GovConnect

Gestor de Incidencias Urbanas

Tecnologías Web

Curso 2025/2026

Autores:

Alfredo Iniesta García

Javier Maestre Cerdeño

Carlos Mayorga Santiago

Helena Moncada Ocaña

Índice

1. Introducción

GovConnect es una aplicación web que tiene como objetivo principal servir como canal de comunicación entre la ciudadanía y la administración pública de una ciudad. Esta plataforma facilitará la gestión de incidencias urbanas, permitiendo colaborar en el reporte de problemas de distinta naturaleza detectados en la vía pública de manera rápida, sencilla y efectiva.

La plataforma permitirá entre otras cosas, registrar incidencias relacionadas con limpieza, infraestructuras, iluminación… así como seguridad ciudadana. Para ello, se proporcionan páginas clave desde las que se puede adjuntar información detallada y ubicación geográfica concreta mediante un mapa interactivo. El ciudadano estará informado en todo momento del seguimiento del estado de cada incidencia, lo que aporta una gran transparencia y comunicación.

En cuanto al desarrollo del proyecto, hemos usado una arquitectura clásica cliente-servidor, con el soporte de varios frameworks que más adelante detallaremos. Gracias al patrón Modelo-Vista-Controlador hemos mantenido una estructura muy organizada, y lo que es más importante, escalable y mantenible.

Con esta propuesta se pretende demostrar cómo las tecnologías web actuales derivan en aplicaciones prácticas y funcionales que integran frontend, backend, bases de datos y sistemas de autenticación (con la gestión de diferentes tipos de usuarios) que sería perfectamente válida en una aplicación real orientada al mercado.

2. Tecnologías empleadas

Para desarrollar *GovConnect* se han utilizado distintas tecnologías y herramientas orientadas en su mayoría a la creación de aplicaciones web modernas, tanto en el lado del servidor como en el lado del cliente. La fusión de todas estas nos ha permitido construir una plataforma dinámica, responsive y útil.

Backend

Debido a que es la tecnología principal que ha sido el foco en las clases de teoría, hemos empleado un framework basado en PHP llamado Laravel. Este permite desarrollar aplicaciones usando el patrón arquitectónico MVC que nos permite separar las distintas funcionalidades, así como las vistas, el acceso a datos o la lógica. La propia estructura de carpetas que genera hace el desarrollo más sencillo y orientativo, creando carpetas como app, config, database, resources…

Frontend

A la hora de diseñar la interfaz, no se requería ningún framework de CSS concreto. Por eso, tomamos la decisión de usar Tailwind CSS, basado en utilidades que permite construir interfaces modernas así como responsive. El resultado fue un diseño adaptable a cualquier dispositivo, así como componentes reutilizables y una gran consistencia estética, cosa que nos

parecía un requerimiento básico, ya que a veces la consistencia en trabajos cooperativos puede perderse. Todo esto es además gracias al gestor de plantillas Blade, proporcionada por Laravel.

Base de Datos

La persistencia de datos de nuestro sistema se ha gestionado mediante una plataforma de base de datos SQL llamada TiDB Cloud, la cual es distribuida y compatible con MySQL. Esto nos ha permitido trabajar abiertamente y de manera cooperativa, lo que nos ha facilitado el almacenamiento y gestión de forma eficiente de toda la información.

La integración con Laravel se realiza mediante Eloquent ORM (Object Relational Mapping), lo que permite ejecutar consultas y operaciones de tipo CRUD de manera sencilla sin necesidad de tener que escribir consultas SQL manualmente. En esta Base de Datos almacenamos principalmente información sobre los usuarios registrados, incidencias urbanas reportadas con sus estados…

La principal razón por la que hemos usado un Cloud es porque nos aporta ventajas como acceso remoto, alta disponibilidad y compatibilidad con las herramientas que estamos usando. Además al estar familiarizados con consultas SQL su compatibilidad nos resultó muy ventajosa.

Interactividad

Para las partes interactivas y funcionales en el lado del cliente, hemos usado JavaScript con el objetivo de que el usuario tenga la mejor experiencia a la hora de hacer ciertas consultas sin necesidad de recargar la página. Se ha usado por ejemplo para funcionalidades relacionadas con la interacción dinámica con los mapas en los que seleccionamos ubicaciones, permitiendo colocar y arrastrar el marcador, o vista previa de la imagen al añadirla al formulario de reporte de incidencia. Para ello hemos usado además la librería Leaflet, que permite selección interactiva. Una vez creadas las incidencias, si el usuario es administrador puede visualizarlas en su panel y poder cambiar el estado de las incidencias y a su vez eliminarlas una vez estén completadas.

Geolocalización y Mapas

Uno de los requisitos más importantes y visuales a la hora de hablar de las incidencias de una ciudad es ubicarlas en el espacio-tiempo, decidimos que un listado no era suficiente y que lo más visual era que el propio usuario pudiese ubicarlas en el mapa de su ciudad. Por ello, integramos mapas de OpenStreetMap que nos ofrece mapas libres y gratuitos.

Herramientas de Desarrollo

Además de todas las tecnologías mencionadas, hemos usado Visual Studio code como entorno de desarrollo principal y editor de código. Gracias a la infinidad de extensiones que posee, nos ha facilitado el trabajo al ser compatible con todas las tecnologías. Por otro lado, composer ha sido el principal gestor de dependencias PHP, puesto que instala y actualiza

todas las librerías y estructura del proyecto necesaria para el ecosistema Laravel. Por último, para el control de versiones hemos usado GitHub, lo que facilita la colaboración y el seguimiento detallado.

Además, para el despliegue tras haber probado varias alternativas, elegimos Railway, al ser una herramienta con un mes de prueba gratuito y compatibilidad con PHP. Se explicará más a detallado en el apartado dedicado a ello.

3. Análisis de Requisitos

3.1. Requisitos de Interfaz

Para la presentación y diseño del sitio web, hemos seguido por un criterios modernos de usabilidad y accesibilidad, para garantizar una experiencia coherente en cualquier tipo de dispositivo y los requisitos que se solicitan en los requisitos de la práctica. Además, se ha priorizado en todo momento un enfoque lo más centrado en la experiencia del usuario posible, priorizando la facilidad de uso. Para ello hemos usado componentes como tarjetas y bloques visuales que mejoran la legibilidad, navegación intuitiva o formularios organizados de forma clara y estructurada, lo que incita al ciudadano a querer compartir información.

Diseño responsive

Uno de los requisitos más importantes era desarrollar el sistema bajo un enfoque responsive, el cual permite su correcta visualización sin importar el tamaño de pantalla, es decir, ya sea en un ordenador, tablet, móvil…

Para llevarlo a cabo, hemos usado Tailwind CSS, que facilita la creación de diseños flexibles mediante un sistemas de clases capaces de adaptar la disposición de cualquier elemento según el tamaño del dispositivo. Esto combinado con primitivas del estilo grid o flexbox.

Cumplimiento de estándares W3C (adaptabilidad)

Para desarrollar la interfaz seguimos los estándares web recomendados por el World Wide Web Consortium, utilizando las últimas versiones de HTML y CSS (HTML5 y CSS3). Garantizamos así mayor mantenimiento de código y uso correcto de la estructura, así como compatibilidad con diferentes navegadores.

Para ello, usamos varias de las herramientas que se nos proporcionan en el guión de la práctica (https://www.w3.org/WAI/test-evaluate/tools/list/). Cada una de ellas nos indicaron un porcentaje entre un 75% y un 85%, indicando que nuestra página cumple los estándares de adaptabilidad.

Estructura de la interfaz

Para garantizar una gran coherencia visual, todas las páginas de GovConnect comparten ciertos elementos comunes. La estructura se compone de:

- Cabecera para navegación principal
- Pie de página con información del proyecto, contacto y enlace a esta misma memoria.
- Menú lateral izquierdo, donde podemos navegar por todas las páginas del sistema.
- Menú lateral derecho, que tiene una función de panel informativo, donde se muestra información resumida del sistema.

3.2. Requisitos funcionales

Este tipo de requisitos son los encargados de describir las funcionalidades concretas que ofrece *GovConnect* y todas las acciones que los usuarios, sin importar que tipo, pueden realizar dentro de la plataforma.

El sistema es un ejemplo completo de un gestor de incidencias urbanas, donde cualquier ciudadano podrá reportar un problema que le surja, siendo resuelto por los gestores o técnicos, quienes más tarde se encargará de comunicar que el problema fue solucionado.

1. Registro e inicio de sesión

El sistema deberá admitir nuevos usuarios mediante formularios HTML estructurados en los que solicitan los datos básicos de la persona.

Una vez registrada, la persona podrá iniciar sesión y mantener una sesión activa en el sitio web hasta que decida cerrarla.

En el propio registro de usuario puedes seleccionar si eres administrador.

2. Gestión de roles y permisos

El sistema es capaz de diferenciar entre los distintos tipos de usuario. Cada uno de estos tendrá unos permisos y movimientos específicos según el rol. Por ello se definen tres tipos de usuarios:

- Usuario no identificado, quien accederá al sitio web para visualizar información pública o abrir una incidencia.
- Usuario registrado, quien contará con su propio área personal.
- Administrador, capaces de gestionar todas las incidencias del sistema.

3. Creación de incidencias

Se crea un formulario específico para que cualquier usuario pueda registrar una incidencia urbana mediante un formulario específico con datos obligatorios como la categoría, la descripción, ubicación y fotografía.

4. Validación de datos

El sistema tiene una estructura robusta para asegurar la calidad de los datos introducidos. Es por eso que incorpora validaciones como campos obligatorios, formatos de correo electrónico y contraseña, validación de imágenes…

5. Consulta de incidencia

Una vez creada la incidencia, cualquier usuario podrá consultar la incidencia y acceder a su detalle, observando información como el Estado (Pendiente, En proceso, Solucionado), la descripción o ubicación, entre otros.

6. Gestión de incidencias (administrador)

Aquellos usuarios que hayan iniciado sesión como administradores, podrán gestionar cualquier incidencia del sistema, pudiendo cambiar el estado, supervisar los reportes o eliminar alguna incidencia. También podrán visualizar los mensajes que se han enviado con el formulario de contacto y eliminarlos si no resultan útiles.

Si el usuario no es administrador no podrá visualizar el panel con permisos especiales que gestiona las incidencias.

7. Interfaz y experiencia de usuario

El sistema debe diseñarse bajo un enfoque que priorice la experiencia del usuario, con una interfaz clara, intuitiva y responsive, que pueda ser consultada en cualquier momento desde cualquier dispositivo.

8. Seguridad del sistema

Se deben tener en cuenta primitivas básicas de seguridad, como el control de accesos mediante autenticación, validación de datos en el servidor o gestión segura de las sesiones.

9. Información de contacto

El sistema tiene un panel con toda la información de contacto de la página web de incidencias del ayuntamiento. Puedes mandar el mensaje a través del formulario que puedes encontrar en el panel de contacto o mandar un mensaje al correo que puedes encontrar en la información que te redirige directamente a tu app de contacto como outlook o llamar por teléfono.

3.3. Requisitos no funcionales

En este caso, definen aspectos que no están directamente relacionados con las funcionalidades, pero que son fundamentales para que funcione correctamente y con un buen rendimiento.

1. Rendimiento

La interfaz ha sido diseñada para poder crear incidencias de manera óptima, asegurando tiempos de respuesta adecuados.

2. Usabilidad

Al enfocarnos en la experiencia de usuario, debemos priorizar la facilidad de uso y claridad, así como hacerla responsive para permitir el acceso desde cualquier dispositivo.

3. Seguridad

Incorporamos medidas básicas de seguridad como autenticación o control de acceso mediante roles, así como validar los datos de los formularios antes de introducirlos.

4. Escalabilidad

Gracias al uso de una base de datos en un Cloud, el sistema puede crecer en número de usuarios y volumen de datos. Podríamos ampliar la plataforma sin necesidad de cambios estructurales muy significativos.

5. Mantenimiento

Al seguir un patrón MVC debemos facilitar la organización del código y su mantenimiento a largo plazo. Además al usar OR-Mapping facilitamos la escalabilidad y legibilidad.

4. Diseño del Sistema

4.1. Arquitectura de la Información

La plataforma mantiene una estructura jerárquica con varias secciones principales que ya se han mostrado anteriormente, con diferentes áreas definidas según el usuario. De esta manera, definimos la estructura general del sistema con las siguientes páginas principales:

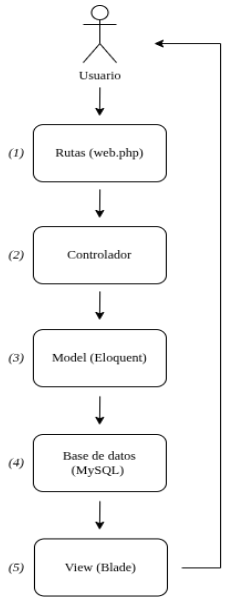
- Página principal
- Incidencias
- Panel del ayuntamiento
- Contacto
- Panel administrador

Dentro de cada una, encontramos funcionalidades específicas relacionadas con diferentes gestiones. Además, se cuenta con la página de Iniciar Sesión así como Registrar Usuario.

Todas estas páginas tienen una estructura en común que facilita la navegación: el header, el footer y los menús laterales.

Por otro lado, en cuanto a arquitectura, ya se ha mencionado que el sistema sigue el patrón Modelo-Vista-Controlador, que además es el propio del framework Laravel. De esta forma, se separa la lógica de negocio de la presentación y del acceso a datos gracias a que los controladores hacen de intermediarios. Conseguimos así un sistema más escalable,

mantenible y seguro. A continuación, explicaremos el flujo normal del sistema, el cual podemos ver reflejado además en el diagrama siguiente:



(1) El usuario interactúa con la web y solicita algún recurso, por ejemplo, consultar una incidencia o rellenar un formulario.
(2) Mediante las rutas definidas, Laravel recibe la petición y la envía al controlador correspondiente.
(3) El controlador recibe la petición, pide datos al modelo y prepara la información.
(4) El modelo se comunica con la base de datos.
(5) Se busca en la base de datos donde está almacenado todo el contenido del sistema la información requerida.
(6) Desde la capa de presentación se muestran todos los datos al usuario.

Figura 1. Modelo-Vista-Controlador (flujo del sistema)

4.2. Diseño de la Base de Datos

Para el almacenamiento y la gestión de la información de *GovConnect*, se ha usado un modelo de base de datos relacional alojado en TiDB Cloud. El esquema de la base de datos se ha diseñado evitando la redundancia de datos y garantizando la integridad referencial.

El modelo se articula en torno a entidades principales que interactúan entre sí mediante relaciones clave. Las tablas relevantes del sistema son:

- **Users**

Esta tabla se ha hecho para almacenar la información de las personas autenticadas que interactúan con el sistema. Algunos de los campos principales son el identificador único (`user_id`), nombre, correo electrónico, contraseña encriptada y el rol del usuario.

A la hora de crear una nueva cuenta, el usuario puede marcar la opción de registrarse como administrador. El control de este tipo de rol se refleja en la organización de datos como una variable lógica llamada role, que puede tener dos valores ‘usuario’ o ‘admin’ según el rol que desempeñan en la página web.

El objetivo de esta tabla es garantizar el control de acceso, permitir el seguimiento de las acciones de cada usuario y controlar sus acciones distinguiendo al administrador del resto de usuarios.

Algún ejemplo de cómo acceder al sistema como administrador (para validar y comprobar sus gestiones) se presenta en la tabla a continuación. Para entrar como usuario normal lo recomendable es crear uno nuevo mediante la página de Registro.

Tabla 1. Credenciales para acceder al sistema como usuario administrador

Rol	***Correo electrónico***	***Contraseña***
Administrador	admin@govconnect.es	administrador

- **Incidencias.**

La tabla de incidencias es una de las principales y más importantes del sistema, puesto que es donde almacenamos todos los datos de los reportes que los ciudadanos hayan compartido.

Los campos principales incluyen atributos como el identificador, título, descripción detallada, ubicación, ruta de la fotografía y el estado actual del reporte (pendiente, en progreso o resuelta). Este último valor sólo podrá ser modificado por un administrador que compruebe el seguimiento y lo valide.

Añadimos también una clave foránea (user_id) que vincula esta tabla con la de usuarios. Esta técnica es usada para registrar qué usuario ha reportado una incidencia, información que se refleja en la página "Detalle de incidencia". En esta página, usando `$incidencia->user->email` conseguimos que cada reporte contenga el dato del autor, concretamente el email con el que creó su cuenta. En caso de no estar registrado (usuario invitado), saldrá como anónimo, gracias a haber definido user_id como nullable.

Para mantener el sistema organizado y facilitar el filtrado de reportes, la tabla Incidencias, decidimos crear una clasificación con los principales tipos de problemas que suelen surgir en la vía pública, definiendo categorías como Limpieza, Infraestructura, Iluminación y Seguridad.

- **Mensajes**

Desde la página de contacto permitimos que los usuarios registren dudas o quejas. Cada uno de estos mensajes, enviados mediante un formulario, se almacenan en la base de datos del sistema *GovConnect*. Posteriormente los mensajes podrán visualizarse desde el panel de administrador.

Todos los campos que el usuario completa se almacenan directamente en la base de datos, siendo todos estos obligatorios. En cuanto a parámetros, encontramos en primer lugar el id (con #) que se genera al enviar el formulario, además del nombre introducido por el ciudadano, su correo electrónico, el mensaje que el individuo desee compartir y por último la fecha cuando se envió el mensaje.

Toda la información de la base de datos queda reflejada en el siguiente diagrama Entidad/Relación:

Figura 2. Diagrama E/R del sistema GovConnect

4.3. Diseño de navegación

El objetivo principal de la aplicación web era ofrecer una estructura clara y lo más intuitiva posible, por lo que hemos definido distintas rutas principales que permitirán al usuario desplazarse por las diferentes páginas y funcionalidades del sistema. Las rutas principales de *GovConnect* son las siguientes:

```
/
├── Página principal
│
├── Autenticación
│   ├── /login
│   ├── /register
│
├── Incidencias
│   ├── /incidencias
│   ├── /incidencias/{id}
│   ├── /reportar
│
├── Panel del Ayuntamiento
│   ├── /panel
│
├── Usuario
│   ├── profile
│
├── Información
│   ├── /contacto
```

El propio nombre indica cual es la funcionalidad de cada una de estas páginas. Además, a lo largo de esta memoria se explica la utilidad y objetivo de cada una de ellas.

5. Implementación Técnica (Desarrollo)

5.1. Configuración del Entorno Laravel

Hemos aprovechado la arquitectura que ofrece Laravel, configurando el entorno con el patrón Modelo-Vista-Controlador. Además, posee una estructura interna muy completa que facilita el desarrollo de aplicaciones web, con directorios definidos y bien estructurado. Por lo tanto, gestionamos esta separación lógica a través de:

Rutas (routes/web.php)

Actúa como punto de entrada de la web. En este archivo se han definido las URLs a las que accede el usuario mediante la interfaz. Para ello, se han empleado peticiones GET que se encargan de servir las vistas (por ejemplo, cargar el formulario del mapa o mostrar el detalle de una incidencia).

Las peticiones POST han sido muy útiles para la recepción y procesamiento de datos sensibles (como el registro de una nueva incidencia o el envío de un mensaje de contacto), asegurando además estas rutas con la directiva @csrf, que ayuda a evitar ataques de falsificación de peticiones en sitios cruzados.

Controladores (Controllers)

Conforman el núcleo de la plataforma, actuando como intermediarios entre lo que el usuario solicita y los datos almacenados. Un ejemplo clave de la implementación es `IncidenciaController`.

Este controlador agrupa la lógica de negocio necesaria para gestionar los reportes urbanos. Entre sus responsabilidades destacan:

- Recibir los datos del formulario de reporte y validarlos.
- Determinar el origen de la petición: si el usuario ha iniciado sesión, el controlador asocia el reporte a su identificador; si es un visitante, permite la creación del registro asociándolo a un estado anónimo (valor nulo).
- Preparar la información recuperada de la base de datos y retornarla a las vistas Blade correspondientes, como `reportarIncidencia` o `detalleIncidencia`.

Modelos (Models)

Los modelos representan las entidades de las bases de datos dentro del código PHP, como User, Incidencia y Mensaje. Para garantizar la seguridad, cada modelo define explícitamente qué atributos pueden ser rellenados directamente desde un formulario mediante la propiedad *Fillable*.

Además en los modelos se definen las relaciones estructurales mediante funciones. En el modelo Incidencia se ha implementado la función user(), lo que permite vincular cada reporte de manera dinámica con su creador, de esta forma conseguimos consultas ágiles sin necesidad de escribir sentencias complejas en SQL.

5.2. Frontend con Tailwind CSS

Para el desarrollo de la interfaz de usuario, hemos combinado el motor de plantillas Blade de Laravel con el *framework* Tailwind CSS. Esta integración ha permitido construir un diseño visualmente atractivo y adaptable a cualquier dispositivo.

Con el objetivo de evitar la duplicidad de código y asegurar la coherencia visual en toda la plataforma GovConnect, se han extraído los elementos estructurales comunes en componentes independientes. Específicamente header, footer y menú se han programado como fragmentos separados.

Menú superior. Actúa como la barra principal de la aplicación. Mediante el uso de contenedores flexibles, los elementos (como el logotipo, el perfil del usuario y los accesos rápidos) se alinean y distribuyen equitativamente en la parte superior.

Menú lateral. En pantallas de escritorio, este menú se mantiene visible de forma permanente, (Incidencias, Panel del Ayuntamiento, Contacto). Con las clases condicionales de Tailwind, el menú lateral desaparece automáticamente en dispositivos pequeños. En su lugar, el sistema muestra botones interactivos (tipo "hamburguesa") que permiten al usuario desplegar la navegación lateral solo cuando la necesita.

5.3. Gestión de Usuarios y Roles

La gestión de usuarios y roles se basa en el modelo User de Laravel y en un campo principal llamado role.

En `User.php` , el usuario tiene campos como name, surname, email, phone, address, city, postal_code, password y role. La contraseña está protegida porque Laravel la oculta en serialización y además se castea como hashed, aunque en tu controlador también la estás cifrando manualmente con `Hash::make`, asi evitamos que la contraseña se guarde en texto plano en la base de datos

El sistema de roles funciona con el campo **`role: usuario | admin.`**

En el modelo User.php se presentan varios métodos para comprobar la naturaleza del usuario. Para ello, se compara “role” con las dos posibilidades de usuario registrado: administrador o ciudadano común. Los métodos tienen el siguiente aspecto:

```
public function isAdmin(): bool {
   return $this->role === 'admin';
}
}
```

```
public function isNormalUser(): bool {
   return $this->role === 'usuario';
}
```

Por tanto, solo un usuario autenticado y con role = admin puede acceder a las rutas protegidas. Las rutas protegidas están en web.php y permiten al administrador gestionar incidencias y mensajes de contacto.

En la interfaz también se usa el rol. Por ejemplo, en el menú `menu.blade.php`, solo se muestra el enlace al panel admin si: **`Auth::user()->isAdmin()`**.En el header además, se muestra una etiqueta distinta según el rol: Admin o Usuario Normal.

5.4. Formularios

El primer formulario imprescindible para gestionar cuentas es el proceso de registro de usuarios. Los formulario de registro y de login cuentan con validaciones y mensajes de error para todos los campos en caso de que no se cumplan las validaciones:

Tabla 2. Campos y comprobaciones en el formulario de Registro de Usuario

Campo	***Validaciones***	***Mensaje de Error***
Nombre	• Requerido • Máximo 255 caracteres	"El nombre es obligatorio"
Apellidos	• Requerido • Máximo 255 caracteres • Solo texto	"Los apellidos son obligatorios"
Email	• Requerido • Debe ser email válido • Email único (no puede existir en la BD)	"El email es obligatorio" "El email debe ser válido" "El email ya está registrado"
Teléfono	• Requerido • Máximo 20 caracteres	"El teléfono es obligatorio"
Dirección	• Requerido • Máximo 255 caracteres • Solo texto	"La dirección es obligatoria"
Ciudad	• Requerido • Máximo 100 caracteres • Solo texto	"La ciudad es obligatoria"
Código Postal	• Requerido • Máximo 20 caracteres • Solo texto	"El código postal es obligatorio"
Contraseña	• Requerido • Mínimo 8 caracteres • Debe coincidir con confirmación	"La contraseña es obligatoria" "La contraseña debe tener al menos 8 caracteres" "Las contraseñas no coinciden"

Confirmar contraseña	• Requerido • Debe coincidir con contraseña	"Las contraseñas no coinciden"
Términos y condiciones	• Debe estar marcado	"Debes aceptar los términos y condiciones"

Para almacenar las contraseñas, usamos el algoritmo por defecto de Laravel llamado bcrypt, aplicando hash a la contraseña antes de introducirla en la base de datos. Esto hace que la información de nuestros clientes esté segura y es un paso en la construcción de un sistema seguro y robusto.

Para acceder a tu perfil de usuario (Login), debes completar otro formulario con los siguientes datos:

Tabla 3. Campos y comprobaciones en el formulario de Inicio de Sesión

Campo	***Validaciones***	***Mensaje de Error***
Email	• Requerido • Debe ser email válido	"El correo electrónico es obligatorio" "El correo electrónico debe ser válido"
Contraseña	• Requerido • Mínimo 8 caracteres • Debe ser contraseña válida	"La contraseña es obligatoria" "La contraseña debe ser válida"

El formulario para reportar una incidencia contiene varios parámetros que permiten detallar con precisión el problema que ha surgido.

Tabla 4. Campos y comprobaciones en el formulario de Reporte de Incidencia

Campo	***Validaciones***	***Mensaje de Error***
Título	• Requerido	"Rellene este campo"
Categoría	• Requerido	"Seleccione un elemento de la lista"
Descripción	• Requerido	"Rellene este campo"
Ubicación	• Requerido	"Haz click en el mapa para situar la incidencia"
Fotografía	• Opcional	"No se ha podido subir la imagen"

Para enviar un mensaje mediante la página de Contacto, debemos rellenar los siguientes campos:

Tabla 5. Campos y comprobaciones en el formulario de Contacto

Campo	***Validaciones***	***Mensaje de Error***
Nombre completo	• Requerido • Debe ser distinto de nulo	"El campo debe ser completado"
Email	• Requerido • Debe ser email válido	"El correo electrónico es obligatorio" "El correo electrónico debe ser válido"
Mensaje	• Requerido • Debe ser distinto de nulo	"El campo debe ser completado"

6. Despliegue

Tras probar varias herramientas decidimos desplegar nuestro sitio web desarrollado en Laravel usando la plataforma Railway, desde donde podemos gestionar automáticamente la base de datos. Nuestra primera opción fue usar Vercel, pero pronto nos dimos cuenta de que no era compatible con proyectos de nuestro tipo (desarrollados con Laravel)

Railway facilitó el proceso gracias a que se puede conectar directamente con Git, en nuestro caso, ya estábamos trabajando colaborativamente y gestionando el control de versiones mediante GitHub. Además, *GovConnect* funcionaba a la perfección en local.

Para configurar el entorno, debemos configurar las variables relacionadas con la base de datos MySQL y enlazarla al proyecto. Una vez configurado, con cada push al repositorio se activa un nuevo despliegue automático, lo que hace que el sitio esté siempre actualizado en su última versión.

7. Pruebas y Validación

7.1 Pruebas Funcionales

En este apartado, veremos algunas de las prueba funcionales realizadas sobre el sistem *GovConnect*, con el objetivo de verificar que todas las acciones dinámicas se pueden llevar a cabo. Las pruebas se centran en mostrar el comportamiento del lado de los usuarios (sin importar el rol), validando actividades como el registro de usuario, la creación de incidencias o el envío de mensajes, entre otros. Una vez explicada la lógica detrás de cada formulario, se muestran ejemplos del funcionamiento. Una vez explicados los requisitos, veremos algún ejemplo del funcionamiento.

1. Crear incidencia

El flujo normal, sería crearla desde la página “Reportar Incidencia” y debemos recibir el mensaje de confirmación: “¡Su reporte se ha enviado con éxito!

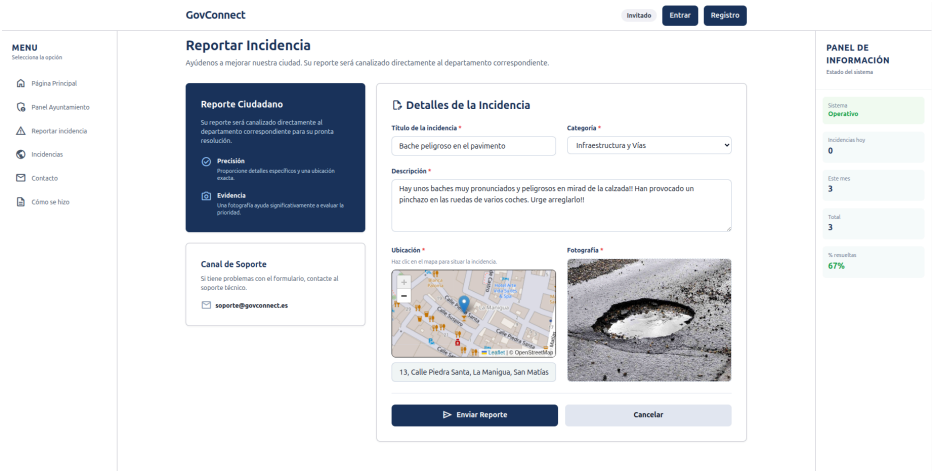


Figura 3. Página de Reportar Incidencia

Una vez suceda, podremos verificar su estado y otros desde la página “Incidencias” donde deberán aparecer todos sus datos junto a un enlace para ver detalle.

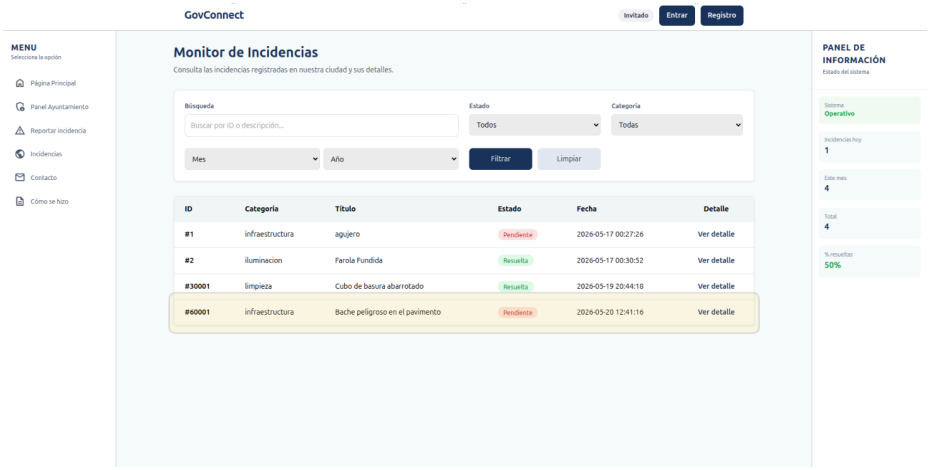


Figura 4. Página de Incidencias

Si accedemos al detalle, nos aparecerán todos los datos de la incidencia concreta en vez de un resumen en una tabla global, muy útil para ver el problema exacto.

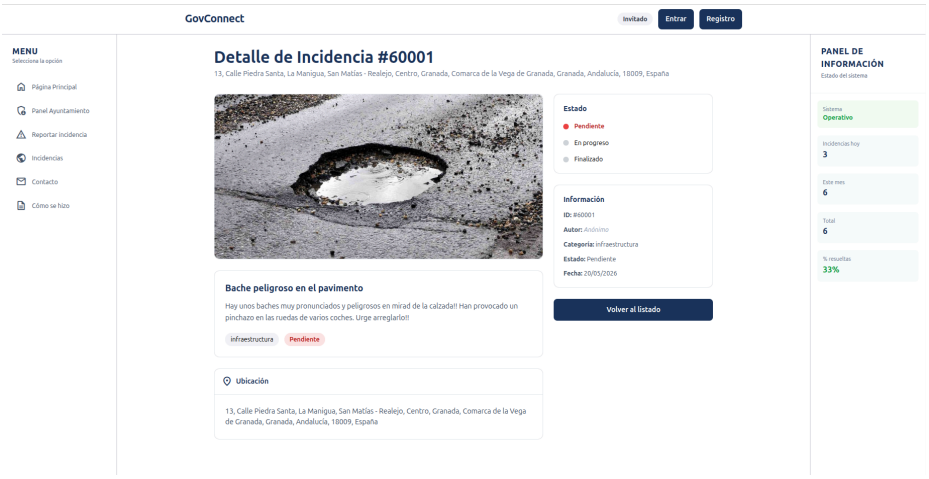


Figura 5. Página de Detalle de Incidencia

Por último, debemos destacar la importancia de las incidencias en el panel del administrador, puesto que es el único que puede cambiar el estado y gestionarlas. En este caso como ejemplo, el administrador ha cambiado el estado de la incidencia #60001 a "En progreso", aunque también se podría eliminar.

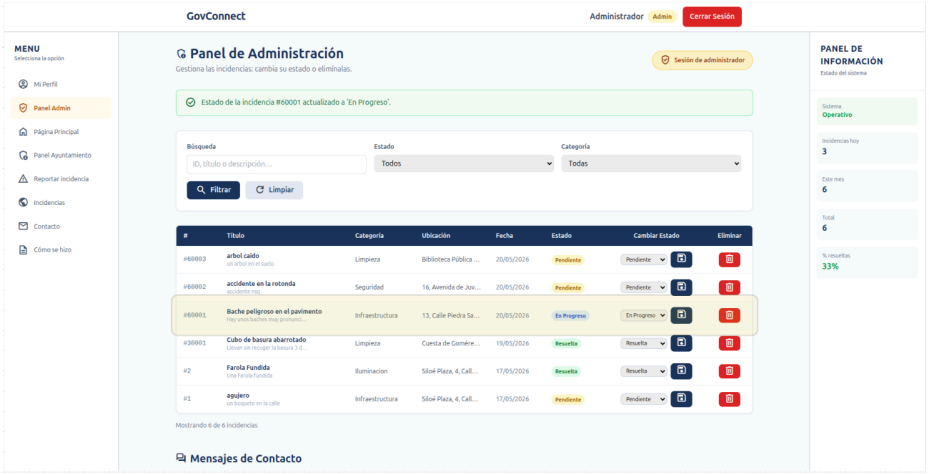


Figura 6. Panel del administrador

2. Registrar usuario

Esta acción es un clásico de cualquier sistema. El usuario accederá a la página de Registro y completará todos los parámetros que se le solicitan. En este caso son todos obligatorios.

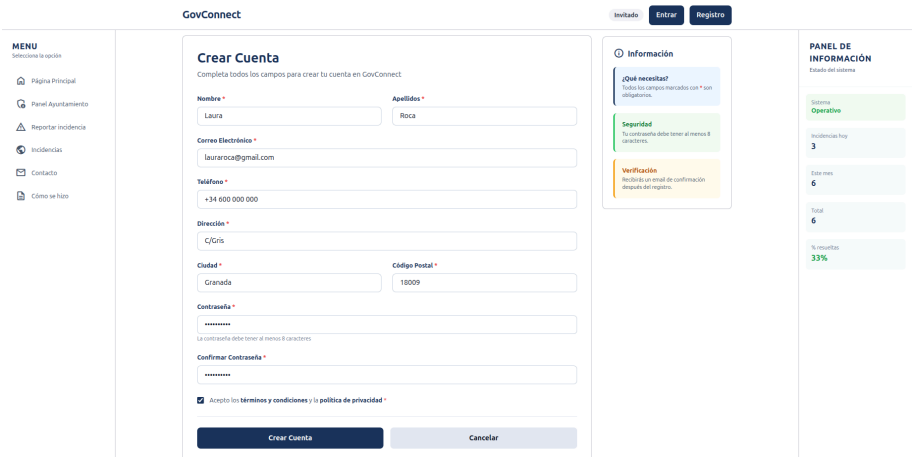
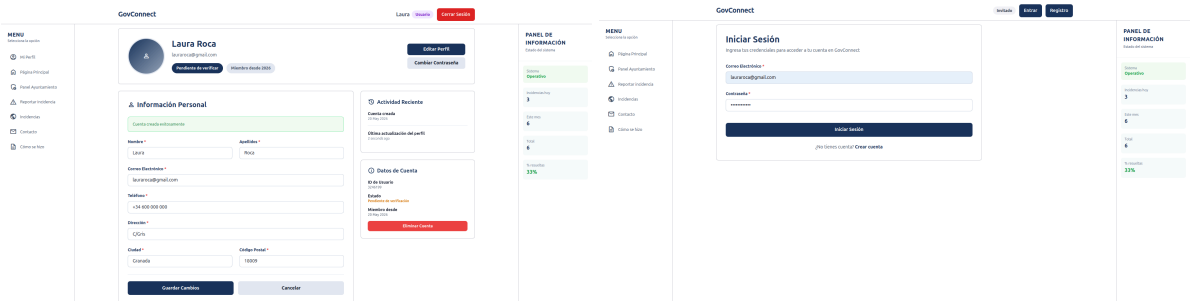


Figura 7. Página de Registro de Usuario (crear una cuenta)

Si se creó con éxito el usuario, nos redirigirá a la página de Mi Perfil, sino, indicará cual es el problema o inconveniente. Con estos mismos datos introducidos, podremos iniciar sesión y acceder cuando queramos a nuestra página personal, donde podremos actualizar nuestros datos en cualquier momento.



Figuras 8 y 9. Perfil de Usuario e Inicio de Sesión

3. Enviar mensaje (Contacto)

Como última acción importante, debemos mencionar el envío de mensajes mediante la página de contacto. El usuario escribirá una comunicación, preocupación o duda y llegará directamente al panel del administrador. Si se ha realizado con éxito, nos devolverá el mensaje “Formulario enviado correctamente, en breve nos pondremos en contacto con usted”

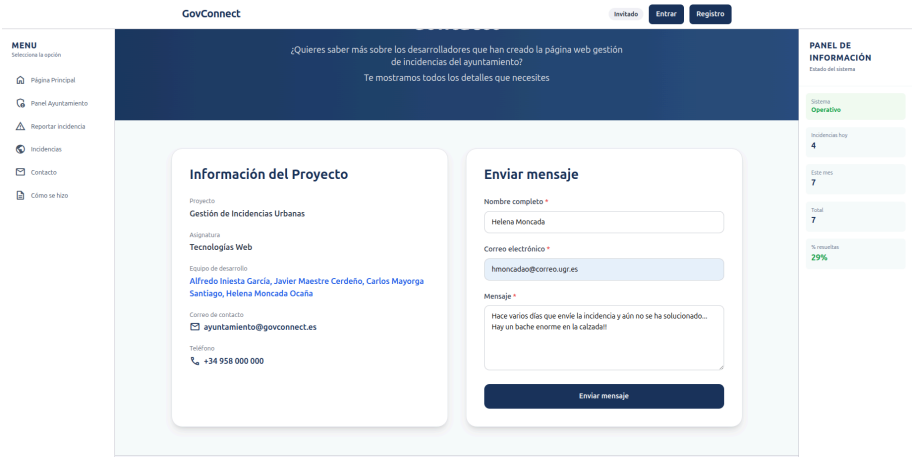


Figura 10. Página de Contacto

El administrador podrá decidir si quiere eliminar el mensaje o no una vez se haya solucionado la consulta. Si se ha eliminado correctamente, devolverá un mensaje como el siguiente “Mensaje #id eliminado correctamente”.

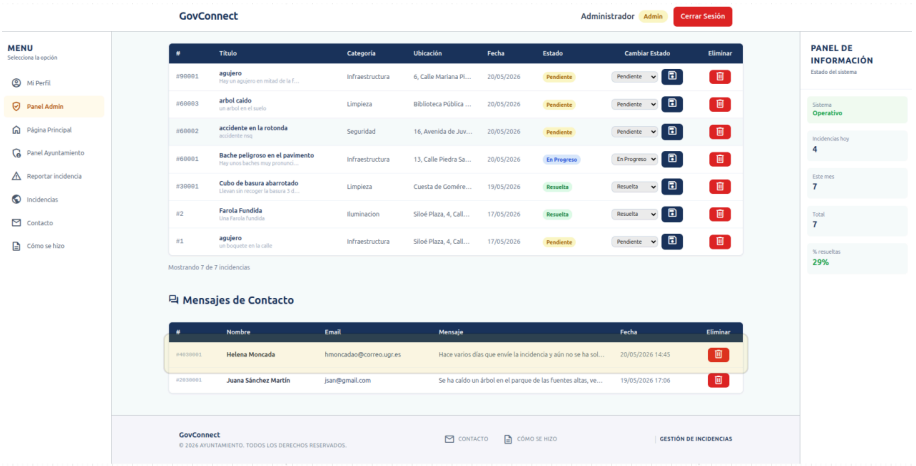


Figura 11. Visualización de mensajes desde el Panel del Administrador

7.2 Validación Responsive

A continuación, se muestran ejemplos de cómo se ve *GovConnect* desde dispositivos con diferentes resoluciones, desde móvil hasta tablet, pasando por supuesto por la versión escritorio (ordenador), que ha sido la vista sobre la que hemos desarrollado. Además, se procura mostrar ejemplos con diferentes roles y páginas, como se ve en las tablas presentadas a continuación:

Tabla 6. Vista responsive desde ordenador

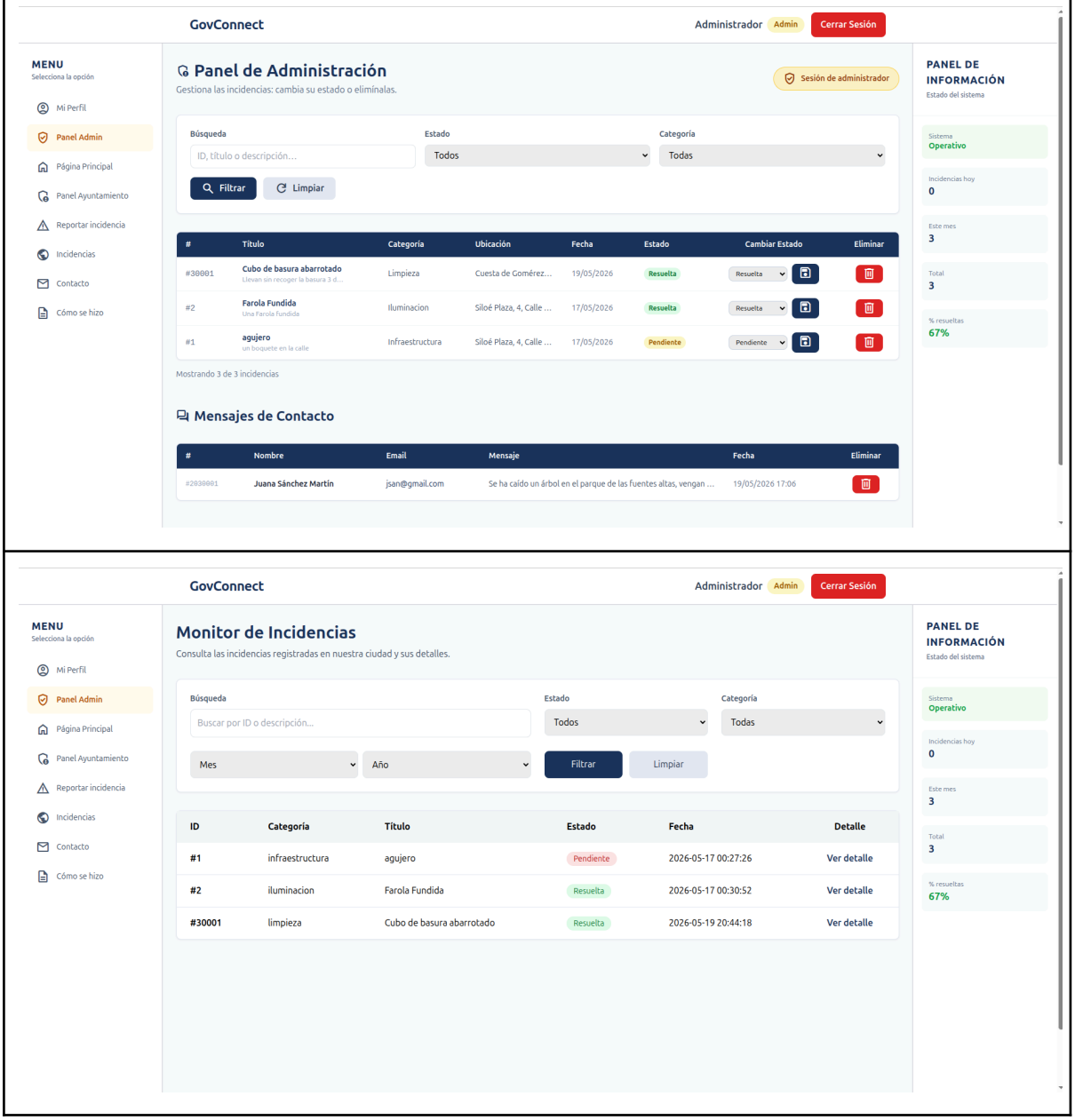
Vista desde Ordenador


Tabla 7. Vista responsive desde Tablet

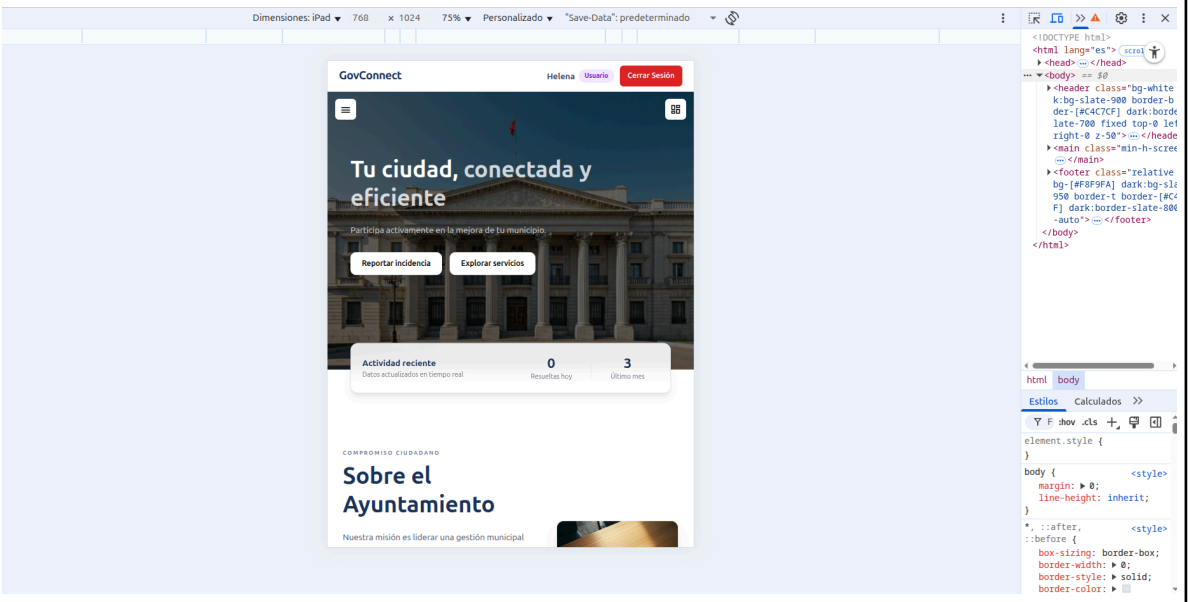
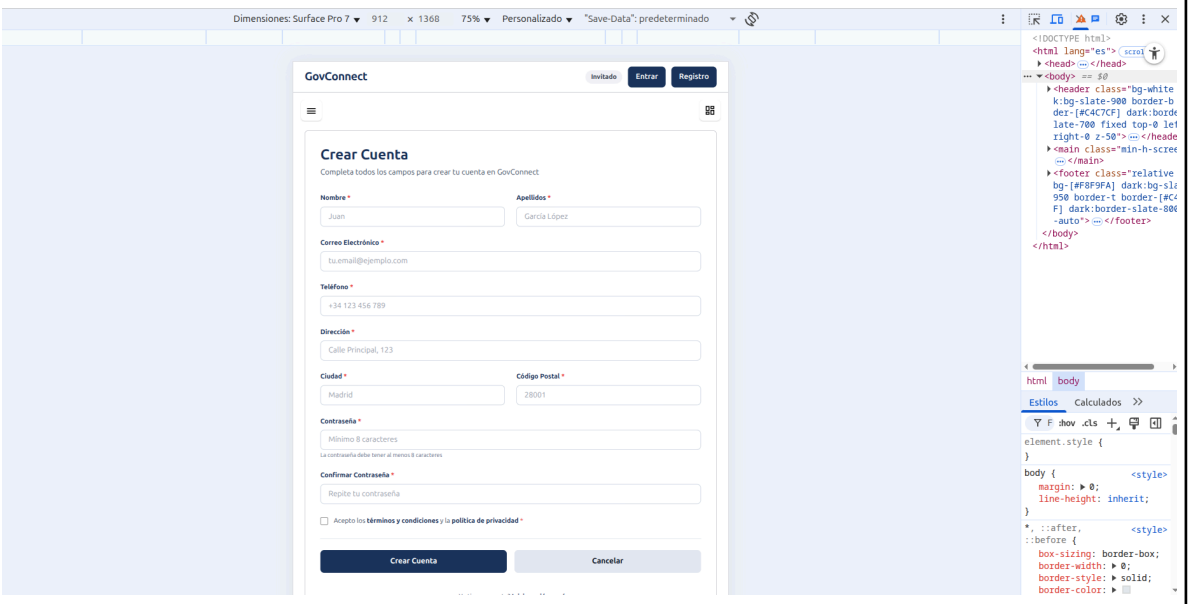
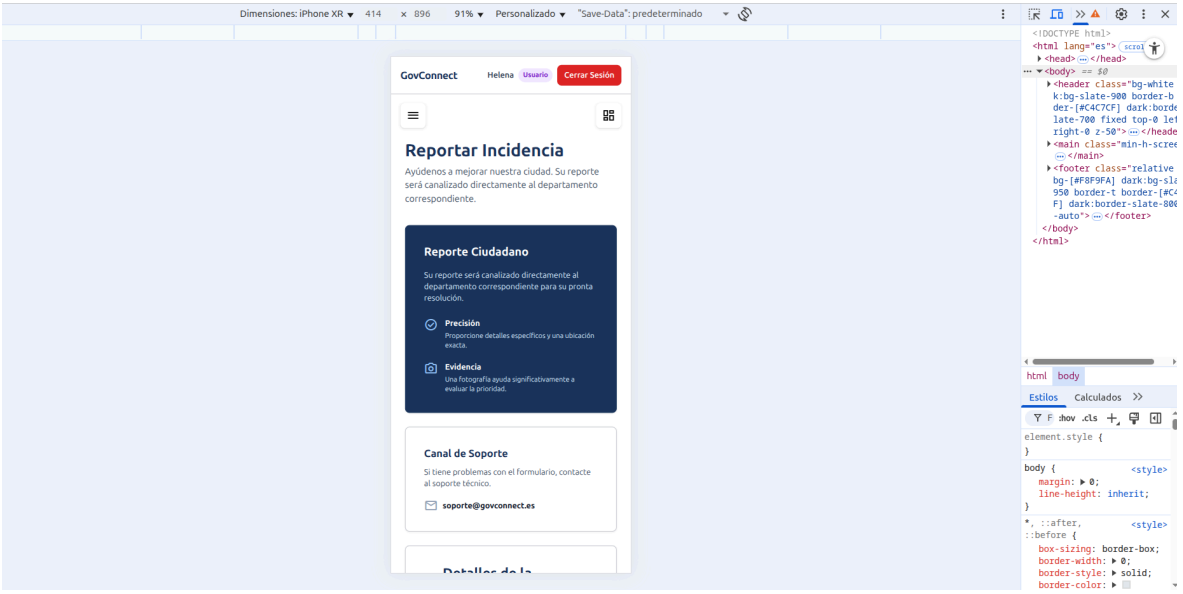
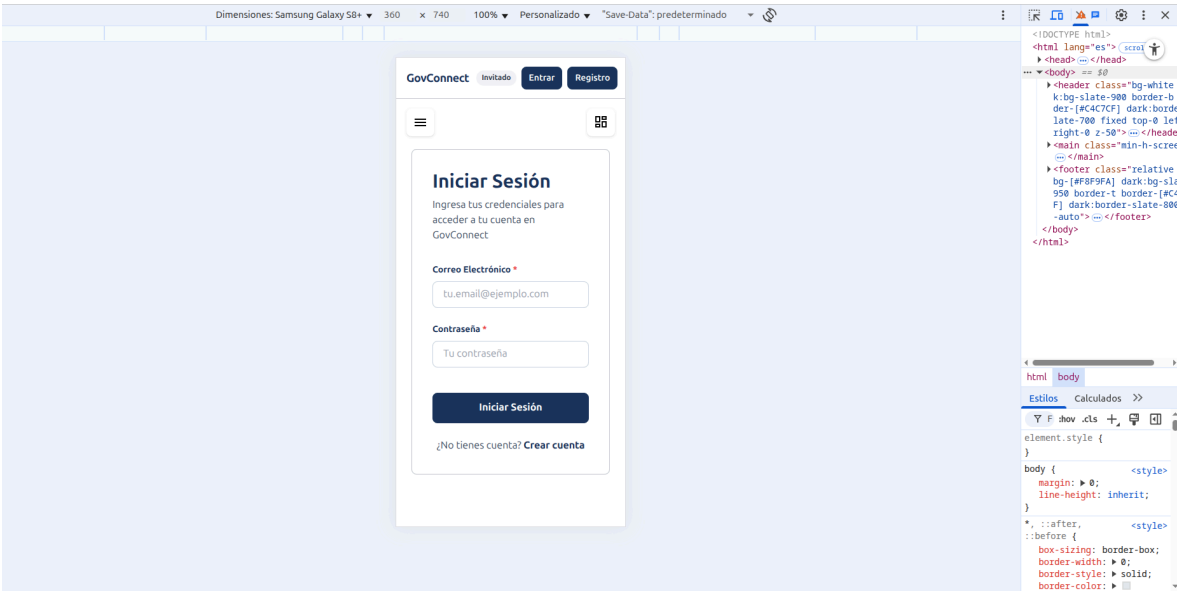
Vista desde Tablet

Dimensiones: Ipad (768 x 100)

Dimensiones: Surface Pro 7 (912 x 1368)

Tabla 7. Vista responsive desde móvil

Vista desde Móvil

Dimensiones: iPhone XR (414 x 895)

Dimensiones: Samsung Galaxy S8+ (360 x 740)

8. Funcionalidades adicionales

Además de cumplir todos los requisitos básicos que se piden en el guión de la práctica, nos pareció interesante añadir ciertas funcionalidades que convierten *GovConnect* en un sistema completo que prioriza la experiencia del usuario. Algunas de las mejoras que hemos implementado son:

- Implementación de un menú lateral derecho con un "Panel de Información" que muestra el estado del sistema.
- Integración de diversos mapas interactivos que permiten seleccionar una ubicación o visualizar las incidencias que se han dado en una zona concreta.
- Buscadores de incidencias con diferentes filtros según el parámetro por el que quieras buscar.
- Página de inicio con información sobre el ayuntamiento y el objetivo de *GovConnect*.
- Creación e implementación del favicon (presente además en la cabecera de esta misma memoria) que mejora la identidad y reconocimiento de nuestro sitio web.

9. Conclusión y Mejoras

Como mejora, una gran idea sería aumentar la funcionalidad del administrador y su interacción con los distintos usuarios. Por ejemplo, sería interesante que el pudiera conceder ciertos permisos según la persona y la experiencia en el sistema, como si fuese un sistema de beneficio.

Además, en cuanto a usuarios, sería interesante que una vez dados de alta reciban un correo electrónico para poder verificar su cuenta en nuestra plataforma, haciendo que los usuarios sean seguros. Además, extrapolar el perfil y permitir personalizarla añadiendo fotos o manteniendo un registro de todos las acciones que ha realizado un determinado perfil desde el día en el que se registró.

Por último, GovConnect de momento está enfocado únicamente en la gestión de incidencias urbanas, pero no deberíamos descartar la idea de convertirlo en una página dónde se puedan hacer más trámites relacionados con la institución clave de cualquier ciudad, como solicitud de licencias, tributos o pago de multas.

Como conclusión final, nos ha parecido un proyecto completo, que nos ha dotado de grandes conocimientos técnicos. Con más tiempo de desarrollo, podría convertirse en una aplicación mucho más robusta y segura, añadiendo por ejemplo notificaciones en tiempo real, mensajes al correo electrónico de cada usuario o suscripciones a listas de noticias.

9. Anexos

Para consultar el código fuente y el repositorio en el que hemos trabajo visitar el siguiente enlace: https://github.com/hmoncadao/TW-Proyecto.

La plataforma *GovConnect*, visible hasta el 20 de junio de 2026, se puede visitar en el siguiente enlace: https://govconnect.up.railway.app/.