



Por: Ramiro Israel Vivanco Gualán

Paralelo: B

Fecha: 12-9-2020

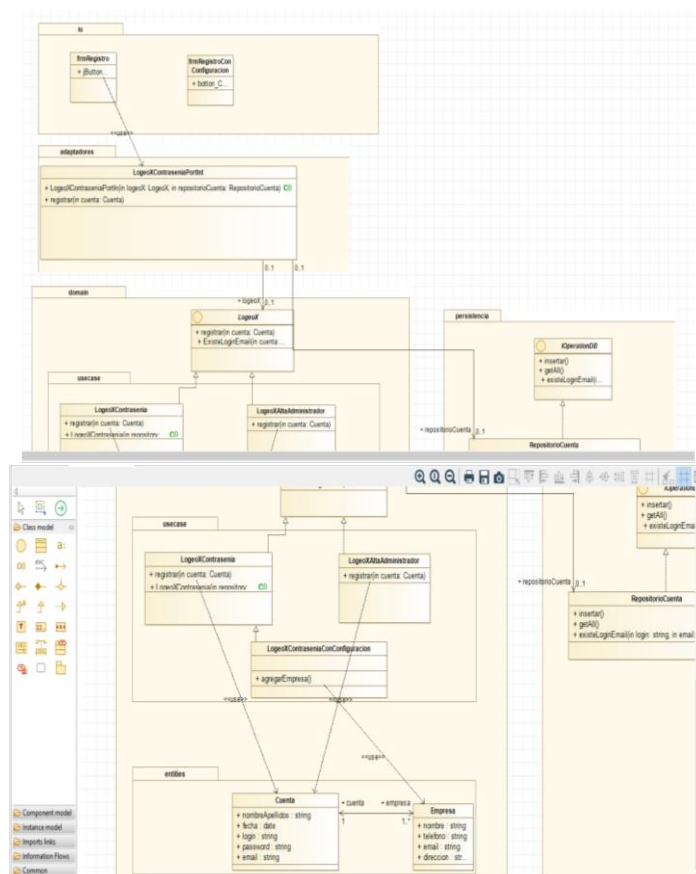
Presentación

Es presente documento describe las actividades que debe realizar para el examen del primer bimestre, de la materia de Arquitectura de aplicaciones

Desarrollo

Continuando con el desarrollo del registro de cuentas de los estacionamientos, se desea implementar la configuración de la empresa, que se describe a continuación:

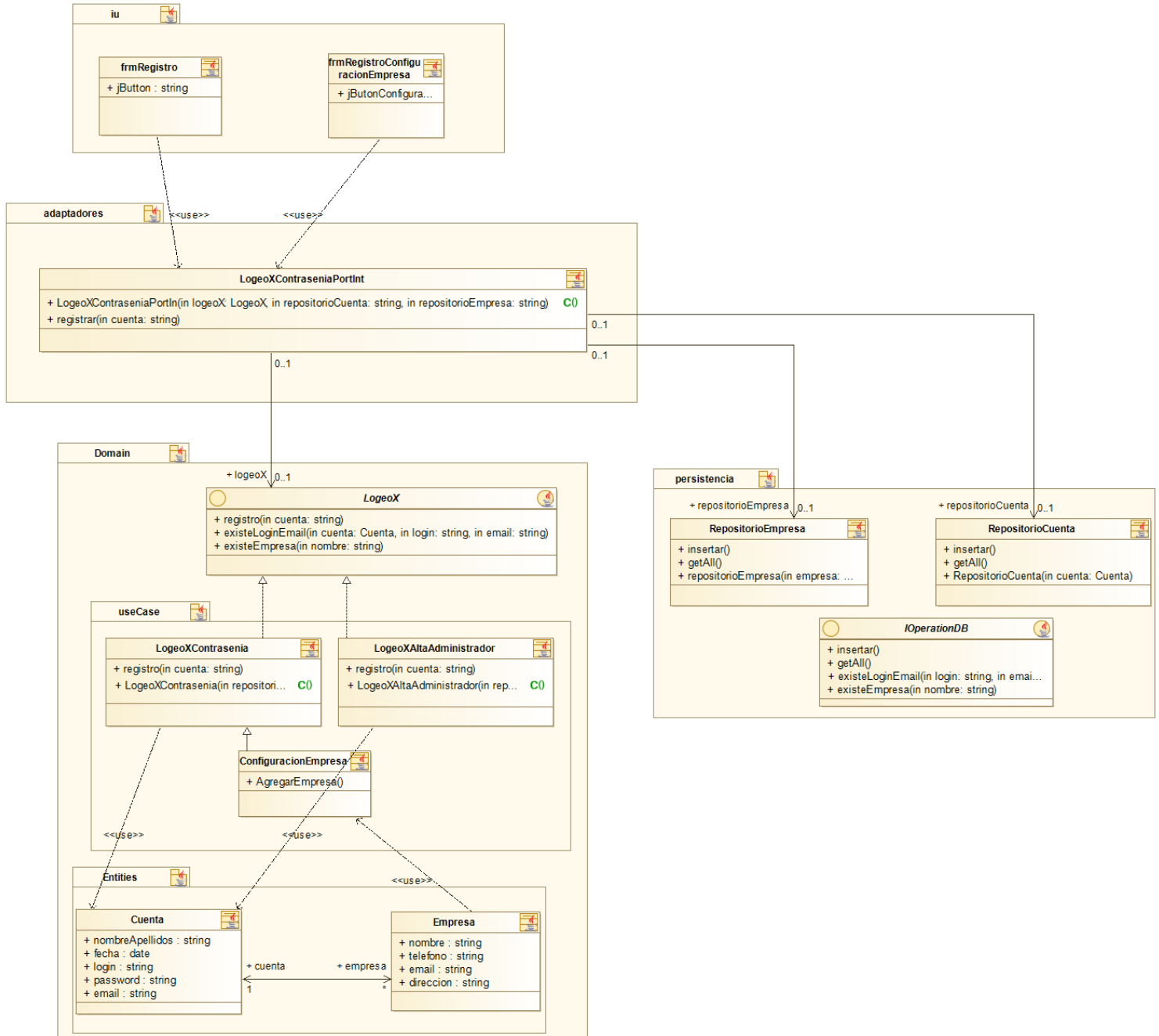
Cuando se registre una cuenta por cualquier mecanismo (Con contraseñas, alta de administrador, validación por correo), se debe registrar también los datos de empresa (nombre, telefono, email, direccion); y existe una relación entre cuenta y empresa de 1 a muchos. Debe considerar que la funcionalidad no debería entorpecer las funcionalidades ya desarrolladas. Además, se debería crear un frontal y el mecanismo de persistencia respectivo. El equipo de análisis y diseño han desarrollado los cambios en el modelo, que servirán de guía para esta tarea.





Desarrollar:

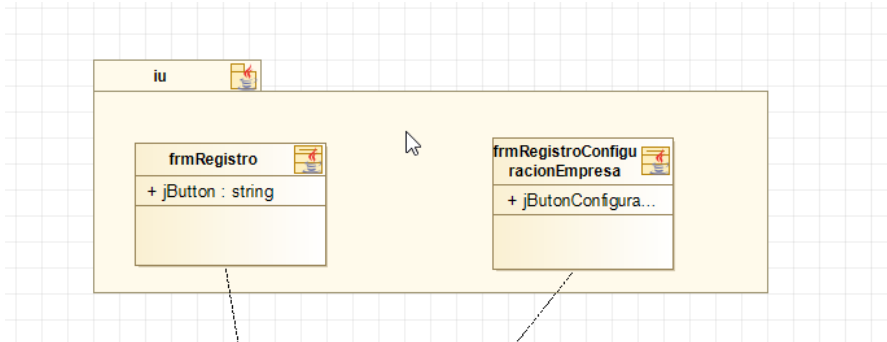
1. Correcciones en el modelo de ser necesario, e indicar si se hizo, entregado una captura del nuevo modelo





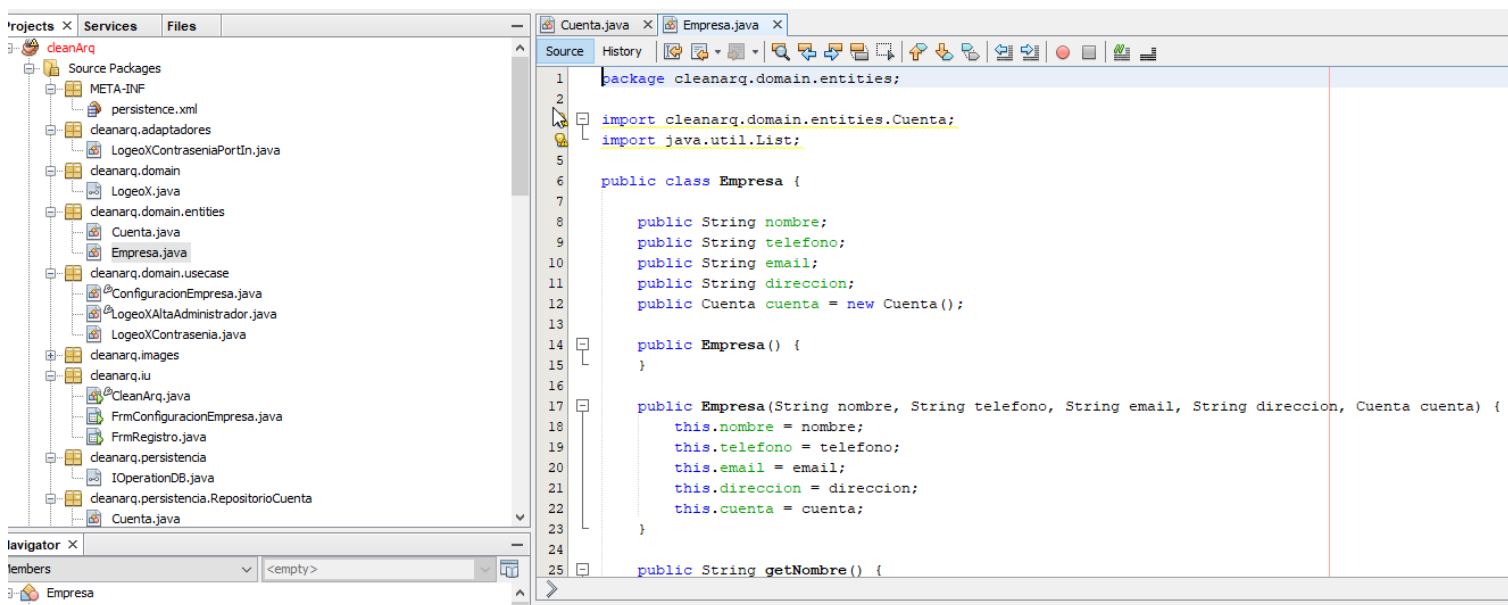
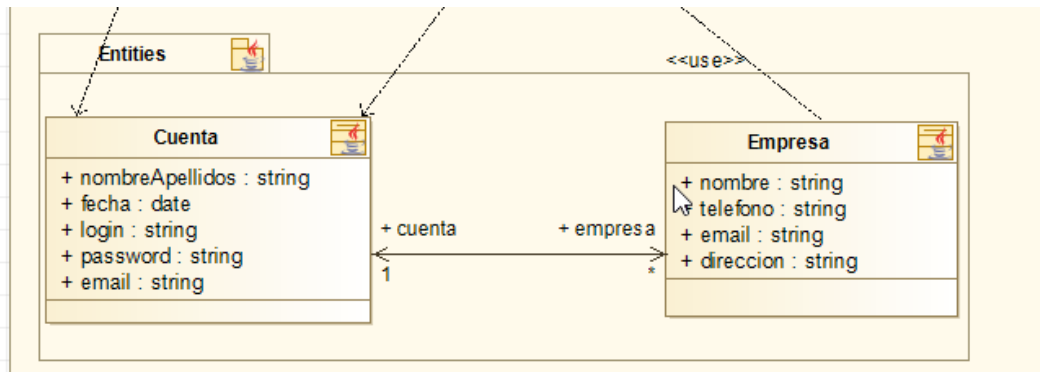
2. Implementar las clases indicadas en los modelos que incluyan:
 - 2.1. Logica del negocio e integración de los componentes de la arquitectura

Implementación de interfaces

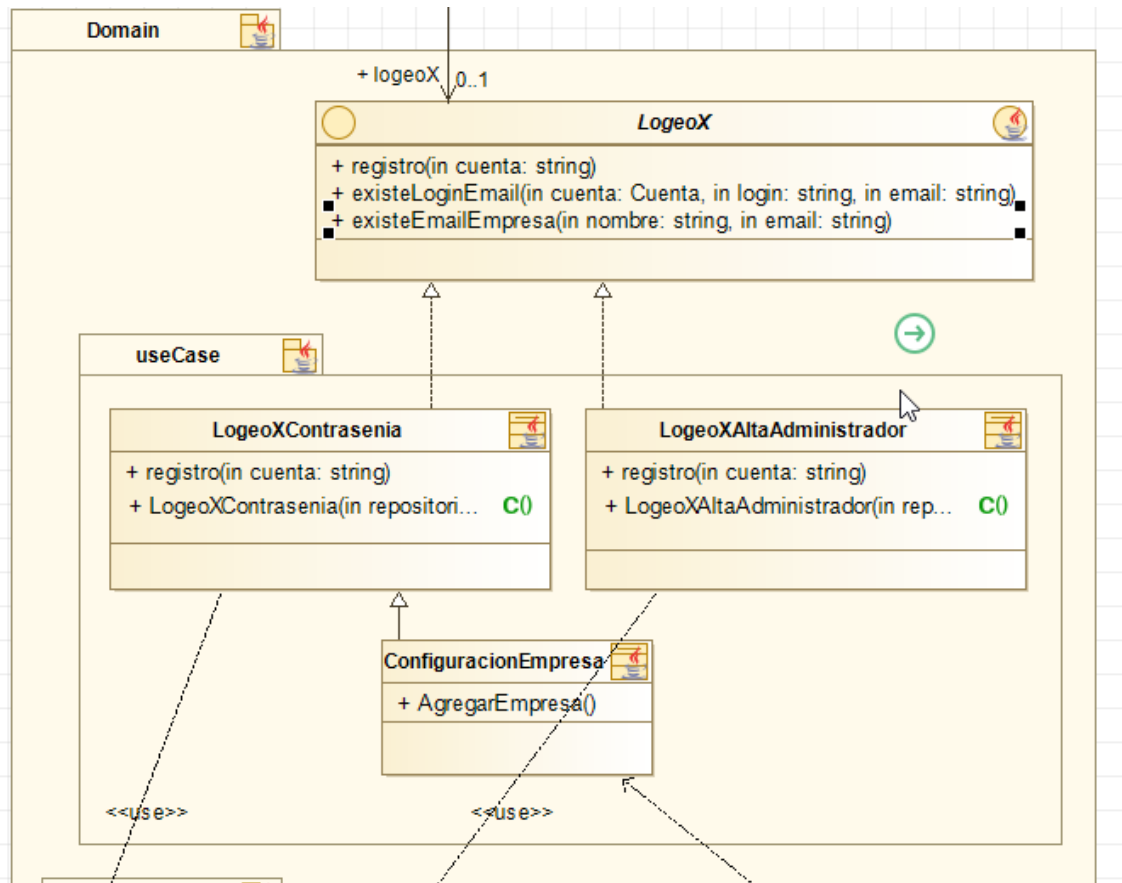




Entidades Empresa



Agregación de empresa



objects X Services Files

cleanArq

- Source Packages
 - META-INF
 - persistence.xml
 - cleanarq.adaptadores
 - LogeoXContraseniaPortIn.java
 - cleanarq.domain
 - LogeoX.java
 - cleanarq.domain.entities
 - Cuenta.java
 - Empresa.java
 - cleanarq.domain.usecase
 - ConfiguracionEmpresa.java
 - LogeoXAltaAdministrador.java
 - LogeoXContrasenia.java
 - cleanarq.images
 - cleanarq.iu
 - CleanArq.java
 - FrmConfiguracionEmpresa.java
 - FrmRegistro.java
 - cleanarq.persistencia
 - IOperacionDB.java
 - cleanarq.persistencia.RepositorioCuenta
 - Cuenta.java

Source History

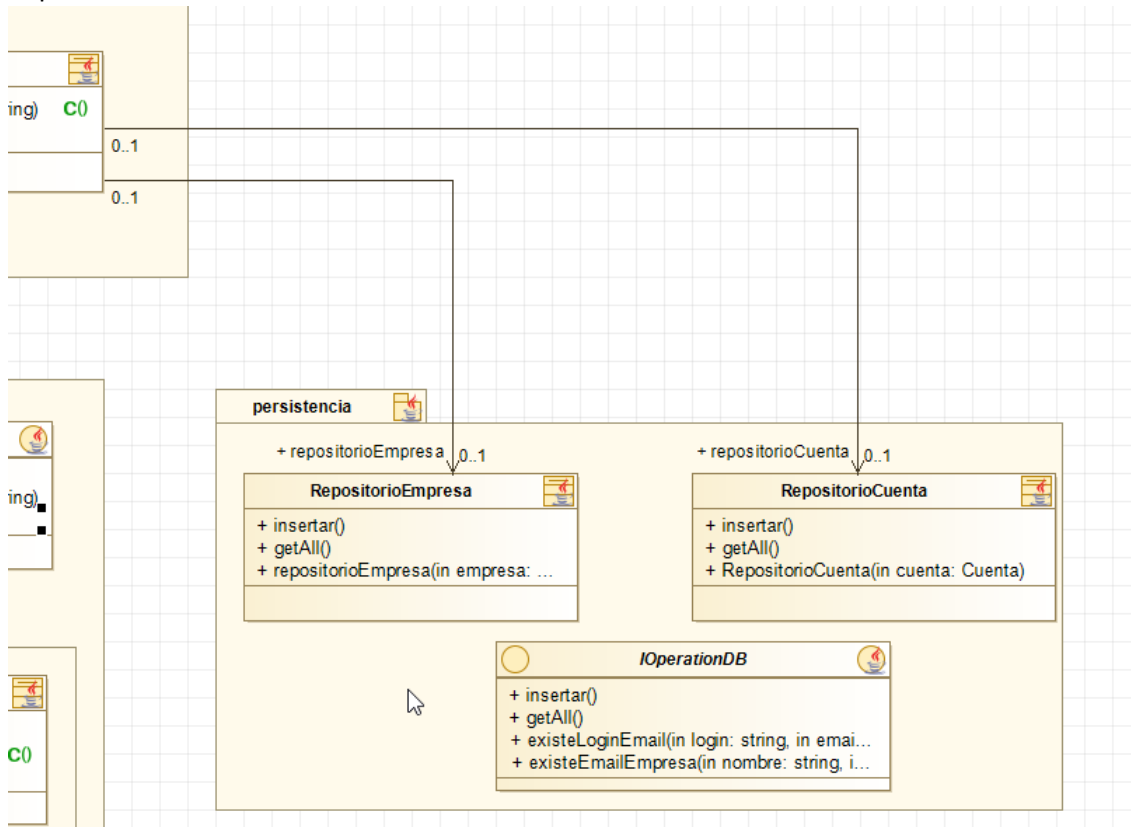
```
35 objCuenrepo.setNombreaellido(cuenta.getNombreaellido())
36 objCuenrepo.setPassword(cuenta.getPassword());
37 objCuenrepo.setId(1);
38 repository.InsertarCuenta(objCuenrepo);
39 return 1;
40 }
41 return -1;
42 }
43
44 @Override
45 public boolean ExisteLoginEmail( String login, String email) {
46     return repository.existeLoginEmail(login, email);
47 }
48
49 @Override
50 public boolean ExisteEmailEmpresa(String nombre, String email) {
51     return repository.existeLoginEmail(nombre, email);
52 }
53 }
54
55 }
56
57 }
```

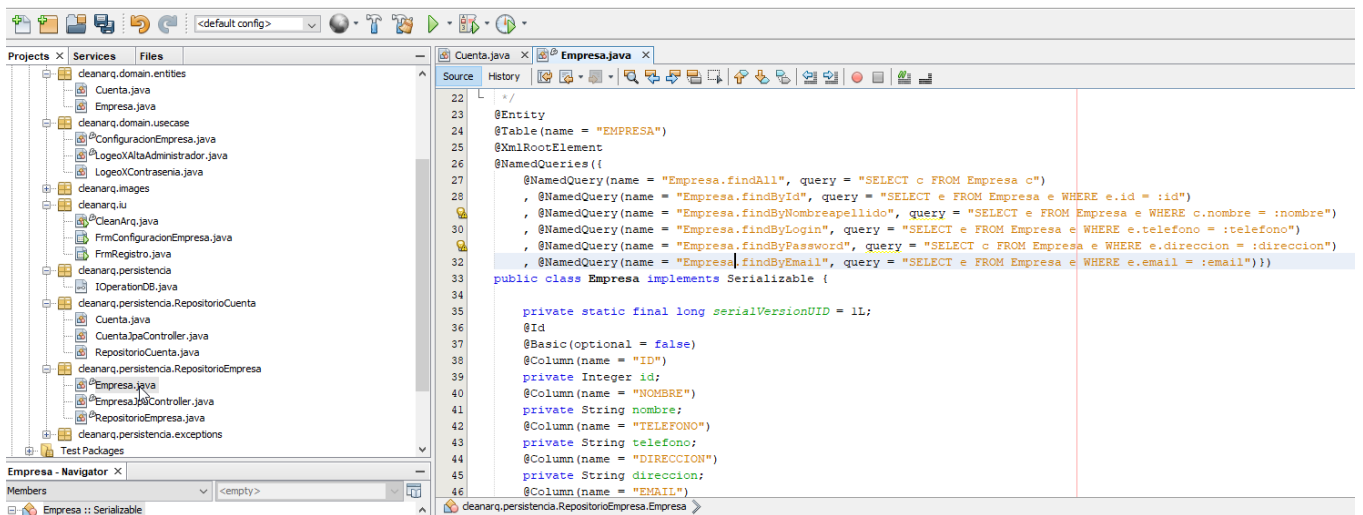


The screenshot shows an IDE with the project structure on the left and the source code of `ConfiguracionEmpresa.java` on the right. The project structure includes packages like `cleanArq`, `cleanArq.domain`, `cleanArq.domain.entities`, `cleanArq.domain.usecase`, `cleanArq.images`, `cleanArq.persistence`, and `cleanArq.persistence.RepositorioCuenta`. The source code of `ConfiguracionEmpresa.java` is as follows:

```
1 package cleanArq.domain.usecase;
2
3 import cleanArq.persistence.IOperationDB;
4
5
6 public class ConfiguracionEmpresa extends LogeoXContrasenia {
7
8     public ConfiguracionEmpresa(IOperationDB respository) {
9         super(respository);
10    }
11
12    public void AgregarEmpresa() {
13    }
14
15 }
16
```

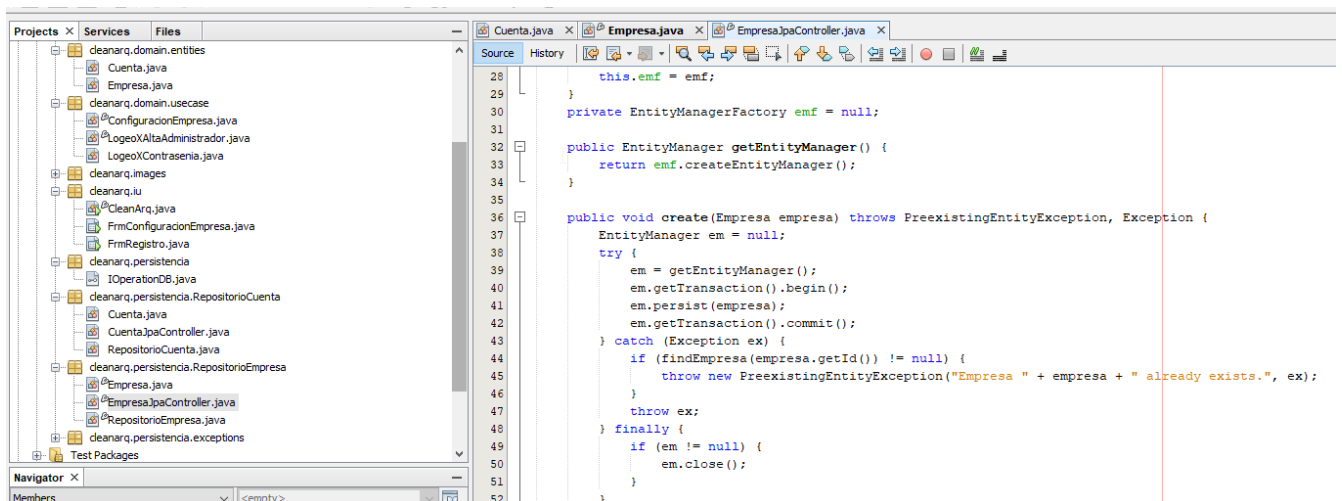
Repositorios





The screenshot shows an IDE with the file explorer on the left displaying a project structure. The main editor window shows the source code of `Empresa.java`. The code defines an `EntityManager` interface with methods `findAll`, `findById`, `findByNombreApellido`, `findByLogin`, `findByPassword`, and `findByEmail`. It also defines a `Empresa` class that implements `Serializable` and has attributes `id`, `nombre`, `telefono`, `direccion`, and `email`.

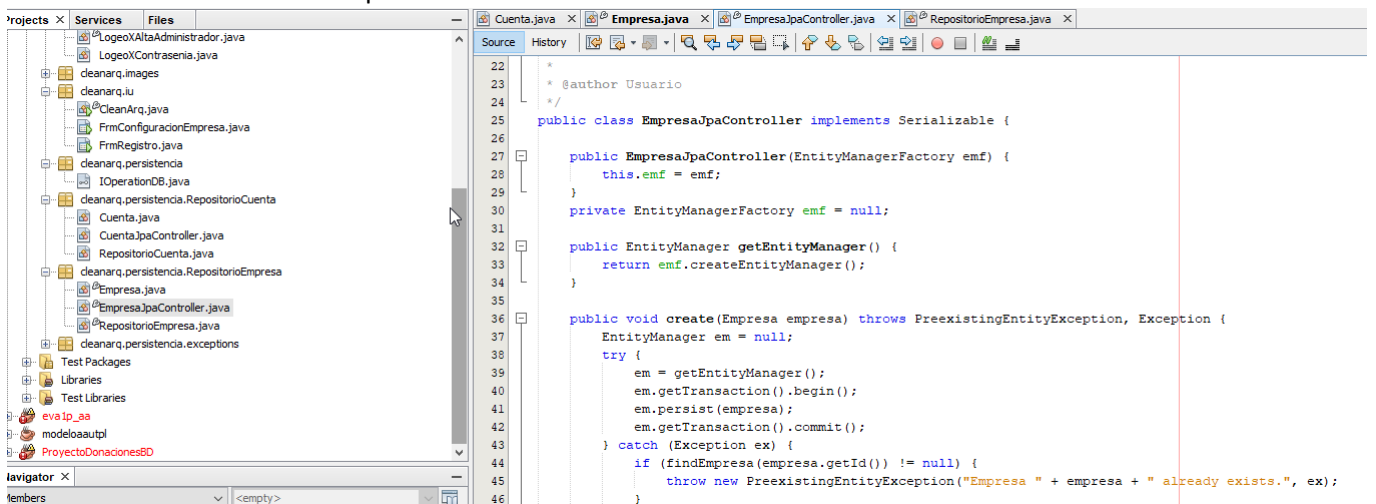
```
22  /*
23  */
24  @Entity
25  @Table(name = "EMPRESA")
26  @XmlRootElement
27  @NamedQueries({
28      @NamedQuery(name = "Empresa.findAll", query = "SELECT c FROM Empresa c")
29      , @NamedQuery(name = "Empresa.findById", query = "SELECT e FROM Empresa e WHERE e.id = :id")
30      , @NamedQuery(name = "Empresa.findByNombreApellido", query = "SELECT e FROM Empresa e WHERE e.nombre = :nombre")
31      , @NamedQuery(name = "Empresa.findByLogin", query = "SELECT e FROM Empresa e WHERE e.telefono = :telefono")
32      , @NamedQuery(name = "Empresa.findByPassword", query = "SELECT c FROM Empresa e WHERE e.direccion = :direccion")
33      , @NamedQuery(name = "Empresa.findByEmail", query = "SELECT e FROM Empresa e WHERE e.email = :email")})
34
35  public class Empresa implements Serializable {
36
37      private static final long serialVersionUID = 1L;
38      @Id
39      @Basic(optional = false)
40      @Column(name = "ID")
41      private Integer id;
42      @Column(name = "NOMBRE")
43      private String nombre;
44      @Column(name = "TELEFONO")
45      private String telefono;
46      @Column(name = "DIRECCION")
47      private String direccion;
48      @Column(name = "EMAIL")
49      private String email;
```



The screenshot shows the same IDE with the file explorer on the left. The main editor window shows the source code of `EmpresaJpaController.java`. The code implements the `EntityManager` interface methods using JPA. It includes methods `findAll`, `findById`, `findByNombreApellido`, `findByLogin`, `findByPassword`, and `findByEmail`. It also includes a `create` method that throws `PreexistingEntityException` if the entity already exists.

```
28      this.emf = emf;
29  }
30  private EntityManagerFactory emf = null;
31
32  public EntityManager getEntityManager() {
33      return emf.createEntityManager();
34  }
35
36  public void create(Empresa empresa) throws PreexistingEntityException, Exception {
37      EntityManager em = null;
38      try {
39          em = getEntityManager();
40          em.getTransaction().begin();
41          em.persist(empresa);
42          em.getTransaction().commit();
43      } catch (Exception ex) {
44          if (findEmpresa(empresa.getId()) != null) {
45              throw new PreexistingEntityException("Empresa " + empresa + " already exists.", ex);
46          }
47          throw ex;
48      } finally {
49          if (em != null) {
50              em.close();
51          }
52      }
```

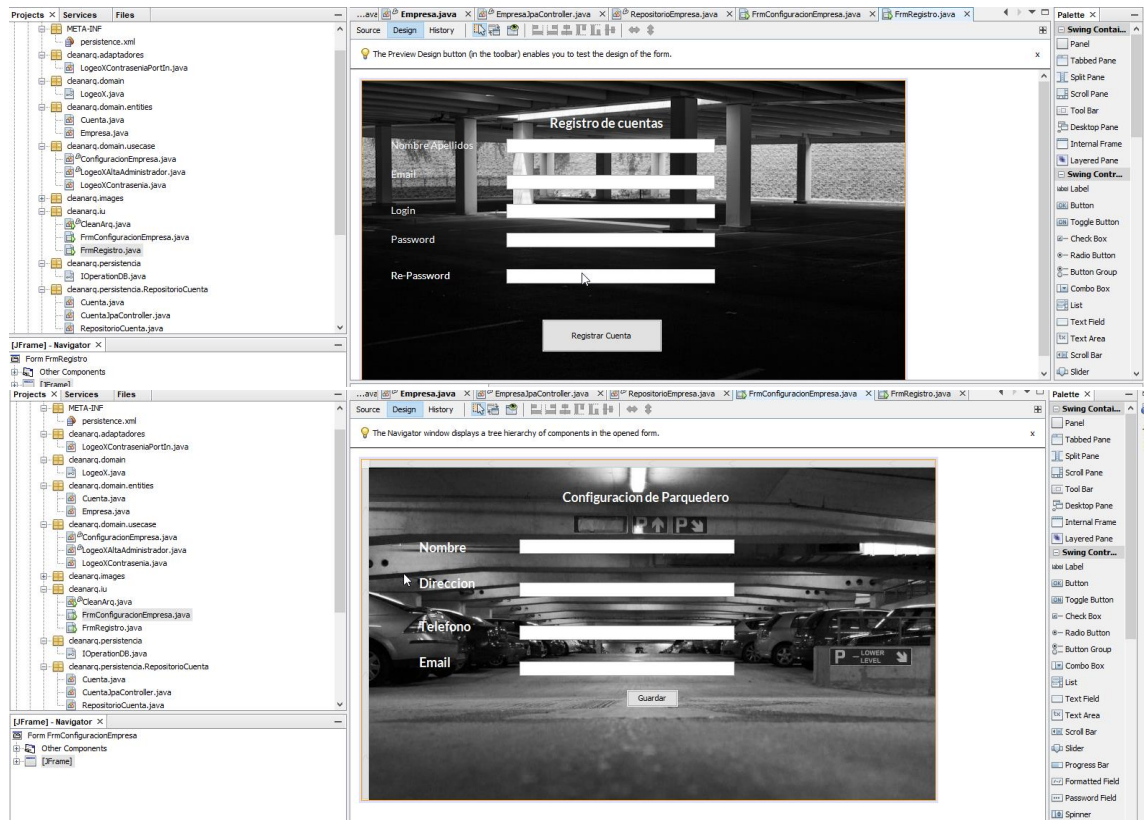
2.2. mecanismo de persistencia



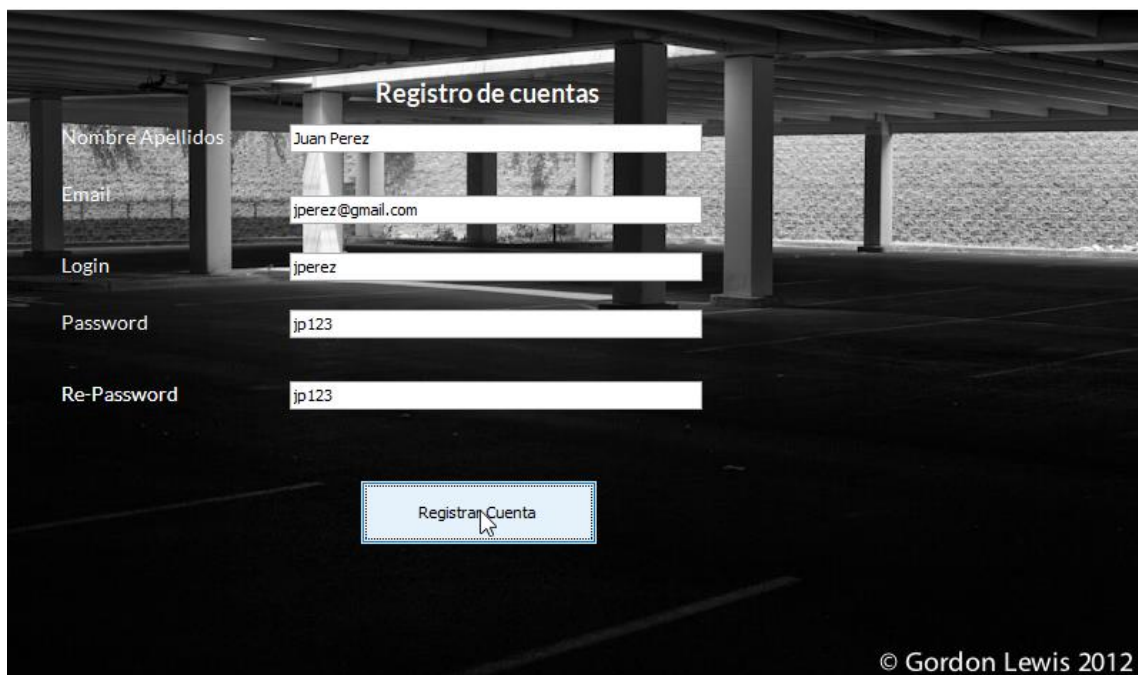
The screenshot shows the same IDE with the file explorer on the left. The main editor window shows the source code of `EmpresaJpaController.java`. The code implements the `EntityManager` interface methods using JPA. It includes methods `findAll`, `findById`, `findByNombreApellido`, `findByLogin`, `findByPassword`, and `findByEmail`. It also includes a `create` method that throws `PreexistingEntityException` if the entity already exists.

```
22  *
23  * @author Usuario
24  */
25  public class EmpresaJpaController implements Serializable {
26
27      public EmpresaJpaController(EntityManagerFactory emf) {
28          this.emf = emf;
29      }
30      private EntityManagerFactory emf = null;
31
32      public EntityManager getEntityManager() {
33          return emf.createEntityManager();
34      }
35
36      public void create(Empresa empresa) throws PreexistingEntityException, Exception {
37          EntityManager em = null;
38          try {
39              em = getEntityManager();
40              em.getTransaction().begin();
41              em.persist(empresa);
42              em.getTransaction().commit();
43          } catch (Exception ex) {
44              if (findEmpresa(empresa.getId()) != null) {
45                  throw new PreexistingEntityException("Empresa " + empresa + " already exists.", ex);
46              }
```

2.3. frontal



3. Capturas de pantalla de la ejecución del proyecto.





UTPL
UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

Universidad Técnica Particular de Loja
Ciencia de la Computación.
Arquitectura de Aplicaciones

Configuracion de Parquedero

Nombre

Direccion

Telefono

Email

The background of the form is a grayscale photograph of a multi-level parking garage. Several cars are parked on different levels. A sign with a 'P' and an arrow pointing down is visible on the right side, with the text 'LOWER LEVEL' below it.