Desarrollo del Caso de Uso "Gestionar Viviendas"

Este documento detalla el desarrollo e implementación del caso de uso "Gestionar Viviendas" para la aplicación "Anti-Elfec", utilizando un entorno de consola en Java (como BlueJ o Visual Studio Code con JDK).

1. Contexto del Caso de Uso

El caso de uso "Gestionar Viviendas" es fundamental para la aplicación "Anti-Elfec", ya que permite a los usuarios registrados administrar las diferentes propiedades donde desean monitorear y controlar el consumo energético. Según la documentación de casos de uso proporcionada, este caso de uso engloba las funcionalidades de:

- **Añadir vivienda:** Permite al usuario registrar una nueva propiedad.
- **Modificar vivienda:** Permite actualizar los detalles de una vivienda existente.
- **Eliminar vivienda:** Permite desvincular una vivienda de la cuenta del usuario.

2. Diseño del Modelo (Clases)

Para implementar esta funcionalidad, se definieron dos clases principales en el paquete com.umss.antielfec.model:

a. Usuario.java

Esta clase representa al actor principal del sistema, el usuario que interactúa con la aplicación. Aunque para esta implementación de consola se usa de forma simplificada (simulando un usuario logueado), es crucial para futuras expansiones y la asociación de viviendas.

Atributos clave:

- id: Identificador único del usuario.
- nombre: Nombre del usuario.
- email: Correo electrónico del usuario (para autenticación).
- contrasena: Contraseña del usuario.

b. Vivienda.java

Esta clase modela una vivienda que el usuario puede gestionar.

Atributos clave:

- id: Identificador único de la vivienda.
- nombre: Nombre descriptivo de la vivienda (ej. "Mi Casa", "Departamento de

Verano").
- direccion: Dirección física de la vivienda.
- idUsuario: Un identificador que vincula la vivienda a un Usuario específico.

3. Lógica de Negocio (Servicio)

La lógica central para la gestión de viviendas se encapsula en la clase ViviendaService.java, ubicada en el paquete com.umss.antielfec.service. Este enfoque sigue el principio de separación de responsabilidades, manteniendo la lógica de negocio alejada de la interfaz de usuario.

a. ViviendaService.java

Este servicio gestiona una colección de objetos Vivienda en memoria (una ArrayList en este caso, que simula una base de datos simple). Incluye los siguientes métodos principales:

- **añadirVivienda(String nombre, String direccion, int idUsuario):**
  - Crea una nueva instancia de Vivienda.
  - Asigna un ID único a la nueva vivienda.
  - Valida que no exista otra vivienda con el mismo nombre para el *mismo usuario* para evitar duplicados, un requisito funcional importante.
  - Añade la vivienda a la lista interna.
  - Muestra un mensaje de confirmación o error en consola.
- **modificarVivienda(int idVivienda, String nuevoNombre, String nuevaDireccion, int idUsuario):**
  - Busca la vivienda por su idVivienda y idUsuario (asegurando que el usuario solo modifique sus propias viviendas).
  - Permite actualizar el nombre y/o la direccion.
  - Realiza una validación para asegurar que el nuevoNombre (si se cambia) no genere un duplicado con otra vivienda del mismo usuario.
  - Muestra un mensaje de confirmación o error.
- **eliminarVivienda(int idVivienda, int idUsuario):**
  - Elimina la vivienda de la lista interna, buscando por idVivienda y idUsuario.
  - Muestra un mensaje de confirmación o error.
- **obtenerViviendasPorUsuario(int idUsuario):**
  - Retorna una lista de todas las viviendas asociadas a un idUsuario específico. Utilizado para mostrar las viviendas al usuario.
- **obtenerViviendaPorId(int idVivienda, int idUsuario):**
  - Busca y retorna una vivienda específica por su ID y el ID del usuario propietario.

4. Interfaz de Usuario (Consola)

La interacción con el usuario se implementa en la clase principal AntiElfecApp.java, que reside en el paquete raíz com.umss.antielfec. Esta clase actúa como la vista y el controlador en un patrón simple para una aplicación de consola.

a. AntiElfecApp.java

- **main(String[] args):** El punto de entrada de la aplicación.
  - Inicializa el ViviendaService y un Scanner para la entrada del usuario.
  - Simula un usuarioLogueado (con ID 1) para permitir las operaciones.
  - Llama al mostrarMenuPrincipal().
- **mostrarMenuPrincipal():** Presenta las opciones generales (Gestionar Viviendas, Salir).
- **menuGestionarViviendas():** Es el corazón de la interacción para este caso de uso. Permite al usuario elegir entre:
  - Ver mis Viviendas: Llama a verMisViviendas().
  - Añadir Vivienda: Llama a añadirVivienda().
  - Modificar Vivienda: Llama a modificarVivienda().
  - Eliminar Vivienda: Llama a eliminarVivienda().
  - Volver al Menú Principal.
- **Métodos específicos para cada operación CRUD:**
  - verMisViviendas(): Muestra las viviendas del usuarioLogueado utilizando viviendaService.obtenerViviendasPorUsuario().
  - añadirVivienda(): Solicita al usuario el nombre y la dirección de la nueva vivienda y llama a viviendaService.añadirVivienda().
  - modificarVivienda(): Pide el ID de la vivienda a modificar, luego el nuevo nombre y dirección (opcionales), y llama a viviendaService.modificarVivienda().
  - eliminarVivienda(): Solicita el ID de la vivienda a eliminar y pide confirmación antes de llamar a viviendaService.eliminarVivienda().
- **leerEntero():** Una utilidad para manejar la entrada de números enteros, incluyendo validación básica para evitar errores de formato.

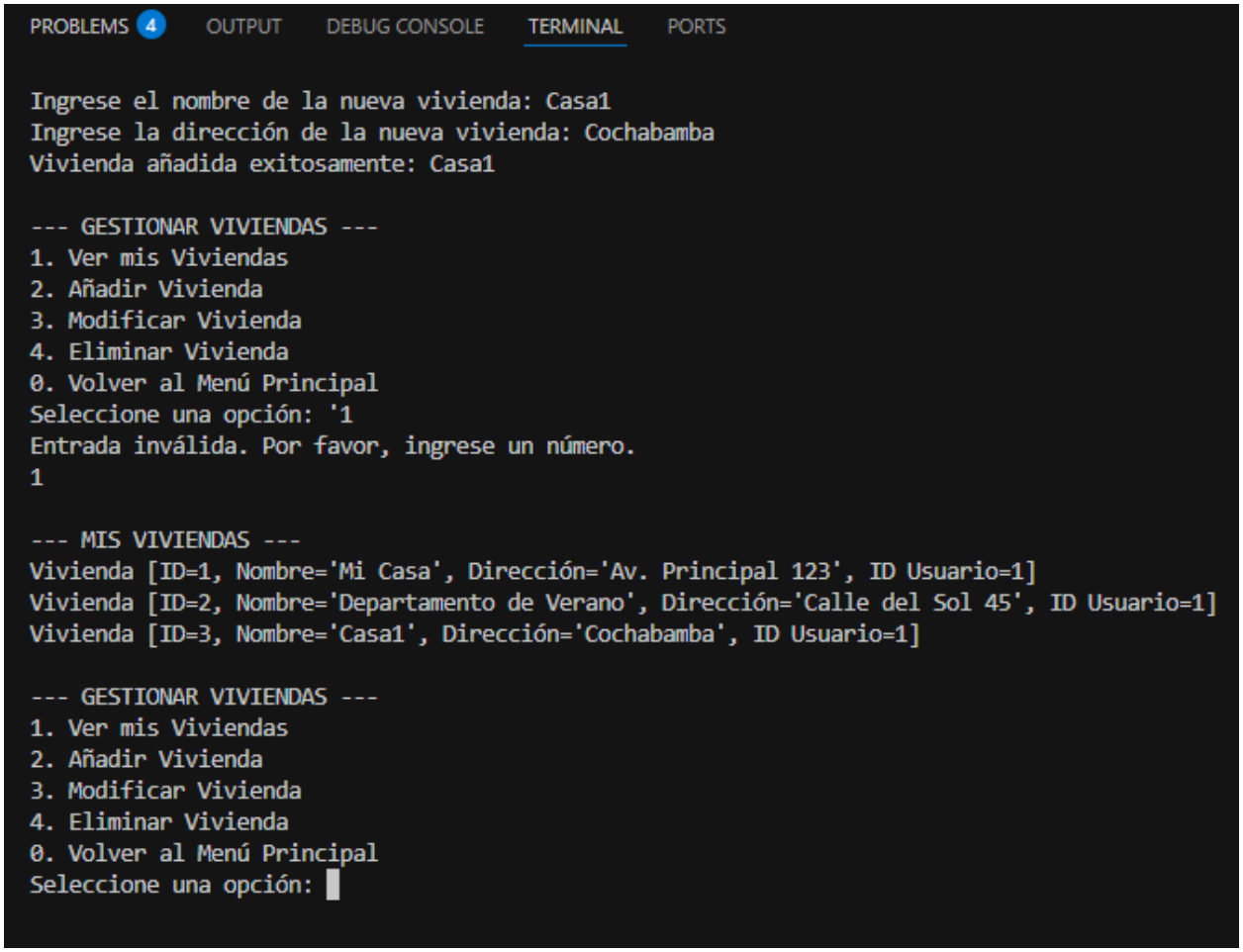
5. Funcionamiento y Evidencia

La captura de pantalla de BlueJ muestra claramente el flujo de ejecución:

1. **Inicio:** La aplicación simula el inicio de sesión y muestra el menú principal.
2. **Acceso a "Gestionar Viviendas":** El usuario selecciona la opción 1.
3. **Visualización Inicial:** Se muestran las viviendas precargadas ("Mi Casa" y "Departamento de Verano").

4. **Regreso al menú:** El usuario selecciona "0" para volver al menú principal.
5. **Salida:** El usuario selecciona "0" para salir de la aplicación.

Este resultado confirma que las clases Usuario, Vivienda, ViviendaService y AntiElfecApp están correctamente integradas y que las operaciones básicas del caso de uso "Gestionar Viviendas" se ejecutan como se esperaba en un entorno de consola.



6. Próximos Pasos

Este prototipo de consola es un excelente punto de partida. Los siguientes pasos para un desarrollo más robusto incluirían:

- **Persistencia de Datos Real:** Integrar una base de datos (como MySQL o PostgreSQL, como mencionaste en tus diagramas de despliegue) para que los datos no se pierdan al cerrar la aplicación. Esto implicaría el uso de JDBC o un ORM como Hibernate.
- **Interfaz Gráfica de Usuario (GUI):** Desarrollar una interfaz de usuario visual (con JavaFX, Swing, o una aplicación móvil real con Android/Kotlin) para mejorar la experiencia del usuario.
- **Gestión de Dispositivos:** Expandir la lógica para integrar el registro, control y monitoreo de dispositivos inteligentes dentro de cada vivienda.
- **Manejo de Errores Avanzado:** Implementar un manejo de excepciones más detallado y logging para situaciones inesperadas.
- **Pruebas Unitarias:** Escribir pruebas automatizadas para cada método del ViviendaService para asegurar su correcto funcionamiento.