Documentación del Sistema Gestor de Acceso - Zona Acme

Luis Nicolás Mendez Palacios
José Daniel Carvajal Montañez
Laura Sofía Amaya Casado

Campuslands - 2024

Bienvenido al sistema Gestor de Entradas y Salidas - Zona Acme, este proyecto busca permitir gestionar de manera dinámica y sencilla las entradas y salidas de trabajadores en una zona empresarial, facilitando la labor de guardias, supervisores y funcionarios mediante un sistema centralizado.

Usuarios

1. **Superusuario:** Gestiona empresas y configura la base de datos.
2. **Guardias**: Controlan el acceso de empleados y vehículos.
3. **Supervisores**: Monitorean registros y gestionan usuarios.
4. **Funcionarios**: Gestionan personas y realizan salidas manuales.

Consiguiendo así la simplificación de la gestión manual de entradas y salidas, permitiendo un monitoreo en tiempo real a través de concurrencia.

Tecnologías utilizadas

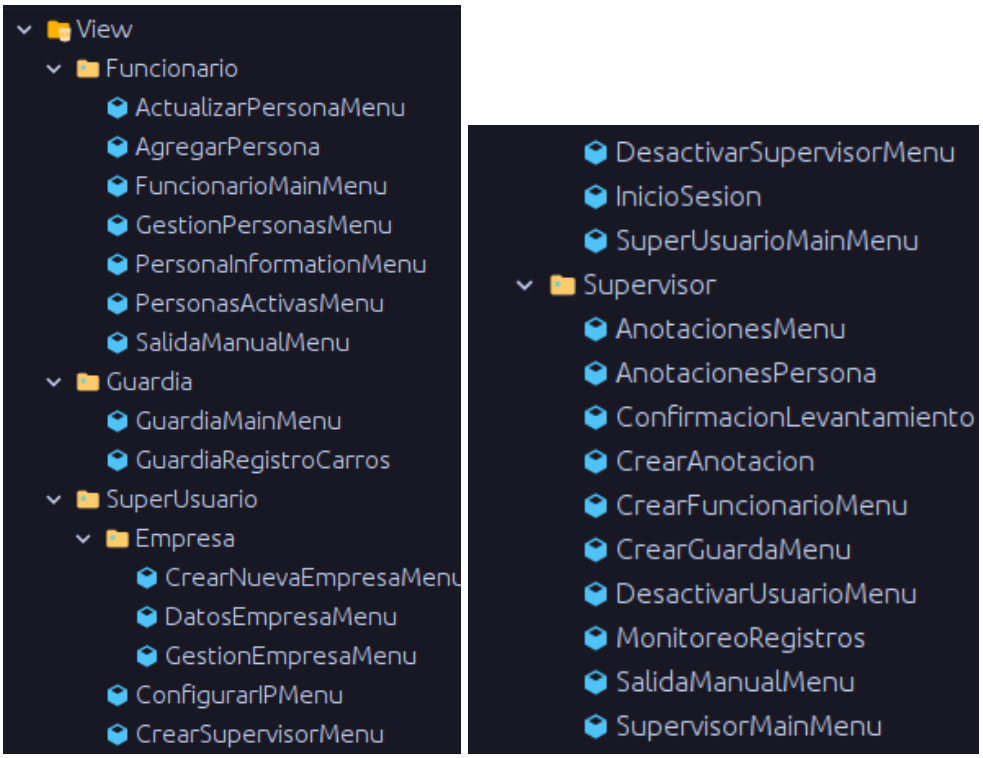
- **Lenguaje**: Java
- **IDE**: IntelliJ IDEA
- **Base de datos**: MySQL
- **Arquitectura**: MVC con patrón DAO
- **Control de concurrencia**: Mecanismos multihilo de Java
- **Patrones Adicionales**: Singleton, Proxy, Factory Method, Strategy.

Arquitectura del Sistema

Patrón MVC (Modelo-Vista-Controlador)

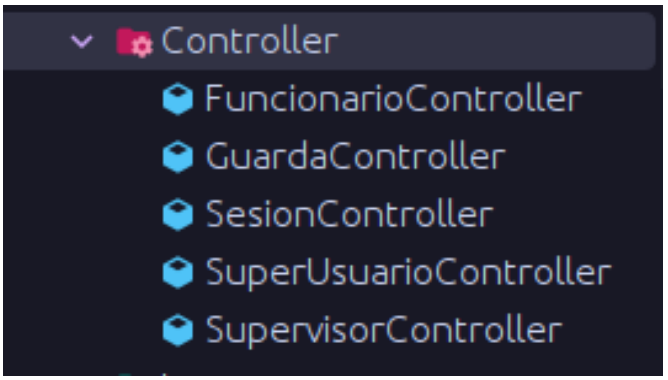
Modelo: Implementado en el paquete Repository.DAO y las clases Impl encargadas de las implementaciones lógicas. Aquí se encuentran las distintas entidades y métodos que representan la lógica del negocio y la base de datos.

Vista: Todas las interfaces gráficas están organizadas en la carpeta View organizadas para cada tipo de usuario.



Service: Capa intermedia entre el modelo y el controlador el cual permite organizar las operaciones de cada clase del modelo.

Controlador: Implementaciones y lógica de negocio que está en el paquete Repository.Impl.



Patrón DAO (Data Access Object)
Organiza el acceso a los datos y la lógica de la base de datos.

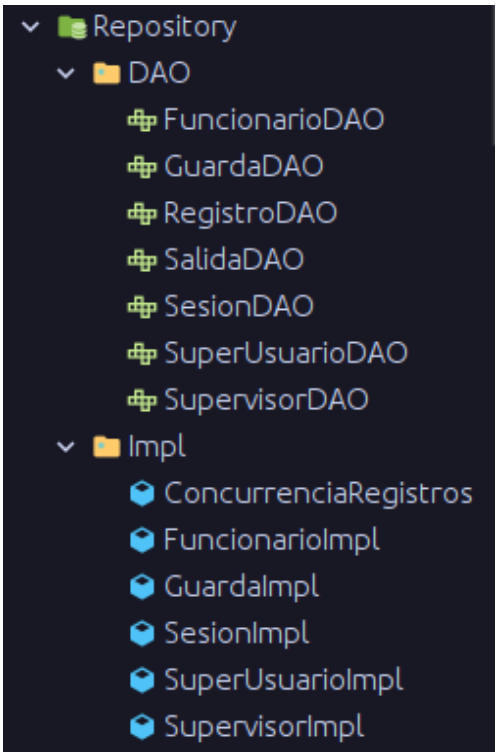
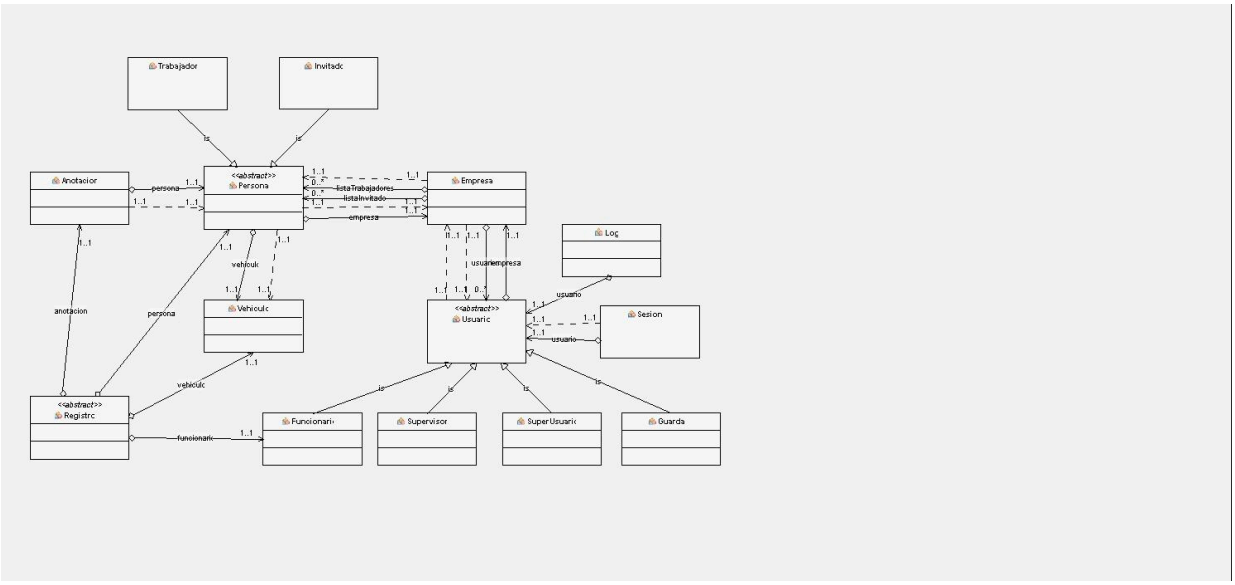


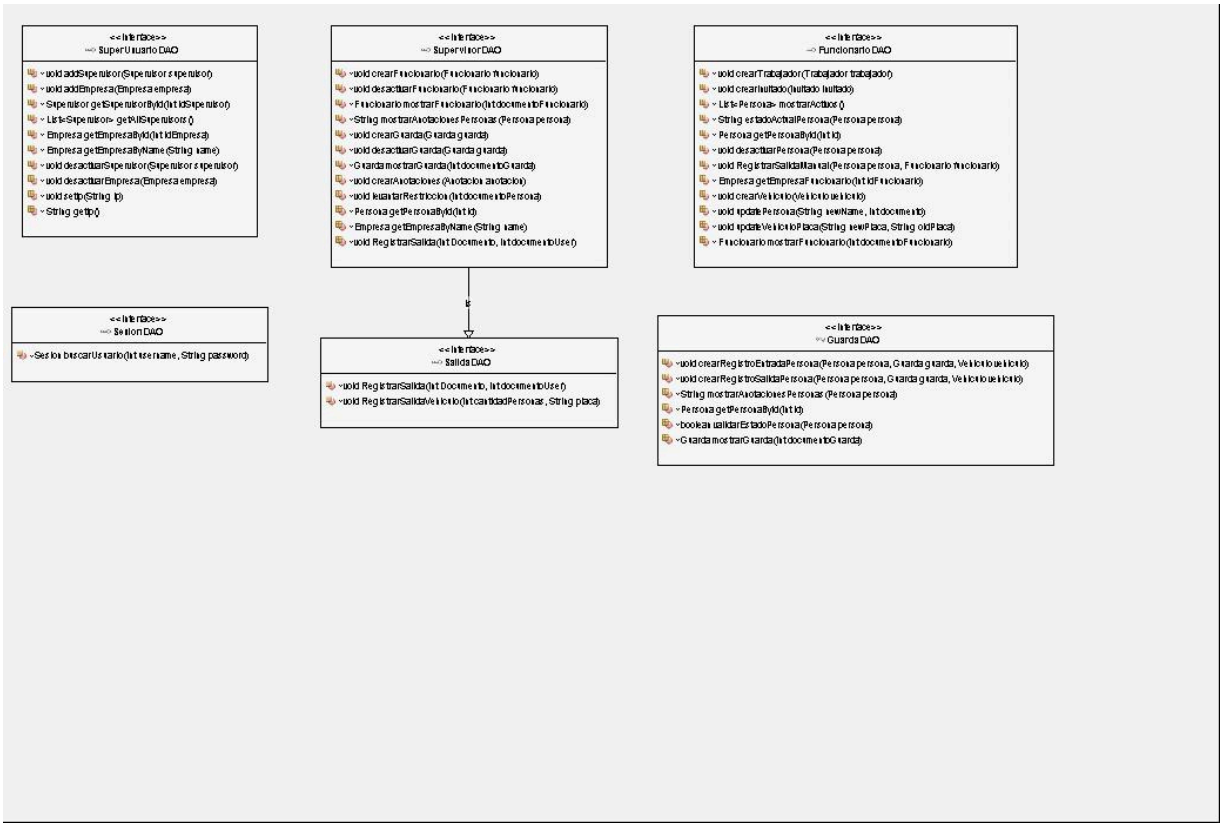
Diagrama de Clase UML

Se decidió dividir el diagrama de clases en los paquetes que componen el proyecto:

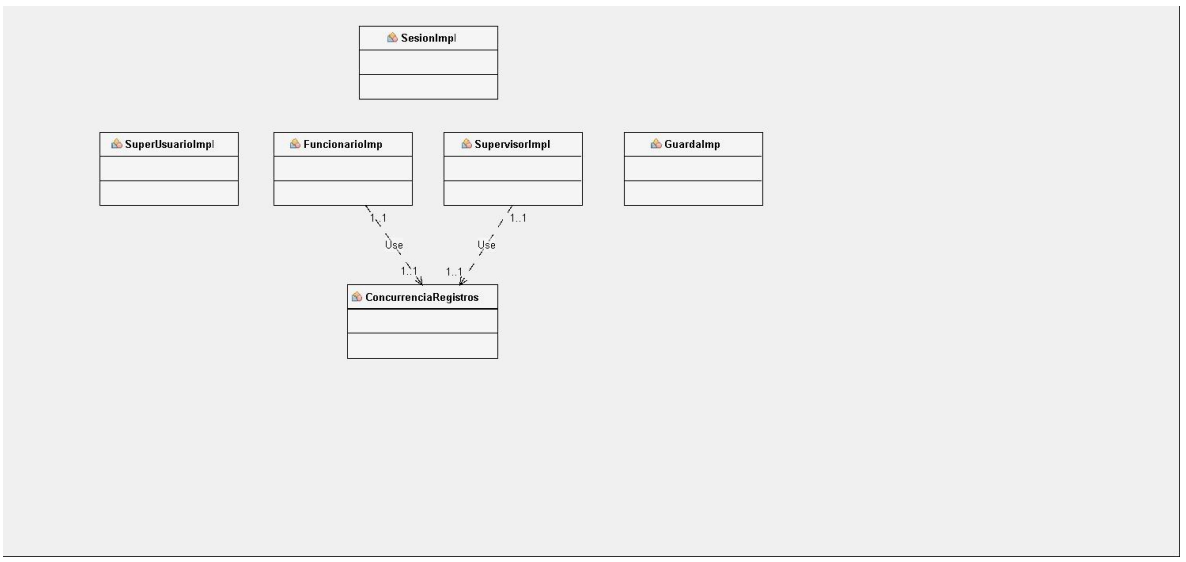
Modelo



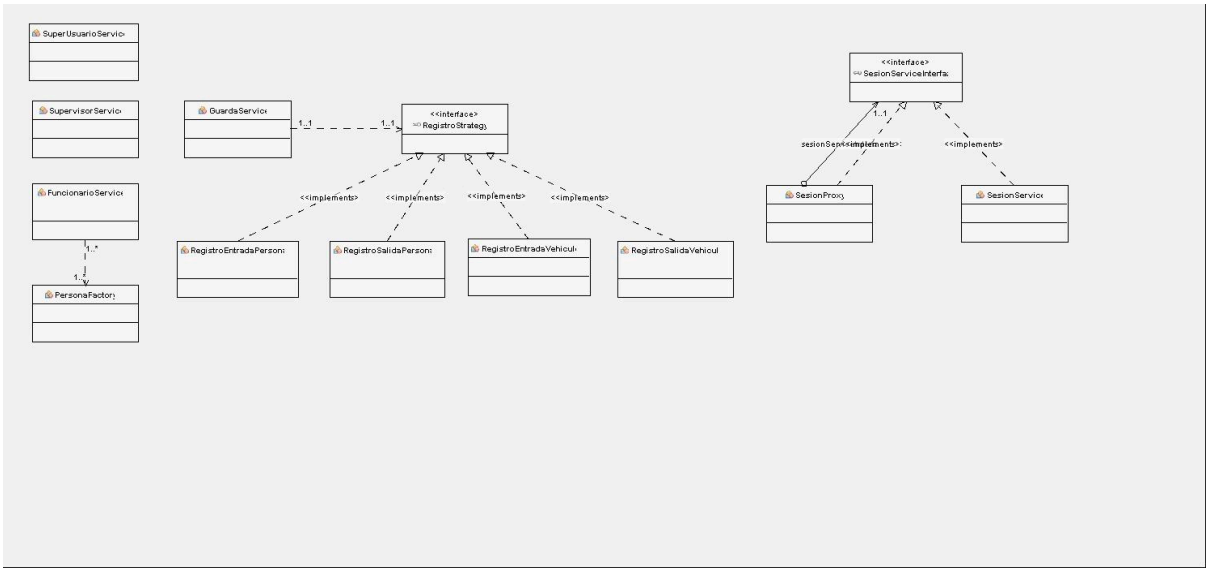
DAO



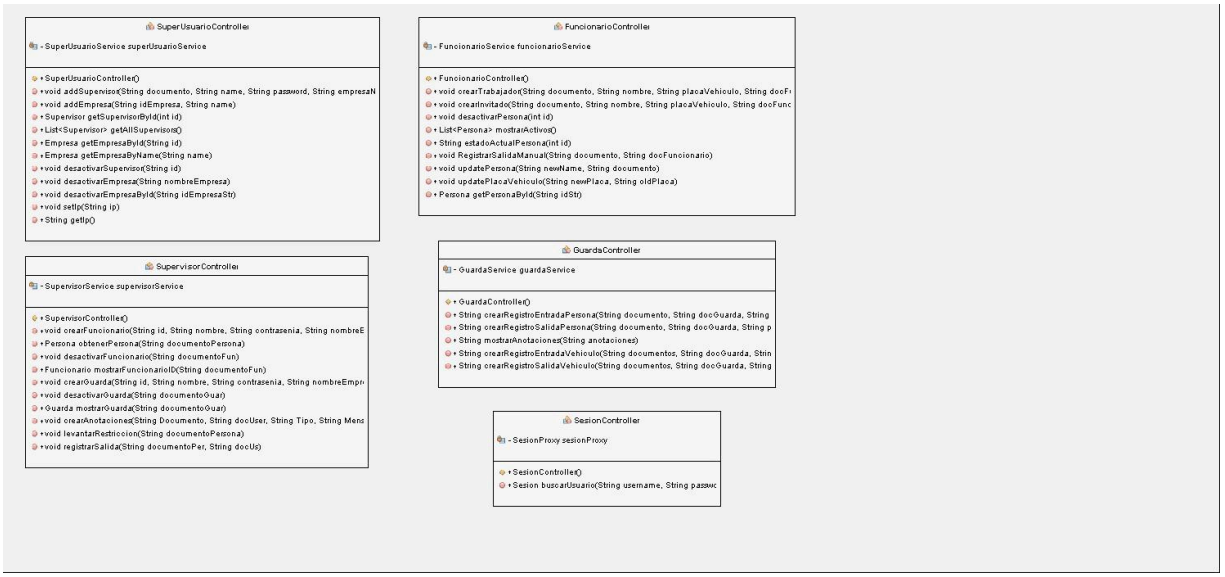
Implementación



Strategy y Service



Controlador



2. Sql (Base de Datos)

Archivos que estructuran y controlan la base de datos:
ddl.sql: Creación de tablas.
dql.sql: Consultas de selección.
test.sql: Scripts de prueba para la base de datos.
triggers2.sql: Triggers automatizados.

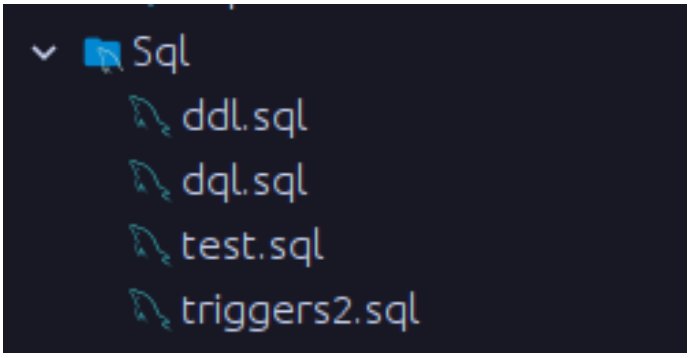
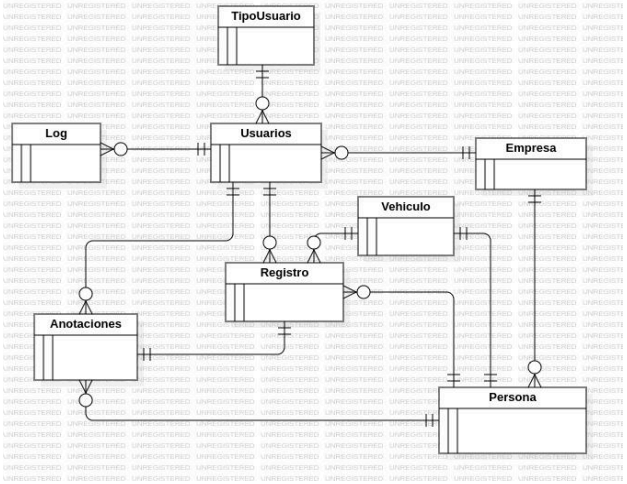
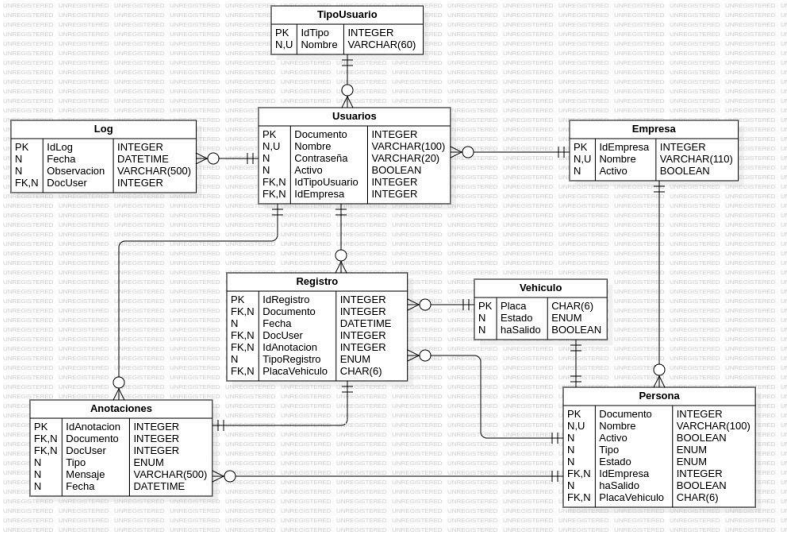


Diagrama Entidad- Relación



Primer nivel



Tercer nivel

Consultas de prueba

INSERT INTO
Registro`(`Documento`,`Fecha`,`DocUser`,`IdAnotacion`,`TipoRegistro`,`PlacaVehiculo`)
VALUES
(1102359291, "2024-12-16 08:55:46.00", 1102359888, NULL, "Salida", NULL),
(1102359291, "2024-12-16 10:55:46.00", 1102359888, NULL, "Entrada", NULL),
(1112223334, "2024-12-16 10:55:46.02", 1102359888, NULL, "Salida", NULL),
(1231231231, "2024-12-16 11:55:46.00", 1102359888, NULL, "Salida", NULL);

3. Funcionalidades del Sistema

SuperUsuario:

1. Crear y configurar empresas en la base de datos.

2. Administrar y desactivar supervisores.
3. Administrar la base de datos.

Supervisor:

1. Crear y desactivar usuarios (funcionarios y guardias).
2. Monitorear registros de entradas y salidas en tiempo real.
3. Realizar anotaciones sobre incidentes.

Funcionario:

1. Gestionar y actualizar personas registradas en la zona en la empresa.
2. Monitorear registros de entradas y salidas en tiempo real.
3. Registrar manualmente salidas de empleados.

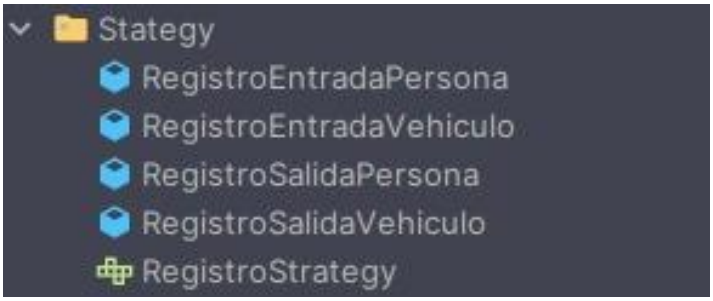
Guardia:

1. Permitir o denegar accesos de trabajadores y vehículos.

4. Patrones de diseño

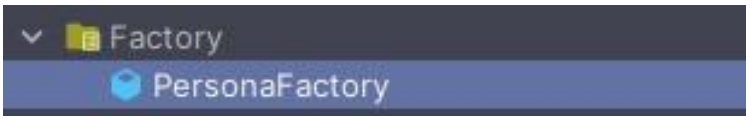
Singleton: Utilizado para crear una sola instancia de la clase de conexión a la base de datos.

Strategy: Utilizado para encapsular la lógica de los métodos de registro de entrada y salida de personas y vehículos, para que el servicio del guarda tenga sólo la estrategia y el controller lo llame con la estrategia específica dependiendo la interfaz en la que se encuentre.



Proxy: Utilizado para brindar una capa adicional de seguridad en el inicio de sesión de usuarios.

Factory Method: utilizada para crear clases hijas (trabajador e invitado) de la clase Factory persona.



Con esto se resuelve efectivamente la gestión de acceso en una zona empresarial, se implementaron los principios SOLID por medio de la distribución de clases, buscando que tuvieran responsabilidad única y fueran extensibles, Las siguientes mejoras podrían implementarse en futuras versiones:

- Generación automática de reportes en PDF.
- Sistema de notificaciones en tiempo real para supervisores.
- Integración de reconocimiento facial o tarjetas RFID para mayor automatización.