

**UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS**

**CARRERA DE INGENIERIA DE SISTEMAS**

**PROYECTO “BASTANTEO DE PODERES BANCARIOS”**

**CURSO IS148 PROGRAMACION ORIENTADA A OBJECTOS**

**INTEGRANTES**

* Jose Vásquez, U201121210.
* Cesar Soriano, U201121295.

**PROFESOR**

* Jose Díaz.

**INDICE**

**INTRODUCCION** **3**

**CASO PROPUESTO** **3**

**HISTORIAS DE USUARIO** **3**

**PRUEBAS UNITARIAS** **5**

**ARQUITECTURA Y CODIGO FUENTE** **5**

**DIAGRAMA UML** **5**

BIBLIOGRAFIA 5

1. **INTRODUCCION**

PeruBank debe asegurar que sólo los representantes que cada cliente jurídico designe legalmente puedan efectuar las operaciones bancarias especificadas. Para este efecto, ha contratado una notaría que se encargará de realizar el proceso de bastanteo y poner a disposición de la red de agencias los resultados para la verificación operativa.

1. **CASO PROPUESTO**
2. **HISTORIAS DE USUARIO**
3. **Registrar clientes**.- Como supervisor deseo registrar clientes de modo que los abogados puedan bastantear sus poderes bancarios.

|  |  |
| --- | --- |
| Cuando:   * Registro un cliente debo indicar su RUC, su código de cliente interno (8 dígitos), razón social, fecha de inicio de operaciones, tipo de empresa (SAA, SAC, SRL, etc.) y cantidad de empleados. | Espero:   * Confirmación del registro OK. * Mensaje de error si no se ingresan los datos requeridos. * Mensaje de error si el cliente ya existe (RUC o código de cliente repetido). |

1. **Inscribir representantes**.- Como abogado deseo inscribir los representantes de cada cliente de modo que luego puedan tener los poderes bastanteados.

|  |  |
| --- | --- |
| Cuando:   * Inscribo un representante debo indicar su tipo y número de documento, sus nombres y apellidos, cargo en la empresa y un grupo de bastanteo (A, B, C, etc.). | Espero:   * Confirmación de la inscripción Ok. * Mensaje de error si no se ingresan los datos requeridos. * Mensaje de error si ya es representante de la empresa (según tipo y número de documento). |

1. **Agregar poderes**.- Como supervisor deseo agregar el listado de poderes para que los abogados puedan asignarlos a los representantes según bastanteo.

|  |  |
| --- | --- |
| Cuando:   * Agrego un poder debo indicar su código corto de consulta (ej.: CHCO, EFRE), el nombre (ej.: Cobro de Cheques, Retiro en Efectivo) y el tipo producto/servicio (ej.: Activo, Pasivo). | Espero:   * Confirmación de la agregación Ok. * Mensaje de error si no se ingresan los datos requeridos. * Mensaje de error si ya se agregó el poder (código corto). |

1. **Bastantear poderes**.- Como abogado deseo poder anotar el resultado del bastanteo de modo que el personal de agencia pueda consultarlo antes de cada operación.

|  |  |
| --- | --- |
| Cuando:   * Bastanteo debo indicar qué poderes puede realizar cada grupo de bastanteo de cada cliente. Además, para esta asignación debo indicar el tipo de intervención (ej.: A Sola Firma, En Conjunto Con) y con qué otro(s) grupo(s) de bastanteo si corresponde (combinación), así como el importe y la fecha de vencimiento (normalmente 6 meses). | Espero:   * Mensaje de confirmación Ok. * Mensaje de error si ya se anotó el mismo poder para el mismo grupo de bastanteo, en el mismo cliente, considerando el mismo tipo de intervención y combinación. |

1. **Verificar poderes**.- Como personal de agencia deseo poder consultar si un representante tiene poder para una operación o no de manera que pueda realizarla.

|  |  |
| --- | --- |
| Cuando:   * Consulto debo indicar el RUC del cliente, la identificación del representante, el código de poder (o elegirlo de una lista), y el importe. | Espero:   * Mensaje indicando que el representante tiene poder y cómo. * Mensaje indicando que el poder está dado pero ya venció. * Mensaje indicando que el representante no tiene poder para realizar la operación indicada o el importe es superior. |

1. **Consultar poderes**.- Como cliente deseo poder elegir y reservar una sala de ensayo de manera que pueda ir con mi banda y hacer uso de las facilidades seleccionadas.

|  |  |
| --- | --- |
| Cuando:   * Consulto debo indicar el RUC del cliente, la identificación del representante, el grupo de bastanteo o tipo de intervención. | Espero:   * Lista de registros de bastanteo que se tienen según los criterios de búsqueda ingresados. * Mensaje indicando que no hay resultados según criterios. |

1. **Controlar usuarios**.- Como administrador deseo poder controlar a los usuarios de modo que al ingresar puedan acceder a sus funcionalidades ya sea como supervisor, como abogado o empleado.

|  |  |
| --- | --- |
| Cuando:   * Doy de alta un usuario debo indicar qué rol tiene asignado. Cada usuario debe identificarse por una clave al ingresar. | Espero:   * Mensaje de bienvenida indicando el rol que tiene así como las opciones que le corresponde. * Mensaje de error si no se autentica correctamente. |

1. **PRUEBAS UNITARIAS**

**//clase RolTest.java**

**package** bastanteo;

**import** **static** org.junit.Assert.*assertEquals*;

**import** org.junit.Test;

**public** **class** RolTest {

@Test

**public** **void** UsuarioDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

String codigoRol = "R001";

String descripcionRol = "Supervisor";

**boolean** accesoMenuClientes = **true**;

**boolean** accesoMenuPoderes = **true**;

**boolean** accesoMenuBastanteo = **true**;

**boolean** accesoMenuUsuarios = **true**;

//Ejecuto el escenario

Rol rol = **new** Rol(codigoRol,descripcionRol,accesoMenuClientes,accesoMenuPoderes,accesoMenuBastanteo,accesoMenuUsuarios);

//Verifico

*assertEquals*(codigoRol,rol.getCodigoRol());

*assertEquals*(descripcionRol,rol.getDescripcionRol());

*assertEquals*(accesoMenuClientes,rol.getAccesoMenuClientes());

*assertEquals*(accesoMenuPoderes,rol.getAccesoMenuPoderes());

*assertEquals*(accesoMenuBastanteo,rol.getAccesoMenuBastanteo());

*assertEquals*(accesoMenuUsuarios,rol.getAccesoMenuUsuarios());

}

}

**//clase UsuarioTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class UsuarioTest {

@Test

public void UsuarioDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

String codigoUsuario = "csoriano";

String nombre = "Cesar";

String apellido = "Soriano";

String clave = "asd";

String codigoRol = "R001";

//Ejecuto el escenario

Usuario usuario = new Usuario(codigoUsuario,nombre,apellido,clave,codigoRol);

//Verifico

assertEquals(codigoUsuario,usuario.getCodigoUsuario());

assertEquals(nombre,usuario.getNombre());

assertEquals(apellido,usuario.getApellido());

assertEquals(clave,usuario.getClave());

assertEquals(codigoRol,usuario.getCodigoRol());

}

}

**//clase RepresentanteTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class RepresentanteTest {

@Test

public void representanteDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

//Arrange

//Preparar los datos de prueba

String tipoDocumento = "DNI";

String nroDocumento = "40341631";

String nombre = "JOSE";

String apellido = "VASQUEZ BUSTAMANTE";

String cargoEnLaEmpresa = "ABOGADO";

String grupoBastanteo = "A"; //A,B,C

//Ejecuto el escenario

Representante representante = new Representante(tipoDocumento,nroDocumento,nombre,apellido,

cargoEnLaEmpresa,grupoBastanteo);

//Verifico

assertEquals(tipoDocumento,representante.getTipoDocumento());

assertEquals(nroDocumento,representante.getNroDocumento());

assertEquals(nombre,representante.getNombre());

assertEquals(apellido,representante.getApellido());

assertEquals(cargoEnLaEmpresa,representante.getCargoEnLaEmpresa());

assertEquals(grupoBastanteo,representante.getGrupoBastanteo());

}

}

**//clase PoderTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class PoderTest {

@Test

public void PoderesDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

String codigoPoder = "CHCO";

String nombrePoder = "Cobreo de Cheques";

String tipoPoder = "Activo";

//Ejecuto el escenario

Poder poder = new Poder(codigoPoder,nombrePoder,tipoPoder);

//Verifico

assertEquals(codigoPoder,poder.getCodigoPoder());

assertEquals(nombrePoder,poder.getNombrePoder());

assertEquals(tipoPoder,poder.getTipoPoder());

}

}

**//clase GrupoTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class GrupoTest {

@Test

public void GrupoDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

String grupoBastanteo = "A";

String descripcionGrupo = "Grupo A";

//Ejecuto el escenario

Grupo grupo = new Grupo(grupoBastanteo,descripcionGrupo);

//Verifico

assertEquals(grupoBastanteo,grupo.getGrupoBastanteo());

assertEquals(descripcionGrupo,grupo.getDescripcionGrupo());

}

}

**//clase ClienteTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class ClienteTest {

@Test

public void ClienteDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

//Arrange

//Preparar los datos de prueba

String ruc = "20107012011";

String codigoCliente = "CL001";

String razonSocial = "ACME Solutions";

String fechaInicio = "01/01/2011";

String tipoEmpresa = "SAC";

int empleados = 10;

//Ejecuto el escenario

Cliente cliente = new Cliente(codigoCliente,ruc,razonSocial,fechaInicio,tipoEmpresa,empleados);

//Verifico

assertEquals(codigoCliente,cliente.getCodigoCliente());

assertEquals(ruc,cliente.getRuc());

assertEquals(razonSocial,cliente.getRazonSocial());

assertEquals(fechaInicio,cliente.getFechaInicio());

assertEquals(tipoEmpresa,cliente.getTipoEmpresa());

assertEquals(empleados,cliente.getEmpleados());

}

}

**//clase ClienteRepresentanteTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class ClienteRepresentanteTest {

@Test

public void SiIngresoClienteRepresentante() throws MensajeException{

//Preparamos Escenario

String codigoCliente = "CL001";

String ruc = "20107012011";

String razonSocial = "ACME Solutions";

String fechaInicio = "01/01/2011";

String tipoEmpresa = "SAC";

int empleados = 10;

//Ejecuto el escenario

Cliente cliente = new Cliente(codigoCliente,ruc,razonSocial,fechaInicio,tipoEmpresa,empleados);

String tipoDocumento = "DNI";

String nroDocumento = "40341631";

String nombre = "JOSE";

String apellido = "VASQUEZ BUSTAMANTE";

String cargoEnLaEmpresa = "ABOGADO";

String grupoBastanteo = "A"; //A,B,C

Representante representante = new Representante(tipoDocumento,nroDocumento,nombre,apellido,cargoEnLaEmpresa,grupoBastanteo);

ClienteRepresentante clienteRepresentante = new ClienteRepresentante(representante.getTipoDocumento(),

representante.getNroDocumento(),

cliente.getCodigoCliente());

//Verifico

assertEquals(representante.getTipoDocumento(),clienteRepresentante.getTipoDocumento());

assertEquals(representante.getNroDocumento(),clienteRepresentante.getNroDocumento());

assertEquals(cliente.getCodigoCliente(),clienteRepresentante.getCodigoCliente());

}

}

**//clase BastanteoTest.java**

package bastanteo;

import static org.junit.Assert.assertEquals;

import org.junit.Test;

public class BastanteoTest {

@Test

public void BastanteoDeberiaCrearseConDatosIniciales(){

//Preparo el Escenario

String grupoBastanteo = "A";

String codigoCliente = "CL001";

String tipoDocumento = "DNI";

String nroDocumento = "40341631";

String tipoIntervencion = "A Sola firma";

long importe = 0;

String fechaVencimiento = "05/05/2012";

//Ejecuto el escenario

Bastanteo bastanteo = new Bastanteo(grupoBastanteo,codigoCliente,

tipoDocumento,nroDocumento,tipoIntervencion,importe,fechaVencimiento);

//Verifico

assertEquals(grupoBastanteo,bastanteo.getGrupoBastanteo());

assertEquals(codigoCliente,bastanteo.getCodigoCliente());

assertEquals(tipoDocumento,bastanteo.getTipoDocumento());

assertEquals(nroDocumento,bastanteo.getNroDocumento());

assertEquals(tipoIntervencion,bastanteo.getTipoIntervencion());

assertEquals(importe,bastanteo.getImporte());

assertEquals(fechaVencimiento,bastanteo.getFechaVencimiento());

}

}

**//clase AdmUsuarioTest.java**

package bastanteo;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;

public class AdmUsuarioTest {

@Test

public void siIngresoDatosDeberiaRegistarUsuario() throws MensajeException{

//Preparo el Escenario

String codigoUsuario = "csoriano";

String nombre = "Cesar";

String apellido = "Soriano";

String clave = "asd";

String codigoRol = "R001";

AdmUsuarios adm = new AdmUsuarios();

//ejecutar

adm.registrarUsuario(codigoUsuario,nombre,apellido,clave,codigoRol);

Usuario usuario = adm.buscarUsuario(codigoUsuario);

//Verificar

assertNotNull(usuario);

}

//Este test expera que haya una exception de tipo....

@Test(expected=MensajeException.class)

public void deberiaValidarDatosRequeridos() throws MensajeException{

//Preparo el Escenario

String codigoUsuario = "";

String nombre = "";

String apellido = "";

String clave = "";

String codigoRol = "";

AdmUsuarios adm = new AdmUsuarios();

//ejecutar

adm.registrarUsuario(codigoUsuario,nombre,apellido,clave,codigoRol);

}

@Test(expected=MensajeException.class)

public void noDeberiaRegistarUsuarioDuplicados() throws MensajeException{

//preparar el ejemplo

String codigoUsuario = "csoriano";

String nombre = "Cesar";

String apellido = "Soriano";

String clave = "asd";

String codigoRol = "R001";

AdmUsuarios adm = new AdmUsuarios();

//ejecutar

//El primero se ejecuta sin problemas

adm.registrarUsuario(codigoUsuario,nombre,apellido,clave,codigoRol);

//El segundo debe de arrojar una excepcion

adm.registrarUsuario(codigoUsuario,nombre,apellido,clave,codigoRol);

}

@Test

public void DeberiaLogearseConClaveCorreta() throws MensajeException{

//Inserto Primer Rol

String codigoRol = "R001";

String descripcionRol = "Supervisor";

boolean accesoMenuClientes = true;

boolean accesoMenuPoderes = true;

boolean accesoMenuBastanteo = true;

boolean accesoMenuUsuarios = true;

AdmRoles admr = new AdmRoles();

admr.registrarRol(codigoRol, descripcionRol, accesoMenuClientes,

accesoMenuPoderes, accesoMenuBastanteo, accesoMenuUsuarios);

//Inserto Segundo Rol

codigoRol = "R002";

descripcionRol = "Administrador";

accesoMenuClientes = false;

accesoMenuPoderes = false;

accesoMenuBastanteo = true;

accesoMenuUsuarios = true;

admr.registrarRol(codigoRol, descripcionRol, accesoMenuClientes,

accesoMenuPoderes, accesoMenuBastanteo, accesoMenuUsuarios);

//Inserto un usuario

String codigoUsuario = "csoriano";

String nombre = "Cesar";

String apellido = "Soriano";

String clave = "asd";

codigoRol = "R001";

AdmUsuarios adm = new AdmUsuarios();

adm.registrarUsuario(codigoUsuario,nombre,apellido,clave,codigoRol);

//Verifica logeo

codigoUsuario = "csoriano";

clave = "asd";

adm.verificaClaveUsuario(codigoUsuario, clave);

//Mensaje bienvenida y rol asignado

Rol rol = admr.buscarRol(codigoRol);

String sDescripcionRol = rol.getDescripcionRol();

System.out.println("Bienvenido como " + sDescripcionRol + " al Sistema de Bastanteo");

}

@Test(expected=MensajeException.class)

public void noDeberiaLogearseConClaveIncorreta() throws MensajeException{

//preparar el ejemplo

String codigoUsuario = "csoriano";

String nombre = "Cesar";

String apellido = "Soriano";

String clave = "asd";

String codigoRol = "R001";

AdmUsuarios adm = new AdmUsuarios();

//Grabo para poder validar en lista

adm.registrarUsuario(codigoUsuario,nombre,apellido,clave,codigoRol);

//ejecutar

adm.verificaClaveUsuario(codigoUsuario, "abc");

}

}

**//clase AdmRolTest.java**

package bastanteo;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;

public class AdmRolTest {

@Test

public void siIngresoDatosDeberiaRegistarRol() throws MensajeException{

//Preparo el Escenario

String codigoRol = "R001";

String descripcionRol = "Supervisor";

boolean accesoMenuClientes = true;

boolean accesoMenuPoderes = true;

boolean accesoMenuBastanteo = true;

boolean accesoMenuUsuarios = true;

AdmRoles adm = new AdmRoles();

//ejecutar

adm.registrarRol(codigoRol, descripcionRol, accesoMenuClientes,

accesoMenuPoderes, accesoMenuBastanteo, accesoMenuUsuarios);

Rol rol = adm.buscarRol(codigoRol);

//Verificar

assertNotNull(rol);

}

//Este test expera que haya una exception de tipo....

@Test(expected=MensajeException.class)

public void deberiaValidarDatosRequeridos() throws MensajeException{

//Preparo el Escenario

String codigoRol = "";

String descripcionRol = "";

boolean accesoMenuClientes = true;

boolean accesoMenuPoderes = true;

boolean accesoMenuBastanteo = true;

boolean accesoMenuUsuarios = true;

AdmRoles adm = new AdmRoles();

//ejecutar

adm.registrarRol(codigoRol, descripcionRol, accesoMenuClientes,

accesoMenuPoderes, accesoMenuBastanteo, accesoMenuUsuarios);

}

@Test(expected=MensajeException.class)

public void noDeberiaRegistarRolDuplicados() throws MensajeException{

//preparar el ejemplo

String codigoRol = "R001";

String descripcionRol = "Supervisor";

boolean accesoMenuClientes = true;

boolean accesoMenuPoderes = true;

boolean accesoMenuBastanteo = true;

boolean accesoMenuUsuarios = true;

AdmRoles adm = new AdmRoles();

//ejecutar

//El primero se ejecuta sin problemas

adm.registrarRol(codigoRol, descripcionRol, accesoMenuClientes,

accesoMenuPoderes, accesoMenuBastanteo, accesoMenuUsuarios);

//El segundo debe de arrojar una excepcion

adm.registrarRol(codigoRol, descripcionRol, accesoMenuClientes,

accesoMenuPoderes, accesoMenuBastanteo, accesoMenuUsuarios);

}

}

**//clase AdmRepresentanteTest.java**

package bastanteo;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;

public class AdmRepresentanteTest {

@Test

public void siIngresoDatosDeberiaRegistarRepresentante() throws MensajeException{

//Preparamos el ejemplo

String tipoDocumento = "DNI";

String nroDocumento = "40341631";

String nombre = "JOSE";

String apellido = "VASQUEZ BUSTAMANTE";

String cargoEnLaEmpresa = "ABOGADO";

String grupoBastanteo = "A"; //A,B,C

AdmRepresentante adm = new AdmRepresentante();

//ejecutar

adm.registrarRepresentante(tipoDocumento,nroDocumento,nombre,apellido,cargoEnLaEmpresa,grupoBastanteo);

Representante representante = adm.buscarRepresentante(tipoDocumento,nroDocumento);

//Verificar

assertNotNull(representante);

}

//Este test expera que haya una exception de tipo....

@Test(expected=MensajeException.class)

public void deberiaValidarDatosRequeridos() throws MensajeException{

//prepara el ejemplo

String tipoDocumento = "";

String nroDocumento = "40341631";

String nombre = "JOSE";

String apellido = "VASQUEZ BUSTAMANTE";

String cargoEnLaEmpresa = "ABOGADO";

String grupoBastanteo = "A"; //A,B,C

AdmRepresentante adm = new AdmRepresentante();

//ejecutar

adm.registrarRepresentante(tipoDocumento,nroDocumento,nombre,apellido,cargoEnLaEmpresa,grupoBastanteo);

assertNotNull(adm);

}

@Test(expected=MensajeException.class)

public void noDeberiaRegistarClienteDuplicados() throws MensajeException{

//preparar el ejemplo

String tipoDocumento = "DNI";

String nroDocumento = "40341631";

String nombre = "JOSE";

String apellido = "VASQUEZ BUSTAMANTE";

String cargoEnLaEmpresa = "ABOGADO";

String grupoBastanteo = "A"; //A,B,C

AdmRepresentante adm = new AdmRepresentante();

//ejecutar

//El primero se ejecuta sin problemas

adm.registrarRepresentante(tipoDocumento,nroDocumento,nombre,apellido,cargoEnLaEmpresa,grupoBastanteo);

//El segundo debe de arrojar una excepcion

adm.registrarRepresentante(tipoDocumento,nroDocumento,nombre,apellido,cargoEnLaEmpresa,grupoBastanteo);

}

}

**//clase AdmPoderTest.java**

package bastanteo;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;

public class AdmPoderTest {

@Test

public void siIngresoDatosDeberiaRegistrarPoderes() throws MensajeException{

//Preparo el Escenario

String codigoPoder = "CHCO";

String nombrePoder = "Cobreo de Cheques";

String tipoPoder = "Activo";

AdmPoder adm = new AdmPoder();

//ejecutar

adm.registrarPoder(codigoPoder, nombrePoder, tipoPoder);

Poder poder = adm.buscarPoder(codigoPoder);

//Verificar

assertNotNull(poder);

}

//Este test expera que haya una exception

@Test(expected=MensajeException.class)

public void deberiaValidarDatosRequeridos() throws MensajeException{

//Preparo el Escenario

String codigoPoder = "";

String nombrePoder = "";

String tipoPoder = "";

AdmPoder adm = new AdmPoder();

//ejecutar

adm.registrarPoder(codigoPoder, nombrePoder, tipoPoder);

}

@Test(expected=MensajeException.class)

public void noDeberiaRegistarPoderesDuplicados() throws MensajeException{

//Preparo el Escenario

String codigoPoder = "CHCO";

String nombrePoder = "Cobreo de Cheques";

String tipoPoder = "Activo";

AdmPoder adm = new AdmPoder();

//ejecutar

adm.registrarPoder(codigoPoder, nombrePoder, tipoPoder);

//Lo vuelvo a ejecutar para probar el error

adm.registrarPoder(codigoPoder, nombrePoder, tipoPoder);

}

}

**//clase AdmGrupoTest.java**

package bastanteo;

import static org.junit.Assert.assertNotNull;

import org.junit.Test;

public class AdmGrupoTest {

@Test

public void siIngresoDatosDeberiaRegistrarGrupos() throws MensajeException{

//Preparo el Escenario

String grupoBastanteo = "A";

String descripcionGrupo = "Grupo A";

AdmGrupo adm = new AdmGrupo();

//ejecutar

adm.registrarGrupo(grupoBastanteo, descripcionGrupo);

Grupo grupo = adm.buscarGrupo(grupoBastanteo);

//Verificar

assertNotNull(grupo);

}

//Este test expera que haya una exception

@Test(expected=MensajeException.class)

public void deberiaValidarDatosRequeridos() throws MensajeException{

//Preparo el Escenario

String grupoBastanteo = "";

String descripcionGrupo = "";

AdmGrupo adm = new AdmGrupo();

//ejecutar

adm.registrarGrupo(grupoBastanteo, descripcionGrupo);

}

@Test(expected=MensajeException.class)

public void noDeberiaRegistarGruposDuplicados() throws MensajeException{

//Preparo el Escenario

String grupoBastanteo = "";

String descripcionGrupo = "";

AdmGrupo adm = new AdmGrupo();

//ejecutar

adm.registrarGrupo(grupoBastanteo, descripcionGrupo);

//Lo vuelvo a ejecutar para probar el error

adm.registrarGrupo(grupoBastanteo, descripcionGrupo);

}

}

**//clase AdmClienteTest.java**

package bastanteo;

import static org.junit.Assert.\*;

import org.junit.Test;

public class AdmClienteTest {

@Test

public void siIngresoDatosDeberiaRegistarClientes() throws MensajeException{

//Preparamos el ejemplo

String ruc = "20107012011";

String codigo = "CL001";

String razonSocial = "ACME Solutions";

String fechaInicio = "01/01/2011";

String tipoEmpresa = "SAC";

int empleados = 10;

AdmClientes adm = new AdmClientes();

//ejecutar

adm.registrarCliente(codigo,ruc,razonSocial,fechaInicio,tipoEmpresa,empleados);

Cliente cliente = adm.buscarCliente(codigo);

//Verificar

assertNotNull(cliente);

}

//Este test expera que haya una exception de tipo....

@Test(expected=MensajeException.class)

public void deberiaValidarDatosRequeridos() throws MensajeException{

//prepara el ejemplo

String ruc = "";

String codigo = "";

String razonSocial = "";

String fechaInicio = "01/01/2011";

String tipoEmpresa = "SAC";

int empleados = 10;

AdmClientes adm = new AdmClientes();

//ejecutar

adm.registrarCliente(codigo,ruc,razonSocial,fechaInicio,tipoEmpresa,empleados);

}

@Test(expected=MensajeException.class)

public void noDeberiaRegistarClienteDuplicados() throws MensajeException{

//preparar el ejemplo

String ruc = "20107012011";

String codigo = "CL001";

String razonSocial = "ACME Solutions";

String fechaInicio = "01/01/2011";

String tipoEmpresa = "SAC";

int empleados = 10;

AdmClientes adm = new AdmClientes();

//ejecutar

//El primero se ejecuta sin problemas

adm.registrarCliente(codigo,ruc,razonSocial,fechaInicio,tipoEmpresa,empleados);

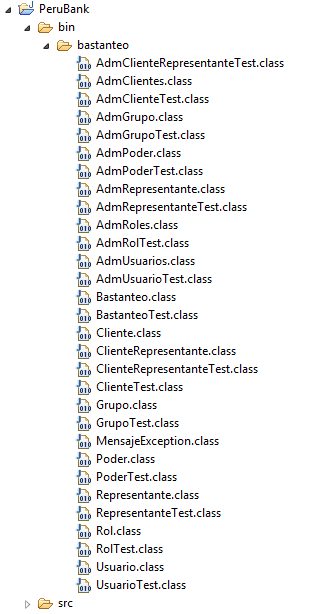
//El segundo debe de arrojar una excepcion

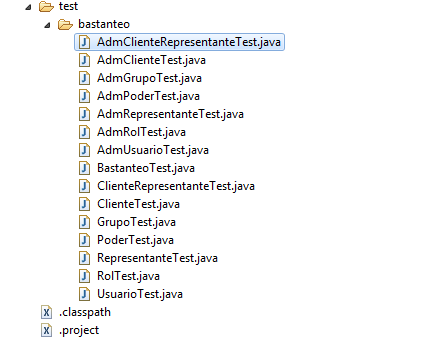
adm.registrarCliente(codigo,ruc,razonSocial,fechaInicio,tipoEmpresa,empleados);

}

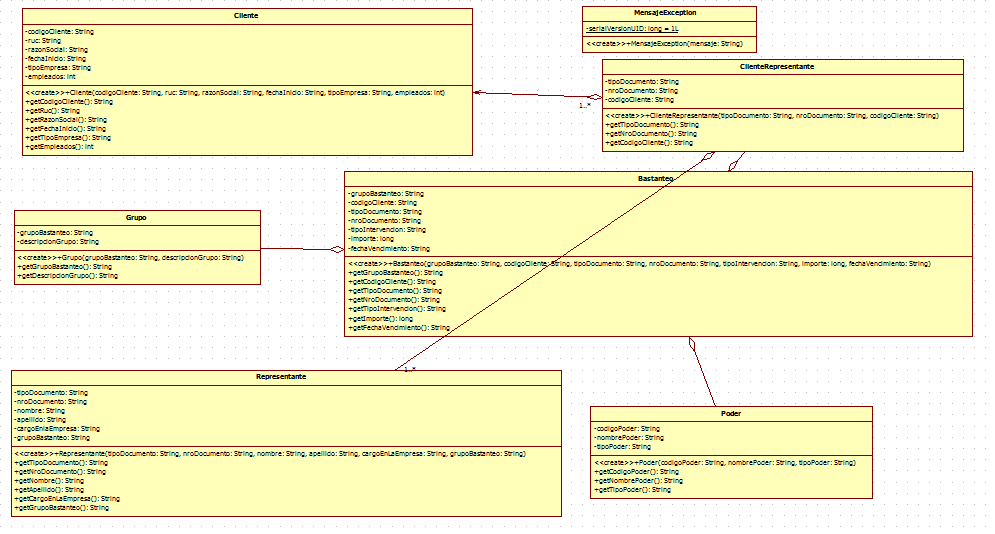
}

1. **ARQUITECTURA Y CODIGO FUENTE**





1. **DIAGRAMA UML**



1. **BIBLIOGRAFIA**