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 bases de datos y patrón MVC

UPATECO UNIVERSIDAD PROVINCIAL
ADMINISTRACIÓN • TECNOLOGÍA • OFICIOS

Ministerio de Economía y Servicios Públicos
Gobierno de Salta

academia CIMNE IBER

Trabajo Práctico 3.2 - Acceso a bases de datos y patrón MVC

El objeto de este trabajo práctico será aprender a implementar el modelo MVC en una API REST diseñada con Flask.

Para los siguientes ejercicios se solicita trabajar con 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`.

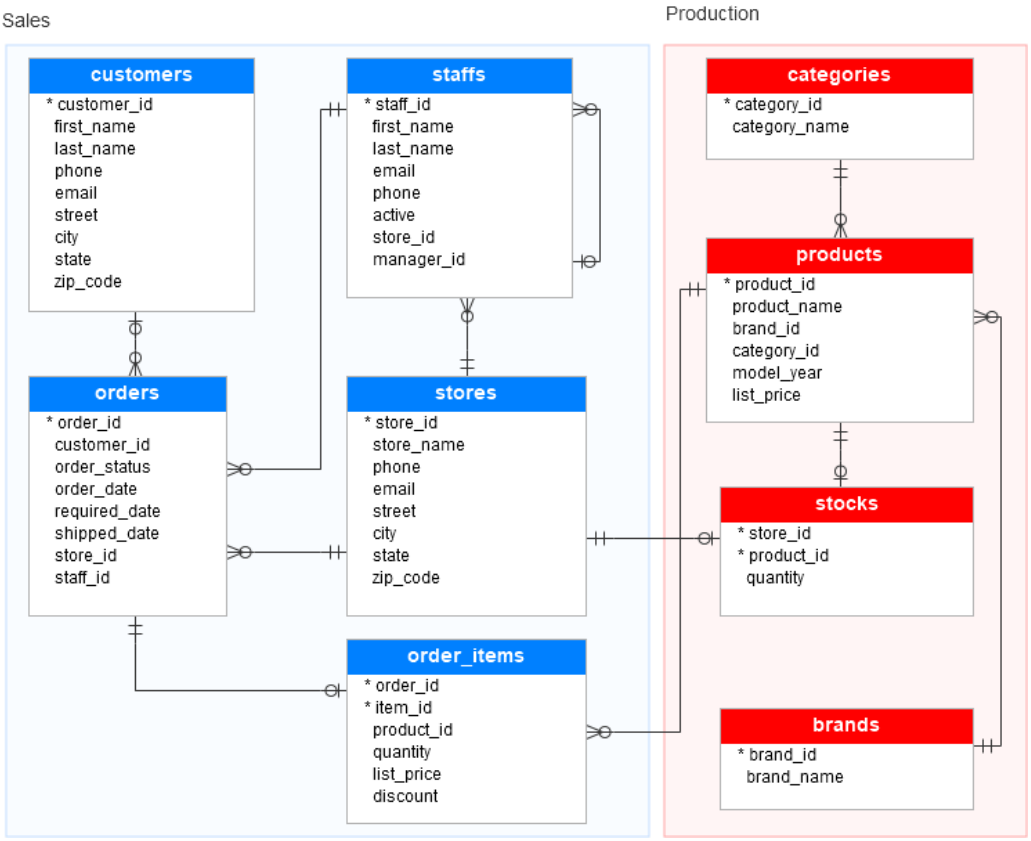

Diagrama de la BikeStores

Ejercicio 1

Empleando la base de datos `sales`, se solicita implementar los siguientes endpoints para una API REST:

1.1. Obtener un cliente

GET `/customers/<int:customer_id>`

Request

Parámetros de ruta:

- customer_id: el ID del cliente que quiere obtenerse.

Response

Código de estado: `200`

Cuerpo de la respuesta:

El cuerpo de la respuesta debe tener los siguientes datos, los cuales se encuentran estructurados de la siguiente manera:

Esquema de `customer`:

Campo	Tipo	Descripción
customer_id	integer	ID del cliente
first_name	string	Nombre del cliente
last_name	string	Apellido del cliente
email	string	Email del cliente
phone	string	Número de teléfono del cliente
street	string	Dirección del cliente
city	string	Ciudad donde reside el cliente
state	string	Estado donde reside el cliente
zip_code	string	Código postal

Ejemplo:

```
{
      "city": "Monroe",
      "customer_id": 10,
      "email": "pamelia.newman@gmail.com",
      "first_name": "Pamelia",
      "last_name": "Newman",
      "phone": null,
      "state": "NY",
      "street": "476 Chestnut Ave. ",
      "zip_code": "10950"
}
```

1.2. Obtener el listado de clientes

GET `/customers`

Request

Query Parameters:

Opcionalmente podemos incluir el siguiente parámetro a la petición para aplicar un filtro sobre la consulta que haremos.

- state: estado donde residen los clientes que deseamos obtener.

Response

Código de estado: `200`

Cuerpo de la respuesta:

El cuerpo de la respuesta debe tener los siguientes datos, los cuales se encuentran estructurados de la siguiente manera:

Campo	Tipo	Descripción
customers	array de objetos	Arreglo de objetos cliente
total	integer	El total de clientes obtenido

> Si no existe ningún cliente que cumpla con la petición solicitada, el arreglo `customers` estará vacío.

Ejemplo:


```
{
      "customers": [
      {
          "city": "Orchard Park",
          "customer_id": 1,
          "email": "debra.burks@yahoo.com",
          "first_name": "Debra",
          "last_name": "Burks",
          "phone": null,
          "state": "NY",
          "street": "9273 Thorne Ave. ",
          "zip_code": "14127"
      },
      {
          "city": "Campbell",
          "customer_id": 2,
          "email": "kasha.todd@yahoo.com",
          "first_name": "Kasha",
          "last_name": "Todd",
          "phone": null,
          "state": "CA",
          "street": "910 Vine Street ",
          "zip_code": "95008"
      }
      ],
    "total": 2
}
```


1.3. Registrar un cliente

POST `/customers`

Request

Cuerpo de la petición:

El cuerpo de la petición debe incluir los siguientes datos que no sean opcionales:

Campo	Tipo	Descripción	Opcional (puede excluirse en el cuerpo de la solicitud)
first_name	string	Nombre del cliente	No
last_name	string	Apellido del cliente	No
email	string	Email del cliente	No
phone	string	Número de teléfono del cliente	Si
street	string	Dirección del cliente	Si
city	string	Ciudad donde reside el cliente	Si
state	string	Estado donde reside el cliente	Si
zip_code	string	Código postal	Si

Ejemplo:

```
{
      "first_name": "Carlos",
      "last_name": "Santana",
      "email": "carlitosantana@gmail.com",
      "phone": "(312) 555-9208"
}
```

Response

Código de estado: `201`

Cuerpo de la respuesta:

```
{ }
```

1.4. Modificar un cliente

PUT `/customers/<int:customer_id>`

Request

Parámetros de ruta:

- customer_id: el ID del cliente que se busca modificar los datos.

Cuerpo de la petición:

El cuerpo de la petición puede incluir uno o más de los siguientes datos:

Campo	Tipo	Descripción
first_name	string	Nombre del cliente
last_name	string	Apellido del cliente
email	string	Email del cliente
phone	string	Número de teléfono del cliente
street	string	Dirección del cliente
city	string	Ciudad donde reside el cliente
state	string	Estado donde reside el cliente
zip_code	string	Código postal

Ejemplo:

```
{
      "email": "carlossantana@gmail.com",
      "phone": "(312) 555-1209"
}
```

Response

Código de estado: `200`

Cuerpo de la respuesta:

```
{ }
```

1.5. Eliminar un cliente

DELETE `/customers/<int:customer_id>`

Request

Parámetros de ruta:

- customer_id: el ID del cliente que quiere eliminarse.

Response

Código de estado: `204`

Cuerpo de la respuesta:

```
{ }
```

Ejercicio 2

Empleando la base de datos `production`, se solicita implementar los siguientes endpoints para una API REST. Los cuales tratan sobre las tablas `brands`, `categories` y `products`. Deberán implementarse las respectivas capas del modelo MVC para cada uno de los endpoints.

2.1. Obtener un producto

GET `/products/<int:product_id>`

Request

Parámetros de ruta:

- product_id: el ID del producto que quiere obtenerse.

Response

Código de estado: `200`

Cuerpo de la respuesta:

El cuerpo de la respuesta debe tener los siguientes datos, los cuales se encuentran estructurados de la siguiente manera:

Esquema de `product`:

Campo	Tipo	Descripción
product_id	integer	ID del producto
product_name	string	Nombre del producto
brand	object	Nombre de la marca del producto
category	object	Nombre de la categoría del producto
model_year	integer	Año del modelo del producto
list_price	float	Precio de lista (en dólares). No confundir con precio al momento de la venta.

Esquema de `brand`:

Campo	Tipo	Descripción
brand_id	integer	ID de la marca
brand_name	string	Nombre de la marca

Esquema de `category`:

Campo	Tipo	Descripción
category_id	integer	ID de la categoría
category_name	string	Nombre de la categoría

Ejemplo de respuesta para un producto:


```
{
      "brand": {
          "brand_id": 8,
          "brand_name": "Surly"
      },
          "category": {
          "category_id": 4,
          "category_name": "Cyclocross Bicycles"
      },
          "list_price": "1549.00",
          "model_year": 2016,
          "product_id": 10,
          "product_name": "Surly Straggler - 2016"
}
```

2.2. Obtener un listado de productos

GET `/products`

Request

Query Parameters:

Opcionalmente podemos incluir los siguientes parámetros a la petición para aplicar filtros sobre la consulta que haremos.

- brand_id: filtro por marca del producto.
- category_id: filtro por categoría del producto.

Response

Código de estado: `200`

Cuerpo de la respuesta:

Campo	Tipo	Descripción
products	array de objetos	Arreglo de objetos producto
total	integer	El total de productos obtenido

Ejemplo:


```
{
    "products": [
      {
          "brand": {
          "brand_id": 1,
          "brand_name": "Electra"
      },
          "category": {
          "category_id": 1,
          "category_name": "Children Bicycles"
      },
          "list_price": "269.99",
          "model_year": 2016,
          "product_id": 21,
          "product_name": "Electra Cruiser 1 (24-Inch) - 2016"
      },
      {
          "brand": {
          "brand_id": 1,
```

```
            "brand_name": "Electra"
      },
            "category":
            "category_id":
            "category_name": "Children Bicycles"
      },
            "list_price": "269.99",
            "model_year": 2016,
            "product_id": 22,\
            "product_name": "Electra Girl's Hawaii 1 (16-inch) - 2015/20
16"
      }
      ],
      "total": 2
}
```


2.3. Registrar un producto

POST `/products`

Request

Cuerpo de la petición:

El cuerpo de la petición debe incluir los siguientes datos:

Campo	Tipo	Descripción
product_name	string	Nombre del producto
brand_id	integer	ID de la marca del producto
category_id	integer	ID de la categoría del producto
model_year	integer	Año del modelo del producto
list_price	float	Precio de lista (en dólares). No confundir con precio al momento de la venta.

Ejemplo:


```
{
      "product_name": "Trek Hiperfly XLR8 - 2019",
      "brand_id": 9,
      "category_id": 4,
      "model_year": 2019,
      "list_price": 994.99
}
```

Response

Código de estado: `201`

Cuerpo de la respuesta:

```
{ }
```

2.4. Modificar un producto

PUT `/products/<int:product_id>`

Request

Parámetros de ruta:

- product_id: el ID del producto que se busca modificar.

Cuerpo de la petición:

El cuerpo de la petición puede incluir uno o más de los siguientes datos:

Campo	Tipo	Descripción
product_name	string	Nombre del producto
brand_id	integer	ID de la marca del producto
category_id	integer	ID de la categoría del producto
model_year	integer	Año del modelo del producto
list_price	float	Precio de lista (en dólares). No confundir con precio al momento de la venta.

Ejemplo:

```
{
      "list_price": 1014.99
}
```

Response

Código de estado: `200`

Cuerpo de la respuesta:

```
{ }
```

1.5. Eliminar un producto

DELETE `/products/<int:product_id>`

Request

Parámetros de ruta:

- product_id: el ID del producto que quiere eliminarse.

Response

Código de estado: `204`

Cuerpo de la respuesta:

```
{ }
```