta funcionalidad permite asignar un asesor a un agricultor, al principio se cargarán todos los agricultores y asesores en sus respectivos select y una vez seleccionado tanto un agricultor como un asesor se mostrarán sus datos en los inputs y se podrá realizar la asignación.

Desasignar asesores

En esta funcionalidad podemos eliminar la relación entre un asesor y un agricultor. Al principio se cargarán todos los agricultores de la aplicación en el select de agricultores y una vez seleccionado un agricultor se mostrarán sus datos y posteriormente se desbloqueará el select de asesores con los asesores de ese agricultor para poder designarlos. Tanto en el select de agricultores como en el de asesores se permite filtrar por nombre, apellido y DNI.

Explotaciones

En esta sección contamos con 3 funcionalidades que permiten la gestión de la información de las explotaciones de los agricultores. En cada funcionalidad se cargarán en el select de

agricultores todos los agricultores y cuando haya un agricultor seleccionado ya se podrá realizar cualquier operación con una explotación, en el caso de baja y edición se cargarán en los select de explotación las explotaciones de ese agricultor y se podrán filtrar por el nombre de la explotación.

Alta de explotaciones

Permite dar de alta una explotación, solamente requiere el nombre ya que la superficie depende de las parcelas de la explotación y por defecto al crear una explotación su superficie será 0 hectáreas.

Baja de explotaciones

Esta funcionalidad permite eliminar de la base de datos una explotación, si se elimina una explotación se borrarán todos los datos de la explotación incluidos los equipos asociados, parcelas, recintos y tratamientos. Antes de eliminar una explotación se muestran los datos de la explotación (id, nombre, parcelas totales y superficie total) y se pregunta si está seguro de eliminarla.

Editar datos explotaciones

Con esta funcionalidad se permite cambiar el nombre de la explotación, antes de actualizar los datos, se muestran en la interfaz y se debe de escribir el nombre que el usuario desee, en el input se cargará el nombre que tenía previamente.

Equipos de tratamiento

En esta sección hay 6 funcionalidades, los equipos de tratamiento son necesarios para poder realizar un tratamiento, el equipamiento se asocia a las explotaciones de un agricultor, se puede asignar el mismo equipamiento para distintas explotaciones y una explotación puede tener varios equipos de tratamiento. A continuación se explican las funcionalidades de esta sección.

Alta de equipamientos

Permite dar de alta un equipamiento en la aplicación pero no se asocia a ninguna explotación, todos los campos son obligatorios rellenarlos.

Baja de equipamientos

Permite eliminar un equipo de tratamiento, dar de baja un equipo de tratamiento conlleva desasociarse de todas las explotaciones y eliminarlo del sistema, sin embargo si un equipo ya ha realizado un tratamiento no se podrá eliminar ya que el borrado es restringido. Cuando se inicia la página web se cargarán todos los equipos de la base de datos en un select para poder seleccionar un equipo, se puede filtrar por nombre y por número de ROMA. Una vez seleccionado un equipo de mostrarán sus datos en la interfaz y será necesario pulsar el botón de eliminar un equipo, se preguntará al usuario si está seguro de eliminarlo.

Editar equipamientos

Esta funcionalidad nos permite actualizar los datos de un equipo de tratamiento, es necesario seleccionar un equipo al principio, están todos cargados en el select y se puede filtrar por nombre y por número de ROMA. Una vez seleccionado un equipo se cargarán todos los datos en los inputs y se podrán actualizar según el usuario lo requiera. Antes de realizar la actualización se pregunta al usuario si está seguro de actualizar los datos del equipo de tratamiento.

Asignar equipamientos

Permite asignar un equipo de tratamiento a una explotación, al principio se cargarán todos los agricultores en un select para poder seleccionar uno, se podrá filtrar por nombre, apellido o DNI, posteriormente se cargarán las explotaciones de ese agricultor y una vez seleccionada una explotación se desbloqueará el apartado del equipo de tratamiento con todos los equipos de tratamiento para poder seleccionar un equipo y asignarlo a una explotación.

Desasignar equipamientos

Esta funcionalidad permite eliminar la relación de un equipo de tratamiento y una explotación, su funcionamiento es similar al de asignar equipamiento pero cuando llegamos al paso final no se cargarán todos los equipos de la aplicación, solo se cargarán los equipos de la explotación de un agricultor previamente seleccionado. Una vez seleccionado un equipo se podrá designar de la explotación y se le preguntará al usuario si está seguro de realizar esa acción.

Crear y asignar equipamientos

Permite crear un equipo de tratamiento y asociarlo a una explotación a la vez, es una mezcla de crear equipamientos con asignar equipamientos, al principio se cargan todos los agricultores de la aplicación, posteriormente una vez seleccionado un agricultor se cargarán todas sus explotaciones y una vez seleccionada una explotación se introducen los datos del equipo de tratamiento. Una vez pulsemos el botón de crear y asignar equipamiento, se creará el equipo y posteriormente se asociará a la explotación.

Parcelas

En esta sección hay 4 funcionalidades, utilizamos la API del SIGPAC en esta sección, veremos detalladamente de qué manera la usamos en cada funcionalidad. En todas las funcionalidades siempre será necesario seleccionar un agricultor y una explotación (en ambos casos hay opción de filtrar) antes de que se desbloquee la sección de parcelas.

Alta de parcelas

Esta funcionalidad permite dar de alta una parcela en una explotación. Usamos la API del SIGPAC para verificar que los datos de la parcela introducidos por el usuario son reales, damos la opción al usuario mediante desplegables de seleccionar una provincia y municipio filtrando por nombre, también tenemos dos campos que sirven para identificar el nombre de

la parcela y la superficie declarada, estos dos campos no se verifican con la API ya que solo son añadidos extra. Una vez creada la parcela también usamos la API para crear todos los recintos de una parcela junto con sus usos y medidas. La suma de todos los recintos conforman la parcela que se hace automáticamente con triggers en la base de datos y la suma de todas las parcelas da lugar a la superficie total de la explotación que también se calcula sola mediante triggers. El número de referencia catastral también se guarda en la base de datos una vez creada una parcela existente.

Crear parcela nueva

Buscar código de provincia	Seleccione la provincia	Buscar municipio por nombre	Seleccione un municipio
<input type="text"/> Nombre de la provincia	<input type="text"/> Seleccione provincia	<input type="text"/> Buscar explotación	<input type="text"/> Seleccione municipio
Nombre de la parcela identificativa	Código de provincia	Código de municipio	Agregado
<input type="text"/> Nombre de la parcela	<input type="text"/> Código de la provincia parcela	<input type="text"/> Código del municipio parcela	<input type="text"/> Número de agregado
Zona			
Nombre de zona			
Polygono	Parcela	Superficie declarada en ha	
<input type="text"/> Número de polígono	<input type="text"/> Número de parcela	<input type="text"/> Superficie declarada	
Crear parcela			

Baja de parcelas

Esta funcionalidad permite eliminar una parcela de una explotación de un agricultor, dar de baja una parcela conlleva eliminar los recintos y los tratamientos de la parcela. El funcionamiento es similar de cargar los datos de agricultores y explotaciones es similar a la de alta de explotaciones, una vez seleccionada una explotación se cargarán toda la parcelas de la misma y se podrán filtrar por su nombre. Una vez seleccionada una explotación se visualizarán todos sus datos y se podrá dar de baja o visualizarla en el SIGPAC para verla en el mapa esa parcela en concreto.

Datos de la parcela

Buscar parcela por nombre	Seleccione la parcela del agricultor			
<input type="text"/> Buscar parcela	<input type="text"/> Olivos 1.8902 ha			
Número de identificación	Número del catastro	Nombre de la parcela identificativa	Código de provincia	Código de municipio
<input type="text"/> 45:63:0:0:9:4	<input type="text"/> 45063A009000040000WP	<input type="text"/> Olivos	<input type="text"/> 45	<input type="text"/> 63
Nombre del municipio	Agregado	Zona	Polygono	Parcela
<input type="text"/> ESPINOSO DEL REY	<input type="text"/> 0	<input type="text"/> 0	<input type="text"/> 9	<input type="text"/> 4
Superficie SIGPAC en ha	Superficie declarada en ha	Recintos	Tratamientos	
<input type="text"/> 1.8902	<input type="text"/> 10.0000	<input type="text"/> 6	<input type="text"/> 0	
Visualizar en SIGPAC		Eliminar parcela		



Editar datos parcelas

Esta funcionalidad permite editar el nombre identificativo y la superficie declarada de una parcela, el funcionamiento de cargar agricultores, explotaciones y parcelas es igual a la de baja de parcelas. También se puede visualizar los datos de la parcela y visualizarla en el SIGPAC una vez seleccionada una parcela.

Automatizar parcelas

Con esta funcionalidad podemos actualizar todos los datos de los recintos con los datos oficiales del SIGPAC, se creó esta funcionalidad ya que si un agricultor realiza cambios en su parcela añadiendo más recintos (en la vida real) los datos en el SIGPAC se actualizarán pero en la aplicación no, para ello existe automatizar parcelas que actualizará todos los recintos con sus usos, medidas, nuevos posibles recintos en la parcela y eliminación de recintos que ya no existen, también se puede visualizar la parcela en el mapa del SIGPAC. El funcionamiento de cargar el agricultor, sus explotaciones y parcelas es similar al de baja de parcelas.

Cultivos

Esta sección cuenta con 2 funcionalidades que nos permiten gestionar los cultivos de nuestros recintos o eliminarlos. El funcionamiento es el siguiente: se cargan al principio todos los agricultores que se podrán filtrar por su nombre o DNI, luego se seleccionará una explotación de todas las que se han cargado, posteriormente las parcelas de esa explotación finalmente nos aparecerán los recintos en un select que dependiendo de la funcionalidad es un poco diferente, pero se explicará detalladamente en la funcionalidad correspondiente.

Gestionar cultivos

En esta funcionalidad podemos establecer o cambiar el tipo de cultivo que tiene un recinto, por defecto no va a tener cultivos pero podemos establecerlos mediante un select donde estarán cargados todos los recintos y se puede ver la información de cada recinto. En el caso de que tuviera un cultivo aparecerá el cultivo que tiene actualmente ese recinto.

Selección de recintos

Seleccione los recintos para establecer cultivos

Selección

Seleccione los recintos

- 1 | PASTO ARBUSTIVO | 0.0613 | Sin cultivo
- 10 | VIALES | 0.0179 | Sin cultivo
- 2 | PASTO ARBUSTIVO | 0.1016 | Sin cultivo
- 4 | TIERRAS ARABLES | 1.0123 | Sin cultivo

Una vez seleccionados los recintos que queremos establecer un cultivo debemos de seleccionar el tipo de cultivo que queremos y el tipo de regadío. Como mínimo debe de haber un recinto seleccionado, un cultivo seleccionado y un tipo de regadío para poder establecer el cultivo. Se puede filtrar el tipo de cultivo por su nombre de todos los cargados en el select del tipo de cultivo y también se pueden visualizar la parcela seleccionada.

Selección del tipo de cultivo

Buscar tipo de cultivo por nombre Tipo de cultivo

Nombre del cultivo Tipo de regadío

Visualizar en SIGPAC **Establecer cultivo**

Baja de cultivos

Esta funcionalidad permite eliminar el tipo de cultivo y el tipo de regadío de un recinto, esto es importante ya que dar de baja un cultivo conlleva poner como null en la base de datos éstos campos y a la hora de realizar un tratamiento no van a salir los recintos que tengan como cultivo null. Hemos decidido hacerlo de manera individual ya que se puede visualizar el recinto seleccionado de forma individual en el mapa del SIGPAC y además podemos ver todos los datos del recinto.

🔍

Seleccione el recinto del agricultor

4 | 1.0123 ha

Número identificación	Recinto	Uso del recinto	Descripción del uso	Superficie del recinto en ha
45:63:0:0:9:4:4	4	TA	TIERRAS ARABLES	1.0123 ha

Tipo de cultivo

Tipo de cultivo	Tipo de regadío
Pastos y praderas	Secano

[Visualizar en SIGPAC](#)

[Eliminar cultivo](#)

Tratamientos

En esta sección hay 3 funcionalidades relacionadas con los tratamientos, todas tienen un común la carga de datos iniciales: selección del agricultor, explotación y parcela. Dependiendo de la funcionalidad requerirá unos datos u otros que se verán detalladamente en cada funcionalidad.

Alta de tratamientos

Esta funcionalidad permite realizar tratamientos en una parcela. Cuando tenemos seleccionada una parcela se cargarán en un select los tipos de cultivo que hay en esa parcela. La suma de las superficies de los recintos que tengan un mismo tipo de cultivo será la resultante a la hora de calcular las dosis del producto ya que las operaciones se hacen con la superficie del mismo tipo de cultivo restando todo lo que no sea igual.

Selección del tipo de cultivo

🔍

Tipo de cultivo

Pastos y praderas | 1.0736 ha

Nombre del cultivo	Superficie
Pastos y praderas	1.0736

Seleccione un cultivo

Pastos y praderas | 1.0736 ha

Melón | 0.0179 ha

Una vez seleccionado el tipo de cultivo el siguiente paso será seleccionar el tipo de plaga que afectan a ese cultivo, esto se realiza mediante consultas SQL.

18

Tipo de plaga

Buscar tipo de plaga

Nombre de la plaga

Plaga a controlar

Dicotiledóneas anuales

Dicotiledóneas anuales

Dicotiledóneas anuales

Dicotiledóneas plurianuales

Heliothis, Helicoverpa spp.

Una vez seleccionada la plaga, el siguiente paso sería seleccionar el producto que se pueda utilizar al tipo de cultivo anterior y que afecte a la plaga seleccionada. Ya que un producto se podría utilizar para un tipo de cultivo pero no afecta a un determinado agente o que si afecte al agente pero no se pueda utilizar en el tipo de cultivo, debe de cumplir ambas condiciones y lo controlamos con consultas SQL.

Selección y datos del producto

Buscar producto por nombre

Número

Productos para la plaga

CLINIC TF | ES-01098

Nombre del producto

CLINIC TF

Número

ES-01098

Fecha de Caducidad: 31-07-2028

Estado: Vigente

Dosis Mínima: 2

Dosis Máxima: 8

Unidad de Medida: l/ha

Plazo de Seguridad: NO PROCEDE

Volumen Caldo:

Aplicaciones: 1

Intervalo de Aplicaciones: -

Condicionamiento específico: En periodo vegetativo, aplicar mediante tratamiento localizado a una dosis de 2-8 l/ha. Para tratamientos de destrucción y renovación de pastos y praderas, aplicar mediante pulverización normal para eliminar el cultivo existente antes de volver a sembrar, a una dosis entre 3-8 l/ha.

Método de Aplicación: Otros (ver condic.)

Volumen Mínimo: 100

Volumen Máximo: 400

Unidades Volumen: l/ha

Último Tratamiento Realizado: No se han realizado tratamientos previos.

Cantidad de dosis máxima

8

Unidad de medida de la dosis

l/ha

Una vez seleccionado el producto nos aparecerán todos los datos del mismo para que el especialista decida si se puede usar el tratamiento según los datos que se le ofrecen. Los campos de realizar tratamiento serán desbloqueados cuando haya un producto seleccionado, el siguiente paso será establecer la cantidad de la dosis respecto a las hectáreas introducidas que no podrán ser de mayor tamaño de las hectáreas del mismo tipo de cultivo seleccionado anteriormente.

Realizar tratamiento

Cantidad de producto fitosanitario y superficie tratada

La Dosis Máxima del Producto CLINIC TF para 1.0736 ha es: 8.589 l

Cantidad del Producto en: l/ha	Superficie tratada expresada en Ha	Fecha del tratamiento
Cantidad aplicada de producto	Superficie tratada de la parcela	dd / mm / aaaa <input type="button" value=""/>
		<input type="button" value="Validar datos"/>

Si pulsamos el botón amarillo, solo nos servirá para saber si los datos que hemos introducido son correctos, se comprueba que las dosis mínimas y máximas son las correctas, que la fecha del tratamiento no sea mayor al día de hoy y que las hectareas introducidas no sean mayor a las de los recintos con ese tipo de cultivo, además de que los datos introducidos sean números. Posteriormente seleccionaremos el carnet del aplicador del tratamiento que podrá ser el del propio agricultor o el de sus asesores.

Carnet del aplicador del tratamiento

Filtrar por nombre o DNI del asesor	Aplicadores disponibles
<input type="text"/> Nombre o DNI del aplicador o asesc	Juan Sánchez 92904489Z <input type="button" value=""/>
Número de carnet del aplicador seleccionado	Seleccione el aplicador
12	Agricultor Juan Sánchez 92904489Z <input type="button" value=""/>
	Asesores Hilario Sánchez 40043601A

Finalmente se introduce el equipo de tratamiento, aparecerán todos los equipos asociados a la explotación. Cuando pulsemos el botón verde se comprobará que todos los campos estén rellenos y que los datos de la dosis, superficie y las fechas sean correctas.

Equipo de tratamiento

Buscar equipo tratamiento por nombre	Seleccione un equipo de tratamiento		
<input type="text"/> Buscar equipo de tratamiento	Roll Royce 123456 <input type="button" value=""/>		
Nombre del equipo de tratamiento	Número de R	Fecha de inspección	Fecha de última inspección
Roll Royce	123456	06/05/2025	13/05/2025
<input type="button" value="Realizar tratamiento"/>			

Eliminar tratamientos

Esta funcionalidad permite eliminar un tratamiento ya creado, al igual que en alta de tratamientos hay que seleccionar un agricultor, después la explotación, la parcela y en este caso hay que seleccionar el tratamiento de la parcela que queramos eliminar, se puede filtrar por fechas. A continuación se muestra una captura de esta funcionalidad.

Datos del tratamiento

Fecha de inicio	Fecha final	Seleccione el tratamiento a eliminar		
<input type="text" value="05/05/2025"/> 	<input type="text" value="13/05/2025"/> 	<input type="button" value="Pastos y praderas CLINIC TF 13-05-2025"/>		
ID tratamiento	Número de identificación parcela	Tipo de cultivo de la parcela	Fecha del tratamiento	Plaga controlada
1	45:63:0:0:9:4	Pastos y praderas	13/05/2025	Dicotiledóneas anuales
Producto fitosanitario aplicado	Número de registro del producto	Cantidad de producto aplicada	Unidad de medida de la dosis	Superficie tratada expresada en Ha
CLINIC TF	ES-01098	3	l/ha	0.8000
Superficie en ha del cultivo	Número de carnet del aplicador			
1.0736	12			

Datos del equipo de tratamiento

Número de ROMA del equipo	Nombre del equipo de tratamiento	Fecha de adquisición	Fecha de última inspección
123456	Roll Royce	06/05/2025	13/05/2025

 Eliminar tratamiento

Cuando pulsamos el botón rojo se le preguntará al usuario si está seguro de eliminar el tratamiento.

Generar informes

Esta funcionalidad permite crear informes de un tratamiento, la selección de datos del tratamiento es igual que la de eliminar tratamiento solo que ahora tenemos dos opciones, podemos enviar el informe mediante un PDF por correo electrónico o que se descargue en nuestro ordenador.

Enviar informes

Enviar informe por correo electrónico	 <input type="text"/>	<input type="button" value="Enviar informe correo"/>
	<input type="button" value="Generar informe PDF"/>	

El PDF se visualiza de la siguiente forma:



Informe del Tratamiento

Fecha de generación: 14/5/2025

Campo	Valor
ID Tratamiento	1
ID Parcela	45:63:0:0:9:4
Tipo de Cultivo	Pastos y praderas
Fecha de Tratamiento	2025-05-13
Plaga Controlada	Dicotiledóneas anuales
Producto Aplicado	CLINIC TF
N.º Registro Producto	ES-01098
Cantidad Aplicada	3
Unidad de Medida	l/ha
Superficie Tratada (ha)	0.8000
Superficie de Cultivo (ha)	1.0736
Carnet del Aplicador	12

Campo	Valor
N.º ROMA	123456
Nombre del Equipo	Roll Royce
Fecha de Adquisición	2025-05-06
Última Inspección	2025-05-13

Actualizar JSON productos

En esta sección tenemos solamente una sola funcionalidad pero dentro de ella tenemos dos opciones, podemos actualizar el JSON de productos autorizados de forma manual o de forma automática, a continuación se explican ambas funcionalidades y que requisitos se necesitan.

Actualizar JSON de productos



Introduzca el archivo JSON para actualizar los productos

 Ningún archivo seleccionado

Actualizar el JSON de productos de manera automática

Actualización manual

La actualización manual requiere tener el JSON previamente descargado en nuestro ordenador, se puede obtener el link de donde se encuentra el archivo si pulsamos el título “Actualizar JSON de productos” que nos redirigirá a la página oficial donde podemos obtener el JSON ([Inicio - Consultas Fitosanitarios](#) en el apartado Resúmenes el cuarto enlace de la columna izquierda), posteriormente tenemos que adjuntarlo donde pone “Seleccionar archivo” y finalmente pulsar el botón verde para actualizar, añadir los productos y los usos en la base de datos.

Actualización automática

En la actualización automática usamos la librería puppeteer (se explica su funcionamiento en el apartado de desarrollo) simplemente con pulsar el botón amarillo donde pone “actualización automática” se podrán actualizar, añadir los productos y los usos en la base de datos en la última versión del JSON proporcionado por el ministerio, este proceso se ejecuta en segundo plano, pero se puede visualizar como se hace todo el proceso de obtener el JSON en la página web.

Requisitos no funcionales

Rendimiento

La aplicación debe responder siempre de forma fluida, sin llegar a bloquearse en ningún momento. Solamente a la hora de actualizar el JSON tanto de forma manual como de forma automática, se requerirá un intervalo de 10 a 30 segundos dependiendo del procesador del sistema.

Usabilidad

La interfaz ha sido diseñada con HTML y CSS de forma clara e intuitiva, apta para cualquier tipo de usuario ya que no requiere de tutoriales o formación para su entendimiento. Está estructurada con formularios, listas seleccionables y botones minimizando errores o bugs.

Seguridad

Aunque la aplicación se ejecuta en entorno local y no requiere de autenticación, las contraseñas almacenadas en la base de datos se cifran utilizando la librería *bcrypt*.

Mantenibilidad

El código está organizado modularmente: primero se definen las variables que capturan elementos del DOM, después se declaran funciones con lógica reutilizable, y por último se gestionan las acciones del sistema mediante programación orientada a eventos. Gracias a

esta estructura se puede modificar o ampliar funcionalidades de forma notablemente sencilla.

Compatibilidad y portabilidad

Funciona correctamente en navegadores modernos si se ejecuta con un emulador de servidor local (Live Server) que contenga una dirección IP *loopback* (127.0.0.1 - 127.255.255.254). Requiere un entorno que soporte Node.js, MySQL y librerías externas como *puppeteer*.

Fiabilidad

Los productos fitosanitarios y sus especificaciones se obtienen de un archivo JSON generado por el Gobierno de España (<https://servicio.mapa.gob.es/regfiweb>). Los recintos generados automáticamente al dar de alta una parcela se obtienen del [Catálogo de Servicios SIGPAC](#).

La aplicación funciona perfectamente sin conexión a internet salvo las funcionalidades: enviar por correo el informe generado, actualización del JSON de forma automática, automatización y alta de parcelas.

En cuanto a la base de datos, hay restricciones en los campos que no deben ser duplicados, claves foráneas con *ON DELETE CASCADE* y *ON UPDATE CASCADE* y tenemos *TRIGGERS* que mantienen la coherencia en cálculos de la superficie de parcelas y explotaciones.

Escalabilidad

La interfaz está diseñada y programada para un único ingeniero, pero la base de datos admite varios y una sencilla integración de una nueva tabla para un super administrador (gestor de ingenieros).

El login está diseñado, aunque sin su correspondiente funcionalidad ya que solamente contamos con un usuario para la aplicación.

Diagrama de casos de uso

A continuación se muestra un código QR para acceder a una carpeta de drive donde se encontrará el diagrama de casos de uso y su explicación.



<https://drive.google.com/drive/folders/1GNSSpibgzbjtWXt5fOGKKRtzEYyuflc7M?usp=sharing>

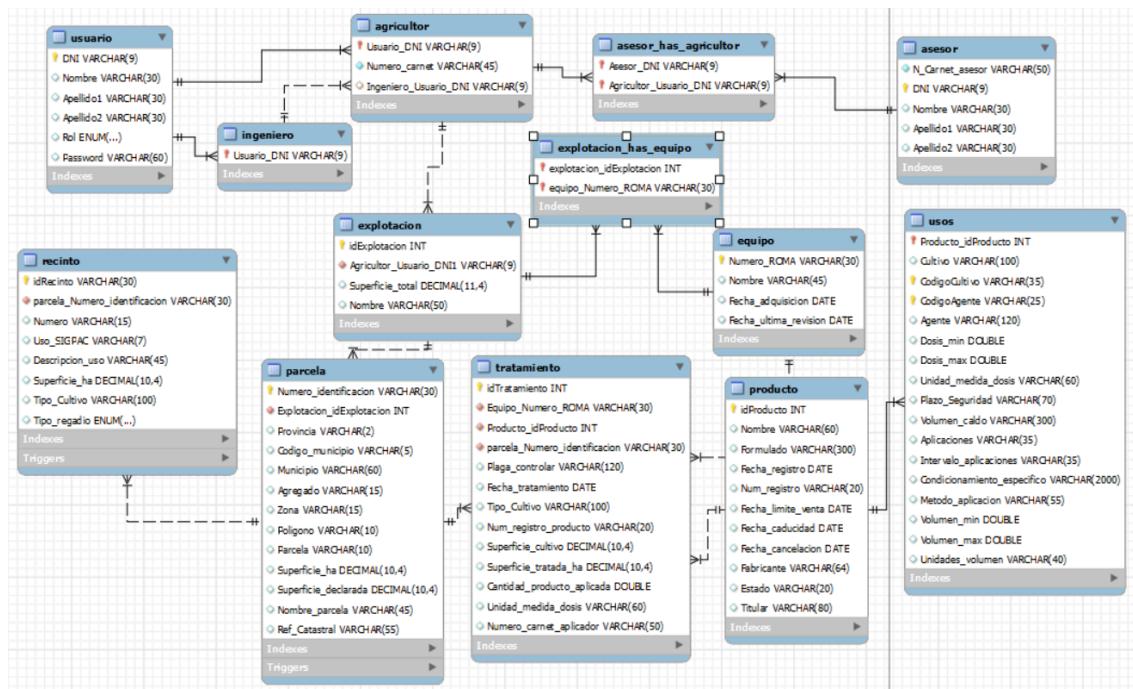
Diseño

Base de datos

Para la base de datos hemos decidido usar una relacional que va a ser MySQL, tuvimos un debate al principio del proyecto sobre si usar una base de datos relacional o no relacional como MongoDB. Hemos decidido el sistema gestor de bases de datos MySQL (base de datos relacional) porque en nuestra aplicación hay muchos datos relacionados y se puede aprovechar mejor el lenguaje de consulta SQL, ya que en nuestro caso va a ser más sencillo poder relacionar bien los datos gracias al modelado relacional. Para la comunicación y la programación de la base de datos hemos utilizado Node.js.

Diagrama entidad-relación

A continuación se muestra el diagrama entidad-relación de nuestra base de datos en MySQL Workbench.



Relaciones

Tabla usuario

Es la tabla que almacena la información de todos los usuarios del sistema.

- Relación 1:1 con agricultores.
- Relación 1:1 con ingenieros.

Tabla ingeniero

En esta tabla se guardan los ingenieros del sistema, en nuestro caso solamente hay un ingeniero pero la aplicación se podría escalar para que hubiera varios ingenieros y que controlen a sus agricultores.

- Relación 1:N con agricultores

Tabla agricultor

En la tabla de agricultores se guardan todos los agricultores de nuestra aplicación, tienen a un ingeniero asignado como vimos anteriormente, los agricultores pueden tener explotaciones y asesores.

- Relación N:M con asesores (se genera tabla intermedia)
- Relación 1:M con explotaciones

Tabla de asesores

En la tabla de asesores se guardan los profesionales que informan a los agricultores sobre sus explotaciones, como se mencionó anteriormente tiene una relación N:M con la tabla de agricultores.

Tabla de explotaciones

Son las unidades productivas de los agricultores.

- Relación N:M con equipos de tratamiento (se genera tabla intermedia)
- Relación 1:N con parcelas

Tabla parcela

En la tabla de parcelas se guardan las divisiones físicas de las explotaciones.

- Relación 1:N con recintos
- Relación 1:N con tratamientos

Tabla recinto

Son las divisiones físicas que hay dentro de una parcela, como se mencionó anteriormente un recinto está asociado a una única parcela.

Tabla equipos

En la tabla de equipos se guardan el material con el que se realiza un tratamiento y está asociado a una o varias explotaciones como se mencionó anteriormente.

- Relación 1:N con tratamientos

Tabla tratamiento

En esta tabla se guardan las altas de tratamientos que se realizan en las parcelas, se requiere de un equipo de tratamiento como se mencionó anteriormente, también se usará un producto por cada tratamiento.

- Relación N:1 con producto

Tabla producto

En esta tabla se guardan los productos fitosanitarios que provienen del JSON que ofrece el ministerio de agricultura, se relaciona con tratamientos como se mencionó anteriormente y también se registran los usos en otra tabla.

- Relación 1:N con usos

Tabla usos

En esta tabla se registran los diferentes usos que un producto tiene según la plaga o el tipo de cultivo.

Triggers

Usamos triggers en las tablas: parcelas y recintos para la actualización automática de la superficie de las parcelas y las explotaciones.

Hay tres triggers programados en las tablas mencionadas anteriormente para cada operación (*INSERT, UPDATE, DELETE*) que afecta de manera directa a la superficie de las explotaciones o parcelas.

- La superficie total de una explotación es la suma de las superficies de sus parcelas.
- La superficie total de una parcela es la suma de las superficies de sus recintos.

Borrados y actualizaciones

En toda la base de datos tenemos *ON DELETE CASCADE* y *ON UPDATE CASCADE* en las claves foráneas, solo hay una excepción y es en la tabla de tratamiento la clave foránea del equipo de tratamiento tiene *ON DELETE RESTRICT* ya que si un equipo ha sido utilizado en un tratamiento no se podrá borrar el equipo, de esta forma podemos mantener el historial de tratamientos de forma correcta.

Diagrama de componentes/Diagrama de despliegue

Introducción

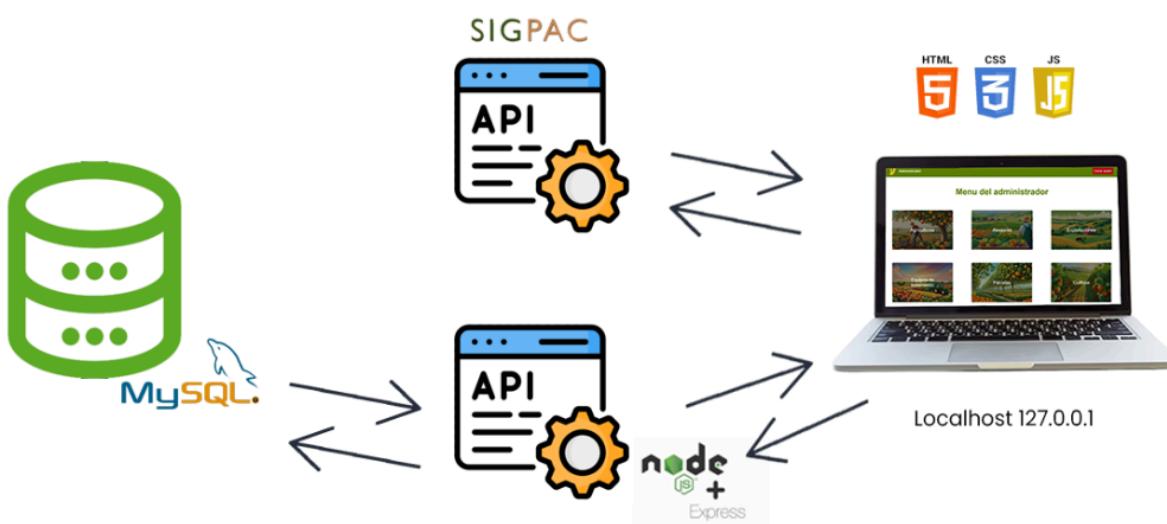
Nuestra aplicación consta de una base de datos MySQL, una API desarrollada con Node.js y Express, el front-end está diseñado con HTML, CSS y JavaScript y también usamos la API pública del SIGPAC.

Funcionamiento

Desde el front-end se realizan peticiones tanto a nuestra API como a la API del SIGPAC, se hace una petición a la API propia y una consulta a la base de datos para llenar los campos del front-end. La API del SIGPAC la utilizamos para verificar que los datos que el ingeniero ha introducido son correctos al crear una parcela, también para obtener todos los recintos de una parcela con sus respectivos datos, referencias catastrales y también se usa para obtener datos de las provincias y sus municipios.

Localhost

La aplicación funciona en local, requiere de un emulador que haga esta funcionalidad como por ejemplo Live Server y una dirección IP en el rango (127.0.0.1 - 127.255.255.254). También es necesario Node.js instalado en nuestro equipo con sus respectivos módulos que ya están incluidos en el proyecto. La base de datos y el front-end también están alojados dentro del mismo localhost.



Diseño de la interfaz del usuario

Para crear la aplicación web hemos utilizado HTML, CSS y JavaScript. La aplicación cuenta con una página de inicio donde presentamos nuestro proyecto, un login, un menú de administrador donde se encuentran ordenadas las funcionalidades según su uso y dentro de cada sección se encuentran las funcionalidades individuales. En todas las páginas web de nuestro proyecto hay un *footer* y en cada funcionalidad individual hay un *navigation* (nav) que nos permite retroceder al lugar que el usuario desee. También el diseño es *responsive* por lo que se puede usar en diferentes dispositivos con tamaños de pantallas distintos.

Página de inicio

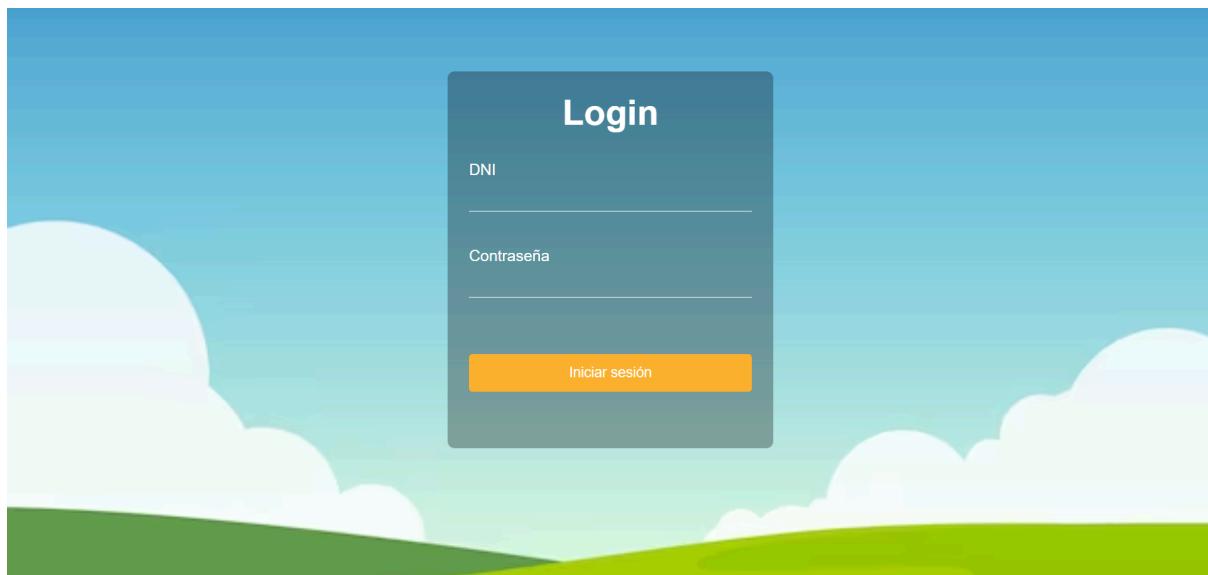


Nuestra aplicación WEB

Gestione su cuaderno de campo de forma digital gracias a nuestra aplicación web que le permite organizar de manera detallada y fácil: agricultores, asesores, maquinaria agrícola, explotaciones, parcelas, cultivos y tratamientos con productos fitosanitarios según la normativa vigente.



Login



Menú del administrador



The administrator menu has a green header bar. On the left is a golden gear icon followed by the text "Administrador". On the right is a red "Cerrar sesión" button. Below the header is a title "Menu del administrador" in green. The menu items are arranged in a grid:

- Agricultores**: An illustration of a farmer harvesting oranges from a tree.
- Asesores**: An illustration of a farmer standing next to a table full of various fruits and vegetables.
- Explotaciones**: An illustration of a tractor working in a field under a cloudy sky.
- Equipos de tratamiento**: An illustration of a green tractor spraying a field of fruit trees.
- Parcelas**: An illustration of a long row of orange trees in a field.
- Cultivos**: An illustration of a field with rows of strawberry plants.
- Tratamientos**: An illustration of a dense crop of green plants with small red and yellow fruits.
- Actualizar JSON productos**: An illustration of various agricultural products in containers, including bottles and jugs.



Gestión de agricultores



The main page features a large illustration of a smiling farmer with a beard and hat, driving a red tractor across a green field with rolling hills in the background. The title "Gestión de agricultores" is centered over the tractor. At the top right are "Menu administrador" and "Cerrar sesión" buttons.



Alta de agricultores



Baja de agricultores



Editar datos agricultores



The footer includes the Salesianos Domingo Savio logo, the SIGPAC logo, the coat of arms of Spain, and the text "GOBIERNO DE ESPAÑA" and "MINISTERIO DE AGRICULTURA, PESCA Y ALIMENTACIÓN". Below the footer is the text "Cuaderno de campo".

Gestión de asesores



The main page features a large illustration of a smiling farmer with a beard and cap, standing next to a large orange tractor. The title "Gestión de asesores" is centered over the farmer. At the top right are "Menu administrador" and "Cerrar sesión" buttons.



Alta de asesores



Baja de asesores



Editar datos asesores



Asignar asesores

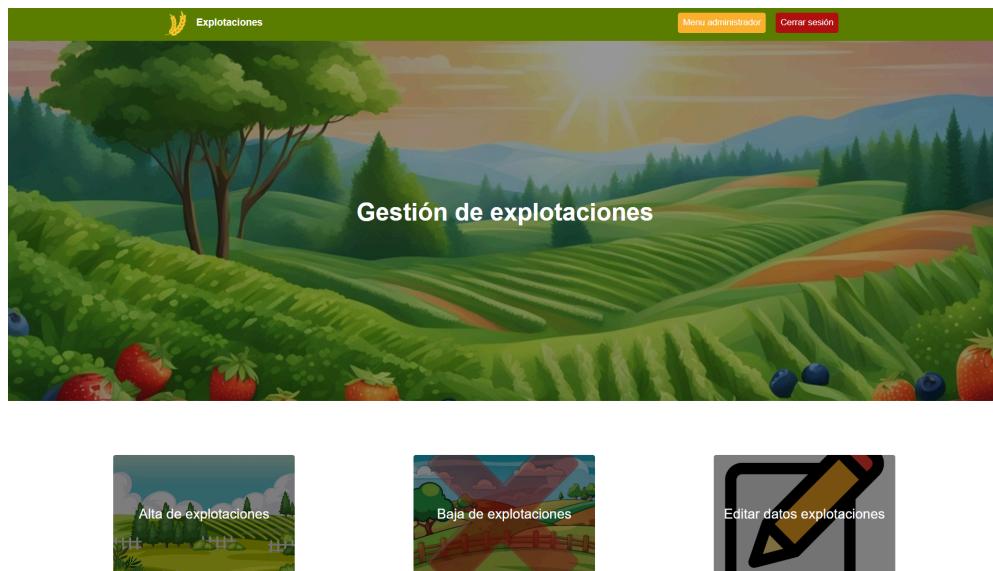


Desasignar asesores

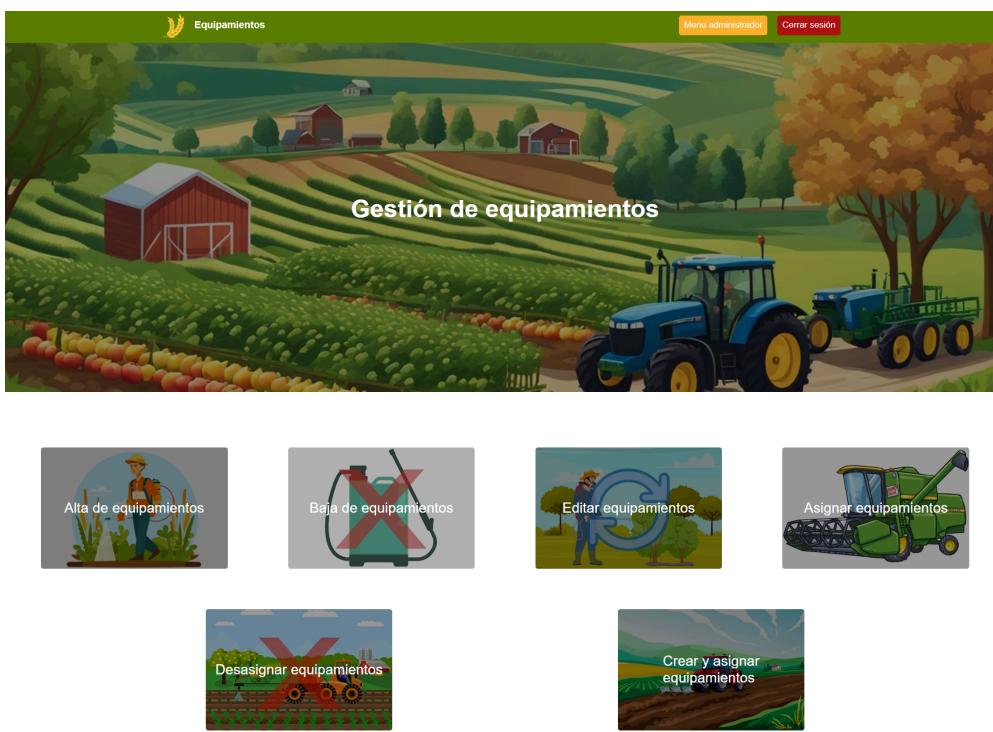


The footer includes the Salesianos Domingo Savio logo, the SIGPAC logo, the coat of arms of Spain, and the text "GOBIERNO DE ESPAÑA" and "MINISTERIO DE AGRICULTURA, PESCA Y ALIMENTACIÓN". Below the footer is the text "Cuaderno de campo".

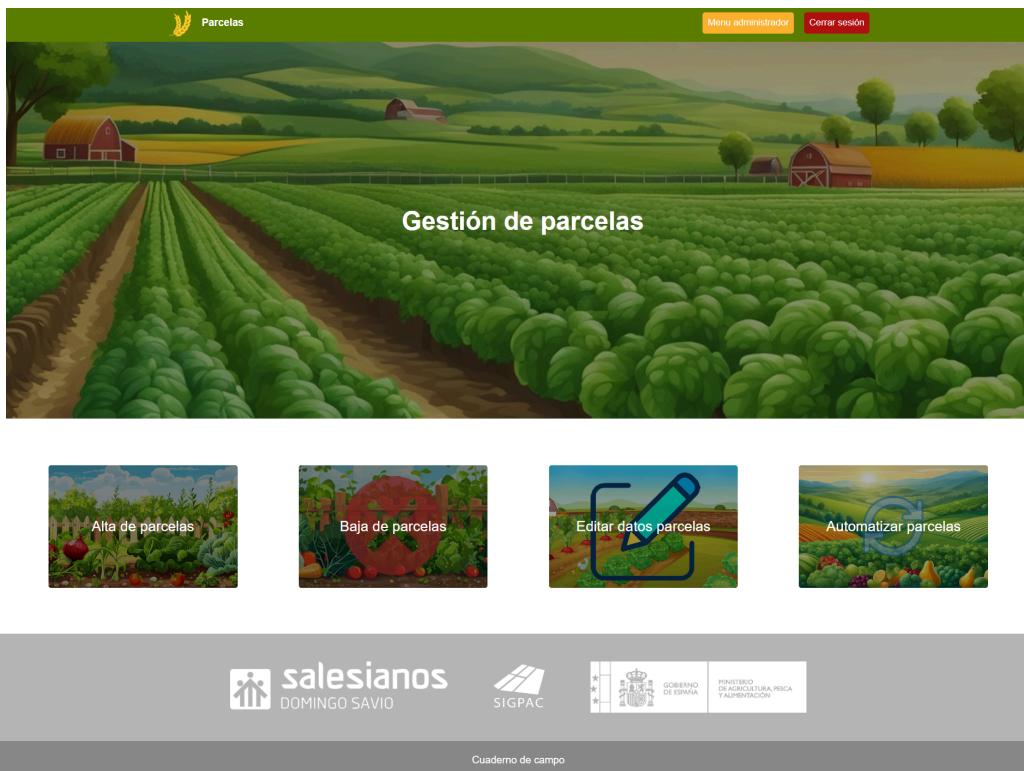
Gestión de explotaciones



Gestión de equipamientos




Gestión de parcelas



Parcelas

Gestión de parcelas

Alta de parcelas

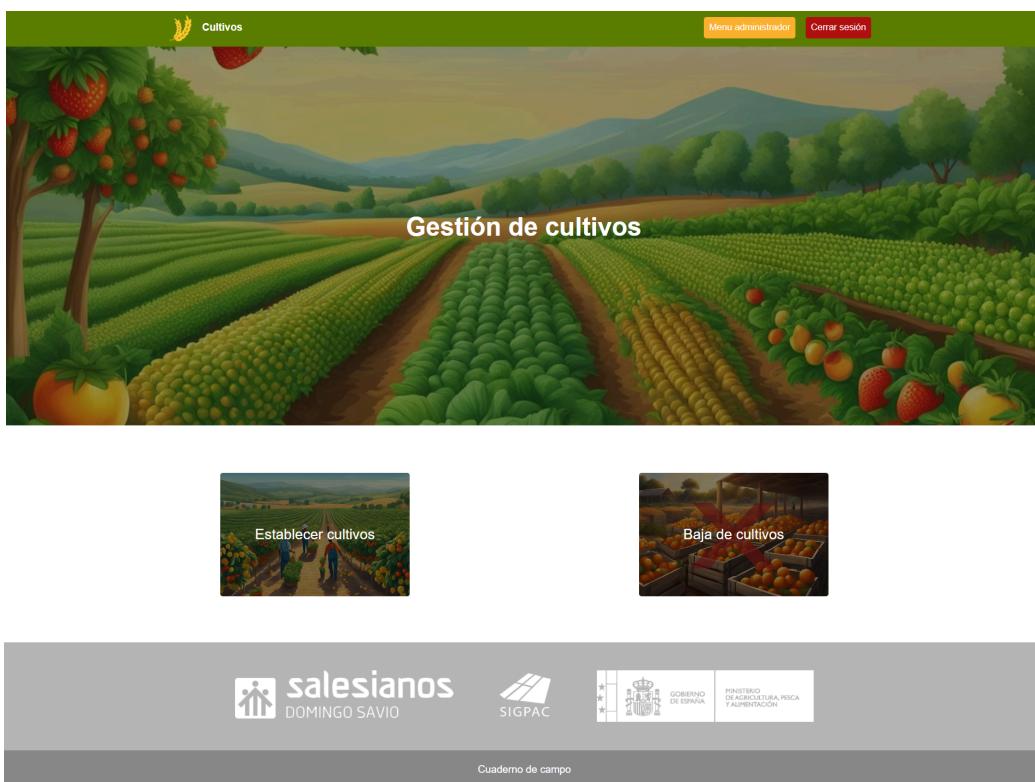
Baja de parcelas

Editar datos parcelas

Automatizar parcelas

Cuaderno de campo

Gestión de cultivos



Cultivos

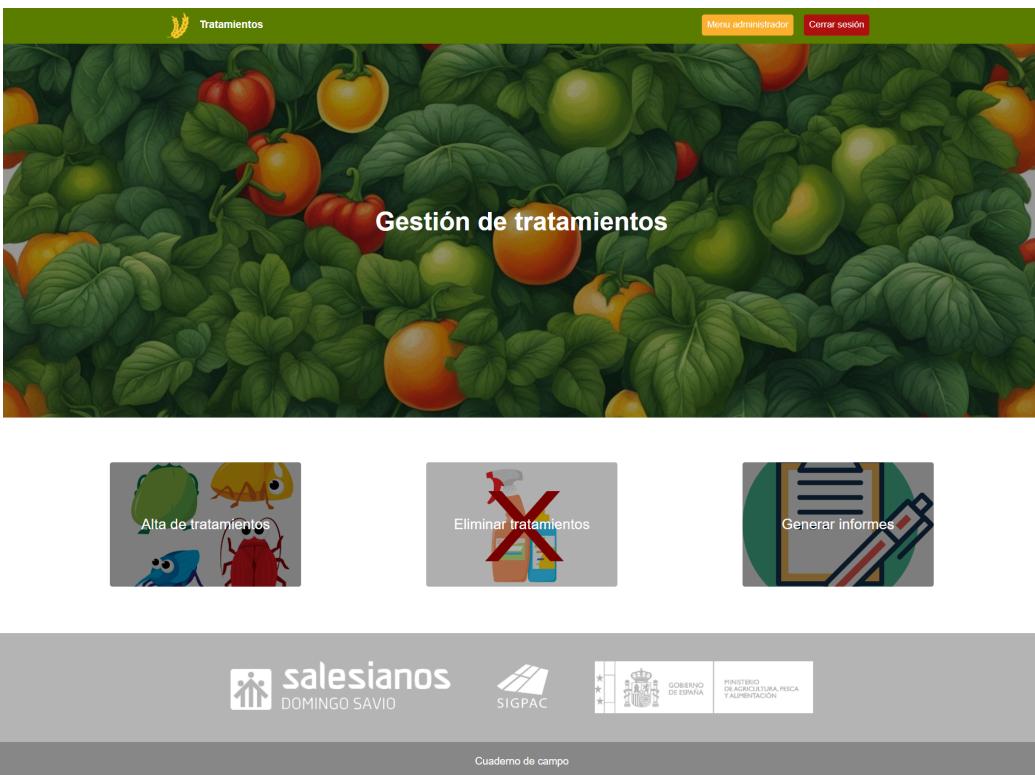
Gestión de cultivos

Establecer cultivos

Baja de cultivos

Cuaderno de campo

Gestión de tratamientos



Actualizar JSON de productos

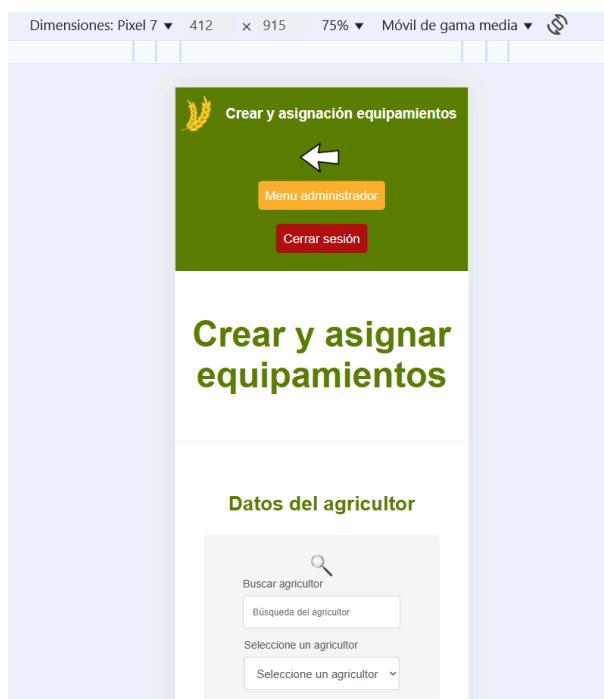


Diseño responsive

El CSS está diseñado como responsive, es decir se ajusta a pantallas de diferentes tamaños.

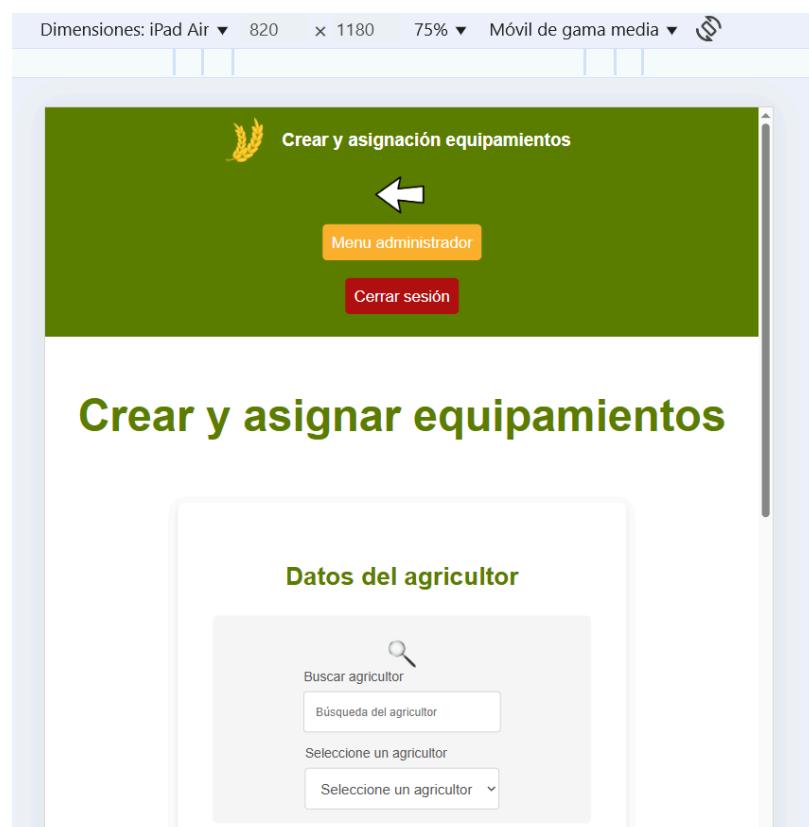
Móviles

A continuación se muestra como se visualiza la aplicación en un Pixel 7.



Tablets

A continuación se muestra como se visualiza la aplicación en un iPad Air.



Desarrollo

Librería jsPDF para Generación de Informes

Descripción general

La librería *jsPDF* permite generar archivos PDF directamente desde el navegador. En este proyecto, se ha utilizado para generar informes PDF con información sobre los tratamientos realizados y el respectivo equipo utilizado.

Integración en el proyecto

En el archivo generar_informes.js, se utiliza en 2 funciones:

- generarPDFInformeTratamientosYEquipo(): genera un documento PDF con los datos del tratamiento seleccionado.
- generarPDFPromesa(): permite generar el PDF como una promesa para ser enviado por correo.

Accedemos a la librería mediante:

```
1 const { jsPDF } = window.jspdf;
```

Se utiliza el objeto global window.jspdf dónde está cargado el módulo jsPDF, a través de un script en el HTML:

```
<script  
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js">  
</script>
```

Para poner la información en formato tabla, utilizamos la extensión autoTable:

```
<script  
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-autotable/3.5.28/jspdf.plugin.autotable.min.js">  
</script>
```

Esto sería un ejemplo breve de generación de PDF usando jsPDF y autoTable:

```
1 const doc = new jsPDF();  
2 doc.text("Título del informe", 105, 25, { align: "center" });  
3 doc.autoTable({  
4   head: [["Campo", "Valor"]],  
5   body: [["Producto", "Herbicida X"], ["Cantidad", "2 L"]],  
6 });  
7 doc.save("informe.pdf");
```

Envío del Informe por Correo Electrónico

El sistema permite enviar por correo electrónico a la dirección introducida por el usuario el informe generado con jsPDF (explicado anteriormente). Este proceso se lleva a cabo mediante una interacción entre el frontend (usuario web) y el backend (servidor Express).

Generación del PDF

Cuando el usuario hace clic en el botón “Enviar informe por correo”, se ejecuta el siguiente flujo:

- Se valida que se haya introducido un correo electrónico válido y que se haya seleccionado un tratamiento.
- Se genera el informe en PDF utilizando jsPDF.
- El PDF se convierte a un objeto Blob y se adjunta a un objeto FormData, junto con el correo de destino.

```
1 // Convertir el documento en un objeto Blob con tipo PDF
2 const pdfBlob = new Blob([doc.output("blob")], { type: "application/pdf" });
```

Blob (Binary Large Object)

Es un objeto que representa datos binarios inmutables. Lo utilizamos para convertir el PDF generado en el navegador en un formato que pueda ser tratado como un archivo real. El Blob permite que el PDF generado con jsPDF pueda adjuntarse como archivo a un formulario.

FormData

Es una estructura que simula un formulario HTML con campos clave-valor. Permite enviar datos al servidor en el formato multipart/form-data, que es el tipo necesario para transmitir archivos. En el proyecto usamos FormData para incluir tanto el Blob del PDF como el correo electrónico del destinatario y enviarlos juntos al backend.

```
1 // Crear un objeto FormData para enviar el PDF y el correo electrónico
2 const formData = new FormData();
3 formData.append("correo", correo); // correo electrónico del destinatario
4 formData.append(
5   "informe",
6   pdfBlob,
7   `Informe_Tratamiento_${cultivo}_${producto}_${fecha}.pdf`
8 );
```

Envío del PDF desde el Servidor

Con una petición POST en la ruta /informes/enviar, el servidor Express hace lo siguiente:

```

1 // Enviar el formulario mediante una petición POST al backend
2 const res = await fetch("http://localhost:3000/informes/enviar", {
3   method: "POST",
4   body: formData,
5 });

```

- Uso de multer para recibir y guardar temporalmente el archivo PDF.

```
1 const upload = multer({ dest: "API/uploads/" });
```

- Configura un transportador con nodemailer usando las credenciales del archivo de configuración, dónde se guardan las variables de entorno:

```

1 email: {
2   USER: process.env.EMAIL_USER,
3   PASS: process.env.EMAIL_PASS,
4   HOST: process.env.EMAIL_HOST,
5   PORT: process.env.EMAIL_PORT,
6 },

```

- Envía un correo con el PDF adjunto al destinatario indicado, usando los parámetros:

```

1 await transporter.sendMail({
2   from: `Cuaderno de Campo <${email.USER}>`,
3   to: correo,
4   subject: "Informe de Tratamiento y Equipo",
5   text: "Adjunto encontrará el informe generado.",

```

- Una vez se haya realizado el envío exitosamente, elimina el archivo temporal del servidor evitando acumulación de datos innecesarios.

Multer

Es un middleware de Node.js que permite manejar formularios con archivos adjuntos (multipart/form-data). Importación en el proyecto:

```
1 const multer = require("multer");
```

Nodemailer

Es una librería que permite enviar correos electrónicos desde una aplicación Node.js. Se configura con credenciales SMTP (usuario, contraseña, host y puerto del servidor del correo) y se encarga de componer y enviar el mensaje. Importación en el proyecto:

```
1 const nodemailer = require("nodemailer");
```

Inserción del JSON de productos y usos en la base de datos

La lista de productos fitosanitarios autorizados y sus usos, la ofrece el Gobierno de España en un archivo JSON. Para actualizar la base de datos de nuestro sistema hemos desarrollado dos posibilidades, de forma manual o actualización automática.

Inserción Manual

El usuario debe tener descargado el archivo JSON con el listado de productos autorizados pudiendo seleccionarlo como se pide en la web. Una vez seleccionado, se limpia el archivo procesando únicamente los campos necesarios y estructurarlos en un formato apto para ser insertado en la base de datos.

```

1 jsonLimpio = {
2     Productos: jsonData.Productos.map((itemProducto) => ({
3         IdProducto: itemProducto.DATOSPRODUCTO.IdProducto,
4         Num_Registro: itemProducto.DATOSPRODUCTO.Num_Registro,
5         Nombre: itemProducto.DATOSPRODUCTO.Nombre,
6         Formulado: itemProducto.DATOSPRODUCTO.Formulado,
7         Titular: itemProducto.DATOSPRODUCTO.Titular,
8         Fabricante: itemProducto.DATOSPRODUCTO.Fabricante,
9         Fecha_Registro: itemProducto.DATOSPRODUCTO.Fecha_Registro,
10        Estado: itemProducto.DATOSPRODUCTO.Estado,
11        Fecha_Caducidad: itemProducto.DATOSPRODUCTO.Fecha_Caducidad,
12        Fecha_Cancelacion: itemProducto.DATOSPRODUCTO.Fecha_Cancelacion,
13        Fecha_LimiteVenta: itemProducto.DATOSPRODUCTO.Fecha_LimiteVenta,
14        Usos: itemProducto.USOS.map((itemUso) => ({
15            CodigoCultivo: itemUso.CodigoCultivo,
16            Cultivo: itemUso.Cultivo,
17            CodigoAgente: itemUso.CodigoAgente,
18            Agente: itemUso.Agente,
19            Dosis_Min: itemUso.Dosis_Min,
20            Dosis_Max: itemUso.Dosis_Max,
21            UnidadMedidaDosis: itemUso["Unidad Medida dosis"],
22            PlazoSeguridad: itemUso["Plazo Seguridad"],
23            VolumenCaldo: itemUso["Volumen Caldo"],
24            Aplicaciones: itemUso.Aplicaciones,
25            IntervaloAplicaciones: itemUso.IntervaloAplicaciones,
26            CondicionamientoEspecifico: itemUso.CondicionamientoEspecifico,
27            MetodoAplicacion: itemUso.MetodoAplicacion,
28            Volumen_Min: itemUso.Volumen_Min,
29            VolumenMax: itemUso.VolumenMax,
30            UnidadesVolumen: itemUso["Unidades Volumen"],
31        })),
32    })),
33};

```

La inserción comienza haciendo click en el botón “Actualizar JSON”. Mediante una petición POST a la ruta /subir-json se envía al backend el JSON limpio.

```

1 const response = await fetch("http://localhost:3000/subir-json", {
2   method: "POST",
3   headers: { "Content-Type": "application/json" },
4   body: JSON.stringify(jsonLimpio),
5 });

```

Para asegurar que si ocurre un error no se inserten datos parciales, el proceso se hace dentro de una transacción. Se insertan primero los productos y luego sus usos asociados, usando *INSERT (INTO “tabla”(“campos”)) ON DUPLICATE KEY* evitando duplicados y manteniendo actualizada la información existente.

Inserción Automática

El proceso manual está automatizado, el sistema se encarga de descargar, limpiar e insertar el archivo JSON en la base de datos. Como el archivo de los productos fitosanitarios no tiene un enlace de descarga directo, hemos utilizado la librería *puppeteer* que lanza un test simulando la interacción con la web, permitiendo la descarga automática del archivo. El resto del proceso es igual que en la forma manual.

Al hacer clic en “Actualización automática” se envía una petición GET a la ruta (/productos_y_usos/automatizar_json). Para evitar leer un archivo desactualizado, se elimina el contenido de la carpeta dónde se realizará la descarga:

```

1 // Eliminar archivos anteriores en la carpeta
2 const archivosExistentes = fs.readdirSync(downloadPath);
3 for (const archivo of archivosExistentes) {
4   fs.unlinkSync(path.join(downloadPath, archivo));
5 }

```

Iniciamos puppeteer:

```

1 const browser = await puppeteer.launch({ headless: false });
2 const page = await browser.newPage();

```

Como por defecto puppeteer no descarga archivos automáticamente debemos configurarlo. Creamos una sesión del protocolo *DevTools* (CDP) asociada a la página actual, permitiendo enviar comandos de bajo nivel directamente al navegador Chrome:

```
1 const client = await page.target().createCDPSession();
```

Después, enviamos el comando que nos permite las descargas y que guarde los archivos descargados en una carpeta específica:

```

1 await client.send("Page.setDownloadBehavior", {
2   behavior: "allow",
3   downloadPath,
4 });

```

Navegamos a la web y esperamos a que no haya más de 2 peticiones de red activas durante al menos 500 ms, asegurando que la página haya cargado completamente:

```

1 await page.goto("https://servicio.mapa.gob.es/regfiweb", {
2     waitUntil: "networkidle2",
3 });

```

Definimos las acciones necesarias para acceder al enlace de descarga con tiempos de espera intermedios para evitar fallos y cerramos el test:

```

1 await page.waitForSelector('#lnkResumenes a', { timeout: 15000 });
2 await page.click('#lnkResumenes a');
3 await new Promise(resolve => setTimeout(resolve, 1000));
4
5 await page.waitForSelector("#btnProductosAutorizadosJson", { timeout: 15000 });
6 await page.click("#btnProductosAutorizadosJson");
7
8 await new Promise(resolve => setTimeout(resolve, 10000));
9 await browser.close();

```

El resto del proceso es igual que en la inserción manual. Sin embargo, como el archivo JSON tiene un tamaño considerable se requiere un tiempo de respuesta muy largo y el navegador por seguridad cancela la petición porque la conexión HTTP queda abierta demasiado tiempo sin respuesta. La solución fue dar una respuesta anticipada evitando el error por respuesta vacía y lanzar el proceso en segundo plano.

Esto provoca que el frontend no sepa directamente cuándo termina todo el proceso, por ello implementamos un sistema de seguimiento de estado (*polling*).

```

1 let procesoEstado = "idle"; // Estado global del proceso

```

Ruta para consultar el estado:

```

1 router.get("/estado_proceso", (req, res) => {
2   res.json({ estado: procesoEstado });
3 });

```

Consulta periódica desde el navegador:

```

1 const intervalId = setInterval(async () => {
2   const estadoRes = await fetch("http://localhost:3000/productos_y_usos/estado_proceso");
3   const estadoData = await estadoRes.json();
4
5   if (estadoData.estado === "completado") {
6     clearInterval(intervalId);
7     alert("El proceso ha finalizado correctamente.");
8     // reactivar botón, actualizar interfaz
9   } else if (estadoData.estado === "fallido") {
10    clearInterval(intervalId);
11    alert("El proceso ha fallado.");
12  }
13 }, 5000); // Cada 5 segundos

```

Puppeteer

Es una librería de Node.js que permite controlar un navegador Chromium de forma automatizada. Simula la interacción humana con páginas web: navegar, hacer clics, etc.

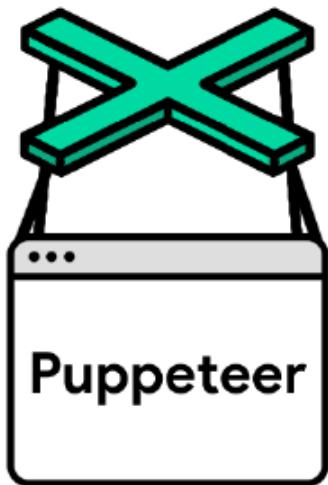
Solución a error posible

Puede ocurrir durante el proceso de descarga automática del archivo JSON. Aparecerá una alerta avisando del fallo y para consultarla habrá que dirigirse a la consola del servidor.

```
1 Error: Could not find Chrome (ver. 136.0.7103.92)
```

Significa que *Puppeteer* no encuentra el navegador Chrome que necesita para ejecutarse. Esto se puede solucionar instalando manualmente Chromium en la caché local con este comando:

```
1 npx puppeteer browsers install chrome
```



Repositorio en GitHub

Para desarrollar, mantener el código y trabajar en grupo hemos usado Github, el cual nos ha permitido organizarnos y trabajar de una forma segura y cómoda en grupo.

Link del repositorio: <https://github.com/DanielBS28/TFG-Cuaderno-de-campo>

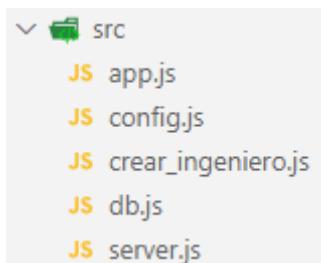
API

Una API (Interfaz de Programación de Aplicaciones) es un conjunto de reglas y definiciones permitiendo que diferentes programas o sistemas se comuniquen entre sí. Es una forma de acceder a funciones o datos de otras aplicaciones sin necesidad de conocer su funcionamiento interno.

Las herramientas que utilizamos en nuestra interfaz de programación de aplicaciones son: un entorno de ejecución para Javascript como Node.js y una API de tipo RESTful, es decir que hace peticiones de tipo HTTP como PUT, POST, GET, DELETE desde los endpoints configurados previamente.

Como módulos utilizamos: **express** para el framework, **cors** para la seguridad de los navegadores y las solicitudes HTTP que provienen de un origen distinto, **body-parser** para leer y procesar formularios o datos formato JSON, y **mysql2** para conectarnos a la base de datos.

La estructura que utilizamos se conforma en cuatro ficheros principales que están en el directorio “src” dentro del directorio “API”, los ficheros son app.js, config.js, server.js, db.js.



El archivo `crear_ingeniero.js` que aparece en la imagen no pertenece a la estructura, es un script ejecutable que nos permite insertar el ingeniero necesario para el funcionamiento de la aplicación en la base de datos, con el siguiente comando (desde el directorio API):

```
node .\src\crear_ingeniero.js
```

App.js

En este fichero se encuentran principalmente las rutas que luego son utilizadas por los endpoints que hacen las solicitudes HTTP (PUT, POST, GET, DELETE).

```

1 // Middleware
2 app.use(cors());
3 app.use(bodyParser.json({ limit: "80mb" })); // Aumentar el límite de tamaño del cuerpo a 80mb
4 app.use(bodyParser.urlencoded({ extended: true, limit: "80mb" }));
5
6 // Ruta para automatizar JSON de productos y usos
7 app.use("/productos_y_usos", productos_y_usos);
8
9 // Ruta para subir JSON de productos y usos
10 app.use("/subir-json", subirJsonRouter);
11
12 // Ruta para funcionalidades agricultores
13 app.use("/agricultores", agricultoresRouter);

```

En la parte del *middleware* inicializamos el **cors()**, el **body-parser** y entendemos datos de tipo JSON con un límite de 80 megabytes* de datos en una solicitud y que se pueda entender datos como formularios tipo JSON.

Las rutas como **app.use("/productos_y_usos", productos_y_usos);** generan una conexión a otros ficheros donde está la lógica de los endpoints que veremos más adelante, ademas **app** es la variable que representa el host y el puerto y lo llamamos desde el fichero config.js.

```

1 app: {
2     HOST: process.env.HOST || "localhost",
3     PORT: process.env.PORT || 3000,
4 },

```

En conclusión este fichero es la base de la API, ya que es el encargado de importar los módulos del framework como el de la base de datos, además de llamar a los ficheros donde están los endpoints y generar una ruta que se utilizará para la comunicación entre el frontend y el backend.

Por último genera una ruta principal y exporta la instancia de Express para que pueda ser utilizada en otros ficheros como server.js.

```
1 module.exports = app;
```

*Definimos este límite porque el archivo de los productos y usos es de gran tamaño, llegando a pesar en algunas versiones hasta 78 mb.

Config.js

Aquí requerimos a **dotenv** donde se cargan las variables de entorno desde un archivo llamado **.env** al objeto `process.env` lo que permite manejar credenciales, contraseñas, puertos, claves sensibles.

Luego exportamos un objeto llamado `app`, este se encarga de definir el host y el puerto llamados por defecto localhost y 3000 respectivamente. Después definimos el objeto `email` en donde tiene variables de credenciales de un correo como user, pass, host, port.

```
1  require("dotenv").config();
2
3  module.exports = {
4    app: {
5      HOST: process.env.HOST || "localhost",
6      PORT: process.env.PORT || 3000,
7    },
8    email: {
9      USER: process.env.EMAIL_USER,
10     PASS: process.env.EMAIL_PASS,
11     HOST: process.env.EMAIL_HOST,
12     PORT: process.env.EMAIL_PORT,
13   },
14 };
```

Esto nos ayuda a centralizar nuestras variables de entorno y permite cambiar configuraciones sin modificar el código fuente, además de que se puede importar fácilmente a otros módulos.

Db.js

En este fichero encontramos la conexión a la base de datos con **createPool** en lugar de **createConnection**, ya que permite la reutilización de conexiones, mejor manejo de concurrencia, espera en la cola y controlar el límite de conexiones.

Dentro de este definimos el host, el usuario de la base de datos y su contraseña con el nombre de la base de datos, además confirma la espera de la conexión y pone un límite de conexiones que son 10.

Por último exporta la instancia de Express para que pueda ser utilizado en otros ficheros.

```
1 const mysql = require("mysql2");
2
3 // Conexión a MySQL con createPool (mejor manejo de conexiones)
4 const db = mysql.createPool({
5   host: "localhost",
6   user: "root",
7   password: "root", // Cambia esto si tienes otra contraseña
8   database: "cuaderno_de_campo",
9   waitForConnections: true,
10  connectionLimit: 10,
11  queueLimit: 0
12 });
13
14 // Exportar la conexión para usarla en otros módulos
15 module.exports = db;
```

Server.js

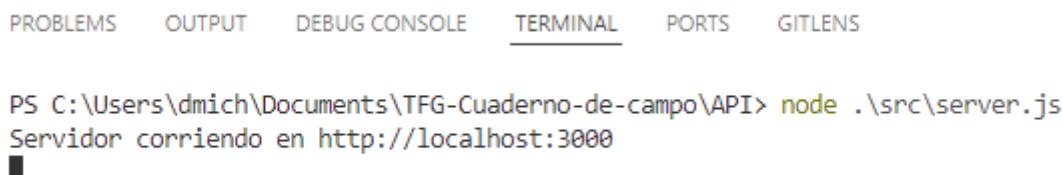
Por último, tenemos el fichero con el que iniciamos el servidor desde la consola. Este archivo se encarga de llamar a app.js y a config.js.

Recordemos que el primer fichero (app.js) contiene todas las rutas de los endpoints y es donde se importan todos los módulos necesarios, como express y mysql2, además de establecer el puerto del servidor.

El segundo fichero (config.js) se encarga de configurar parámetros como el puerto y el correo electrónico.

```
1 const app = require("./app");
2 const { app: configApp } = require("./config");
3
4 app.listen(configApp.PORT, () => {
5   console.log(`Servidor corriendo en http://:${configApp.HOST}:${configApp.PORT}`);
6 })
```

Para iniciar el servidor es tan simple como abrir una consola, dirigirte a la ruta en API y escribir “node /src/server.js”. Si todo salió bien tiene que mostrar un mensaje el cual es: “Servidor corriendo en http://HOST: PORT”.



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\dmich\Documents\TFG-Cuaderno-de-campo\API> node .\src\server.js
Servidor corriendo en http://localhost:3000
```

Existe un comando muy útil para que el servidor se refresque automáticamente cuando se guarden cambios del código, este es añadir –watch antes de la ruta.

```
PS C:\Users\dmich\Documents\TFG-Cuaderno-de-campo\API> node --watch .\src\server.js
Servidor corriendo en http://localhost:3000
```

Podemos comprobar que al guardar un cambio aparece tal que así:

```
Restarting '.\src\server.js'
Servidor corriendo en http://localhost:3000
```

Rutas y Endpoints

Las rutas son las URLs específicas que asignamos a un método para que el servidor las reconozca y pueda responder. Por ejemplo, en `app.get("/usuario")`, la ruta es `/usuario` y está asociada al método GET. Entendido esto, podemos decir que un endpoint es la combinación de una ruta con un método HTTP que solicita o envía algún tipo de dato.



Nuestras rutas se encuentran en el directorio API/Routes, donde están definidos los endpoints responsables de gestionar los recursos del servidor y mantener su lógica mediante los métodos GET, POST, PUT y DELETE. Además, estos endpoints permiten la comunicación entre el frontend y el backend.

```

1 // Obtener todos los agricultores
2 router.get("/todos", async (req, res) => {
3     try {
4         const [rows] = await db.promise().query(`
5             SELECT u.DNI, u.Nombre, u.Apellido1, u.Apellido2
6             FROM Usuario u
7             INNER JOIN Agricultor a ON u.DNI = a.Usuario_DNI
8         `);
9
10        res.json(rows);
11    } catch (error) {
12        console.error("Error al obtener todos los agricultores:", error);
13        res.status(500).json({ error: "Error del servidor al obtener agricultores." });
14    }
15});
```

En este ejemplo del fichero **agricultores.js**, nos muestra que para el método GET de agricultores tenemos la ruta **/todos** el cual hace una solicitud a la base de datos mediante una query que llama a todas las filas que coincidan ambas tablas, usuarios y agricultores. Si se produce un error está como status el número **500** el cual muestra un mensaje de error en consola.

Por otro lado, este endpoint hace de puente entre ciertas partes del frontend y el backend.

En el fichero **baja_agricultores.js** hay un evento utilizado en el backend en donde se carga el contenido de un HTML mediante un objeto global window, el cual tiene una función asíncrona para poder utilizar await dentro del código. Esta concurrencia se hace ya que funciona mejor cuando se trabaja con peticiones HTTP con fetch, luego hacemos una petición con fetch en donde solicitamos al servidor que cargue a todos los agricultores al iniciar con la ruta **/agricultores/todos**. Si nos damos cuenta esta es la ruta que justo tiene el endpoint que carga a los agricultores de la base de datos.

```
1 // ### EVENTOS ###
2 // Cargar todos los agricultores al iniciar
3 window.addEventListener("DOMContentLoaded", async () => {
4     try {
5         const res = await fetch("http://localhost:3000/agricultores/todos");
6         const data = await res.json();
7         listaAgricultores = data;
8         actualizarOpcionesSelect(data);
9     } catch (error) {
10         console.error("Error al cargar agricultores:", error);
11     }
12});
```

Hosting

A finales de Mayo de manera **extraordinaria** realizamos una **demo** de nuestro proyecto en versión online.

Elegimos la plataforma **Railway** para desplegar nuestra aplicación porque su plan gratuito permite alojar tanto archivos estáticos (HTML, CSS, JS) como la API y la base de datos. Además ofrece integración directa con Github, por lo que el despliegue se realiza automáticamente al hacer push. Railway en nuestro caso detecta que es un proyecto de Node.js y configura el entorno en consecuencia.

Conexión con la base de datos

En Railway creamos una instancia de base de datos MySQL, que proporciona una URL de conexión segura. Usamos esta conexión para inicializar nuestras tablas ejecutando el script correspondiente (nosotros lo hicimos desde MySQL Workbench).

Servir Frontend desde Express

Para alojar el frontend dentro del mismo servidor que la API, configuramos **Express** para servir archivos estáticos desde las carpetas HTML, CSS, Javascript y otras. Las rutas del archivo **app.js** están definidas para que el navegador pueda acceder correctamente al contenido desde cualquier entorno.

```
1 app.use("/", express.static(path.join(__dirname, "..", "HTML_publico")));
2 app.use("/css", express.static(path.join(__dirname, "..", "CSS")));
3 app.use("/js", express.static(path.join(__dirname, "..", "Javascript")));
4 app.use("/fotos", express.static(path.join(__dirname, "..", "Fotos")));
5 app.use("/data", express.static(path.join(__dirname, "..", "Data")));
```

Seguridad en el acceso al HTML privado

Además de servir archivos públicos, implementamos una capa de seguridad para proteger las páginas privadas accesibles sólo a usuarios autenticados. Creamos una carpeta específica (HTML_privado) con contenido sensible destinado únicamente a los ingenieros que han iniciado sesión.

Antes de servir cualquier archivo dentro de esta carpeta, añadimos un **middleware de control de sesión** que comprueba si el usuario ha iniciado sesión correctamente. Si no es así, se redirige automáticamente al login.

```

1  const htmlPrivadoDir = path.join(__dirname, "..", "HTML_privado");
2
3  fs.readdirSync(htmlPrivadoDir).forEach((archivo) => {
4      if (archivo.endsWith(".html")) {
5          const ruta = "/" + archivo;
6
7          app.get(ruta, (req, res) => {
8              if (!req.session?.ingeniero) {
9                  return res.redirect("/login.html");
10             }
11             res.sendFile(path.join(htmlPrivadoDir, archivo));
12         });
13     }
14 });

```

Sistema de login y control de sesiones

El sistema de autenticación permite que solo los ingenieros registrados puedan acceder a la zona privada de la aplicación. Este proceso se compone de dos partes: **validación de credenciales** y **gestión de sesión activa**.

Validación de credenciales

Cuando el usuario envía el formulario de login, la aplicación realiza una consulta a la base de datos para comprobar si existe un ingeniero con el **DNI** y la **contraseña** introducidos. Las contraseñas están cifradas con **bcrypt**.

Gestión de sesiones

Utilizamos la librería **express-session** para mantener la sesión activa tras el login. Almacenamos el objeto **usuario** en **req.session**, lo que nos permite verificar su existencia en cada ruta protegida.

```

1  req.session.ingeniero = {
2      dni: usuario.DNI,
3      nombre: usuario.Nombre,
4  };

```

Por defecto, **express-session** almacena las sesiones en memoria, lo cual no es adecuado en producción, ya que las sesiones se pierden si se reinicia el servidor. Para solucionarlo, usamos la librería **express-mysql-session**, que permite guardar las sesiones directamente en nuestra base de datos MySQL alojada en Railway.

Primero se configura un **almacenamiento externo** para las sesiones, creando una instancia de **MySQLStore** con los datos de conexión de la base de datos:

```

1 const session = require("express-session");
2 const MySQLStore = require("express-mysql-session")(session);
3
4 const sessionStore = new MySQLStore({
5   host: process.env.MYSQL_HOST,
6   user: process.env.MYSQL_USER,
7   password: process.env.MYSQL_PASSWORD,
8   database: process.env.MYSQL_DATABASE
9 });

```

Luego, se inicializa *express-session* utilizando ese almacenamiento:

```

1 app.use(
2   session({
3     key: "cuaderno_sesion_id",
4     secret: "super_secreto_seguro",
5     store: sessionStore,
6     resave: false,
7     saveUninitialized: false,
8     cookie: {
9       secure: process.env.NODE_ENV === "production", // activa solo con HTTPS
10      sameSite: "lax", // para evitar bloqueos
11    },
12  })
13 );

```

Una vez iniciada la sesión, el usuario puede navegar por las páginas privadas sin volver a autenticarse, siempre que la sesión siga activa. Para permitir el cierre de sesión manual, definimos una ruta */logout* que destruye los datos de sesión y redirige al usuario al *index*:

```

1 router.get("/logout", (req, res) => {
2   req.session.destroy(() => {
3     res.redirect("/index.html");
4   });
5 });

```

Detección del entorno de ejecución

Para que el frontend utilice automáticamente la URL correcta de la API dependiendo de si estamos trabajando en local o en producción (Railway), añadimos una lógica condicional en el código JavaScript del navegador:

```

1  const API_URL =
2    window.location.hostname === "localhost"
3      ? "http://localhost:3000"
4      : "https://cuadernodecampo.up.railway.app";

```

Con esta estructura, todas las peticiones que realiza el frontend a la API se redirigen automáticamente al backend local durante el desarrollo y al dominio de Railway una vez desplegada. Esto nos permite mantener un único código para ambos entornos, sin necesidad de cambiar manualmente URLs al hacer el despliegue.

Ejemplo de la petición `/login`:

```

1  try {
2    const res = await fetch(` ${API_URL}/login`, {
3      method: "POST",
4      credentials: "include",
5      headers: {
6        "Content-Type": "application/x-www-form-urlencoded",
7      },
8      body: new URLSearchParams({ DNI, contrasena })
9    });
10
11   if (res.redirected) {
12     window.location.href = res.url;
13   } else {
14     const errorText = await res.text();
15     alert(errorText);
16   }
17 } catch (err) {
18   alert("Error de conexión con el servidor.");
19   console.error(err);
20 }

```

Repositorio en GitHub

A continuación se muestra el enlace donde se encuentra el repositorio de la versión online de nuestra aplicación.

Link del repositorio: https://github.com/cuadernodecampo/cuadernodecampo_online

Diagrama de flujo

A continuación se muestra el diagrama o proceso que se debe de seguir para que la aplicación funcione correctamente y se pueda completar el flujo completo de un tratamiento, mediante un ejemplo.

Carga del JSON

La primera vez que usemos la aplicación se deberá tener el JSON de productos cargado en la base de datos, no es necesario que sea lo primero, pero sí que será necesario para establecer cultivos y por consiguiente realizar un tratamiento, es recomendable que se haga como primera acción en la APP y también que lo actualicemos todas las semanas.

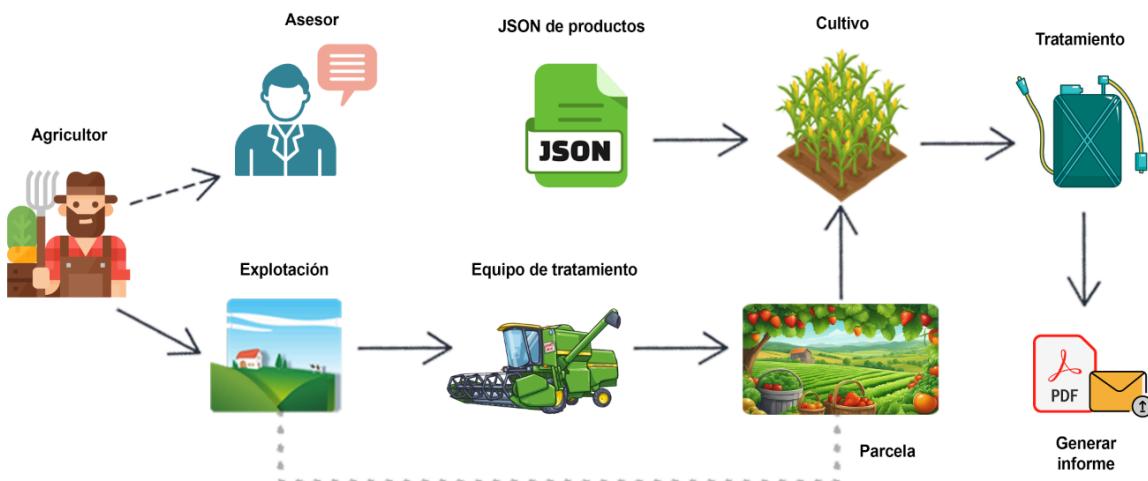
Flujo desde el principio

Para iniciar el flujo tenemos que crear un agricultor, una vez creado tenemos las opciones de editar sus datos o darlo de baja. Posteriormente de manera opcional se puedan dar de alta asesores para asignarlos a los agricultores que deseen.

El siguiente paso es crear una explotación que le pertenece al agricultor que hemos creado anteriormente, de las explotaciones podemos editar sus datos y darlas de baja.

Una vez que tengamos la explotación creada, lo recomendable es crear y asignar un equipo de tratamiento. No es necesario crearlo después de la explotación ya que se podría crear las parcelas y los cultivos, pero sí que será necesario para realizar un tratamiento.

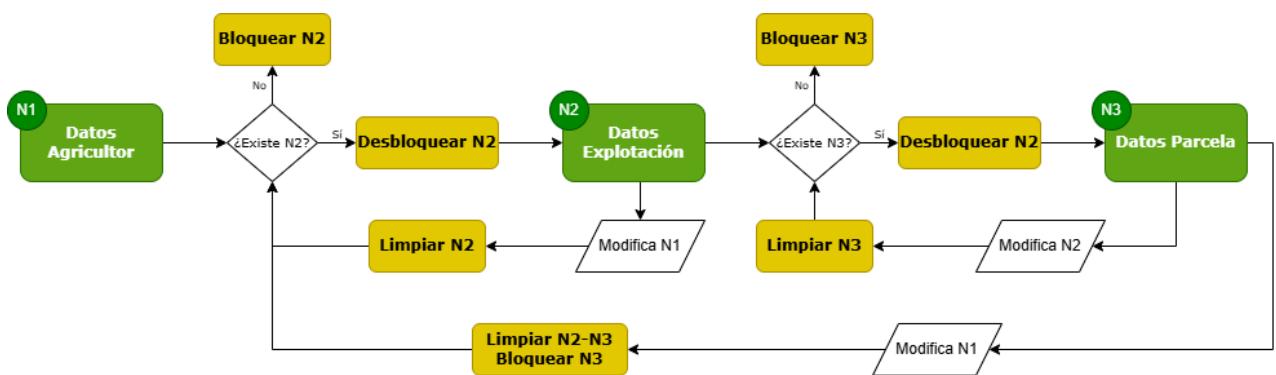
Creada la explotación como se mencionó anteriormente podemos crear una parcela y los datos deben de existir en el SIGPAC para que sea una parcela válida ya que debe de existir. Posteriormente, tenemos que crear uno o varios cultivos en nuestras parcelas. Los cultivos están asociados a los recintos que se han cargado al crear una parcela. Una vez tengamos ya cultivado algo en nuestra parcela se podrá realizar el tratamiento. Cuando ya tengamos un tratamiento registrado ya podemos generar un informe en PDF que se podrá descargar o enviar por correo electrónico.



Pruebas realizadas

Control de bugs general

Las funcionalidades de la web están estructuradas con una nivelación, es decir que unos campos deben ser completados para desbloquear otros de nivel inferior. De tal forma que si modificamos uno y hay campos de nivel inferior rellenos, se limpian o bloquean los niveles inferiores respectivamente.



Validación de las dosis de un tratamiento

En alta de tratamiento ofrecemos una validación informativa sobre: la dosis del producto fitosanitario que se quiere aplicar (según el rango mínimo y máximo)¹, la superficie a tratar (no puede ser mayor a la superficie del cultivo)², además la fecha del tratamiento no debe superar al día actual. Cabe recalcar que los productos que aparecen en el desplegable han sido filtrados dependiendo del cultivo y plaga seleccionados previamente.

1. El rango de la dosis está determinado en el archivo JSON para 1 hectárea (ha), por lo tanto hacemos el cálculo según la superficie declarada: Superficie * Dosis.
2. La superficie del cultivo es la suma del área de todos los recintos de una parcela con el mismo tipo de cultivo, también está expresada en hectáreas (ha).

Importante, la validación sólo está implementada en unidades de medida específicas:

$$\begin{array}{c}
 (\text{Kg}/\text{ha}) \ (\text{g}/\text{ha}) \ (\text{g}/\text{m}^2) \ (\text{g}/10\text{m}^2) \ (\text{l}/\text{ha}) \\
 (\text{ml}/\text{ha}) \ (\text{cc}/\text{ha}) \ (\text{ml}/\text{m}^2) \ (1/10\text{m}^2) \ (\text{ml}/100\text{m}^2) \ (\text{cc}/100\text{m}^2)
 \end{array}$$

Esto se debe a que son las únicas unidades convertibles a Kg/ha o l/ha solamente conociendo el área de la superficie que se quiere aplicar y la dosis para 1 hectárea (ha). Otras unidades como Kg/Hl se necesita saber el volumen aplicado por ha, pero no todos los valores de ese campo pueden ser procesados.

Comprobar si existe una parcela en SIGPAC

A la hora de dar de alta una parcela, para una mayor fiabilidad solo se permiten crear parcelas existentes en el registro del SIGPAC.

Esto se logra mediante una llamada a la API del SIGPAC:

```
const url = `https://sigpac-hubcloud.es/servicioconsultassigpac/query/refcatparcela/
${provincia}/${municipio}/${agregado}/${zona}/${poligono}/${parcela}.json
`;
```

Si la parcela introducida existe, la llamada nos devuelve un mensaje formato json:

```
[  
  {  
    "provincia": 28,  
    "municipio": 111,  
    "agregado": 0,  
    "zona": 0,  
    "poligono": 4,  
    "parcela": 9,  
    "referencia_cat": "28111A004000090000ME"  
  }  
]
```

En caso contrario la respuesta será un array vacío, es decir que la parcela introducida no existe en el SIGPAC.

Inyecciones SQL

Hemos utilizado la librería *mysql2* para evitar inyecciones SQL.

Mysql2

Es un cliente de Node.js que permite conectarse y comunicarse con bases de datos MySQL o MariaDB.

Con el uso de sentencias preparadas que separan los datos de la lógica SQL, evitamos ataques mediante inserción de comandos maliciosos en las consultas.

```
1 const query = 'SELECT * FROM usuarios WHERE nombre = ?';
2 connection.execute(query, [nombre], (err, results) => { ... });
```

Producto final

Objetivos conseguidos

Objetivos mínimos

En cuanto a los objetivos mínimos hemos desarrollado las funcionalidades correspondientes para dar de alta/baja a agricultores, asesores, explotaciones, parcelas, equipos de tratamiento, cultivos y tratamientos. Se permite cambiar ciertos datos de un registro ya creado en la base de datos mediante la interfaz web, excepto en tratamientos. También hemos desarrollado una funcionalidad para parsear y actualizar el archivo JSON de productos de forma manual. Respecto a los tratamientos se puede filtrar según los tipos de cultivo que haya en una parcela, las plagas que afectan a ese tipo de cultivo y los productos que afecten a la plaga y al tipo de cultivo seleccionado. Una vez realizado el tratamiento se podrá generar un informe a modo de resumen que se podrá descargar en PDF o enviarlo mediante email.

Objetivos extraordinarios

En cuanto a los extraordinarios, tras conseguir a finales de mayo una versión online hemos podido crear la aplicación multiusuario para que varias personas puedan interactuar con ella además la plataforma donde la hemos alojado utiliza el protocolo HTTPS para asegurar la protección de los datos en internet, se ha conseguido adaptar la aplicación para dispositivos con diferentes tamaños de pantalla mediante media queries en CSS, se pueden visualizar las parcelas y recintos en el mapa del SIGPAC, también en el HTML, CSS y JavaScript se ha diseñado de forma que la aplicación sea fácil de utilizar, además el archivo JSON en la versión local se puede actualizar de forma automática. El único objetivo que no hemos podido conseguir es programar la aplicación para agricultores.

Objetivos extraordinarios imprevistos

Hemos incorporado a nuestra aplicación tanto la API pública del SIGPAC como nuestra propia API privada y nos ha permitido conseguir que nuestra aplicación sea más escalable, segura y confiable.

Mejoras a realizar

Derivado del anterior objetivo está el hacer la interfaz programada para los agricultores. Esta mejora permitiría acceder a ciertas funcionalidades informativas no desarrolladas, como ver parcelas, recintos y sus especificaciones (tipo de cultivo, tratamientos realizados). La aplicación está enfocada para que un ingeniero (si es la versión local), o varios ingenieros (si es la versión online) controlen las explotaciones de todos sus agricultores. Existe una excepción si “el agricultor es su propio ingeniero”, es decir que él mismo controla su explotación, parcelas, recintos, cultivos y tratamientos.

Conclusiones

Opinión general

Cuando comenzamos con el proyecto teníamos muchas dudas y no veíamos un futuro certero sobre este trabajo, porque todas nuestras propuestas fueron rechazadas y aceptamos una idea interesante pero que al desconocer el sector nos imponía dificultad.

A medida que íbamos avanzando, la estructura del proyecto estaba más clara y con ello esa dificultad que sentíamos al principio iba disminuyendo. También aparecieron problemas pero afortunadamente además de resolverlos descubrimos nuevos recursos que nos potenciaron la eficiencia de la aplicación web.

Tras finalizar el proyecto de fin de grado nuestra opinión con respecto al principio cambió radicalmente. Darnos cuenta de que nuestro esfuerzo estaba dando sus frutos nos fue motivando más y la dificultad se volvía cuesta abajo.

Nuestra conclusión es que el trabajo en equipo y la buena organización nos ha permitido solucionar los problemas que han ido apareciendo eficazmente, y hemos logrado alcanzar los objetivos mínimos y una parte importante de los extraordinarios.

Elementos aprendidos

El proyecto nos ha servido para reforzar y poner en práctica todos los conocimientos aprendidos durante el ciclo. Aunque algunas de las tecnologías que hemos empleado han requerido de una formación autodidacta, como Node.js y express que son esenciales para la API y varias librerías como jspdf, nodemailer y puppeteer.

Aparte de los elementos técnicos hemos aprendido a trabajar y organizarse en equipo para desarrollar un proyecto común.

Dificultades y fallos

El primer fallo con el que nos topamos fue no implementar una selección directa de los campos en todas las funcionalidades, ya que por ejemplo al principio obligamos al usuario a tener que escribir el DNI del agricultor a mano. La solución fue incorporar una lista seleccionable que además puede filtrarse. Esto agiliza bastante la realización de los formularios.

Después, cuando investigamos sobre cómo procesar los valores de las dosis y sus unidades de medida, nos encontramos con que había 62 unidades de medida diferentes y muchas de ellas tenían valores de la dosis imposibles de procesar para hacer operaciones.

El único remedio que encontramos fue comprobar únicamente las unidades con las que podíamos hacer cálculos.

Por último apareció el problema más significativo en el que tuvimos que rediseñar la base de datos y varias funcionalidades. Esto se debió al desconocimiento de cómo se subdividen las explotaciones, al principio sólo teníamos que una explotación se divide en parcelas y cada una tenía únicamente un tipo de cultivo, sin embargo la realidad es que una explotación se divide en parcelas que a su vez se dividen en recintos, dónde cada recinto tiene un cultivo.

Agradecimientos

Tras la finalización del proyecto, queríamos expresar nuestro agradecimiento al profesorado que nos ha acompañado a lo largo de estos dos años por su dedicación y esfuerzo.

Aparte de enseñarnos los contenidos de las asignaturas, también nos habéis transmitido valores como el trabajo en equipo, la constancia y la importancia de dar siempre lo mejor de nosotros mismos.

Bibliografía

A continuación se muestran los enlaces de los materiales que hemos utilizado para resolver nuestras dudas o aprender nuevas tecnologías empleadas en nuestro proyecto.

jsPDF y Autotable

Tutorial práctico:

https://www.youtube.com/watch?v=1nEtQ0_spBU

Documentación oficial:

<https://github.com/parallax/jsPDF>

<https://github.com/simonbengtsson/jsPDF-AutoTable>

Nodemailer y multer

Tutorial práctico:

<https://www.youtube.com/watch?v=pzBO2pVs7GY>

Documentación oficial:

<https://nodemailer.com>

<https://github.com/expressjs/multer>

Puppeteer

Tutorial práctico:

https://www.youtube.com/watch?v=gBnrdedhuU4&ab_channel=FaztCode

Documentación oficial:

<https://pptr.dev/>

API

Curso de API con NodeJS y Express:

https://www.youtube.com/watch?v=F5oOq-FWUI4&list=PLnfMiP0v59hAUA6QJNKBwKJyq5_gFkCYL

Curso de NodeJS:

https://www.youtube.com/watch?v=yB4n_K7dZV8&list=PLUofhDlq_38qm2oPOV-IRTTEKyrVBaU7

JavaScript

Cursos de JavaScript avanzado:

<https://www.youtube.com/watch?v=2SetvwBV-SU&list=PLVq-jlkSeTUZ6QgYYO3MwG9EMqC-KoLXA>

<https://www.freecodecamp.org/learn/javascript-algorithms-and-data-structures-v8/>

UML

Video informativo de UML de casos de uso:

<https://youtu.be/fJa3cshrFWs?si=JlrvycrnHc7Hp3H2>

Curso 2024/2025



¡Descarga aquí los archivos del TFG!

