PROYECTO INTEGRADO

PacMan Activities

Juan Diego Teruel Espejo

MANUAL DE USUARIO....3
Registro e Inicio de Sesión....3
Navegación Principal....3
Página de Inicio....4
Página de Actividades....4
Página de Perfil....5
FUNCIONAMIENTO FRONT END DE LA APLICACIÓN....6
Tecnologías Utilizadas....6
Componentes....7
Servicios....9
FUNCIONAMIENTO BACKEND DE LA APLICACIÓN....10
Tecnologías Utilizadas....10
Documentación de Controladores....12
Json Web Token (JWT)....18
CONCLUSIÓN PERSONAL....20

MANUAL DE USUARIO

Bienvenido a "PacMan Activities", la plataforma donde ofertantes y consumidores pueden conectarse a través de actividades emocionantes. Este manual está diseñado para guiarte a través de las funcionalidades de la aplicación, desde el registro hasta la gestión de actividades y perfil.

Registro e Inicio de Sesión

Registro de Usuario

1. Acceder a la Plataforma:
    - Navega a la página principal de "PacMan Activities".
    - Haz clic en "Registrarse".
2. Completar el Formulario de Registro:
    - Rellena tus datos personales: nombre, correo electrónico, contraseña.
    - Selecciona tu rol: Ofertante, Consumidor o Ambos.
    - Haz clic en "Registrarse" para completar el proceso.

Inicio de Sesión

1. Iniciar Sesión:
    - En la página principal, haz clic en "Iniciar Sesión".
    - Ingresa tu nombre de usuario y contraseña.
    - Haz clic en "Entrar" para acceder a tu cuenta.

Navegación Principal

Barra de Navegación

- Inicio: Página de presentación con actividades relevantes según tu rol.
- Actividades: Listado de todas las actividades disponibles.
- Perfil: Gestión de tu perfil personal.
- Cerrar Sesión: Ícono para cerrar sesión y salir de tu cuenta.

Página de Inicio

Contenido de la Página de Inicio

- Para Ofertantes: Se mostrarán las actividades que has creado.
- Para Consumidores: Se mostrarán las actividades a las que te has inscrito.
- Para Ambos Roles: Se mostrarán tanto las actividades creadas como las inscritas.

Página de Actividades

Explorar Actividades

1. Ver Actividades Disponibles:
    - Navega a la sección "Actividades" desde la barra de navegación.
    - Verás una lista de actividades disponibles de todos los ofertantes.
2. Buscar y Filtrar Actividades:
    - Usa la barra de búsqueda para encontrar actividades por nombre.
    - Usa el selector de tipo de actividad para filtrar por categorías como deporte, cocina, etc.

Crear Actividad (Solo Ofertantes)

1. Publicar Nueva Actividad:
    - Haz clic en el botón "Crear Actividad".
    - Completa el formulario con los detalles de la actividad: nombre, descripción, fecha y hora, ubicación, capacidad de participantes, y precio.
    - Haz clic en "Publicar" para guardar y publicar la actividad.

Detalles de la Actividad

1. Ver Detalles:
    - Haz clic en una actividad para abrir una nueva ventana con más detalles.
    - Verás la descripción completa, ubicación en el mapa, horarios y datos del ofertante.
2. Acciones según el Rol:
    - Ofertante: Si eres el creador, podrás editar o borrar la actividad (si no tiene participantes y no ha comenzado).

- Consumidor: Podrás participar en la actividad y, una vez terminada, valorar la actividad.

Página de Perfil

Gestión del Perfil

1. Editar Perfil:
    - Navega a la sección "Perfil" desde la barra de navegación.
    - Actualiza tu información personal como nombre, correo electrónico, foto de perfil, etc.
    - Guarda los cambios.
2. Cambio de Rol:
    - Puedes hacerte ofertante o consumidor si no tienes algún rol.
    - También puedes dejar de ser ofertante o consumidor.
3. Eliminar Perfil:
    - Tienes la opción de eliminar tu perfil completamente desde la sección de configuración.

FUNCIONAMIENTO FRONT END DE LA APLICACIÓN

El frontend de la aplicación "PacMan Activities" ha sido desarrollado utilizando una serie de tecnologías modernas y prácticas que garantizan una interfaz de usuario atractiva y funcional. La base de esta implementación se encuentra en Angular, un framework de desarrollo de aplicaciones web que facilita la creación de Single Page Applications (SPA). A continuación, se describen las tecnologías utilizadas y su rol en la aplicación.

Tecnologías Utilizadas

Angular: Angular es el framework principal utilizado para desarrollar la aplicación. Proporciona una estructura sólida para construir aplicaciones web dinámicas y de una sola página.

HTML: Se utiliza para estructurar el contenido de la aplicación. Define la estructura básica de los componentes y las vistas. Proporciona una estructura semántica clara y accesible para el contenido web.

TypeScript: Es el lenguaje de programación principal en Angular. Es un superset de JavaScript que añade tipos estáticos y otras características avanzadas. Tiene tipado estático que ayuda a detectar errores en tiempo de compilación, mejorando la robustez del código. Está orientado a objetos lo que facilita la organización y reutilización del código mediante clases e interfaces.

CSS: Se utiliza para estilizar la aplicación. Define la apariencia de los elementos HTML y proporciona un diseño visual coherente.

Bootstrap: Es un framework de CSS que se utiliza para crear un diseño responsivo y atractivo. Facilita la creación de diseños fluidos y responsivos. También proporciona una amplia gama de componentes predefinidos como botones, formularios y modales. Incluye clases de utilidad para manejar el espaciado, la alineación y otros aspectos del diseño.

API de Google Maps: La API de Google Maps se integra en la aplicación para permitir a los usuarios seleccionar y visualizar ubicaciones en un mapa interactivo.

Componentes

ActividadForm

- Descripción: Este componente se utiliza para crear o editar una actividad. Los ofertantes pueden ingresar detalles como el nombre de la actividad, descripción, fecha y hora, ubicación, capacidad de participantes y precio.
- Funcionalidad:
    - Formulario para ingresar y actualizar los detalles de la actividad.
    - Validación de campos requeridos y formato correcto.
    - Envío de datos al backend para guardar o actualizar la actividad.

ActividadView

- Descripción: Este componente muestra los detalles completos de una actividad específica. Dependiendo del rol del usuario y las circunstancias, se pueden realizar varias acciones.
- Funcionalidad:
    - Visualización de la descripción completa, ubicación en el mapa, horarios y datos del ofertante.
    - Ofertante: Editar o borrar la actividad (si no tiene participantes y no ha comenzado).
    - Consumidor: Participar en la actividad, dejar de participar, y valorar la actividad después de finalizada.

Actividades

- Descripción: Este componente muestra una lista de todas las actividades disponibles. Incluye un buscador y un selector para filtrar por tipo de actividad.
- Funcionalidad:
    - Listado de todas las actividades de los ofertantes.
    - Barra de búsqueda para encontrar actividades por nombre.
    - Selector de tipo de actividad para filtrar por categorías como deporte, cocina, etc.
    - Ofertante: Botón para crear una nueva actividad.

Home

- Descripción: La primera página que se muestra al iniciar sesión. Contiene una bienvenida y una lista de actividades dependiendo del rol del usuario.
- Funcionalidad:
    - Ofertante: Muestra las actividades que has creado.
    - Consumidor: Muestra las actividades en las que te has inscrito.
    - Ambos Roles: Muestra tanto las actividades creadas como las inscritas.

Login

- Descripción: Componente para iniciar sesión en la aplicación.
- Funcionalidad:
    - Formulario de inicio de sesión para ingresar nombre de usuario y contraseña.
    - Envío de datos al backend para autenticar al usuario.

Register

- Descripción: Componente para registrar una nueva cuenta en la aplicación.
- Funcionalidad:
    - Formulario de registro para ingresar datos personales y seleccionar roles (Ofertante, Consumidor o Ambos).
    - Validación de campos requeridos y formato correcto.
    - Envío de datos al backend para crear una nueva cuenta.

MiPerfil

- Descripción: Componente para ver y gestionar el propio perfil del usuario.
- Funcionalidad:
    - Visualización de la información del perfil.
    - Botones para editar el perfil, cambiar roles (hacerse ofertante o consumidor, dejar de serlo) y eliminar el perfil.

PerfilForm

- Descripción: Formulario para editar el perfil del usuario, accesible desde MiPerfil.
- Funcionalidad:
    - Formulario para actualizar información personal.
    - Validación de campos requeridos y formato correcto.
    - Envío de datos al backend para actualizar el perfil.

Perfil

- Descripción: Componente para ver el perfil detallado de otros usuarios. Se accede desde ActividadView.
- Funcionalidad:
    - Visualización de la información del perfil de otros usuarios.

StarRating

- Descripción: Componente para valorar actividades.
- Funcionalidad:
    - Interfaz para que los consumidores califiquen una actividad con estrellas.
    - Al valorar una actividad, se calcula automáticamente la media de valoraciones de esa actividad y se asigna.
    - También se calcula la media de valoraciones de las actividades del ofertante que ha creado esa actividad y se le asigna.

Servicios

Todos los servicios en "PacMan Activities" realizan peticiones a su correspondiente controlador en el backend. Los servicios disponibles son:

1. ActividadConsumidorService
2. ActividadService
3. AuthService
4. ConsumidorService
5. OfertanteService
6. TipoActividadService

FUNCIONAMIENTO BACKEND DE LA APLICACIÓN

En el desarrollo del backend de la aplicación, se han utilizado tecnologías modernas y prácticas para garantizar un sistema robusto y escalable. La base de esta implementación se encuentra en Spring Boot, un framework de desarrollo de aplicaciones Java que facilita la creación de aplicaciones web y servicios RESTful de manera rápida y sencilla.

Tecnologías Utilizadas

Spring Boot con JPA: La combinación de Spring Boot con JPA (Java Persistence API) proporciona una forma eficiente de interactuar con la base de datos utilizando objetos Java simples (entidades). JPA se encarga de mapear estos objetos a tablas en la base de datos relacional, lo que simplifica enormemente la lógica de acceso a datos.

Generación de Entidades con Ingeniería Inversa: Se ha utilizado la funcionalidad de ingeniería inversa proporcionada por Spring Boot para generar automáticamente las entidades a partir de la estructura de la base de datos existente. Esto reduce significativamente el tiempo y el esfuerzo necesarios para crear las clases de entidad manualmente.

Spring Data JPA Repositories: Se han creado interfaces de repositorio que extienden JpaRepository, una interfaz de Spring Data JPA, para manejar las operaciones de acceso a datos básicas (como guardar, actualizar, eliminar y consultar) de forma simple y declarativa. También se han creado consultas personalizadas para hacer consultas más complejas directamente en el repositorio para aumentar su eficiencia.

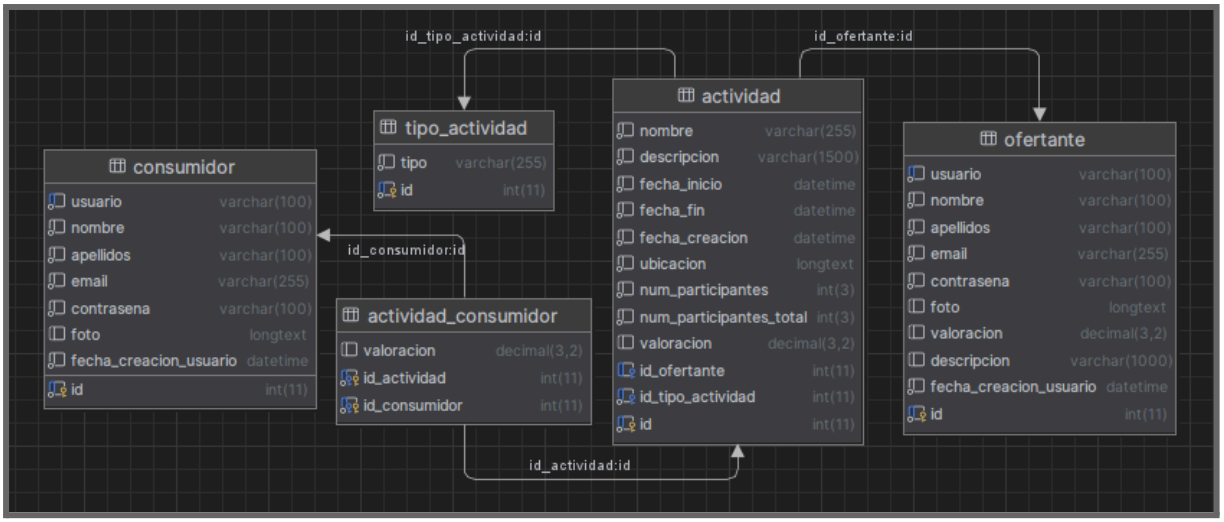
Implementación de Servicios: Se han definido interfaces para los servicios de la aplicación, que luego se han implementado para proporcionar la lógica de negocio necesaria. Estos servicios encapsulan la lógica de acceso a datos y proporcionan una capa de abstracción entre los controladores y los repositorios.

DTOs y Mappers: Se han creado DTOs (Data Transfer Objects) para representar la información que se enviará desde el backend al frontend, evitando la exposición innecesaria de los detalles de la entidad. Además, se han implementado mappers para convertir entre entidades y DTOs de manera eficiente y reutilizable.

JWT (JSON Web Tokens): Se ha implementado un mecanismo de autenticación basado en JWT para proporcionar seguridad en las solicitudes entre el cliente y el

servidor. JWT permite generar tokens de acceso que se utilizan para autenticar y autorizar a los usuarios en cada solicitud.

En resumen, la combinación de estas tecnologías y prácticas de desarrollo proporciona una base sólida para el backend de la aplicación, permitiendo una fácil escalabilidad, mantenibilidad y seguridad en el sistema. La arquitectura sigue los principios de diseño de software modernos y está diseñada para cumplir con los requisitos de rendimiento y seguridad del proyecto.



Documentación de Controladores

AuthController

Ruta Base: /api/auth/

Este controlador gestiona las operaciones relacionadas con la autenticación y autorización de usuarios en el sistema.

- Método: login
  - Ruta: POST /api/auth/login
  - Descripción: Permite a los usuarios autenticarse en el sistema y devuelve el JWT.
  - Parámetros: Credenciales de inicio de sesión (usuario, contraseña).
- Método: registerConsumidor
  - Ruta: POST /api/auth/registerConsumidor
  - Descripción: Permite a los consumidores registrarse en el sistema.
  - Parámetros: ConsumidorDTO
- Método: registerOfertante
  - Ruta: POST /api/auth/registerOfertante
  - Descripción: Permite a los ofertantes registrarse en el sistema.
  - Parámetros: OfertanteDTO
- Método: registerOfertanteConsumidor
  - Ruta: POST /api/auth/registerOfertanteConsumidor
  - Descripción: Permite registrarse como ofertante y consumidor a la vez en el sistema.
  - Parámetros: OfertanteDTO

ConsumidorController

Ruta Base: /api/consumidor/

Este controlador gestiona las operaciones relacionadas con los consumidores en el sistema.

- Método: get
    - Ruta: GET /api/consumidor/get
    - Descripción: Devuelve una lista de todos los consumidores registrados.
- Método: getById
    - Ruta: GET /api/consumidor/get/{id}
    - Descripción: Obtiene un consumidor por su ID.
    - Parámetros: id
- Método: getByUsuario
    - Ruta: GET /api/consumidor/getUsuario/{usuario}
    - Descripción: Obtiene un consumidor por su nombre de usuario.
    - Parámetros: usuario (nombre de usuario)
- Método: exist
    - Ruta: GET /api/consumidor/exist/{usuario}
    - Descripción: Verifica si existe un consumidor con un nombre de usuario dado devuelve true si existe y false si no.
    - Parámetros: usuario (nombre de usuario)
- Método: post
    - Ruta: POST /api/consumidor/post
    - Descripción: Registra un nuevo consumidor en el sistema.
    - Parámetros: ConsumidorDTO
- Método: put
    - Ruta: PUT /api/consumidor/put
    - Descripción: Actualiza los detalles de un consumidor existente.
    - Parámetros: ConsumidorDTO
- Método: delete
    - Ruta: DELETE /api/consumidor/delete/{id}
    - Descripción: Elimina un consumidor por su ID.
    - Parámetros: id

OfertanteController

Ruta Base: /api/ofertante/

Este controlador gestiona las operaciones relacionadas con los oferentes en el sistema.

- Método: get
    - Ruta: GET /api/ofertante/get
    - Descripción: Devuelve una lista de todos los oferentes registrados.
    - Respuesta: Lista de OfertanteDTO
- Método: getById
    - Ruta: GET /api/ofertante/get/{id}
    - Descripción: Obtiene un oferente por su ID.
    - Parámetros: id
- Método: getByUsuario
    - Ruta: GET /api/ofertante/getUsuario/{usuario}
    - Descripción: Obtiene un ofertantepor su nombre de usuario.
    - Parámetros: usuario (nombre de usuario)
- Método: exist
    - Ruta: GET /api/ofertante/exist/{usuario}
    - Descripción: Verifica si existe un ofertante con un nombre de usuario dado.
    - Parámetros: usuario (nombre de usuario)
- Método: post
    - Ruta: POST /api/ofertante/post
    - Descripción: Registra un nuevo ofertanteen el sistema.
    - Parámetros: OfertanteDTO
- Método: put
    - Ruta: PUT /api/ofertante/put
    - Descripción: Actualiza los detalles de un ofertanteexistente.
    - Parámetros: OfertanteDTO
- Método: delete
    - Ruta: DELETE /api/ofertante/delete/{id}
    - Descripción: Elimina un ofertantepor su ID.
    - Parámetros: id

TipoActividadController

Ruta Base: /api/tipoActividad/

Este controlador gestiona las operaciones relacionadas con los tipos de actividad en el sistema.

- Método: get
  - Ruta: GET /api/tipoActividad/get
  - Descripción: Devuelve una lista de todos los tipos de actividad disponibles.
- Método: getById
  - Ruta: GET /api/tipoActividad/get/{id}
  - Descripción: Obtiene un tipo de actividad por su ID.
  - Parámetros: id

ActividadController

Ruta Base: /api/actividad/

Este controlador gestiona las operaciones relacionadas con las actividades en el sistema.

- Método: get
  - Ruta: POST /api/actividad/get
  - Descripción: Devuelve una lista de actividades según criterios de búsqueda.
  - Parámetros: _ActividadRequest (ID del tipo de actividad, ID del oferente, nombre de actividad)
- Método: getById
  - Ruta: GET /api/actividad/get/{id}
  - Descripción: Obtiene una actividad por su ID.
  - Parámetros: id
- Método: getByIdConsumidor
  - Ruta: GET /api/actividad/getConsu/{id}
  - Descripción: Obtiene las actividades en las que está inscrito un consumidor por su ID.
  - Parámetros: id
- Método: post
  - Ruta: POST /api/actividad/post
  - Descripción: Crea una nueva actividad en el sistema.
  - Parámetros: ActividadDTO
- Método: put
  - Ruta: PUT /api/actividad/put
  - Descripción: Actualiza los detalles de una actividad existente.
  - Parámetros: ActividadDTO
- Método: delete
  - Ruta: DELETE /api/actividad/delete/{id}
  - Descripción: Elimina una actividad por su ID.
  - Parámetros: id

ActividadConsumidorController

Ruta Base: /api/actividadConsumidor/

Este controlador gestiona las operaciones relacionadas con la participación de los consumidores en actividades.

- Método: get
  - Ruta: GET /api/actividadConsumidor/get
  - Descripción: Devuelve una lista de todas las relaciones de actividad-consumidor.
- Método: getById
  - Ruta: GET /api/actividadConsumidor/get/{idActividad}/{idConsumidor}
  - Descripción: Obtiene una relación actividad-consumidor por los IDs de actividad y consumidor.
  - Parámetros: idActividad , idConsumidor
- Método: exist
  - Ruta: GET /api/actividadConsumidor/exist/{idActividad}/{idConsumidor}
  - Descripción: Verifica si existe una relación actividad-consumidor con los IDs de actividad y consumidor dados y devuelve true si existe y false si no.
  - Parámetros: idActividad , idConsumidor
- Método: post
  - Ruta: POST /api/actividadConsumidor/post
  - Descripción: Registra la participación de un consumidor en una actividad.
  - Parámetros: ActividadConsumidorDTO
- Método: put
  - Ruta: PUT /api/actividadConsumidor/put
  - Descripción: Actualiza los detalles de la participación de un consumidor en una actividad.
  - Parámetros: ActividadConsumidorDTO
  - Respuesta: ActividadConsumidorDTO
- Método: delete
  - Ruta: DELETE /api/actividadConsumidor/delete/{idActividad}/{idConsumidor}
  - Descripción: Elimina la participación de un consumidor en una actividad por los IDs de actividad y consumidor.
  - Parámetros: idActividad, idConsumidor)

Json Web Token (JWT)

1. JwtGenerator

- Descripción: Esta clase es responsable de generar tokens JWT basados en la información de autenticación proporcionada.
- Métodos:
  - generarToken(Authentication authentication, Long idOfertante, Long idConsumidor): Genera un token JWT utilizando la información de autenticación proporcionada, incluidos el nombre de usuario y los roles. Además, incluye los IDs del ofertante y del consumidor si están disponibles.
  - obtenerUsernameDeJwt(String token): Extrae el nombre de usuario del token JWT proporcionado.
  - validarToken(String token): Valida si el token JWT proporcionado es válido.
  - getSigningKey(): Devuelve la clave utilizada para firmar los tokens.

2. JwtAuthenticationFilter

- Descripción: Este filtro intercepta las solicitudes entrantes y verifica la validez del token JWT. Si el token es válido, establece la autenticación del usuario en el contexto de seguridad.
- Métodos:
  - doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain): Intercepta las solicitudes entrantes, verifica y procesa el token JWT si está presente.

3. JwtAuthenticationEntryPoint

- Descripción: Esta clase maneja las excepciones de autenticación, enviando una respuesta HTTP 401 Unauthorized si el usuario no está autenticado.
- Métodos:
  - commence(HttpServletRequest request, HttpServletResponse response, AuthenticationException authException): Envía una respuesta HTTP 401 Unauthorized cuando se lanza una excepción de autenticación.

4. CustomUsersDetailsService

- Descripción: Esta clase implementa la interfaz UserDetailsService de Spring Security para cargar los detalles del usuario (como nombre de usuario, contraseña y roles) basados en el nombre de usuario proporcionado.
- Métodos:
    - loadUserByUsername(String username): Carga los detalles del usuario (incluidos los roles) basados en el nombre de usuario proporcionado.

Flujo de Autenticación con JWT

1. Cuando un usuario inicia sesión o realiza una solicitud autenticada, el servidor genera un token JWT utilizando el JwtGenerator.
2. El token JWT se incluye en la respuesta del servidor o se proporciona al cliente de alguna manera (por ejemplo, a través de una cookie o un encabezado HTTP).
3. En las solicitudes posteriores, el cliente incluye el token JWT en el encabezado Authorization.
4. El filtro JwtAuthenticationFilter intercepta las solicitudes entrantes y verifica la validez del token JWT.
5. Si el token es válido, el filtro establece la autenticación del usuario en el contexto de seguridad usando el CustomUsersDetailsService.
6. Si el token no es válido o está ausente, el filtro devuelve un error de autenticación.
7. Si el usuario intenta acceder a un recurso protegido sin un token válido, el JwtAuthenticationEntryPoint maneja la excepción de autenticación y devuelve un error HTTP 401 Unauthorized.

CONCLUSIÓN PERSONAL

Estoy muy contento con cómo ha quedado mi proyecto y me siento orgulloso ya que pese a ser sencilla es muy consistente y gestiona con seguridad y controlando los distintos errores que puede tener, ya que le he dedicado mucho tiempo a intentar "romperla" y que surgieran errores.
Me ha resultado difícil el poder realizar consultas complejas manuales a la base de datos, pero ha merecido la pena para que las peticiones sean mucho más rápidas, como por ejemplo el buscar actividades filtrando.
Por último me ha gustado mucho el diseño y la temática que ha acabado teniendo la página, ya que todo empezó porque en internet vi que se podía poner fondos animados con css e investigando un poco más encontré que se podía dar a un div el aspecto de PacMan con css, y eso acabó marcando toda la temática de la página.