

M07_UF4_A9	Diseño de aplicaciones web
------------	----------------------------

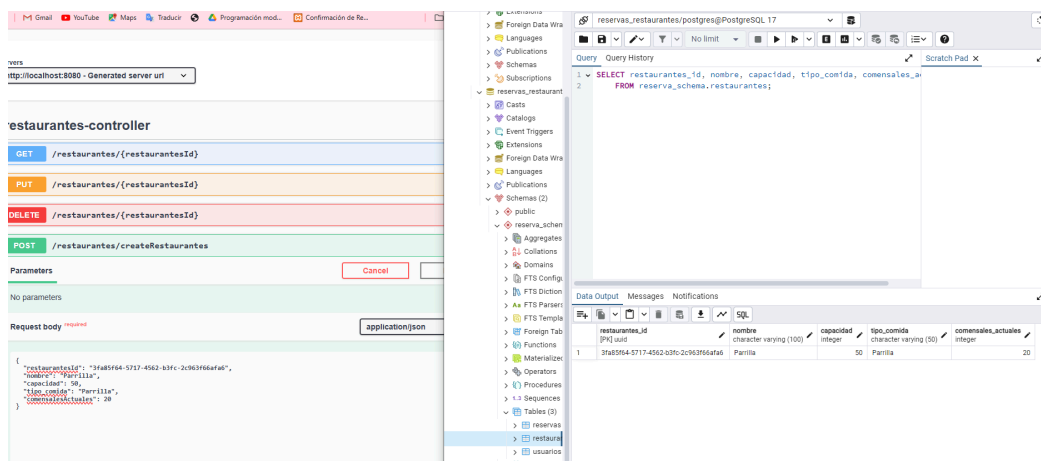
Actividad 9. Spring

Inicio	Entrega	Calificación	Ponderación
06 noviembre 2024	05 diciembre 2024	12 diciembre 2024	40 %

Enunciado:

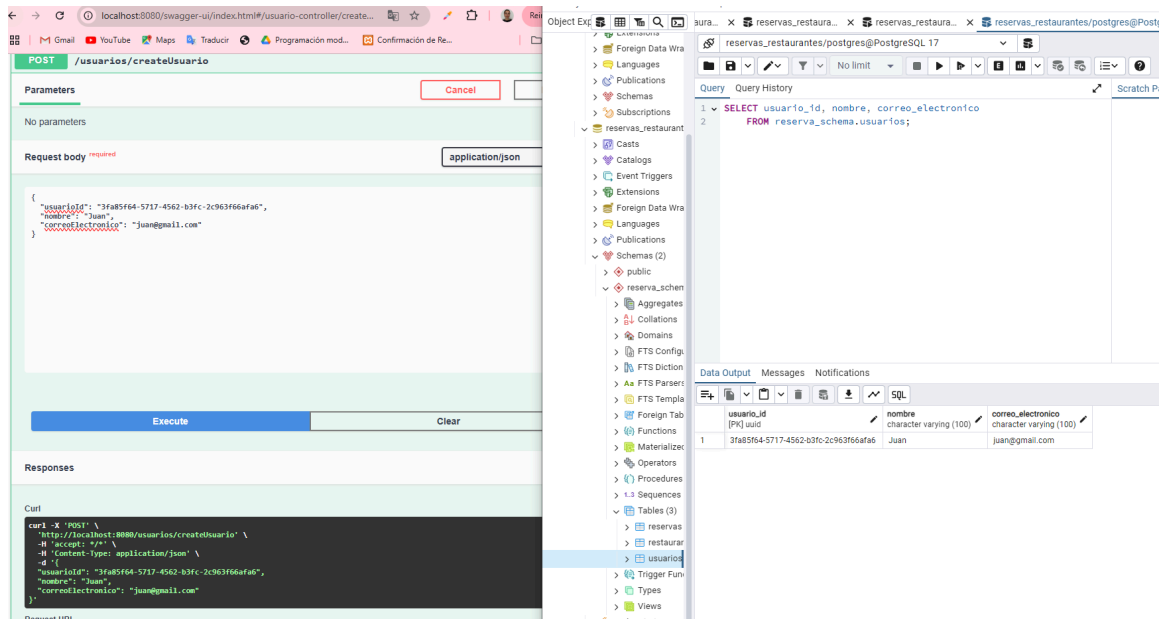
Durante este curso vamos a trabajar una aplicación web en Spring para hacer reservas en restaurantes. Para ello vamos a programar un backend que:

- Permite guardar restaurantes en Base de datos:
 - Nombre
 - Capacidad
 - Tipo de comida
 - Comensales actuales (se inicializará a 0 siempre que se cree de nuevo o ya exista el restaurante en la aplicación)



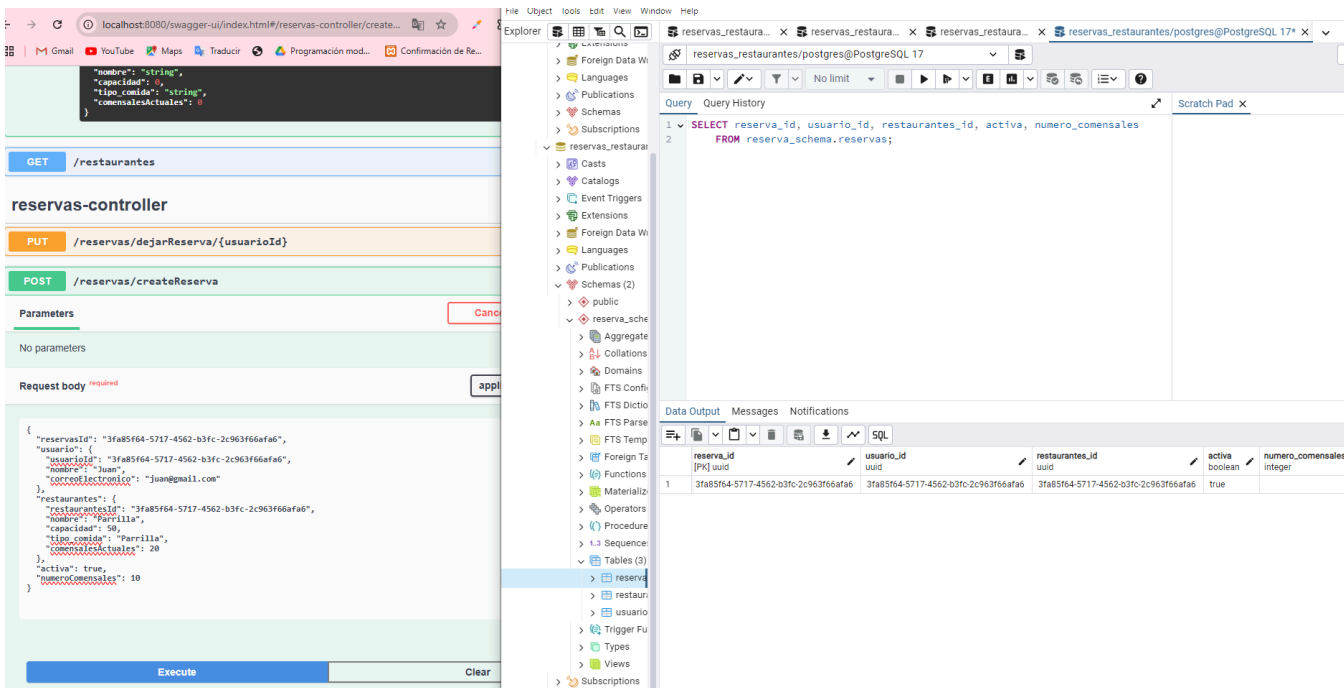
- Permite guardar usuarios
 - Nombre
 - Correo electrónico

Actividad de evaluación



The screenshot shows two windows. On the left, the Swagger UI for a REST API. The endpoint `POST /usuarios/createUsuario` is selected. The request body is a JSON object: `{ "usuarioId": "3fa85f64-5717-4562-b3fc-2c963f66afa6", "nombre": "Juan", "correoElectronico": "juan@gmail.com" }`. The response is a 201 status code. On the right, the PostgreSQL interface shows a query: `SELECT usuario_id, nombre, correo_electronico FROM reserva_schema.usuarios;` with one result row: `usuario_id | nombre | correo_electronico` | `3fa85f64-5717-4562-b3fc-2c963f66afa6 | Juan | juang@gmail.com`.

- Permite guardar reservas que serán las que tengan la relación entre usuarios y restaurantes. Se guardará también si está activa o no y el número de comensales.

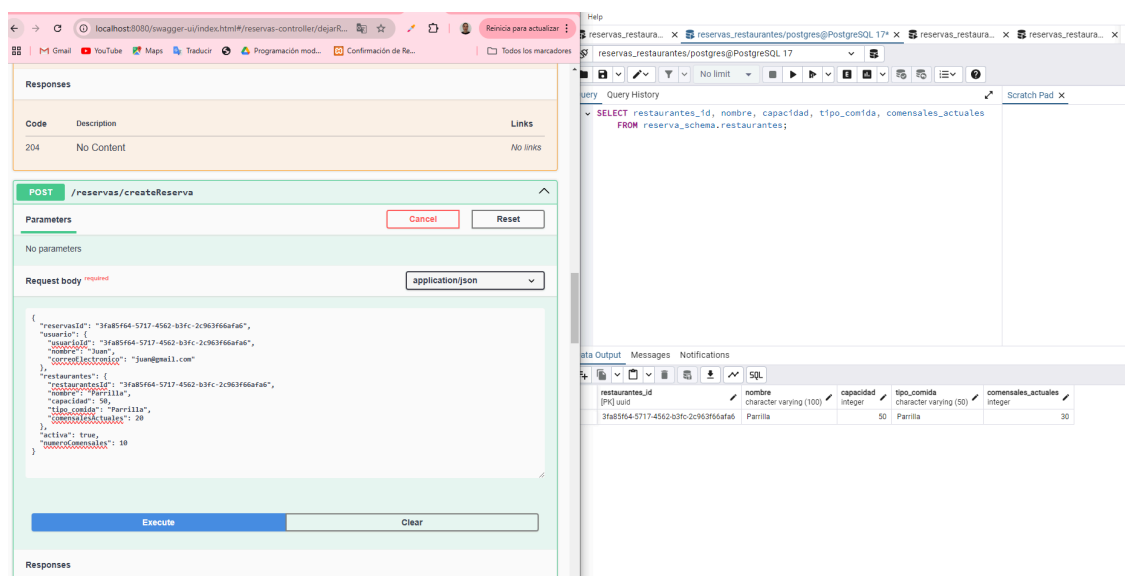


The screenshot shows two windows. On the left, the Swagger UI for a REST API. The endpoint `POST /reservas/createReserva` is selected. The request body is a JSON object: `{ "reservasId": "3fa85f64-5717-4562-b3fc-2c963f66afa6", "usuario": { "usuarioId": "3fa85f64-5717-4562-b3fc-2c963f66afa6", "nombre": "Juan", "correoElectronico": "juan@gmail.com" }, "restaurantes": { "restaurantesId": "3fa85f64-5717-4562-b3fc-2c963f66afa6", "nombre": "Parrilla", "capacidad": 50, "tipo_comida": "Parrilla", "comensalesActuales": 20 }, "activa": true, "numeroComensales": 10 }`. The response is a 201 status code. On the right, the PostgreSQL interface shows a query: `SELECT reserva_id, usuario_id, restaurantes_id, activa, numero_comensales FROM reserva_schema.reservas;` with one result row: `reserva_id | usuario_id | restaurantes_id | activa | numero_comensales` | `3fa85f64-5717-4562-b3fc-2c963f66afa6 | 3fa85f64-5717-4562-b3fc-2c963f66afa6 | 3fa85f64-5717-4562-b3fc-2c963f66afa6 | true | 10`.

- Exponga una API restaurante que:
 - Crear un restaurante
 - Listar todos los restaurantes
 - Listar un restaurante por id
 - Modificar un restaurante (sólo podrás modificar la capacidad y el tipo de comida)
 - Eliminar un restaurante
- Exponer una API de usuarios:
 - Crear usuario

Actividad de evaluación

- Listar un usuario por id
- Listar todos los usuarios
- Eliminar un usuario por id
- Exponer una API de reservas:
 - Hacer una reserva. Dado un usuario, un número de comensales y un restaurante hacer una reserva. Esta reserva va a mirar primero si el restaurante y el usuario existen, si es así se mira también si el número de comensales caben en la capacidad del restaurante.
 - Si los comensales caben, se actualiza la capacidad actual del restaurante y crea una nueva reserva.
 - Si los comensales no caben, se devuelve un error.



The screenshot displays two overlapping windows. The foreground window is the Swagger UI for a REST API, showing a POST endpoint at `/reservas/createReserva`. The 'Request body' is set to `application/json` and contains a JSON object with reservation details. The background window is a PostgreSQL database interface showing a SQL query that selects restaurant information based on the reservation ID.

Swagger UI Details:

- Endpoint:** `POST /reservas/createReserva`
- Request body (JSON):**

```
{
  "reservaId": "3fa85f64-5717-4562-b3fc-2c963f66af66",
  "usuario": {
    "usuarioId": "3fa85f64-5717-4562-b3fc-2c963f66af66",
    "nombre": "Juan",
    "email": "juan@gmail.com"
  },
  "restaurantes": {
    "restaurantesId": "3fa85f64-5717-4562-b3fc-2c963f66af66",
    "nombre": "Parrilla",
    "capacidad": 50,
    "tipo_comida": "Parrilla",
    "comensales_actuales": 10
  },
  "activo": true,
  "comensales_actuales": 10
}
```

PostgreSQL Query and Results:

```
SELECT restaurantes_id, nombre, capacidad, tipo_comida, comensales_actuales
FROM reserva_schema.restaurantes;
```

restaurantes_id	nombre	capacidad	tipo_comida	comensales_actuales
3fa85f64-5717-4562-b3fc-2c963f66af66	Parrilla	50	Parrilla	30

- Dejar reserva. Dado un usuario se mira si tiene reservas activas.
 - Si tiene reservas activas se van a poner a no activas y se van a restar los comensales actuales a los restaurantes.

[illegible]

Recordad que todo esto debes de validar que también se actualiza en la Base de datos.

Formato de entrega:

Se entrega en PDF

Contenidos y recursos didácticos:

Podéis usar el libro y recursos de la plataforma.