Actividad Evaluada: Desarrollo de un Sistema con API Integrada

Descripción

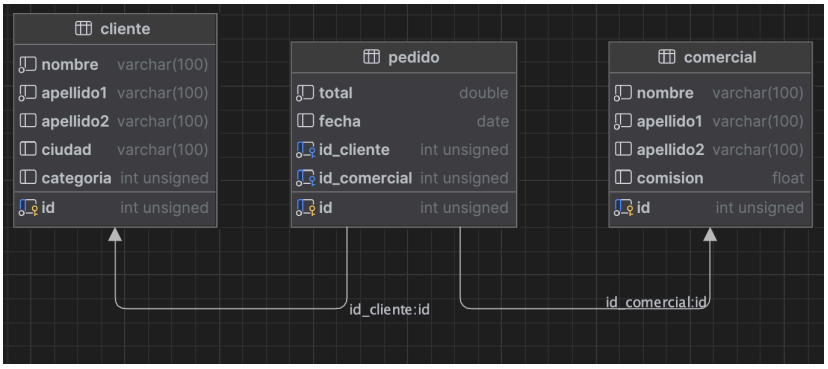
En esta actividad, los estudiantes trabajarán en grupos de 3 para desarrollar un sistema que consuma una API en Django proporcionada por el profesor. Cada grupo podrá elegir si desarrollan las vistas de la aplicación utilizando:

1. Un nuevo proyecto en Django (utilizando la biblioteca Re**quest** de Python para la integración con la API).

2. Una aplicación con Angular (utilizando la herramienta **Axios** para realizar las peticiones a la API).

El objetivo de esta actividad es que los estudiantes comprendan el funcionamiento de APIs RESTfull y las integren de manera efectiva en un sistema completo.

Considere que la **Api** funciona como un CRUD exponiendo **Endpoints** para dar estructura y operativa a la siguiente base de datos:



Para lograr el funcionamiento correcto del API debes crear la base de datos de forma local en tu mysql con el nombre “ventas”:

```
CREATE DATABASE ventas;

USE ventas;
```

```
1  api_cliente
2  api_comercial
3  api_pedido
```

Luego en Django genera las migraciones para que las tablas se creen automáticamente en la API:

```
python manage.py makemigrations
python manage.py migrate
```

Puedes usar las siguiente inserciones para tener datos de prueba dentro de tu DB local, este es un extracto del archivo .sql de la semana 3 disponible en Canvas:

```sql
INSERT INTO api_cliente VALUES(1, 'Aarón', 'Rivero', 'Gómez', 'Almería', 100);
INSERT INTO api_cliente VALUES(2, 'Adela', 'Salas', 'Díaz', 'Granada', 200);
INSERT INTO api_cliente VALUES(3, 'Adolfo', 'Rubio', 'Flores', 'Sevilla', NULL);
INSERT INTO api_cliente VALUES(4, 'Adrián', 'Suárez', NULL, 'Jaén', 300);
INSERT INTO api_cliente VALUES(5, 'Marcos', 'Loyola', 'Méndez', 'Almería', 200);
INSERT INTO api_cliente VALUES(6, 'María', 'Santana', 'Moreno', 'Cádiz', 100);
INSERT INTO api_cliente VALUES(7, 'Pilar', 'Ruiz', NULL, 'Sevilla', 300);
INSERT INTO api_cliente VALUES(8, 'Pepe', 'Ruiz', 'Santana', 'Huelva', 200);
INSERT INTO api_cliente VALUES(9, 'Guillermo', 'López', 'Gómez', 'Granada', 225);
INSERT INTO api_cliente VALUES(10, 'Daniel', 'Santana', 'Loyola', 'Sevilla', 125);
INSERT INTO api_comercial VALUES(1, 'Daniel', 'Sáez', 'Vega', 0.15);
INSERT INTO api_comercial VALUES(2, 'Juan', 'Gómez', 'López', 0.13);
INSERT INTO api_comercial VALUES(3, 'Diego','Flores', 'Salas', 0.11);
INSERT INTO api_comercial VALUES(4, 'Marta','Herrera', 'Gil', 0.14);
INSERT INTO api_comercial VALUES(5, 'Antonio','Carretero', 'Ortega', 0.12);
INSERT INTO api_comercial VALUES(6, 'Manuel','Domínguez', 'Hernández', 0.13);
INSERT INTO api_comercial VALUES(7, 'Antonio','Vega', 'Hernández', 0.11);
INSERT INTO api_comercial VALUES(8, 'Alfredo','Ruiz', 'Flores', 0.05);
INSERT INTO api_pedido VALUES(1, 150.5, '2017-10-05', 5, 2);
INSERT INTO api_pedido VALUES(2, 270.65, '2016-09-10', 1, 5);
INSERT INTO api_pedido VALUES(3, 65.26, '2017-10-05', 2, 1);
INSERT INTO api_pedido VALUES(4, 110.5, '2016-08-17', 8, 3);
INSERT INTO api_pedido VALUES(5, 948.5, '2017-09-10', 5, 2);
INSERT INTO api_pedido VALUES(6, 2400.6, '2016-07-27', 7, 1);
INSERT INTO api_pedido VALUES(7, 5760, '2015-09-10', 2, 1);
INSERT INTO api_pedido VALUES(8, 1983.43, '2017-10-10', 4, 6);
```

Instrucciones

1. **Consumo de la API**: Utilicen la API proporcionada por el profesor. Esta API gestionará datos relacionados con un modelo de ventas.
    1. Desde este enlace podrás acceder a la documentación básica de la API: https://www.postman.com/crimson-astronaut-185624/workspace/derrollo-web-y-movil-unab/collection/10936845-c21314b9-eccb-425c-92a5-0a9e72619ab8?action=share&creator=10936845

2. **Elección de la tecnología:**
    1. Si eligen Django, deberán desarrollar las vistas y las interfaces para consumir la API utilizando la biblioteca **request**.
    2. Si eligen Angular, deberán desarrollar los componentes necesarios y hacer las peticiones a la API utilizando **Axios**.
3. **Despliegue de la aplicación**: La aplicación debe estar completamente funcional, mostrando los datos consumidos de la API de manera interactiva y eficiente.
4. **Interfaz de usuario**: La interfaz debe ser clara, intuitiva y permitir la interacción con los datos obtenidos de la API (visualización **[GET]**, creación **[POST]**).
    1. Requerimientos de UI:
        1. Generar un vista principal para mostrar la siguiente información:
            1. Crear cliente (debe generar registro en la DB).
            2. Crear comercial (debe generar registro en la DB).
            3. Crear pedido (debe generar registro en la DB).
            4. Listado de todos los pedidos mostrando sus atributos como tabla.
        2. Realiza un gráfico que represente el total de todas los pedidos presentadas en un gráfico tipo serie de tiempo utiliza la biblioteca **chart.js** para este ítem, puede tomar como referencia el siguiente enlace https://www.chartjs.org/docs/latest/samples/line/line.html
        3. Realiza un gráfico de Pie mostrando las compras por cliente, utiliza la biblioteca **chart.js** para este ítem, puede tomar como referencia el siguiente enlace https://www.chartjs.org/docs/latest/samples/other-charts/pie.html
    2. Tienes libertad absoluta para proponer la estructura sobre estos requerimientos visuales, priorizando el orden y buena presentación de los componentes, puedes igualmente ayudarte con Boostrap o Tailwind, incluso

es posible hacer una aplicación tipo one page lo importante es que se expongan los requerimientos mencionado de buena forma.

5. **Documentación:** Cada grupo debe entregar un documento explicando:
    1. Cómo integraron la API en la aplicación.
    2. Decisiones técnicas tomadas (como la elección de Django o Angular).
    3. Descripción de los componentes/vistas creados.
6. **Plazo de entrega**: Se otorgarán 2 semanas para completar esta actividad, siendo la fecha de entrega tope el día **Miercoles 30 de Octubre.**

Rúbrica de Evaluación

Criterios	Descripción	Ponderación
Integración de la API	El sistema se conecta correctamente a la API proporcionada y realiza las operaciones CRUD solicitadas (crear, leer, actualizar, eliminar).	25%
Funcionalidad de la aplicación	La aplicación es completamente funcional, con vistas o componentes que permiten la interacción adecuada con los datos recibidos desde la API.	20%
Uso adecuado de tecnologías	El grupo utiliza adecuadamente las herramientas (`request` para Django, `Axios` para Angular), demostrando comprensión en su integración con la API.	20%
Interfaz de usuario (UI)	La interfaz es clara, intuitiva y presenta correctamente los datos de la API. Se valora la experiencia de usuario y la organización de los elementos visuales.	15%
Trabajo en equipo	Participación equitativa de los miembros del grupo. Se evaluará mediante una reflexión individual de los integrantes sobre las contribuciones de cada uno y los desafíos grupales.	10%
Documentación del proyecto	El grupo entrega una documentación clara que describe los pasos seguidos, decisiones técnicas y el flujo de trabajo en la integración con la API.	10%