

# Ejercicio Axede

**Nota: SE RECOMIENDA LEER EL ARCHIVO README.MD DEL REPOSITORIO PARA ENTENDER COMO INSTALAR LA APLICACIÓN Y ENTENDER ALGUNOS CONCEPTOS DE DISEÑO.**

## 1. Creación de Modelos de Entidad-Relación:

- En esta fase inicial del ejercicio, se plantean tres modelos que se relacionarán entre sí para establecer la estructura de la base de datos. Estos modelos son los siguientes:

### 1. Hotel:

- Esta entidad representa la ciudad donde estarán ubicadas las habitaciones y actuará como la tabla inicial que el usuario consultará. Dado que inicialmente hay un único hotel por ciudad, este se denominará de la misma manera que la ciudad (sujeto a cambios según el progreso de la prueba).

### 2. Habitación:

- Este modelo contendrá características como el tipo de habitación (estándar, premium, VIP), disponibilidad, la ciudad/hotel asociado y la capacidad máxima. Por defecto, la capacidad máxima se establece en 4 personas inicialmente, pero se va ajustando según la ciudad en el panel de Admin/Django.

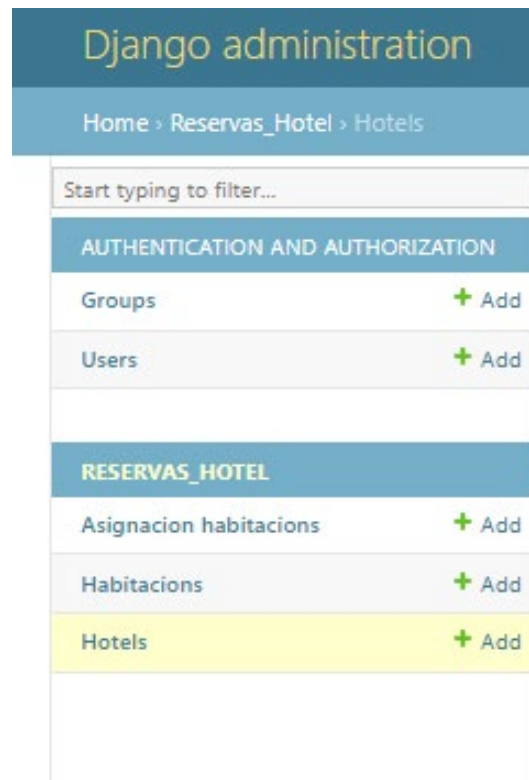
### 3. AsignacionHabitacion:

- Este modelo definirá cuántas habitaciones de diferentes características están asociadas a un hotel específico. Por ejemplo, indicará cuántas habitaciones estándar están asociadas a la ciudad de Bogotá, cuántas premium y VIP. Esta tabla servirá para realizar el conteo de las habitaciones que no estarán disponibles y se restarán al hotel.

## 2. Interfaz Administrativa (Panel /admin):

- Se presenta una interfaz administrativa, evidenciada en la imagen, donde se lleva a cabo la creación y gestión de los tres modelos mencionados. Desde este panel, los usuarios pueden realizar las siguientes acciones:
  - Crear nuevas ciudades.
  - Definir tipos de habitaciones.

- Asignar habitaciones a hoteles.

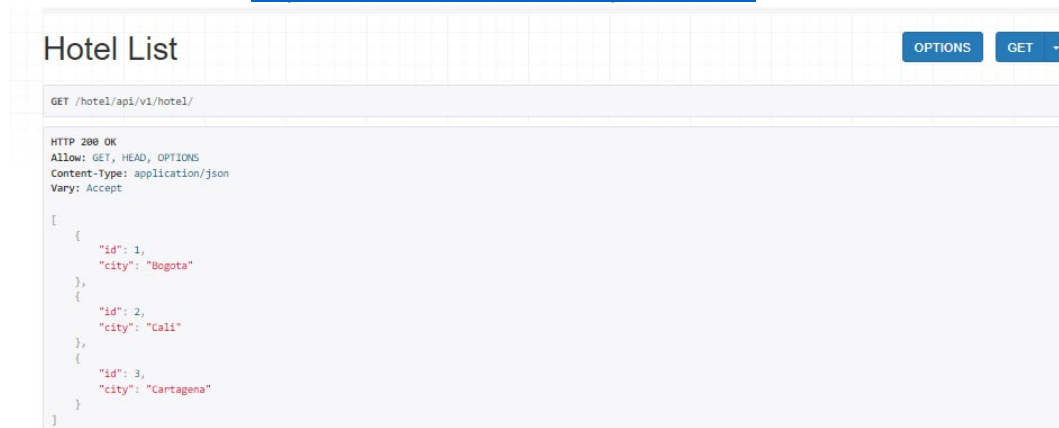


### 3. Creación del api de consumo

#### 3.1. Api para ciudades disponibles:

- La API se construye mediante el uso de un serializador. Esta API devuelve las ciudades creadas, que operarán como una lista para las opciones de ciudad en el front-end. Actualmente, solo está configurada para la operación GET.

**Función en la ruta:** <http://127.0.0.1:8000/hotel/api/v1/hotel/>



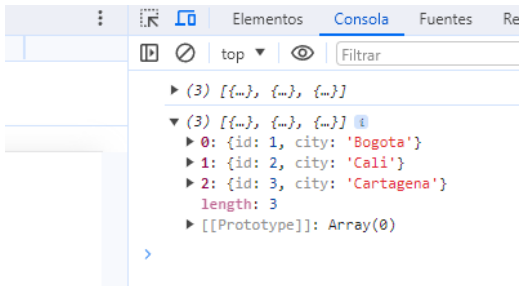
#### 4. Cliente

Se desarrolla un cliente para consumir la API desde el front-end, utilizando React con Vite. Axios se emplea para realizar las solicitudes de hoteles desde la base de datos. Esta información se integra en la barra de búsqueda, donde los hoteles y ciudades se presentan como opciones seleccionables.

```
//Obtenermos los hoteles disponibles de la BD
export const getAllHotels=()=>{
  return axios.get('http://127.0.0.1:8000/hotel/api/v1/hotel/')//direccion de la api
}
```

Se ejecuta de manera asíncrona una función que trae las ciudades/hoteles de la BD.

```
useEffect(()=>{
  async function loadHotels(){
    const res=getAllHotels();
    console.log(res)
  }
  loadHotels()
},[]);
```



```
(3) [{"id": 1, "city": "Bogota"}, {"id": 2, "city": "Cali"}, {"id": 3, "city": "Cartagena"}]
```

Y esa información es la que se usara como opción en la barra de busqueda .

## Bienvenido a Buscador de Hotel Axede

Buscar ciudad:

Bogota

Cali

Cartagena

Se incorporan dos campos adicionales para permitir al cliente especificar sus preferencias con respecto al tipo de habitación deseada y la cantidad de personas. Este ajuste tiene como objetivo proporcionar al cliente información detallada sobre las habitaciones disponibles según la ciudad y

su tarifa asociada. La información recopilada se utilizará para actualizar la base de datos, restándole las habitaciones correspondientes a la ciudad indicada. La

## Bienvenido a Buscador de Hotel Axede

Buscar ciudad:  Este campo es requerido  
Tipo de habitación:  Este campo es requerido  
Cantidad de personas:   
Fecha de inicio:  Este campo es requerido  
Fecha de finalización:  Este campo es requerido

Se agregan campos de inicio/fin de hospedaje para calcular la tarifa, La información solo se guardará cuando los campos estén llenos.

### • Segunda Api.

Se traerán las habitaciones asociadas a la ciudad y tipo que escoge el usuario, únicamente se mostrarán las habitaciones cuyo atributo disponible sea igual a True en el cliente, cada ciudad/hotel se identifica con un id único Bogotá 1, Cali 2, Cartagena 3 y en el campo de habitación viene el tipo de habitación y cuantas hay creadas (disponibles).

Eventualmente al escoger una habitación se enviara un POST para restar ese número al campo cantidad del modelo/tabla **AsignacionHabitacion**

Para probar el api se dirige a la ruta: **<http://127.0.0.1:8000/hotel/api/v1/habitacion/>**

```
[
  {
    "id": 1,
    "habitacion": {
      "id": 1,
      "tipo": "Standard",
      "disponible": true,
      "capacidad_maxima": 6
    },
    "cantidad": 20,
    "hotel": 1
  },
  {
    "id": 2,
    "habitacion": {
      "id": 2,
      "tipo": "Premium",
      "disponible": true,
      "capacidad_maxima": 6
    },
    "cantidad": 20,
    "hotel": 1
  },
  {
    "id": 3,
    "habitacion": {
      "id": 3,
      "tipo": "VIP",
      "disponible": true,
      "capacidad_maxima": 6
    },
    "cantidad": 2,
    "hotel": 1
  },
  {
    "id": 4,
    "habitacion": {
      "id": 4,
      "tipo": "Standard",
      "disponible": true,
      "capacidad_maxima": 6
    },
    "cantidad": 20,
    "hotel": 2
  }
]
```

**SE RECOMIENDA LEER EL ARCHIVO README.MD DEL  
REPOSITORIO PARA ENTENDER COMO INSTALAR LA APLICACIÓN Y  
ENTENDER ALGUNOS CONCEPTOS DE DISEÑO.**