**Aplicación Vigía de Precios**

1. Primer paso configuramos el proyecto Spring Boot en Visual Basic Code.
2. Configuramos el archivo application.properties con los siguientes datos de conexión temporal, antes de conectarse con Herocu.

spring.datasource.url=jdbc:mysql://localhost/vigia\_precio?serverTimezone=America/Santiago&useSSL=false

spring.datasource.username=appvigia

spring.datasource.password=appvigia

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

logging.level.org.hibernate.SQL=debug

spring.jpa.show-sql=true

1. Dentro de la carpeta principal, creamos un folder llamado entities, aquí vamos a programa nuestro ORM; dentro de esta carpeta creamos la primera clase llamada Usuario.java.

package com.grupo27\_1.appvigia.entities;

//importamos las herramientas de javax.persistence

import javax.persistence.\*;

//Creamos las anotaciones claves

@Entity

@Table(schema = "appvigia", name = "usuarios")

public class Usuario {

// Creamos los constructores uno con los argumentos y otro vacio

public Usuario(String nombre, String usuario, String clave, String correo, String prefijo, String celular) {

this.nombre = nombre;

this.setUsuario(usuario);

this.setClave(clave);

this.correo = correo;

this.prefijo = prefijo;

this.celular = celular;

}

public Usuario() {

}

// crear el identificador

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long id;

@Column(length = 40, name = "nombre", nullable = false, unique = true)

private String nombre;

@Column(length = 15, name = "usuario", nullable = false, unique = false)

private String usuario;

@Column(length = 15, name = "clave", nullable = false, unique = true)

private String clave;

@Column(length = 20, name = "correo", nullable = false, unique = false)

private String correo;

@Column(length = 2, name = "prefijo", nullable = false, unique = false)

private String prefijo;

@Column(length = 10, name = "movil", nullable = false, unique = true)

private String celular;

public long getId() {

return id;

}

public void setId(long id) {

this.id = id;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public String getUsuario() {

return usuario;

}

public void setUsuario(String usuario) {

this.usuario = usuario;

}

public String getClave() {

return clave;

}

public void setClave(String clave) {

this.clave = clave;

}

public String getCorreo() {

return correo;

}

public void setCorreo(String correo) {

this.correo = correo;

}

public String getPrefijo() {

return prefijo;

}

public void setPrefijo(String prefijo) {

this.prefijo = prefijo;

}

public String getCelular() {

return celular;

}

public void setCelular(String celular) {

this.celular = celular;

}

// Creamos el toString

@Override

public String toString() {

return "Usuario [celular=" + celular + ", clave=" + clave + ", correo=" + correo + ", id=" + id + ", nombre="

+ nombre + ", prefijo=" + prefijo + ", usuario=" + usuario + "]";

}

// HashCode nos permite asignar un unico espacio en memoria

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + (int) (id ^ (id >>> 32));

return result;

}

// Equal nos permite ver si dos objetos de la misma clase son iguales

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

Usuario other = (Usuario) obj;

if (id != other.id)

return false;

return true;

}

}

Ya en nuestro proyecto, el Backen este compuesto por 3 secciones:

* Servidor Web
* Rest Api
* ORM

En este proceso, comenzaremos haciendo las pruebas unitarias.

1. Creamos las demás entidades, que para el caso de nuestra base de datos son 4 (usuarios, productos, tiendas, precios).
2. Posteriormente, creamos los repositorios para cada entidad, a continuación, un ejemplo de los repositorios.

package com.grupo27\_1.appvigia.repositories;

import org.springframework.data.jpa.repository.JpaRepository;

import com.grupo27\_1.appvigia.entities.Precio;

public interface PrecioRepository extends JpaRepository<Precio, Float> {

}

1. Una vez creadas las entidades y los repositorios, procedemos a realizar el primer test; por lo cual configuramos y/o modificamos el archivo inicial que se creo al crear el proyecto y ejecutando el archivo, los resultados fueron satisfactorios, utilizando el repositorio UsuarioRepository.java

Imagen cuando inician la prueba:

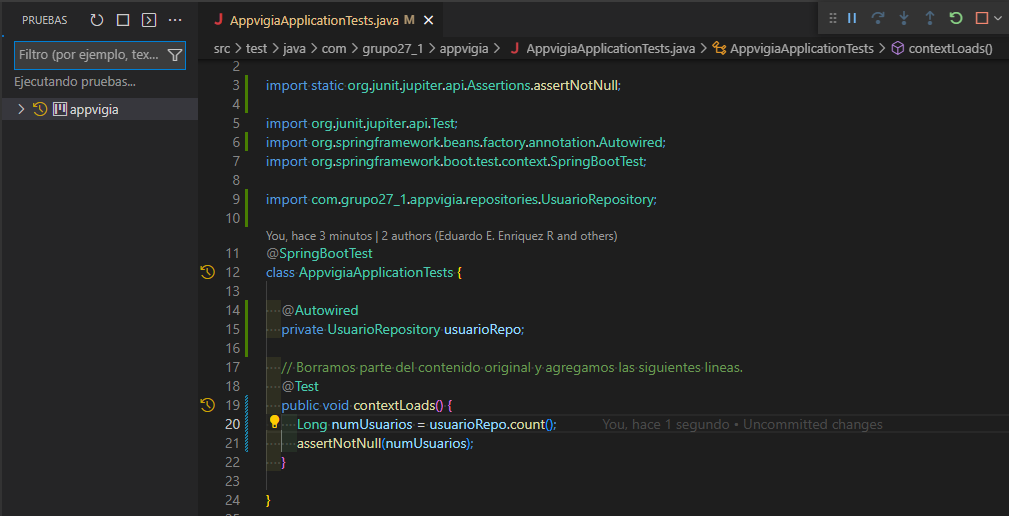
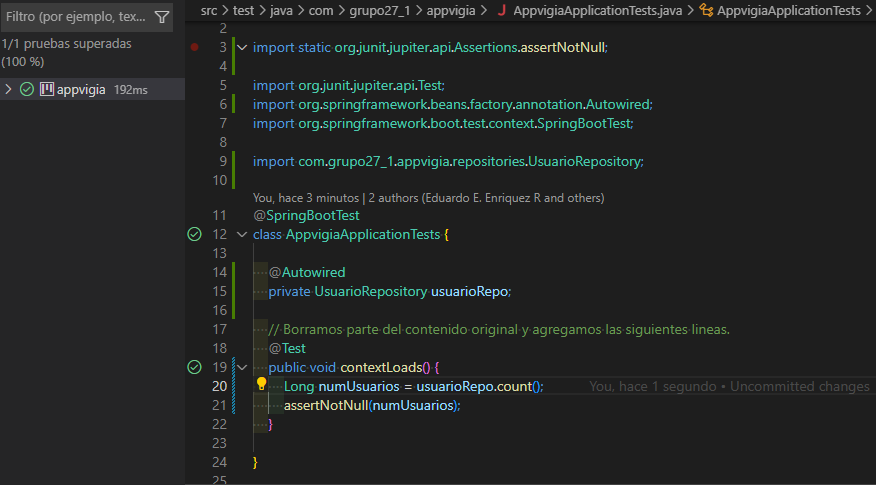
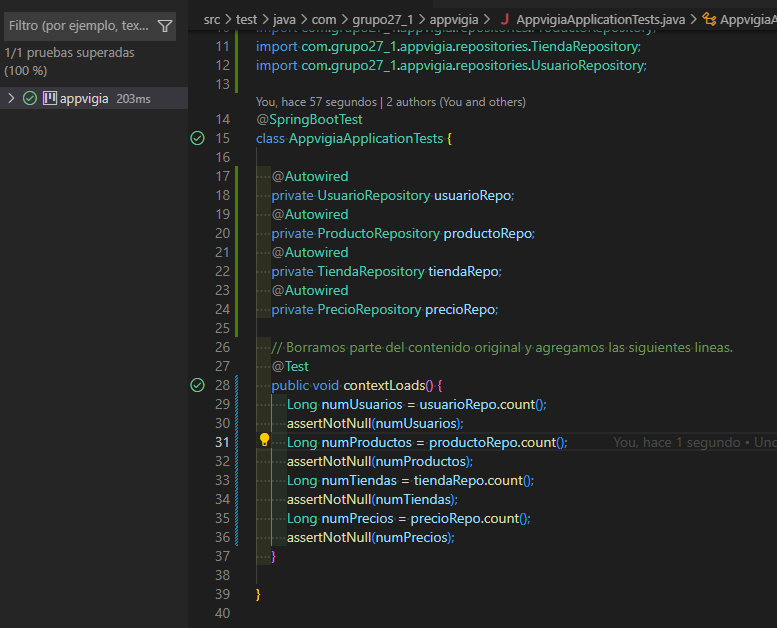


Imagen después de realizado la prueba:



La siguiente imagen es el resultado de las pruebas de todos los repositorios elaborados en el proyecto hasta este momento:



1. Creamos la carpeta ems y dentro de ella una clase llamada “UserEntityManager.java”, y con ella nos ayudamos para realizar otra prueba, seguidamente creamos un archivo de test llamado “UsuarioCrudTest.java”, y se prueba.

package com.grupo27\_1.appvigia;

import static org.junit.jupiter.api.Assertions.\*;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase;

import org.springframework.boot.test.autoconfigure.jdbc.AutoConfigureTestDatabase.Replace;

import org.springframework.boot.test.autoconfigure.orm.jpa.DataJpaTest;

import com.grupo27\_1.appvigia.entities.Usuario;

import com.grupo27\_1.appvigia.repositories.UsuarioRepository;

//Con las siguientes instrucciones estamos indicando que es un test tipo dato JPA

@DataJpaTest

@AutoConfigureTestDatabase(replace = Replace.NONE)

public class UsuarioCrudTest {

@Autowired

private UsuarioRepository usrRepo;

@Test

public void createUser() {

Usuario usr = new Usuario("nombre", "usuario", "clave", "correo", "prefijo", "celular");

Usuario result;

try {

result = usrRepo.save(usr);

assertNotNull(result.getId());

assertTrue(result.getId() > 0);

} catch (Exception e) {

System.err.println(e.getMessage());

}

}

}

Resultado:

