Aplicación web TravelApp

Ciclo 3 – Misión TIC 2022

Grupo P68_C3

Giraldo Erazo Natalia Andrea

Ramos Semanate Henry Fabian

Mazorra Medina Valentina

Fajardo Diaz Camilo Andrés

Abelardes Muñoz Yinna Yulieth

Docente:

Juan Camilo Castro Pinto

Universidad Nacional de Colombia

Facultad de ingeniería sede Bogotá

2021

TRAVEL APP BACK END

Categoría Escogida: Transporte

Problema Propuesto: Se solicita el desarrollo de una aplicación web, a la que se pueda ingresar en forma de Usuario y en forma de Administrador. Como administrador, la aplicación debe permitirle agregar, actualizar, eliminar y cancelar vuelos, además de ver las reservas existentes. Como usuario la plataforma debe permitirle filtrar y buscar vuelos, realizar reserva, ver ofertas de vuelos, realizar reserva, ver ofertas de vuelos, recomendar opciones de hospedaje en la ciudad destino y realizar reseñas o comentarios.

Scrum Máster: Natalia Giraldo

Diagrama relacional:

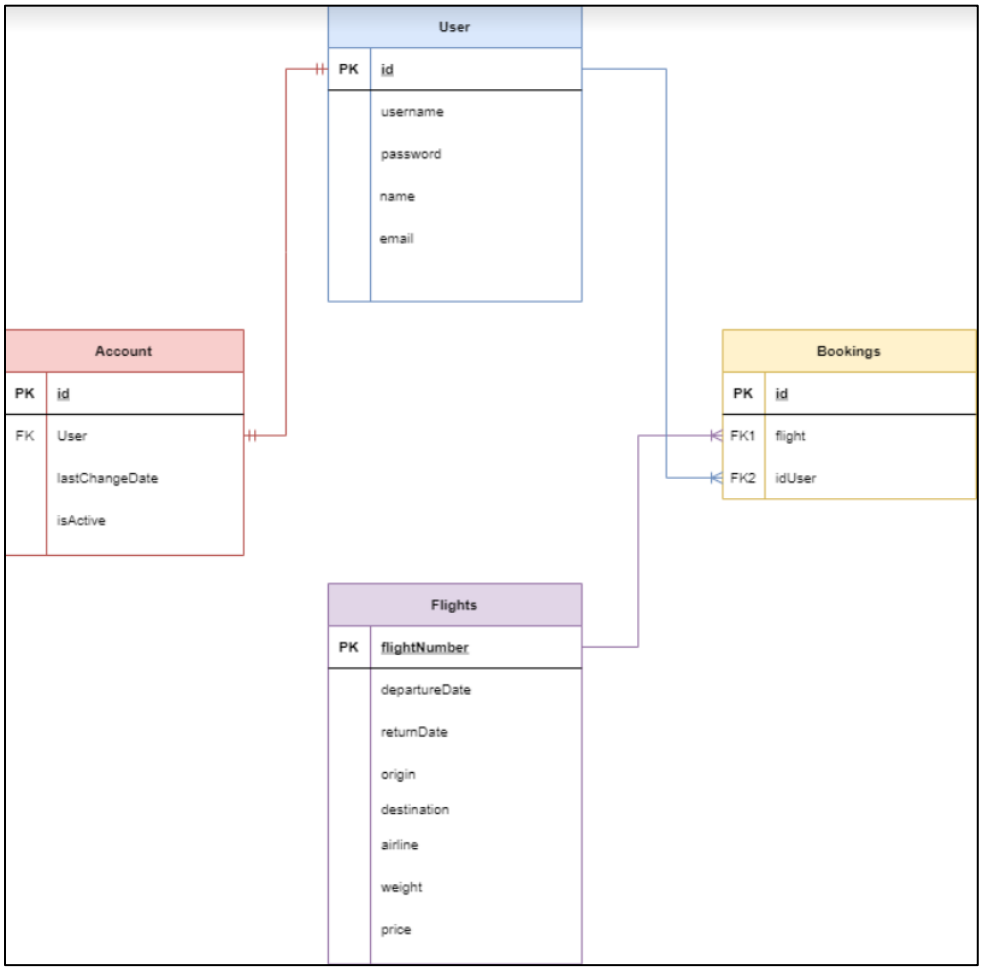

Ilustración 1 ***Diagrama relacional*** *travelApp*

HISTORIAS DE USUARIO

TABLA: USER

Modelo: User.py

1. El usuario debe poder registrarse en la aplicación.

1. No requiere autenticación.
2. Recepción de datos mediante una petición POST (en formato JSON).
3. Los datos se guardan en la base de datos en Heroku.
4. Generar los tokens correspondientes (Access y Refresh).
5. Método CREATE. (base de datos).
6. URL: https://be-travel-app.herokuapp.com/user

2. El usuario debe poder iniciar sesión en la aplicación (autenticación).

1. Se utiliza la clase Login que ofrece Django TokenObtainPairView.
2. URL: https://be-travel-app.herokuapp.com/login/

3.El usuario debe poder actualizar sus datos (username, name, email, password).

1. Actualización de datos mediante la petición PUT.
2. Requiere autenticación.
3. Decodificar el Access token para obtener el id del usuario autenticado.
4. Con el id del usuario autenticado, se actualizan los datos.
5. Respuesta con los datos correspondientes del usuario autenticado.
6. Método UPDATE.
7. URL: https://be-travel-app.herokuapp.com/user-view/

4.El usuario debe poder visualizar sus datos (id, username, email, password, account).

1. Requiere autenticación.
2. Se obtienen los datos del usuario autenticado, cuando el usuario lo requiera.
3. Petición GET.
4. Método READ.
5. URL: https://be-travel-app.herokuapp.com/user-view/

5.El usuario debe poder eliminar su cuenta.

1. Requiere autenticación.
2. Requiere el id del usuario en un archivo JSON.
3. Petición DELETE.
4. Método DELETE.
5. URL: https://be-travel-app.herokuapp.com/user-view/

TABLA: FLIGHTS

Modelo: Modelo: flights.py

6.El usuario debe poder ver todos los vuelos disponibles en la BD.

1. Método READ.
2. Petición GET
3. No requiere autenticación.
4. URL: https://be-travel-app.herokuapp.com/flightDetail

7.El usuario debe poder filtrar los vuelos por su id = flightNumber.

1. Método READ.
2. Petición GET.
3. No requiere autenticación.
4. URL: https://be-travel-app.herokuapp.com/flight/AVC2930/

8.El administrador debe poder crear un vuelo en la BD.

1. Petición POST.
2. Método CREATE.
3. Si el usuario es administrador, debe poder registrar el vuelo a la BD.
4. Requiere autenticación.
5. URL: https://be-travel-app.herokuapp.com/flightCreate

9.El administrador debe poder actualizar un vuelo de la BD.

1. Petición PUT.
2. Método UPDATE.
3. Si el usuario es administrador, debe poder actualizar un vuelo en la BD.
4. Requiere autenticación.
5. URL: https://be-travel-app.herokuapp.com/flight/AVC2930/

10. El administrador debe poder eliminar un vuelo de la BD.

1. Petición DELETE.
2. Método DELETE.
3. Si el usuario es administrador, debe poder eliminar un vuelo en la BD.
4. Requiere autenticación.
5. URL: https://be-travel-app.herokuapp.com/flight/AVC2930/

TABLA: BOOKINGS

Modelo: Bookings.py

11.El usuario debe poder reservar un vuelo por su id = flightNumber.

1. Requiere autenticación.
2. Recepción de datos mediante una petición POST (en formato JSON).
3. Se deben verificar los datos del usuario que va a realizar la reserva, mediante un archivo JSON con el flightNumber y el idUser.
4. Se comprueba que la reserva fue exitosa mediante un mensaje.
5. Método CREATE.
6. URL: https://be-travel-app.herokuapp.com/booking

12.El usuario debe poder ver las reservas que tiene por su id (de la reserva).

1. Requiere autenticación.
2. Petición GET.
3. Método READ.
4. URL: https://be-travel-app.herokuapp.com/booking/7/

13.El usuario debe poder eliminar una reserva por su pk = id.

1. Requiere autenticación.
2. Petición DELETE.
3. Método DELETE.
4. URL: https://be-travel-app.herokuapp.com/booking/2/

TABLA: ACCOUNT

14. El usuario debe poder tener una cuenta.

1. La cuenta se crea al momento que se registra un usuario.
2. Para acceder a la cuenta se requiere el Login.
3. URL: https://be-travel-app.herokuapp.com/login/

RELACIONES DE LAS TABLAS

TABLA USER – TABLA BOOKINGS

- **Relación uno a muchos:** Un usuario puede tener varias reservas, pero una reserva solo esta asignada a un usuario.

TABLA BOOKINGS- TABLA FLIGHTS

- **Relación uno a muchos:** Un vuelo puede tener varias reservas, pero una reserva solo esta asignada a un vuelo.

TABLA USER – TABLA ACCOUNT

- **Relación uno a uno:** Un usuario solo puede tener una cuenta y una cuenta solo tiene un usuario.

REQUERIMIENTOS PARA EL BACK-END:

```
django==3.2.7
djangorestframework==3.12.4
djangorestframework-simplejwt==4.8.0
psycopg2==2.9.1
gunicorn==20.1.0
django-heroku==0.3.1
```

REPOSITORIO BACK-END

1. Servicio de alojamiento web: Bitbucket.
2. Enlace repositorio: https://bitbucket.org/nataliager/travelapp/

HERRAMIENTA EN LINEA

1. Metodología ágil: Scrum.
2. Herramienta en línea: Jira.
3. Link Jira: https://nataliagiraldo.atlassian.net/jira/software/projects/TEAM/boards/1

SERVICIO EN LA NUBE

1. Plataforma usada: Heroku
2. Link app en Heroku: https://be-travel-app.herokuapp.com/

TRAVEL APP FRONT END

El front-end de la aplicación travelApp fue desarrollado mediante las siguientes tecnologías:

-Framework Vue.js

-HTML, CSS, JavaScript

-Servicio en la nube: Heroku

COMPONENTES DE LA APLICACIÓN:

- **Account:** Los usuarios pueden revisar y actualizar la información de su cuenta, si están logueados. Igualmente, el usuario puede ver sus reservas filtradas por id y borrar su cuenta.
- **Booking:** Los usuarios pueden buscar y eliminar reservas por id.

- **Flight:** El usuario puede revisar los vuelos de la base de datos sin necesidad de estar autenticado, y puede realizar una reserva, pero para dicha acción si se requiere autenticación.
  -Solo el administrador tiene los permisos para agregar, actualizar y eliminar vuelos del sistema.
- **FlightCreate:** Vista donde el administrados puede crear un vuelo mediante un formulario ofrecido por Vue.js.
- **FlightUpdate:** El administrador será redirigido a una página con un formulario mediante el cual podrá actualizar la información de un vuelo ya existente.
- **Home:** Componente donde se muestra información relevante de la empresa travelApp, puede visualizarse sin necesidad de autenticación.
- **LogIn:** Si el usuario ya está registrado, esta vista le permitirá iniciar sesión en la aplicación y luego, será redirigido al componente Account.
- **SignUp:** Una persona puede registrarse en la aplicación travelApp mediante el formulario ofrecido por este componente.
- **App:** Componente principal de la aplicación, contiene header y footer estáticos.

ROLES DE USUARIO

TravelApp cuenta con dos roles de usuario:

-Rol usuario: La vista le permite al usuario poder crear una cuenta, autenticarse, además ver vuelos en la plataforma y realizar, buscar y eliminar reservas. Igualmente manipular sus datos personales o eliminar su cuenta.

-Rol administrador: La vista le permite al administrador manipular los datos relacionados con los vuelos de la aplicación.

Cuenta administrador:

Usernarme: AdminTest4

Password: travelapp

SERVICIO EN LA NUBE

1. Plataforma usada: Heroku
2. Link app en Heroku: https://fe-travel-app.herokuapp.com/user/home

VISTA DESDE LA WEB TRAVELAPP

Ilustración 2 Componente: Home