Deusto
Universidad de Deusto
Deustuko Unibertsitatea
University of Deusto

DISEÑO DE SOFTWARE

GRUPO 3

Markos Agote
Alex Calderón
Bastiaan Handels
Pablo De Erauso

ÍNDICE

1. INTRODUCCIÓN

BAMP S.L. es un grupo de estudiantes y amigos de la Universidad de Deusto que emplean la función de desarrolladores de software dentro de la propia sociedad limitada. El grupo fue fundado por Markos Agote, Alex Calderón, Pablo De Erauso y Bastiaan Handels en 2023 con el fin de realizar una aplicación para una asignatura que íbamos a cursar.

Tras el éxito del anterior lanzamiento, una app delivery llamada APP-BAMP, nos hemos propuesto realizar un nuevo proyecto en relación a una nueva aplicación relacionada con la gestión de diferentes tipos de alojamientos.

Este nuevo reto nos motiva a indagar nuevas tecnologías y aplicar nuestros conocimientos con el fin de innovar y mejorar la experiencia de los usuarios. Estamos fascinados por este desafío y por la confianza y oportunidad que se nos brinda y confiamos en la obtención de una exitosa solución que marque una diferencia en el mercado.


Markos Agote


Alex Calderón


Bastiaan Handels


Pablo De Erauso

2. DESCRIPCIÓN Y ALCANCE 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.

En cuanto al **alcance** final de nuestro proyecto, preferimos no definirlo de manera rígida desde el principio debido a posibles contratiempos que podrían surgir, lo cual podría ocasionar retrasos imprevistos. No obstante, lo que sí que tenemos claro es la base que queremos cubrir y hacia dónde queremos dirigirnos: en primer lugar, nos proponemos ofrecer a nuestros clientes la capacidad de buscar una variedad de alojamientos utilizando diferentes filtros, tales como tipo de alojamiento, número de personas, fechas de entrada y salida, rango de precios, entre otros. A continuación, estos clientes podrán realizar su reserva indicando detalles como fechas, número de clientes, etc. Para confirmar esto, queremos que nuestra aplicación genere un código QR de manera automática que proporcione al cliente mediante el envío de un mail de confirmación al correo electrónico del mismo, resumiendo los datos más importantes de su reserva. Por último, queremos que nuestros clientes puedan ver las reservas que han realizado previamente en el historial de reservas.

3. DISEÑO DE LAS API

3.1. MODELO ENTIDAD RELACIÓN

A continuación presentamos el modelo entidad relación del proyecto para una mejor comprensión y visualización estructural.

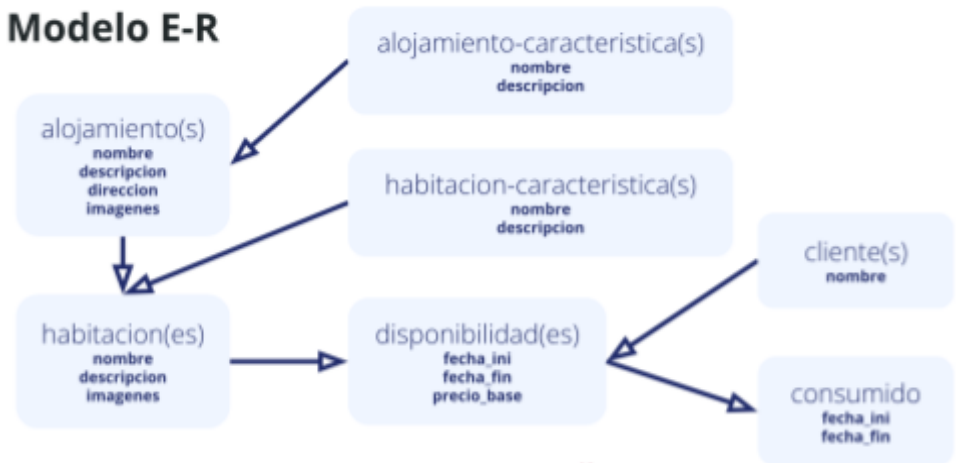

Image 1: Modelo Entidad Relación

3.2. MÉTODOS DEL SERVIDOR

En relación a los métodos definidos que expondrá la interfaz remota del servidor, hemos diferenciado diferentes grupos o áreas que dispondrán los diferentes métodos que se implementarán.

Nota: El output y los atributos que requiere el método se podrán ver en la definición de los mismos.

1. Gestión de Reservas: Métodos en relación a la realización y cancelación de reservas.

- **public Reserva CrearReserva (String ClienteID, String Alojamiento, Date FechaInicio, Date FechaFin)**

  Este método va a permitir a un usuario crear una nueva reserva, generando un código QR y el envío de un mail con el fin de una entrada autónoma e independiente al alojamiento.

- **public String CancelarReserva (String ReservaID)**

  Este método va a permitir realizar la cancelación de una reserva existente y actualizar el estado de la misma en el sistema.

  La salida de este método va a devolver un String indicando el estado de la reserva. (Valoramos la modificación a un boolean).

- **public boolean ModificarReserva (String reservaID, Date nuevaFechaInicio, Date nuevaFechaFin)**

  Este método permite a los usuarios modificar las fechas de una reserva existente. Al proporcionar un reservaID junto con las nuevas fechas de inicio y fin, el sistema verifica la disponibilidad y, si es posible, actualiza la reserva con las nuevas fechas.

  El output es true si la modificación de la reserva se ha realizado con éxito o false en caso contrario.

- **public List<Reserva> ObtenerReservasUsuario (String usuarioID)**

  Este método devuelve un listado de todas las reservas realizadas por un usuario en específico, identificando por su usuarioID. Esto incluye reservas pasadas y actuales, proporcionando una vista completa del historial de reservas del usuario en el sistema.

  La salida devuelve una lista de objetos (List<Reserva>) en la que cada objeto representa una reserva realizada por el usuario.

2. Gestión de Alojamientos: Métodos en relación a la búsqueda y detalles de alojamientos.

- **public List<Alojamiento> BuscarAlojamientosPorFiltro (Date FechaInicio, Date FechaFin, String Ubicacion, int CantidadHuespedes, float Precio, String tipoAlojamiento)**

  Este método se encarga de devolver una serie de alojamientos disponibles según los criterios de búsqueda impuestos por el cliente.

  El output devuelve una lista de alojamientos.

- **public Alojamiento ObtenerDetallesAlojamiento (String AlojamientoID)**

  Este método proporciona los detalles completos de un alojamiento específico.

  La salida de este método devolverá los detalles del alojamiento especificado (Alojamiento). Al ser de tipo Alojamiento, podremos saber el nombre, descripción, dirección y también ver las imágenes del alojamiento.

3. Gestión de autenticación: Métodos relacionados con el registro, inicio de sesión, autenticación y datos del usuario.

- **public boolean IniciarSesion(String correoElectronico, String contraseña)**

  Este método sirve para poder acceder a la aplicación de la empresa y así buscar pisos y hacer reservas. Para ello deberá validar el correo y la contraseña introducida por el usuario. Si no es correcta o no existe no le dejará entrar y tendrá que volver a intentarlo o registrase en el sistema si es la primera vez que accede.

  La salida devuelve un boolean (true si se ha iniciado correctamente y false si no se ha podido iniciar sesión)

- **public boolean Registrarse(String nombre, String apellidos, String DNI, String correoElectronico, String telefono, String password, int codPostal)**

  Este método crea un perfil para poder acceder al sistema. Los datos que tendrá que introducir el usuario para crear el perfil son los siguientes: nombre, apellidos, DNI, teléfono, correo electrónico, contraseña y código postal.

  El método devuelve un boolean (true si se ha creado el perfil correctamente y false si no se ha podido crear el perfil).

- **public boolean ModificarDatosUsuario (String nombre, String apellidos, String DNI, String correoElectrónico, String teléfono, String password, int codPostal)**

  Este método sirve para modificar cualquiera de los datos del perfil que se ha creado el usuario previamente.

  Devuelve un boolean (true si se ha actualizado el perfil correctamente y false si no se ha podido actualizar el perfil)

4. Gestión de Puertas: Métodos en relación al acceso de alojamientos.

- **public boolean ValidarAcceso (String CodigoQR)**

  El método recibe un código QR como entrada, lo compara con las reservas existentes para verificar su validez y, según el resultado, retorna un estado que indica si el acceso a la habitación o apartamento está permitido o no.

  Este método devuelve el acceso el estado del acceso (String).

5. Reseñas: Métodos en relación a las reseñas y experiencias de los usuarios.

- **public boolean AñadirReseña (String usuarioID, String alojamientoID, int calificacion, String comentario)**

  Este método permite a un usuario añadir una reseña sobre un alojamiento específico. El usuario debe proporcionar su ID, el ID del alojamiento que desea reseñar, una calificación numérica (De 1 a 5 estrellas) y un comentario opcional que detalle su experiencia.

  Devuelve un boolean (*true* si la reseña se ha añadido correctamente, *false* si no se ha podido añadir por algún motivo, como por ejemplo, restricciones sobre la adición de múltiples reseñas para el mismo alojamiento por el mismo usuario).

- **public List<Reseña> ObtenerReseña** (String alojamientoID)

  Este método recupera todas las reseñas asociadas a un alojamiento específico. Al proporcionar el ID de un alojamiento, el sistema buscará y devolverá todas las reseñas que los usuarios hayan dejado sobre dicho alojamiento.

  Devuelve una lista de objetos de tipo Reseña, donde cada Reseña incluye el usuarioID, calificacion, comentario, y la fecha en que se realizó la reseña.

- **public double CalcularEstrellasPromedio (String alojamientoID)**

  Este método calcula la calificación promedio en estrellas de un alojamiento, basándose en todas las reseñas dejadas por los usuarios. Utiliza el ID del alojamiento para identificar todas las reseñas relevantes y calcular el promedio.

  Devuelve un número decimal que representa la calificación promedio en estrellas del alojamiento. Si el alojamiento no tiene reseñas, podría devolver un valor predeterminado o indicar que aún no ha sido calificado.

- **public List<Alojamiento> ObtenerTop5Alojamientos**()

  Este método identifica y devuelve los 5 alojamientos con las calificaciones promedio más altas de entre todos los alojamientos registrados en el sistema.

  Devuelve una lista de 5 objetos de tipo Alojamiento, cada uno con sus detalles como el AlojamientoID, nombre, descripción, dirección, y calificación promedio. Estos objetos representan los 5 alojamientos mejor calificados en todo el sistema.

Estos métodos son el punto de partida para nuestro proyecto, pero lo más probable es que no sean todos. A medida que avancemos en la implementación, es probable que encontremos la necesidad de ajustar algunos aspectos o incluso de añadir nuevos métodos para adaptarnos mejor a las necesidades que nos surjan.

3.3. DIAGRAMA DE DESPLIEGUE

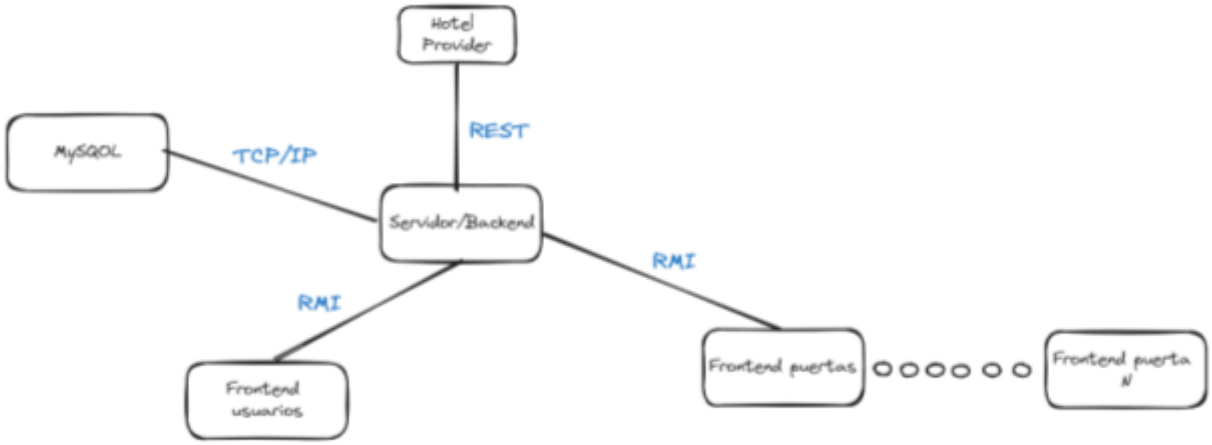

Image 2: Diagrama de despliegue.

3.4. DIAGRAMA DE COMPONENTES

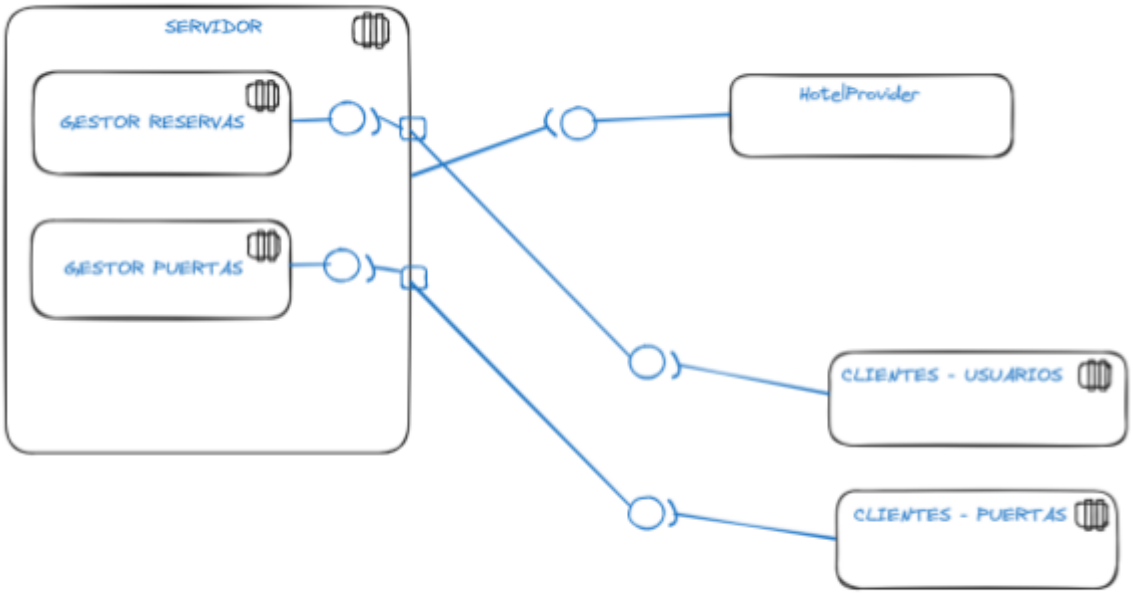

Image 3: Diagrama de componentes.

4. DISEÑO DE LA INTERFAZ GRÁFICA

A continuación mostraremos una aproximación al diseño final de la interfaz gráfica de los clientes para toda la funcionalidad del sistema que podrá ser modificada en la implementación, según vayamos desarrollando la aplicación.

En primer lugar, nada más entrar a la aplicación encontraremos la interfaz gráfica de Inicio de sesión, donde el usuario podrá iniciar sesión introduciendo su correo electrónico y contraseña que ha creado previamente.

Image 4.1: Inicio de sesión.

Si los datos ingresados son correctos y se hace clic en el botón de Iniciar Sesión, nos dirigiremos a la ventana principal de la aplicación, donde se presentan los alojamientos destacados basados en las reseñas de los usuarios [Image 4.3].

De lo contrario, si los datos introducidos son incorrectos, se ofrecerá la posibilidad de reintentar el inicio de sesión hasta un máximo de 3 veces. Si después de tres intentos los datos siguen siendo incorrectos, la aplicación se cerrará automáticamente.

Además, desde la pantalla de inicio de sesión, se ofrece la opción de registrarse para nuevos usuarios. Al hacer clic en ‘Registrarse’, seremos redirigidos a la página de Registro [Image 4.2].

Image 4.2: Registro

En la ventana de Registro [Image 4.2], es esencial completar todos los campos requeridos, incluyendo nombre, apellido, correo electrónico, número de teléfono, contraseña y código postal. Este paso es necesario para asegurar que el proceso de registro se realice correctamente y que los datos se almacenen de manera segura en la base de datos del sistema.

Una vez dentro de la aplicación, en la ventana principal [Image 4.3], se muestra una interfaz que exhibe los 6 alojamientos mejor valorados según las reseñas de otros usuarios. Además, contamos con una serie de filtros accesibles para personalizar la búsqueda según nuestras preferencias específicas, tales como destino, fechas de salida y llegada, y cantidad de viajeros.

Debido a la implementación de las interfaces gráficas del proyecto mediante la librería gráfica Java Swing y el alcance limitado de la misma, la selección de las opciones de filtrado se realiza a través del uso de JComboBox, lo que facilita la interacción sin complicaciones excesivas.

Image 4.3: Ventana principal

Al explorar la aplicación y encontrar un alojamiento de tu interés, podrás hacer clic en él para obtener más información. Esto abrirá la ventana de detalles del alojamiento [Image 4.4], donde encontraremos información más detallada como la ubicación, el precio por noche y una descripción completa del lugar. Además, esta ventana muestra una galería de imágenes que te permite visualizar mejor el espacio y todas sus comodidades antes de hacer una reserva, permitiéndonos así tener una mejor apreciación del lugar antes de realizar una reserva.

Image 4.4: Detalles alojamiento

Por otro lado, en la parte superior derecha de la ventana principal [Image 4.3] encontraremos un icono que nos permite acceder a la configuración de nuestro perfil, donde podemos actualizar nuestros datos personales o revisar reservas anteriores. Al seleccionar esta opción, aparecerá la ventana de perfil [Image 4.5], donde se podrán ver y editar los datos personales como nombre, apellido, número de teléfono, correo electrónico, código postal y contraseña. Si se desea cambiar la contraseña, simplemente hay que hacer clic en el enlace correspondiente. Aquí también se tendrá la oportunidad de revisar tus reservas anteriores al seleccionar el botón 'RESERVAS ANTERIORES'.

Image 4.5: Ventana perfil

Finalmente, si se desean revisar todas las reservas que han sido realizadas anteriormente, pueden verse en la ventana de 'Reservas anteriores' [Image 4.6]. Aquí se mostrarán las imágenes de los alojamientos que han sido reservados en el pasado, y se tendrá la opción de ver más detalles de cada reserva. En caso de que no se haya realizado ninguna reserva

anterior, la ventana indicará que no hay más reservas previas y ofrecerá la opción de realizar una nueva reserva.

Imagen 4.6: Reservas anteriores

5. CONCLUSIÓN

En conclusión, apreciamos la importancia de realizar un informe previo antes de la implementación del proyecto con el fin de garantizar una estructuración adecuada antes de comenzar. Sin embargo, es importante tener en cuenta que la planificación e informe inicial puede requerir ajustes y modificaciones a medida que avanza el desarrollo de la aplicación.

Por último, queremos destacar que el nuevo proyecto de BAMP S.L. promete ser una solución integral para la gestión de alojamientos, marcando una diferencia significativa en el mercado, ofreciendo a los usuarios el valor de una herramienta valiosa para planificar y disfrutar de sus viajes y estancias, combinando un diseño sólido de API con una interfaz gráfica atractiva y funcional.