

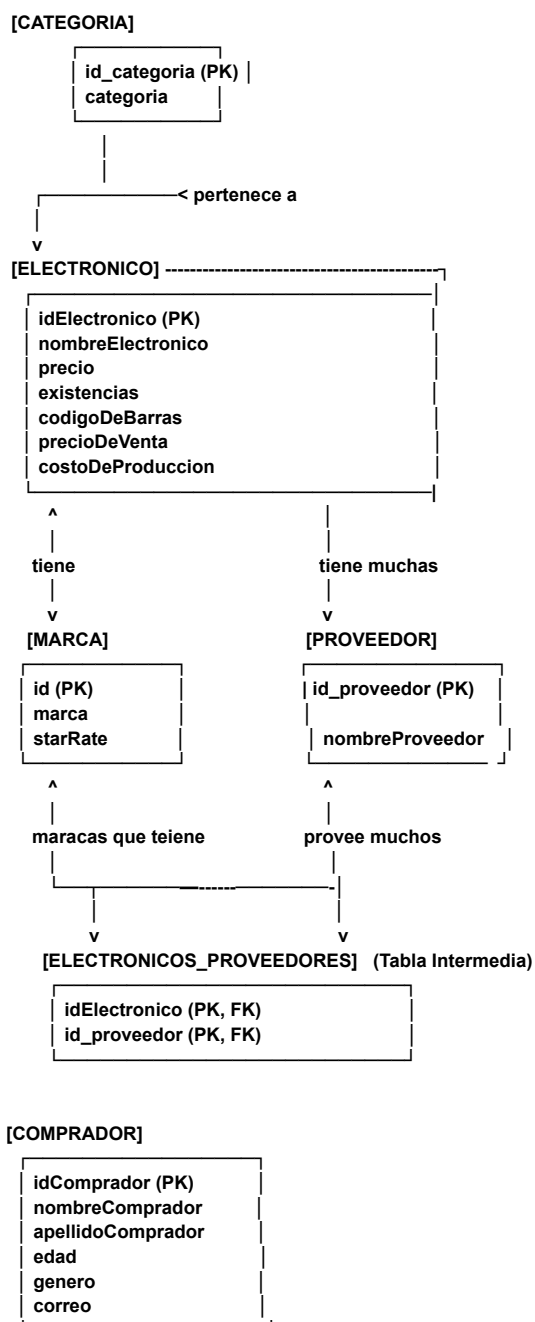
Desarrollo de Sistemas con Tecnología

Java Emisión 15

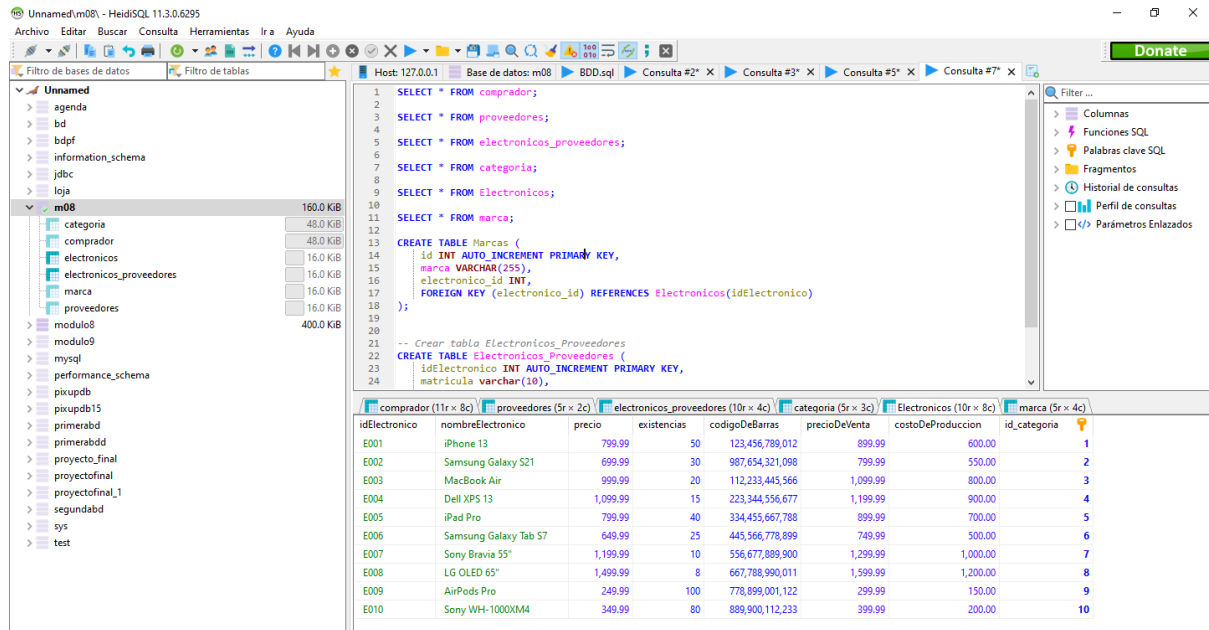
Avance Proyecto Final :

Nombre: Noyola Nazario Alejandro

1. Tener al menos cinco tablas para el proyecto:
 - i. Tablas de catálogo
 - ii. Tablas de relación
 - iii. Incluir en el pdf un Diagrama Entidad Relacion



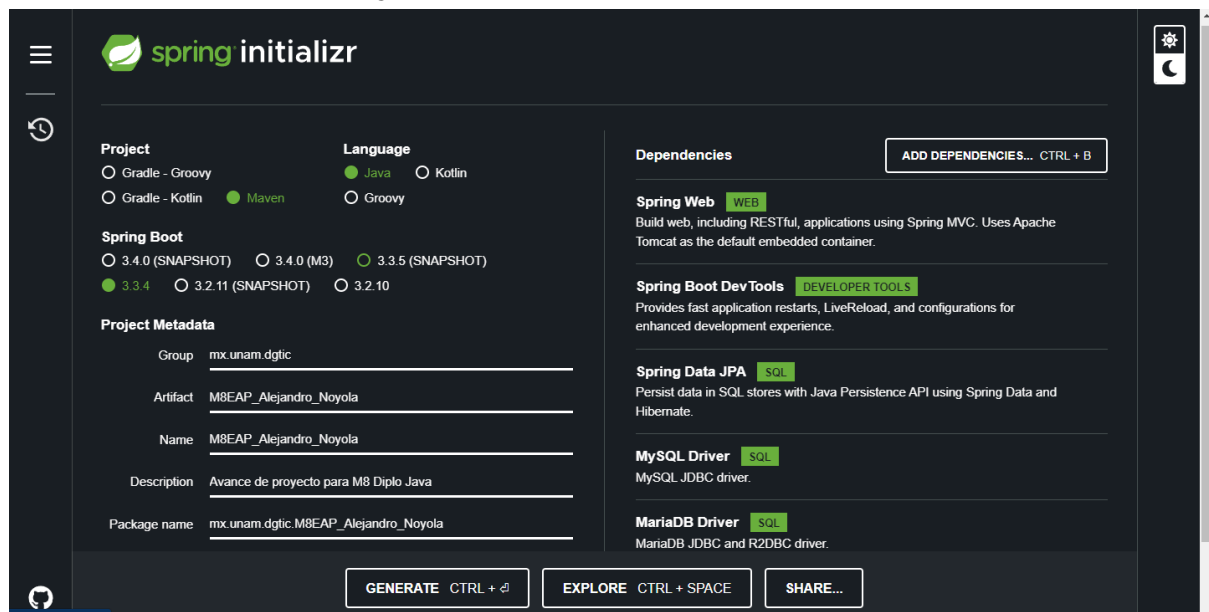
iv. Incluir scripts sql para crear el esquema y el estado de la bd



2. Se debe crear un proyecto Spring nuevo con las siguientes características:

- Nombre del proyecto:
- M8AP<PrimerNombre>_<PrimerApellido>
- Ejemplo: M8AP_Omar_Mendoza

3. Con soporte JPA y MariaDb



4. Debe implementar las clases e interfaces relacionadas con

- CrudRepository
- PagingAndSortingRepository

```
3 import org.springframework.data.domain.Page;
4 import org.springframework.data.domain.Pageable;
5 import org.springframework.data.jpa.repository.Query;
6 import org.springframework.data.repository.PagingAndSortingRepository;
7 import org.springframework.data.repository.query.Param;
8
9 import java.time.LocalDate;
10
11
12 public interface CompradorPagingAndSortingRepository extends PagingAndSortingRepository<Comprador, Integer> {
13
14     // Consulta Derivada Paginada
15     Page<Comprador> findByNombreCompradorLike(String patron, Pageable pagina); 1 usage
16
17     //Page<Comprador> findByApellidoCompradorNotLike(String patron, Pageable pagina);
18
19     /*
20      * Page<Comprador> findByGenero(String genero, Pageable pagina);
21
22      * Page<Comprador> findByEdadGreaterThan(Integer edad, Pageable pagina);
23
24      * Page<Comprador> findByNombreCompradorIgnoreCase(String nombre, Pageable pagina);
25
26      */
27
28     // Consulta nombrada paginada
29     @Query("SELECT DISTINCT c.nombreComprador FROM Comprador c") no usages
30     Page<String> findDistinctNombreComprador(Pageable pageable);
31
32     @Query("SELECT DISTINCT c.apellidoComprador FROM Comprador c") no usages
33     Page<String> findDistinctApellidoComprador(Pageable pageable);
34
35     @Query(value = "SELECT c.genero AS campo, COUNT(*) AS conteo " + 2 usages
36           "FROM Comprador c " +
```

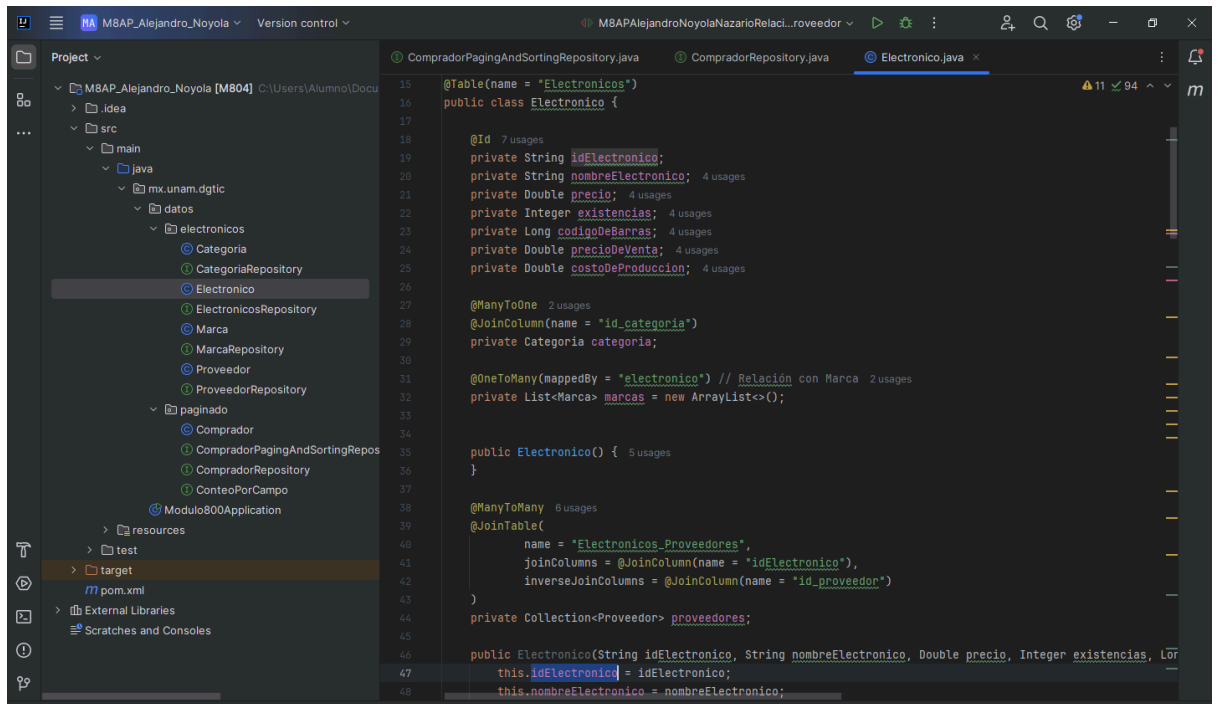
```
1 package mx.unam.dgtic.datos.paginado;
2
3 import mx.unam.dgtic.datos.paginado.Comprador;
4 import org.springframework.data.repository.CrudRepository;
5
6 public interface CompradorRepository extends CrudRepository<Comprador, Integer> { no usages
7
8 }
```

5. Implementar relaciones entre las tablas

i. ManyToOne

ii. OneToMany

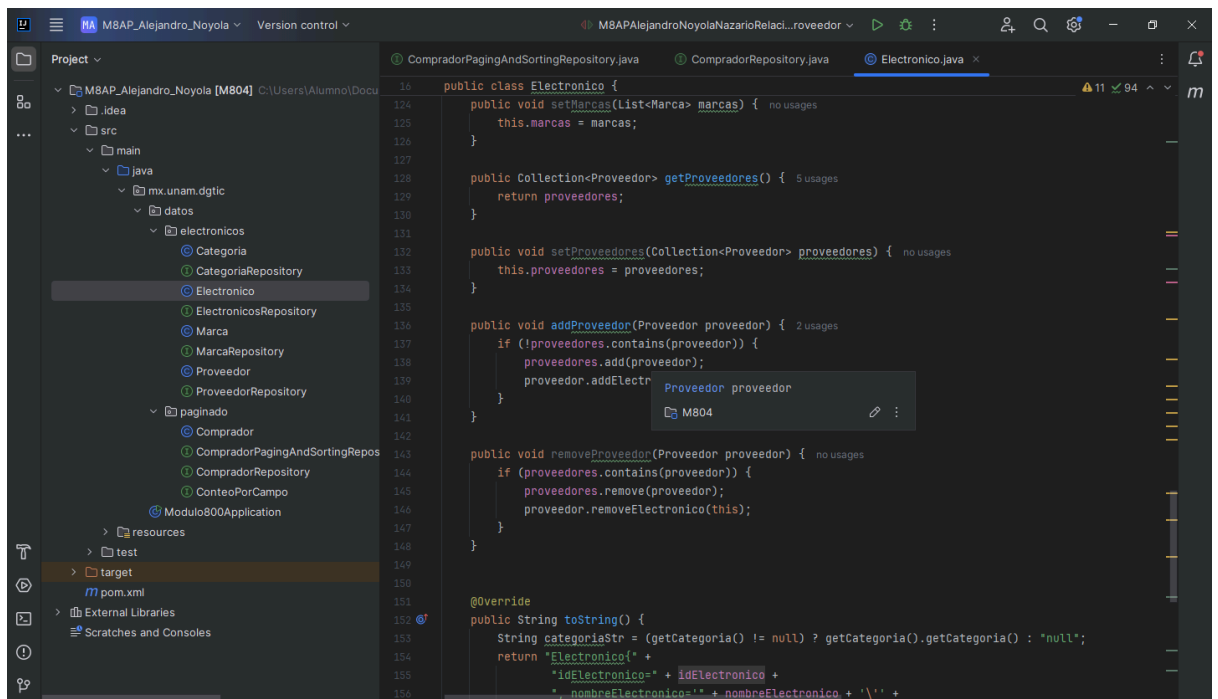
iii. ManyToMany



The screenshot shows the IntelliJ IDEA interface with the 'Electronico.java' file open. The left sidebar displays the project structure, including the 'datos' package and the 'Electronico' entity. The main editor shows the following code:

```
15 @Table(name = "Electronicos")
16 public class Electronico {
17
18     @Id 7 usages
19     private String idElectronico;
20     private String nombreElectronico; 4 usages
21     private Double precio; 4 usages
22     private Integer existencias; 4 usages
23     private Long codigoDeBarras; 4 usages
24     private Double precioDeVenta; 4 usages
25     private Double costoDeProduccion; 4 usages
26
27     @ManyToOne 2 usages
28     @JoinColumn(name = "id_categoria")
29     private Categoria categoria;
30
31     @OneToMany(mappedBy = "electronico") // Relación con Marca 2 usages
32     private List<Marca> marcas = new ArrayList<>();
33
34     public Electronico() { 5 usages
35     }
36
37     @ManyToMany 6 usages
38     @JoinTable(
39         name = "Electronicos_Proveedores",
40         joinColumns = @JoinColumn(name = "idElectronico"),
41         inverseJoinColumns = @JoinColumn(name = "id_proveedor")
42     )
43     private Collection<Proveedor> proveedores;
44
45     public Electronico(String idElectronico, String nombreElectronico, Double precio, Integer existencias, Long
46         this.idElectronico = idElectronico;
47         this.nombreElectronico = nombreElectronico;
```

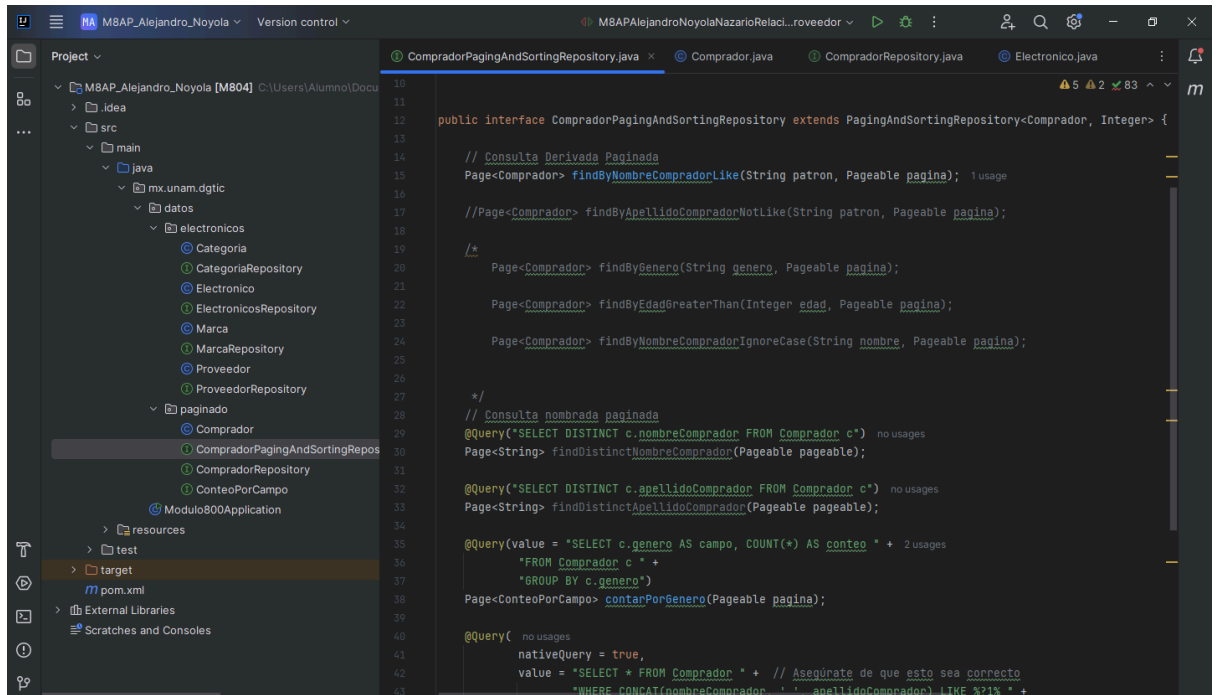
6. Implementar operaciones CRUD para diferentes tablas de catalogo y de relación



The screenshot shows the IntelliJ IDEA interface with the 'Electronico.java' file open, displaying the implementation of CRUD operations. The left sidebar shows the project structure. The main editor shows the following code:

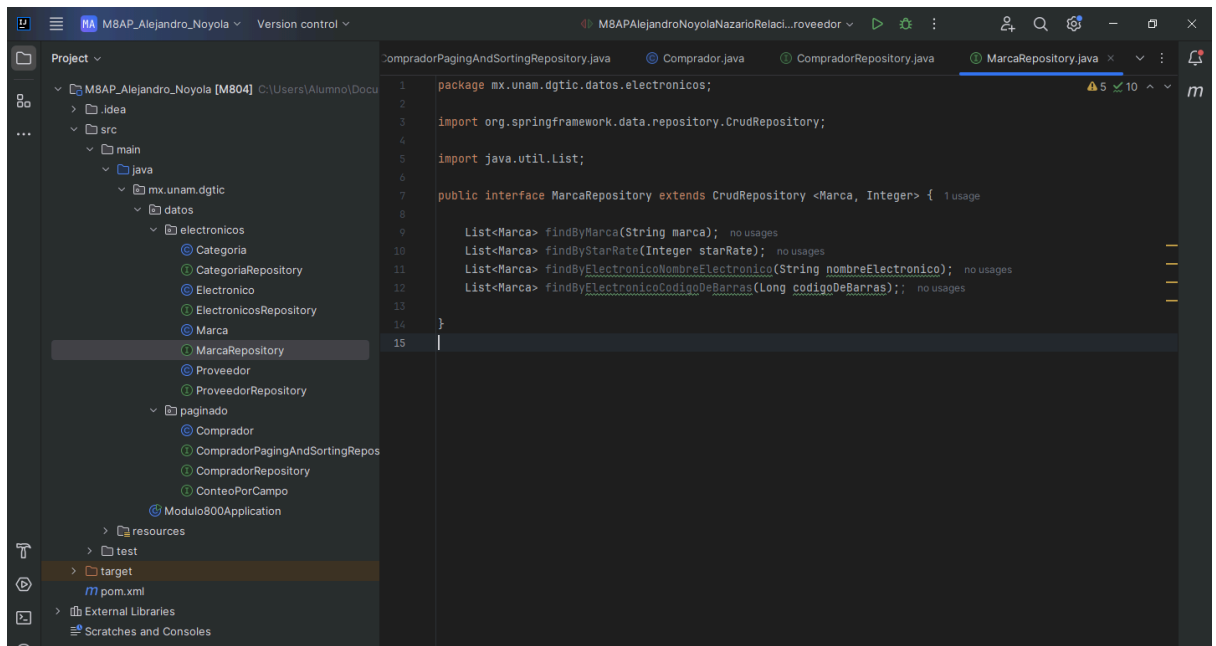
```
124 public void setMarcas(List<Marca> marcas) { no usages
125     this.marcas = marcas;
126 }
127
128 public Collection<Proveedor> getProveedores() { 5 usages
129     return proveedores;
130 }
131
132 public void setProveedores(Collection<Proveedor> proveedores) { no usages
133     this.proveedores = proveedores;
134 }
135
136 public void addProveedor(Proveedor proveedor) { 2 usages
137     if (!proveedores.contains(proveedor)) {
138         proveedores.add(proveedor);
139         proveedor.addElectr
140     }
141 }
142
143 public void removeProveedor(Proveedor proveedor) { no usages
144     if (proveedores.contains(proveedor)) {
145         proveedores.remove(proveedor);
146         proveedor.removeElectronico(this);
147     }
148 }
149
150 @Override
151 public String toString() {
152     String categoriaStr = (getCategoria() != null) ? getCategoria().getCategoria() : "null";
153     return "Electronico{" +
154         "idElectronico=" + idElectronico +
155         ", nombreElectronico=" + nombreElectronico + '\n' +
```

7. Implementar metodos de CrudRepository y PagingAndSortingRepository



The screenshot shows the IntelliJ IDEA IDE with the project 'M8AP_Alejandro_Noyola' open. The file explorer on the left shows the project structure, with the 'CompradorPagingAndSortingRepository.java' file selected. The main editor displays the code for this interface, which extends 'PagingAndSortingRepository<Comprador, Integer>'. The code includes several methods for finding and counting records, with some methods using JPA @Query annotations for complex SQL queries.

```
18 public interface CompradorPagingAndSortingRepository extends PagingAndSortingRepository<Comprador, Integer> {
19
20     // Consulta Derivada Paginada
21     Page<Comprador> findByNombreCompradorLike(String patron, Pageable pagina); 1 usage
22
23     //Page<Comprador> findByApellidoCompradorNotLike(String patron, Pageable pagina);
24
25     /*
26     Page<Comprador> findByGenero(String genero, Pageable pagina);
27
28     Page<Comprador> findByEdadGreaterThan(Integer edad, Pageable pagina);
29
30     Page<Comprador> findByNombreCompradorIgnoreCase(String nombre, Pageable pagina);
31
32     */
33     // Consulta nombrada paginada
34     @Query("SELECT DISTINCT c.nombreComprador FROM Comprador c") no usages
35     Page<String> findDistinctNombreComprador(Pageable pageable);
36
37     @Query("SELECT DISTINCT c.apellidoComprador FROM Comprador c") no usages
38     Page<String> findDistinctApellidoComprador(Pageable pageable);
39
40     @Query(value = "SELECT c.genero AS campo, COUNT(*) AS conteo " + 2 usages
41           "FROM Comprador c " +
42           "GROUP BY c.genero")
43     Page<ConteoPorCampo> contarPorGenero(Pageable pagina);
44
45     @Query( no usages
46           nativeQuery = true,
47           value = "SELECT * FROM Comprador " + // Asegurate de que esto sea correcto
48           "WHERE CONCAT(nombreComprador, ' ', apellidoComprador) LIKE %?%" +
```



The screenshot shows the IntelliJ IDEA IDE with the project 'M8AP_Alejandro_Noyola' open. The file explorer on the left shows the project structure, with the 'MarcaRepository.java' file selected. The main editor displays the code for this interface, which extends 'CrudRepository<Marca, Integer>'. The code includes several methods for finding records, with some methods using JPA @Query annotations for complex SQL queries.

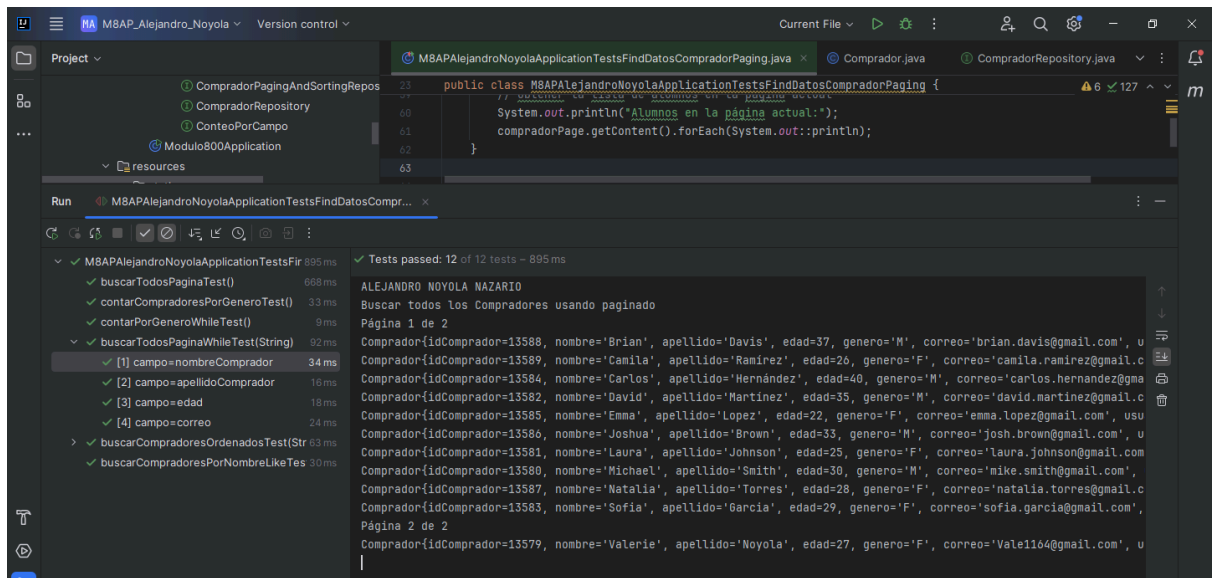
```
1 package mx.unam.dgtic.datos.electronicos;
2
3 import org.springframework.data.repository.CrudRepository;
4
5 import java.util.List;
6
7 public interface MarcaRepository extends CrudRepository <Marca, Integer> { 1 usage
8
9     List<Marca> findByMarca(String marca); no usages
10    List<Marca> findByStarRate(Integer starRate); no usages
11    List<Marca> findByElectronicoNombreElectronico(String nombreElectronico); no usages
12    List<Marca> findByElectronicoCodigoDeBarras(Long codigoDeBarras); no usages
13
14 }
15
```

8. Implementar consultas sobre las tablas

- Consultas derivadas (al menos 5)
- Consultas nombradas (al menos 5)
- Consultas con relaciones (al menos 5)

9. Probar las implentaciones en pruebas unitarias organizadas por tema (diferentes clases de prueba para cada tema que incluyen varias pruebas

las pruebas de nombradas y derivadas se encuentran en el test
M8APAlejandroNoyolaApplicationTestsFindDatosCompradorPaging



las pruebas relacionales se encuentran en el test
M8APAlejandroNoyolaNazarioRelacional[Tema]ElectronicoTest

