Proyecto Final ED

Idea → Desarrollar una aplicación de gestión.

Orden del desarrollo:

1. Creación ERS
2. Creación de diagramas
3. Plantilla del proyecto (solo las clases)

Documentación:

1. Usar javadoc
2. Explicar cómo funciona la interfaz gráfica
3. Diseñar un conjunto de pruebas de unidad
4. Alojarlo en GitHub
5. Planificar el desarrollo con Scrum

Entrega: Apariencia profesional

Entregar memoria sobre dificultades.

INDICE

Identidad del proyecto (se adecúa a lo que se pide) - 0,5 puntos

Un sistema para comprar, descargar y gestionar videojuegos digitales al más puro estilo de Steam.

E.R.S - 1 punto

Contexto:

Tenemos que desarrollar una web y una aplicación que permita a los usuarios comprar, descargar y gestionar videojuegos digitales (propios), además también debe permitir solicitar un reembolso al soporte, siempre y cuando cumpla unos requisitos. No será necesario ninguna licencia de pago para su uso.

Los desarrolladores deben poder publicar sus propias creaciones y a su vez eliminarlas de la tienda.

Por último, los administradores deben poder gestionar a los usuarios y los videojuegos de la tienda. Además, pueden acceder a las peticiones del soporte.

Objetivos:

Tiene que permitir a los usuarios registrarse e iniciar sesión con sus credenciales.

Se tiene que ver el catálogo de juegos comprables.

Dar acceso a la compra de los videojuegos del catálogo.

Mostrar los juegos ya adquiridos en una biblioteca accesible para facilitar su instalación.

Permitir descargar, instalar o desinstalar los juegos ya adquiridos.

Dar la opción de publicar videojuegos a los usuarios. (con un coste de 100€)

Un sistema de comunidad para poder comunicarse por vía oral o escrita con otros usuarios.

Requisitos Funcionales:

Registro e inicio de sesión.

Compra de videojuegos y ver la biblioteca personal.

Descarga de juegos.

Gestión administrativa.

Requisitos No Funcionales:

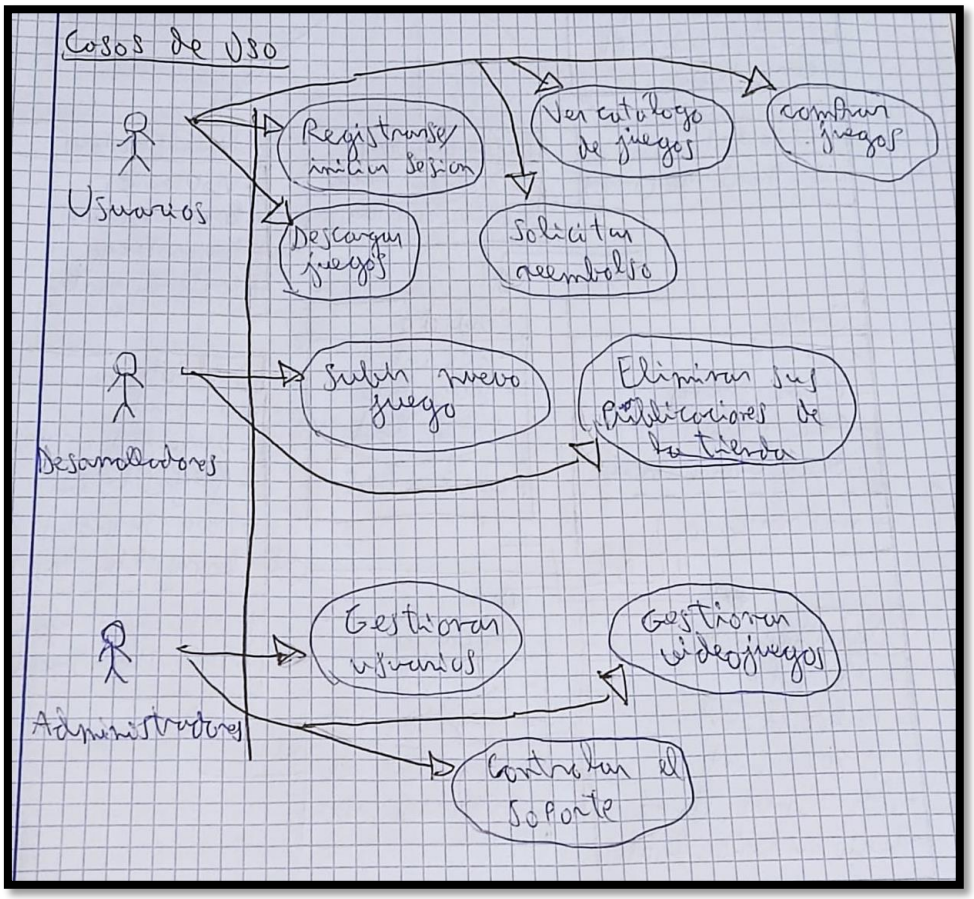
Uso de tecnologías libres.

Seguridad de datos.

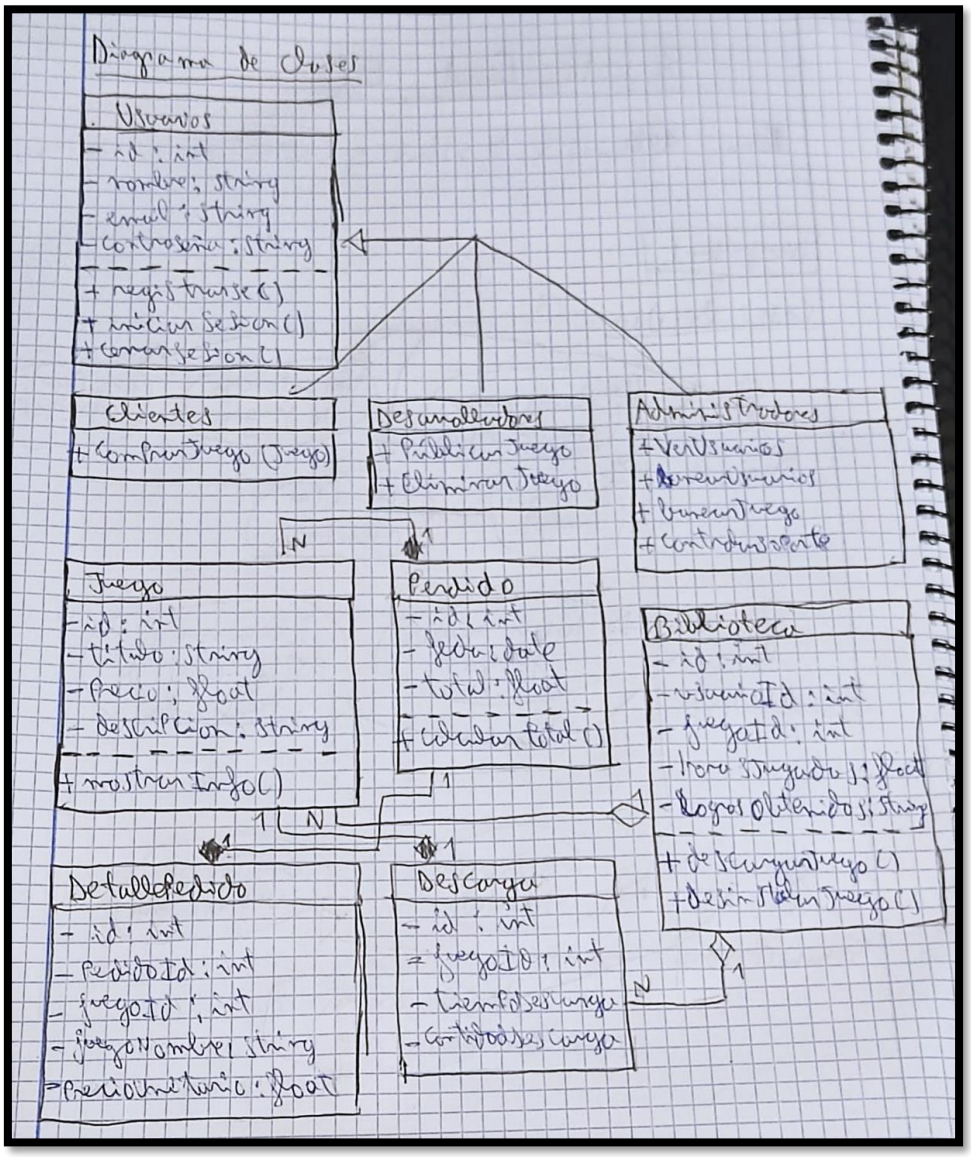
Interfaz clara y simple.

Estabilidad.

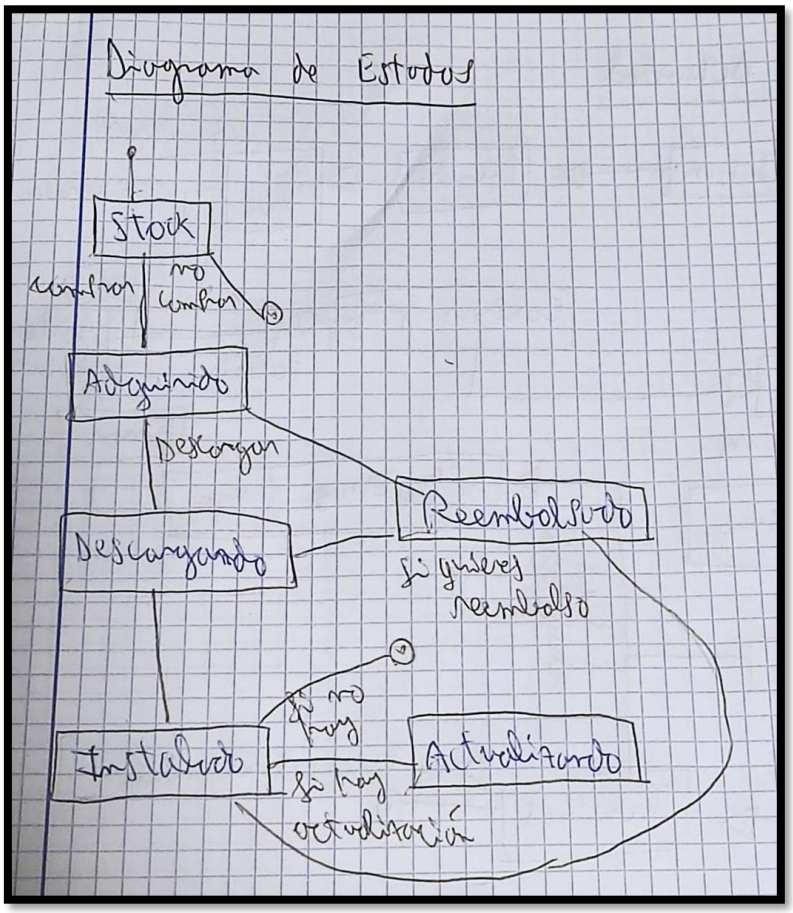
Diagramas: Casos de uso - 0,5 puntos



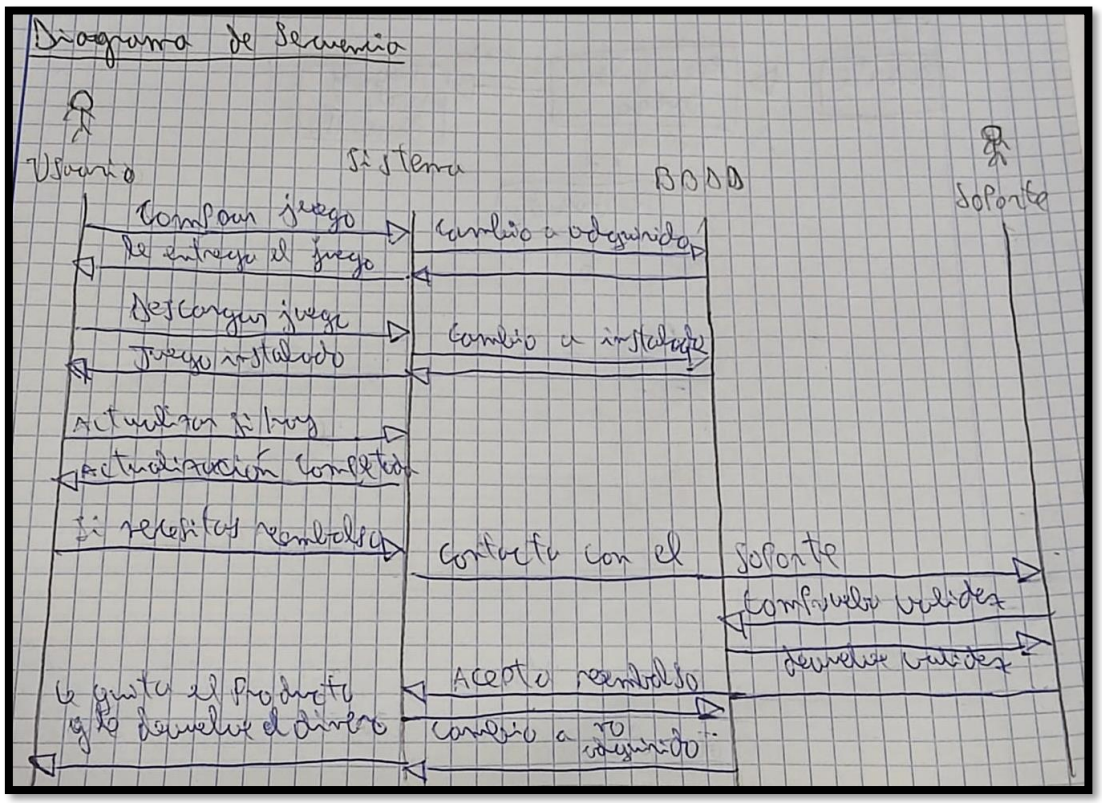
Diagramas: Clases - 0,5 puntos



Diagramas: Estados - 0,5 puntos



Diagramas: Secuencia - 0,5 puntos



Documentación de la interfaz - 1 punto

Estructura de la Interfaz

1. Pantalla de Inicio de Sesión / Registro

- **Componentes**:
  - Campos: Email, Contraseña
  - Botones: Iniciar sesión, Registrarse
- **Funciones**:
  - Verifica credenciales y redirige según el tipo de usuario
  - Permite registro como Cliente, Desarrollador o Administrador

2. Menú Principal

- Se adapta según el tipo de usuario:
  - Cliente: Tienda, Biblioteca, Descargas, Perfil
  - Desarrollador: Mis Juegos, Publicar Juego, Estadísticas
  - Administrador: Usuarios, Soporte, Juegos, Reportes

3. Tienda

- **Componentes**:
  - Barra de búsqueda
  - Filtros (categoría, precio, orden)
  - Lista de juegos (con título, miniatura, precio y botón "Comprar")
- **Funciones**:
  - Ver detalle del juego
  - Comprar juego (añadir a pedido)
  - Descargar si ya está comprado

4. Detalle del Juego

- **Componentes**:
  - Información: título, descripción, precio, valoraciones, imágenes
  - Botones: Comprar, Descargar (si ya se posee)
- **Funciones**:
  - Muestra información detallada del juego
  - Permite la compra o la descarga

5. Biblioteca del Usuario

- **Componentes**:
  - Lista de juegos adquiridos

- Opciones: Jugar, Descargar, Desinstalar, Ver Logros

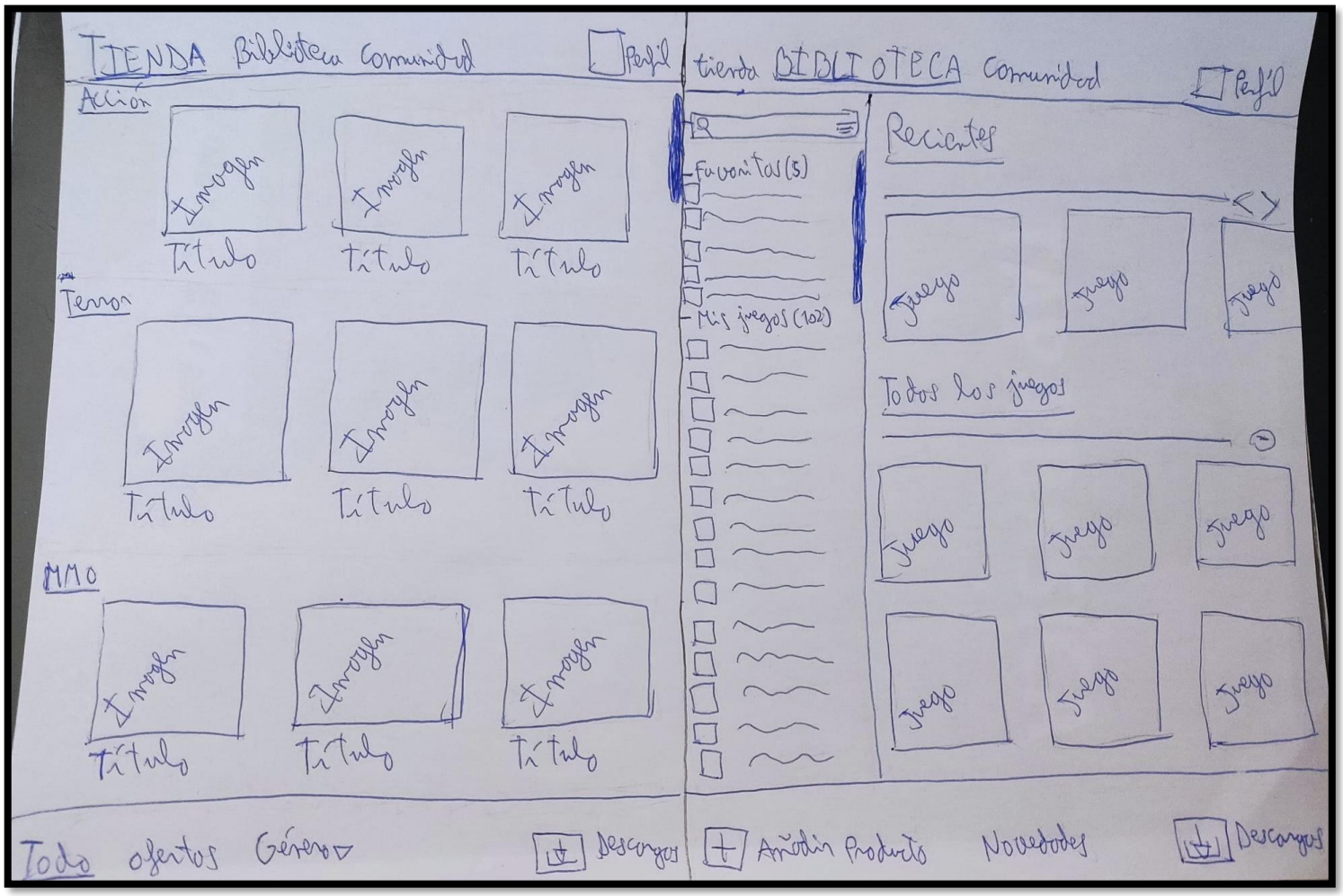
- **Funciones**:
  - Gestionar juegos instalados
  - Visualizar estadísticas personales

6. Panel del Desarrollador

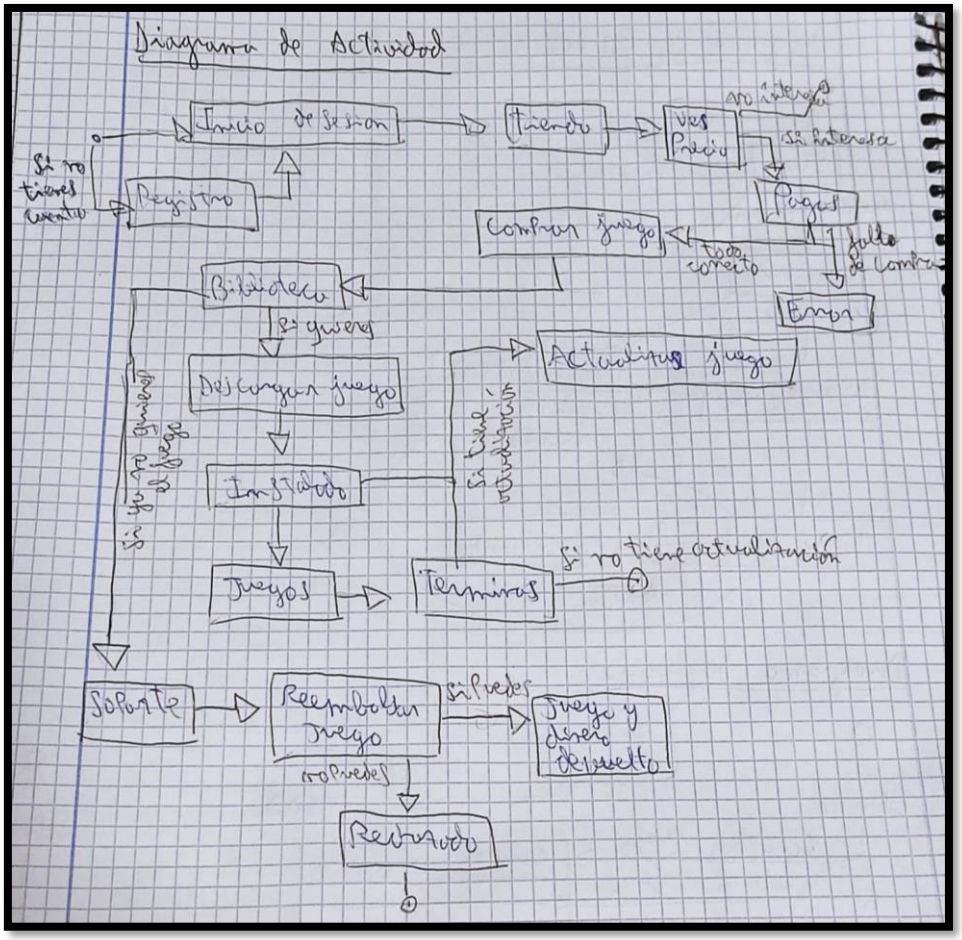
- **Componentes**:
  - Lista de juegos publicados
  - Botones: Publicar nuevo, Editar, Eliminar, Ver estadísticas
- **Funciones**:
  - Subida y gestión de títulos
  - Consulta de ventas y descargas

7. Panel del Administrador

- **Componentes**:
  - Listas de usuarios y juegos
  - Botones: Banear, Revisar, Controlar Soporte
- **Funciones**:
  - Moderación de usuarios
  - Control de contenido
  - Gestión de soporte técnico


TIENDA Biblioteca Comunidad Perfil
Acción
Imagen
Título
Terror
MMO
Todo ofertas Género
Descargas
tienda BIBLIOTECA Comunidad Perfil
Favoritos (5)
Mis juegos (102)
Recientes
Juego
Todos los juegos
Añadir Producto
Novedades
Descargas

Diagramas: Actividad - 0,5 puntos



Diseño de pruebas - 1 punto

Pruebas Unitarias

ID	Nombre	Entrada	Resultado Esperado	Resultado Real
1	Seleccionar Nombre de usuario	Nombre valido	Usuario correcto	
2	Elegir Email	Email existente	Error: email ya registrado	
3	Introducir Contraseña	Contraseña incorrecta	Error: contraseña incorrecta	
4	Introducir Usuario	Usuario inexistente	Error: usuario no encontrado	
5	Compra juego	Pago realizado	Pago confirmado	
6	Juego ya adquirido	Comprobar juego	Mensaje: ya tienes ese juego	
7	Descargar juego comprado	Descargar en la biblioteca	Descarga iniciada y finalizada	
8	Descargar juego no comprado	Descargar juego que no está en la biblioteca	Mensaje: debes comprar este juego antes	
9	Publicar juego	Juego valido	Juego publicado con éxito	
10	Confirmar publicación	Campos vacíos	Error: es necesario rellenar todos los campos	

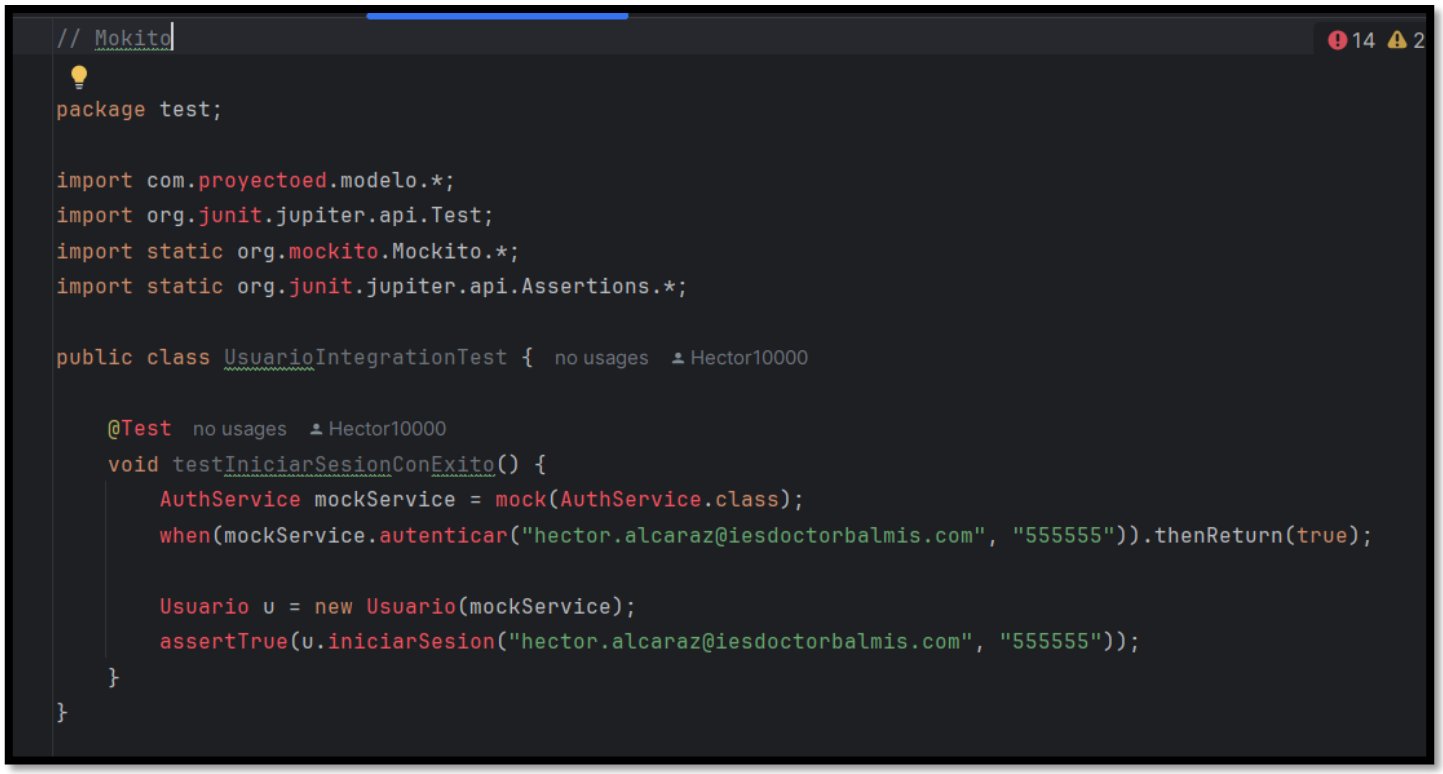
```java
package test;

import com.proyectoed.modelo.Usuario;
import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class UsuarioTest {  no usages  Hector10000

    @Test  no usages  Hector10000
    void testNombre() {
        Usuario u = new Usuario();
        u.setNombre("Héctor");
        assertEquals("Héctor", u.getNombre());
    }
}
```

Pruebas de Integración (Mokito)



```java
// Mokito
package test;

import com.proyectoed.modelo.*;
import org.junit.jupiter.api.Test;
import static org.mockito.Mockito.*;
import static org.junit.jupiter.api.Assertions.*;

public class UsuarioIntegrationTest {  no usages  Hector10000

    @Test  no usages  Hector10000
    void testIniciarSesionConExito() {
        AuthService mockService = mock(AuthService.class);
        when(mockService.autenticar("hector.alcaraz@iesdoctorbalmis.com", "555555")).thenReturn(true);

        Usuario u = new Usuario(mockService);
        assertTrue(u.iniciarSesion("hector.alcaraz@iesdoctorbalmis.com", "555555"));
    }
}
```

Pruebas de Aceptación (Cucumber)

```
// Cucumber

/*
Feature: Inicio de sesión

Scenario: Usuario inicia sesión con éxito
Given el usuario "ana@example.com" con contraseña "abc123"
When intenta iniciar sesión
Then el sistema muestra "Inicio de sesión exitoso"
*/

package test;

import com.proyectoed.modelo.*;
import io.cucumber.java.en.*;
import static org.junit.jupiter.api.Assertions.*;

public class Login {  no usages  Hector10000
    private Usuario usuario;  2 usages
    private boolean resultado;  2 usages

    @Given("el usuario {string} con contraseña {string}")  no usages  Hector10000
    public void crearUsuario(String email, String contraseña) {
        AuthService authService = (e, c) -> e.equals(email) && c.equals(contraseña);
        usuario = new Usuario(authService);
    }

    @When("intenta iniciar sesión")  no usages  Hector10000
    public void iniciarSesion() {
        resultado = usuario.iniciarSesion("hector.alcaraz@iesdoctorbalmis.com", "555555");
    }

    @Then("el sistema muestra {string}")  no usages  Hector10000
    public void verificarMensaje(String mensaje) {
        assertTrue(resultado);
    }
}
```

Pruebas de Seguridad

```
<build>
  <plugins>
    <plugin>
      <groupId>org.owasp</groupId>
      <artifactId>dependency-check-maven</artifactId>
      <version>8.4.0</version>
      <configuration>
        <outputDirectory>${project.basedir}/informes/seguridad</outputDirectory>
      </configuration>
    </plugin>
  </plugins>
</build>
```

Plantilla de código - 1 punto

Documentación de código - 0,5 puntos

Presentación - 1 punto