Corrección Diseño y Modelo de Dominio

Contenido

[Modelo de Dominio 1](#_Toc74950848)

[Diseño 3](#_Toc74950849)

[Puja de un producto 3](#_Toc74950850)

[CrearPujasGUI 4](#_Toc74950851)

[BLFacade 4](#_Toc74950852)

[DataAccess 5](#_Toc74950853)

# Modelo de Dominio



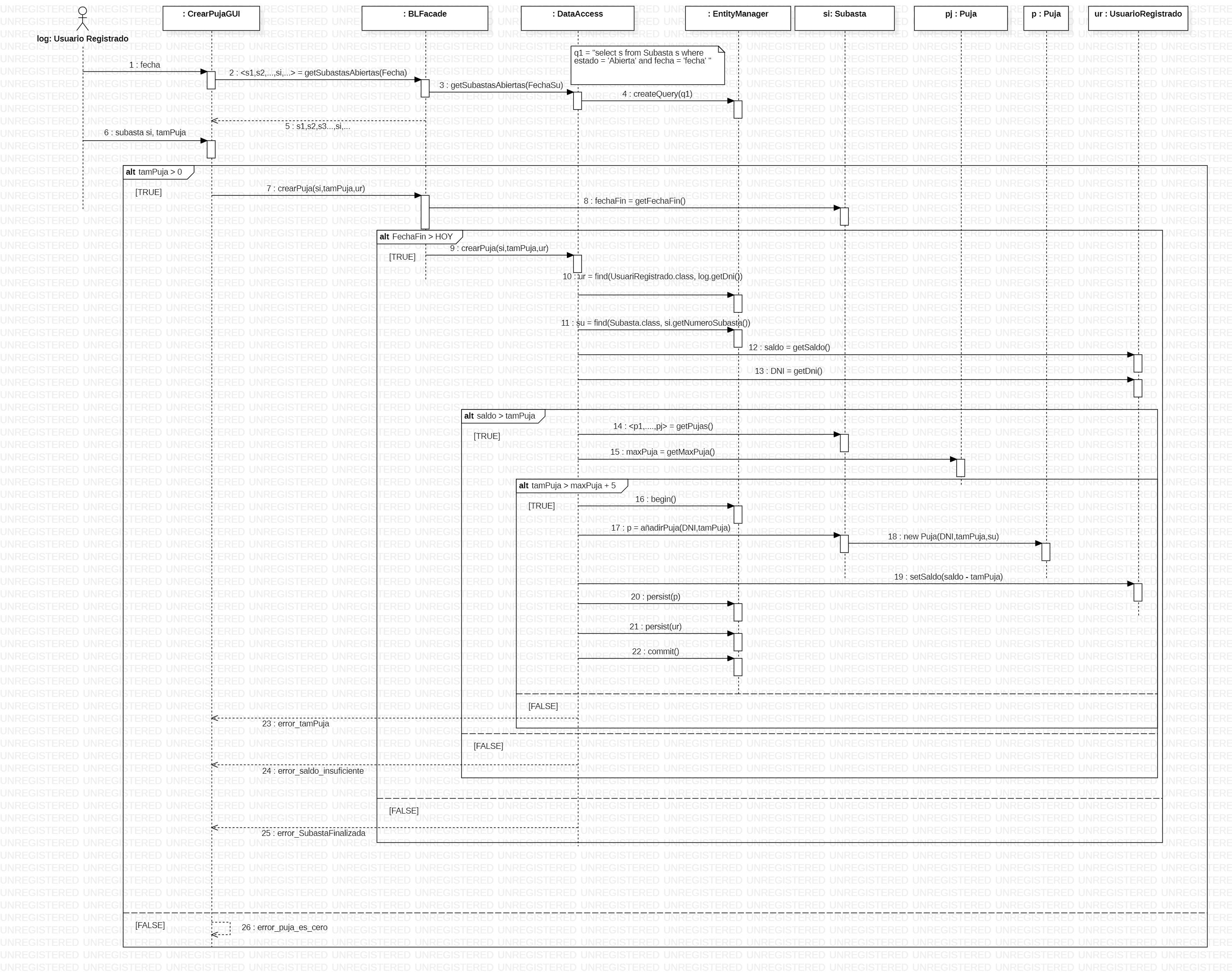
Los principales cambios en este caso han sido dentro de la clase “Subastas”, los siguientes:

* Movimiento del atributo “Estado” a la última posición.
* Renombrado del atributo “ID” por “Numero de Subasta”, convirtiendo este en el atributo principal.
* Eliminación del atributo “Fecha de finalización”.
* Eliminación del atributo “Usuario ganador”.
* Eliminación del atributo “Puja máxima”.

Además, he convertido en atributos principales en las clases “Usuario Registrado” y “Puja”, “DNI” y “ID” respectivamente. Esto se debe a que en la implementación estaban correctamente atribuidos, para el caso de uso “Crear Puja” por ejemplo, pero no estaban correctos en el modelo de dominio.

# Diseño

## Puja de un producto



Para este primer diseño de un caso de uso, he debido de modificar gran parte de la implementación, ya que como se pudo ver en la demostración del proyecto, la implementación distaba de un correcto diseño.

Para empezar, he distinguido correctamente dentro de la implementación las diferentes clases dentro del Modelo de Dominio, separando de este modo, la clase “Login” en tres diferentes clases: “Empleado”, “Propietario” y “UsuarioRegistrado”. Además he renombrado correctamente las clases tales como “Event” por “Subasta” o “Question” por “Puja” junto con sus atributos y funciones. De este modo, y para el diseño de este caso de uso el código implementado queda tal que asi:

### CrearPujasGUI

float tamPuja = Float.*parseFloat*(jTextFieldPrice.getText());

if (tamPuja <= 0 ) jLabelError.setText(ResourceBundle.*getBundle*("Etiquetas").getString("ErrorNumber"));

else {

BLFacade facade = LoginGUI.*getBusinessLogic*();

facade.crearPuja(event, tamPuja, login);

jLabelMsg.setText(ResourceBundle.*getBundle*("Etiquetas").getString("QueryCreated"));

}

Vector<domain.Subasta> subastas = facade.getSubastasAbiertas(firstDay);

### BLFacade

public Puja crearPuja(Subasta subasta, float tamPuja, UsuarioRegistrado login) throws EventFinished, QuestionAlreadyExist, PujaPequeña, SaldoInsuficiente{

dbManager.open(false);

Puja qry=null;

if(new Date().compareTo(subasta.getFechaSubasta())>0)

throw new EventFinished(ResourceBundle.*getBundle*("Etiquetas").getString("ErrorEventHasFinished"));

qry=dbManager.crearPuja(subasta, tamPuja, login);

dbManager.close();

return qry;

};

### DataAccess

public Puja crearPuja(Subasta subasta, float tamPuja, UsuarioRegistrado login) throws QuestionAlreadyExist, PujaPequeña, SaldoInsuficiente {

UsuarioRegistrado ur = *db*.find(UsuarioRegistrado.class, login.getDni());

Subasta su = *db*.find(Subasta.class, subasta.getNumeroSubasta());

double saldo = ur.getSaldo();

String DNI = ur.getDni();

if (tamPuja > saldo) throw new SaldoInsuficiente(ResourceBundle.*getBundle*("Etiquetas").getString("ErrorSaldoInsuficiente"));

Vector<Puja> pujas = su.getPujas();

Puja ultimaPuja = pujas.lastElement();

double maxPuja = ultimaPuja.getBetMinimum();

if (maxPuja + 5 > tamPuja) throw new PujaPequeña(ResourceBundle.*getBundle*("Etiquetas").getString("ErrorPujaPequeña"));

*db*.getTransaction().begin();

Puja p = su.anadirPuja(DNI, tamPuja);

ur.setSaldo(saldo - tamPuja);

*db*.persist(ur);

*db*.persist(p);

*db*.getTransaction().commit();

return p;

}