

MEMORIA DE PROYECTO DESARROLLO APLICACIONES WEB

Asturutas una web de rutas de la C.A. de Asturias

Iker Alonso García

2 DAW IES Bernaldo de Quirós

1. INTRODUCCIÓN.....	2
PRESENTACIÓN Y OBJETIVOS	2
CONTEXTO	2
PLANTEAMIENTO DEL PROBLEMA.....	3
ANÁLISIS DE COSTES.....	3
PLAN DE FINANCIACIÓN	3
PLAN DE RECURSOS HUMANOS.....	3
2. ESPECIFICACIÓN DE REQUISITOS	4
INTRODUCCIÓN	4
DESCRIPCIÓN GENERAL.....	4
REQUERIMIENTOS FUNCIONALES	4
REQUERIMIENTOS DE INTERFACES EXTERNAS.....	4
REQUERIMIENTOS DE RENDIMIENTO.....	4
OBLIGACIONES DEL DISEÑO	4
ATRIBUTOS	5
3. ANÁLISIS	6
INTRODUCCIÓN	6
DIAGRAMA DE CLASES	6
DIAGRAMAS DE CASOS DE USO	7
4. DISEÑO	9
INTRODUCCIÓN	9
CAPA DE PRESENTACIÓN	9
CAPA DE NEGOCIO O LÓGICA DE LA APLICACIÓN	9
CAPA DE PERSISTENCIA O DATOS.....	9
5. IMPLEMENTACIÓN	10
TECNOLOGÍAS UTILIZADAS EN EL DESARROLLO DEL PROYECTO	10
CAPA DE PRESENTACIÓN	10
CAPA DE NEGOCIO O LÓGICA DE LA APLICACIÓN	23
CAPA DE PERSISTENCIA O DE DATOS	32
6. DESCARGA DE HERRAMIENTAS, CONFIGURACIÓN Y DESPLIEGUE DE LA APLICACIÓN	33
DESCARGA Y CONFIGURACIÓN DE HERRAMIENTAS	33
DESPLIEGUE DE LA APLICACIÓN.....	38
7. EVALUACIÓN	39
INTRODUCCIÓN	39
VALIDACIONES DE PÁGINAS DE ESTILO.....	39
VALIDACIÓN DE ENLACES.....	39
VALIDACIÓN DE LA RESOLUCIÓN	39
VALIDACIÓN DE NAVEGADORES	40
8. VALORACIÓN PERSONAL DEL TRABAJO REALIZADO (ANÁLISIS DAFO + ANÁLISIS CAME)	41
ANÁLISIS DAFO	41
ANÁLISIS CAME	42
9. POSIBLES AMPLIACIONES.....	42



1. Introducción

Presentación y objetivos

Esta aplicación web intenta ofrecer al usuario la posibilidad de conocer (o también reconocer) nuevas rutas del Principado de Asturias, indistintamente de donde se ubiquen. Es muy útil porque una vez dentro de la web, no es necesario tener cuenta para poder disfrutar de muchas de las funcionalidades mas interesantes, como por ejemplo poder filtrar por municipios, por actividad, por si pueden ir personas con movilidad reducida, se puede ir con niñ@s o si también se puede llevar el perro.

Además, cualquier usuario puede subir rutas. Tan solo necesitas crearte una cuenta para poder acceder a todas las funcionalidades de la web, todo totalmente gratuito.

Contexto

Las tecnologías utilizadas son:

Java	Java para el desarrollo del back-end. Conexión con la BBDD y mapeo de identidades, desarrollo de repositorios, servicios, controladores... Se ha utilizado Spring, que es un framework de Java junto a thymeleaf que es un motor de desarrollo de html dinámicos. Dentro de Spring se han utilizado Spring Boot, orientado a desarrollo web, Spring Data, Spring Security y el modelo MVC
HTML	HTML para el desarrollo del esqueleto de la aplicación web.
CSS y Bootstrap	Utilizados para dar color y personalidad a la aplicación
JS	JavaScript para agregar funcionalidad a los botones de la aplicación
PostgreeSQL	Lenguaje utilizado para la creación e implementación de la BBDD



La aplicación de momento estará disponible en versión web, accesible siempre que tengas conexión a internet.

Planteamiento del problema

Los senderistas buscan rutas fiables y compartir experiencias. La información está dispersa y no siempre es precisa, lo que puede resultar en una mala experiencia para el usuario.

El senderismo ha ganado en popularidad, especialmente a raíz de la pandemia, donde las actividades al aire libre se han convertido en una opción preferida por muchos.

Asturutas está dirigida a personas todas las edades, indistintamente, que están interesadas en actividades al aire. Esto incluye tanto a senderistas experimentados como a principiantes que buscan una manera sencilla y fiable de encontrar y compartir rutas.

Análisis de costes

Costes de desarrollo:

- Personal: 30,000 euros (incluye salarios de desarrolladores, diseñadores, y personal de marketing).
- Licencias de software: 10,000 euros (herramientas de desarrollo, servicios en la nube, etc.).
- Infraestructura y servidores: 10,000 euros (alojamiento web, bases de datos, y servicios de mantenimiento).

Costes de operación:

- Mantenimiento de servidores: 5,000 euros anuales.
- Actualizaciones y mejoras de software: 8,000 euros anuales.
- Marketing y promoción: 7,000 euros anuales.

Plan de financiación

El proyecto será financiado mediante una combinación de fondos propios y una subvención de 20,000 euros del programa de apoyo a emprendedores tecnológicos.

Plan de recursos humanos

El equipo estará compuesto por:

- Un jefe de proyecto.
- Dos desarrolladores web.
- Un diseñador UX/UI.



2. Especificación de requisitos

Introducción

Vamos a detallar las necesidades y expectativas de la aplicación web

Descripción general

La aplicación permitirá a los usuarios registrar, subir, y compartir rutas de senderismo y ciclismo de la región de Asturias. Los usuarios podrán ver y filtrar las rutas que comparten el resto de usuarios.

Requerimientos funcionales

- Registro y autenticación de usuarios.
- Subida de rutas con detalles (nombre, descripción, longitud, dificultad, etc.).
- Visualización de rutas.
- Búsqueda y filtrado de rutas según varios criterios (ubicación, dificultad, para gente con movilidad reducida, para ir con niños, con perros...).

Requerimientos de interfaces externas

Interfaces de los Usuarios

- Interfaz web que permite a los usuarios ver y usar la aplicación tanto desde el navegador del móvil como desde el navegador del ordenador

Interfaces Hardware

- Servidores para alojar la aplicación web y la BBDD.
- Equipos de los usuarios, los móviles, los ordenadores...

Interfaces Software

- Sistemas de gestión de BBDD, este caso PostgreSQL.

Interfaces de Comunicaciones

- Protocolo HTTPS para una comunicación segura entre cliente y servidor.
- API REST para la comunicación entre el frontend y el backend.
- Modelo MVC.

Requerimientos de rendimiento

- Capacidad para almacenar a 100 usuarios simultáneos.
- Respuesta rápida del servidor al realizar todas las operaciones sobre el front.

Obligaciones del diseño

- Diseño fácil que permita añadir sin dificultad nuevas funcionalidades
- Uso de frameworks, por ejemplo Bootstrap para el front-end y Spring para el back-end



Atributos

Seguridad

- Cifrado de contraseñas en el registro de usuario. Solo tú sabes tu clave.

Facilidad de mantenimiento

- Código bien estructurado en carpetas, con control de versiones en GitHub.

Portabilidad

- Compatible con todos los navegadores del mercado.
- Capacidad de migrar la BBDD y el back y alojarlo en un servidor si fuese necesario

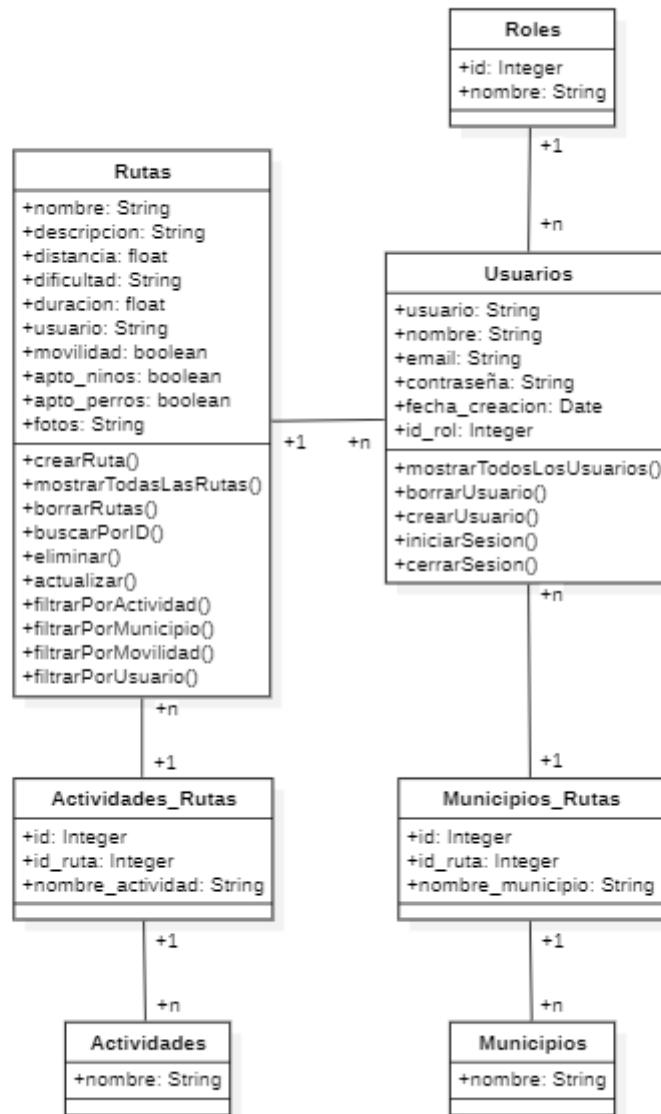


3. Análisis

Introducción

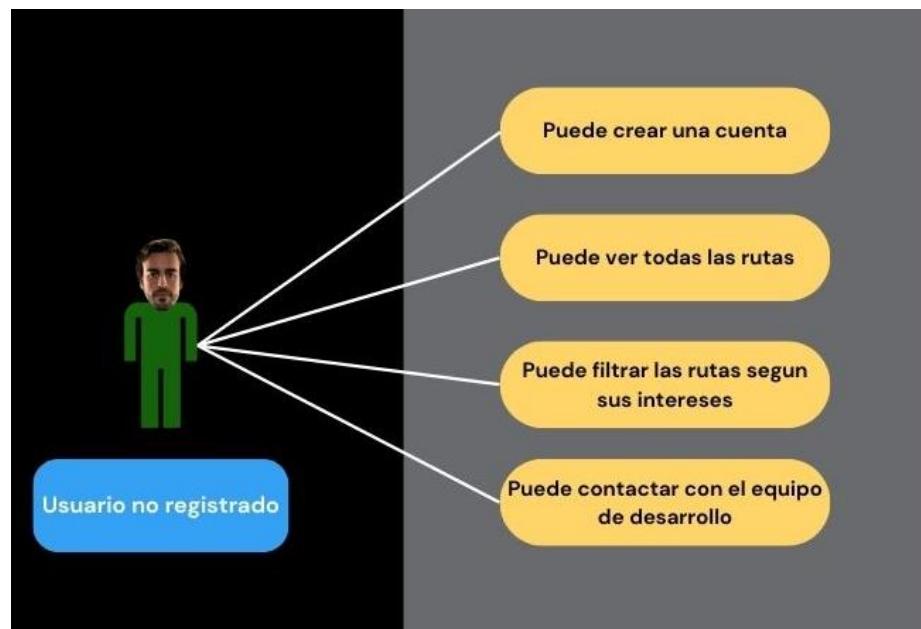
Aquí haremos un desarrollo detallado de la funcionalidad de la aplicación, incluyendo diagramas que ayudarán a entender la estructura y el comportamiento de la misma. Esto incluye un diagrama de clases y los diagramas de casos de uso necesarios.

Diagrama de clases

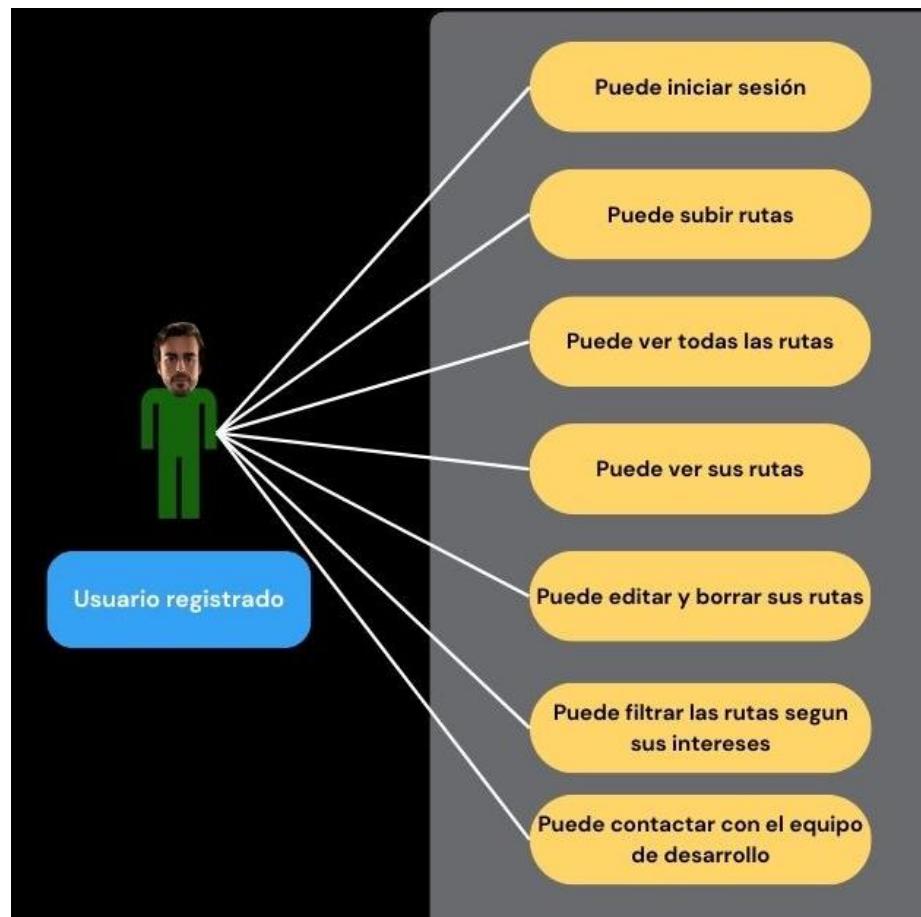


Diagramas de casos de uso

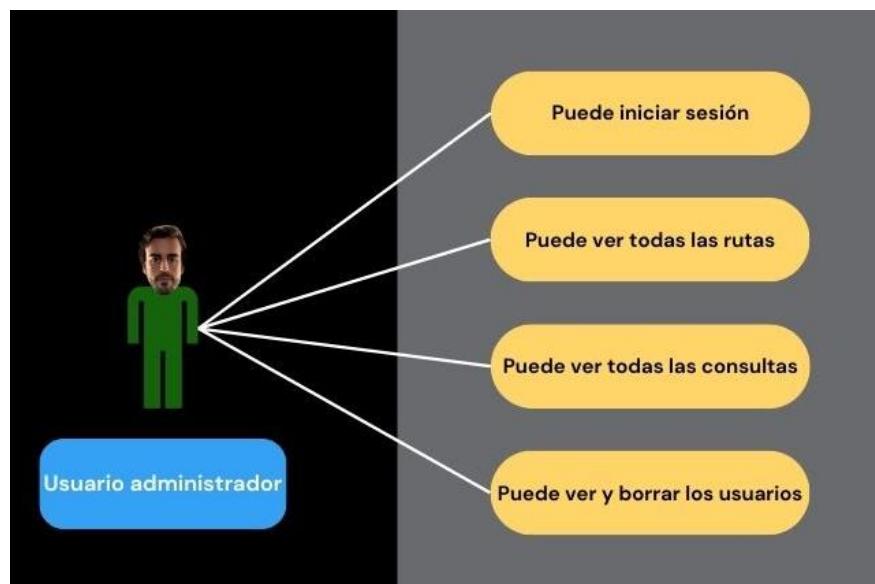
Usuario NO registrado:



Usuario registrado:



Usuario administrador:



4. Diseño

Introducción

La aplicación es útil y accesible, ya que permite disfrutar de muchas de sus funcionalidades sin necesidad de registrarse. Los usuarios pueden filtrar rutas por municipios, actividades, accesibilidad para personas con movilidad reducida, si son adecuadas para niños y si se puede llevar a mascotas. Además, los usuarios pueden subir rutas propias creando una cuenta, con todas las funcionalidades siendo gratuitas.

Capa de presentación

La interfaz de usuario estará diseñada para ser intuitiva y fácil de usar, con un diseño responsive que se adapta a diferentes dispositivos. Para ello utilizaremos Bootstrap. La navegación principal incluirá un menú con enlaces a los distintos sitios de la web y el flujo de navegación será sencillo, permitiendo a los usuarios acceder rápidamente a los sitios principales sin tener que hacer muchos clicks

Capa de negocio o lógica de la aplicación

La aplicación seguirá el patrón de diseño MVC (Modelo-Vista-Controlador)

- Los usuarios no registrados pueden filtrar y ver rutas, pero necesitan registrarse para subir nuevas rutas.
- Las rutas deben incluir información sobre accesibilidad y compatibilidad con niños y mascotas.

Capa de persistencia o datos

La base de datos será diseñada utilizando PostgreSQL.



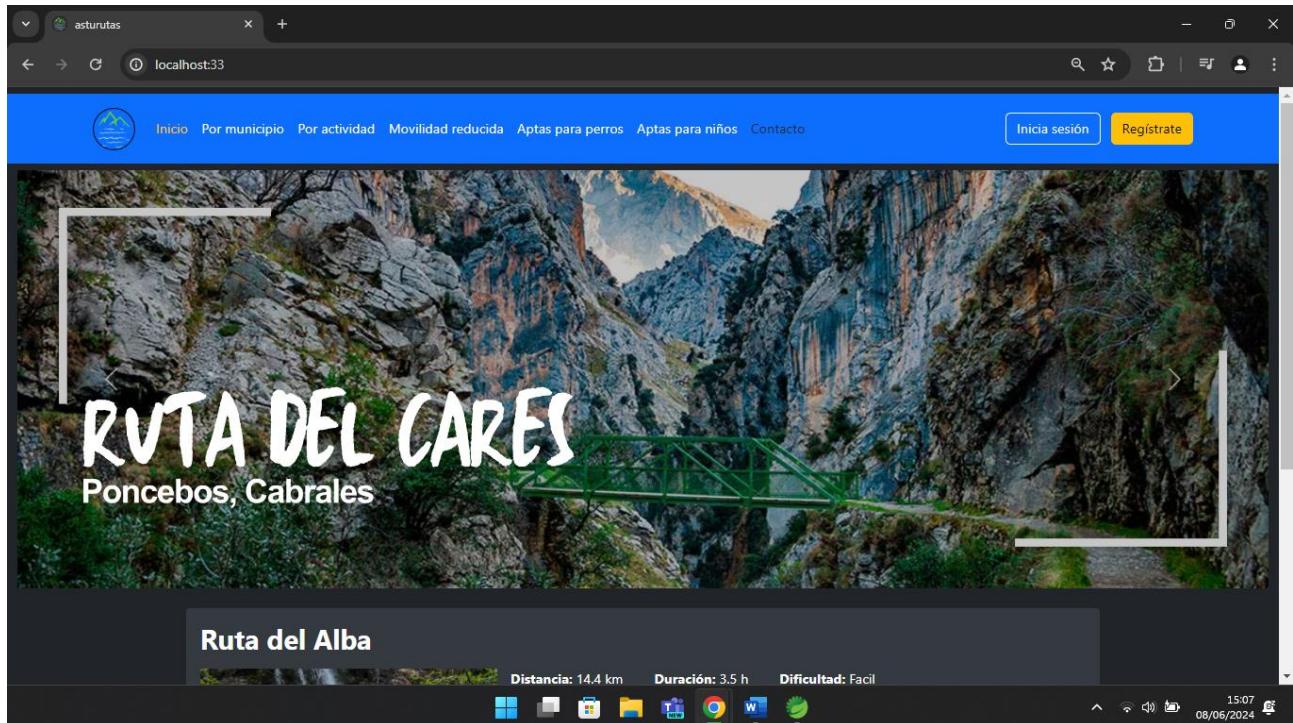
5. Implementación

Tecnologías utilizadas en el desarrollo del proyecto

- HTML: Para generar el esqueleto de la aplicación web.
- CSS y Bootstrap: Para hacer una aplicación más visual y acorde a lo que muestra, además de una interfaz responsive.
- JS: Para agregar funcionalidad a lo que ve el usuario, como los botones, o bien el carrusel.
- Java y Spring: Para la parte que “no se ve” de la aplicación. Mapeo de entidades, conexión con la base de datos, filtros de seguridad, consultas SQL. Se emplea el modelo vista controlador.
- Thymeleaf: Motor de JAVA para hacer plantillas de HTML5 dinámicas.
- PostGreSQL: Lenguaje utilizado para la gestión de la base de datos, es donde se almacena toda la información que envía el usuario desde la interfaz.

Capa de presentación

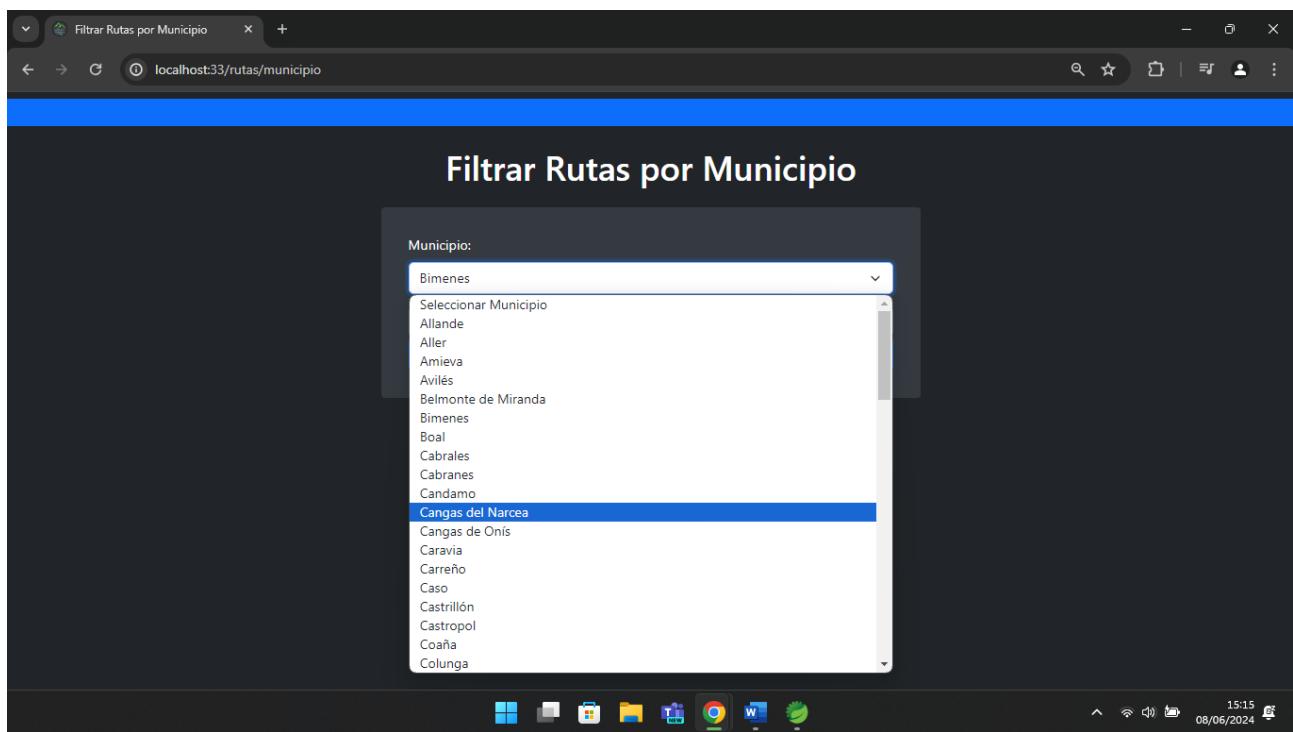
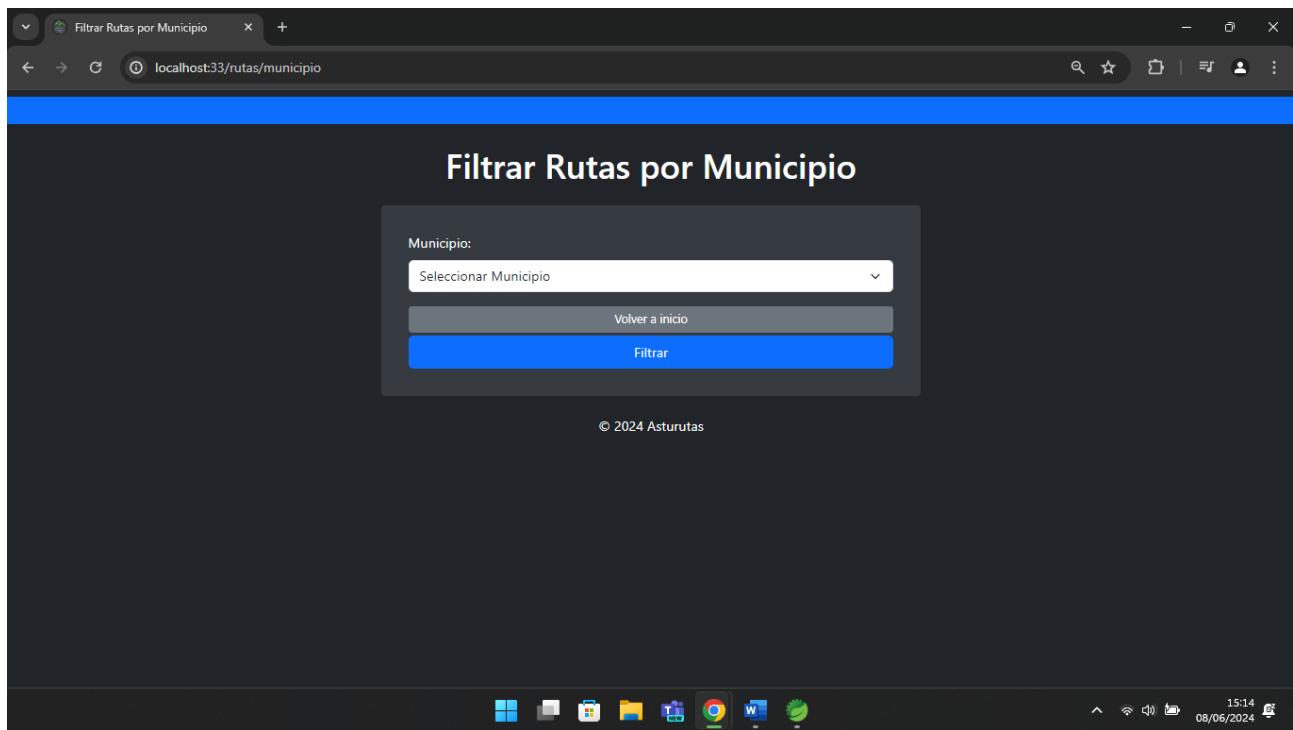
Pantalla Inicial



Esta es la pantalla principal que se encuentra el usuario al entrar en la aplicación. Como dijimos previamente en los casos de uso, el usuario puede hacer diferentes tareas sin la necesidad de haberse dado de alta con anterioridad. En el menú de la parte superior aparecen los diferentes filtros. Se puede filtrar por municipio, por actividad, por si son válidas para personas con movilidad reducida, por si son aptas para llevar perros o sin aptas para llevar niños pequeños. Vamos a ir uno a uno mostrando como se ven.



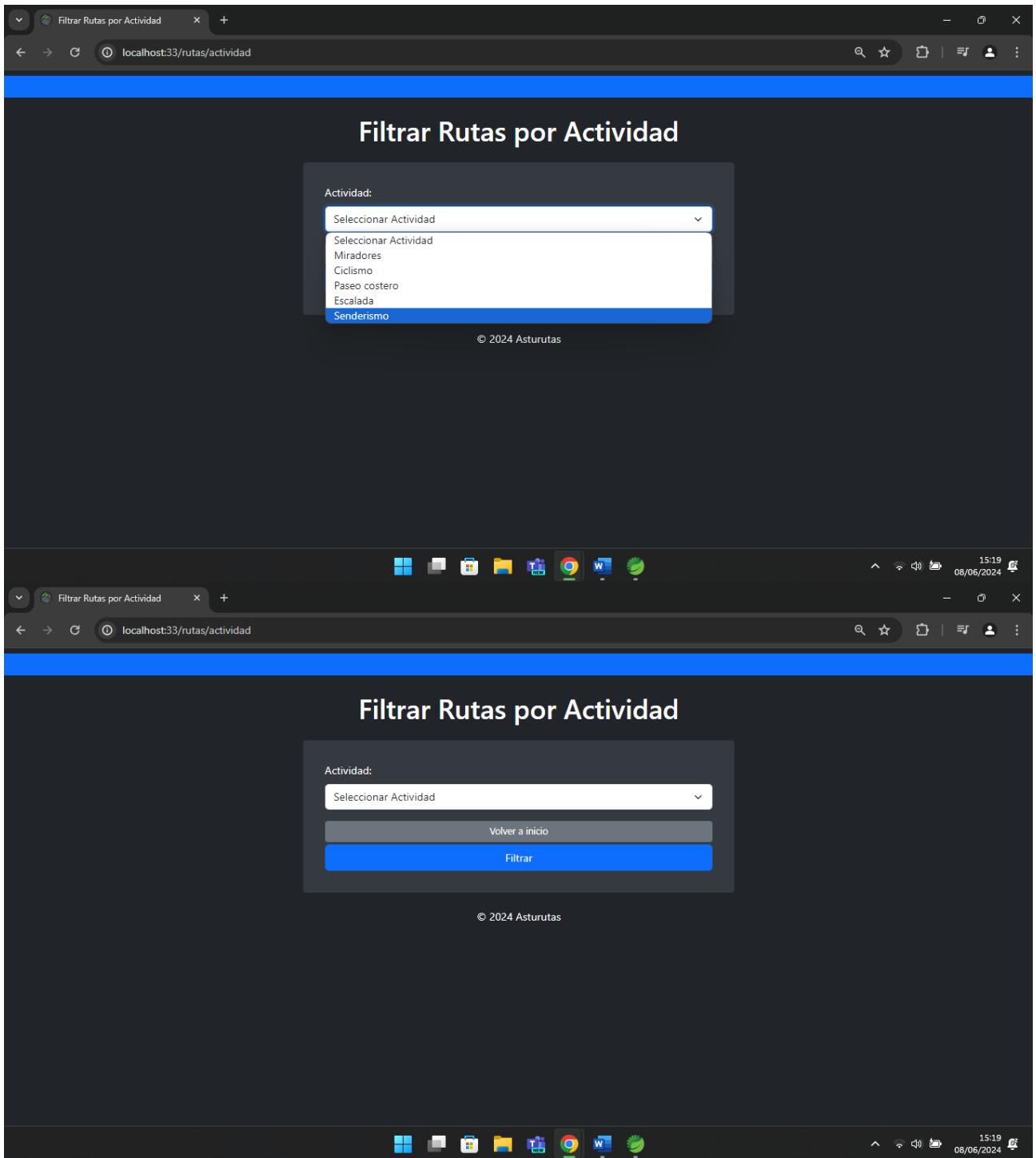
Filtro por municipio



Como su propio nombre indica esta página es la que usaremos para filtrar las rutas por municipio, podremos seleccionar la ruta en el desplegable o bien volver al inicio en caso de que nos hayamos confundido con el filtro.



Filtrar por actividad



The screenshots show a user interface for filtering routes by activity. The top screenshot displays a dropdown menu with the following options:

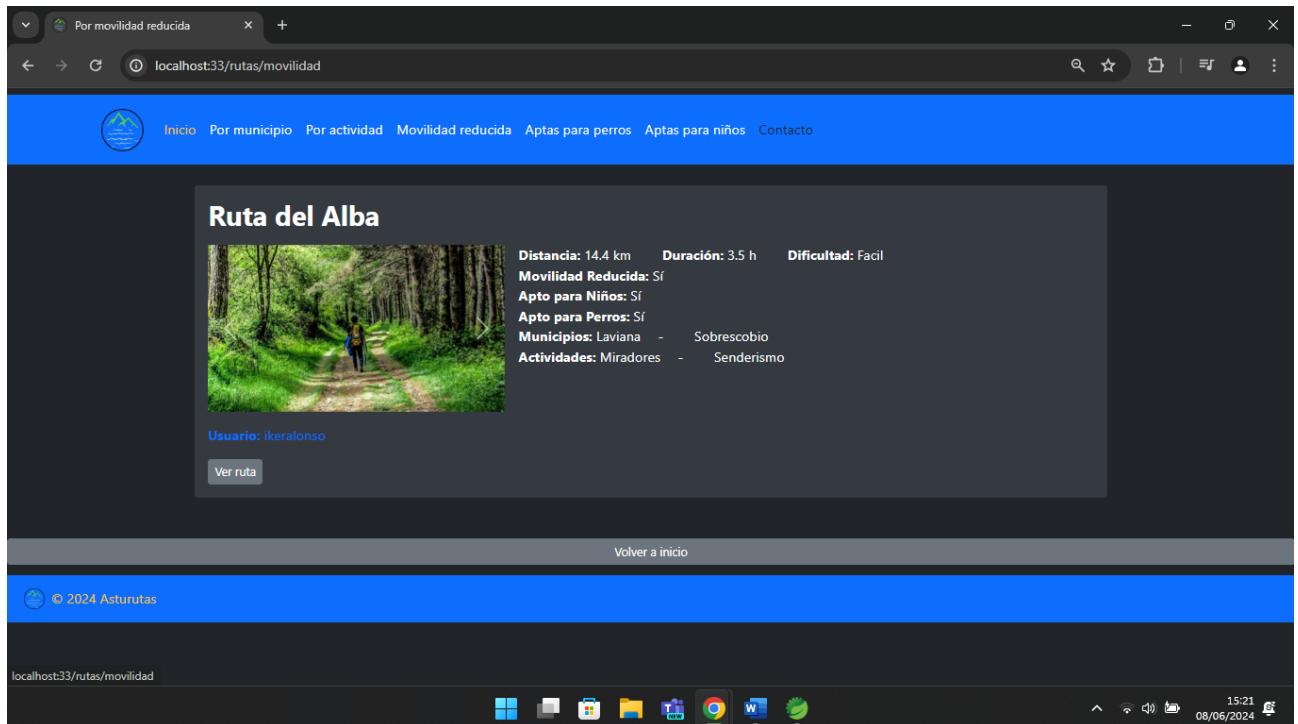
- Seleccionar Actividad
- Miradores
- Ciclismo
- Paseo costero
- Escalada
- Senderismo**

The bottom screenshot shows the same interface after a selection has been made. The 'Filtrar' button is highlighted in blue, indicating it is the active or next step.

Al igual que el filtro anterior, este tambien se selecciona con un desplegable. Podemos seleccionar la actividad que deseemos o volver a inicio, en caso de que hayamos hecho click en el filtro equivocado.



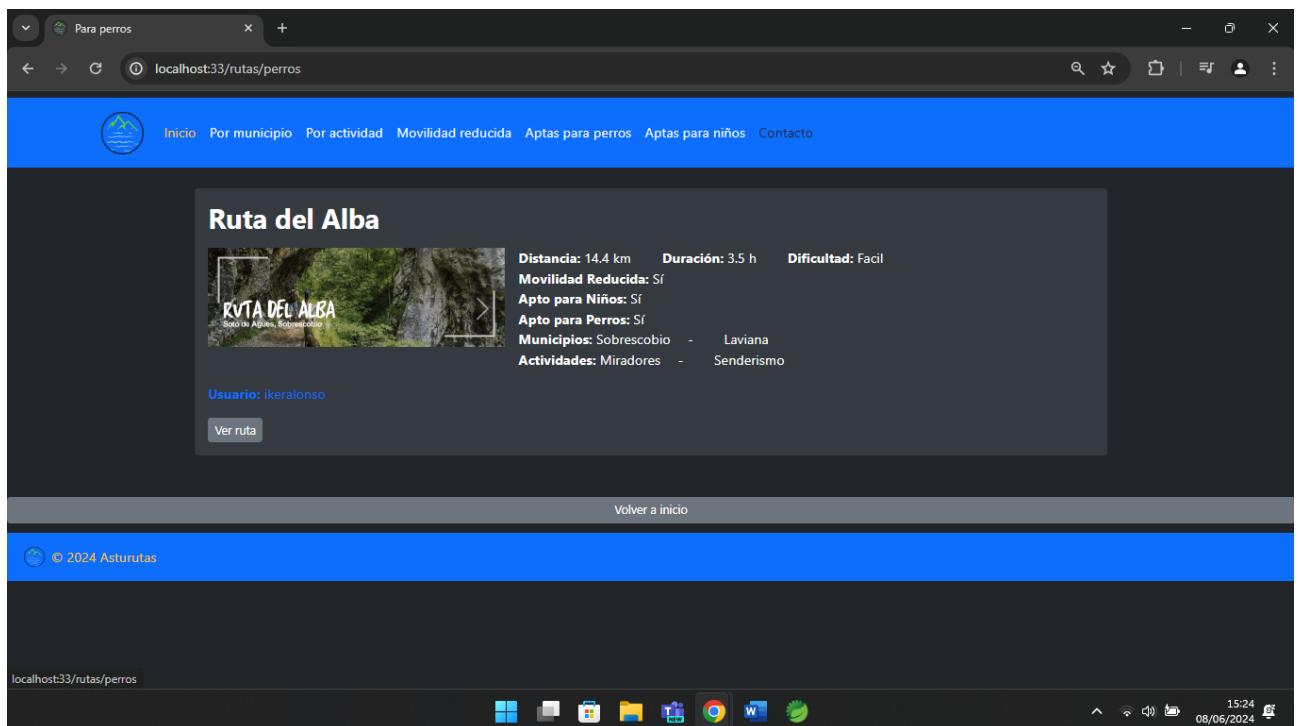
Filtrar por si es apta para movilidad reducida



The screenshot shows a web browser displaying a route page titled "Ruta del Alba". The page includes a photo of a person walking on a path through a forest. Key details listed are: Distancia: 14.4 km, Duración: 3.5 h, Dificultad: Facil, Movilidad Reducida: Sí, Apto para Niños: Sí, Apto para Perros: Sí, Municipios: Laviana - Sobrescobio, and Actividades: Miradores - Senderismo. Below the details, it says "Usuario: ikeralonso" and has a "Ver ruta" button. At the bottom, there's a "Volver a inicio" link and a copyright notice "© 2024 Asturutas". The browser's address bar shows "localhost:33/rutas/movilidad".

En este caso, al solo ser una opción de si o no, ya nos devuele directamente las rutas que tienen esta opción

Filtrar por si es apta para ir con perros

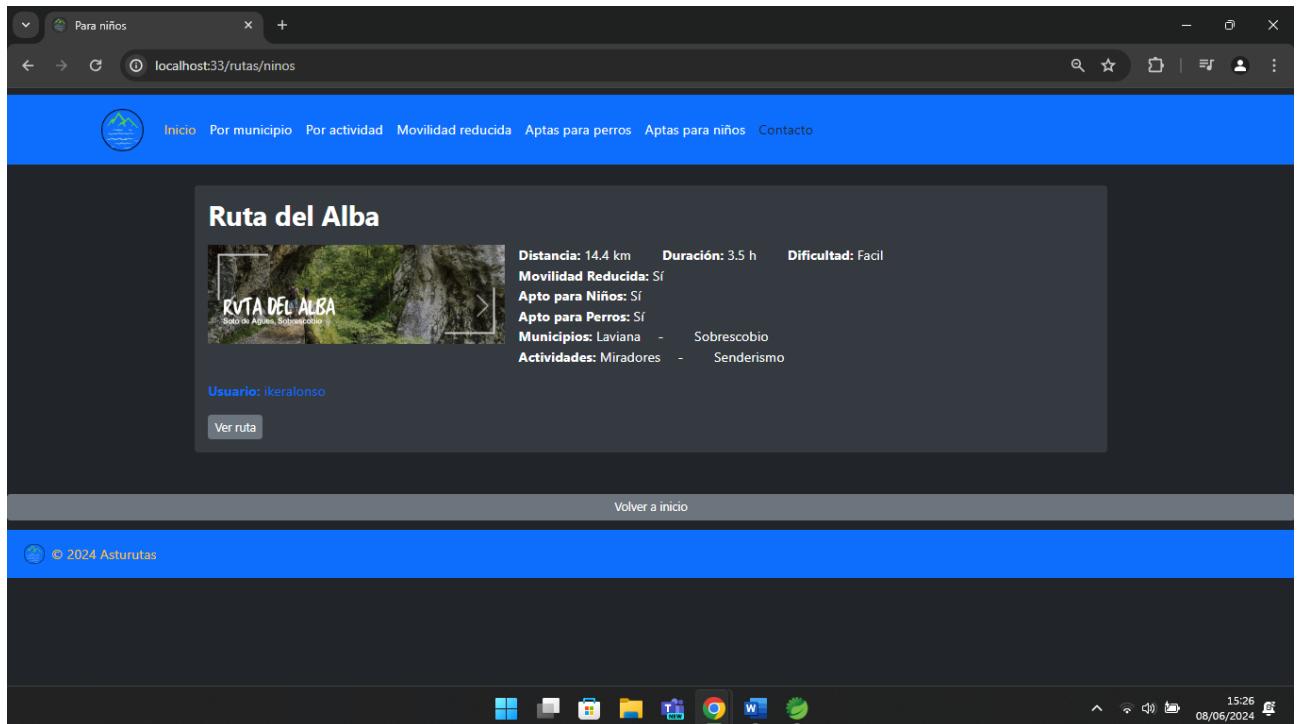


The screenshot shows a web browser displaying a route page titled "Ruta del Alba". The page includes a photo of a path through a forest. Key details listed are: Distancia: 14.4 km, Duración: 3.5 h, Dificultad: Facil, Movilidad Reducida: Sí, Apto para Niños: Sí, Apto para Perros: Sí, Municipios: Sobrescobio - Laviana, and Actividades: Miradores - Senderismo. Below the details, it says "Usuario: ikeralonso" and has a "Ver ruta" button. At the bottom, there's a "Volver a inicio" link and a copyright notice "© 2024 Asturutas". The browser's address bar shows "localhost:33/rutas/perros".

Al igual que el filtro anterior, como no hay opciones para seleccionar, cuando hagamos click ya recibiremos las rutas que cumplan nuestros filtros

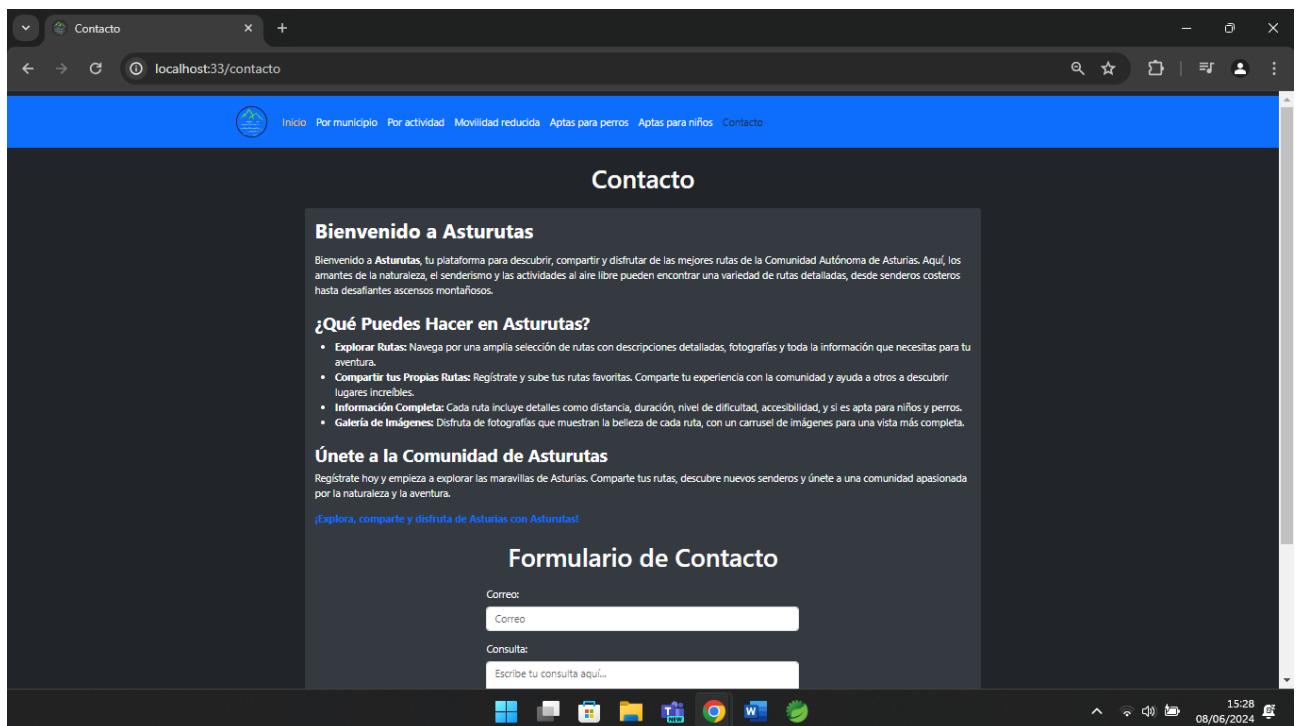


Filtrar por si es apta para ir con niños



Como las dos anteriores, solo tiene la posibilidad de si o no. Una vez hagamos click recibiremos las rutas que cumplan con esta característica.

Contacto



Esta parte de nuestra aplicación web ya no es ningun filtro. Nos da la posibilidad de enviar un mensaje a los desarrolladores para que puedan leer nuestras quejas, tanto si somos miembros de la página o no.



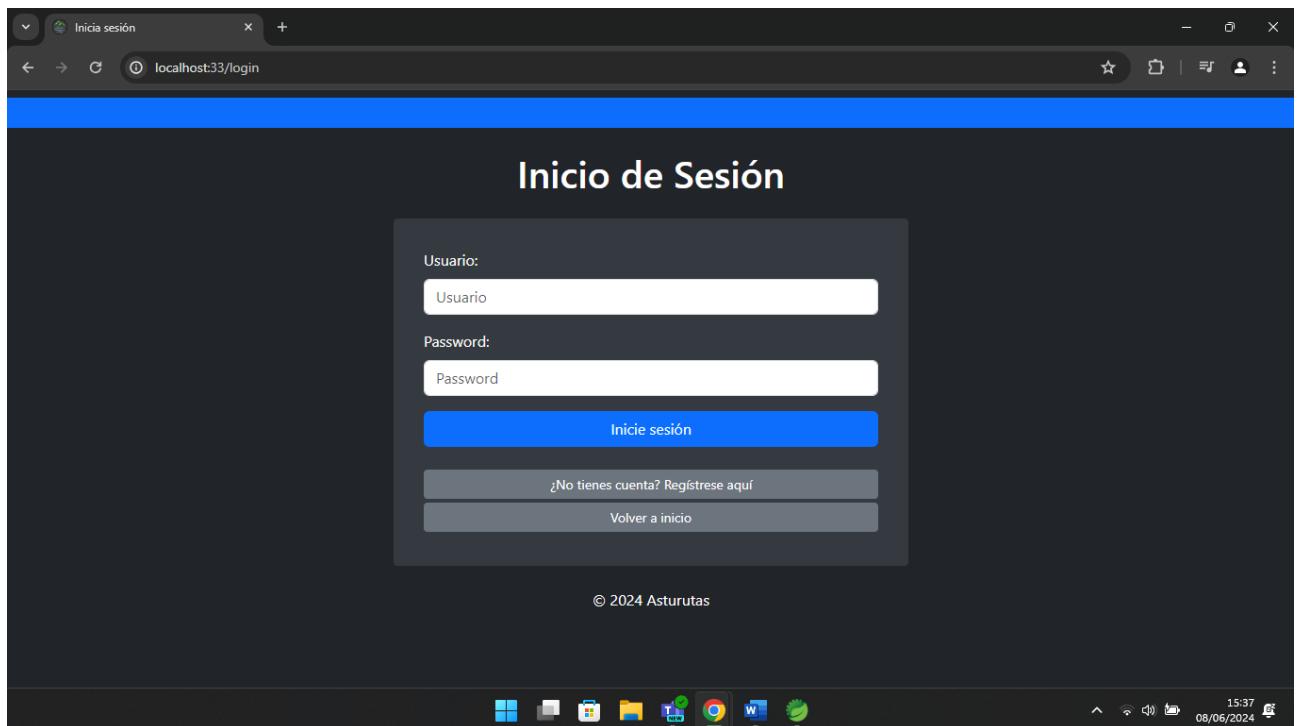
Registro de usuario

The screenshot shows a web browser window with a dark-themed registration form. The title bar reads "Registro de usuario" and the address bar shows "localhost:33/register". The main content area is titled "Registro de Usuario" and features a logo of green mountains and blue water. The form includes fields for "Nombre y Apellidos" (Name and Surname) containing "Nombre Apellido", "Identificador de Usuario" (User Identifier) containing "usuario33", "Dirección de Correo Electrónico" (Email Address) containing "nombre@ejemplo.com", "Contraseña" (Password) containing "Contraseña", and "Confirmar Contraseña" (Confirm Password) containing "Confirmar Contraseña". There is also a checkbox for "Mostrar Contraseñas" (Show Passwords). A large blue button at the bottom is labeled "Registrar mi Cuenta" (Register my Account). Below the button are links for "Volver a inicio" (Return to Home) and "¿No es tu primera vez? Inicia sesión" (Not your first time? Log in). The bottom right corner of the screen shows the date and time as "08/06/2024 15:31".

Si previamente no tienes un usuario, y a parte de leer rutas también quieres subir tus propias rutas, necesitas crear una cuenta. Para ello, deberás rellenar este formulario. También tienes debajo del botón de registrar la oportunidad de volver a inicio, en caso de que te hayas arrepentido y no quieras crear una cuenta, o bien el de iniciar sesión en caso de que hayas recordado que tenías una cuenta.



Inicio de sesión

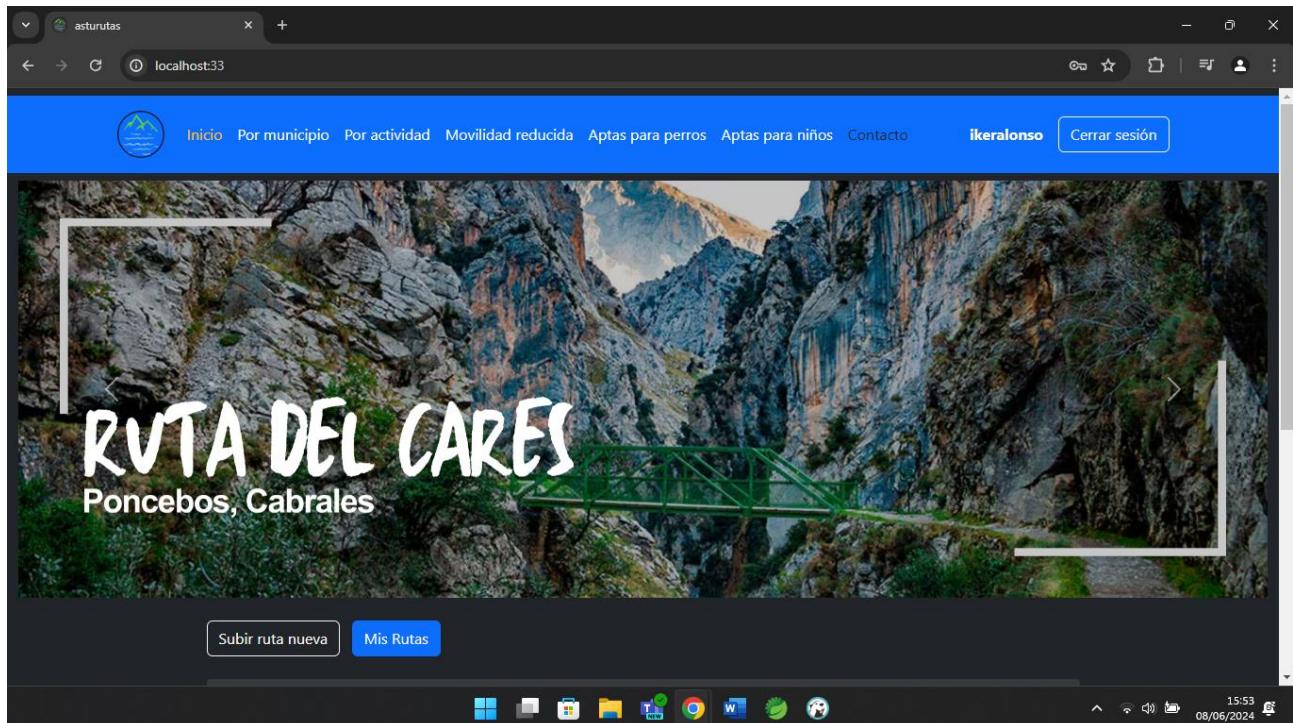


Menos apartoso que el de registro, este es el usuario que debes llenar para iniciar sesión en caso de que ya tengas una cuenta. Si no tienes cuenta, puedes acceder directamente al formulario de registro desde aquí o bien, también puedes volver a inicio.



Esto se muestra para el usuario que no está registrado, ¿pero ahora, que se muestra al usuario que ha iniciado sesión) Veremos que comparten muchas funcionalidades, pero como dijimos en los casos de uso, el usuario con cuenta tiene mas “privilegios”.

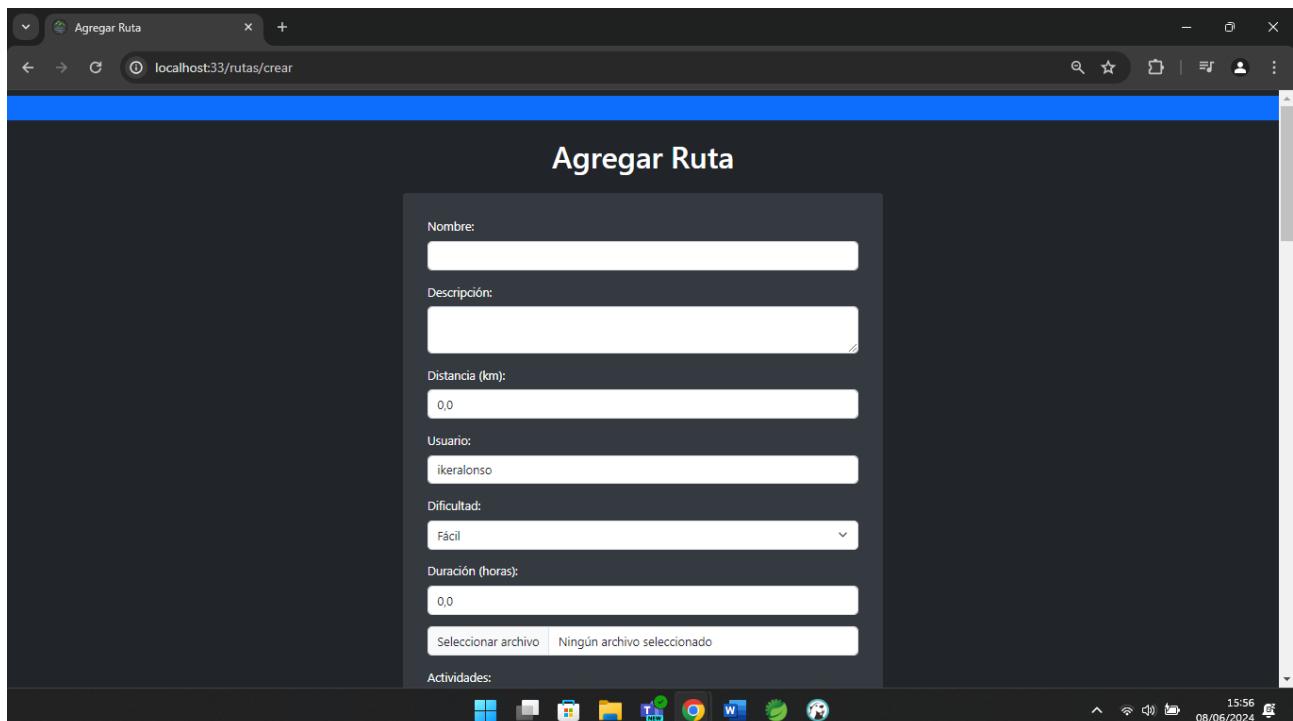
Pantalla Inicial



Como se puede ver, una vez hemos iniciado sesión, automáticamente nos redirige a la página principal. ¿Se pueden ver diferencias respecto a la anterior, verdad? Ahora que ya hemos iniciado sesión nos aparecen dos botones nuevos. En uno podemos subir una ruta nueva y en otro, podemos ver todas nuestras rutas. Vamos a verlos uno a uno.



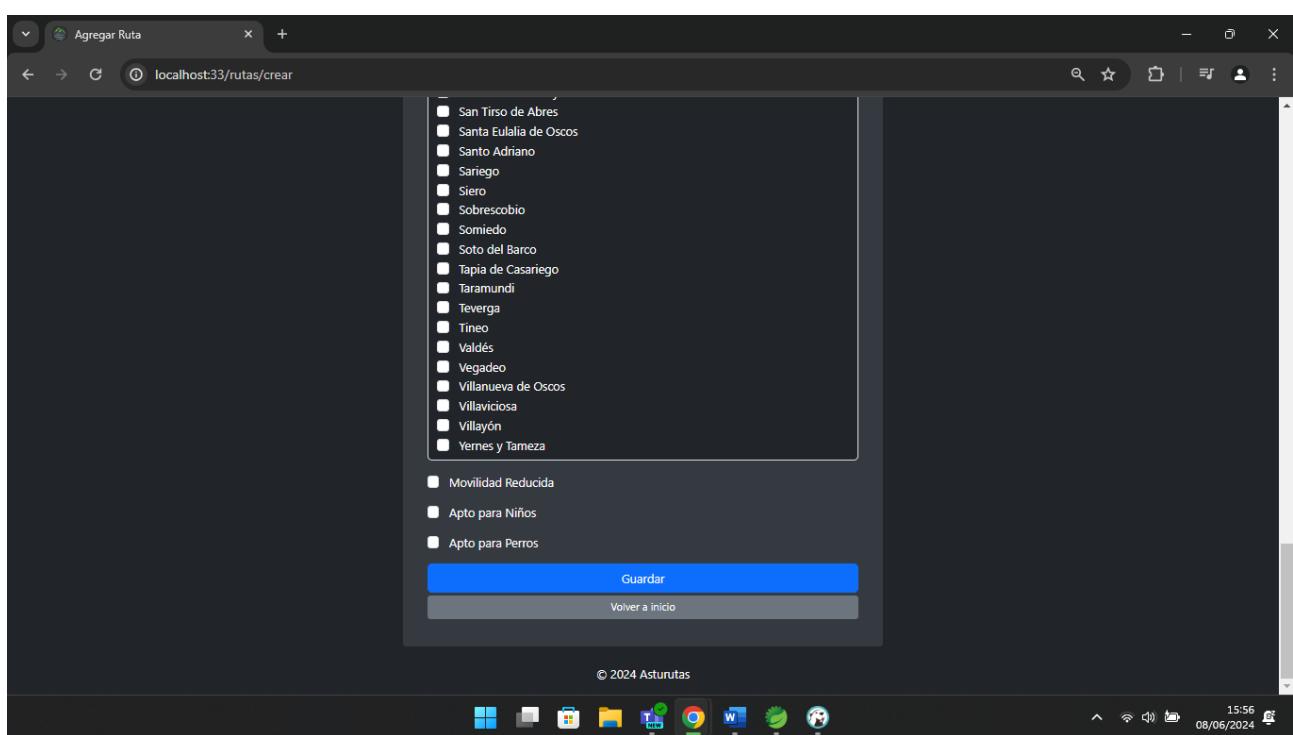
Subir ruta nueva



The screenshot shows a web browser window titled "Agregar Ruta" with the URL "localhost:33/rutas/crear". The page has a dark theme with blue header bars. It contains a form with the following fields:

- Nombre: (text input)
- Descripción: (text input)
- Distancia (km): (text input with value "0.0")
- Usuario: (text input with value "ikeralonso")
- Dificultad: (dropdown menu with option "Fácil")
- Duración (horas): (text input with value "0.0")
- Actividades: (checkboxes list)
 - San Tirso de Abres
 - Santa Eulalia de Oscos
 - Santo Adriano
 - Sariego
 - Siero
 - Sobrescobio
 - Somiedo
 - Soto del Barco
 - Tapia de Casariego
 - Taramundi
 - Teverga
 - Tineo
 - Valdés
 - Vegadeo
 - Villanueva de Oscos
 - Villaviciosa
 - Villayón
 - Yernes y Tameza
- Movilidad Reducida
- Apto para Niños
- Apto para Perros

At the bottom are two buttons: "Guardar" (blue background) and "Volver a inicio" (grey background).



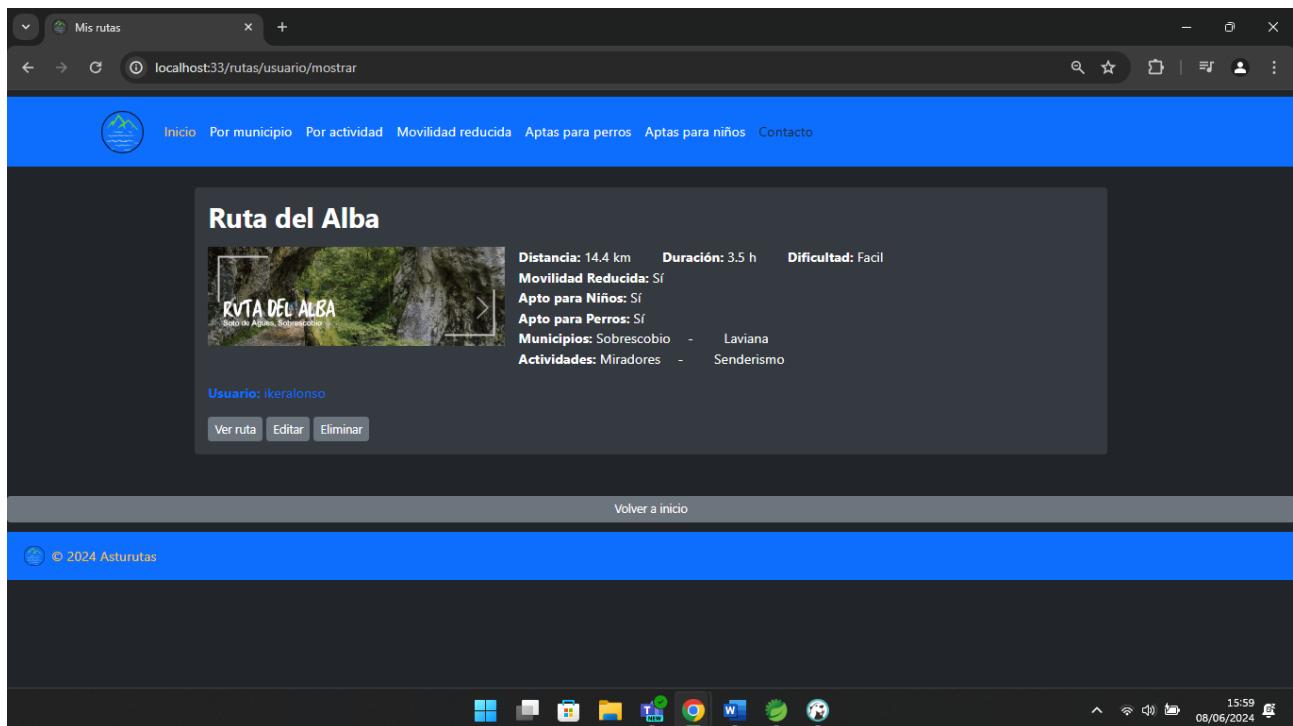
The second screenshot shows the same form after some checkboxes have been selected. The "Actividades" section now includes:

- San Tirso de Abres
- Santa Eulalia de Oscos
- Santo Adriano
- Sariego
- Siero
- Sobrescobio
- Somiedo
- Soto del Barco
- Tapia de Casariego
- Taramundi
- Teverga
- Tineo
- Valdés
- Vegadeo
- Villanueva de Oscos
- Villaviciosa
- Villayón
- Yernes y Tameza
- Movilidad Reducida
- Apto para Niños
- Apto para Perros

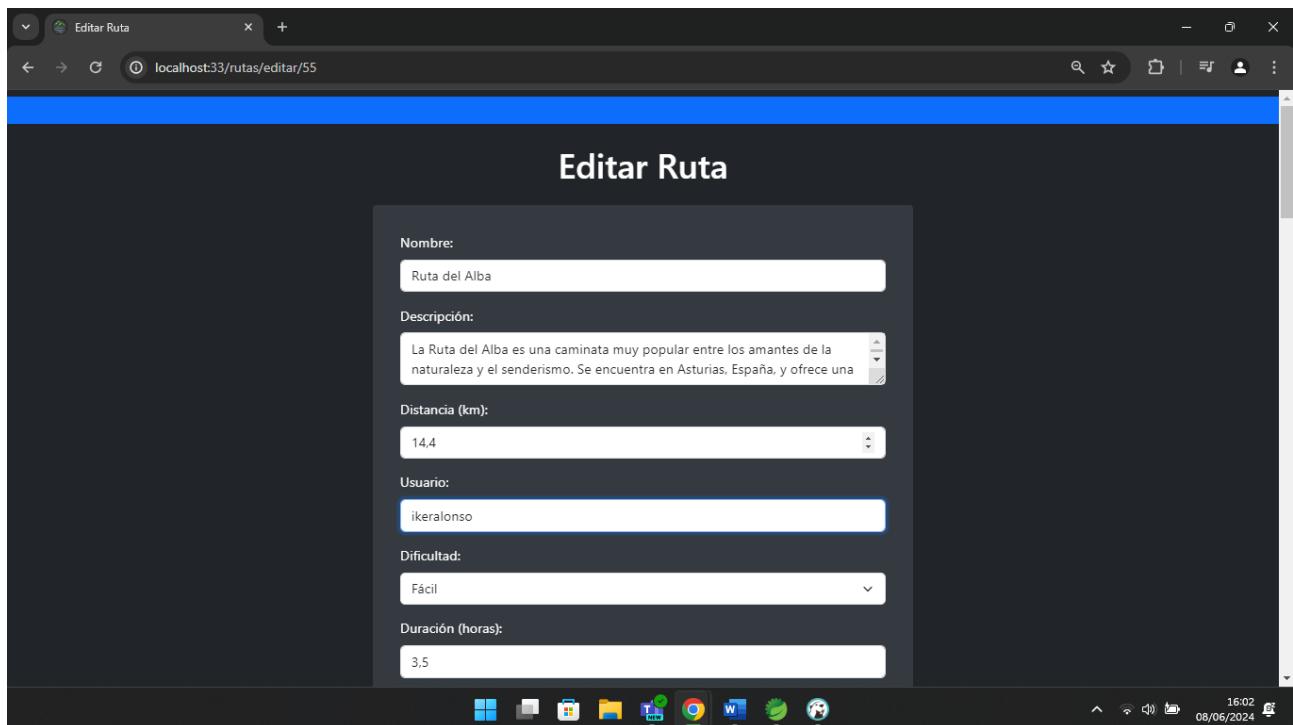
Con este formulario podremos subir una ruta nueva a la aplicación web, tan solo tenemos que rellenar los campos según deseemos. Debajo de guardar, tenemos otro botón para volver al inicio en caso de que nos hayamos equivocado y no queramos subir una ruta.



Mostrar mis rutas

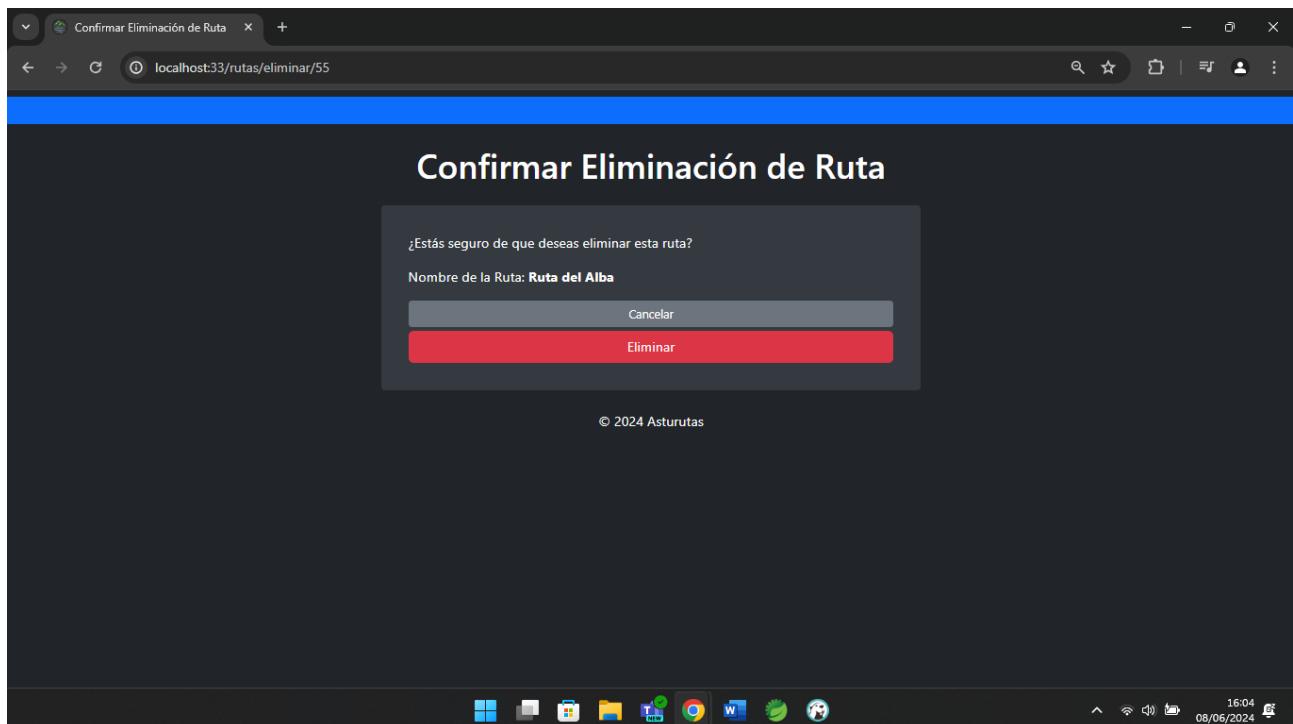


Si queremos ver todas aquellas rutas que hemos subido, hacemos click en mostrar mis rutas. Como se puede ver, a parte del botón ver ruta que nos aparecía antes, también nos aparece uno que es “editar” y otro que es “eliminar”. Como su propio nombre dice uno edita la ruta y otro la borra. Veamos cual es la apariencia visual de cada botón



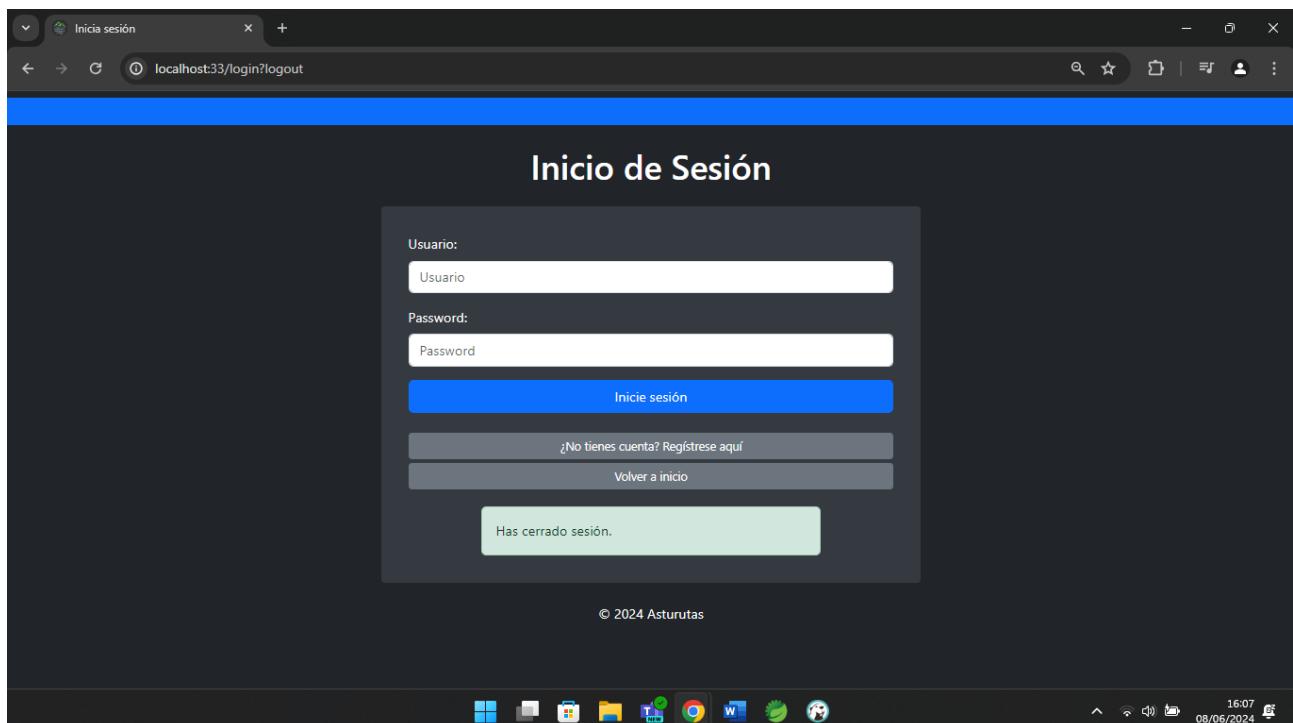
Al hacer click en editar la ruta nos rellena las entradas con los datos que hemos enviado cuando creamos la ruta o bien cuando la editamos por ultima vez. Nos da la opción de editarla, evidentemente, o bien volver atrás en caso de que nos hayamos arrepentido y no queramos editarla





También podemos borrar la ruta. Si hacemos click en borrar esta ruta nos aparecerá esta pantalla, en la que podremos confirmar que queremos borrar la ruta o bien que queremos volver atrás porque ya no la queremos borrar.

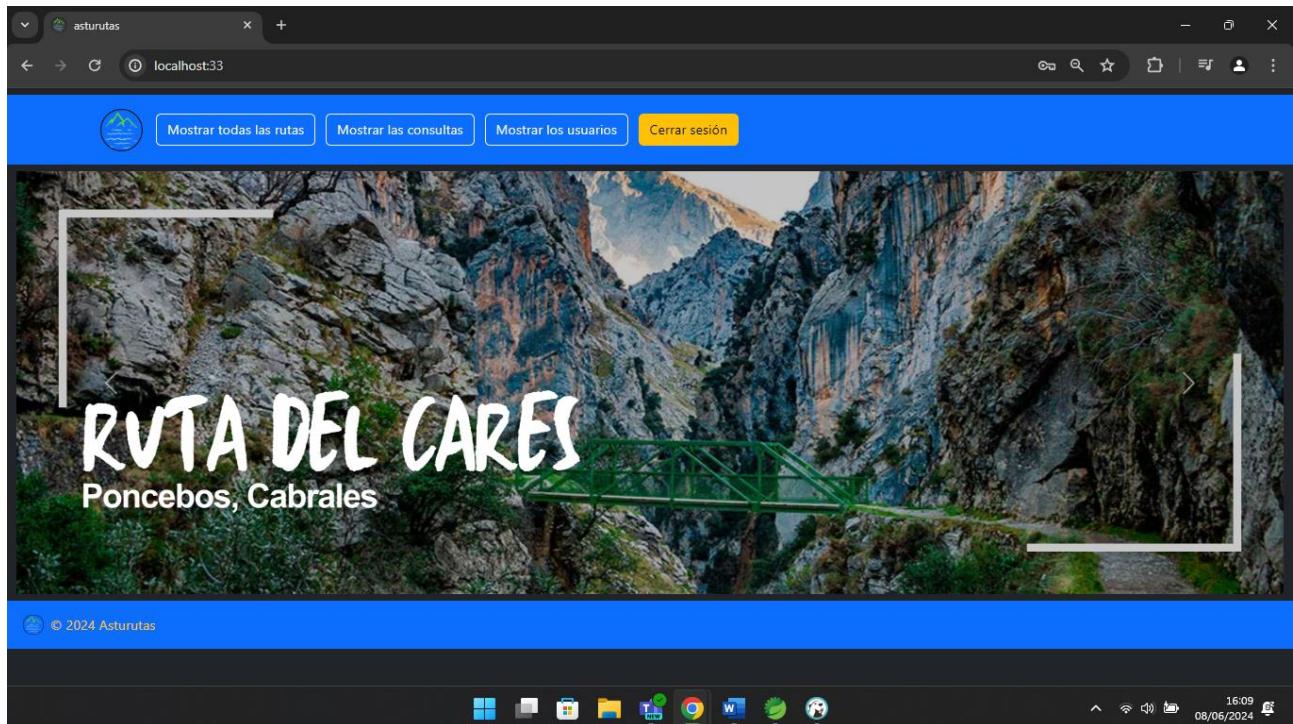
¿Hay alguna diferencia más en la página inicial? ¡Sí! Evidentemente también tenemos que tener la opción de cerrar sesión.



Una vez hagamos click a cerrar sesión, volvemos al login y nos muestra un mensaje. A partir de aquí podemos volver a movernos a donde deseemos.

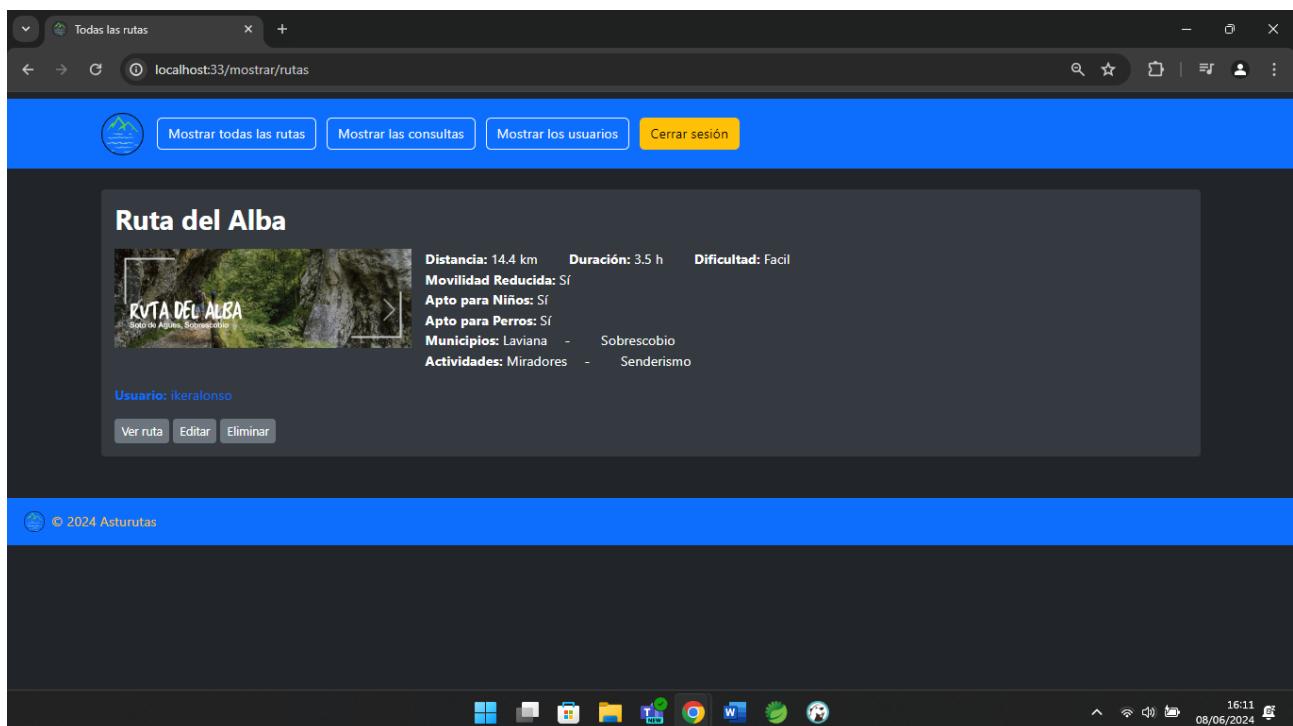


Ahora bien, ¿y el administrador? ¿Cómo se ve la aplicación para él?



Como se puede ver, la apariencia no cambia en absoluto, pero si que cambia el menú, y mucho. El administrador o los administradores de la aplicación, pueden ver todas las rutas, todas las consultas, y todos los usuarios. Vayamos funcionalidad a funcionalidad. Por supuesto también puede cerrar sesión.

Mostrar todas las rutas



Si hacemos click en mostrar todas las rutas, pues nos devuelve todas las rutas



Mostrar todas las consultas

The screenshot shows a web browser window with a blue header bar. The header contains a logo, four buttons labeled 'Mostrar todas las rutas', 'Mostrar las consultas', 'Mostrar los usuarios', and 'Cerrar sesión', and a search bar. Below the header, there are two dark grey rectangular boxes, each labeled 'Consulta'. The first box contains the text: 'Correo: correo@prueba.es', 'Mensaje: fawegbv agn', and a checkbox labeled 'Leído' which is unchecked. The second box contains the text: 'Correo: hqch@gmail.com', 'Mensaje: CQAFVAVS', and a checkbox labeled 'Leído' which is checked. At the bottom of the page, there is a blue footer bar with the text '© 2024 Asturutas' and a system tray with various icons.

Aquí vemos todos los mensajes que dejan los usuarios en la pestaña contacto de nuestra aplicación web.

Mostrar todos los usuarios

The screenshot shows a web browser window with a blue header bar. The header contains a logo, four buttons labeled 'Mostrar todas las rutas', 'Mostrar las consultas', 'Mostrar los usuarios', and 'Cerrar sesión', and a search bar. Below the header, there are two dark grey rectangular boxes, each labeled 'Usuario'. The first box contains the text: 'Nombre: admin', 'Usuario: admin', 'Email: admin@asturutas.com', and 'Fecha de Creación:'. Below this text is a red 'Eliminar' button. The second box contains the text: 'Nombre: Iker Alonso García', 'Usuario: ikeralonso', 'Email: ikeralonso@gmail.com', and 'Fecha de Creación:'. Below this text is a red 'Eliminar' button. At the bottom of the page, there is a blue footer bar with the text '© 2024 Asturutas' and a system tray with various icons.

Aquí podemos ver cuales son todos los usuarios que están dados de alta en nuestra aplicación, y también podremos borrar en caso de que hagan algún uso indebido de la misma

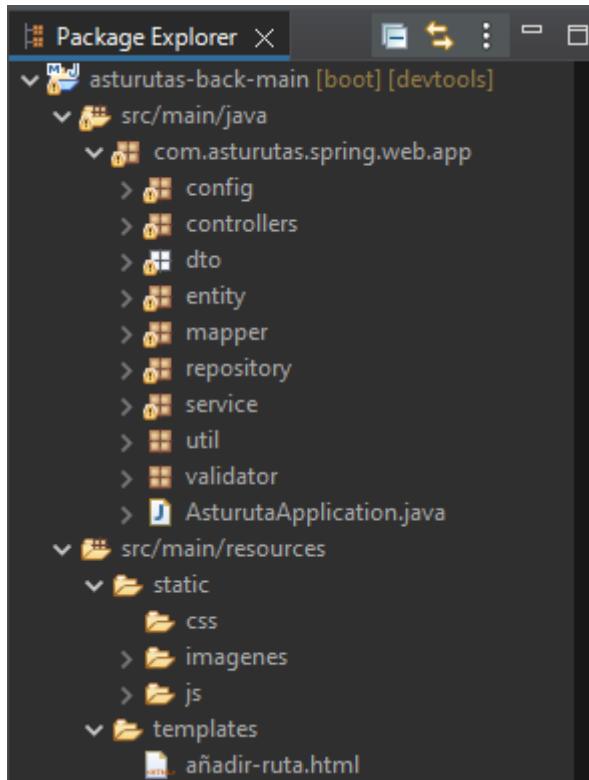


Capa de negocio o lógica de la aplicación

Ahora bien, ¿para que esto se vea así, que se ha hecho por detrás?

Empecemos primero por el patrón de diseño utilizado, en este caso Spring MVC (Modelo Vista Controlador).

Este es la estructura de los paquetes de la aplicación:

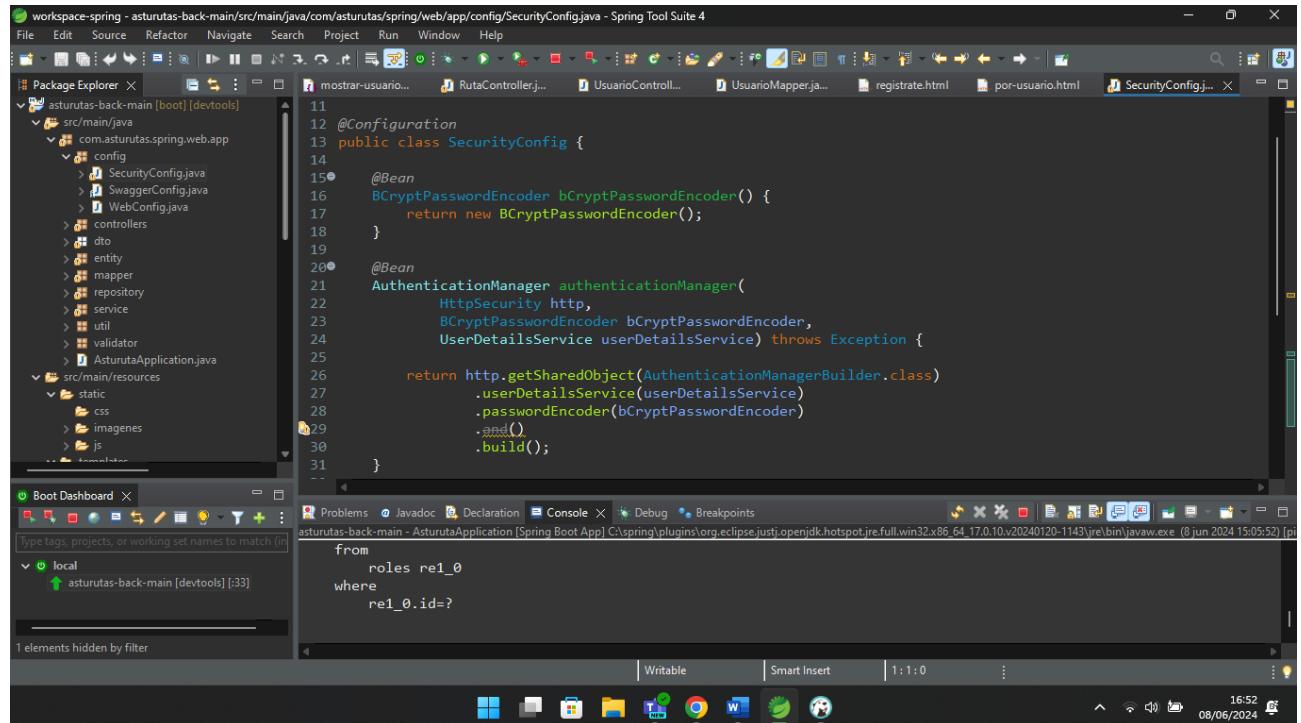


- Config: Es donde se guarda el archivo de seguridad SecurityConfig, dividiendo los permisos según los roles. Además también se guardan algunos conversores para guardar los datos que enviamos mediante los formularios.
- Controllers: Ahí guardamos las clases que se comunican con el front.
- Dto: Para no trabajar directamente sobre las entidades (que son las que se utilizarán para realizar el mapeo), creamos unos dto que nos mostrarán lo que queremos de cada entidad.
- Entity: Son las clases de toda la vida, son las que utilizamos para comunicarnos con la BBDD y para mapear entre el resto de las mismas.
- Mapper: Utilizamos mapper para mapear de entity a dto y viceversa sin tener que hacerlo cada vez que lo necesitemos. Al contener interfaces, solo tenemos que invocar a la interfaz, nos ahorraremos cientos y cientos de líneas al finalizar, además de tiempo.
- Repository: Extendiendo del JpaRepository se utiliza para el CRUD básico además de para añadir nuestras propias Query personalizadas.
- Service: Es donde se trabaja con nuestros repositorios. Se crean como interfaces y posteriormente, dentro de este paquete creamos uno que sea impl donde desarrollaremos toda la lógica de cada método



Vayamos paquete a paquete haciendo una breve explicación del funcionamiento de cada clase / interfaz.

Config

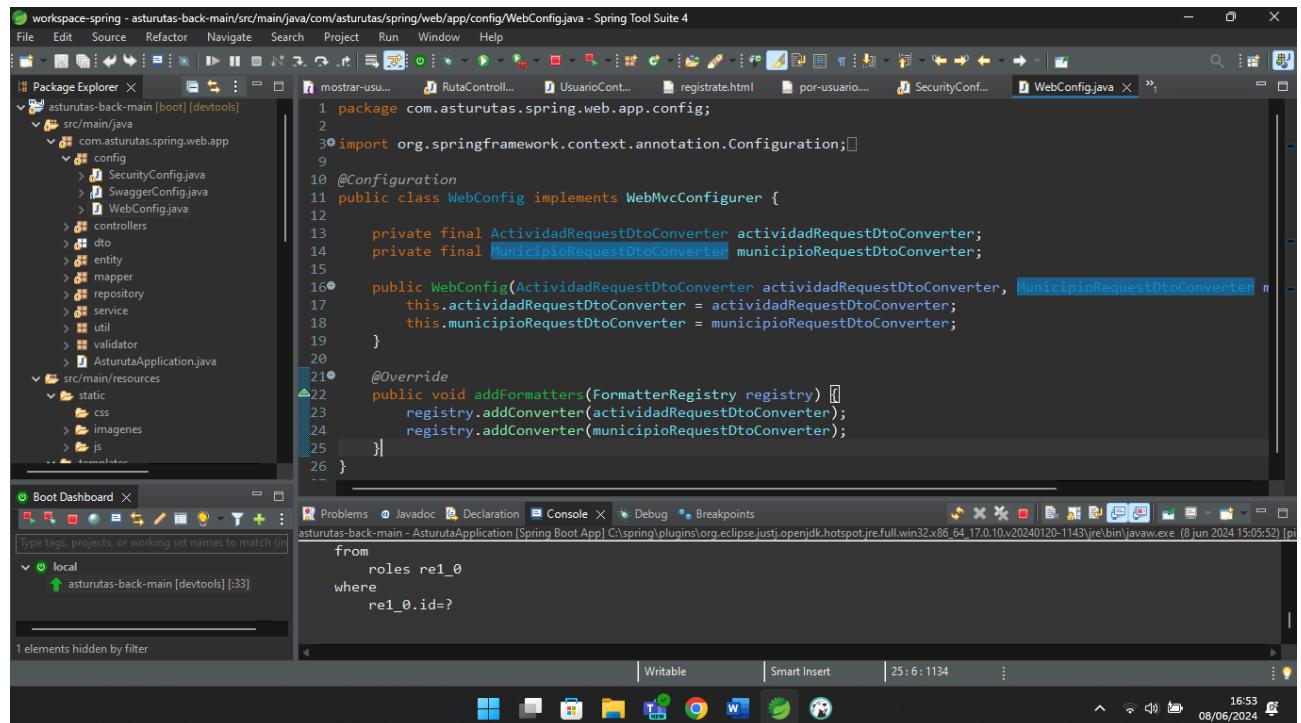


```

11  @Configuration
12  public class SecurityConfig {
13
14      @Bean
15      BCryptPasswordEncoder bCryptPasswordEncoder() {
16          return new BCryptPasswordEncoder();
17      }
18
19      @Bean
20      AuthenticationManager authenticationManager(
21          HttpSecurity http,
22          BCryptPasswordEncoder bCryptPasswordEncoder,
23          UserDetailsService userDetailsService) throws Exception {
24
25          return http.getSharedObject(AuthenticationManagerBuilder.class)
26              .userDetailsService(userDetailsService)
27              .passwordEncoder(bCryptPasswordEncoder)
28              .and()
29              .build();
30      }
31  }

```

Estos beans son los encargados de cifrar la clave una vez se envía en el formulario desde el front.



```

1 package com.asturutas.spring.web.app.config;
2
3 import org.springframework.context.annotation.Configuration;
4
5 @Configuration
6 public class WebConfig implements WebMvcConfigurer {
7
8     private final ActividadRequestDtoConverter actividadRequestDtoConverter;
9     private final MunicipioRequestDtoConverter municipioRequestDtoConverter;
10
11     public WebConfig(ActividadRequestDtoConverter actividadRequestDtoConverter, MunicipioRequestDtoConverter municipioRequestDtoConverter) {
12         this.actividadRequestDtoConverter = actividadRequestDtoConverter;
13         this.municipioRequestDtoConverter = municipioRequestDtoConverter;
14     }
15
16     @Override
17     public void addFormatters(FormatterRegistry registry) {
18         registry.addConverter(actividadRequestDtoConverter);
19         registry.addConverter(municipioRequestDtoConverter);
20     }
21 }

```

Esta clase incorpora los métodos necesarios para convertir un tipo de datos String al tipo de Datos deseado que en este caso es un dto.



Controllers

```

workspace-spring - asturutas-back-main/src/main/java/com/asturutas/spring/web/app/controllers/RutaController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X
asturutas-back-main [boot] [devtools]
src/main/java
  com.asturutas.spring.web.app
    config
    controllers
      ActividadController.java
      ConsultaController.java
      MunicipioController.java
      RutaController.java
      UsuarioController.java
    dto
    entity
    mapper
    repository
    service
    util
    validator
  AsturutaApplication.java
src/main/resources
  static
    css
    images
Boot Dashboard X
Type tags, projects, or working set names to match (in)
  local
asturutas-back-main [devtools] [33]
Problems Javadoc Declaration Console X Debug Breakpoints
asturutas-back-main - AsturutaApplication [Spring Boot App] C:\spring\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64.17.0.10.v20240120-1143\jre\bin\javaw.exe (8 jun 2024 15:05:52) [p]
from
  roles re1_0
where
  re1_0.id=?

```

“Controlan” el intercambio de información entre el front-end y el back-end. Por ejemplo:

En el método findAll() hacemos una petición GET para obtener todas las rutas que tenemos almacenadas en la base de datos. En el mapping ponemos la ruta que deseamos. Posteriormente creamos un array de objetos RutaResponseDto donde almacenamos todas las rutas que obtenemos gracias al método del service que creamos con anterioridad. Se hablará de su lógica en las siguientes páginas. Añadimos el atributo al modelo y retornamos la vista, en este caso la del index.

```

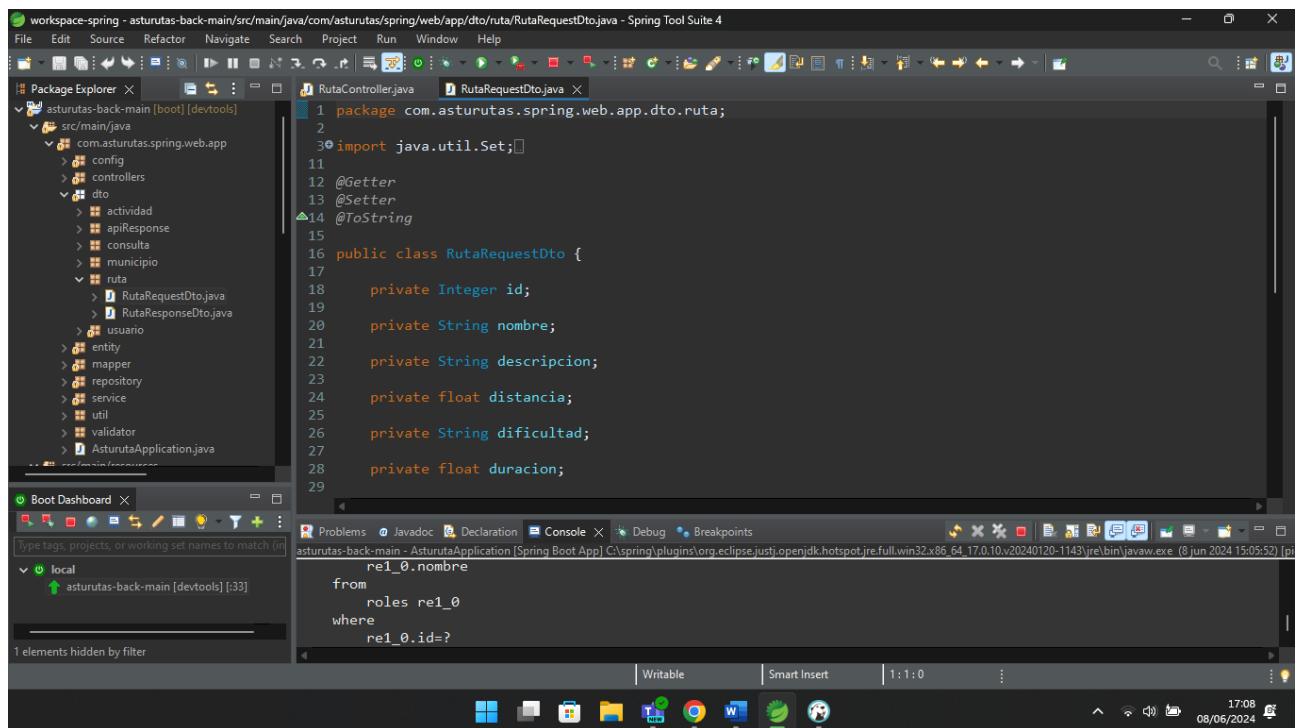
workspace-spring - asturutas-back-main/src/main/java/com/asturutas/spring/web/app/controllers/RutaController.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer X
asturutas-back-main [boot] [devtools]
src/main/java
  com.asturutas.spring.web.app
    config
    controllers
      ActividadController.java
      ConsultaController.java
      MunicipioController.java
      RutaController.java
      UsuarioController.java
    dto
    entity
    mapper
    repository
    service
    util
    validator
  AsturutaApplication.java
src/main/resources
  static
    css
    images
Boot Dashboard X
Type tags, projects, or working set names to match (in)
asturutas-back-main [devtools] [33]
Problems Javadoc Declaration Console X Debug Breakpoints
asturutas-back-main - AsturutaApplication [Spring Boot App] C:\spring\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64.17.0.10.v20240120-1143\jre\bin\javaw.exe (8 jun 2024 15:05:52) [p]
from
  roles re1_0
where
  re1_0.id=?

```

Este es la lógica que empleamos en el controlador para almacenar la ruta. Añadimos el rutaRequest, que será donde se guardarán todos los datos que introduzcamos, y añadimos también las actividades y los municipios, esto es lo que hace que los desplegables del formulario de insertar una ruta nueva no se muestren vacíos. Una vez enviado, y en la ruta que hayamos puesto en el action (tiene que coincidir con el PostMapping) tan solo tenemos que llamar al service para que nos guarde la ruta que enviamos. El funcionamiento de todos los métodos es muy parecido, porque utiliza métodos del CRUD básico.



Dto



```
workspace-spring - asturutas-back-main/src/main/java/com/asturutas/spring/web/app/dto/RutaRequestDto.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer RutaController.java RutaRequestDto.java
1 package com.asturutas.spring.web.app.ruta;
2
3 import java.util.Set;
4
5 @Getter
6 @Setter
7 @ToString
8
9 public class RutaRequestDto {
10
11     private Integer id;
12
13     private String nombre;
14
15     private String descripcion;
16
17     private float distancia;
18
19     private String dificultad;
20
21     private float duracion;
22
23 }
asturutas-back-main - AsturutaApplication [Spring Boot App] C:\spring\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32-x86_64.17.0.10.v20240120-1143\re\bin\javaw.exe (8 jun 2024 15:05:52) [pid: 1144]
asturutas-back-main [devtools] [33]
rel_0.nombre
from
roles rel_0
where
rel_0.id=?
```

Para ahorrar líneas de código se ha utilizado una librería de JAVA llamada loombok que nos ahorra tener que generar los getters, los setters y los to string de todos los dto y todas las entidades de nuestra aplicación. El dto tiene los atributos que nosotros queremos que reciba y devuelva, no es necesario que tenga los mismos que su entidad.



Entidades (O bien modelos)

```

workspace-spring - asturutas-back-main/src/main/java/com/asturutas/spring/web/app/entity/RutaEntity.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer RutaController.java RutaRequestDto.java RutaEntity.java
src/main/java com.asturutas.spring.web.app
  config controllers dto entity
    ActividadEntity.java ConsultaEntity.java MunicipioEntity.java RolEntity.java RutaEntity.java UsuarioEntity.java
    mapper repository service util validator
  AsturutaApplication.java
src/main/resources static
Boot Dashboard
Problems Javadoc Declaration Console Debug Breakpoints
asturutas-back-main-AsturutaApplication [Spring Boot App] C:\spring\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\ire\bin\javaw.exe (8 jun 2024 15:05:52) [pi
  re1_0.nombre
  from
  roles re1_0
  where
  re1_0.id=?
  Writable Smart Insert 1:1:0
  17:13 08/06/2024
  
```

Como dijimos previamente son las que se ocupan de la conexión con la base de datos y el mapeo entre ellas mismas. Agregando la anotación `@entity` le decimos a Spring que esa clase es una entidad y con `@table` indicamos a que tabla de nuestra bbdd corresponde.

```

workspace-spring - asturutas-back-main/src/main/java/com/asturutas/spring/web/app/entity/RutaEntity.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Search Project Run Window Help
Package Explorer RutaController.java RutaRequestDto.java RutaEntity.java
src/main/java com.asturutas.spring.web.app
  config controllers dto entity
    ActividadEntity.java ConsultaEntity.java MunicipioEntity.java RolEntity.java RutaEntity.java UsuarioEntity.java
    mapper repository service util validator
  AsturutaApplication.java
src/main/resources static
Boot Dashboard
Problems Javadoc Declaration Console Debug Breakpoints
asturutas-back-main-AsturutaApplication [Spring Boot App] C:\spring\plugins\org.eclipse.jdt.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\ire\bin\javaw.exe (8 jun 2024 15:05:52) [pi
  re1_0.nombre
  from
  roles re1_0
  where
  re1_0.id=?
  Writable Smart Insert 1:1:0
  17:17 08/06/2024
  
```

Aquí podemos ver como se realiza el mapeo entre diferentes entidades. Muchas rutas tienen muchas actividades y muchas rutas tienen muchos municipios, por lo que necesitamos una relación many to many. Para realizar esa relación agregamos la anotación `@ManyToMany`. En `@JoinTable` añadimos el nombre de la tabla en la BBDD



que contiene dicha relación, y en `joincolumns` e `inversejoincolumns` las columnas que corresponden a cada parte de la relación. Hacemos lo propio en ambos lados de la relación también la BBDD.

Mapper

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer (left):** Shows the project structure for "asturatas-back-main". The `RutaMapper.java` file is selected in the `src/main/java/com.asturatas.spring.web.app.mapper` package.
- RutaMapper.java Content:** The code defines a `RutaMapper` interface with methods for mapping `RutaEntity` objects from `RutaResponseDto` and `RutaRequestDto`.
- Boot Dashboard (bottom left):** Displays a summary of the application's status, including memory usage and database connections.
- Java Persistence Editor (bottom right):** Shows a query editor with the following DQL:

```
re1_.nombre  
from  
roles re1_0  
where  
re1_.id=?
```

Como dijimos antes, para pasar de dto a entidad o viceversa, podemos crear unos mapper para que lo haga por nosotros, nos ahorre cientos de lineas de codigo, y principalmente tiempo.

```
workspace-spring - asturatas-back-main/target/generated-sources/annotations/com/asturatas/spring/web/app/mapper/RutaMapperImpl.java - Spring Tool Suite 4
File Edit Source Refactor Navigate Project Run Window Help
Package Explorer X RutaMapper.java RutaMapperImpl.java X
src/main/resources
src/main/java
src/test/java
src/test/resources
JRE System Library [JavaSE-17]
Maven Dependencies
target/generated-sources/annotations
com.asturatas.spring.web.app.mapper
ActividadMapperImpl.java
ConsultaMapperImpl.java
MunicipioMapperImpl.java
RutaMapperImpl.java
UsuarioMapperImpl.java
RutaMapper.java
RutaMapperImpl.java
17     date = "2024-06-08T15:05:59+0200",
18     comments = "version: 1.5.5.Final, compiler: Eclipse JDT (IDE) 3.37.0.v20240215-1558, environment: Java 17"
19 }
20 @Component
21 public class RutaMapperImpl implements RutaMapper {
22
23     @Override
24     public RutaEntity responseDtoToEntity(int id, RutaResponseDto rutaResponseDto) {
25         if ( rutaResponseDto == null ) {
26             return null;
27         }
28
29         RutaEntity rutaEntity = new RutaEntity();
30
31         if ( rutaResponseDto != null ) {
32             rutaEntity.setActividades( actividadRequestDtoSetToActividadEntitySet( rutaResponseDto.getActividades() ) );
33             rutaEntity.setAptoNinos( rutaResponseDto.isAptoNinos() );
34             rutaEntity.setAptoPerros( rutaResponseDto.isAptoPerros() );
35             rutaEntity.setDescripcion( rutaResponseDto.getDescripcion() );
36             rutaEntity.setDificultad( rutaResponseDto.getDificultad() );
37             rutaEntity.setDistancia( rutaResponseDto.getDistancia() );
38             rutaEntity.setDuracion( rutaResponseDto.getDuracion() );
39         }
40
41         return rutaEntity;
42     }
43 }
```

Esta es la implementación de un mapper, que vuelvo a repetir, se hace de forma automática. Imagina tener que hacer eso para cada servicio de cada entidad. Es muy cómodo.



Repository

The screenshot shows the Eclipse IDE interface with the following details:

- Packaging Explorer:** Shows the project structure under "asturutas-back-main [boot] [devtools]". It includes packages like "src/main/java/com.asturutas.spring.web.app" containing "RutaMapper.java", "RutaMapperImpl.java", and "RutaRepository.java". Other packages like "config", "controllers", "dto", "entity", "mapper", and "repository" also contain various repository and service classes.
- RutaRepository.java Content:**

```

1 package com.asturutas.spring.web.app.repository;
2
3 import java.util.List;
4
5 @Repository
6 public interface RutaRepository extends JpaRepository<RutaEntity, Integer> {
7
8     @Query("SELECT r FROM RutaEntity r JOIN r.actividades a WHERE upper(a.nombre) = upper(?1)")
9     List<RutaEntity> findRutasByActividad(String actividad);
10
11     @Query("SELECT r FROM RutaEntity r JOIN r.municipios m WHERE upper(m.nombre) = upper(?1)")
12     List<RutaEntity> findRutasByMunicipio(String municipio);
13
14     @Query("SELECT r FROM RutaEntity r WHERE upper(r.usuario) = upper(?1)")
15     List<RutaEntity> findRutasByUsuario(String usuario);
16
17     @Query("SELECT r FROM RutaEntity r WHERE r.movilidadReducida = true")
18     List<RutaEntity> findRutasByMovilidad();
19
20     @Query("SELECT r FROM RutaEntity r WHERE r.aptoPerros = true")
21     List<RutaEntity> findRutasByPerros();
22
23 }

```
- SQL Editor:** Shows a partial SQL query being typed:

```

re1_0.nombre
from
    roles re1_0
where
    re1_0.id=?

```
- Bottom Status Bar:** Displays system icons and the date/time: 08/06/2024 17:26.

Al repositorio debemos pasarle la entidad y ademas de la entidad, el tipo de valor de la clave primaria. Hemos escogido un tipo primitivo

El repositorio es el que se encarga de realizar el CRUD básico de cada entidad. Crear, mostrar, actualizar, borrar... A parte de esto, también podemos crear nuestras propias Querys personalizadas.

Por ejemplo en la primera, queremos obtener un array de objetos dependiendo de la actividad. Para ello, seleccionamos todo los campos de la entidad, uniendolo con las actividades, poniendo la condicion de que solo obtenga aquellas rutas con la actividad que pasemos como parámetro. Lo mismo para el resto de querys



Services

```

1 package com.asturutas.spring.web.app.service;
2
3 import java.util.List;
4
5 public interface RutaService {
6
7     List<RutaResponseDto> findAll();
8
9     RutaResponseDto findById(int id);
10
11     RutaResponseDto create(RutaRequestDto rutaRequestDto);
12
13     RutaResponseDto update(int id, RutaRequestDto rutaRequestDto);
14
15     RutaResponseDto delete(int id);
16
17     List <RutaResponseDto> findRutasByActividad (String actividad);
18
19     List <RutaResponseDto> findRutasByMunicipio (String municipio);
20
21     List <RutaResponseDto> findRutasByUsuario (String usuario);
22
23
24
25

```

Creamos una interfaz para almacenar el nombre de los metodos, despues desarrollaremos su logica en el paquete impl:

```

78     RutaEntity rutaEntity = rutaMapper.responseDtoToEntity(id, rutaResponseDto);
79     rutaRepository.delete(rutaEntity);
80     return rutaResponseDto;
81 }
82
83 @Override
84 public List<RutaResponseDto> findRutasByActividad(String actividad) {
85     List<RutaEntity> rutaEntityList = rutaRepository.findRutasByActividad(actividad);
86     if(rutaEntityList.isEmpty()) {
87         return new ArrayList<>();
88     } else {
89         List <RutaResponseDto> rutaResponseDtoList = new ArrayList<>();
90         for(RutaEntity rutaEntity : rutaEntityList) {
91             RutaResponseDto rutaResponseDto = rutaMapper.entityToRutaResponseDto(rutaEntity);
92             rutaResponseDtoList.add(rutaResponseDto);
93         }
94         return rutaResponseDtoList;
95     }
96 }
97
98 @Override
99 public List<RutaResponseDto> findRutasByMunicipio(String municipio) {

```

Seguimos explicando el findByActividad. Primero almacenamos en un array de RutaEntity todas aquellas rutas que tenga la actividad que pasamos por parametro (que es la que enviamos desde el front). Como trabajar con las entidades no es lo mas optimo, ahora tenemos que pasar este array a un array de dto. Recorremos el array y vamos mapeando cada entidad con el mapper, posteriormente lo guardamos y lo devolvemos.



Conversores

The screenshot shows the Eclipse IDE interface with the following details:

- Top Bar:** workspace-spring - asturatas-back-main/src/main/java/com/asturatas/spring/web/app/validator/ActividadRequestDtoConverter.java - Spring Tool Suite 4
- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and help.
- Package Explorer:** Shows the project structure:
 - asturatas-back-main [root] [devtools]
 - src/main/java
 - com.asturatas.spring.web.app
 - config
 - controllers
 - dto
 - entity
 - mapper
 - repository
 - service
 - util
 - validator
 - ActividadRequestDtoConverter.java
 - MunicipioRequestDtoConverter.java
 - ActivaturaApplication.java
 - src/main/resources
 - static
 - css
 - imagenes
 - js
 - templates
- Code Editor:** The current file is ActividadRequestDtoConverter.java, containing the following Java code:

```
1 package com.asturatas.spring.web.app.validator;
2
3 import org.springframework.core.convert.converter.Converter;
4
5 @Component
6 public class ActividadRequestDtoConverter implements Converter<String, ActividadRequestDto> {
7
8     @Override
9     public ActividadRequestDto convert(String source) {
10         ActividadRequestDto actividad = new ActividadRequestDto();
11         actividad.setNombre(source);
12         return actividad;
13     }
14
15 }
16
17 }
```
- Bottom Status Bar:** Shows the operating system taskbar with icons for file, copy, paste, and browser, along with the date and time (08/06/2024 17:39).

Convertimos el valor que enviamos en el formulario, por ejemplo, Miercoles, y lo convertimos al Dto que queremos.

Application.properties

The screenshot shows the Spring Tool Suite 4 interface. The title bar reads "workspace-spring - asturatas-back-main/src/main/resources/application.properties - Spring Tool Suite 4". The menu bar includes File, Edit, Navigate, Search, Project, Run, Window, and Help. The toolbar has icons for New, Open, Save, Run, Stop, and others. The Package Explorer view on the left shows the project structure with packages like entity, mapper, repository, service, util, validator, and AstarutaApplication.java, along with resources static, templates, and application.properties. The central workspace shows the application.properties file open, displaying the following configuration:

```
1 spring.application.name=asturatas
2
3 server.port=33
4
5 spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
6 spring.datasource.username=postgres
7 spring.datasource.password=admin
8 spring.datasource.driver-class-name=org.postgresql.Driver
9 spring.jpa.open-in-view=false
10 spring.jpa.show-sql=true
11 spring.jpa.properties.hibernate.format_sql=true
12
13 spring.thymeleaf.cache=false
```

Añadimos el nombre de la aplicación, el puerto donde queremos que se nos muestre, la dirección de localhost donde está la base de datos arrancada, el nombre y la clave del usuario, el driver, y agregamos algunas funciones para que nos muestre cuando hagamos una acción la query



Capa de persistencia o de datos

Todos los datos se deben guardar en algún lado, para ello hemos utilizado PostgreSQL. Ahí guardamos los usuarios, las rutas, las actividades, los municipios... Gran parte de lo que mostramos en el front-end, por no decir casi todo, está contenido en la base de datos.

	id	nombre	usuario	email	contrasena	fecha_creacion
1	admin	admin	admin@asturutas.com	\$2a\$10\$YvOrPKT9EfmrW29WXj3Q/uHoon927g58x9kt95KB1eohDSIRly5/y		[NULL]
2	Iker Alonso Garcia	ikeralonso	ikeralonso@gmail.com	\$2a\$10\$R78ZCRoBMxwCrnHWOFxJQ9dPxSSxP4IXLV5kSV.7Dz7wkQaAT		[NULL]

En la parte izquierda se pueden ver todas las tablas que están almacenadas. La tabla que se muestra a la izquierda es la de los usuarios. Esos datos serán los que luego busque Spring cuando iniciemos sesión, ya sea para ver que todo coincide, o al registrar, para comprobar que no introducimos algo que ya existe.

Vamos a usar una tabla que contiene relaciones para explicarla detalladamente

Column Name	#	Tipo de datos	Identidad	Collation	No Nulo	Por defecto
id	1	serial4			[v]	nextrval('municipios_rutas_id_seq')
id_ruta	2	int4			[v]	
nombre_mu...	3	varchar(255)		default	[v]	

El tipo de datos serial4 permite un id autoincremental sin necesidad de usar constraints, para el id de la ruta, que va a ser una clave foránea utilizamos int y lo mismo para el nombre del municipio, con la diferencia de que este es un varchar. Evidentemente ningún campo de esta tabla puede ser nulo. Todas las rutas tienen al menos un municipio

Name	Atributo	Owner	Type	Expresión	Comentario
municipios_rutas_pk	—	municipios_rutas	PRIMARY KEY		

Ponemos la clave primaria que queremos, en este caso es el id de la tabla, el que es serial.

Name	Atributo	Owner	Type	Columna referenciada	Entidad Asociada	Match Type	Regla
fk_municipio	—	municipios_rutas	FOREIGN KEY	municipios_pk	municipios	SIMPLE	Cascade
fk_ruta	—	municipios_rutas	FOREIGN KEY	rutas_pk	rutas	SIMPLE	Cascade

Las relacionamos con las columnas de las tablas involucradas, es decir con Rutas y Municipios



6. Descarga de herramientas, configuración y despliegue de la aplicación

Añadimos una guía para desplegar la aplicación web desde un dispositivo nuevo.

Descarga y configuración de herramientas

Descargamos Java en caso de que no lo tengamos instalado ya

https://www.java.com/es/download/ie_manual.jsp

Recursos de ayuda

- ¿Qué es Java?
- Eliminar versiones antiguas
- Desactivar Java
- Mensajes de error
- Solución de problemas de Java
- Más ayuda

Usuarios de Windows de 64 bits

- ¿Usa exploradores tanto de 32 como de 64 bits?
- Preguntas frecuentes sobre Java de 64 bits para Windows

Instalación fuera de línea

- ¿Tiene problemas al descargar?
- Pruebe con el instalador fuera de línea

Descargar Java para Windows

Version 8 Update 411 (filesize: 65.60 MB) Por qué se recomienda Java 8?

Fecha de publicación: 10 de abril de 2024

Información importante sobre la licencia de Oracle Java

La licencia de Oracle Java ha cambiado para las versiones publicadas a partir del 16 de abril de 2019.

El acuerdo de licencia de Oracle Technology Network para Oracle Java SE es sustancialmente diferente a las licencias de Oracle Java anteriores. Esta licencia permite ciertos usos, como el uso personal y de desarrollo, sin coste alguno (aunque podría haber otros usos autorizados en licencias de Oracle Java anteriores que ya no estén disponibles). Revise las condiciones con atención antes de descargar y utilizar este producto. Puede consultar las preguntas frecuentes aquí.

La licencia comercial y el soporte están disponibles con una suscripción de Java SE de bajo coste.

Descargar Java

Al descargar Java, confirma que ha leído y acepta las condiciones del acuerdo de licencia de Oracle Technology Network para Oracle Java SE

Será necesario para descomprimir el .jar de Spring Tool Suite, que se descargará aquí:

<https://spring.io/tools>

Spring Tools 4 for Eclipse

The all-new Spring Tool Suite 4. Free. Open source.

4.22.1 - LINUX X86_64

4.22.1 - LINUX ARM_64

4.22.1 - MACOS X86_64

4.22.1 - MACOS ARM_64

4.22.1 - WINDOWS X86_64

https://cdn.spring.io/spring-tools/release/STS4/4.22.1.RELEASE/dist/e4.31/spring-tool-suite-4-4.22.1.RELEASE-e4.31.0-win32.win32.x86_64.self-extracting.jar



Descargamos e instalamos PostgreSQL:

<https://www.postgresql.org/download/>

The screenshot shows the PostgreSQL Downloads page. On the left, there's a 'Quick Links' sidebar with options like 'Downloads', 'Packages', 'Source', 'Software Catalogue', and 'File Browser'. The main content area is titled 'Downloads' with a download icon. Below it is a section titled 'PostgreSQL Downloads' with a sub-section 'Packages and Installers'. A note states: 'PostgreSQL is available for download as ready-to-use packages or installers for various platforms, as well as a source code archive if you want to build it yourself.' There is a heading 'Select your operating system family:' followed by four buttons: 'Linux' (with a penguin icon), 'macOS' (with an apple icon), 'Windows' (with a Windows logo icon), and 'BSD' (with a BSD logo icon).

A la hora de instalar, acordarse de las credenciales (usuario y contraseña) será necesaria para añadirla en el application.properties de Spring. Puedes poner mis mismas credenciales:

Usuario: postgres

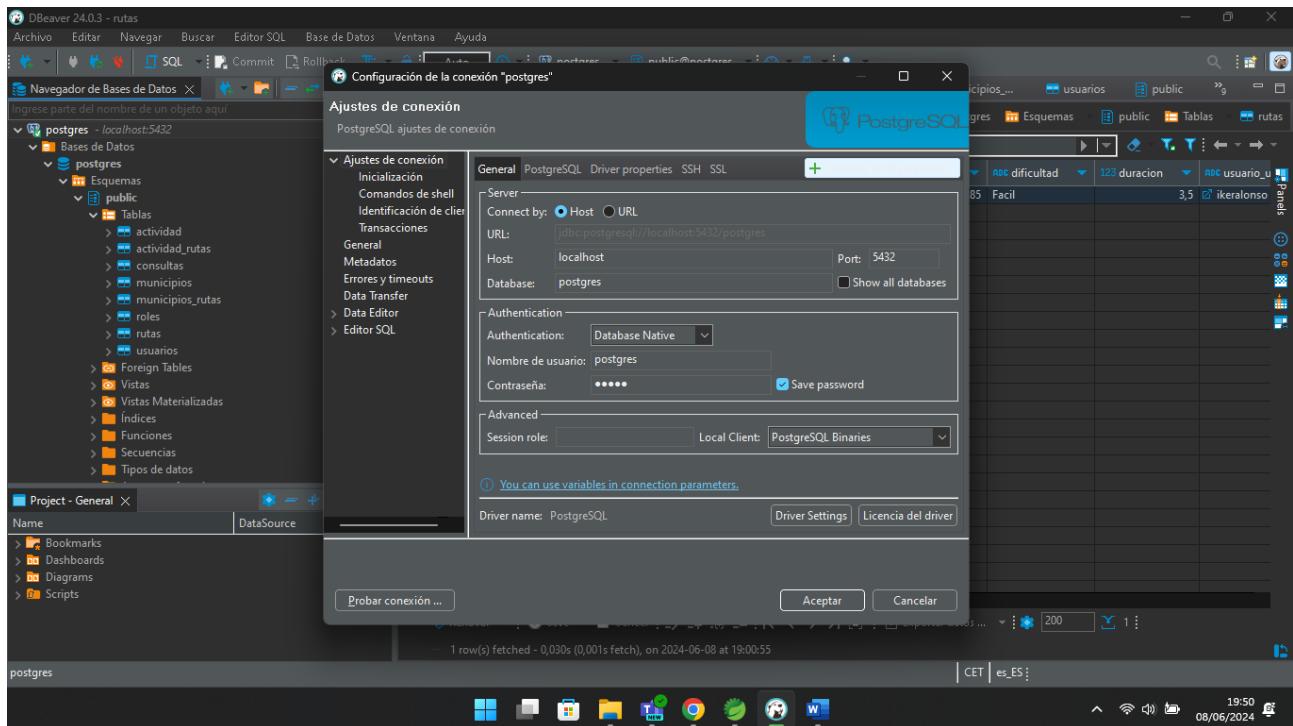
Contraseña: admin

Descargamos Dbeaver para gestionar la BBDD

The screenshot shows the DBeaver Community download page. At the top, there's a navigation bar with links for 'Home', 'About', 'Download', 'Documentation', 'News', 'Support', 'DBeaver PRO', 'CloudBeaver', 'DBeaver Merch', and 'Join our team'. The main content area is titled 'Download' and features two large sections: 'DBeaver Community 24.1' and 'DBeaver PRO 24.0'. Both sections provide download links for various platforms including Windows, macOS, and Linux. A note at the bottom of the page states: 'This website uses cookies to improve your experience. We'll assume you're ok with this, but you can opt-out if you wish.' with buttons for 'Accept', 'Reject', and 'Read more'. The status bar at the bottom right shows the date as 08/06/2024.



Una vez instalado PostgreSQL y Dbeaver, esta es la conexión que se debe hacer en Dbeaver para poder gestionar Postgree:



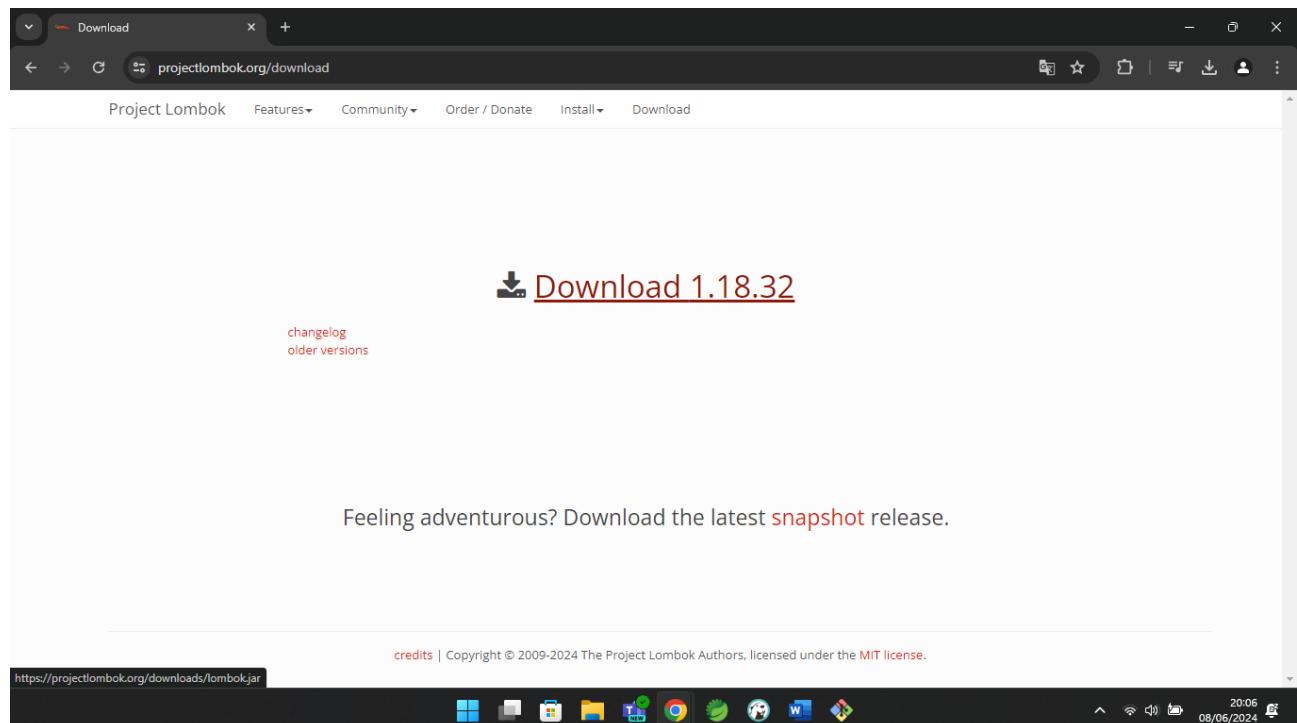
Una vez hecha la conexión, en este enlace puedes encontrar el script para crear todas las tablas y sus relaciones:

<https://github.com/IKALG/asturutas-back/blob/main/despliegue/scriptbbdd.txt>

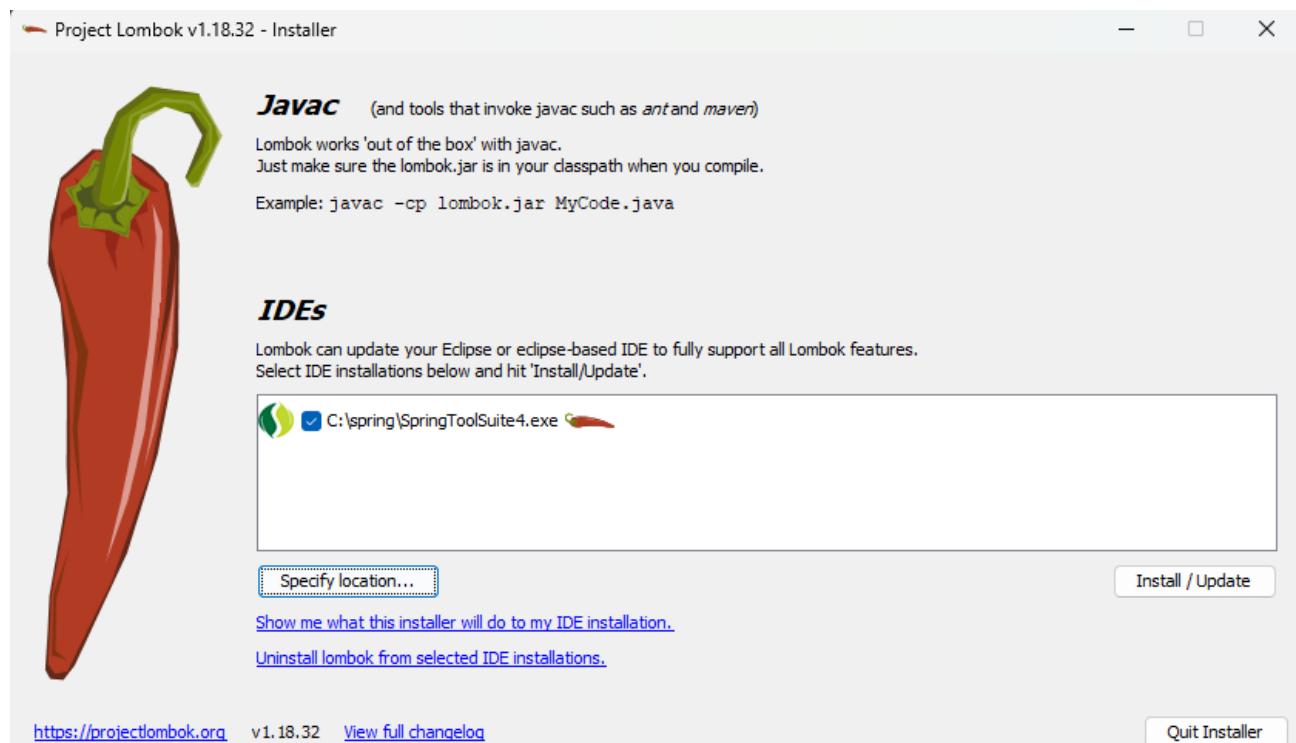
Una vez creada las tablas, metemos una actividad y un municipio, solo para probar, para que no nos muestre los desplegables de crear una ruta vacíos.



Configurada ya la capa de datos, nos vamos con la capa de negocio / lógica de la aplicación. Tenemos que descargar la librería de loombok para que no nos de error al compilar la aplicación. Para ello:



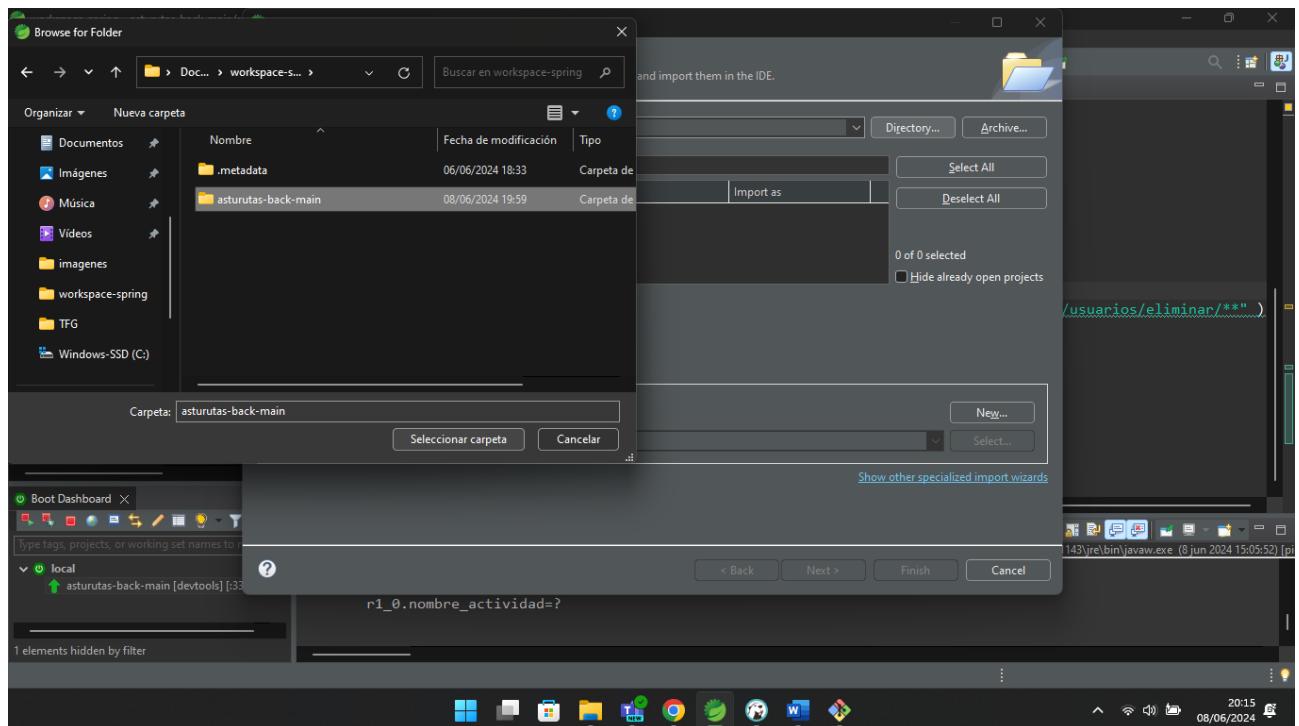
Abrimos el .jar que hemos descargado y buscamos la ruta donde está instalado nuestro STS



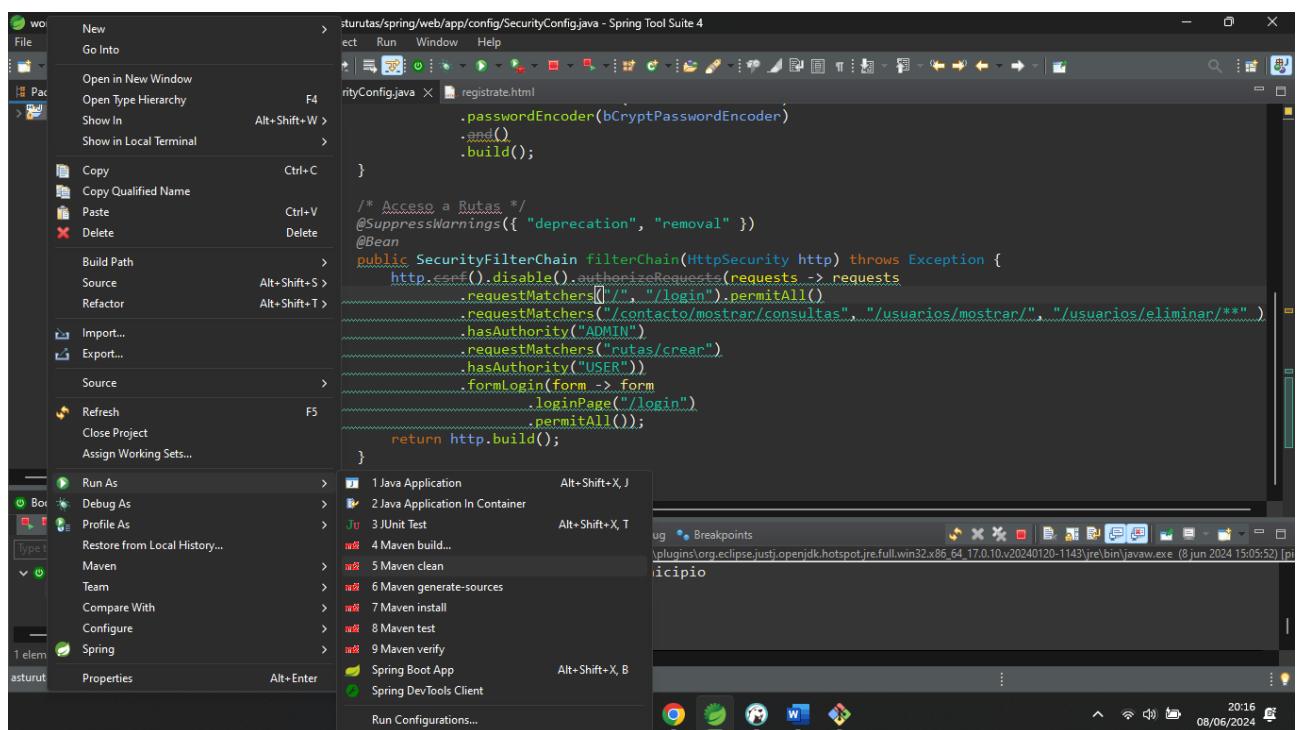
Le damos a install / update y ya está, lombok ya está completamente funcional y no nos dará ningun error al importar nuestro proyecto



Ahora, importamos el proyecto, que se encuentra en el GitHub proporcionado:



Una vez importado hacemos un maven clean y despues un maven install



Si has usado las mismas credenciales que tenía yo no hace falta que toques el application.properties, en caso de que tengas unas diferentes debes cambiar esas dos líneas con tus datos, además de recordar que el puerto donde está alojada la aplicación es el 33 > <http://localhost:33/>

The screenshot shows the Eclipse IDE interface. In the top left, the 'Package Explorer' view displays various HTML files and the 'application.properties' file. The 'application.properties' file is open in the center, showing configuration settings like 'spring.application.name=asturutas', 'server.port=33', and 'spring.datasource.url=jdbc:postgresql://localhost:5432/postgres'. In the bottom right, the 'Console' tab is active, showing log messages from the application. One message reads: '2024-06-08T20:21:51.361+02:00 INFO 12124 --- [asturutas] [(352)-127.0.0.1] j.LocalContainerEntityManagerFactoryBean'. The status bar at the bottom right indicates the date and time as '08/06/2024' and '20:20'.

Despliegue de la aplicación

Una vez hecho todo lo anterior, solo tienes que darle click a ese botón para arrancar la aplicación. No debería dar ningún problema.

This screenshot shows the Eclipse IDE after the application has been started. The 'Boot Dashboard' view on the left shows the application 'asturutas-back-main [devtools]' is running. The 'Console' tab on the right displays the application's startup logs. It includes messages like 'INFO 12124 --- [asturutas] [(352)-127.0.0.1] j.LocalContainerEntityManagerFactoryBean' and 'INFO 12124 --- [asturutas] [(352)-127.0.0.1] com.zaxxer.hikari.HikariDataSource'. The status bar at the bottom right shows the date and time as '08/06/2024' and '20:21'.



7. Evaluación

Introducción

El propósito de esta evaluación es comprobar que la aplicación web cumpla con unos estándares de calidad, accesibilidad y compatibilidad dando una buena experiencia al usuario en diferentes dispositivos y navegadores

Validaciones de páginas de estilo

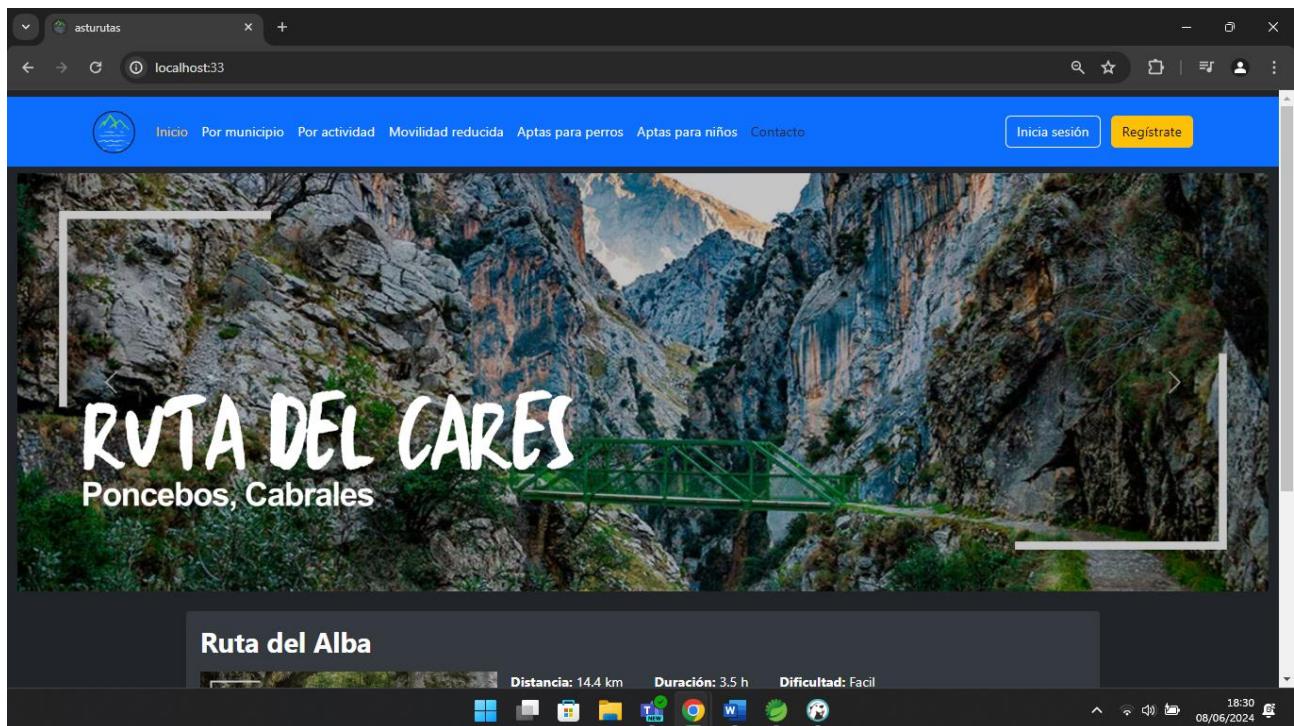
No es necesario realizar validación de las páginas de estilo ya que el estilos se han configurado mediante Bootstrap.

Validación de enlaces

Después de comprobar todos los enlaces y todos los botones que hay en la aplicación, hemos podido llegar a comprobar que están bien configurados y no hay ninguna ruta rota

Validación de la resolución

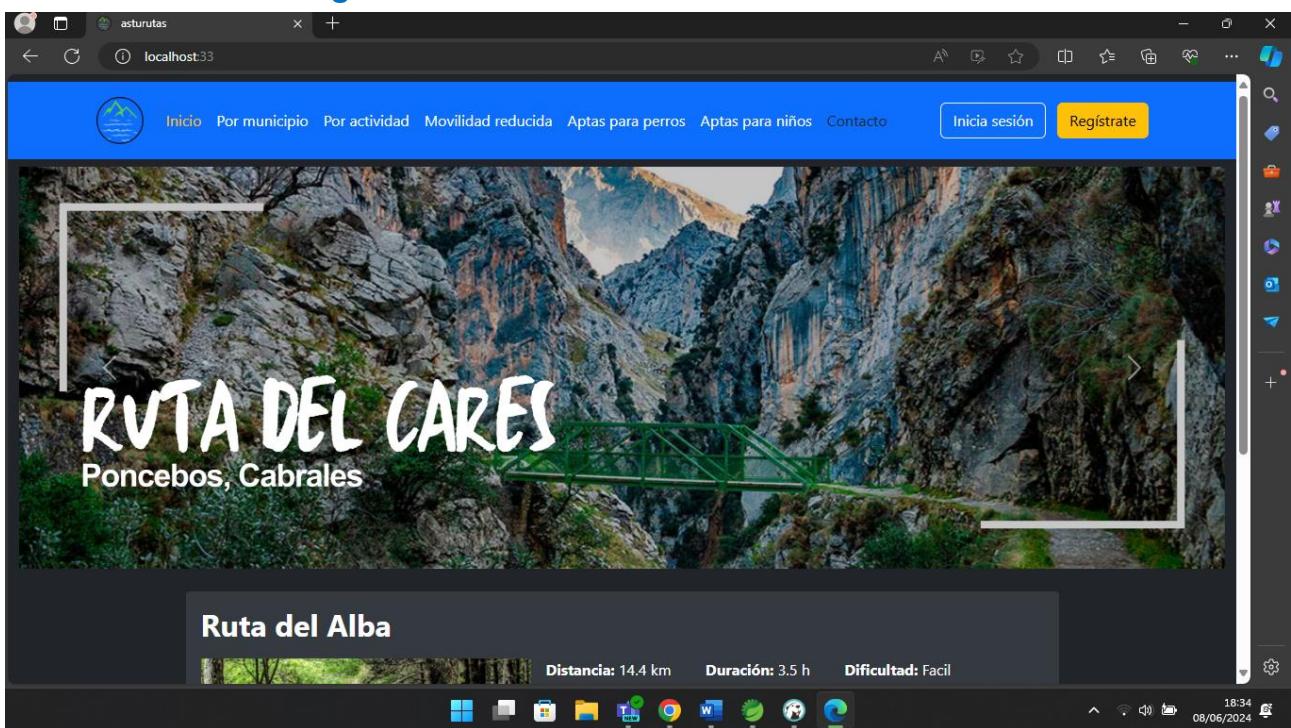
Así es como se ve en ordenadores:



Así es como se vería para tablets y móviles respectivamente:



Validación de navegadores



En Microsoft Edge, por ejemplo, se sigue viendo igual que se veía en Google Chrome



8. Valoración personal del trabajo realizado (análisis DAFO + análisis CAME)

En general, estoy muy orgulloso del trabajo realizado, o al menos así lo veo ahora. Ya no solo porque lo he hecho en un mes después de estar decidido a no hacerlo, si no porque además he utilizado para el back-end un framework que no había nunca utilizado cuando lo mas fácil, para ahorrarme quebraderos de cabeza, hubiese sido utilizar lo dado en clase. A parte de la formación que recibí en la empresa de las FCT también le tuve que dedicar estudio desde casa al llegar, ver videos... La dedicación que he puesto es en sí lo que me hace estar mas orgulloso del trabajo realizado.

Análisis Dafo

Fortalezas

- Los usuarios pueden disfrutar de muchas funcionalidades sin necesidad de registrarse, lo que facilita el acceso.
- Ofrecer filtros por municipios, actividades, accesibilidad, compatibilidad con niños y mascotas aumenta la relevancia y utilidad.
- Permitir a los usuarios subir sus rutas fomenta la participación y el sentido de la web.
- El acceso gratuito a todas las funcionalidades puede atraer a una amplia audiencia.

Debilidades

- La calidad y cantidad de las rutas puede variar según la participación de los usuarios.
- Los usuarios no pueden guardar preferencias o rutas favoritas, lo que podría limitar la experiencia del usuario.

Oportunidades

- La creciente tendencia hacia el turismo local puede incrementar el interés y uso de la aplicación.
- Asociaciones con municipios, oficinas de turismo y empresas locales pueden proporcionar beneficios mutuos y mejorar la visibilidad de la aplicación.
- Incorporar características adicionales como guías de audio, recomendaciones personalizadas, y eventos locales puede aumentar el valor de la aplicación.

Amenazas

- La existencia de otras aplicaciones y plataformas similares puede dificultar la captación y retención de usuarios.
- La necesidad de conexión a Internet puede limitar el uso de la aplicación en áreas rurales con baja conectividad.



Análisis CAME

Corregir

- Moderaciones de las rutas subidas para asegurar su calidad.
- Invertir en pruebas y validaciones exhaustivas antes del lanzamiento. Falta de personalización avanzada:
- Implementar medidas de seguridad robustas, como el cifrado de datos y la autenticación de dos factores (2FA).

Afrontar

- Diferenciar la aplicación ofreciendo funcionalidades únicas y un valor añadido, como rutas exclusivas o colaboraciones con guías locales.
- Optimizar la aplicación para que funcione en condiciones de baja conectividad.

Mantener

- Continuar permitiendo el acceso a funcionalidades clave sin necesidad de registro
- Seguir mejorando y ampliando las opciones de filtrado y las funcionalidades disponibles.
- Crear una comunidad activa alrededor de la aplicación.
- Mantener el acceso gratuito a las funcionalidades esenciales, pero considerar la implementación de un modelo freemium con características premium opcionales.

Explorar

- Colaborar con oficinas de turismo y entidades locales para promover la aplicación.
- Crear campañas de marketing dirigidas a turistas locales
- Introducir nuevas características como guías de audio, recomendaciones personalizadas basadas en las preferencias del usuario, y eventos locales relacionados con las rutas.

9. Posibles ampliaciones

Hay muchas opciones de mejora que se pueden implementar, y que no pude añadir, ya sea por falta de tiempo o por falta de conocimiento. Una de las cosas que quería haber añadido y no tuve tiempo era la posibilidad de añadir comentarios, dar me gusta... también incorporar una pestaña para el usuario, con una cuenta donde añadir una biografía, una foto de perfil... Darle mas un aspecto de red social o de foro, principalmente. También la posibilidad de añadir un mapa, que era una de las ideas principales, pero por tiempo ha sido imposible. La idea (real) es seguir trabajando sobre esta base un grupo de amigos y en un futuro sacarla a la red.

