Programación 2 > módulo 3

>Tecnicatura Universitaria en Desarrollo de Software

UPATECO UNIVERSIDAD PROVINCIAL
ADMINISTRACIÓN • TECNOLOGÍA • OFICIOS

Ministerio de Economía
y Servicios Públicos
Gobierno de Salta

academia
CIMNE IBER

módulo 3

Tecnicatura Universitaria en Desarrollo de Software

Programación 2

Módulo 3: Backend

Tema 4: Acceso a base de datos y patrón MVC

UPATECO UNIVERSIDAD PROVINCIAL
ADMINISTRACIÓN • TECNOLOGÍA • OFICIOS

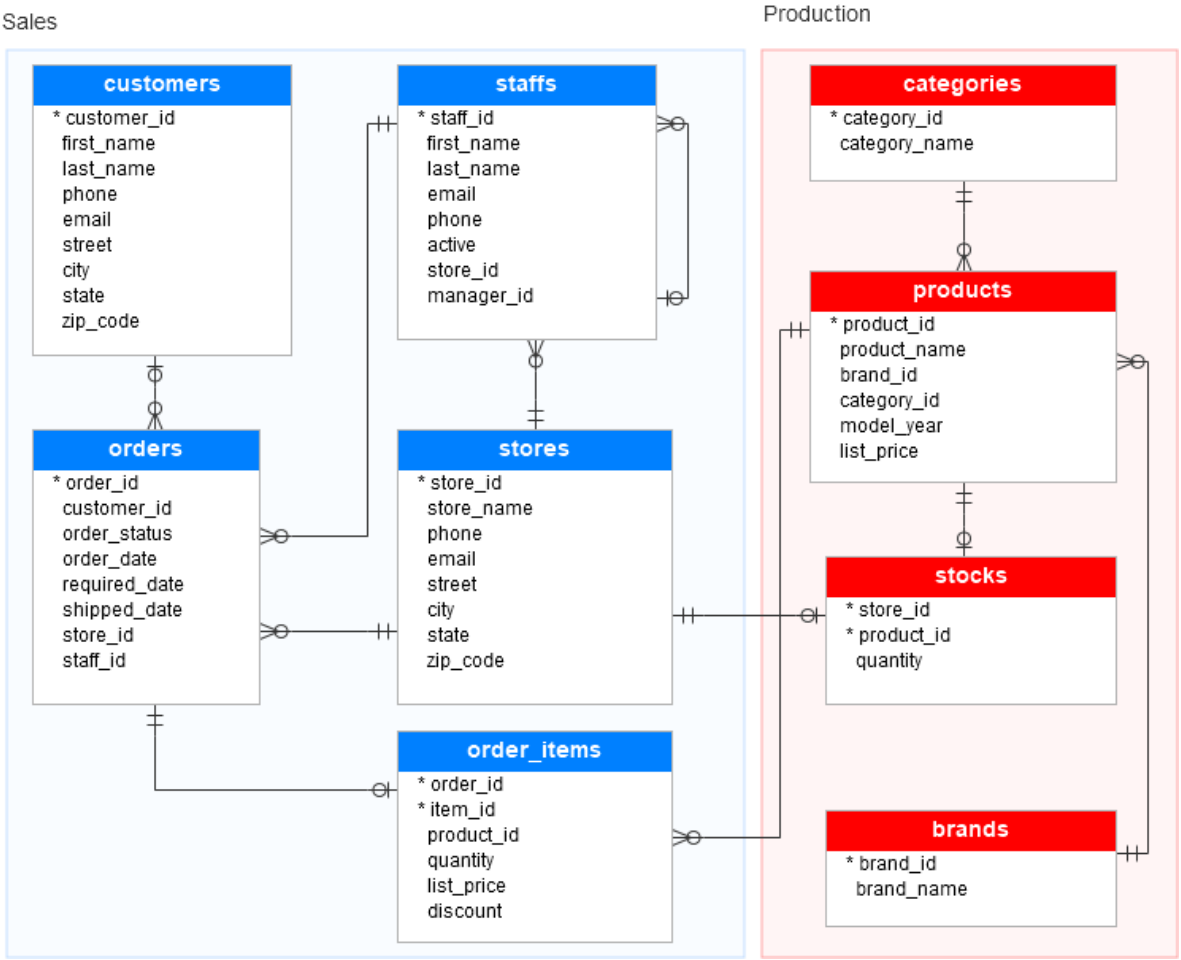
Ministerio de Economía y Servicios Públicos
Gobierno de Salta

academia CIMNE IBER

Ejercicio Adicional - Flask API

En el siguiente ejercicio se solicita trabajar la base de datos `BikeStores` (base de datos de muestra de **SQL Server**), la cual es una base de datos ficticia que contiene tablas y datos relacionados con una tienda de bicicletas, como productos, clientes, empleados, órdenes, etc. La misma está estructurada en dos esquemas: `production` y `sales`, y estos esquemas contienen las siguientes tablas:

1. `production`: `brands`, `categories`, `products` y `stocks`.
2. `sales`: `customers`, `employees`, `orders`, `order_items` y `stores`.



Se encuentra dividida de esta manera para poder separar las tablas que contienen información de la producción de las bicicletas (`production`), de las tablas que contienen información de las ventas de las mismas (`sales`).

A fines prácticos podemos establecer dos bases de datos separadas, una para producción y otra para ventas, y asociarlas mediante el uso de esquemas.

Requerimientos

Usando la base de datos `BikeStores`, se solicita implementar los siguientes endpoints para una API REST enfocada en el recurso **staff**. Para esto se establecen las siguientes restricciones:

Campo	Tipo	Descripción	Opcional (puede ser nulo)
staff_id	integer	ID del miembro del personal	No
first_name	string	Nombre del miembro del personal	No
last_name	string	Apellido del miembro del personal	No
email	string	Email del miembro del personal	No
phone	string	Teléfono del miembro del personal	Sí
active	boolean	Estado activo del miembro del personal	No
store_id	integer	ID de la tienda asociada	No
manager_id	integer	ID del gerente del miembro del personal	Sí

> Acotamos el hecho de que en la tabla antes enunciada vemos el tipo de dato ***string***, queriendo hacer referencia a cadenas de texto. El tipo de dato que este tenga en la base de datos dependerá de quien administra dicha DB. De igual manera podemos establecer esto para el tipo de dato Boolean.

1.1. Obtener un miembro del personal

GET `/staff/<int:staff_id>`

Request

Parámetros de ruta:

- staff_id: el ID del miembro del personal que quiere obtenerse.

Response

Código de estado: 200

Ejemplo:

```
{
      "staff_id": 1,
      "first_name": "John",
      "last_name": "Doe",
      "email": "john.doe@example.com",
      "phone": "123-456-7890",
      "active": 1,
      "store_id": 2,
      "manager_id": null
}
```

1.2. Agregar un miembro del personal

POST /staff

Request

Cuerpo de la solicitud (JSON):

```
{
      "first_name": "Jane",
      "last_name": "Smith",
      "email": "jane.smith@example.com",
      "phone": "234-567-8901",
      "active": 1,
      "store_id": 3
}
```

Response

Código de estado: `201`

Cuerpo de la respuesta:

```
{ }
```

1.3. Actualizar un miembro del personal

PUT `/staff/<int:staff_id>`

Request

Parámetros de ruta:

- staff_id: el ID del miembro del personal que quiere actualizarse.

Cuerpo de la solicitud (JSON):

```
{
      "first_name": "Jane",
      "last_name": "Doe",
      "phone": "234-567-8901"
}
```

Response

Código de estado: `200`

Cuerpo de la respuesta:

```
{ }
```

1.4. Eliminar un miembro del personal

DELETE `/staff/<int:staff_id>`

Request

Parámetros de ruta:

- staff_id: el ID del miembro del personal que quiere eliminarse.

Response

Código de estado: `204`

Cuerpo de la respuesta:

```
{ }
```