falabella.

Prueba Técnica Software Engineer

Nombre: Luis Felipe Velasquez Puentes

1. Introducción

En el siguiente documento se evidencia la metodología que se desarrolló en la creación de una Api Rest HTTP para la administración de la reservación de habitaciones y hoteles. Se va a presentar las tecnologías que se utilizaron para el código y las bases de datos, como se desarrolló, como se utiliza y diferentes pruebas de escritorio para la demostración de que la Api funciona correctamente

2. Desarrollo

A continuación, se muestra el procedimiento que se utilizó para la creación de la API, el link del repositorio GIT es CLICK AQUI

2.1 Funcionalidades de la API

Las siguientes son las funciones que la API debe poder desarrollar:

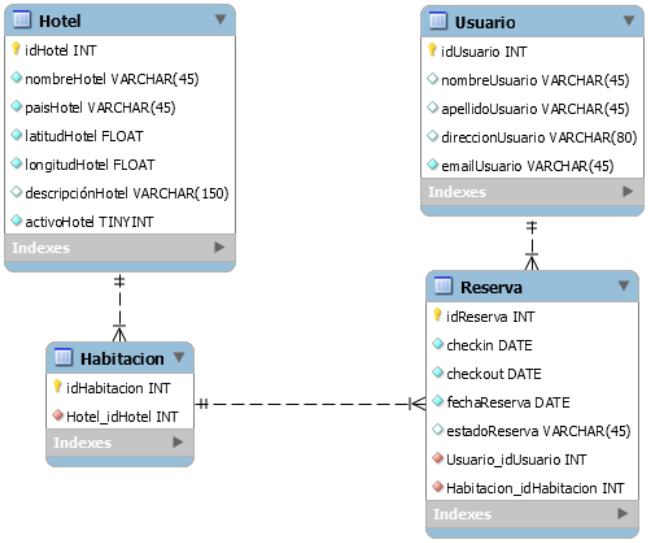
- Registrar una reserva dados un usuario, un hotel y un rango de fechas (checkin y checkout).
- Cancelar una reserva dado su identificador (no se eliminará, se marcará como cancelada y dejará de "ocupar" la habitación del hotel para los días de la reserva).
- Consultar todas las reservas activas de un hotel que estén dentro de un rango de fechas (al menos 1 noche de la reserva debe estar comprendida entre las fechas solicitadas). La respuesta debe contener, además de la información de cada reserva, el nombre del hotel y el mail del usuario asociado a la reserva.

2.2 Tecnologías

Las siguientes son las tecnologías y frameworks que se utilizó para la creación de la API Rest

- MySql como gestor de bases de datos
- Python como lenguaje de programación, utilizando Modelo Vista Controlador como marco para crear la API
- Flask como Framework para el manejo de la API y que pueda empaquetar, recibir y retornar peticiones HTPP

2.3 Creación de Bases de Datos



Basados en esto se creó la base de datos en MySQL workbench, donde además se ingresaron datos iniciales para las entidades de usuario, hotel y habitación para poder realizar pruebas sobre la API

2.4 Creación de la API

2.4.1 Definición de las rutas:

El primer paso fue definir las rutas,inputs y outputs que va a tener la API para la administración de las reservas, las cuales son las siguientes:

- **\crearReserva**

  - Inputs: {
    “idUsuario”:”7567567” ,
    ”idHotel”:5 ,
    ”checkin”: “2022-11-3” ,
    ”checkout”: “2022-11-6”
    }

  - Outputs:{

    “dataProperties”:”success” ,

    “reserva”:{
    “idHabitacion”:8,
    “idReserva”:21
    }

- **\cancelarReserva**

  - Inputs: {
    “idReserva”:21
    }

  - Outputs:{

    “dataProperties”:”success” ,

    “reserva”:{
    “idReserva”:21
    }

- **\consultarReservas**

  - Inputs: {
    ”idHotel”:5 ,
    ”checkin”: “2022-11-3” ,
    ”checkout”: “2022-11-6”
    }

- Outputs:{

“dataProperties”:”success” ,

“reservas”:{
“idHabitacion”:8,
“idReserva”:21,
”checkin”: “2022-11-3” ,
”checkout”: “2022-11-6” ,
“emailUsuario”:”felipe@gamil.com”,
“fechaReserva”:”2022-10-12”,
“nombreHotel“: “Blue Suites”
}

2.4.2 Creación de los modelos y conexiones

- **Conexiones:** Se desarrollo el código que fuese capaz de conextarse a la base de datos en MySql y realizara todas las acciones que puede hacer el gestor de la base de datos (select, insert, update, drop., etc..)

- **Modelos:** En Python por cada Entidad se creo un modelo, de tal manera que se utilizara Programación Orientada a Objetos para realizar las acciones de la API, es decir que se crearon las siguientes clases
  - **Reserva:** Crea, cancela y consulta reservas
  - **Hotel:** Revisa si un hotel existe en la base de datos
  - **Habitacion:** Busca una habitacion que este disponible en un rango de fechas
  - **Usuario:** Revisa si un Usuario existe en la base de datos

2.4.3 Empaquetamiento de la API:

Se procedió a empaquetar la AIP de tal manera que funcionara como un APP, para esto se diseño la estructura que solicita Flask, el cual es un archivo main, uno de arranque, uno de configuración y un __init__ en el controlador

2.4.4 Creación del controlador:

Por ultimo se codifico el controlador de las reservas, el cual este encargado de definir las rutas de la API, recibir las peticiones y retornar los outputs de cada uno de ellos, para esto, la petición llegaba este código, y este lo redireccionaba a cada uno de los modelos utilizando POO según el caso y retornaba el output en base a la acción que se requiera. También es importante señalar que manejaba las excepciones tales como si el checkout es menor al checkin o si un usuario no existe, etc..

3. Pruebas de Escritorio

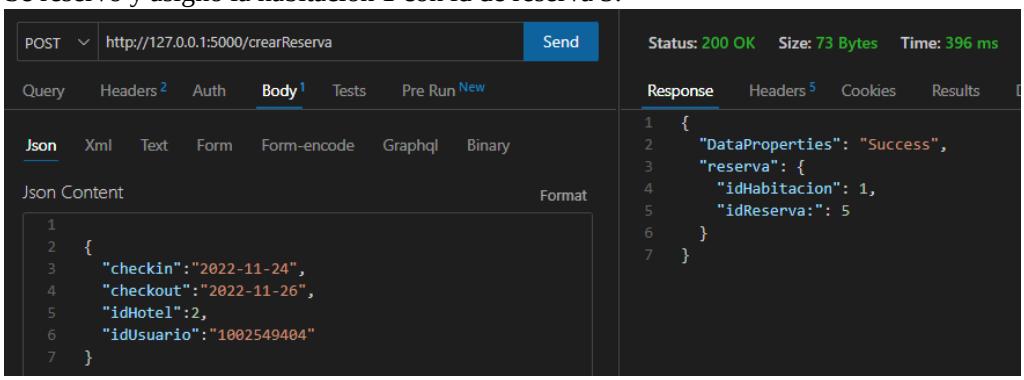
Antes de realizar pruebas, se ejecuto la API, este procedimiento se encuentra en el README del repositorio.

Por último, se realizaron test unitarios para la verificación del funcionamiento de la API mediante una aplicación que permite hacer pruebas de peticiones, tal como lo es Postman, asi se muestra a continuación:

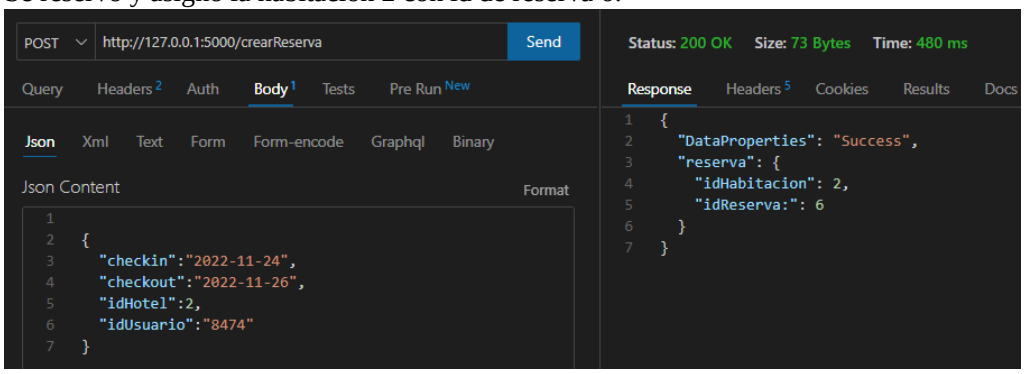
Primero, se debe tener en cuenta las reservas iniciales que se tiene en la base de datos:

idReserva	checkin	checkout	fechaReserva	estadoReserva	Usuario_idUsuario	Habitacion_idHabitacion
1	2022-11-11	2022-11-15	2022-10-28	reservado	1002549404	3
2	2022-11-13	2022-11-14	2022-10-28	reservado	1002549404	4
3	2022-11-19	2022-11-24	2022-10-28	reservado	1002549404	5
4	2022-11-28	2022-11-30	2022-10-28	reservado	1002549404	5

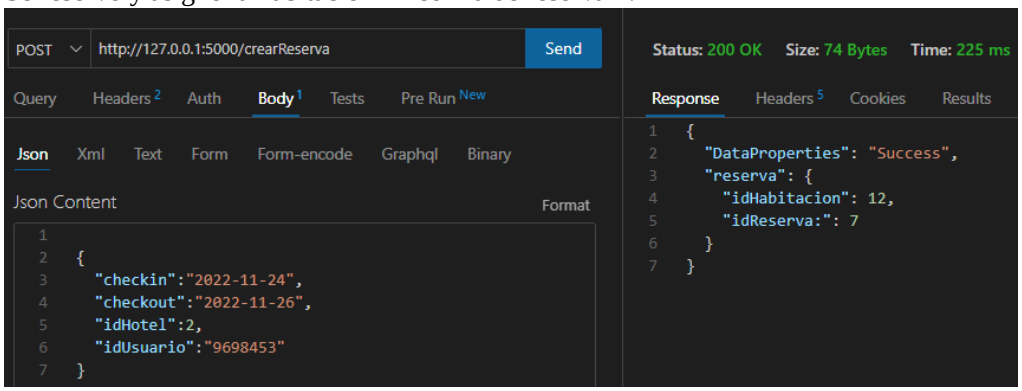
- **Pruebas para crear reservas**
  Para esta prueba se vana registrar reservas en las mismas fechas para el hotel 2 de tal manera de que se llene el hotel y no permita reservar más, también se podrá visualizar como por cada petición va a ir asignando una habitación libre diferente porque se van ocupando:
  Se reservo y asigno la habitación 1 con id de reserva 5:



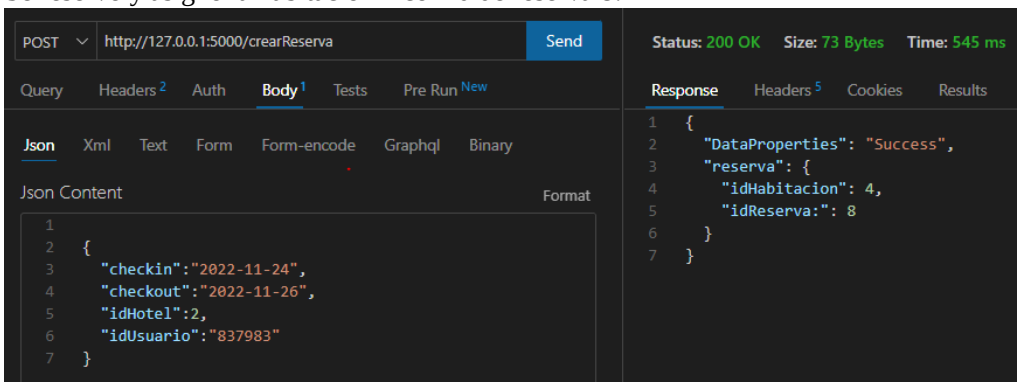
  Se reservo y asigno la habitación 2 con id de reserva 6:



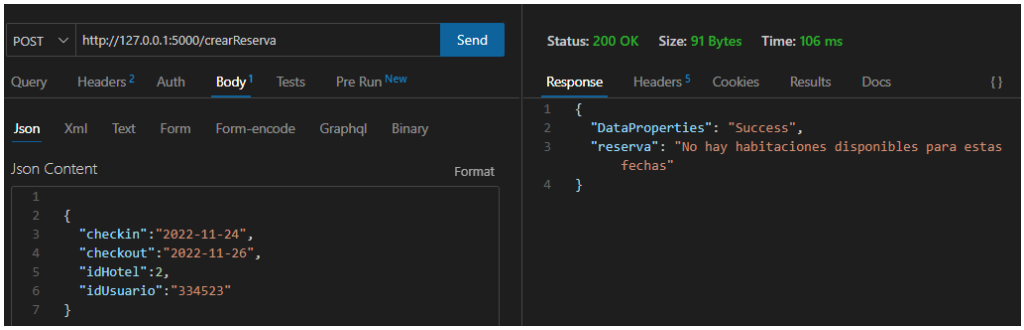
  Se reservo y asigno la habitación 12 con id de reserva 7:



Se reservo y asigno la habitación 4 con id de reserva 8:



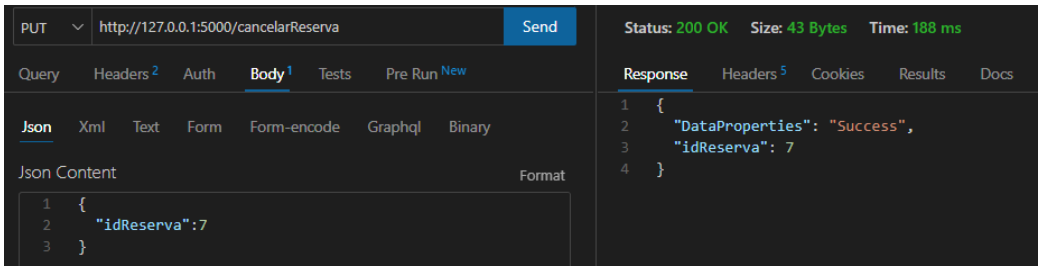
No se pudo reservar porque para esas fechas ya se llenaron todas las habitaciones del hotel:



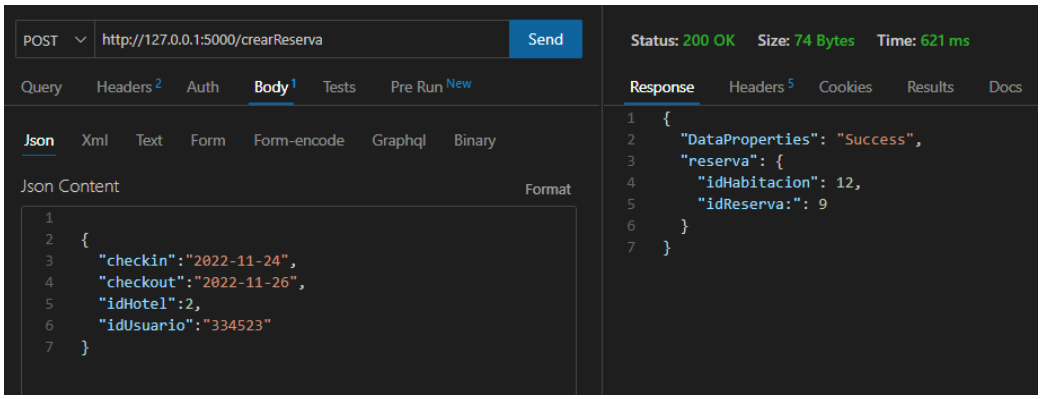
Resultado final de la base de datos luego de registrar las nuevas reservas:

idReserva	checkin	checkout	fechaReserva	estadoReserva	Usuario_idUsuario	Habitacion_idHabitacion
1	2022-11-11	2022-11-15	2022-10-28	reservado	1002549404	3
2	2022-11-13	2022-11-14	2022-10-28	reservado	1002549404	4
3	2022-11-19	2022-11-24	2022-10-28	reservado	1002549404	5
4	2022-11-28	2022-11-30	2022-10-28	reservado	1002549404	5
5	2022-11-24	2022-11-26	2022-10-28	reservado	1002549404	1
6	2022-11-24	2022-11-26	2022-10-28	reservado	8474	2
7	2022-11-24	2022-11-26	2022-10-28	reservado	9698453	12
8	2022-11-24	2022-11-26	2022-10-28	reservado	837983	4

- **Pruebas para cancelar reservas:**
  Para probar la cancelación de una reserva, se va a cancelar una reserva que se hizo en el paso anterior y se intentara a agregar la última que no permitió desarrollar porque ya estaban ocupadas todas las habitaciones
  Se cancela la reserva numero 7:

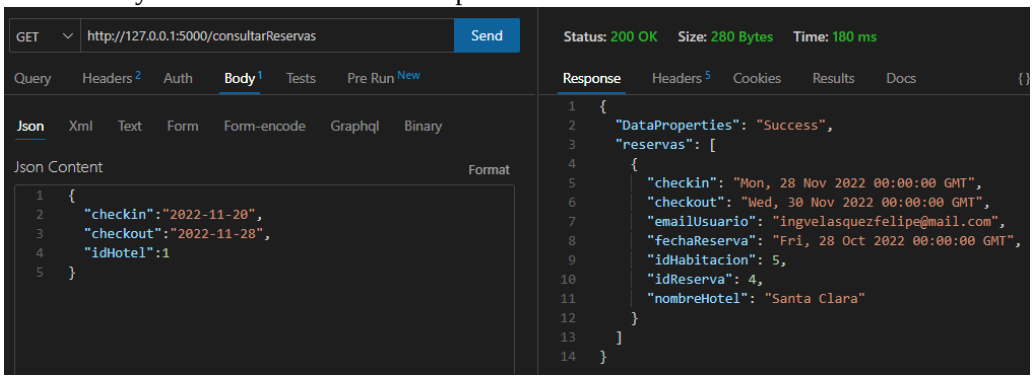


Ahora, se intenta registrar la reserva que no permitió en el paso anterior porque el hotel estaba lleno en esas fechas y efectivamente la habitación se libero y se registro correctamente:

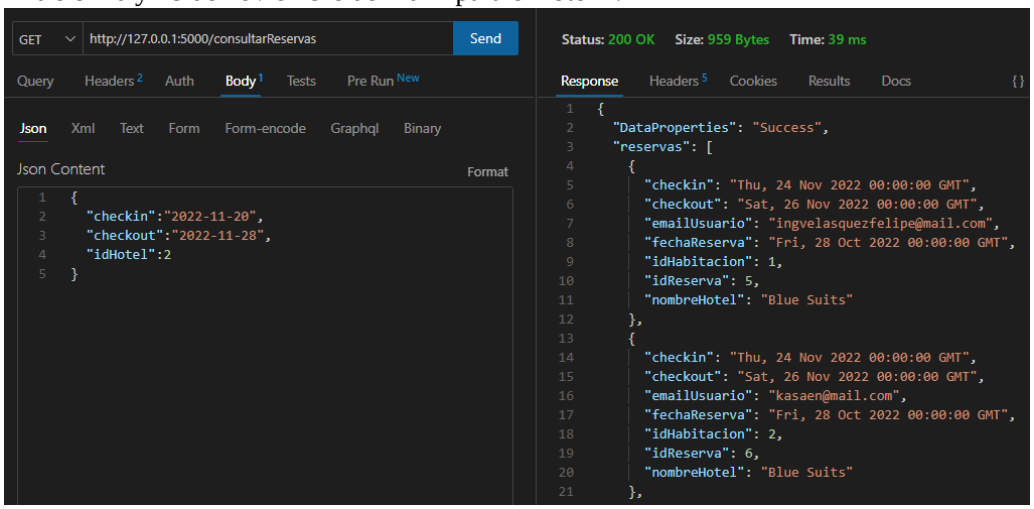


- **Pruebas para consultar reservas:**
  A continuación, se va a probar la consulta de pruebas en un rango de fechas y se podrá visualizar como retorna los datos de las reservas, el hotel y el usuario:

  Entre el 20 y 28 de noviembre del 2022 para el hotel 1:

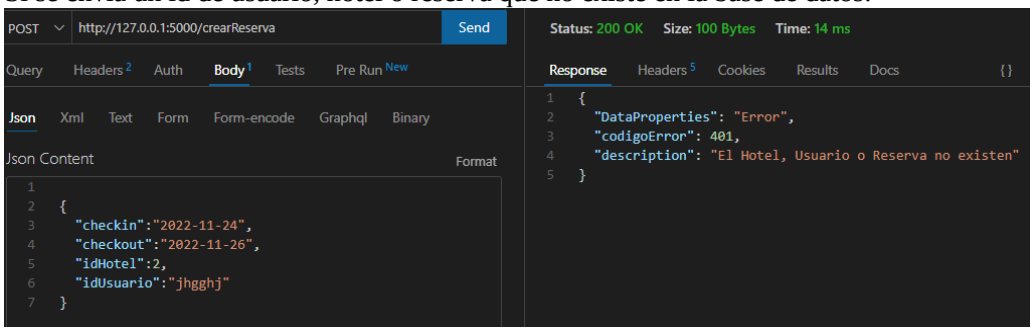


  Entre el 20 y 28 de noviembre del 2022 para el hotel 2:

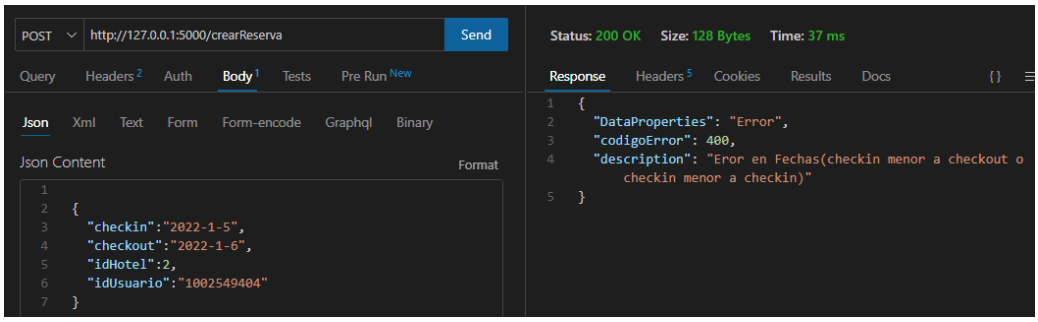


- **Pruebas para excepciones:**
  Por último, se va a probar las excepciones desarrolladas para las peticiones en caso de diferentes errores. Para este caso se intentará con crear reserva, sin embargo, cancelar y consultar reserva también cuenta con ellas:

  Si se envía un id de usuario, hotel o reserva que no existe en la base de datos:



  Si la fecha de reserva deseada es menor a la fecha actual:



  Si el checkin es mayor a el checkout :