

A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

12-10-2021

Prueba Técnica

Logitravel

Several thin, curved lines in dark blue and light grey originate from the bottom left corner and sweep upwards and to the right.

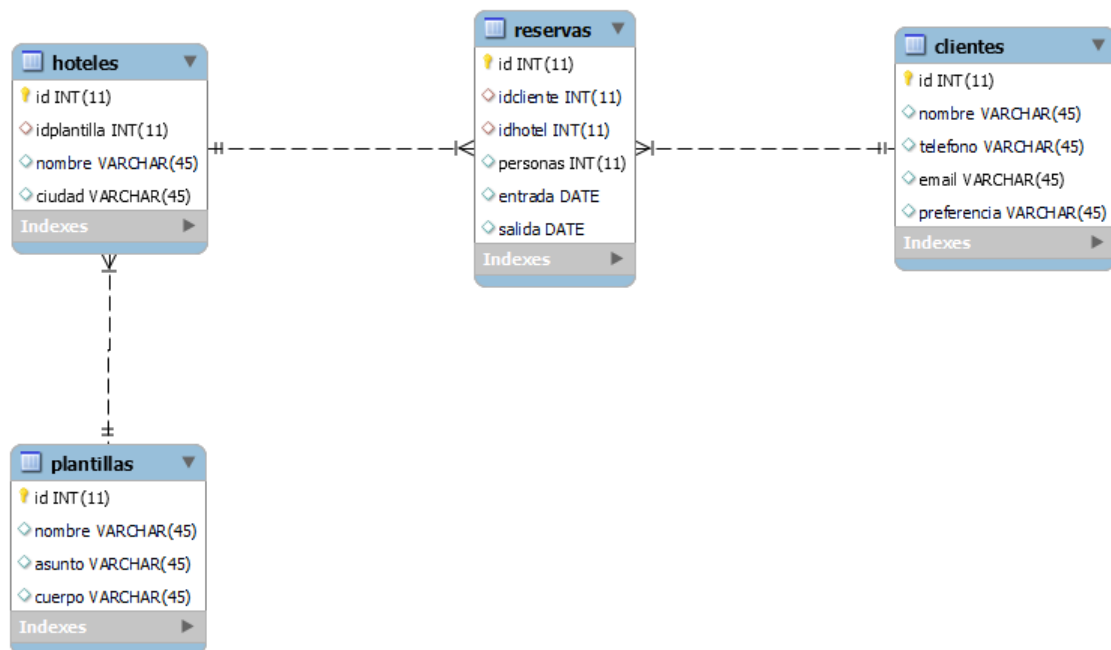
RICARDO JUDEL ALEMANY

INTRODUCCIÓN

Para realizar esta prueba técnica me he basado en los Web Services que utilizamos en mi empresa actual. Esto me ha permitido montarlo en un sistema en producción. El sistema está formado por:

- 2 máquinas virtuales Debian de Google Cloud Platform dentro de la misma red local
- Servidor Tomcat con Web Service Java, haciendo funciones de API Rest, desplegado en una máquina virtual
- Base de datos MySQL desplegada en la otra máquina virtual
- Para el desarrollo se utilizará el IDE Apache Netbeans y para realizar las pruebas la herramienta Postman.

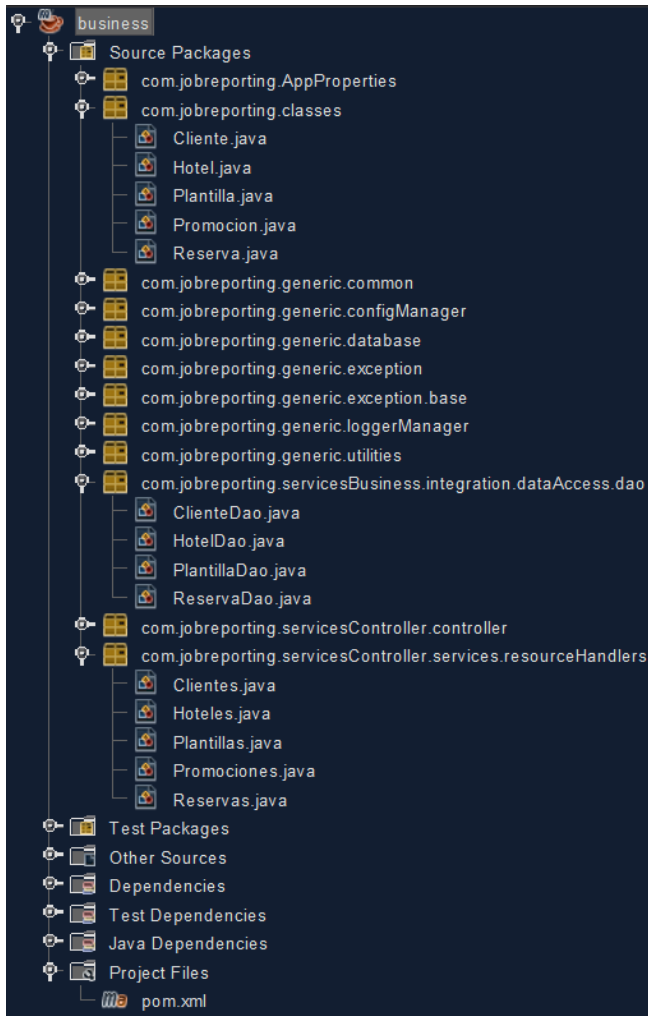
BASE DE DATOS



Esta es la base de datos relacional que utilizaré para la prueba. Por simplificar, solo he creado algunos campos básicos, pero disponer de una base de datos nos proporcionará escalabilidad. Puntos a destacar:

- En la tabla **hoteles**, cada hotel tiene asignada una plantilla de mensaje, definida por clave foránea.
- En la tabla **reservas**, cada reserva tendrá asignado un cliente y un hotel, mediante clave foránea.
- En la tabla **clientes**, se ha definido el campo “preferencia” que nos permitirá guardar la preferencia de método de envío de promociones de cada cliente.

ESTRUCTURA DEL PROYECTO



Puntos a destacar:

- **Classes:** clases que gestionan los objetos usados por los servicios del API Rest.
- **Database:** Gestor de conexiones a la base de datos.
- **Exception – Base:** Contiene los métodos para lanzar excepciones.
- **LoggerManager:** Contiene los métodos para la creación y escritura del log.
- **DAO:** clases para interactuar con la base de datos fuera de la capa de negocio.
- **ResourceHandlers:** clases que gestionan los servicios del API Rest.
- **Pom.xml:** archivo donde están declaradas todas las dependencias del proyecto. El gestor de dependencias utilizado es Maven.

PROBLEMÁTICA

Logitravel desea enviar a sus clientes un e-mail o sms, según los datos y preferencia de cada cliente, promocionando hoteles de la misma zona a los que ha contratado. Cada cliente ha podido reservar varios hoteles y cada tipo de hotel requiere una plantilla de mensaje diferente.

Para resolver el problema, primero se ha planteado cómo poder recomendar hoteles cercanos a otros hoteles ya reservados. Por simplificar, utilizaré el atributo “ciudad” de cada hotel para poder definir la cercanía entre ellos. Para un caso real se podría añadir un atributo de código postal o incluso coordenadas para ser más precisos.

En la tabla de clientes, habrá un atributo “preferencia” que permitirá establecer la preferencia de envío de cada cliente, ya sea email o sms.

Para que cada tipo de hotel pueda tener una plantilla de mensaje diferente, en la tabla de hoteles hay un atributo referenciando a la plantilla asignada. Cada plantilla podría ser creada para un determinado tipo de hotel, por ejemplo “Plantilla Only Adults”. De esta manera, es muy fácil añadir nuevas plantillas.

Referente a los servicios, para clientes, hoteles, plantillas y reservas, solo se crea un servicio para listar la información de la base de datos, pero partiendo de esta base, es muy sencillo crear nuevos servicios.

El programa se centra en el servicio de enviar promociones, donde facilitando el nombre del cliente deseado, el algoritmo comenzará por obtener su preferencia de método de envío de promociones y las reservas realizadas, en el caso de que existan. Después, de cada reserva se obtiene la ciudad del hotel y se realiza una búsqueda de otros hoteles de la misma ciudad. Por último, se guarda toda la información en un JSON con el que se podrá gestionar el envío de la promoción.

SERVICIOS

- Clase: Clientes
 - Nombre: Listar
 - Descripción: Devuelve el listado de clientes
 - URL: <http://tomcat.9sistemas.com/wspruebalogitravel/services/Clientes/Listar>
 - Método: Post
 - Consume: Nada
 - Produce: JSON
-
- Clase: Hoteles
 - Nombre: Listar
 - Descripción: Devuelve el listado de hoteles
 - URL: <http://tomcat.9sistemas.com/wspruebalogitravel/services/Hoteles/Listar>
 - Método: Post
 - Consume: Nada
 - Produce: JSON
-
- Clase: Plantillas
 - Nombre: Listar
 - Descripción: Devuelve el listado de plantillas
 - URL: <http://tomcat.9sistemas.com/wspruebalogitravel/services/Plantillas/Listar>
 - Método: Post
 - Consume: Nada
 - Produce: JSON
-
- Clase: Reservas
 - Nombre: Listar
 - Descripción: Devuelve el listado de reservas
 - URL: <http://tomcat.9sistemas.com/wspruebalogitravel/services/Reservas/Listar>
 - Método: Post
 - Consume: Nada
 - Produce: JSON
-
- Clase: Promociones
 - Nombre: Enviar
 - Descripción: Devuelve las promociones para un cliente determinado
 - URL: <http://tomcat.9sistemas.com/wspruebalogitravel/services/Promociones/Enviar>
 - Método: Post
 - Consume: JSON
 - Produce: JSON

Ejemplo de respuesta del servicio promociones:


POST ▼ <http://tomcat.9sistemas.com/wspruebalogitravel/services/Promociones/Enviar>

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   ... "nombre" : "Miguel Perez"
3 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize **JSON** ▼ 

```
1 {
2   "msg": "Success Operation.",
3   "result": [
4     {
5       "nombrecliente": "Miguel Perez",
6       "metodoenvio": "SMS"
7     },
8     [
9       {
10        "Ciudad": "Inca",
11        "Hotel Ya Reservado": "La Bodega"
12      },
13      [
14        {
15          "Asunto": "Promocion 3",
16          "Cuerpo": "Promocionando 3",
17          "Nombre Hotel": "Son Vivot"
18        }
19      ]
20    ]
21  }
```