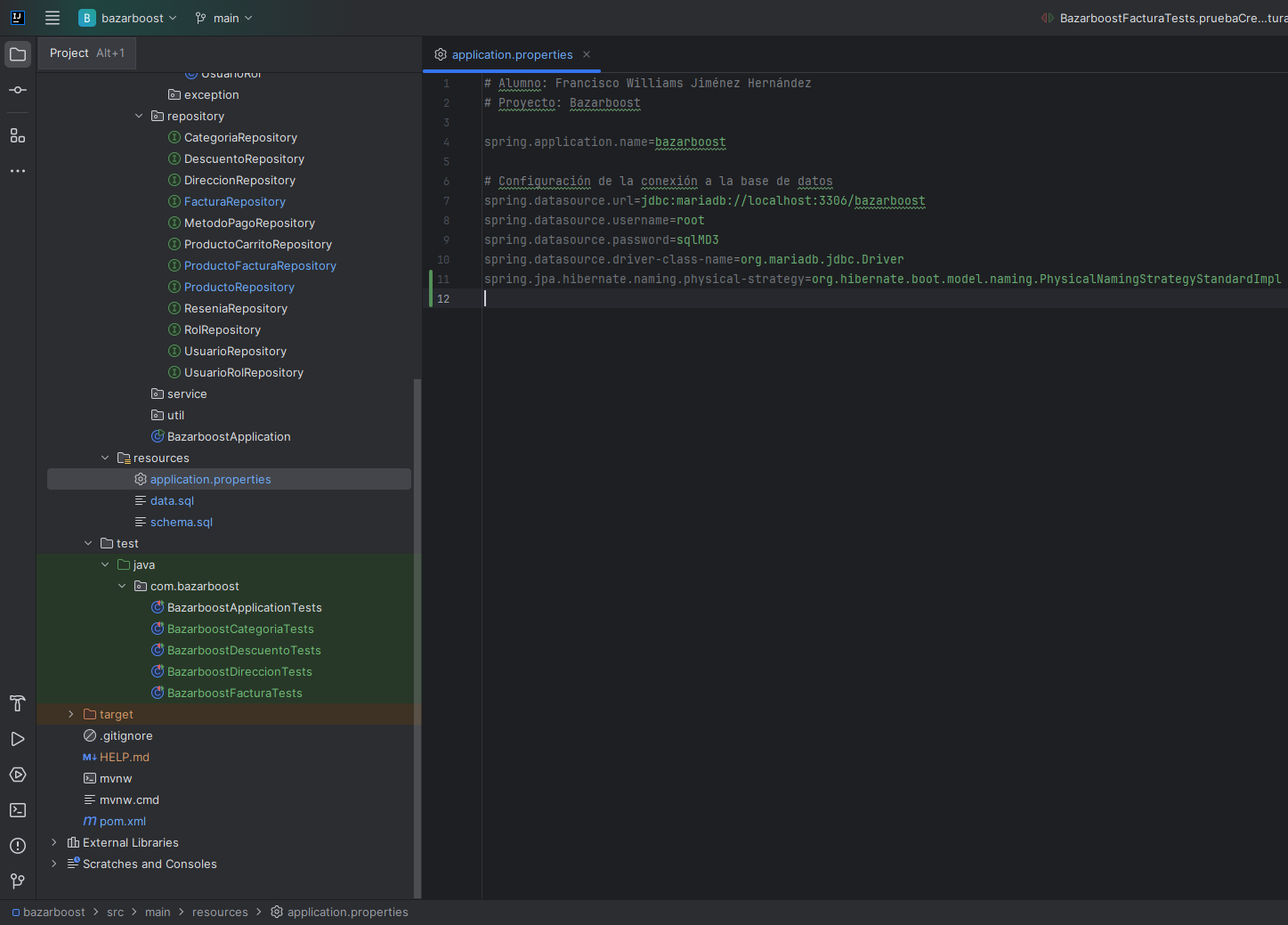
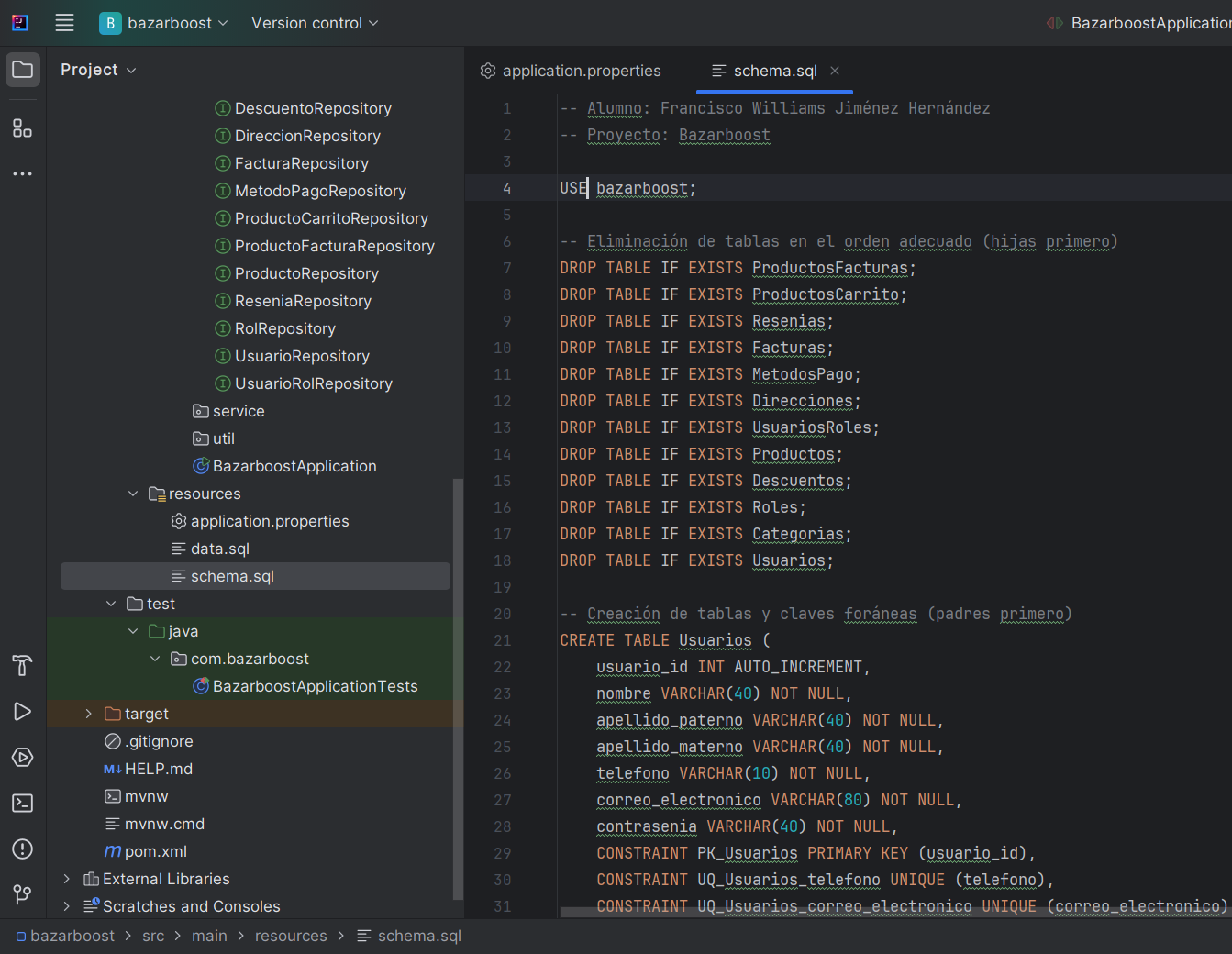
1. **Configuración inicial**

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**

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

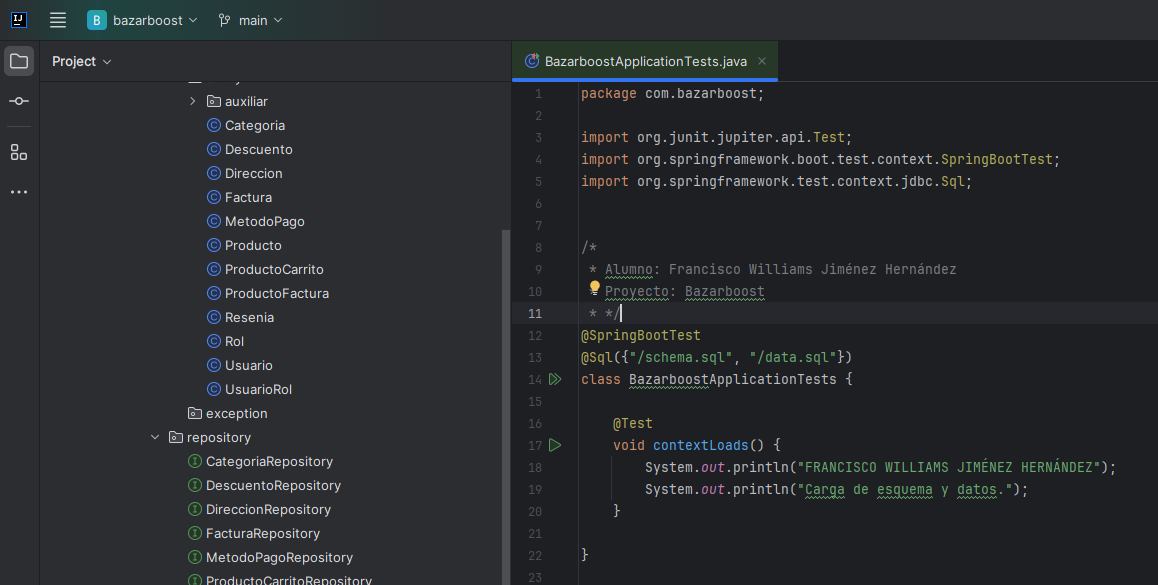
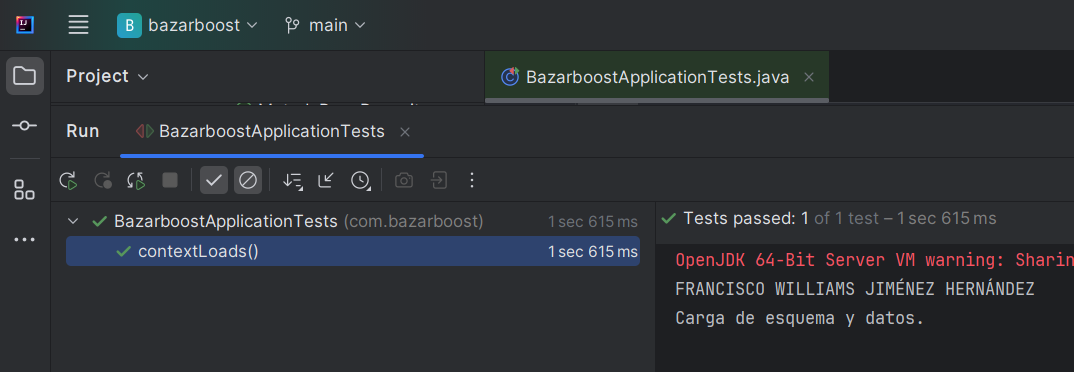
Posteriormente, elaboré los scripts **schema.sql** y **data.sql** para la creación del esquema de la base de datos y la inserción   
de los datos de prueba. Estos archivos fueron ubicados en la carpeta **resources** del proyecto. A continuación, se presenta   
el contenido de ambos archivos.  
  
**Archivo: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

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

**Archivo: data.sql**

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

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.

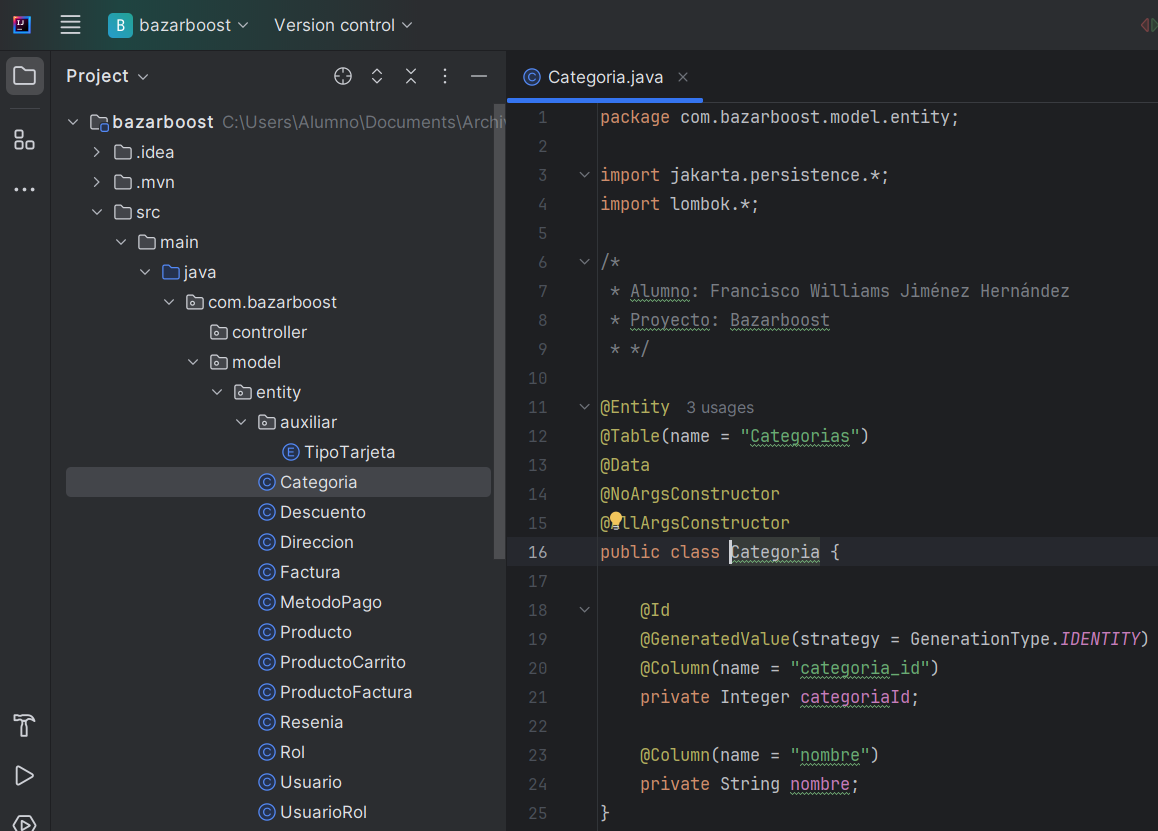
  


Captura 4. Código del archivo BazarboostApplicationTests.java y su salida de ejecución para la creación del   
esquema y la carga de datos de prueba.

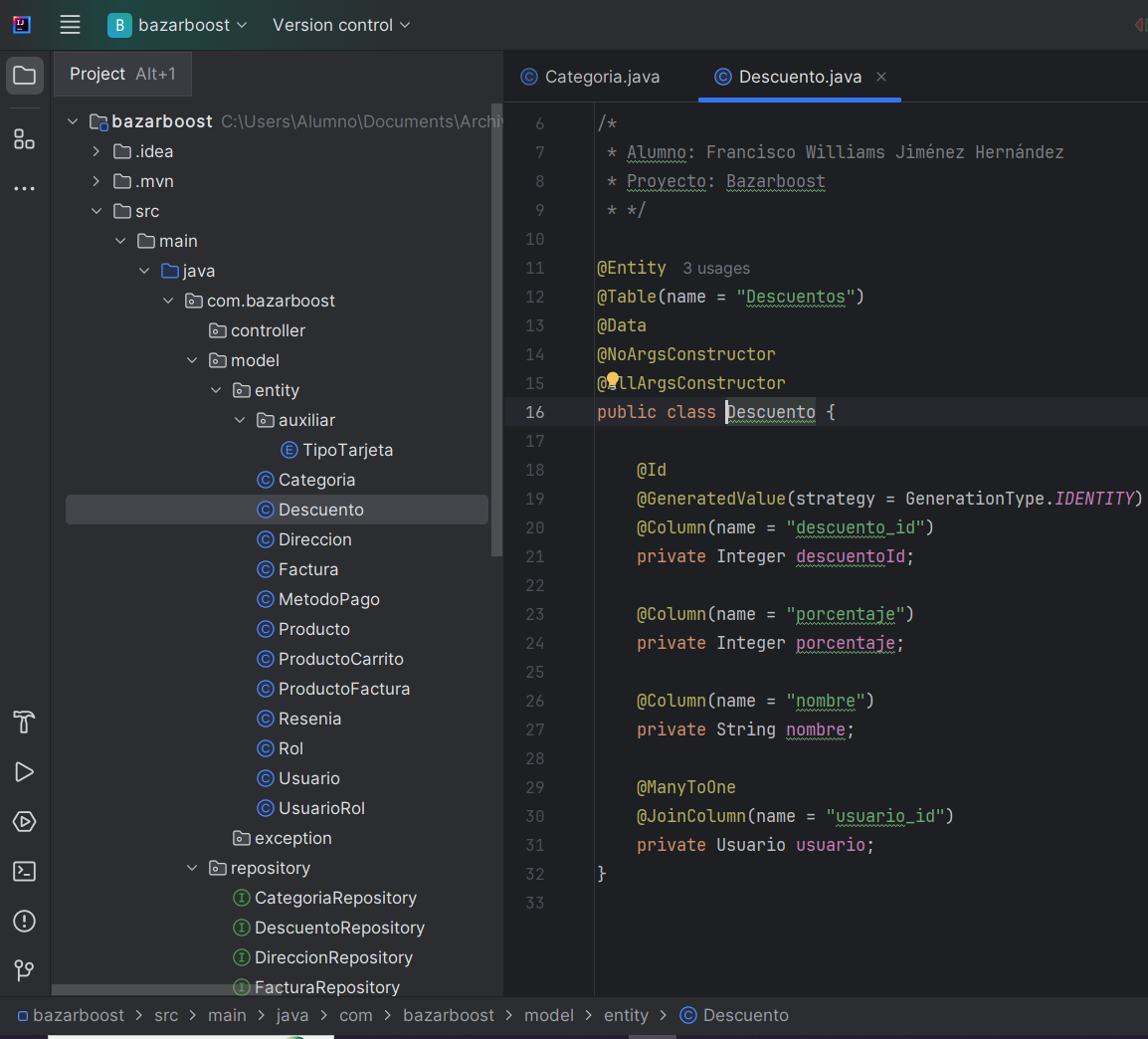
1. **Creación de entidades**

Una vez completadas las configuraciones iniciales para la conexión a la base de datos, así como la carga del esquema y  
los datosde 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.

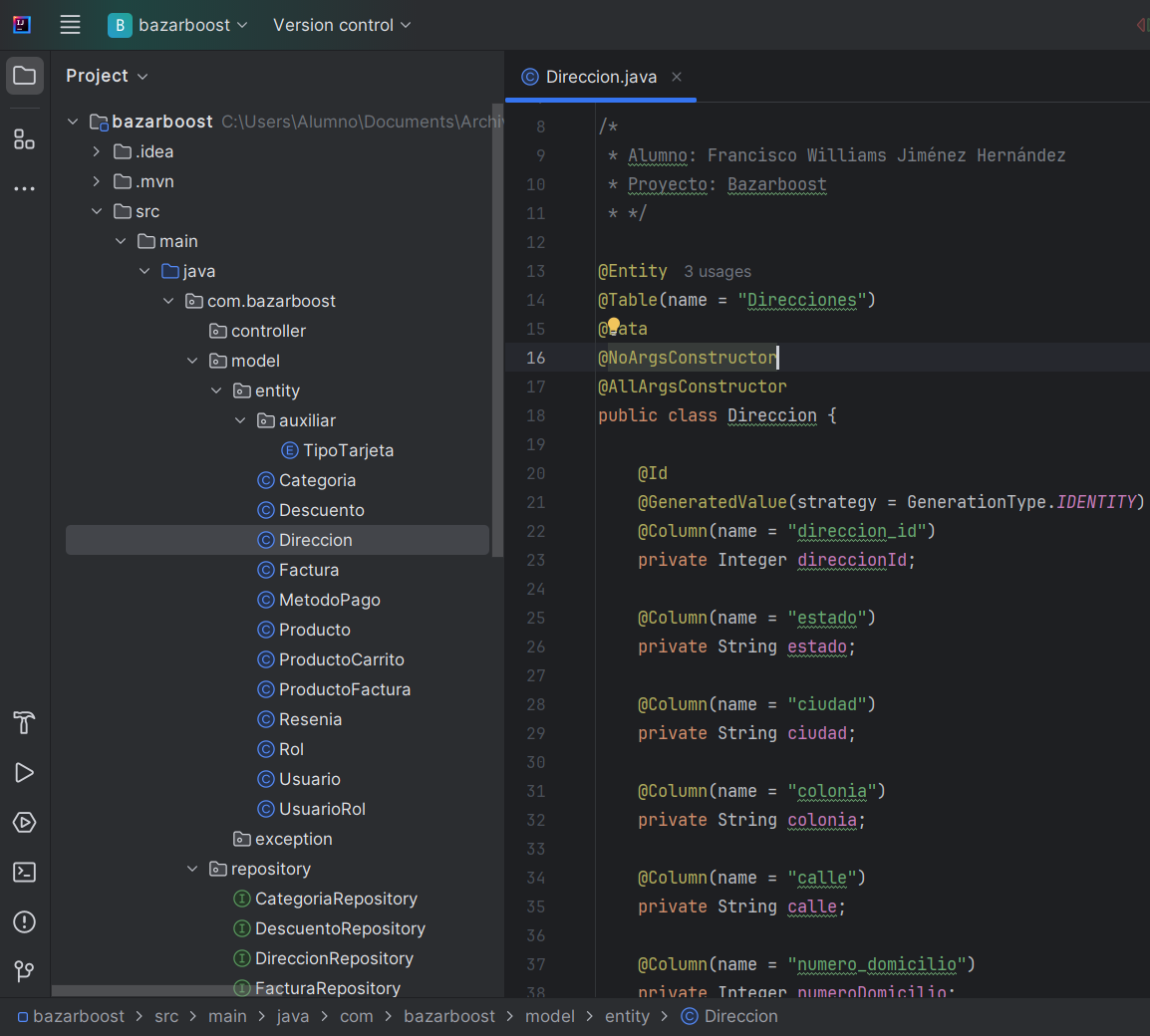
**Nota:** Se muestran 5 entidades de las 12 totales de mi proyecto con el fin de no extender demasiado el documento a la  
vez que se cumple con los requerimientos del PDF.  
  
**Archivo: Categoria.java**

  
Captura 5. Clase de entidad para *Categoria*.

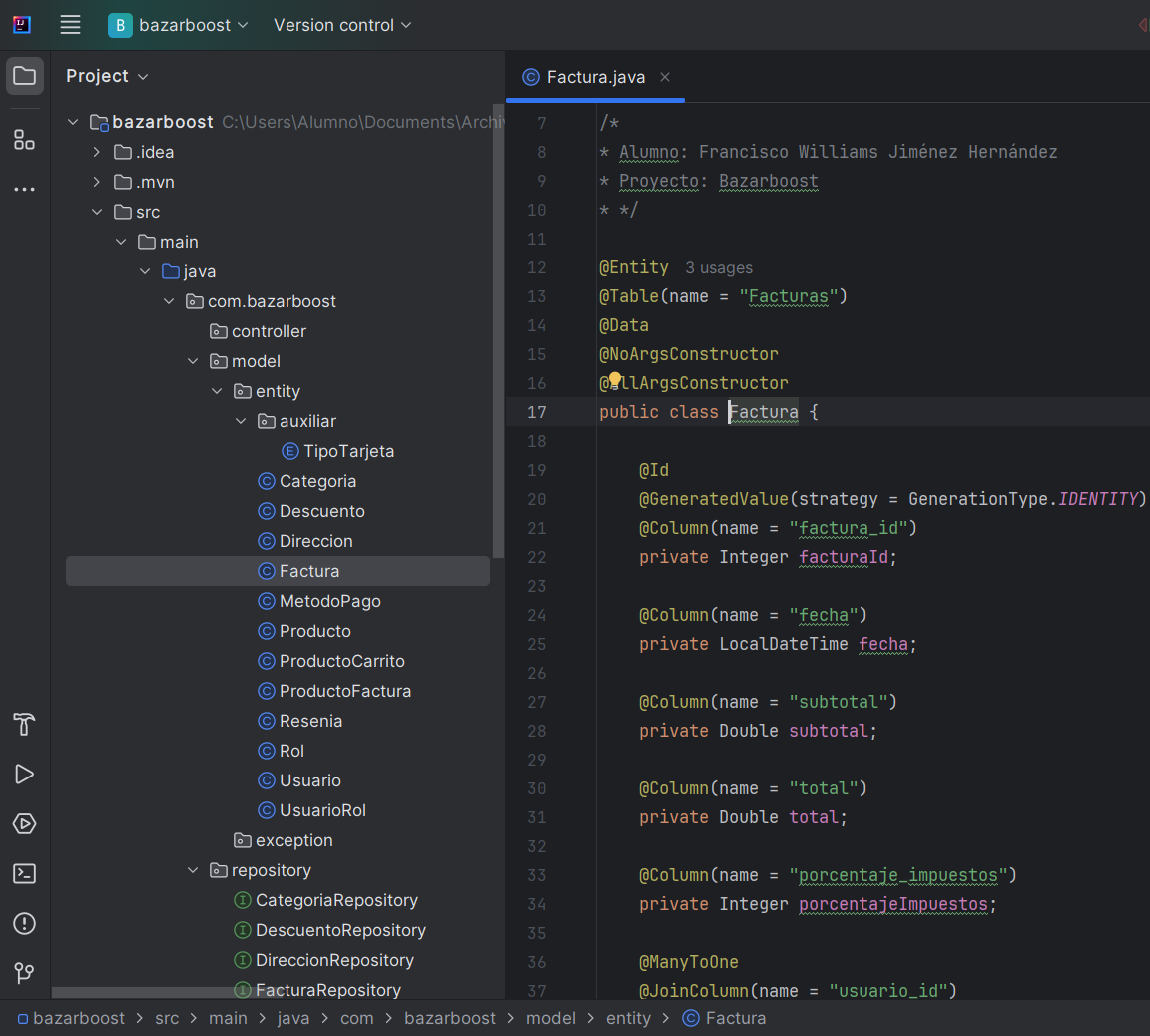
**Archivo: Descuento.java**

  
Captura 6. Clase de entidad para *Descuento*.

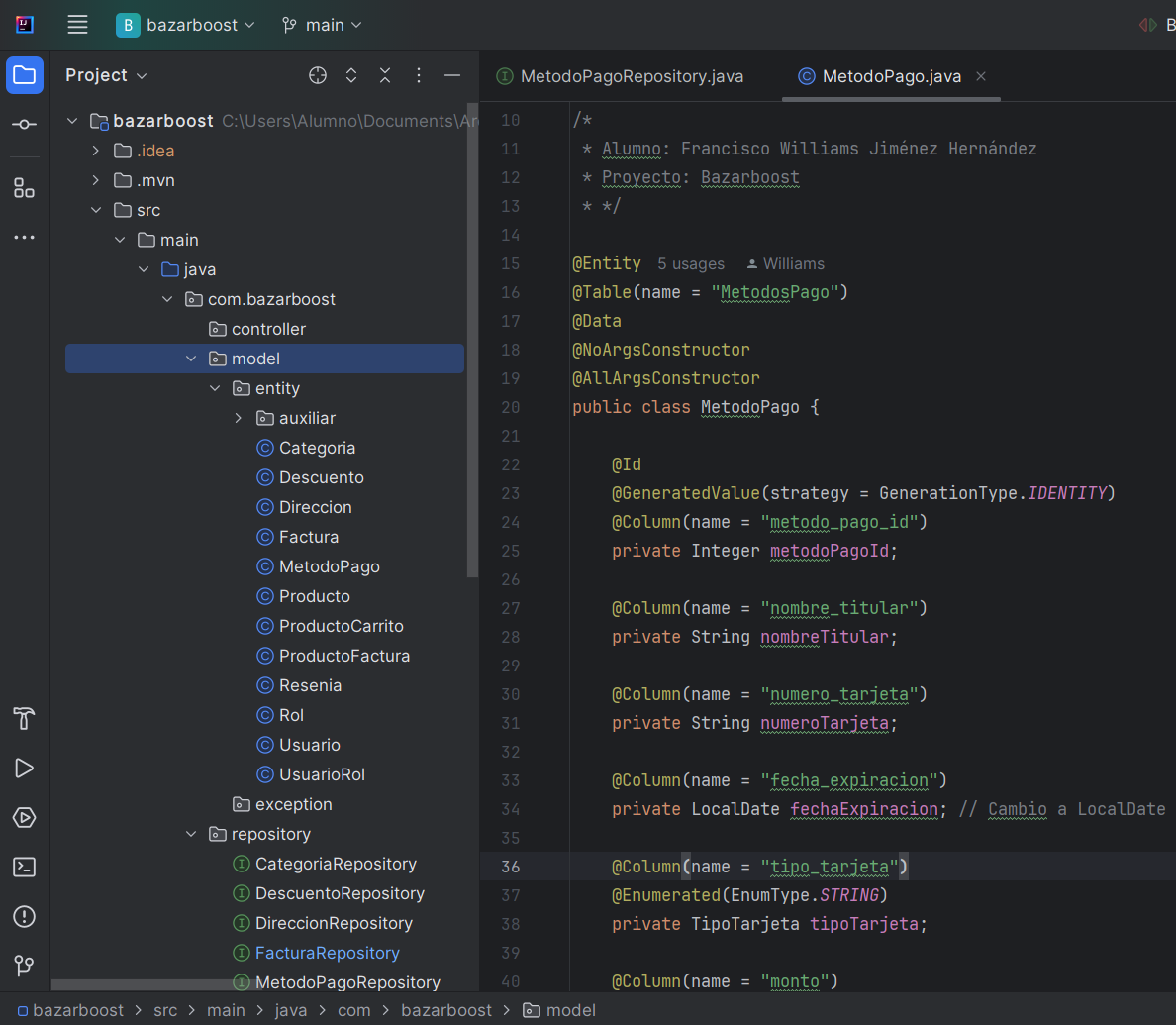
**Archivo: Direccion.java**

  
Captura 7. Clase de entidad para *Direccion*.

**Archivo: Factura.java**

  
Captura 8. Clase de entidad para *Factura*.

**Archivo: MetodoPago.java**

  
Captura 9. Clase de entidad para *MetodoPago*.

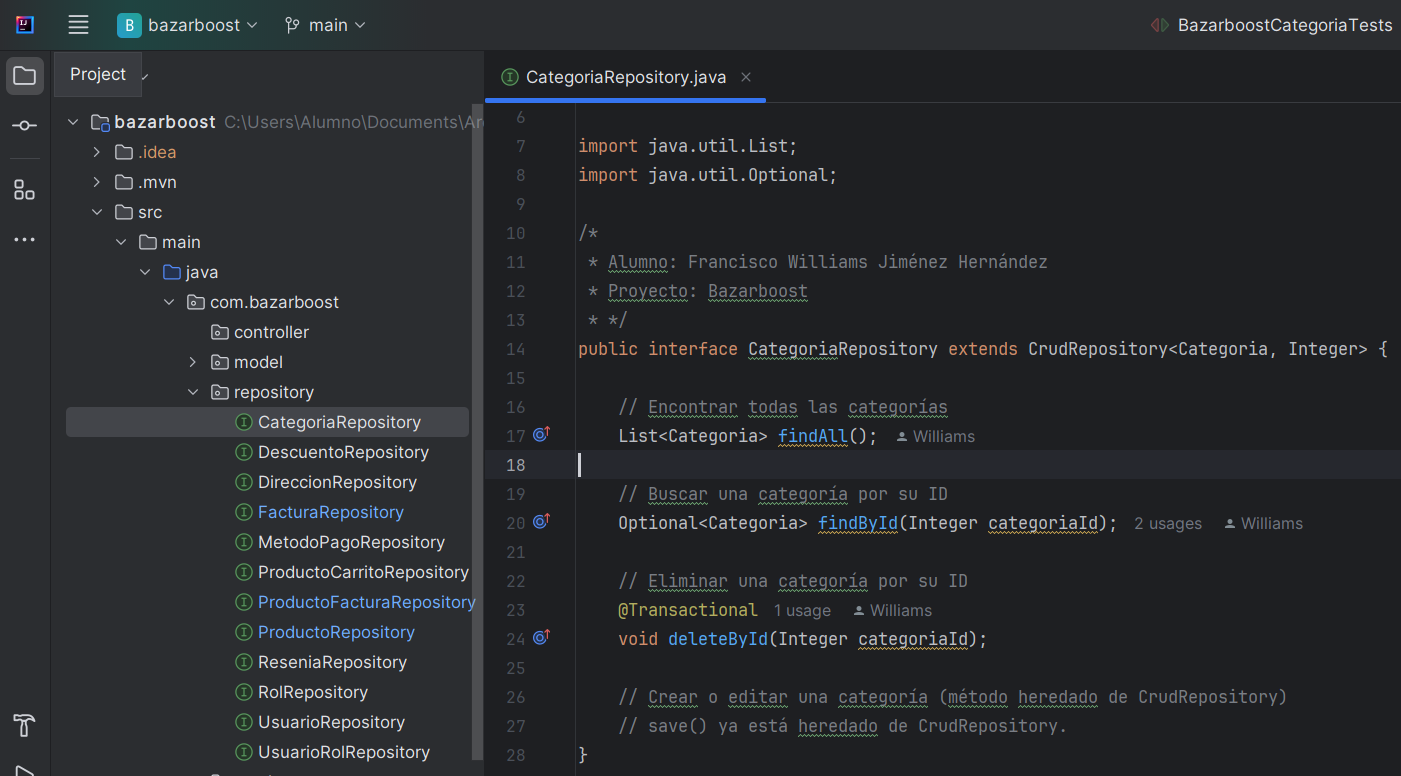
1. **Creación de repositorios**

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.

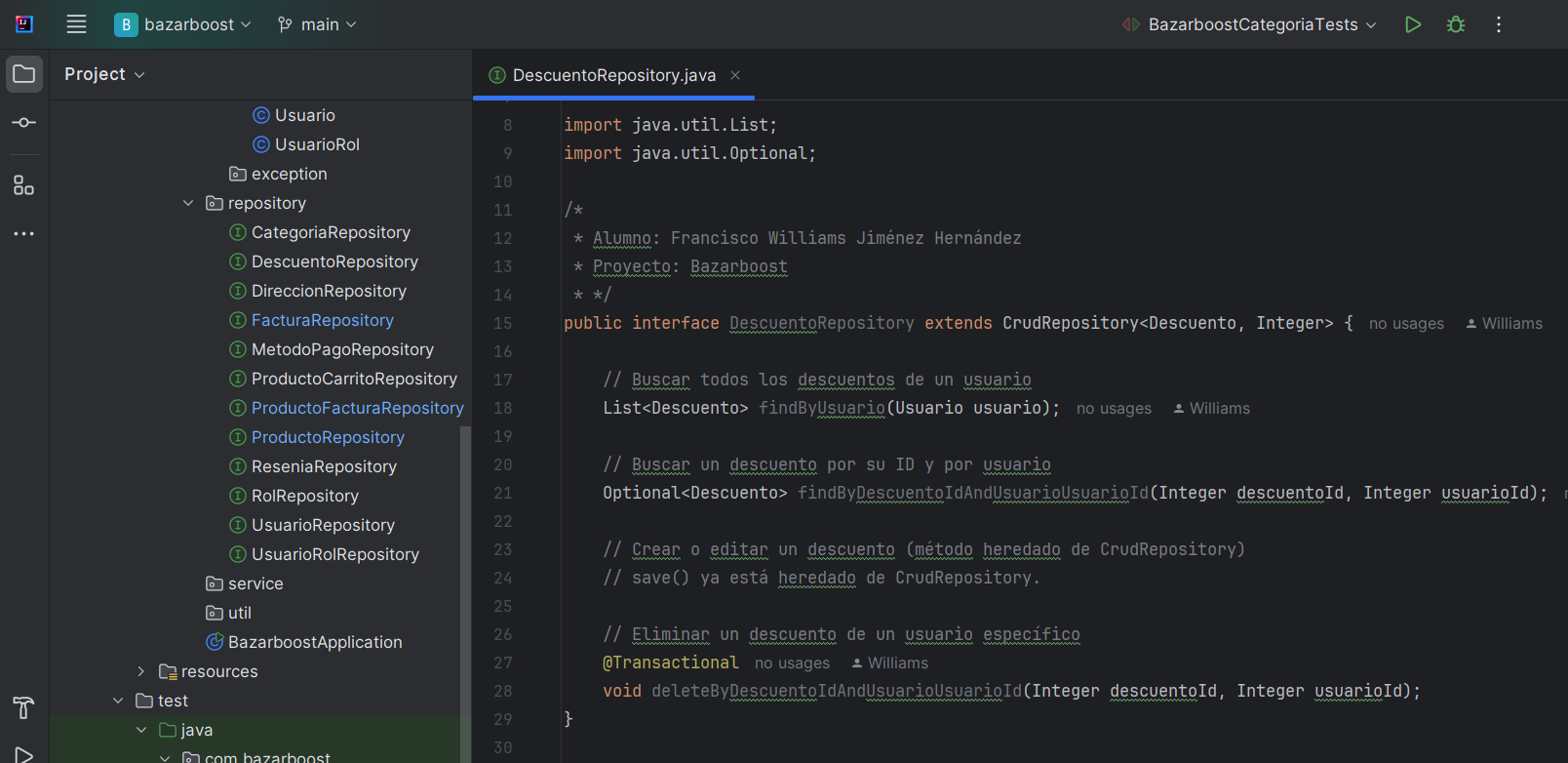
El total de consultas de cada tipo entre las cinco entidades es el siguiente:

* Derivadas: 7
* Nombradas: 7
* De relación: 14

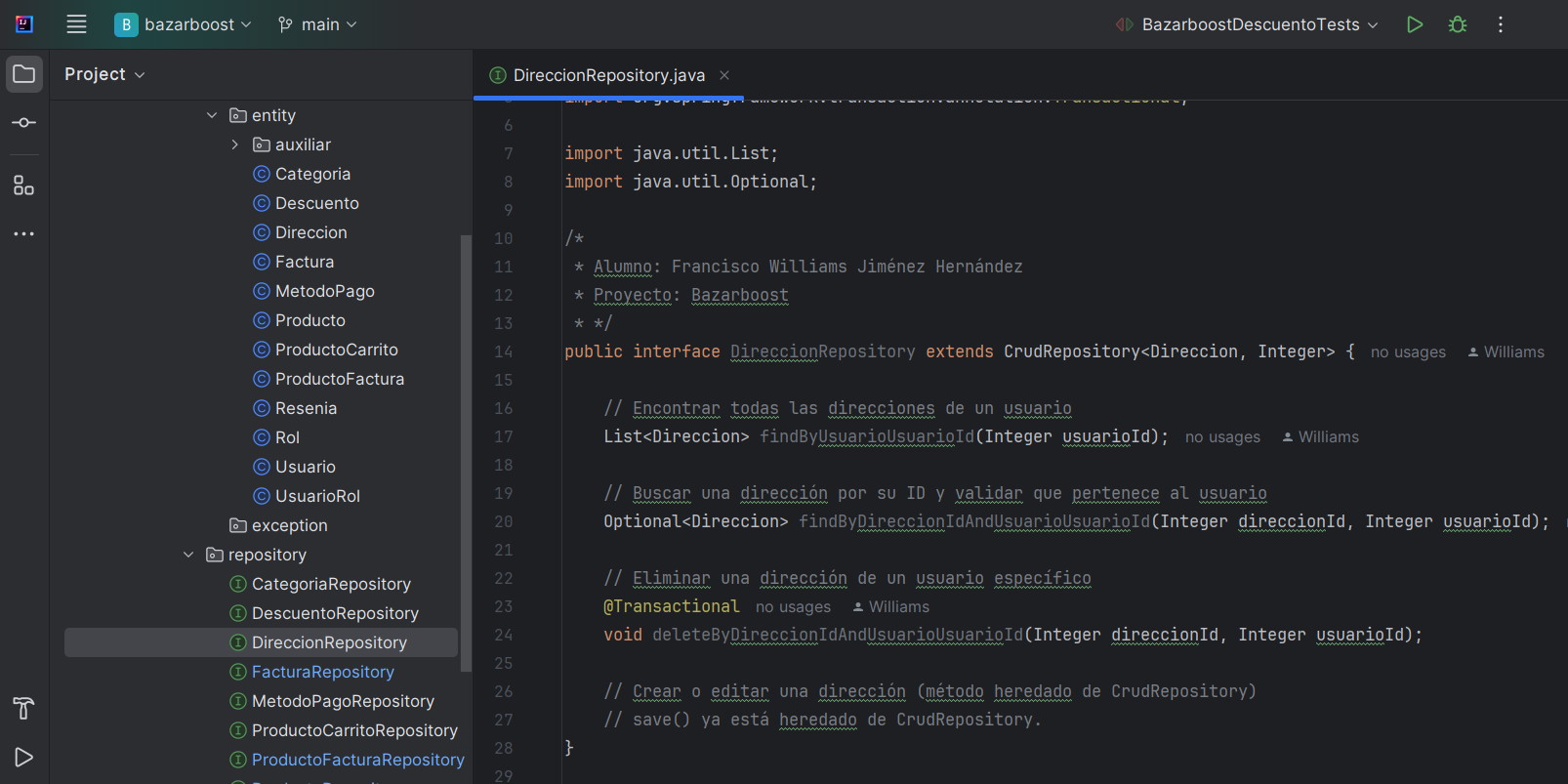
**Archivo:** **CategoriaRepository.java**

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

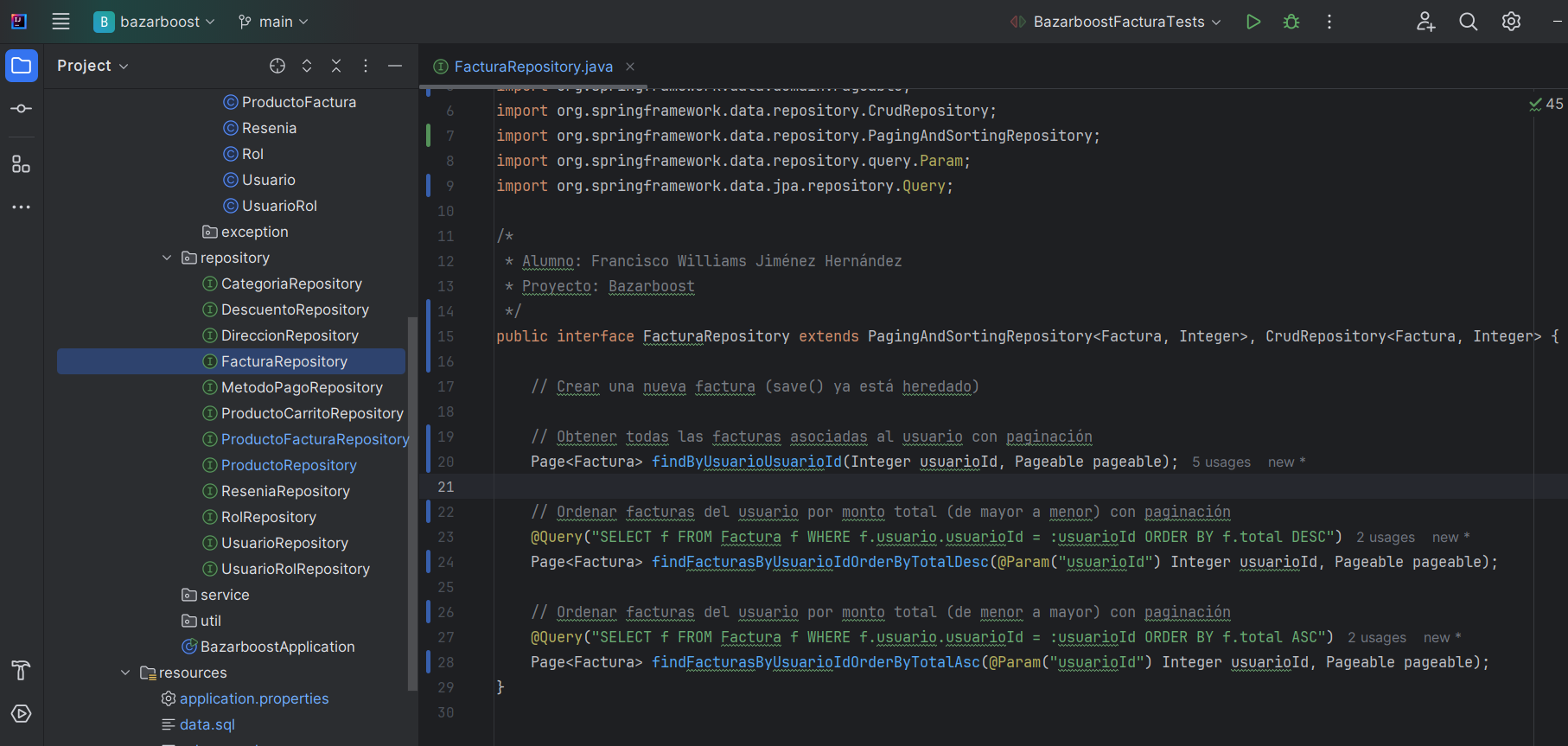
**Archivo:** **DescuentoRepository.java**

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

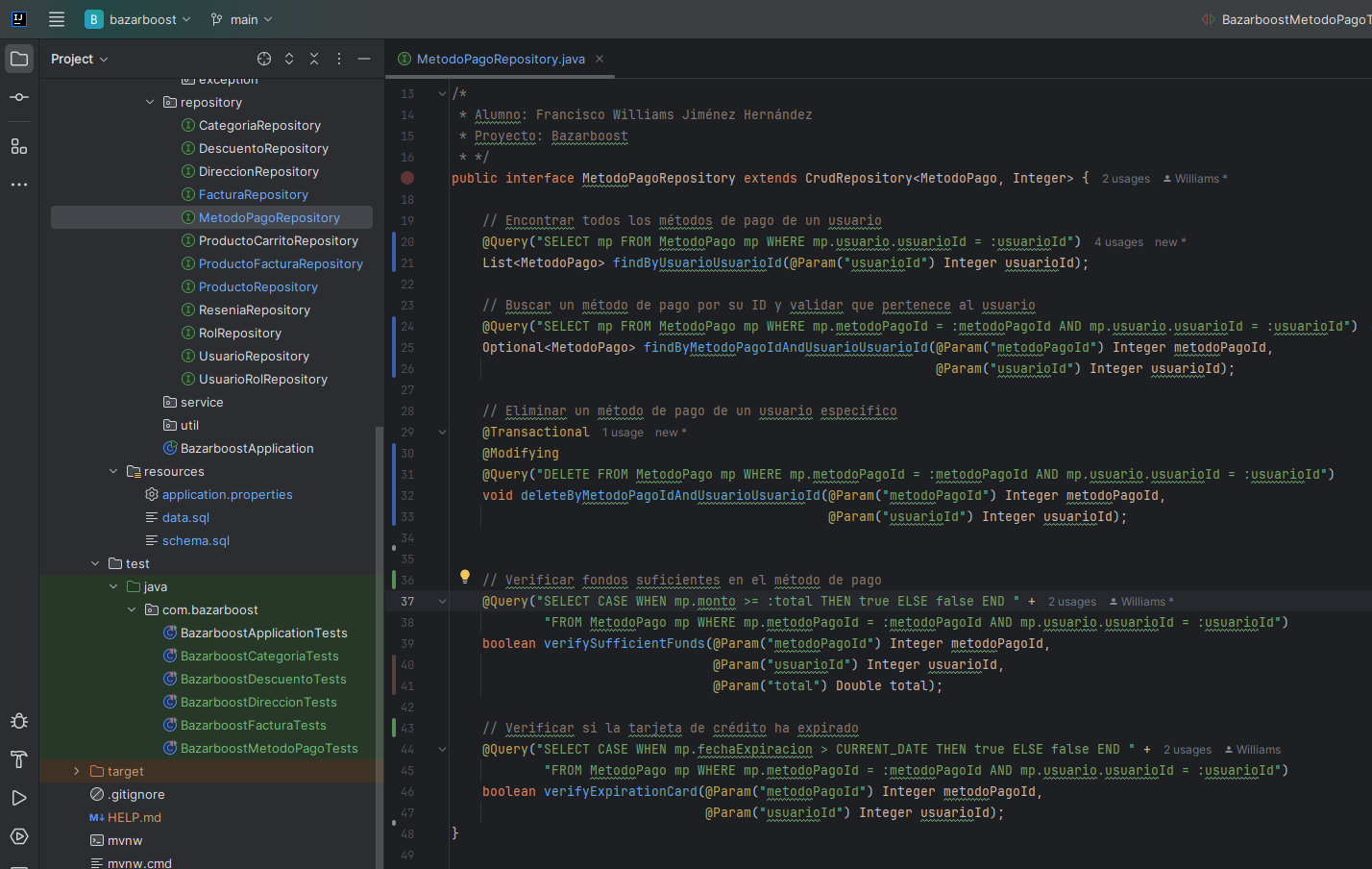
**Archivo: DireccionRepository.java**

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

**Archivo: FacturaRepository.java**

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

**Archivo: MetodoPagoRepository.java**

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

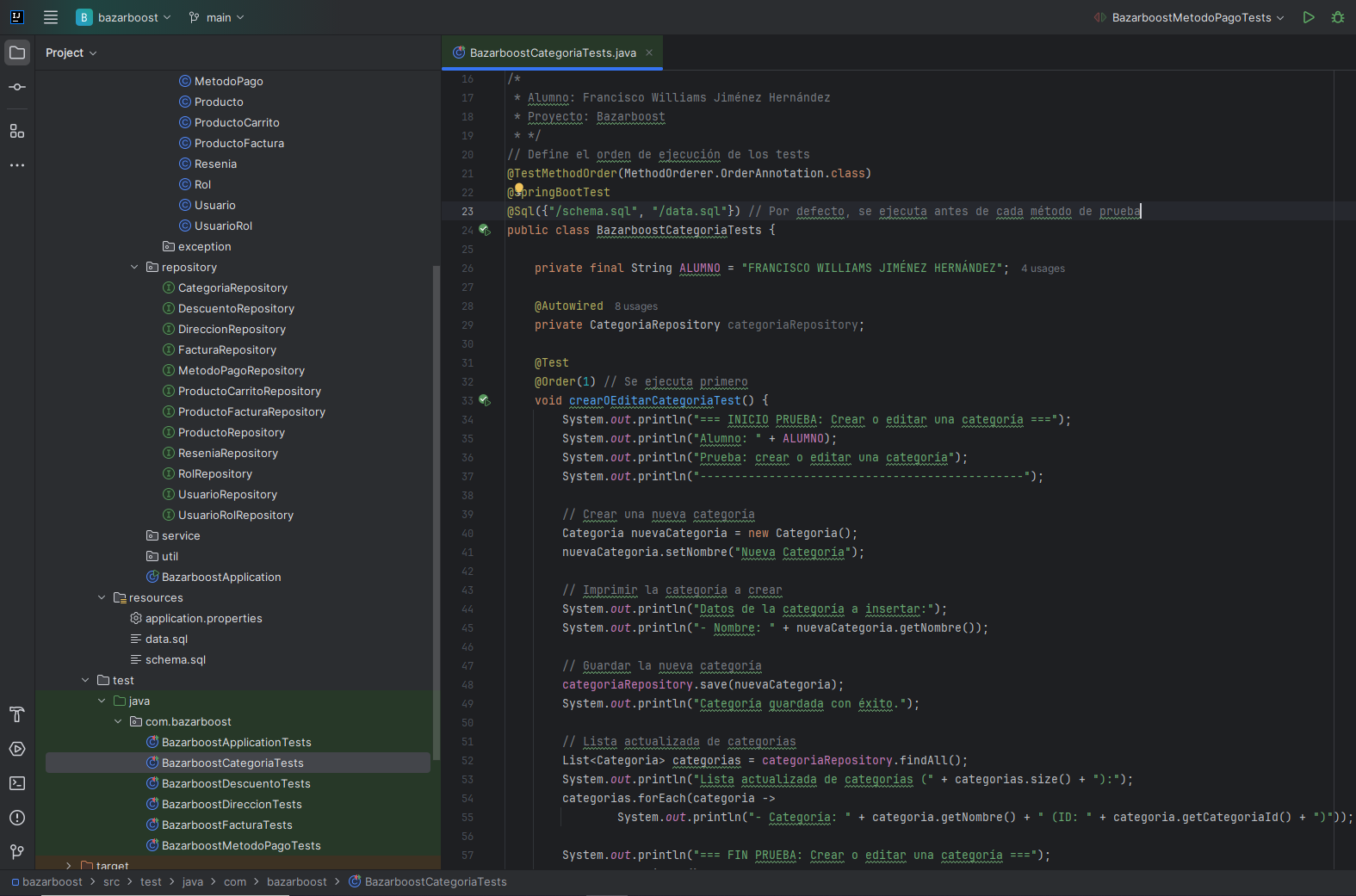
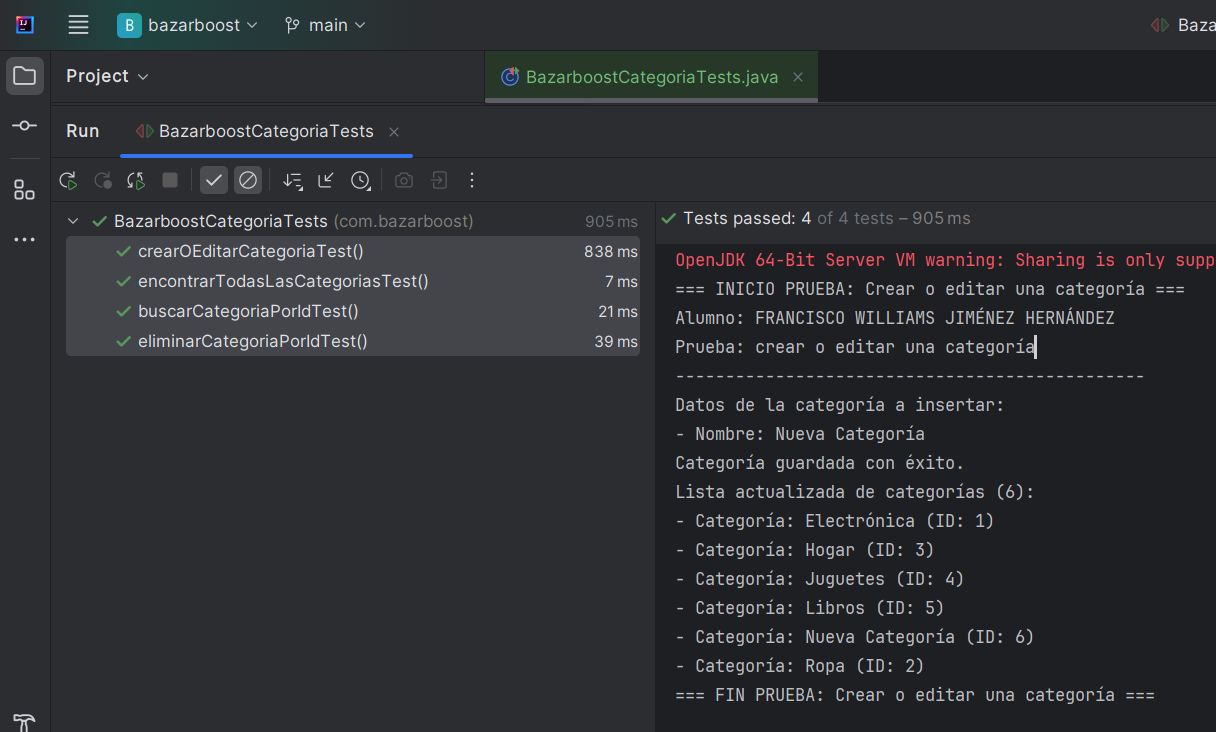
1. **Clases de pruebas y salidas de ejecución**

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.

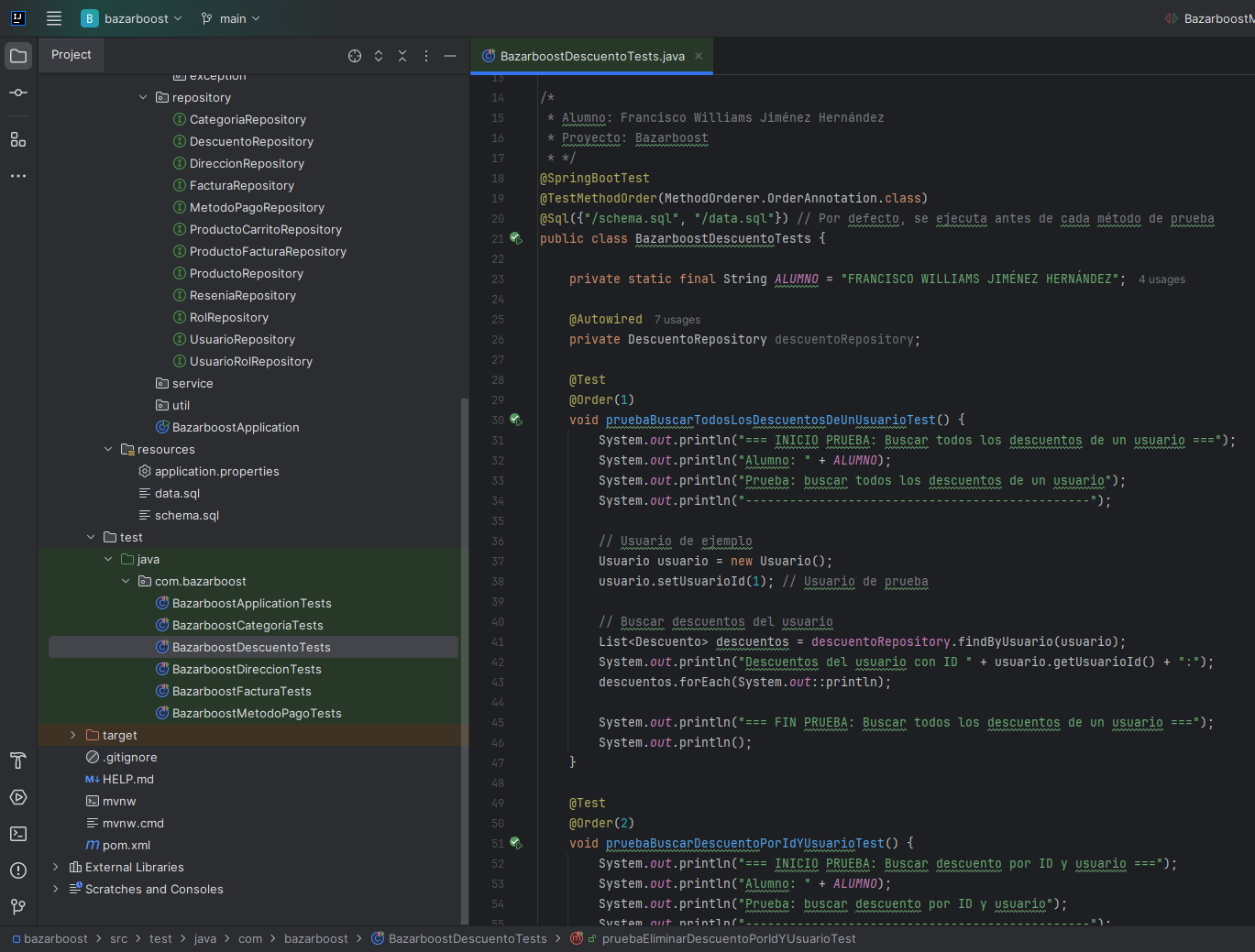
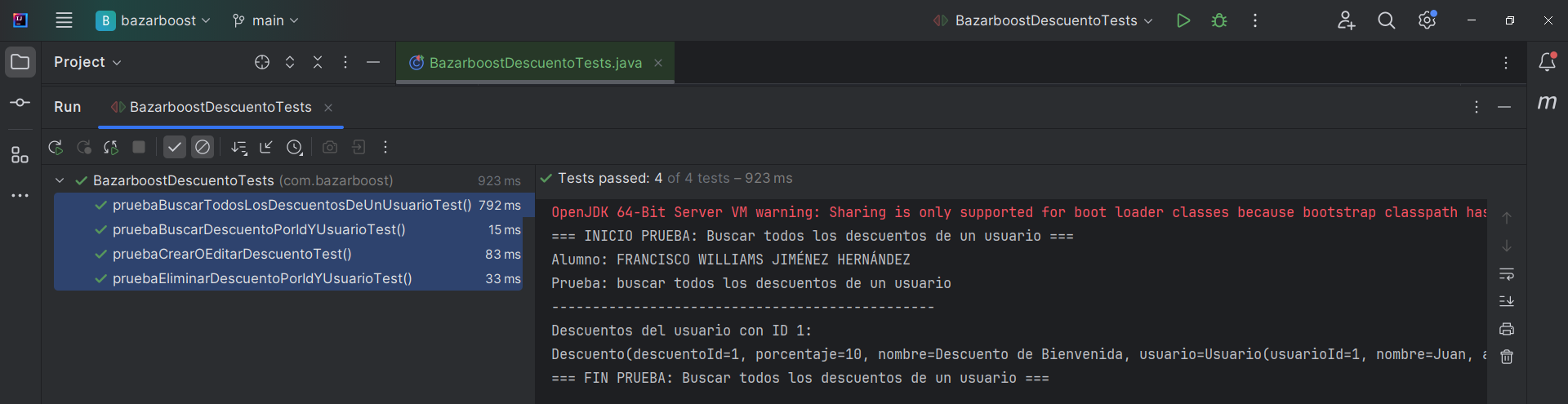
***Las capturas comienzan en la siguiente página.***

**Archivo:** **BazarboostCategoriaTests.java y su salida de ejecución**

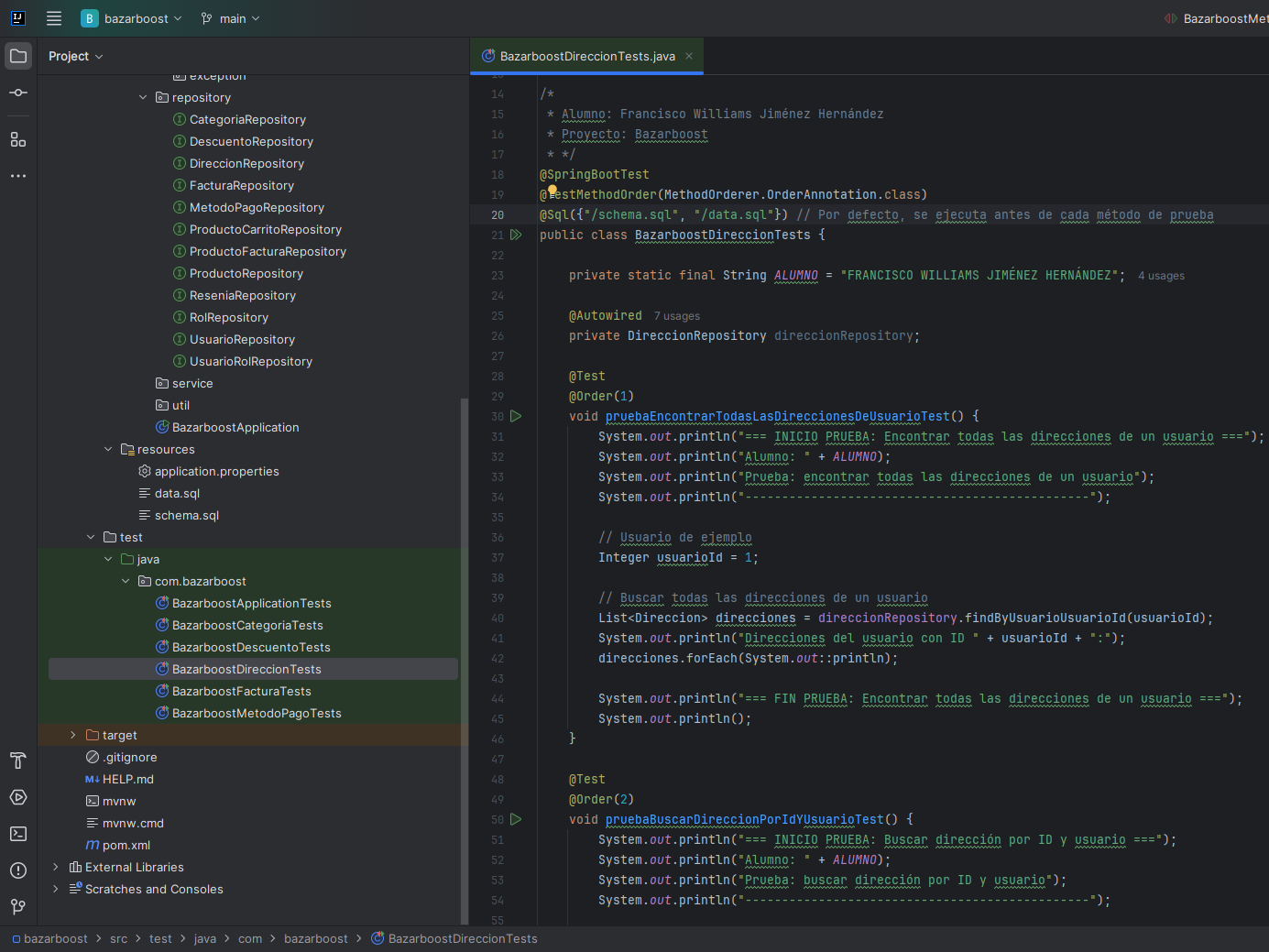
Captura 15. Clase de pruebas para *Categoria* y su salida de ejecución.

**Archivo:** **BazarboostDescuentoTests.java y su salida de ejecución**

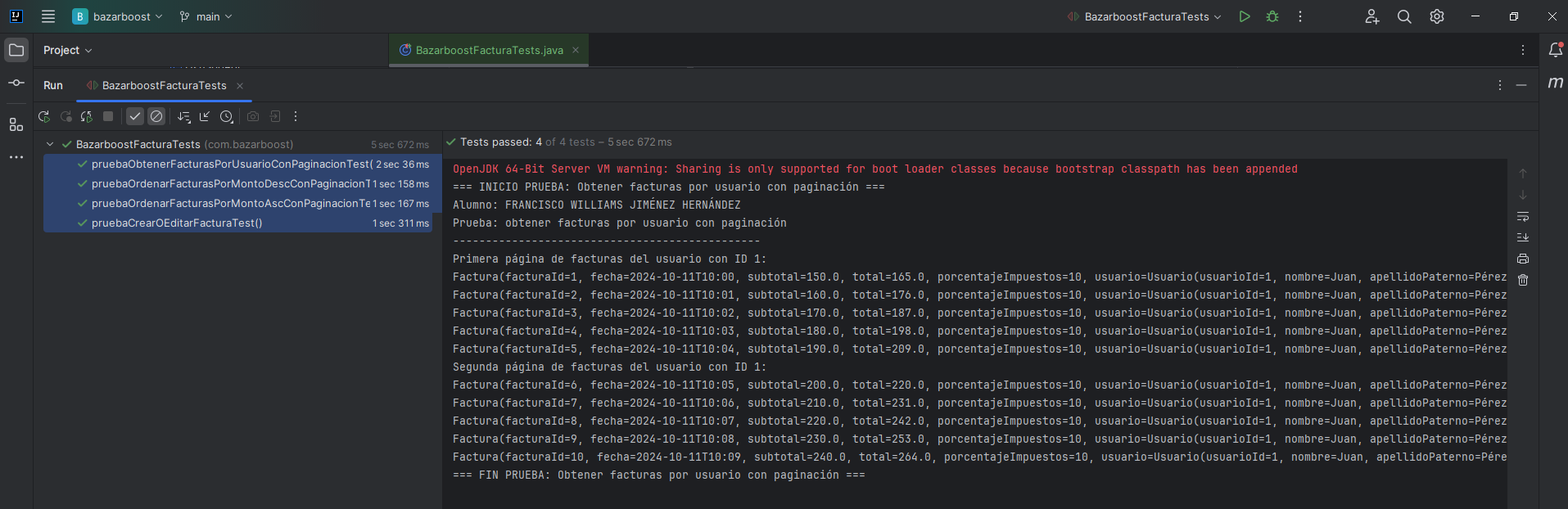
Captura 16. Clase de pruebas para *Descuento* y su salida de ejecución.

**Archivo: BazarboostDireccionTests.java y su salida de ejecución**



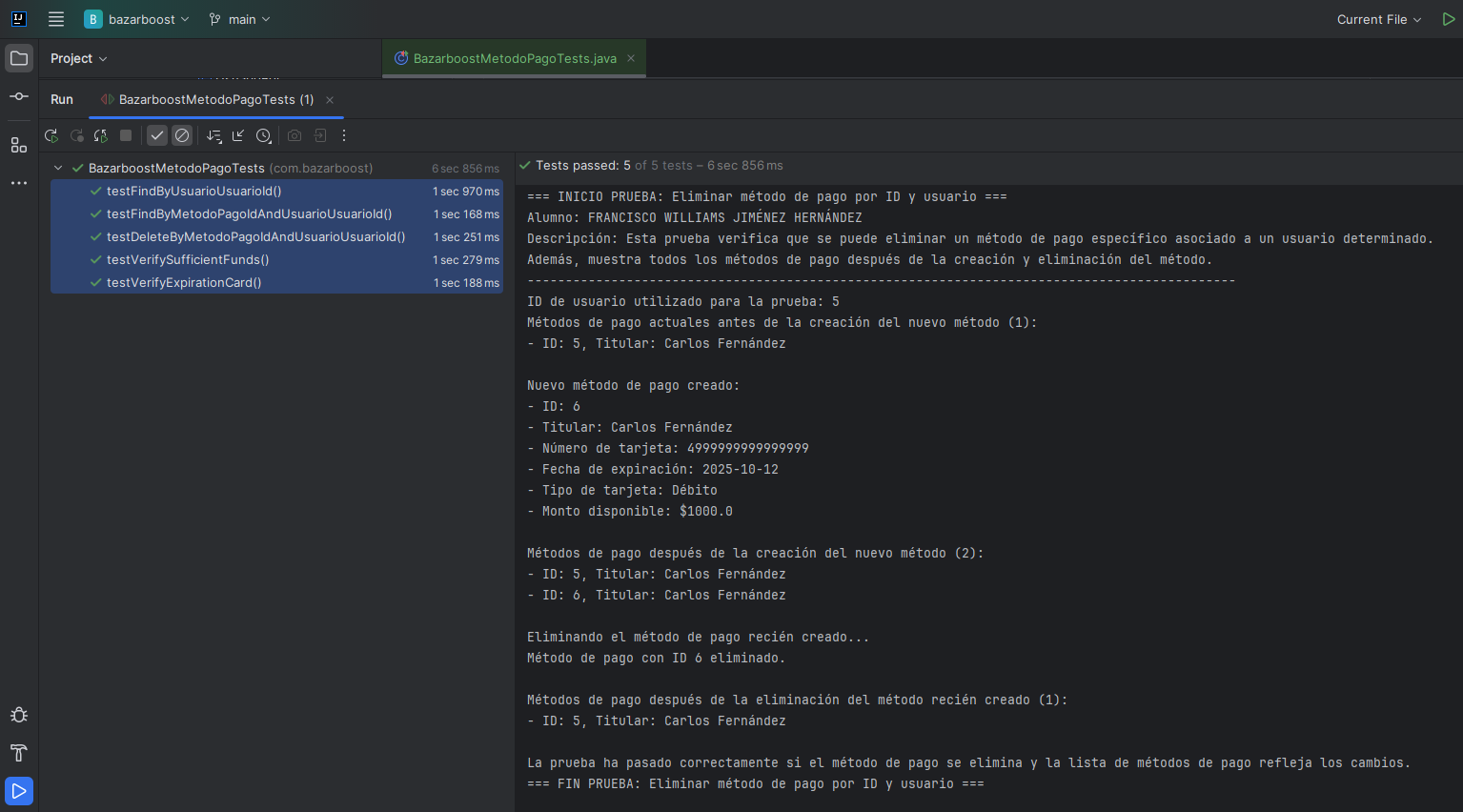
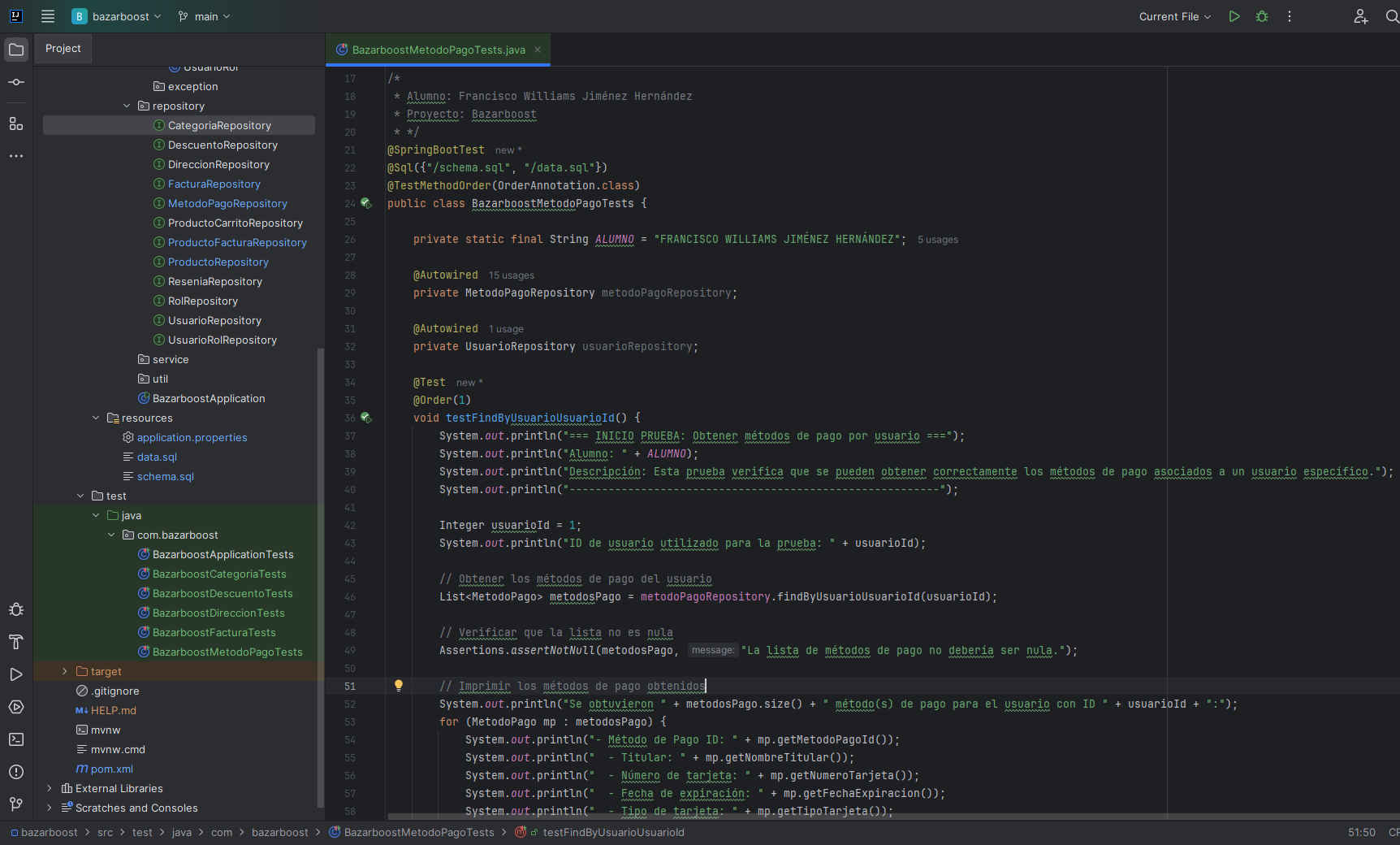
Captura 17. Clase de pruebas para *Direccion* y su salida de ejecución.

**Archivo: BazarboostFacturaTests.java y su salida de ejecución**



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

**Archivo: BazarboostMetodoPagoTests.java y su salida de ejecución**



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