Proyecto Acacias del Mar

Materia: Programación II

Docente: Miguel Silva

Integrantes del grupo: Federico Pared

Link al repositorio:
https://github.com/fedepared/AcaciasDelMar

1. Descripción General del Proyecto

Tema Elegido

El proyecto es un **Sistema de Gestión Integral para el Camping "Acacias del Mar"**. El sistema maneja la administración interna de los recursos del camping, la gestión de personal y un portal de consulta para los socios propietarios.

Objetivos del Sistema

El objetivo principal es centralizar y digitalizar la administración del camping, que anteriormente se manejaba de forma manual.

- **Para Administradores:** Proveer una plataforma centralizada para realizar operaciones de Alta, Baja y Modificación (CRUD) sobre todas las entidades del sistema (socios, empleados, vehículos, zonas, garages, etc.).
- **Para Empleados:** Ofrecer un portal simple donde puedan consultar información relevante para sus tareas, como la lista de compañeros y las asignaciones de trabajo en las diferentes zonas.
- **Para Socios:** Brindar un portal de consulta donde puedan ver la información de sus vehículos, las compras de garages realizadas y las asignaciones de sus vehículos a los garages.

Descripción de los Actores y sus Roles

El sistema identifica tres actores (roles de usuario) con distintos niveles de permiso:

1. **ADMINISTRADOR:**
    a. **Descripción:** Es el superusuario del sistema. Tiene control total sobre todos los módulos.
    b. Roles/Funcionalidades:
        i. Gestión (CRUD completo) de Socios.
        ii. Gestión (CRUD completo) de Empleados.
        iii. Gestión (CRUD completo) de Zonas y Tipos de Vehículo.
        iv. Gestión (CRUD completo) de Garages.
        v. Gestión (CRUD completo) de Vehículos de socios.
        vi. Gestión (CRUD completo) de todas las asignaciones (Vehículo-Garage, Empleado-Zona, Compra-Garage).
2. **EMPLEADO:**
    a. **Descripción:** Es un usuario interno con permisos de solo lectura para información relevante a su trabajo.

   b. **Roles/Funcionalidades:**
      i. Consultar la lista de todos los empleados ("Compañeros").
      ii. Consultar la lista de todas las asignaciones de empleados a zonas.
3. **SOCIO:**
   a. **Descripción:** Es un cliente/propietario con permisos de solo lectura sobre la información general y la que le pertenece.
   b. **Roles/Funcionalidades:**
      i. Consultar la lista de todos los vehículos registrados en el sistema.
      ii. Consultar la lista de todas las asignaciones de vehículos a garages.
      iii. Consultar la lista de todas las compras de garages.

2. Arquitectura y Desarrollo

Descripción de las Capas del Sistema

El proyecto sigue una **Arquitectura de 3 Capas** (o N-Capas), un patrón estándar en el desarrollo con Spring Boot que asegura la separación de responsabilidades:

1. **Capa de Controlador (Controller Layer):**
   a. Es la puerta de entrada a la aplicación. Se divide en dos tipos:
   b. **`@Controller` (MVC):** Controladores que manejan las peticiones web y devuelven vistas HTML (plantillas de Thymeleaf). Son responsables de servir la interfaz de usuario.
   c. **`@RestController` (API):** Controladores que manejan peticiones de datos y devuelven JSON. Son consumidos por el JavaScript del frontend (AJAX/Fetch) para cargar datos dinámicamente sin recargar la página.
2. **Capa de Servicio (Service Layer):**
   a. Es el "cerebro" de la aplicación. Contiene toda la lógica de negocio.
   b. Coordina las operaciones, realiza validaciones (ej. "que no exista un socio con el mismo teléfono"), y maneja la conversión entre DTOs y Entidades.
   c. Desacopla al controlador de la base de datos. El controlador solo habla con el servicio.
3. **Capa de Repositorio (Repository Layer):**
   a. Es la capa de acceso a datos. Utiliza **Spring Data JPA**.
   b. Consiste en interfaces que extienden `JpaRepository`, lo que le da a la aplicación métodos CRUD (como `findAll`, `findById`, `save`, `delete`) sin necesidad de escribir SQL.

c. También define métodos de consulta "derivados" (ej. `existsByMatricula(...)`).

Explicación de la Estructura del Proyecto

El proyecto está organizado con una estructura estándar de Maven:

```
/src/main/java/com/acacias_del_mar/
│
├── config/
│   ├── SecurityConfig.java           (Configuración de Spring
Security: CSP, login, permisos)
│   └── CustomSuccessHandler.java     (Lógica de redirección por
Rol)
│
├── controller/
│   ├── api/
│   │   ├── SocioRestController.java  (Endpoints API para Socios)
│   │   └── ...                       (Otros RestControllers)
│   └── view/
│       ├── SocioViewController.java  (Controlador MVC para la vista
de Socios)
│       └── ...                       (Otros ViewControllers)
│
├── dto/
│   ├── SocioDTO.java                 (DTO de entrada para POST/PUT)
│   ├── SocioResponseDTO.java         (DTO de salida para GET)
│   └── ...                           (Otros DTOs)
│
├── entities/
│   ├── Socio.java                    (Entidad JPA mapeada a la
tabla)
│   └── ...                           (Otras Entidades)
│
├── mapper/
│   └── EntityMapper.java             (Interfaz de MapStruct para
todas las conversiones)
│
├── repository/
│   ├── SocioRepository.java          (Interfaz JPA para la tabla
Socios)
│   └── ...                           (Otros Repositorios)
│
```

```
├── services/
│   ├── SocioService.java            (Interfaz del Servicio)
│   └── impl/
│       └── SocioServiceImpl.java    (Implementación de la lógica
de negocio)
│
└── AcaciasDelMarApplication.java    (Clase principal
@SpringBootApplication)

/src/main/resources/
│
├── templates/
│   ├── admin/
│   │   └── socios.html              (Vista CRUD de Socios)
│   │   └── ...
│   ├── socio/
│   │   └── dashboard.html           (Vista de consulta de Socio)
│   ├── layout.html                  (Plantilla principal con
navbar y sidebar)
│   ├── login.html
│   └── register.html
│
└── application.properties           (Configuración de la BD,
puerto, etc.)
```

Descripción de las Tecnologías y Frameworks Utilizados

- **Java 21:** Versión del lenguaje de programación.
- **Spring Boot 3:** Framework principal que provee un servidor web embebido (Tomcat) e inyección de dependencias.
- **Spring Web (MVC):** Para construir la arquitectura de controladores que sirven tanto vistas (MVC) como datos (REST).
- **Spring Security 6:** Para manejar la autenticación (login) y autorización (permisos por rol), así como la protección de cabeceras (CSP).
- **Spring Data JPA (con Hibernate):** Para el mapeo Objeto-Relacional (ORM) y la capa de acceso a datos.
- **MySQL:** Motor de base de datos relacional.
- **Thymeleaf:** Motor de plantillas del lado del servidor. Se usa para construir el HTML dinámicamente, integrándose con Spring Security.

- **MapStruct:** Librería de "mapeo de beans" que genera automáticamente el código de conversión entre Entidades JPA y DTOs, limpiando la capa de servicio.
- **Bulma CSS:** Framework de CSS moderno (basado en Flexbox) para estilizar el frontend.
- **Font Awesome:** Librería de iconos utilizada en el frontend (navbar, botones, etc.).
- **JavaScript (Puro / Vanilla JS):** Utilizado en el frontend para realizar llamadas `fetch` (AJAX) a la API REST, manejar los modales de CRUD y actualizar la tabla dinámicamente.
- **Lombok:** Para reducir el código "boilerplate" (getters, setters, constructores) en las entidades y DTOs.
- **Maven:** Herramienta de gestión de dependencias y construcción del proyecto.

Explicación de las Clases Principales y sus Responsabilidades

- **`SecurityConfig.java`:** Es la clase más crítica. Define la seguridad de la aplicación.
  - Configura el `PasswordEncoder` (BCrypt).
  - Define el `UserDetailsService` (cómo Spring busca un usuario en nuestra tabla `usuarios`).
  - Configura el `SecurityFilterChain` (el "firewall"): define qué URLs son públicas (ej. `/login`), cuáles son privadas (`.anyRequest().authenticated()`), cómo funciona el `formLogin`, y la política de seguridad de contenidos (CSP).
- **`CustomAuthenticationSuccessHandler.java`:** Clase personalizada que se activa después de un login exitoso. Revisa el rol del usuario (`ROLE_ADMINISTRADOR`, `ROLE_EMPLEADO`) y lo redirige a su dashboard correspondiente.
- **`SocioRestController.java` (Ejemplo de API):**
  - Usa `@RestController` y `@RequestMapping("/api/socios")`.
  - Sus métodos devuelven `ResponseEntity` con DTOs de respuesta (`SocioResponseDTO`).
  - Utiliza `@PreAuthorize` para la seguridad a nivel de método (ej. `GET` es para `SOCIO` y `ADMIN`, pero `POST` es solo para `ADMIN`).
- **`SocioViewController.java` (Ejemplo de MVC):**
  - Usa `@Controller` y `@RequestMapping("/admin/socios")`.
  - Sus métodos devuelven un `String` (el nombre de la plantilla, ej. `"admin/socios"`).
  - Añade atributos al `Model` (como `currentPage`) para que Thymeleaf los use.

- **`SocioServiceImpl.java` (Ejemplo de Servicio):**
  - Implementa la interfaz `SocioService`.
  - Inyecta (`@Autowired`) el `SocioRepository` y el `EntityMapper`.
  - Contiene la lógica de negocio (ej. `if (repo.existsByTelefono(...))`).
  - Orquesta la operación: recibe un `SocioDTO`, lo convierte a `Socio` (usando el mapper), lo guarda, y devuelve un `SocioResponseDTO` (usando el mapper).
- **`EntityMapper.java`:**
  - Interfaz de MapStruct. Define los "contratos" de conversión (ej. `SocioResponseDTO toResponseDTO(Socio socio);`).
  - MapStruct genera la implementación automáticamente en tiempo de compilación.

3. Casos de Uso y Diagramas

Diagrama de Casos de Uso

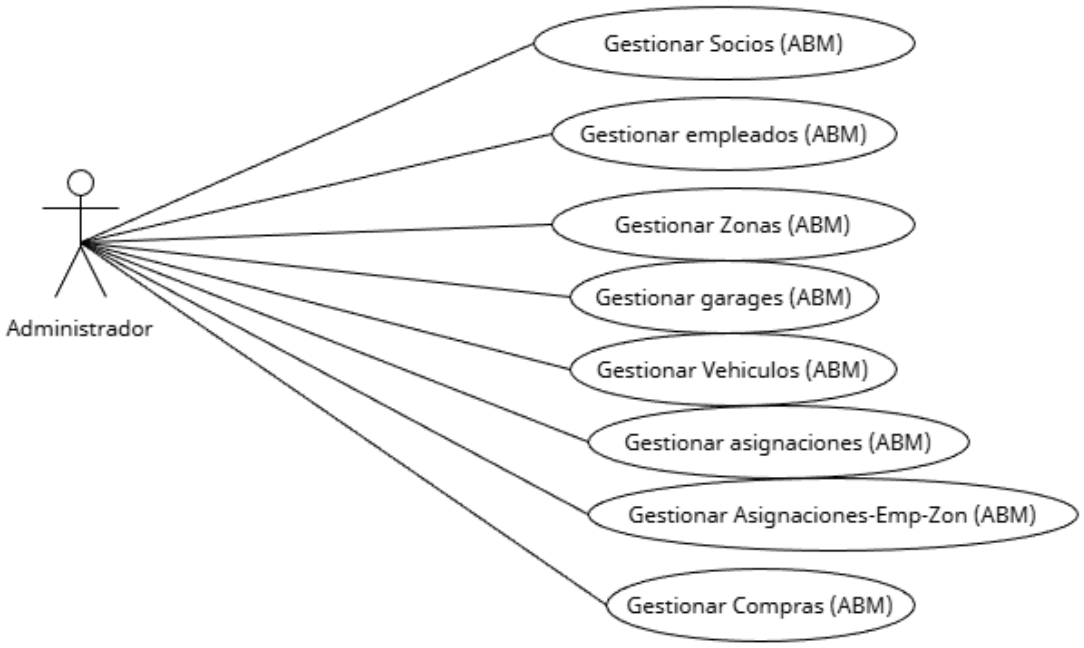
Breve Descripción de Cada Caso de Uso

- **Autenticarse:** (Común a todos) Los tres actores deben ingresar su `nombre` y `pass` en la vista `/login`. El sistema valida sus credenciales y rol.
- **Registrarse:** (Común a todos) Un usuario puede crear una nueva cuenta (con rol `SOCIO` o `EMPLEADO`) desde la vista `/register`.

Casos de Uso del ADMINISTRADOR:

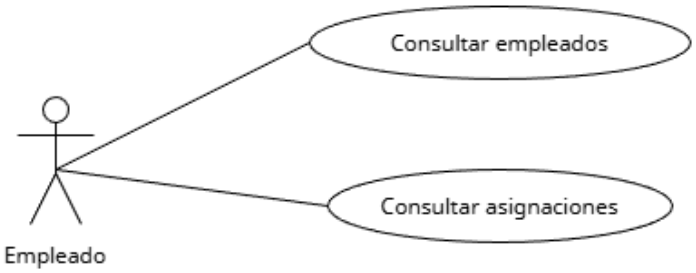
- **Gestionar Socios:** (CRUD) El admin puede ver, crear, editar y eliminar socios desde la vista `/admin/socios`.
- **Gestionar Empleados:** (CRUD) El admin puede ver, crear, editar y eliminar empleados desde la vista `/admin/empleados`.
- **Gestionar Zonas:** (CRUD) El admin puede ver, crear, editar y eliminar zonas y sus tipos de vehículo desde `/admin/zonas`.
- **Gestionar Garages:** (CRUD) El admin puede ver, crear, editar y eliminar garages (y asignarlos a una zona) desde `/admin/garages`.
- **Gestionar Vehículos:** (CRUD) El admin puede ver, crear, editar y eliminar vehículos (y asignarlos a un socio y tipo) desde `/admin/vehiculos`.
- **Gestionar Asignaciones (V-G):** (CRUD) El admin puede asignar, editar y eliminar la relación entre un Vehículo y un Garage.

- **Gestionar Asignaciones (E-Z):** (CRUD) El admin puede asignar, editar y eliminar la relación entre un Empleado y una Zona.
- **Gestionar Compras:** (CRUD) El admin puede registrar, editar y eliminar la compra de un Garage por un Socio.



Casos de Uso del EMPLEADO:

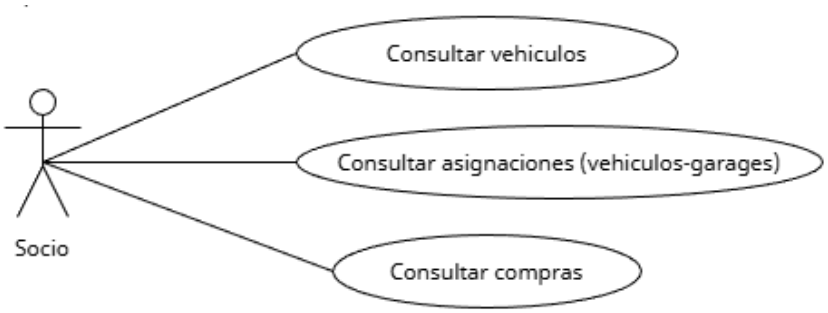
- **Consultar Compañeros:** (Lectura) El empleado accede a `/empleado/dashboard` y consulta una tabla con todos los empleados registrados.
- **Consultar Asignaciones E-Z:** (Lectura) El empleado accede a `/empleado/dashboard` y consulta una tabla con todas las asignaciones de empleados a zonas.



Casos de Uso del SOCIO:

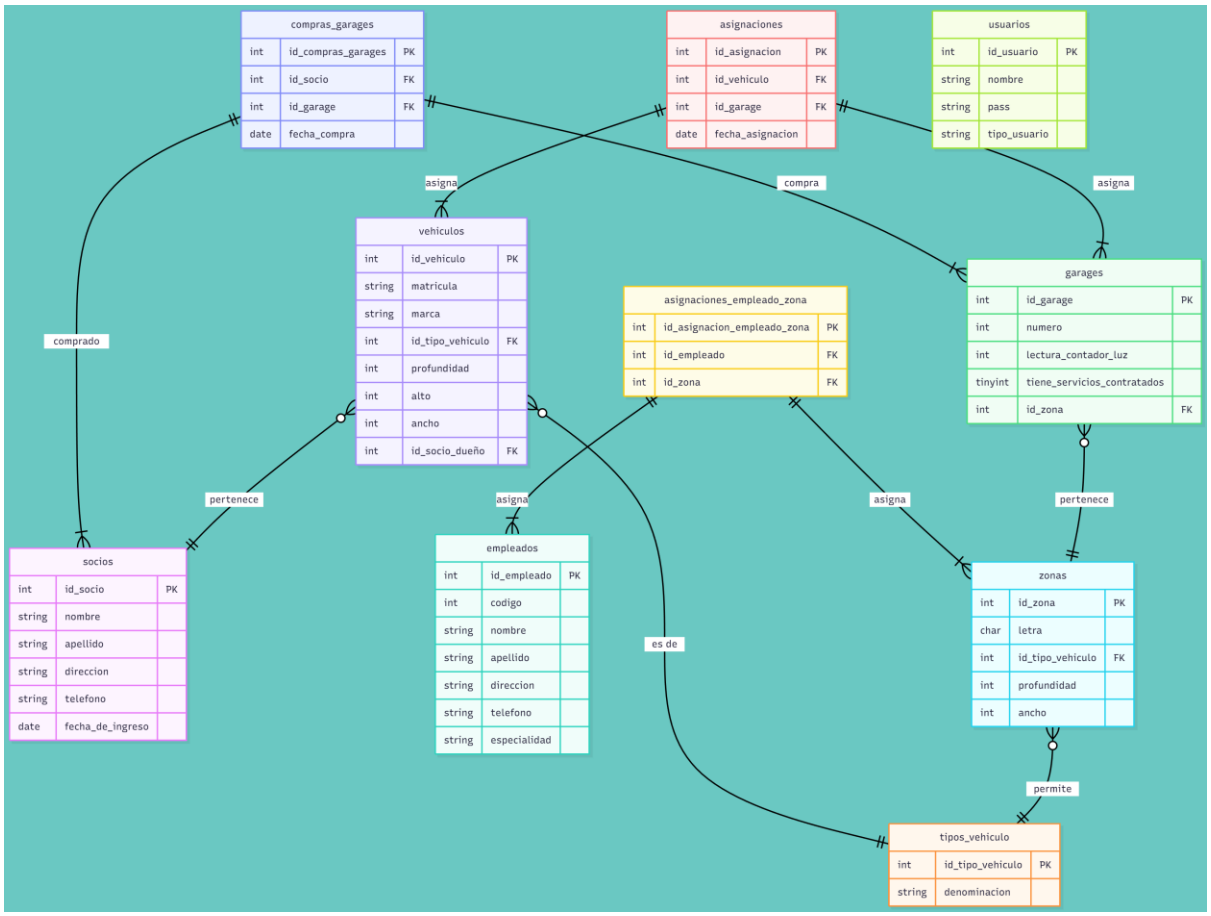
- **Consultar Vehículos:** (Lectura) El socio accede a `/socio/dashboard` y consulta una tabla con todos los vehículos del sistema.
- **Consultar Asignaciones V-G:** (Lectura) El socio accede a `/socio/dashboard` y consulta una tabla con todos los vehículos asignados a garages.

- **Consultar Compras:** (Lectura) El socio accede a `/socio/dashboard` y consulta una tabla con todos los garages que han sido comprados por socios.



4. Base de Datos

Diagrama Entidad-Relación



Descripción de las Tablas y Relaciones

- **usuarios:** Almacena los datos de login (`nombre`, `pass`, `tipo_usuario`). Es la tabla central para Spring Security.

- **socios:** (Tabla principal) Almacena los datos de los clientes/propietarios.
- **empleados:** (Tabla principal) Almacena los datos del personal.
- **tipos_vehiculo:** (Tabla catálogo) Almacena los tipos (ej. "Motorhome", "Trailer").
- **zonas:** (Tabla principal) Define las zonas del camping (ej. "Zona A").
    - *Relación:* Muchos-a-Uno con `tipos_vehiculo` (una zona permite un tipo de vehículo).
- **garages:** (Tabla principal) Define los espacios de garage.
    - *Relación:* Muchos-a-Uno con `zonas` (un garage pertenece a una zona).
- **vehiculos:** (Tabla principal) Almacena los vehículos.
    - *Relación:* Muchos-a-Uno con `socios` (un vehículo pertenece a un socio).
    - *Relación:* Muchos-a-Uno con `tipos_vehiculo` (un vehículo es de un tipo).
- **asignaciones:** (Tabla de unión) Resuelve la relación Muchos-a-Muchos entre `vehiculos` y `garages`. Almacena qué vehículo está en qué garage.
- **asignaciones_empleado_zona:** (Tabla de unión) Resuelve la relación Muchos-a-Muchos entre `empleados` y `zonas`. Almacena qué empleado trabaja en qué zona.
- **compras_garages:** (Tabla de unión) Resuelve la relación Muchos-a-Muchos entre `socios` y `garages`. Almacena qué socio ha comprado qué garage.

5. Manual de Usuario

Manual del Administrador

1. **Acceso:**
    a. Navegue a la URL del sitio. Será redirigido a `/login`.
    b. Ingrese sus credenciales de `ADMINISTRADOR` (ej. usuario: "admin", pass: "admin123").
    c. Tras el éxito, será redirigido a la primera página de gestión (ej. `/admin/socios`).
2. **Funcionalidades:**
    a. **Navegación:** Utilice el menú lateral (sidebar) izquierdo para navegar entre las 8 páginas de gestión (Socios, Empleados, Zonas, Garages, Vehículos, Asignaciones V-G, Asignaciones E-Z, Compras).
    b. **Crear (Alta):** En cualquier página de gestión, haga clic en el botón azul "Crear" (ej. "Crear Socio") en la esquina superior derecha. Se abrirá un modal (ventana emergente).
    c. **Formulario de Creación:** Rellene los campos. Los campos que dependen de otras tablas (ej. "Zona" en el formulario de "Garage") se cargarán como listas desplegables. Haga clic en "Guardar Cambios".

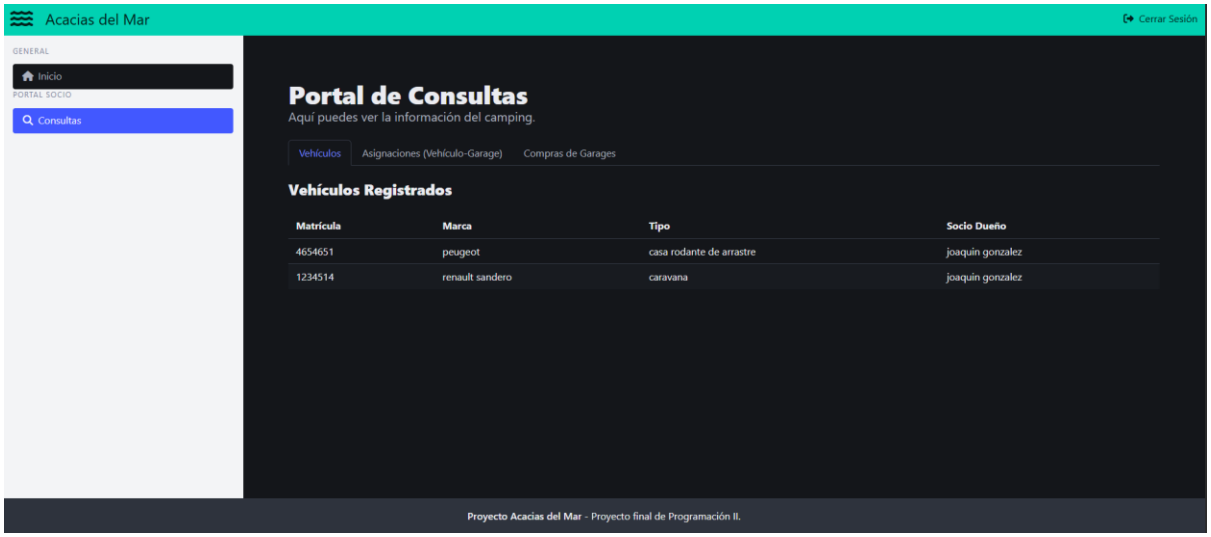
d. **Editar (Modificación):** En la tabla, haga clic en el botón azul de "Editar" (icono de lápiz) en la fila deseada. El mismo modal se abrirá, pero precargado con los datos de esa fila. Modifique los datos y haga clic en "Guardar Cambios".
e. **Eliminar (Baja):** En la tabla, haga clic en el botón rojo de "Eliminar" (icono de basura). Se abrirá un modal de confirmación. Haga clic en "Eliminar" para confirmar la baja permanente.

Manual del Empleado

1. **Acceso:**
    a. Navegue a `la URL del sitio`.
    b. Ingrese sus credenciales de EMPLEADO (creadas desde `/register` o por un admin).
    c. Tras el éxito, será redirigido a su portal en `/empleado/dashboard`.
2. **Funcionalidades:**
    a. **Navegación:** La página principal (`/empleado/dashboard`) contiene una interfaz de pestañas (Tabs).
    b. **Consultar Compañeros:** Haga clic en la pestaña "Compañeros" para ver una lista de solo lectura de todos los empleados y su especialidad.
    c. **Consultar Asignaciones:** Haga clic en la pestaña "Asignaciones de Zona" para ver una lista de solo lectura de qué empleados están asignados a qué zonas de trabajo.

Manual del Socio

1. **Acceso:**
    a. Navegue a `la URL del sitio`.
    b. Ingrese sus credenciales de `SOCIO` (creadas desde `/register` o por un admin).
    c. Tras el éxito, será redirigido a su portal en `/socio/dashboard`.
2. **Funcionalidades:**
    a. **Navegación:** La página principal (`/socio/dashboard`) contiene una interfaz de pestañas (Tabs).
    b. **Consultar Vehículos:** En la pestaña "Vehículos", puede ver una lista de solo lectura de todos los vehículos del sistema, su tipo y el socio dueño.
    c. **Consultar Asignaciones:** En la pestaña "Asignaciones", puede ver qué vehículos están actualmente asignados a qué garages.
    d. **Consultar Compras:** En la pestaña "Compras de Garages", puede ver un historial de qué socios han comprado qué garages.


Acacias del Mar
Cerrar Sesión
GENERAL
Inicio
PORTAL SOCIO
Consultas
Portal de Consultas
Aquí puedes ver la información del camping.
Vehículos
Asignaciones (Vehículo-Garage)
Compras de Garages
Vehículos Registrados
Matrícula
Marca
Tipo
Socio Dueño
4654651
peugeot
casa rodante de arrastre
joaquin gonzalez
1234514
renault sandero
caravana
joaquin gonzalez
Proyecto Acacias del Mar - Proyecto final de Programación II.