

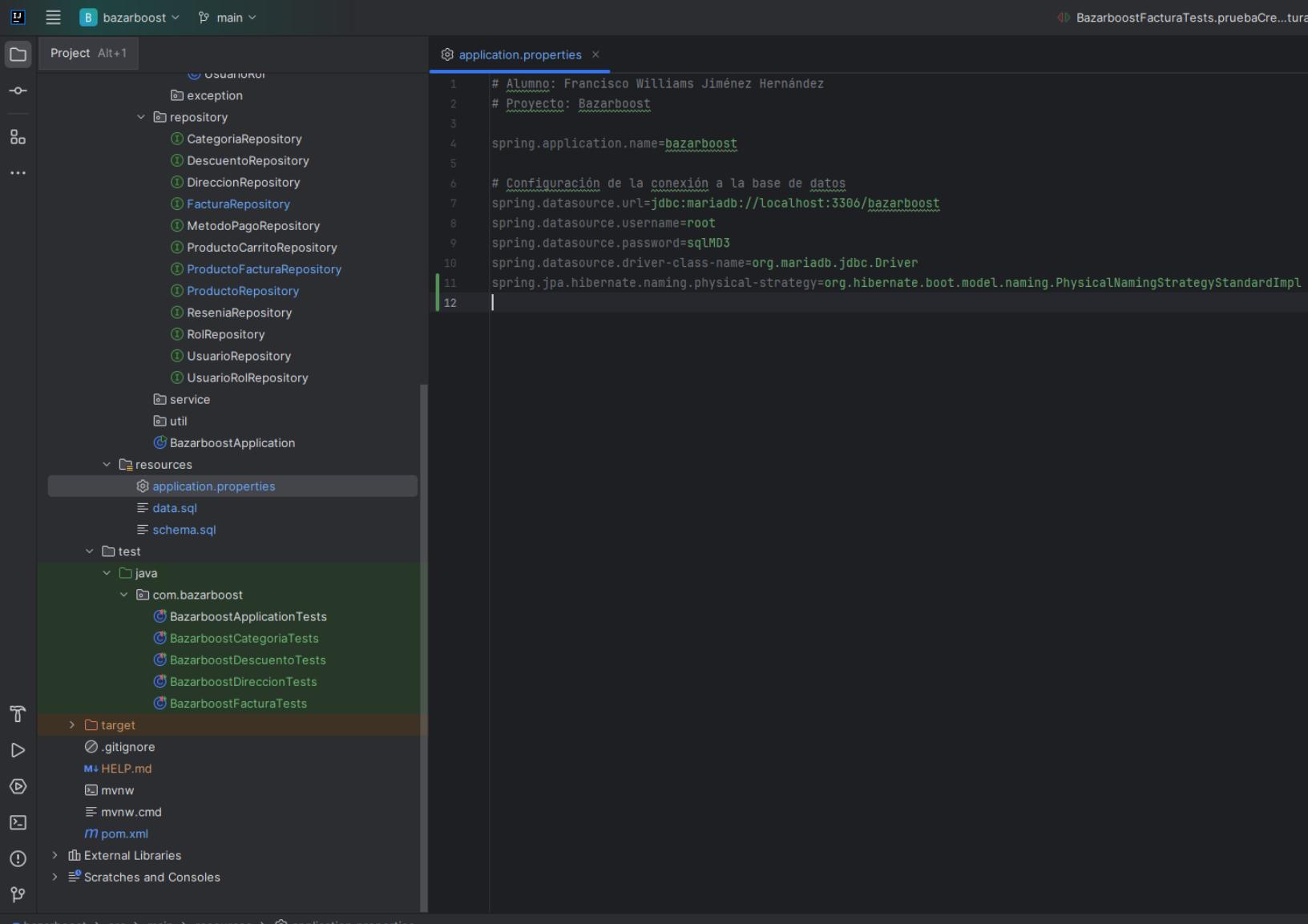
# Francisco Williams Jiménez Hernández

## Ejercicio 2

### 1. Configuración inicial

Para llevar a cabo este ejercicio, inicialmente configuré la conexión a la base de datos a través del archivo **application.properties**. En este archivo, especifiqué los parámetros correspondientes a la base de datos, incluyendo el nombre de la base de datos, el usuario, la contraseña y el controlador (driver) necesario para establecer la conexión de manera adecuada.

#### Archivo: application.properties



The screenshot shows a Java IDE interface with the project structure on the left and the code editor on the right. The project is named 'bazarboost' and contains several packages: 'com.bazarboost' (containing 'BazarboostApplication', 'BazarboostCategoriaTests', 'BazarboostDescuentoTests', 'BazarboostDireccionTests', and 'BazarboostFacturaTests'), 'repository' (containing multiple repository classes), 'service', 'util', and 'resources'. The 'resources' folder contains 'application.properties', 'data.sql', and 'schema.sql'. The 'target' folder is also visible. In the code editor, the 'application.properties' file is open, showing the following configuration:

```
# Alumno: Francisco Williams Jiménez Hernández
# Proyecto: Bazarboost

spring.application.name=bazarboost

# Configuración de la conexión a la base de datos
spring.datasource.url=jdbc:mariadb://localhost:3306/bazarboost
spring.datasource.username=root
spring.datasource.password=sqlMD3
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
```

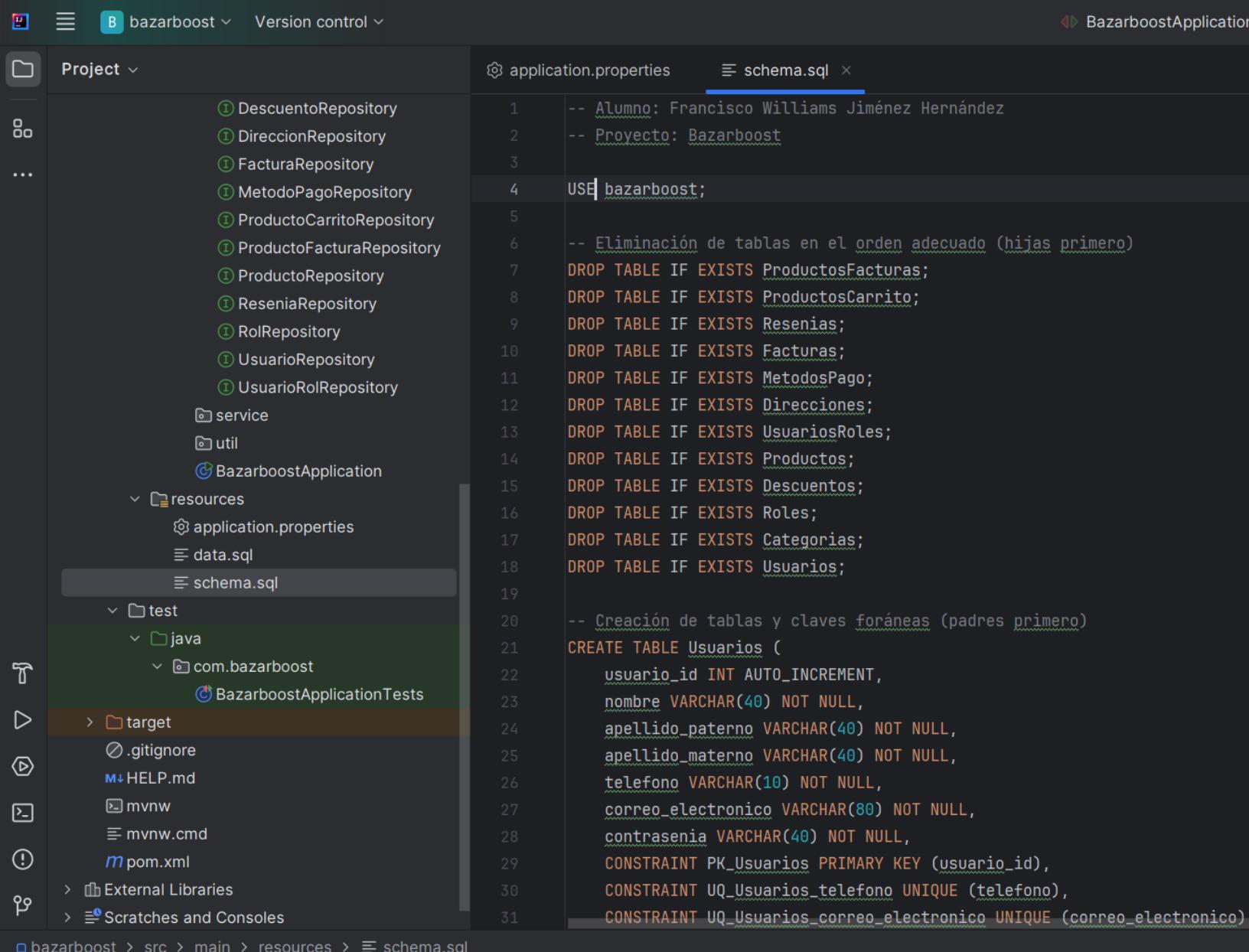
Captura 1. Código del archivo application.properties del proyecto para la configuración de la conexión a la base de datos.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Posteriormente, elaboré los scripts **schema.sql** y **data.sql** para la creación del esquema de la base de datos y la inserción de los datos de prueba. Estos archivos fueron ubicados en la carpeta **resources** del proyecto. A continuación, se presenta el contenido de ambos archivos.

### Archivo: schema.sql



The screenshot shows a Java IDE interface with the following details:

- Project View:** Shows the project structure under "Project". It includes repositories for Descuento, Direccion, Factura, MetodoPago, ProductoCarrito, ProductoFactura, Producto, Resenia, Rol, Usuario, and UsuarioRol. It also contains "service", "util", and the main application class "BazarboostApplication".
- Resources View:** Shows files under "resources": "application.properties", "data.sql", and "schema.sql".
- Editor View:** Shows the content of "schema.sql".
- Bottom Status Bar:** Shows the path: "bazarboost > src > main > resources > schema.sql".

The content of the "schema.sql" file is as follows:

```
-- Alumno: Francisco Williams Jiménez Hernández
-- Proyecto: Bazarboost
USE bazarboost;

-- Eliminación de tablas en el orden adecuado (hijas primero)
DROP TABLE IF EXISTS ProductosFacturas;
DROP TABLE IF EXISTS ProductosCarrito;
DROP TABLE IF EXISTS Resenias;
DROP TABLE IF EXISTS Facturas;
DROP TABLE IF EXISTS MetodosPago;
DROP TABLE IF EXISTS Direcciones;
DROP TABLE IF EXISTS UsuariosRoles;
DROP TABLE IF EXISTS Productos;
DROP TABLE IF EXISTS Descuentos;
DROP TABLE IF EXISTS Roles;
DROP TABLE IF EXISTS Categorias;
DROP TABLE IF EXISTS Usuarios;

-- Creación de tablas y claves foráneas (padres primero)
CREATE TABLE Usuarios (
    usuario_id INT AUTO_INCREMENT,
    nombre VARCHAR(40) NOT NULL,
    apellido_paterno VARCHAR(40) NOT NULL,
    apellido_materno VARCHAR(40) NOT NULL,
    telefono VARCHAR(10) NOT NULL,
    correo_electronico VARCHAR(80) NOT NULL,
    contrasenia VARCHAR(40) NOT NULL,
    CONSTRAINT PK_Usuarios PRIMARY KEY (usuario_id),
    CONSTRAINT UQ_Usuarios_telefono UNIQUE (telefono),
    CONSTRAINT UQ_Usuarios_correo_electronico UNIQUE (correo_electronico)
```

Captura 2. Código del archivo schema.sql para la creación del esquema.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: data.sql

```
-- Alumno: Francisco Williams Jiménez Hernández
-- Proyecto: Bazarboost

-- Inserts de datos de prueba

-- Inserciones en la tabla Usuarios
INSERT INTO Usuarios (nombre, apellido_paterno, apellido_materno, correo_electronico, contrasenia, telefono) VALUES
('Juan', 'Pérez', 'López', 'juan.perez@example.com', 'password123', '55512345671'),
('María', 'Gómez', 'Martínez', 'maria.gomez@example.com', 'password123', '5557654321'),
('Pedro', 'Sánchez', 'Hernández', 'pedro.sanchez@example.com', 'password123', '5556781234'),
('Ana', 'Ramírez', 'García', 'ana.ramirez@example.com', 'password123', '5559876543'),
('Carlos', 'Fernández', 'Pérez', 'carlos.fernandez@example.com', 'password123', '5554567890');

-- Inserciones en la tabla Categorías
INSERT INTO Categorias (nombre) VALUES
('Electrónica'),
('Ropa'),
('Hogar'),
('Juguetes'),
('Libros');

-- Inserciones en la tabla Roles
INSERT INTO Roles (nombre, descripcion) VALUES
('Administrador', 'Acceso total al sistema'),
('Cliente', 'Puede realizar compras y dejar reseñas'),
('Vendedor', 'Puede gestionar productos y ventas');

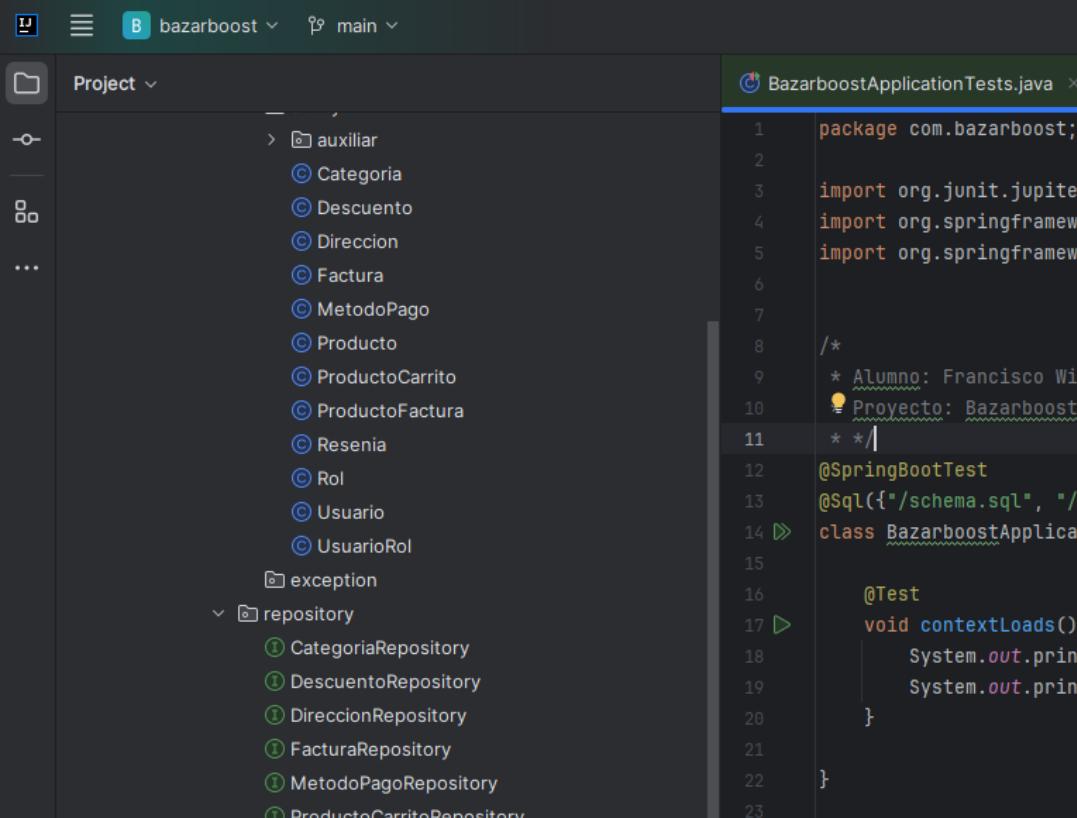
-- Inserciones en la tabla Descuentos
INSERT INTO Descuentos (porcentaje, nombre, usuario_id) VALUES
(10, 'Descuento de Bienvenida', 1),
(15, 'Descuento de Verano', 2),
```

Captura 3. Código del archivo data.sql para la inserción de datos de prueba.

## Francisco Williams Jiménez Hernández

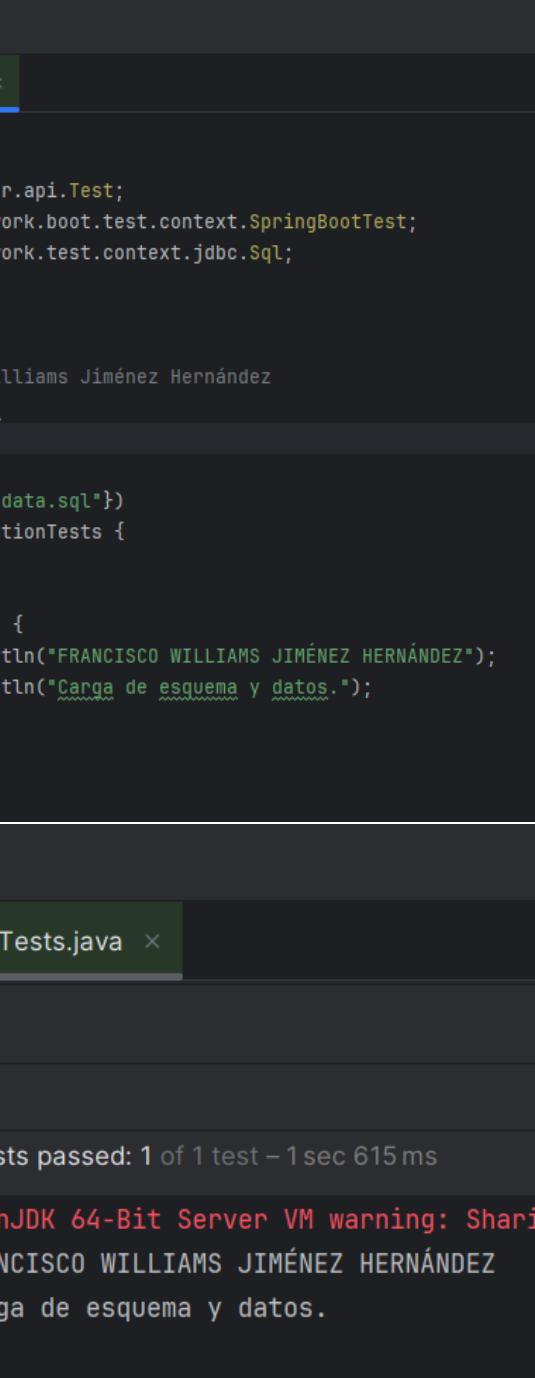
### Ejercicio 2

Finalmente, utilicé la anotación `@Sql` en el archivo de pruebas `BazarboostApplicationTests.java` para indicar la ejecución de los scripts `schema.sql` y `data.sql` cada vez que se ejecute dicha clase de pruebas. Esto permite inicializar y/o restablecer el estado de la base de datos según sea necesario para garantizar un entorno consistente en cada ejecución.



The screenshot shows the IntelliJ IDEA interface with the project 'bazarboost' open. The left sidebar displays the project structure, including packages like auxiliar, repository, and entities such as Categoria, Descuento, Direccion, Factura, MetodoPago, Producto, ProductoCarrito, ProductoFactura, Resenia, Rol, Usuario, and UsuarioRol. Below these are exception and repository interfaces. The main editor window shows the code for `BazarboostApplicationTests.java`:

```
1 package com.bazarboost;
2
3 import org.junit.jupiter.api.Test;
4 import org.springframework.boot.test.context.SpringBootTest;
5 import org.springframework.test.context.jdbc.Sql;
6
7 /*
8 * Alumno: Francisco Williams Jiménez Hernández
9 * Proyecto: Bazarboost
10 */
11
12 @SpringBootTest
13 @Sql({"schema.sql", "data.sql"})
14 class BazarboostApplicationTests {
15
16     @Test
17     void contextLoads() {
18         System.out.println("FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ");
19         System.out.println("Carga de esquema y datos.");
20     }
21 }
22
23 }
```



The Run tool window shows the execution of the `BazarboostApplicationTests` class. The `contextLoads()` test passed in 1 second, 615 milliseconds. The output pane shows the printed messages: "FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ" and "Carga de esquema y datos.". A warning message from the OpenJDK 64-Bit Server VM is also visible.

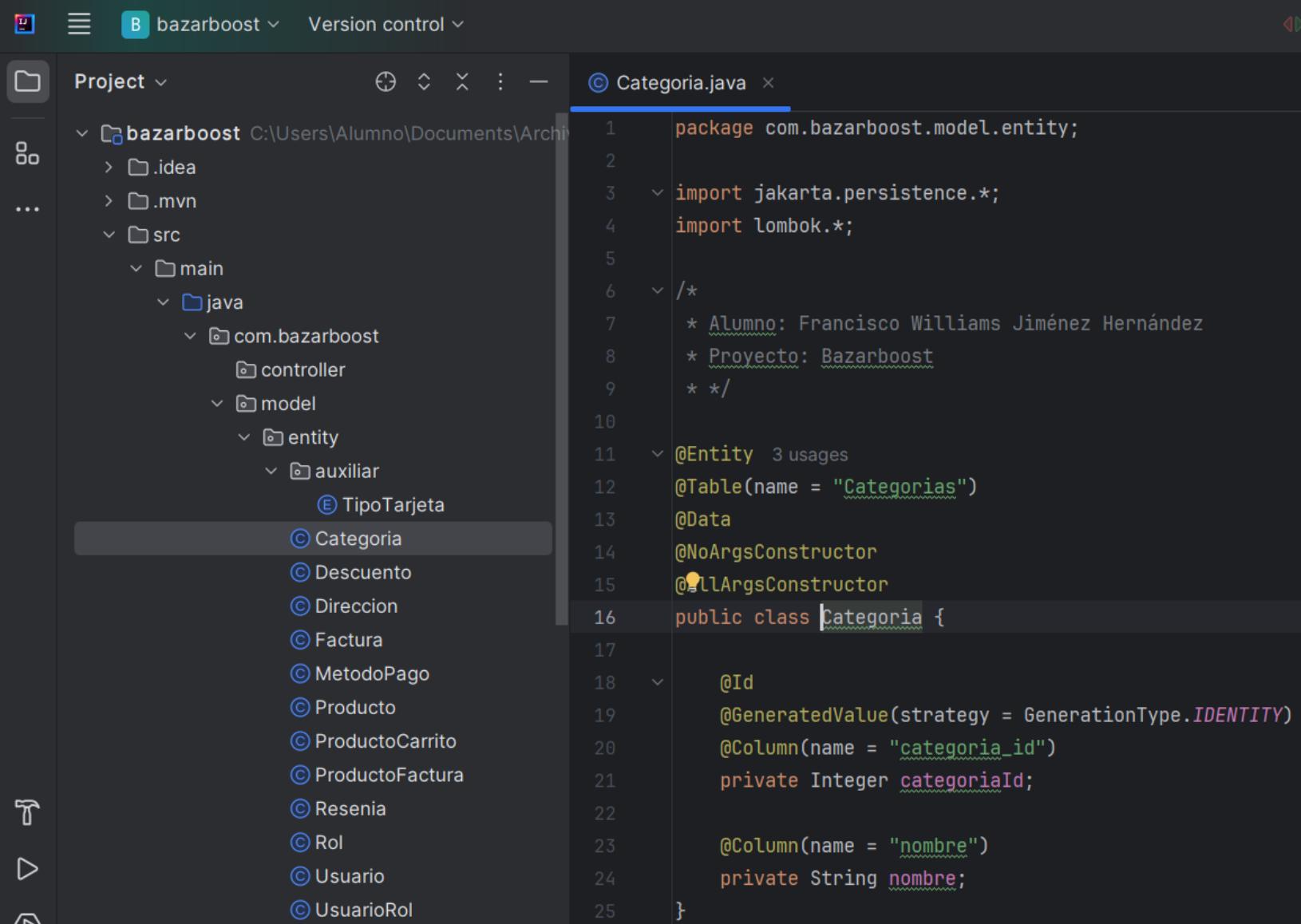
Captura 4. Código del archivo `BazarboostApplicationTests.java` y su salida de ejecución para la creación del esquema y la carga de datos de prueba.

## 2. Creación de entidades

Una vez completadas las configuraciones iniciales para la conexión a la base de datos, así como la carga del esquema y los datos de prueba, el siguiente paso fue la creación de las clases correspondientes a las entidades. Para ello, se emplearon las anotaciones de **Spring Data JPA**, las cuales permitieron el mapeo de las clases a las tablas relacionales de la base de datos. Además, se utilizaron las anotaciones de **Lombok** con el fin de reducir el código repetitivo, como los métodos **constructores, getters, setters**, entre otros.

**Nota:** Se muestran 5 entidades de las 12 totales de mi proyecto con el fin de no extender demasiado el documento a la vez que se cumple con los requerimientos del PDF.

### Archivo: Categoria.java



```
package com.bazarboost.model.entity;

import jakarta.persistence.*;
import lombok.*;

/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */

@Entity 3 usages
@Table(name = "Categorias")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Categoria {
```

The screenshot shows the IntelliJ IDEA interface with the project structure on the left and the code editor on the right. The code editor displays the `Categoria.java` file. The file is part of the `bazarboost` project, located in the `com.bazarboost.model.entity` package. The code defines a `Categoria` entity with an `@Id` field named `categoriaId` and a `@Column` named `nombre`. The code editor highlights several annotations with underlines, indicating they are Lombok annotations.

Captura 5. Clase de entidad para *Categoría*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: Descuento.java

The screenshot shows a Java IDE interface with the following details:

- Project Bar:** Shows the project name "bazarboost" and the current file "Descuento.java".
- Left Sidebar:** Displays the project structure under "bazarboost". The "src" folder contains "main" and "java". The "java" folder contains "com.bazarboost" which has "controller", "model", and "entity". The "entity" folder contains "auxiliar", "TipoTarjeta", "Categoria", "Descuento", "Direccion", "Factura", "MetodoPago", "Producto", "ProductoCarrito", "ProductoFactura", "Resenia", "Rol", "Usuario", "UsuarioRol", "exception", and "repository". The "repository" folder contains "CategoriaRepository", "DescuentoRepository", "DireccionRepository", and "FacturaRepository".
- Current File:** The file "Descuento.java" is open. The code defines a class "Descuento" with annotations for Entity, Table, Data, NoArgsConstructor, and AllArgsConstructor. It includes fields for id (@Id, @GeneratedValue(strategy = GenerationType.IDENTITY), @Column(name = "descuento\_id")), porcentaje (private Integer porcentaje; @Column(name = "porcentaje")), nombre (private String nombre; @Column(name = "nombre")), and usuario (private Usuario usuario; @ManyToOne, @JoinColumn(name = "usuario\_id")).
- Bottom Status Bar:** Shows the full path: bazarboost > src > main > java > com > bazarboost > model > entity > Descuento.

Captura 6. Clase de entidad para *Descuento*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: Direccion.java

The screenshot shows a Java project structure on the left and the code editor on the right. The project structure includes .idea, .mvn, and src folders. The src folder contains main, java, com.bazarboost, controller, model, entity, auxiliar, exception, repository, and various repository classes. The code editor is displaying Direccion.java, which defines a class Direccion with annotations for Entity, Table, and Data. The class has fields for direccion\_id (@Id, @GeneratedValue(strategy = GenerationType.IDENTITY), @Column(name = "direccion\_id")), estado (String, @Column(name = "estado")), ciudad (String, @Column(name = "ciudad")), colonia (String, @Column(name = "colonia")), calle (String, @Column(name = "calle")), and numero\_domicilio (Integer, @Column(name = "numero\_domicilio")). The code editor also shows imports for java.util.List and javax.persistence.Entity.

```
/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */

@Entity 3 usages
@Table(name = "Direcciones")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Direccion {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "direccion_id")
    private Integer direccionId;

    @Column(name = "estado")
    private String estado;

    @Column(name = "ciudad")
    private String ciudad;

    @Column(name = "colonia")
    private String colonia;

    @Column(name = "calle")
    private String calle;

    @Column(name = "numero_domicilio")
    private Integer numeroDomicilio;
}
```

bazarboost > src > main > java > com > bazarboost > model > entity > Direccion.java

Captura 7. Clase de entidad para Direccion.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: Factura.java

```
Project: bazarboost C:\Users\Alumno\Documents\Archivos\bazarboost
File: Factura.java

/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
@Entity 3 usages
@Table(name = "Facturas")
@Data
@NoArgsConstructor
@AllArgsConstructor
public class Factura {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "factura_id")
    private Integer facturaId;

    @Column(name = "fecha")
    private LocalDateTime fecha;

    @Column(name = "subtotal")
    private Double subtotal;

    @Column(name = "total")
    private Double total;

    @Column(name = "porcentaje_impuestos")
    private Integer porcentajeImpuestos;

    @ManyToOne
    @JoinColumn(name = "usuario_id")
    private Usuario usuario;
}

bazarboost > src > main > java > com > bazarboost > model > entity > Factura.java
```

Captura 8. Clase de entidad para *Factura*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: MetodoPago.java

```
Project: bazarboost C:\Users\Alumno\Documents\Archivos\Java\Bazarboost\src\main\java\com\bazarboost\model\MetodoPago.java
10  /*
11   * Alumno: Francisco Williams Jiménez Hernández
12   * Proyecto: Bazarboost
13   */
14
15  @Entity 5 usages  ↳ Williams
16  @Table(name = "MetodosPago")
17  @Data
18  @NoArgsConstructor
19  @AllArgsConstructor
20  public class MetodoPago {
21
22      @Id
23      @GeneratedValue(strategy = GenerationType.IDENTITY)
24      @Column(name = "metodo_pago_id")
25      private Integer metodoPagoId;
26
27      @Column(name = "nombre_titular")
28      private String nombreTitular;
29
30      @Column(name = "numero_tarjeta")
31      private String numeroTarjeta;
32
33      @Column(name = "fecha_expiracion")
34      private LocalDate fechaExpiracion; // Cambio a LocalDate
35
36      @Column(name = "tipo_tarjeta")
37      @Enumerated(EnumType.STRING)
38      private TipoTarjeta tipoTarjeta;
39
40      @Column(name = "monto")
```

bazarboost > src > main > java > com > bazarboost > model

Captura 9. Clase de entidad para *MetodoPago*.

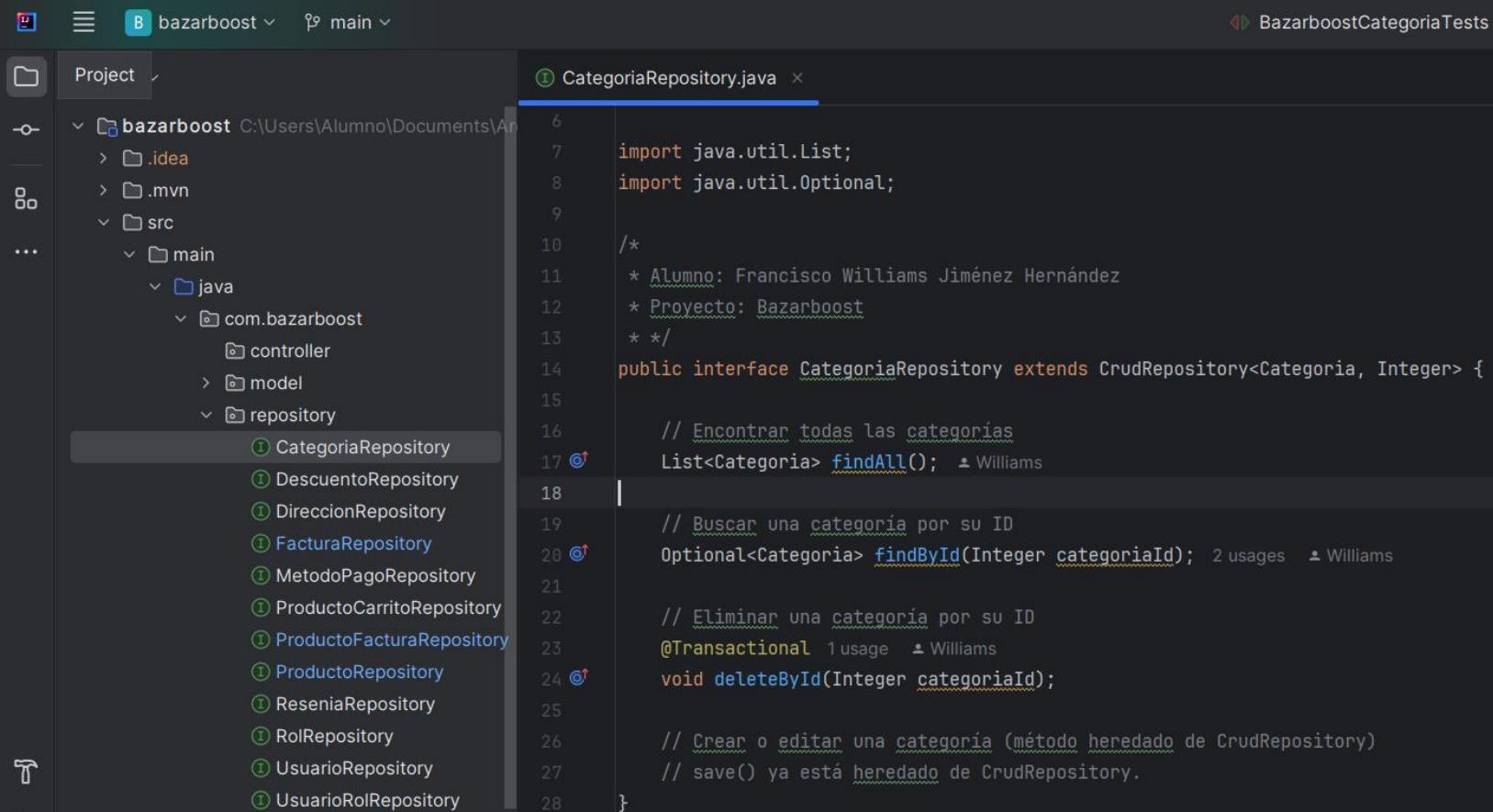
### 3. Creación de repositorios

Una vez creadas las entidades, elaboré los repositorios correspondientes, cada repositorio contiene los métodos para consultas requeridas de acuerdo a las reglas de negocio de mi proyecto final; dichas queries incluyen consultas derivadas, nombradas, de CrudRepository y de PagingAndSorting.

El total de consultas de cada tipo entre las cinco entidades es el siguiente:

- Derivadas: 7
- Nombradas: 7
- De relación: 14

#### Archivo: CategoriaRepository.java



The screenshot shows the IntelliJ IDEA interface. On the left, the project structure is visible under the 'Project' tab, showing a package named 'bazarboost' containing '.idea', '.mvn', and 'src' directories. The 'src/main/java/com/bazarboost/repository' directory contains several repository interface files, with 'CategoriaRepository.java' currently selected and highlighted in blue. The right panel displays the code for 'CategoriaRepository.java'. The code is an interface that extends 'CrudRepository<Categoria, Integer>'. It includes methods for finding all categories ('findAll()'), finding a category by its ID ('findById(Integer categoriaId)'), deleting a category by its ID ('deleteById(Integer categoriaId)'), and creating or editing a category ('save()'). The code is annotated with Javadoc comments indicating the author is 'Francisco Williams Jiménez Hernández' and the project is 'Bazarboost'. The code editor shows syntax highlighting for Java and annotations like '@Transactional'.

```
import java.util.List;
import java.util.Optional;

/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
public interface CategoriaRepository extends CrudRepository<Categoria, Integer> {

    // Encontrar todas las categorías
    List<Categoria> findAll();  ▲ Williams

    // Buscar una categoría por su ID
    Optional<Categoria> findById(Integer categoriaId);  2 usages ▲ Williams

    // Eliminar una categoría por su ID
    @Transactional 1 usage ▲ Williams
    void deleteById(Integer categoriaId);

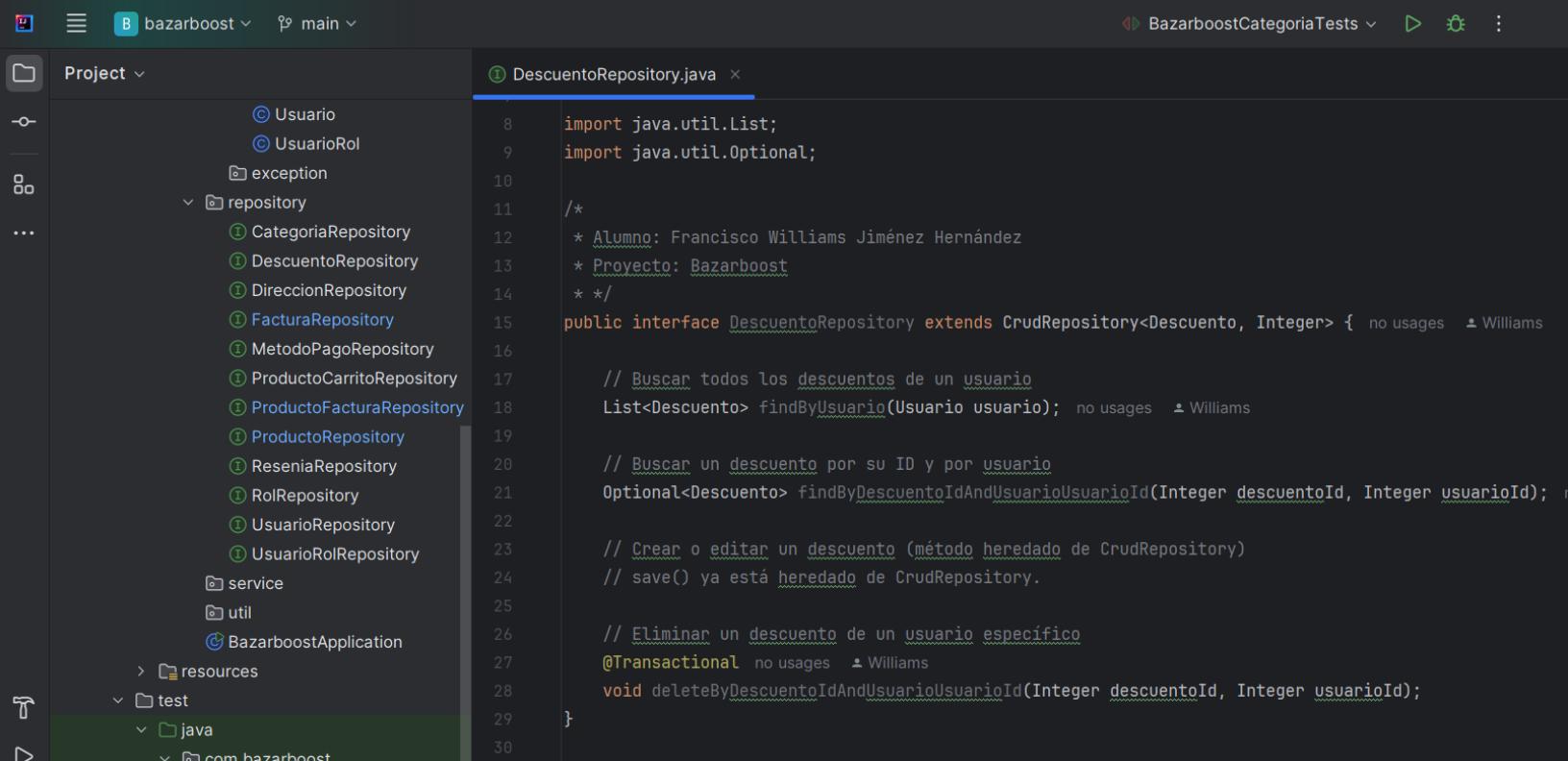
    // Crear o editar una categoría (método heredado de CrudRepository)
    // save() ya está heredado de CrudRepository.
}
```

Captura 10. Clase de interfaz de repositorio para *Categoría*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: DescuentoRepository.java



The screenshot shows a Java IDE interface with the following details:

- Project:** bazarboost
- File:** DescuentoRepository.java
- Content:** The code defines an interface for a repository named DescuentoRepository. It extends the CrudRepository interface for the Descuento entity. The interface includes methods for finding all discounts for a user, finding a discount by its ID and the user's ID, and deleting a specific discount for a user. It also includes a save() method inherited from CrudRepository.
- Annotations:** The code uses annotations such as `@Transactional` and `CrudRepository<Descuento, Integer>`.
- Comments:** The code includes comments indicating the student's name (`* Alumno: Francisco Williams Jiménez Hernández`) and the project name (`* Proyecto: Bazarboost`).

Captura 11. Clase de interfaz de repositorio para *Descuento*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: DireccionRepository.java

The screenshot shows a Java IDE interface with the following details:

- Project:** bazarboost
- File:** DireccionRepository.java
- Content:** The code defines an interface for a repository named DireccionRepository that extends CrudRepository<Direccion, Integer>. The interface includes methods for finding all addresses of a user by ID, finding an address by ID and user ID, deleting an address by ID and user ID, and saving a new or edited address.
- Annotations:** The code uses annotations such as `import java.util.List;`, `import java.util.Optional;`, `@Transactional`, and `@Override`.
- Comments:** A multi-line comment at the top of the class specifies the student's name as "Alumno: Francisco Williams Jiménez Hernández" and the project name as "Proyecto: Bazarboost".
- IDE UI:** The left sidebar shows the project structure with entities like Categoria, Descuento, Direccion, Factura, MetodoPago, Producto, ProductoCarrito, ProductoFactura, Resenia, Rol, Usuario, and UsuarioRol, along with repository interfaces for each entity.

Captura 12. Clase de interfaz de repositorio para *Dirección*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

### Archivo: FacturaRepository.java

The screenshot shows a Java IDE interface with the following details:

- Project:** bazarboost
- File:** FacturaRepository.java
- Content:** The code defines an interface for a repository named FacturaRepository. It extends PagingAndSortingRepository and CrudRepository. The code includes methods for creating a new factura (save), getting all facturas associated with a user with pagination, and ordering facturas by total amount (desc and asc). It also includes annotations like @Query and @Param.
- Annotations:** The code contains several Javadoc-style annotations:
  - Line 12: `/* * Alumno: Francisco Williams Jiménez Hernández`
  - Line 13: `* Proyecto: Bazarboost`
  - Line 14: `*/`
  - Line 22: `// Ordenar facturas del usuario por monto total (de mayor a menor) con paginación`
  - Line 23: `@Query("SELECT f FROM Factura f WHERE f.usuario.usuarioid = :usuarioid ORDER BY f.total DESC")`
  - Line 24: `Page<Factura> findFacturasByUsuarioIdOrderByTotalDesc(@Param("usuarioid") Integer usuarioid, Pageable pageable);`
  - Line 26: `// Ordenar facturas del usuario por monto total (de menor a mayor) con paginación`
  - Line 27: `@Query("SELECT f FROM Factura f WHERE f.usuario.usuarioid = :usuarioid ORDER BY f.total ASC")`
  - Line 28: `Page<Factura> findFacturasByUsuarioIdOrderByTotalAsc(@Param("usuarioid") Integer usuarioid, Pageable pageable);`
- IDE UI:** The IDE has a dark theme with various toolbars and status bars at the top and bottom.

Captura 13. Clase de interfaz de repositorio para *Factura*.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

### Archivo: MetodoPagoRepository.java

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "bazarboost". It contains several packages:
  - repository:** Contains repository classes for CategoriaRepository, DescuentoRepository, DireccionRepository, FacturaRepository, MetodoPagoRepository, ProductoCarritoRepository, ProductoFacturaRepository, ProductoRepository, ReseniaRepository, RolRepository, UsuarioRepository, and UsuarioRolRepository.
  - service**
  - util**
  - BazarboostApplication**
  - resources:** Contains application.properties, data.sql, and schema.sql.
  - test:** Contains java subfolders with test classes for BazarboostApplicationTests, BazarboostCategoriaTests, BazarboostDescuentoTests, BazarboostDireccionTests, BazarboostFacturaTests, and BazarboostMetodoPagoTests.
  - target**
  - .gitignore**
  - HELP.md**
  - mvnw**
  - mvnw.cmd**
- Code Editor:** The file "MetodoPagoRepository.java" is open. The code defines an interface for managing payment methods. It includes annotations for JPA queries and methods for finding payments by user ID, deleting specific payments, and verifying funds and card expiration.

```
/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
public interface MetodoPagoRepository extends CrudRepository<MetodoPago, Integer> { 2 usages □ Williams *

    // Encontrar todos los métodos de pago de un usuario
    @Query("SELECT mp FROM MetodoPago mp WHERE mp.usuario.usuarioid = :usuarioid") 4 usages new *
    List<MetodoPago> findByUsuarioUsuarioid(@Param("usuarioid") Integer usuarioid);

    // Buscar un método de pago por su ID y validar que pertenece al usuario
    @Query("SELECT mp FROM MetodoPago mp WHERE mp.metodopagoId = :metodopagoId AND mp.usuario.usuarioid = :usuarioid")
    Optional<MetodoPago> findByMetodoPagoIdAndUsuarioUsuarioid(@Param("metodopagoId") Integer metodopagoId,
                                                               @Param("usuarioid") Integer usuarioid);

    // Eliminar un método de pago de un usuario específico
    @Transactional 1 usage new *
    @Modifying
    @Query("DELETE FROM MetodoPago mp WHERE mp.metodopagoId = :metodopagoId AND mp.usuario.usuarioid = :usuarioid")
    void deleteByMetodoPagoIdAndUsuarioUsuarioid(@Param("metodopagoId") Integer metodopagoId,
                                                   @Param("usuarioid") Integer usuarioid);

    // Verificar fondos suficientes en el método de pago
    @Query("SELECT CASE WHEN mp.monto >= :total THEN true ELSE false END " + 2 usages □ Williams *
           "FROM MetodoPago mp WHERE mp.metodopagoId = :metodopagoId AND mp.usuario.usuarioid = :usuarioid")
    boolean verifySufficientFunds(@Param("metodopagoId") Integer metodopagoId,
                                  @Param("usuarioid") Integer usuarioid,
                                  @Param("total") Double total);

    // Verificar si la tarjeta de crédito ha expirado
    @Query("SELECT CASE WHEN mp.fechaExpiracion > CURRENT_DATE THEN true ELSE false END " + 2 usages □ Williams *
           "FROM MetodoPago mp WHERE mp.metodopagoId = :metodopagoId AND mp.usuario.usuarioid = :usuarioid")
    boolean verifyExpirationCard(@Param("metodopagoId") Integer metodopagoId,
                                @Param("usuarioid") Integer usuarioid);
}
```

Captura 14. Clase de interfaz de repositorio para *MetodoPago*.

**Francisco Williams Jiménez Hernández**

**Ejercicio 2**

**4. Clases de pruebas y salidas de ejecución**

Después de crear los repositorios, desarrollé las clases de prueba correspondientes, una para cada entidad, e implementé los métodos necesarios para verificar el correcto funcionamiento de los métodos definidos en cada repositorio.

**Nota:** Utilicé la anotación @Sql para reiniciar el estado de la base de datos antes de la ejecución de cada método de pruebas, evitando problemas de inconsistencia causados por modificaciones en pruebas anteriores. Además, apliqué @TestMethodOrder junto con @Order para asegurar que las pruebas se ejecuten en el orden correcto, manteniendo la consistencia de los datos. De este modo, se pueden ejecutar las pruebas repetidamente sin generar inconsistencias.

*Las capturas comienzan en la siguiente página.*

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: BazarboostCategoriaTests.java y su salida de ejecución

The screenshot shows the IntelliJ IDEA interface. On the left, the project structure is visible, including packages like MetodoPago, Producto, ProductoCarrito, ProductoFactura, Resenia, Rol, Usuario, and UsuarioRol, along with exception, repository, service, util, and BazarboostApplication. Below these are resources (application.properties, data.sql, schema.sql) and a test directory containing Java tests for various entities. The file BazarboostCategoriaTests.java is open in the main editor window. The code is a JUnit test for the CategoriaRepository. It includes annotations @Test and @Order(1), and uses System.out.println for logging. The test creates a new Categoria object, saves it, and then prints the updated list of categories. The output window at the bottom shows the execution results and the printed logs.

```
/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
// Define el orden de ejecución de los tests
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
@SpringBootTest
@Sql({"/schema.sql", "/data.sql"}) // Por defecto, se ejecuta antes de cada método de prueba
public class BazarboostCategoriaTests {

    private final String ALUMNO = "FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ";

    @Autowired
    private CategoriaRepository categoriaRepository;

    @Test
    @Order(1) // Se ejecuta primero
    void crearEditarCategoriaTest() {
        System.out.println("== INICIO PRUEBA: Crear o editar una categoría ==");
        System.out.println("Alumno: " + ALUMNO);
        System.out.println("Prueba: crear o editar una categoría");
        System.out.println("-----");

        // Crear una nueva categoría
        Categoria nuevaCategoria = new Categoria();
        nuevaCategoria.setNombre("Nueva Categoría");

        // Imprimir la categoría a crear
        System.out.println("Datos de la categoría a insertar:");
        System.out.println("- Nombre: " + nuevaCategoria.getNombre());

        // Guardar la nueva categoría
        categoriaRepository.save(nuevaCategoria);
        System.out.println("Categoría guardada con éxito.");

        // Lista actualizada de categorías
        List<Categoria> categorias = categoriaRepository.findAll();
        System.out.println("Lista actualizada de categorías (" + categorias.size() + ")");
        categorias.forEach(categoría ->
            System.out.println("- Categoría: " + categoría.getNombre() + " (ID: " + categoría.getId() + ")");
        );
        System.out.println("== FIN PRUEBA: Crear o editar una categoría ==");
    }
}
```

This screenshot shows the run results for the BazarboostCategoriaTests.java test. The left panel displays the test classes and methods, all of which have passed. The right panel shows the detailed output of the 'crearEditarCategoriaTest()' method. The output includes the initial log message, the student's name, the category creation details, the successful save message, the updated list of categories, and the final log message. The output window also shows a warning about sharing support and the final category list.

Method	Time	Status
BazarboostCategoriaTests (com.bazarboost)	905 ms	✓ Tests passed: 4 of 4 tests – 905 ms
crearEditarCategoriaTest()	838 ms	
encontrarTodasLasCategoriasTest()	7 ms	
buscarCategoriaPorIdTest()	21 ms	
eliminarCategoriaPorIdTest()	39 ms	

OpenJDK 64-Bit Server VM warning: Sharing is only supported between processes by multiple threads, some features not supported  
== INICIO PRUEBA: Crear o editar una categoría ==  
Alumno: FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ  
Prueba: crear o editar una categoría  
-----  
Datos de la categoría a insertar:  
- Nombre: Nueva Categoría  
Categoría guardada con éxito.  
Lista actualizada de categorías (6):  
- Categoría: Electrónica (ID: 1)  
- Categoría: Hogar (ID: 3)  
- Categoría: Juguetes (ID: 4)  
- Categoría: Libros (ID: 5)  
- Categoría: Nueva Categoría (ID: 6)  
- Categoría: Ropa (ID: 2)  
== FIN PRUEBA: Crear o editar una categoría ==

Captura 15. Clase de pruebas para *Categoría* y su salida de ejecución.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: BazarboostDescuentoTests.java y su salida de ejecución

The screenshot shows an IDE interface with the following details:

- Project Explorer:** Shows the project structure under the "Project" tab. It includes packages like repository, service, util, and BazarboostApplication, along with resources like application.properties, data.sql, and schema.sql.
- Code Editor:** The active editor tab is "BazarboostDescuentoTests.java". The code is a Java test class for the Descuento repository. It contains two test methods: `pruebaBuscarTodosLosDescuentosDeUnUsuarioTest()` and `pruebaBuscarDescuentoPorIdYUsuarioTest()`. Both tests output messages to `System.out` indicating the start and end of the test, the student's name, and the query being executed.
- Bottom Status Bar:** Shows the current path as "bazarboost > src > test > java > com > bazarboost > BazarboostDescuentoTests >" and the current file as "BazarboostDescuentoTests.java".

The screenshot shows the execution results of the BazarboostDescuentoTests.java file in the IDE. The results are displayed in the "Run" tab:

- Run Configuration:** Set to "BazarboostDescuentoTests".
- Test Results:** The test "BazarboostDescuentoTests (com.bazarboost)" has passed, with a total time of 923 ms. It lists four test methods:
  - pruebaBuscarTodosLosDescuentosDeUnUsuarioTest() (792 ms)
  - pruebaBuscarDescuentoPorIdYUsuarioTest() (15 ms)
  - pruebaCrearOEditarDescuentoTest() (83 ms)
  - pruebaEliminarDescuentoPorIdYUsuarioTest() (33 ms)
- Output Log:** The log shows the test output:

```
Tests passed: 4 of 4 tests - 923 ms
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has
== INICIO PRUEBA: Buscar todos los descuentos de un usuario ==
Alumno: FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ
Prueba: buscar todos los descuentos de un usuario
-----
Descuentos del usuario con ID 1:
Descuento(descuentoId=1, porcentaje=10, nombre=Descuento de Bienvenida, usuario=Usuario(usuarioId=1, nombre=Juan, ap
== FIN PRUEBA: Buscar todos los descuentos de un usuario ==
```

Captura 16. Clase de pruebas para *Descuento* y su salida de ejecución.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: BazarboostDireccionTests.java y su salida de ejecución

The screenshot shows an IDE interface with the following details:

- Project Structure:** The project is named "bazarboost". It contains several packages: exception, repository, service, util, BazarboostApplication, resources, and test. The test package contains sub-folders java and resources, and files like BazarboostApplicationTests, BazarboostCategoriaTests, BazarboostDescuentoTests, BazarboostDireccionTests (which is selected), BazarboostFacturaTests, and BazarboostMetodoPagoTests.
- BazarboostDireccionTests.java Content:**

```
/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
@SpringBootTest
@MethodOrder(MethodOrderer.OrderAnnotation.class)
@Sql({"/schema.sql", "/data.sql"}) // Por defecto, se ejecuta antes de cada método de prueba
public class BazarboostDireccionTests {

    private static final String ALUMNO = "FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ"; 4 usages

    @Autowired 7 usages
    private DireccionRepository direccionRepository;

    @Test
    @Order(1)
    void pruebaEncontrarTodasLasDireccionesDeUsuarioTest() {
        System.out.println("== INICIO PRUEBA: Encontrar todas las direcciones de un usuario ==");
        System.out.println("Alumno: " + ALUMNO);
        System.out.println("Prueba: encontrar todas las direcciones de un usuario");
        System.out.println("-----");

        // Usuario de ejemplo
        Integer usuarioId = 1;

        // Buscar todas las direcciones de un usuario
        List<Direccion> direcciones = direccionRepository.findByUsuarioUsuarioId(usuarioId);
        System.out.println("Direcciones del usuario con ID " + usuarioId + ":");
        direcciones.forEach(System.out::println);

        System.out.println("== FIN PRUEBA: Encontrar todas las direcciones de un usuario ==");
        System.out.println();
    }

    @Test
    @Order(2)
    void pruebaBuscarDireccionPorIdYUsuarioTest() {
        System.out.println("== INICIO PRUEBA: Buscar dirección por ID y usuario ==");
        System.out.println("Alumno: " + ALUMNO);
        System.out.println("Prueba: buscar dirección por ID y usuario");
        System.out.println("-----");
    }
}
```
- File Path:** bazarboost > src > test > java > com > bazarboost > BazarboostDireccionTests

The screenshot shows the execution results of the BazarboostDireccionTests.java file:

- Run Tab:** The "Run" tab is selected, showing the test class "BazarboostDireccionTests" (com.bazarboost) and its methods: pruebaEncontrarTodasLasDireccionesDeUsuarioTest(), pruebaBuscarDireccionPorIdYUsuarioTest(), pruebaCrearOEditarDireccionTest(), and pruebaEliminarDireccionPorIdYUsuarioTest(). All tests passed in 1 second and 70 ms.
- Output Log:** The log output shows the test results and some VM warnings:

```
Tests passed: 4 of 4 tests - 1sec 70 ms
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has:
== INICIO PRUEBA: Encontrar todas las direcciones de un usuario ==
Alumno: FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ
Prueba: encontrar todas las direcciones de un usuario
-----
Direcciones del usuario con ID 1:
Direccion(direccionId=1, estado=Ciudad de México, ciudad=Benito Juárez, colonia=Del Valle, calle=Avenida Insurgentes)
== FIN PRUEBA: Encontrar todas las direcciones de un usuario ==
```

Captura 17. Clase de pruebas para *Direccion* y su salida de ejecución.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: BazarboostFacturaTests.java y su salida de ejecución

The screenshot shows two views of an IDE interface. The left view displays the project structure and the code for `BazarboostFacturaTests.java`. The right view shows the execution results of the test.

**Project View:**

- Project: bazarboost
- src/main/java/com/bazarboost: Contains `BazarboostApplication`, `CategoriaRepository`, `DescuentoRepository`, `DireccionRepository`, `FacturaRepository`, `MetodoPagoRepository`, `ProductoCarritoRepository`, `ProductoFacturaRepository`, `ProductoRepository`, `ReseniaRepository`, `RolRepository`, `UsuarioRepository`, and `UsuarioRolRepository`.
- src/test/java/com/bazarboost: Contains `BazarboostApplicationTests`, `BazarboostCategoriaTests`, `BazarboostDescuentoTests`, `BazarboostDireccionTests`, and `BazarboostFacturaTests`.
- resources: Contains `application.properties`, `data.sql`, and `schema.sql`.

**Code View (BazarboostFacturaTests.java):**

```
/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
@SpringBootTest
@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
@Sql({"/schema.sql", "/data.sql"})
public class BazarboostFacturaTests {

    private static final String ALUMNO = "FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ"; 4 usages

    @Autowired 10 usages
    private FacturaRepository facturaRepository;

    @Test new *
    @Order(1)
    void pruebaObtenerFacturasPorUsuarioConPaginacionTest() {
        System.out.println("== INICIO PRUEBA: Obtener facturas por usuario con paginación ==");
        System.out.println("Alumno: " + ALUMNO);
        System.out.println("Prueba: obtener facturas por usuario con paginación");
        System.out.println("-----");

        // Usuario de ejemplo
        Integer usuarioId = 1;
        Pageable pageable = PageRequest.of( pageNumber: 0, pageSize: 5); // Página 0, 5 facturas por página

        // Obtener y mostrar facturas de la primera página
        Page<Factura> facturasPage1 = facturaRepository.findByUsuarioUserId(usuarioId, pageable);
        System.out.println("Primera página de facturas del usuario con ID " + usuarioId + ":");
        facturasPage1.getContent().forEach(System.out::println);

        // Obtener y mostrar facturas de la segunda página
        pageable = PageRequest.of( pageNumber: 1, pageSize: 5); // Página 1, 5 facturas por página
        Page<Factura> facturasPage2 = facturaRepository.findByUsuarioUserId(usuarioId, pageable);
        System.out.println("Segunda página de facturas del usuario con ID " + usuarioId + ":");
        facturasPage2.getContent().forEach(System.out::println);

        System.out.println("== FIN PRUEBA: Obtener facturas por usuario con paginación ==");
        System.out.println();
    }
}
```

**Execution Results View:**

- Run Configuration: BazarboostFacturaTests
- Test Results:
  - BazarboostFacturaTests (com.bazarboost) 5 sec 672 ms
    - pruebaObtenerFacturasPorUsuarioConPaginacionTest() 2 sec 36 ms
    - pruebaOrdenarFacturasPorMontoDescConPaginacionT1 sec 158 ms
    - pruebaOrdenarFacturasPorMontoAscConPaginacionT1 sec 167 ms
    - pruebaCrearEditarFacturaTest() 1 sec 311 ms
  - Tests passed: 4 of 4 tests ~ 5 sec 672 ms
- Output Log:

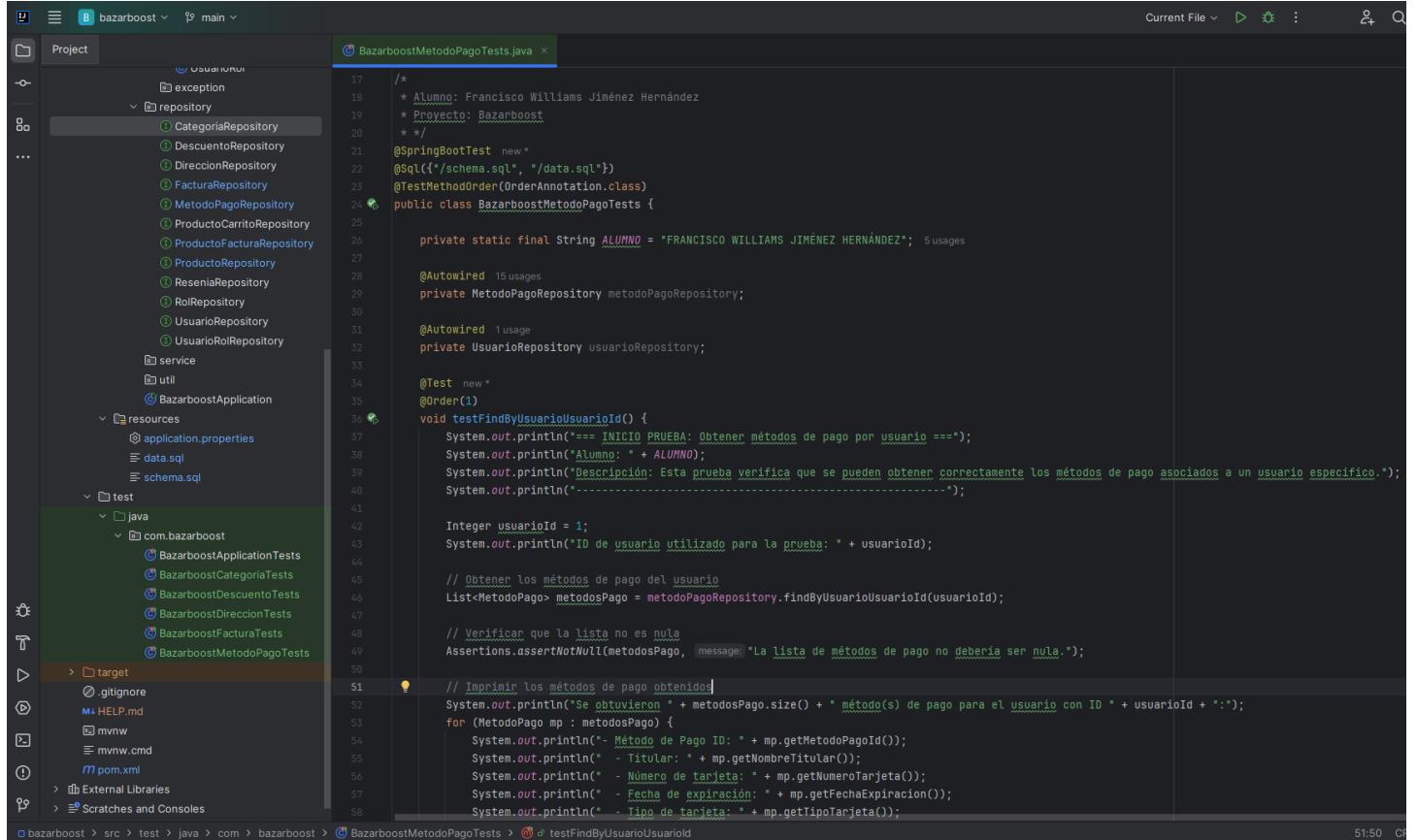
```
OpenJDK 64-Bit Server VM warning: Sharing is only supported for boot loader classes because bootstrap classpath has been appended
== INICIO PRUEBA: Obtener facturas por usuario con paginación ==
Alumno: FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ
Prueba: obtener facturas por usuario con paginación
-----
Primera página de facturas del usuario con ID 1:
Factura(facturaId=1, fecha=2024-10-11T10:00, subtotal=150.0, total=165.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=2, fecha=2024-10-11T10:01, subtotal=160.0, total=176.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=3, fecha=2024-10-11T10:02, subtotal=170.0, total=187.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=4, fecha=2024-10-11T10:03, subtotal=180.0, total=198.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=5, fecha=2024-10-11T10:04, subtotal=190.0, total=209.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Segunda página de facturas del usuario con ID 1:
Factura(facturaId=6, fecha=2024-10-11T10:05, subtotal=200.0, total=220.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=7, fecha=2024-10-11T10:06, subtotal=210.0, total=231.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=8, fecha=2024-10-11T10:07, subtotal=220.0, total=242.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=9, fecha=2024-10-11T10:08, subtotal=230.0, total=253.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
Factura(facturaId=10, fecha=2024-10-11T10:09, subtotal=240.0, total=264.0, porcentajeImpuestos=10, usuario=Usuario(usuarioId=1, nombre=Juan, apellidoPaterno=Pérez
== FIN PRUEBA: Obtener facturas por usuario con paginación ==
```

Captura 18. Clase de pruebas para *Factura* y su salida de ejecución.

# Francisco Williams Jiménez Hernández

## Ejercicio 2

Archivo: BazarboostMetodoPagoTests.java y su salida de ejecución



```
/*
 * Alumno: Francisco Williams Jiménez Hernández
 * Proyecto: Bazarboost
 */
@SpringBootTest
@Sql({"/schema.sql", "/data.sql"})
@TestMethodOrder(OrderAnnotation.class)
public class BazarboostMetodoPagoTests {

    private static final String ALUMNO = "FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ"; 5 usages

    @Autowired 15 usages
    private MetodoPagoRepository metodoPagoRepository;

    @Autowired 1 usage
    private UsuarioRepository usuarioRepository;

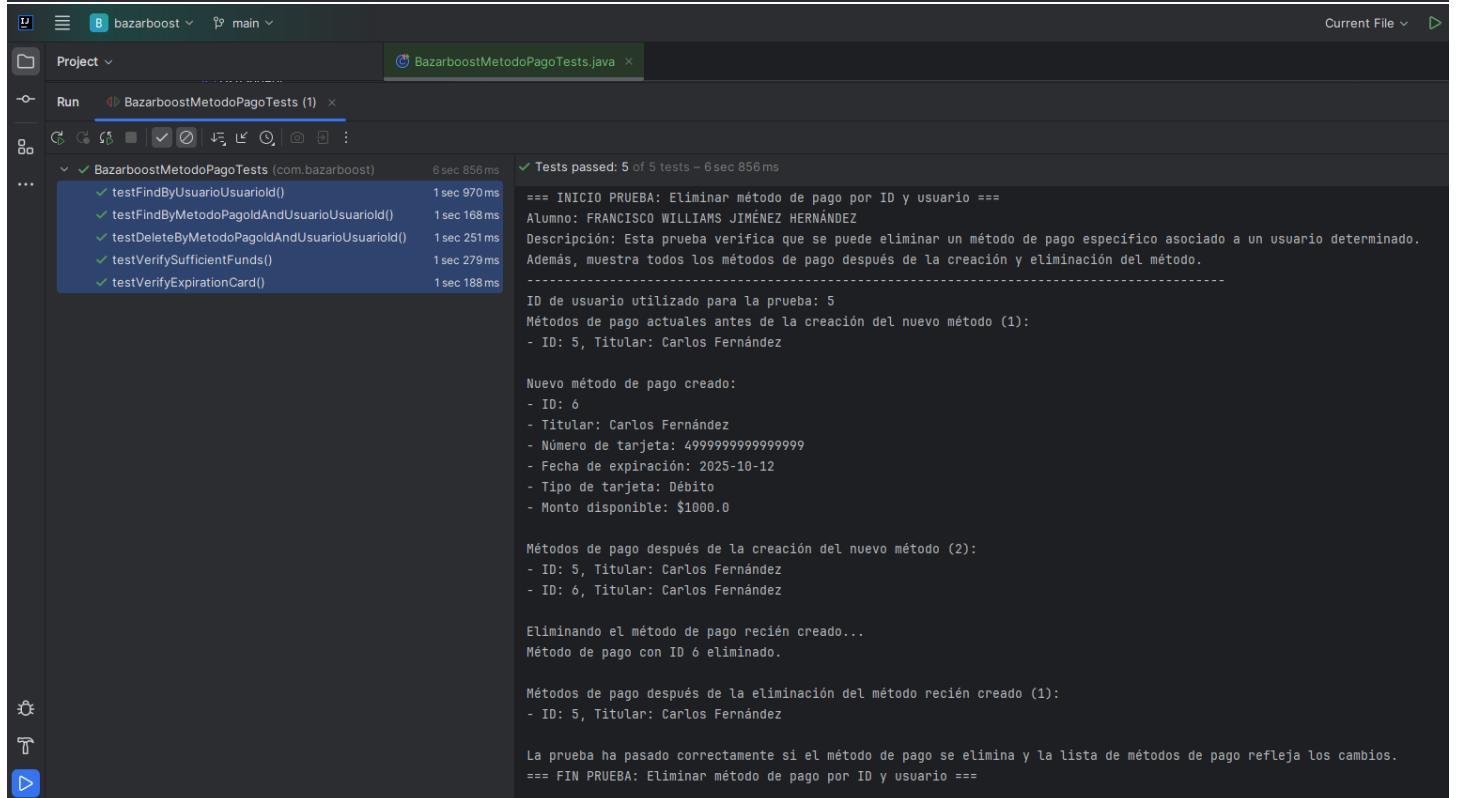
    @Test new *
    @Order(1)
    void testFindByUsuarioUserId() {
        System.out.println("== INICIO PRUEBA: Obtener métodos de pago por usuario ===");
        System.out.println("Alumno: " + ALUMNO);
        System.out.println("Descripción: Esta prueba verifica que se pueden obtener correctamente los métodos de pago asociados a un usuario específico.");
        System.out.println("-----");

        Integer userId = 1;
        System.out.println("ID de usuario utilizado para la prueba: " + userId);

        // Obtener los métodos de pago del usuario
        List<MetodoPago> metodosPago = metodoPagoRepository.findByUsuarioUserId(userId);

        // Verificar que la lista no es nula
        Assertions.assertNotNull(metodosPago, message: "La lista de métodos de pago no debería ser nula");

        // Imprimir los métodos de pago obtenidos
        System.out.println("Se obtuvieron " + metodosPago.size() + " método(s) de pago para el usuario con ID " + userId + ":");
        for (MetodoPago mp : metodosPago) {
            System.out.println(" - Método de Pago ID: " + mp.getId());
            System.out.println(" - Titular: " + mp.getTitular());
            System.out.println(" - Número de tarjeta: " + mp.getNumeroTarjeta());
            System.out.println(" - Fecha de expiración: " + mp.getFechaExpiracion());
            System.out.println(" - Tipo de tarjeta: " + mp.getTipoTarjeta());
        }
    }
}
```



Test Method	Time	Output
testFindByUsuarioUserId()	1sec 970ms	== INICIO PRUEBA: Eliminar método de pago por ID y usuario == Alumno: FRANCISCO WILLIAMS JIMÉNEZ HERNÁNDEZ Descripción: Esta prueba verifica que se puede eliminar un método de pago específico asociado a un usuario determinado. Además, muestra todos los métodos de pago después de la creación y eliminación del método.
testFindByMetodoPagoIdAndUsuarioUserId()	1sec 168ms	ID de usuario utilizado para la prueba: 5 Métodos de pago actuales antes de la creación del nuevo método (1): - ID: 5, Titular: Carlos Fernández
testDeleteByMetodoPagoIdAndUsuarioUserId()	1sec 251ms	Nuevo método de pago creado: - ID: 6 - Titular: Carlos Fernández - Número de tarjeta: 4999999999999999 - Fecha de expiración: 2025-10-12 - Tipo de tarjeta: Débito - Monto disponible: \$1000.0
testVerifySufficientFunds()	1sec 279ms	Métodos de pago después de la creación del nuevo método (2): - ID: 5, Titular: Carlos Fernández - ID: 6, Titular: Carlos Fernández
testVerifyExpirationCard()	1sec 188ms	Eliminando el método de pago recién creado... Método de pago con ID 6 eliminado.

Métodos de pago después de la eliminación del método recién creado (1):  
- ID: 5, Titular: Carlos Fernández

La prueba ha pasado correctamente si el método de pago se elimina y la lista de métodos de pago refleja los cambios.  
== FIN PRUEBA: Eliminar método de pago por ID y usuario ==

Captura 19. Clase de pruebas para *MetodoPago* y su salida de ejecución.