Integracion de Tecnologías Web en el Desarrollo de una Plataforma de Reservas de Hotel

Nombre	Padron	Mail
Yanes Gabriela	108325	@fi.uba.ar
Pedoni Thiago	112611	@fi.uba.ar
Blanco Inti	111597	t@fi.uba.ar
Romero Blas Andres	111473	baromero@fi.uba.ar

Resumen

Este documento ejemplifica el proceso de diseño y desarrollo de un sitio web de una cadena de hoteles, diseñado para satisfacer las necesidades de los clientes y administradores de hotelería. Se consideran los principios de diseño de experiencia de usuario básicos, el diseño de un sistema de reservas de habitaciones que responda de manera fluida y la implementacion de un diseño web responsivo para un rendimiento optimo en todos los dispositivos. Preservando un enfoque centrado en el usuario, nuestro sitio web tiene como objetivo elevar la presencia en línea del hotel, mejorar la satisfaccion de los huéspedes y simplificar tareas administrativas para el hotel

Palabras clave: Hotel, Reservas, Hospedaje, Plantilla, Formulario, Página web, Html, Python, Formulario, Docker, Endpoint, PostrgreSQL, Página web de hotel, Reservas de habitaciones, Registro de usuarios, Login, Base de datos de clientes, Sistema de gestion de reservas, Interfaz de usuario, Diseño responsive, Funcionalidades, Desarrollo web, Backend, Frontend, API, Servicios web, Tecnologías utilizadas, Experiencia de usuario (UX), Pruebas y validacion, Implementacion, Despliegue, Escalabilidad, Rendimiento, Documentacion.

Introduccion: Este documento ahonda en las complejidades de la construccion e implementacion de una página web de un hotel, haciendo hincapié en el uso de numerosas herramientas de desarrollo web para mejorar la experiencia del usuario y al mismo tiempo mejorar la eficiencia operativa.

Soluciones Propuestas

Nuestra solucion se compone de dos aplicaciones desarrolladas en *Flask*: una dedicada al desarrollo de la *API* y otra orientada a los procesos relacionados con el *Frontend*. En el caso del *Frontend*, se diseño una interfaz clara y fácil de usar, con un índice que presenta de manera intuitiva los servicios ofrecidos por el hotel y las características de cada tipo de habitacion. Asimismo, incluye una barra de navegacion que permite al usuario acceder rápidamente a las diferentes vistas de la página, como el catálogo de habitaciones, un formulario de contacto y la funcionalidad de reserva.

Visualmente, la página se complementa con un pie de página que contiene la informacion de contacto del hotel. En cuanto a la gestion de los datos de los usuarios, se utilizo *AWS* como plataforma para el almacenamiento y administracion, organizando la informacion en cuatro tablas principales: usuarios registrados, hoteles, habitaciones(disponibles u ocupadas) y reservas realizadas. Estos datos son consumidos por la *API* y posteriormente presentados en el *Frontend*.

A continuacion, se presentan las *queries* utilizadas para la creacion y gestion de estas tablas:

Listing 1: Definicion de la tabla users

```
CREATE TABLE IF NOT EXISTS users (
    id_user INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(80) NOT NULL,
    lastname VARCHAR(80) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    dni INT(8) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
```

```
);
```

Listing 2: Definicion de las tablas habitaciones

```
CREATE TABLE IF NOT EXISTS type_rooms (
    id_room INT AUTO_INCREMENT PRIMARY KEY,
    id_hotel INT,
    type_room VARCHAR(80) NOT NULL,
    title VARCHAR(80) NOT NULL,
    description VARCHAR(500) NOT NULL,
    image VARCHAR(120),
    price FLOAT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_hotel) REFERENCES hotels (id_hotel)
);

CREATE TABLE IF NOT EXISTS rooms_disponibility (
    id_room INT NOT NULL,
    type_room VARCHAR(80) NOT NULL,
    number_room INT NOT NULL,
    floor_room INT NOT NULL,
    FOREIGN KEY (id_room) REFERENCES type_rooms(id_room)

);
```

Listing 3: Definicion de las tabla reservas

```
CREATE TABLE IF NOT EXISTS reservations (
    id_reservation INT AUTO_INCREMENT PRIMARY KEY,
    id_user INT NOT NULL,
    id_room INT,
    id_hotel INT,
```

```
    number_people INT,
    type_room INT,
    check_in TIMESTAMP,
    check_out TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_user) REFERENCES users (id_user),
    FOREIGN KEY (id_room) REFERENCES rooms_disponibility
        (id_room),
    FOREIGN KEY (id_hotel) REFERENCES hotels (id_hotel)
);
```

Listing 4: Definicion de las tabla hoteles

```
CREATE TABLE IF NOT EXISTS reservations (
    id_reservation INT AUTO_INCREMENT PRIMARY KEY,
    id_user INT NOT NULL,
    id_room INT,
    id_hotel INT,
    number_people INT,
    type_room INT,
    check_in TIMESTAMP,
    check_out TIMESTAMP,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (id_user) REFERENCES users (id_user),
    FOREIGN KEY (id_room) REFERENCES rooms_disponibility
        (id_room),
    FOREIGN KEY (id_hotel) REFERENCES hotels (id_hotel)
);
```

En terminos generales, el *Frontend* le solicita a la *API* la informacion requerida de la *BBDD* a través de solicitudes *HTTP* cargadas con datos

específicos en formato *JSON* en el cuerpo de la solicitud. Luego, la *API* consulta a la base de datos por medio de *queries* y realiza las verificaciones correspondientes. Por último, la *API* envía estos datos en formato *JSON* al *Frontend* para que sean utilizados en las vistas de la página.

Pruebas

Por un lado, se probo de forma independiente la *API*, mediante la aplicacion *Postman* o AWS, a eleccion de cada miembro del equipo, enviando solicitudes tanto correctas como en amplios rangos de invalidez. Por otro lado, la comunicacion entre la *API* y el *Frontend*, se baso en la utilizacion de la app como si se fuese un usuario, intentando producir *bugs*.

Plan de Actividades

Como primera medida, en la etapa inicial se diagramo colaborativamente la hoja de ruta y se realizo una division de tareas acorde a lo que requería el proyecto. Para esto se utilizaron metodologías ágiles como *Scrum* y *Kanban*, de forma que se pudiera maximizar la eficiencia a lo largo del ciclo de trabajo en grupo. Luego, con el objetivo de tener una estructura básica sobre la cual afrontar el proyecto, se eligio una plantilla.

Para comenzar se diseñaron las primeras vistas del proyecto, elementales para el funcionamiento del sitio web. Así, se crearon el **Menú principal** y la seccion **Habitaciones**. Funciono para comenzar a darle cuerpo al proyecto.

Como siguiente paso fue necesario conversar grupalmente para determinar un diseño para la base de datos. De esta manera, se establecio el criterio y orden con el que se llevaría a cabo. A raíz de esa discusion, se procedio a

crear los *Endpoints* de la *API* que se consideraron principales, siendo estos **Usuarios**, **Reservas** y **Habitaciones**.

La siguiente fase fue seguir desarrollando el *Frontend* para que la página tuviera un proposito. Se cambiaron distintias funcionalidades para hacer una experiencia mas agradable al usuario.

De la mano con lo que se hizo anteriormente, se desarrollo la contra parte de **Reservas** en el *API*. También, planificando hacia el futuro se desarrollo un login para llevar a cabo las mismas.

Anexos

Python documentation. — Disponible en: magentahttps://docs.python.org/es/3/library/hashlib.html.

Flask documentation. — Disponible en: magentahttps://flask.palletsprojects.com/en/stable/.