

DISEÑO DE SOFTWARE

GRUPO 3

Markos Agote
Alex Calderón
Bastiaan Handels
Pablo De Erauso



INDICE

1. BREVE DESCRIPCIÓN DEL PROYECTO	3
2. DIAGRAMA DE OBJETOS JDO	3
Entidad USUARIO	4
Entidad RESERVA	4
Entidad ALOJAMIENTO	4
Entidad HABITACIÓN	5
Relaciones entre las Entidades	5
3. EXPLICACIÓN DE LAS PRINCIPALES FUNCIONALIDADES Y LOS EVOLUTIVOS IMPLEMENTADOS EN EL CÓDIGO	5
3.1 Patrones	5
3.2 Ventanas	6



1. BREVE DESCRIPCIÓN DEL PROYECTO

La finalidad de nuestro proyecto se va a centrar en una aplicación que ofrezca diferentes tipos de alojamientos (hoteles, apartamentos, casas rurales...) a los usuarios para que puedan disfrutar de sus vacaciones en sus destinos favoritos. Los alojamientos nos los proporcionará nuestro proveedor de confianza Hotel Provider.

Entre las principales funcionalidades que van a poder tener nuestros clientes destacamos las siguientes: registrarse e iniciar sesión, buscar alojamientos, realizar reservas, acceder a reservas realizadas previamente...

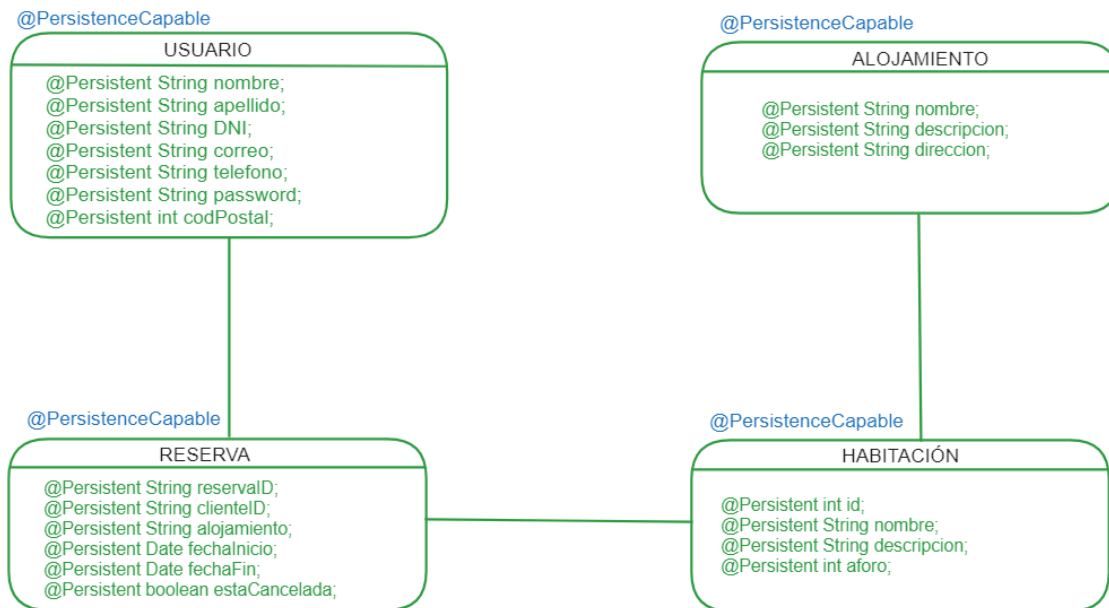
Para ello, vamos a dividir nuestra aplicación en 2 partes: la parte frontend y la parte backend.

Respecto al frontend, vamos a diferenciar 2 tipos: En primer lugar, la del **alquiler de habitaciones**, disponible para cualquier tipo de usuarios; y en segundo lugar, la de los **dispositivos instalados en las puertas** para la comprobación de los códigos QR generados a la hora de realizar una reserva, que hablaremos más tarde de ello.

La conexión entre el backend y los frontends será a través de Java RMI, y para el desarrollo de este proyecto será necesario utilizar los servicios de HotelProvider.

HotelProvider es una empresa ficticia líder en la distribución mayorista de apartamentos y habitaciones de hotel a nivel global la cual nos proporciona una interfaz de programación de aplicaciones (API) basada en REST que permite a los minoristas acceder a información detallada sobre la disponibilidad de habitaciones, así como realizar compras directas de las mismas, entre otras funciones.

2. DIAGRAMA DE OBJETOS JDO



El diagrama JDO (Java Data Objects) presentado describe la estructura del sistema de gestión de HotelProviderI. El diagrama está compuesto por cuatro entidades principales: USUARIO, RESERVA, ALOJAMIENTO, y HABITACIÓN. A continuación, se explica cada una de estas entidades y cómo se relacionan entre sí:

Entidad USUARIO

La entidad USUARIO representa a los clientes o usuarios del sistema. Los campos persistentes incluyen:

- nombre y apellido: Que almacenan el nombre completo del usuario.
- DNI: Un identificador único para el usuario, como un número de documento nacional de identidad.
- correo: La dirección de correo electrónico del usuario.
- telefono: Número de teléfono del usuario.
- password: Contraseña para el acceso seguro al sistema.
- codPostal: Código postal de la dirección del usuario.

Esta entidad es crucial para gestionar la información personal de los clientes y para garantizar que las interacciones dentro del sistema sean seguras y personalizadas.

Entidad RESERVA

La entidad RESERVA gestiona las reservaciones hechas por los usuarios. Los campos incluyen:

- reservaID: Identificador único de cada reserva.
- clienteID: Referencia al DNI del usuario que hace la reserva.
- alojamiento: Nombre del alojamiento reservado.



- fechaInicio y fechaFin: Fechas que definen el período de la estancia.
- estaCancelada: Un booleano que indica si la reserva ha sido cancelada.

Esta entidad conecta directamente con `USUARIO` mediante `clienteID`, asegurando que cada reserva está vinculada a un usuario específico.

Entidad ALOJAMIENTO

La entidad `ALOJAMIENTO` describe los alojamientos disponibles para reservar. Los campos incluyen:

- nombre: Nombre del alojamiento.
- descripcion: Una breve descripción del alojamiento.
- direccion: La dirección física del alojamiento.

`ALOJAMIENTO` está diseñado para almacenar detalles relevantes que los usuarios necesitan conocer al hacer una reserva. Estos detalles ayudan a los usuarios a tomar decisiones informadas sobre dónde desean hospedarse.

Entidad HABITACIÓN

Finalmente, la entidad `HABITACIÓN` se relaciona con `ALOJAMIENTO` y describe las habitaciones individuales disponibles dentro de cada alojamiento. Sus atributos son:

- id: Un identificador único para cada habitación.
- nombre: Un nombre o número de habitación.
- descripcion: Detalles específicos de la habitación.
- aforo: La capacidad máxima de personas que pueden alojarse en la habitación.

La relación entre `HABITACIÓN` y `ALOJAMIENTO` permite gestionar las características y disponibilidad de habitaciones específicas dentro de cada establecimiento, facilitando así la gestión de reservas basadas en las necesidades del cliente y la capacidad del alojamiento.

Relaciones entre las Entidades

Las relaciones entre estas entidades son fundamentales para el funcionamiento del sistema. `RESERVA` conecta `USUARIO` con `ALOJAMIENTO` a través de los campos `clienteID` y `alojamiento`, lo que permite a los usuarios reservar habitaciones específicas en diferentes alojamientos. A su vez, `HABITACIÓN` proporciona detalles específicos que se vinculan al alojamiento general.

Este diagrama JDO ayuda a visualizar cómo se estructura y se gestiona la información en el sistema, lo cual es crucial para el desarrollo y mantenimiento eficiente del software en el contexto de un proveedor de alojamientos.



3. EXPLICACIÓN DE LAS PRINCIPALES FUNCIONALIDADES Y LOS EVOLUTIVOS IMPLEMENTADOS EN EL CÓDIGO

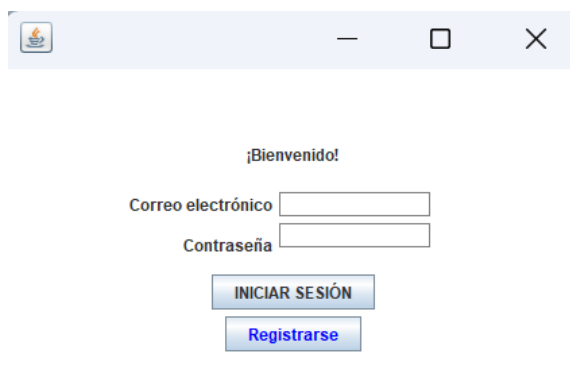
3.1 Patrones

Para esta entrega hemos incluido 6 patrones de diseño, los 4 obligatorios (Controller, ServiceLocator, Faccace y AppService) y otros 2 opcionales (DTO y DAO) que estamos acabando de implementarlos. A continuación vamos a ir analizandolos individualmente:

- **Controller:** es un patrón de la parte de cliente que actúa como “main” del cliente y hace de intermediario entre la parte visual y el servidor. Entre sus principales funciones destacamos las siguientes: crear la parte visual (gui), crear el objeto DomainObject e instanciar el patrón ServiceLocator.
- **ServiceLocator:** es el otro patrón de la parte de cliente que se encarga de realizar la conexión al servidor. Hemos implementado 2 métodos principales: getService() y setService(). El primero de ellos devuelve el objeto StubServer ya instanciado y en este caso se harán tantas llamadas como solicitudes al servicio se hacen. El método setService significa conectarse (lookup) y se hace solo 1 llamada al principio.
- **Faccade:** es un patrón de la parte servidora al cual le llega la conexión del cliente. Además aquí se hace el método rebind.
- **AppService:** es otro patrón de la parte del servidor que actúa de intermediario entre la fachada y otro tipo de patrones/paquetes (DAO, DTO...). Es el núcleo del servidor (como el controller en la parte cliente) y hace la implementación de faccade.
- **DTO:** es un patrón que sirve para transferir información (DATA TRANSFER OBJECT), pero permite mandar únicamente los atributos concretos que nos convienen. Tiene un assembler que transforma un objeto entero en un objeto DTO
- **DAO:** es el patrón que se encarga de conectarse a la base de datos.

3.2 Ventanas

Para esta entrega hemos creado las 3 primeras ventanas; la primera de ellas es una ventana básica donde se pide al usuario iniciar sesión con un correo electrónico y contraseña previamente creado. También nos da la opción de registrarnos en caso de no haberlo hecho previamente.



¡Bienvenido!

Correo electrónico

Contraseña

INICIAR SESIÓN

Registrarse

La segunda ventana es precisamente la que nos permite registrarnos introduciendo los datos que se nos piden: nombre, apellidos, correo electrónico, nº de teléfono, DNI y contraseña.



¡Regístrate!

NOMBRE

APELLIDO

CORREO ELECTRÓNICO

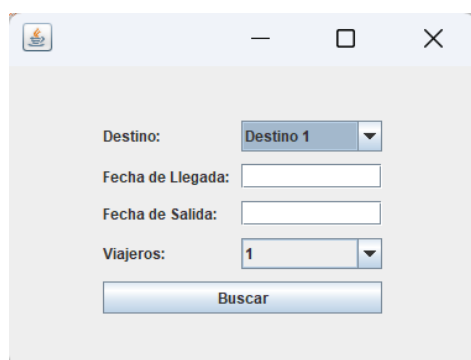
Nº TELÉFONO

DNI

CONTRASEÑA

ATRÁS **Registrarse**

Por último, la última ventana que hemos añadido para esta entrega es la “ventana principal”. Sin embargo, el diseño ha cambiado respecto al prototipo inicial que pensamos en la primera entrega debido a las limitaciones que presenta Java para hacer la parte visual. En esta tercera ventana aparecen 4 filtros (destino, fecha de salida, fecha de llegada y número de viajeros) donde cada usuario introducirá distintos datos y en base a ellos, en la siguiente ventana, le saldrán distintos alojamientos.



Destino:

Fecha de Llegada:

Fecha de Salida:

Viajeros:

Buscar