

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD: INFORMÁTICA Y ELECTRÓNICA

**ESCUELA DE INGENIERÍA EN SISTEMAS**

**CARRERA: SOFTWARE**

# GUÍA DE LABORATORIO DE APLICACIONES INFORMÁTICAS II

## PRÁCTICA No. 1

1. **DATOS GENERALES:**

**NOMBRE DEL ESTUDIANTE CODIGO DEL ESTUDIANTE**

Marlon Medina 6827

**FECHA DE REALIZACIÓN: FECHA DE ENTREGA:**

2024/02/29 2024/02/07

1. **OBJETIVO(S):**

**2.1. GENERAL**

Desarrollar una aplicación de una carta digital para un restaurante utilizando el Python y JavaScript utilizando Django framework, implementando la arquitectura de n capas.

**2.2. ESPECÍFÍCOS**

* Automatizar el proceso de pedidos implementado la carta digital para un restaurante.
* Diseñar un sistema el cual almacene información sobre los platos disponibles, ingredientes, precios, pedidos de los clientes, y perfiles de usuarios o trabajadores del restaurante utilizando la base de datos SQLite3.
* Implementar mecanismos de almacenamiento en caché para mejorar el rendimiento de consultas frecuentes, como la disponibilidad de platos o los ingredientes más populares, reduciendo la carga en la base de datos sqlite3. Esto mejora la eficiencia global de la aplicación de la carta digital.

1. **METODOLOGÍA**

Scrum es un marco ágil que revoluciona el desarrollo de software al enfocarse en la adaptabilidad y la colaboración. Incluye roles clave como el Product Owner, que representa los intereses del cliente, el Scrum Master, que actúa como facilitador del proceso, y el Equipo de Desarrollo, que produce los incrementos del producto al final de cada Sprint. Scrum fomenta la transparencia y la mejora continua, con eventos definidos como reuniones de planificación, revisiones y retrospectivas. Los valores fundamentales de compromiso, coraje y un enfoque abierto son la fuerza motriz para que los equipos entreguen productos de alta calidad en intervalos regulares, agregando valor como la adaptación rápida y sin esfuerzo a los cambios por parte del cliente. Scrum se ha convertido en una de las metodologías ágiles más adoptadas, basada en principios como la colaboración, inspección y adaptación. Fue efectiva en el entorno de desarrollo de software, el enfoque iterativo, centrado en el usuario y de entrega incremental, permitiendo a los equipos de proyecto responder de manera rápida y adecuada a los requisitos cambiantes del cliente y asegurando la excelencia general en la entrega de productos funcionales.

1. **EQUIPOS Y MATERIALES:**

* Computador
* Entorno integrado de desarrollo (IDE)
* Aula virtual
* Acceso a internet
* Bibliografía

1. **MARCO TEORICO:**

**Django:**

Django es un marco de desarrollo web de alto nivel que fomenta el desarrollo rápido y limpio, con un diseño pragmático. Escrito en Python, sigue los patrones de diseño Modelo-Vista-Controlador (MVC) y N-Capas. Proporciona una estructura organizada para el desarrollo web, lo que permite a los desarrolladores construir aplicaciones web complejas de manera eficiente. Django se destaca por su enfoque en la reutilización de componentes, la conectividad a diversas bases de datos, y su sistema de migración integrado para evolucionar los esquemas de base de datos. Viene con un conjunto de herramientas integradas para tareas comunes de desarrollo web, como autenticación de usuarios, paneles de administración, y la prevención de vulnerabilidades comunes. La comunidad de Django es activa y extensa, ofreciendo abundantes recursos, extensiones y soporte.

**Características Principales:**

* **Desarrollo Rápido:** Django está diseñado para facilitar el desarrollo rápido de aplicaciones web. Su lema es "The web framework for perfectionists with deadlines", lo que subraya su objetivo de hacer que el desarrollo web sea lo más eficiente y de alta calidad posible.
* **Arquitectura DRY (Don't Repeat Yourself):** Promueve la reutilización de código y se enfoca en minimizar la duplicación de información en el desarrollo, lo que ayuda a reducir el tiempo de desarrollo y mejora la mantenibilidad del código.
* **Sistema de Migraciones Integrado:** Django incluye un robusto sistema de migraciones para sus modelos de datos, lo que facilita el cambio y la evolución de las bases de datos a lo largo del tiempo sin requerir reconstrucción manual.
* **Sistema de Plantillas:** Django viene con un potente y flexible sistema de plantillas, diseñado para definir la estructura de salida de las aplicaciones web, facilitando la separación entre la lógica de la aplicación y la presentación.

**Arquitectura N – Capas**

La arquitectura de múltiples niveles, o modelo de n capas, se refiere a la estrategia de dividir un sistema de software en varias capas, cada una dedicada a una función distinta. Estas capas interactúan entre sí para lograr los fines del sistema, facilitando así una mayor modularidad, escalabilidad y facilidad de mantenimiento en las aplicaciones. Este método es ampliamente adoptado para organizar de manera eficiente la estructura de los sistemas de software.

**Capa de Presentación (Interfaz de Usuario):**

Esta capa, ubicada en la cima, actúa como la interfaz a través de la cual los usuarios interactúan, gestionando sus solicitudes y mostrando los resultados obtenidos. Se compone de elementos como interfaces web, aplicaciones para dispositivos móviles y comandos de línea de interfaz, entre otros.

**Capa de Lógica de Aplicación (o Negocio):**

En el nivel de lógica de negocio, se desarrolla el conjunto de operaciones y reglas que definen el comportamiento específico de la aplicación. Esta capa es clave para orquestar las actividades y asegurarse de que la manipulación de los datos se realice siguiendo las directrices establecidas por el negocio.

**Capa de Acceso a Datos:**

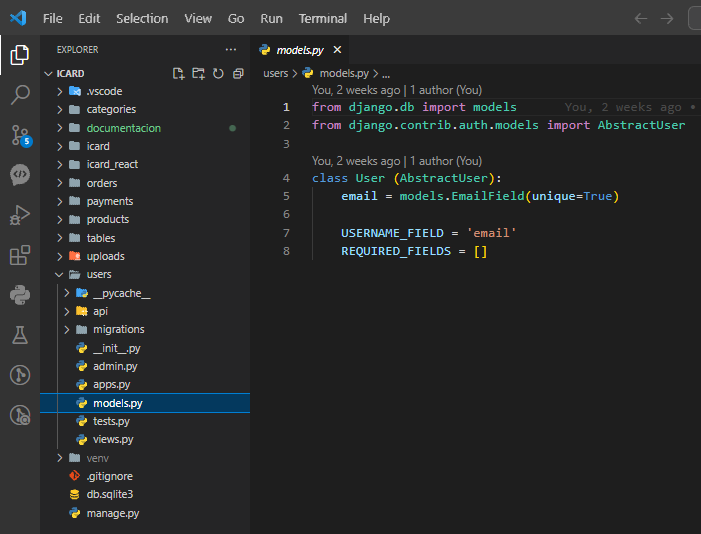
La capa dedicada al acceso de datos se encarga de la interacción con el medio de almacenamiento, ya sea una base de datos, un sistema de archivos u otro tipo de repositorio de persistencia. Aquí se ejecutan las operaciones CRUD (Crear, Leer, Actualizar, Eliminar) y se efectúa la conversión de los datos desde y hacia el formato que utiliza la aplicación.

**Ventajas de la Arquitectura de N Capas:**

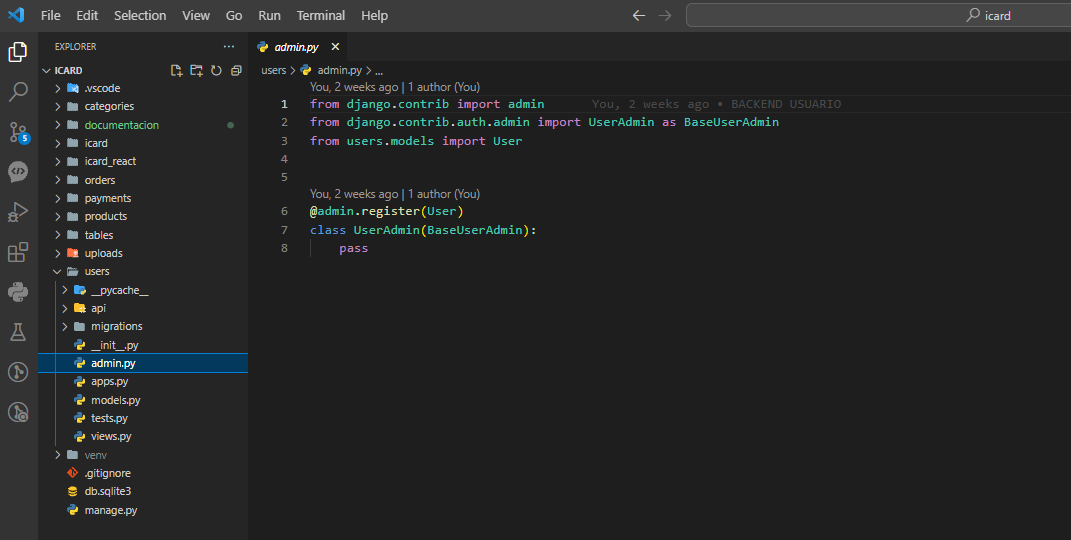
* **Separación de Responsabilidades:** Cada capa tiene una responsabilidad única dentro de la aplicación, lo que simplifica el desarrollo y mantenimiento al permitir a los desarrolladores enfocarse en una parte específica de la funcionalidad sin interferir con el resto. Esto mejora la organización del código y facilita la comprensión del sistema.
* **Flexibilidad y Escalabilidad:** Al separar las funcionalidades en diferentes capas, es más fácil modificar o actualizar una capa específica sin afectar a las demás. Esto permite a las empresas adaptarse rápidamente a los cambios de requerimientos o tecnologías, así como escalar la aplicación añadiendo más recursos o modificando solo las capas necesarias.
* **Facilidad de Pruebas:** La arquitectura de N-Capas facilita las pruebas unitarias y de integración, ya que cada capa puede ser probada de manera independiente. Esto ayuda a identificar y corregir errores de forma más eficiente, mejorando la calidad del software.
* Reutilización de Código: La separación en capas promueve la reutilización del código, ya que las funcionalidades comunes a varias aplicaciones pueden ser implementadas en capas que luego se utilizan en diferentes proyectos. Esto reduce el tiempo de desarrollo y los costos asociados.
* Interoperabilidad: Al utilizar una arquitectura de N-Capas, se facilita la integración con otras aplicaciones y sistemas. Las interfaces bien definidas entre las capas permiten que la aplicación sea más adaptable y capaz de interactuar con diferentes entornos y tecnologías, mejorando la interoperabilidad.

1. **PROCEDIMIENTO:**

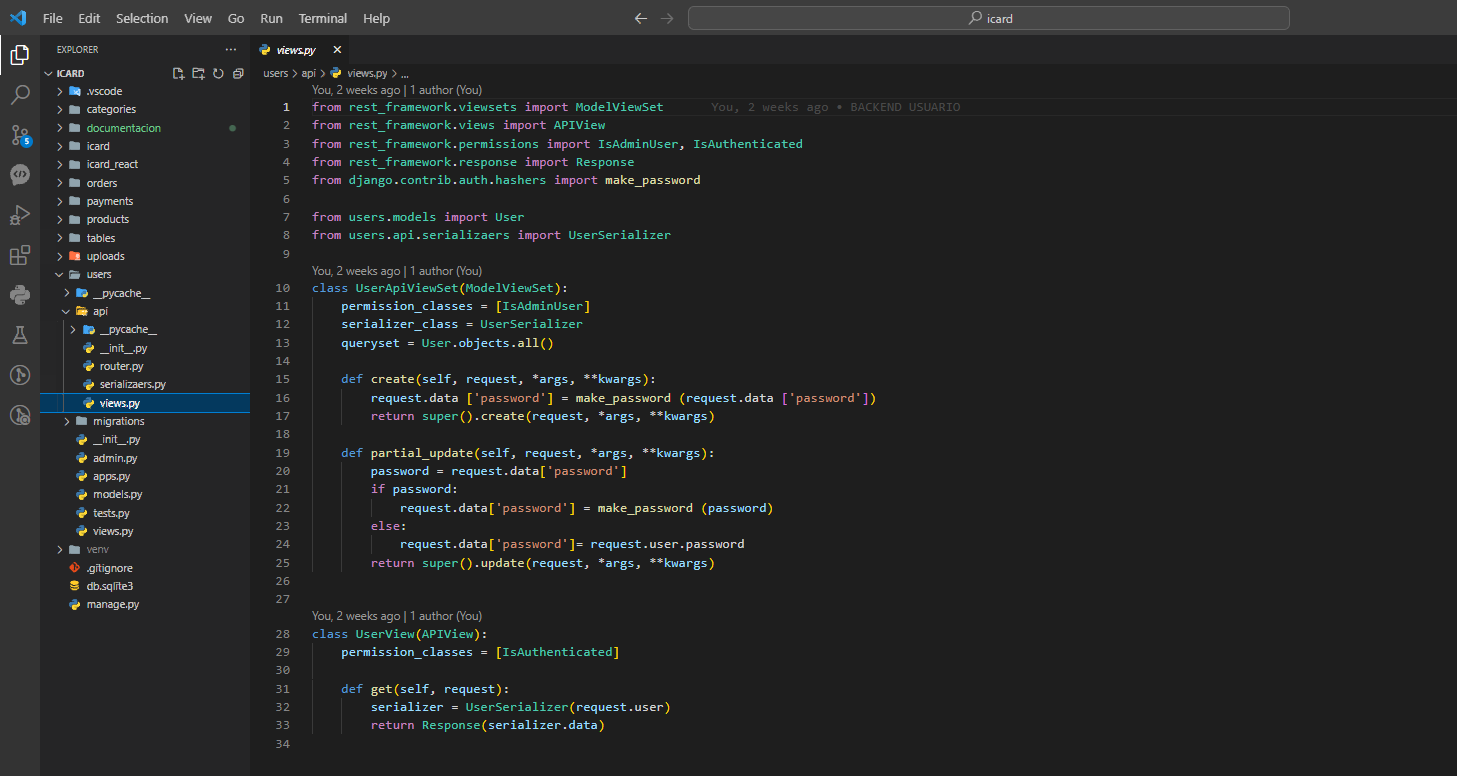
**MODELADO DE LOS USUARIOS**



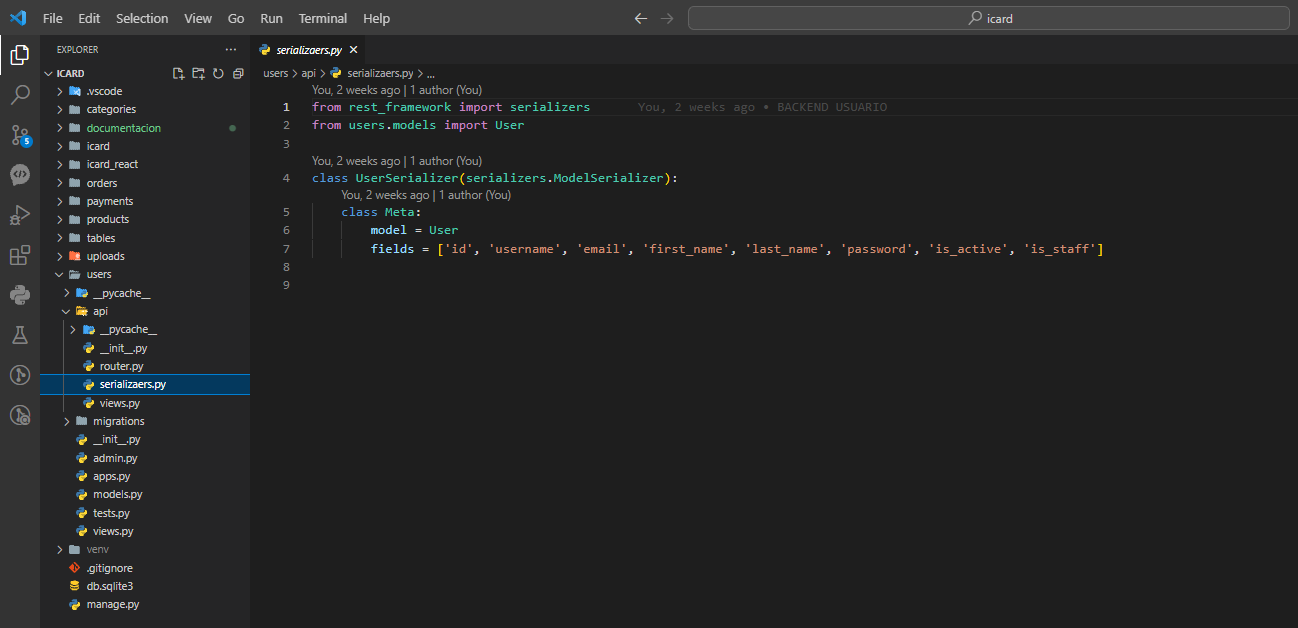
**SISTEMA DE ADMINISTRACIÓN DE LOS USUARIOS**

****

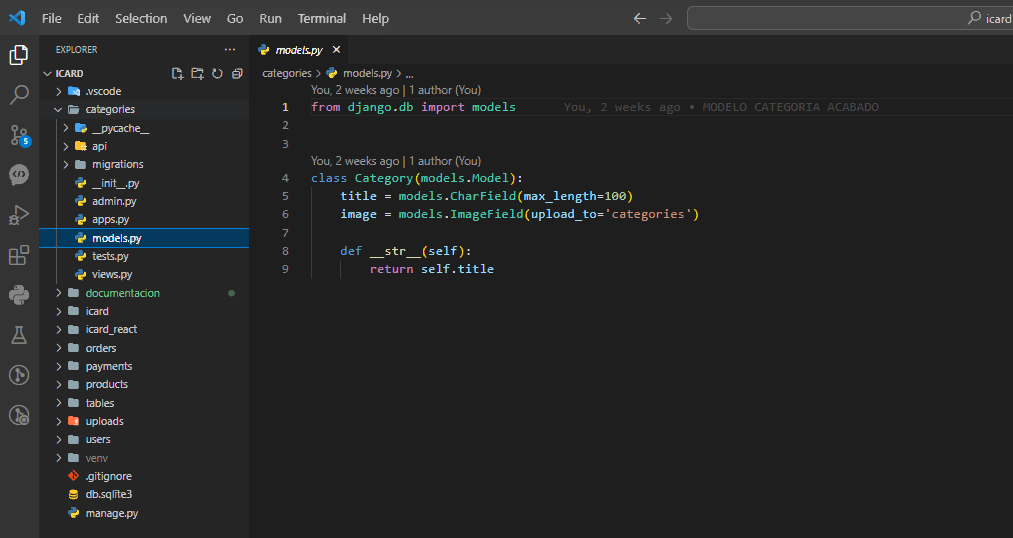
**MODELO VISTA DEL MODELADO DE LOS USUARIOS**

****

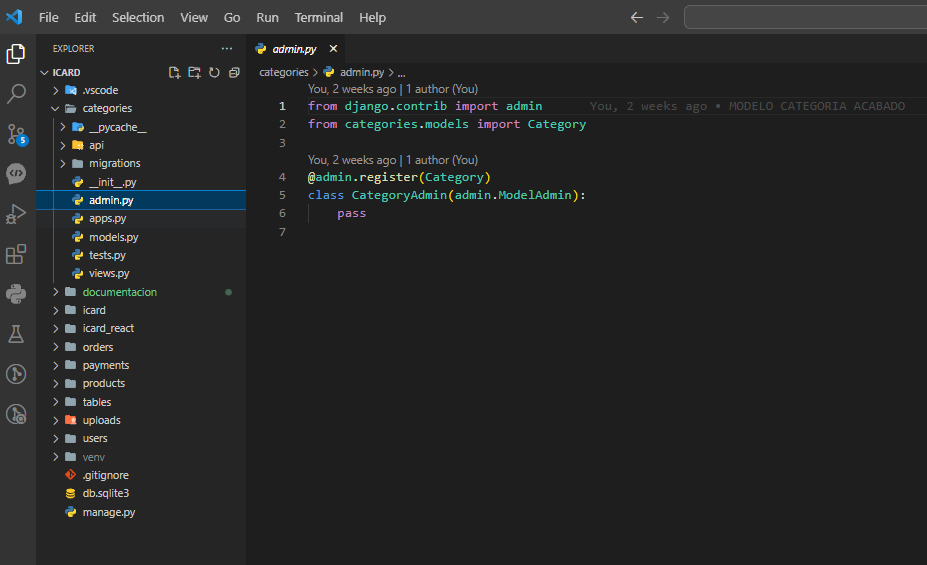
**SERIALIZERS DE LOS USUARIOS**

****

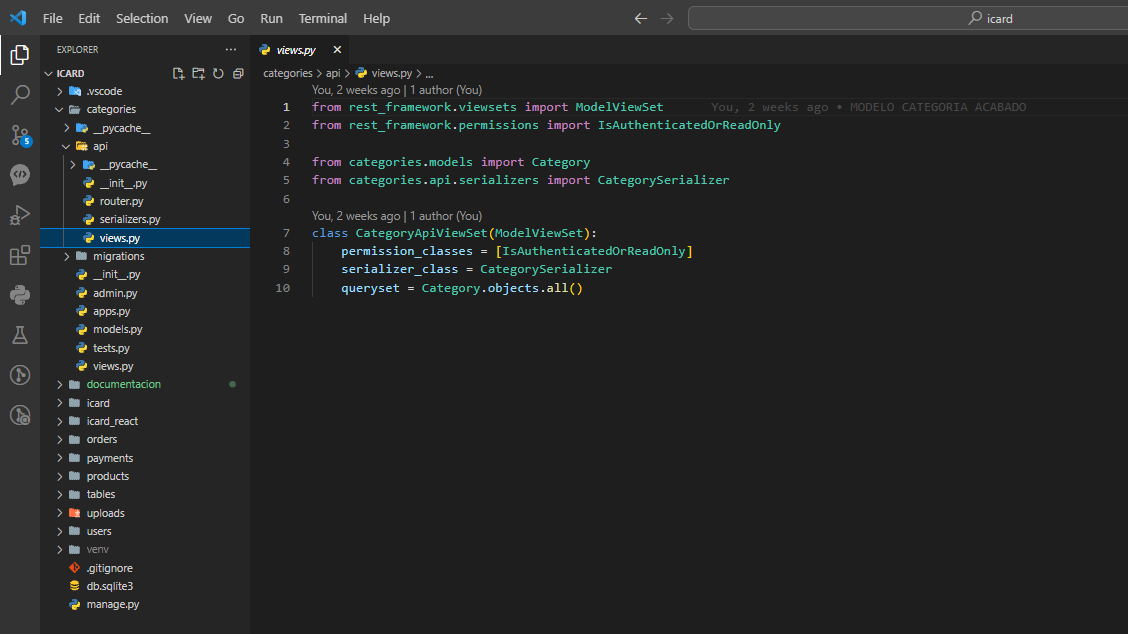
**MODELADO DE LAS CATEGORÍAS**



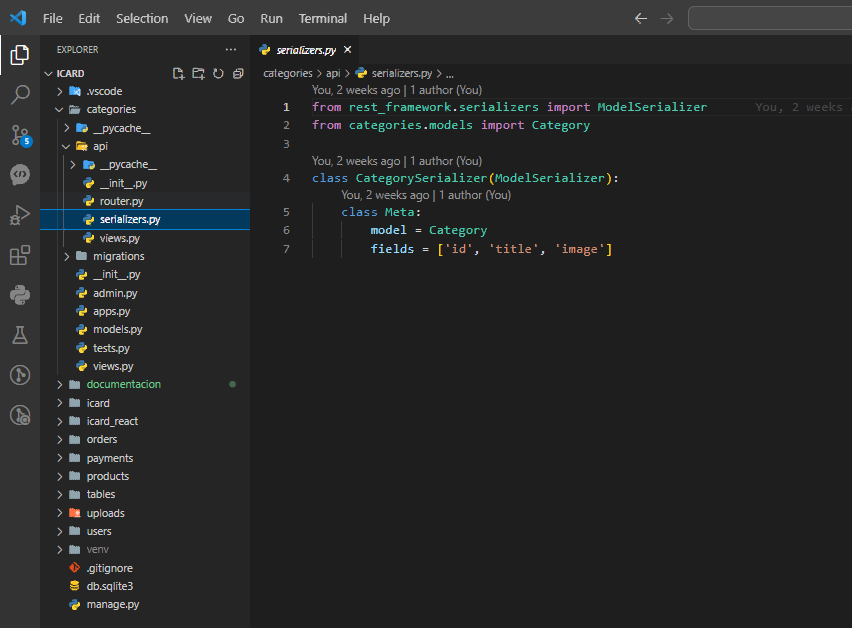
**SISTEMA DE ADMINISTRACIÓN LAS CATEGORÍAS**

****

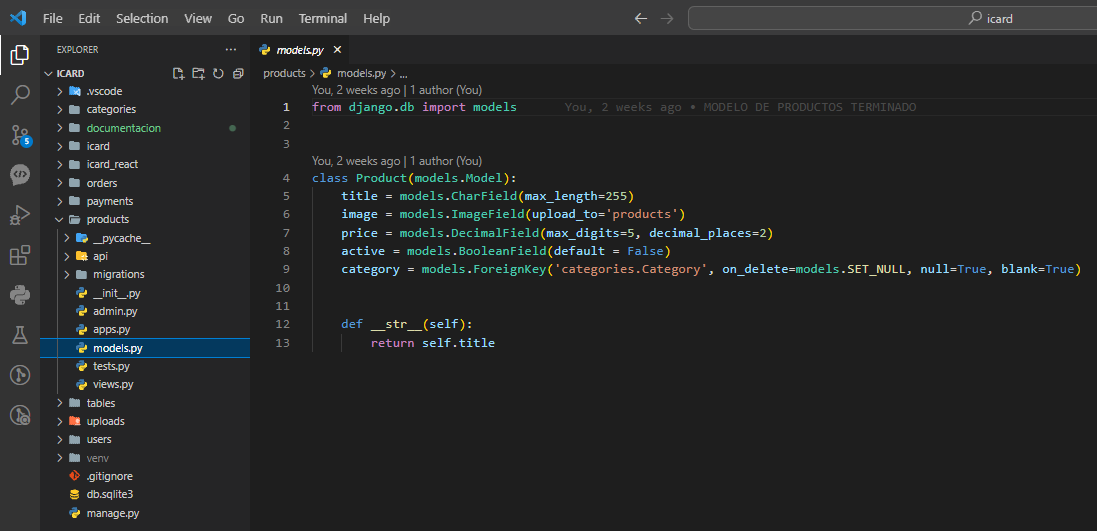
**MODELO VISTA DEL MODELADO DE LAS CATEGORÍAS**

****

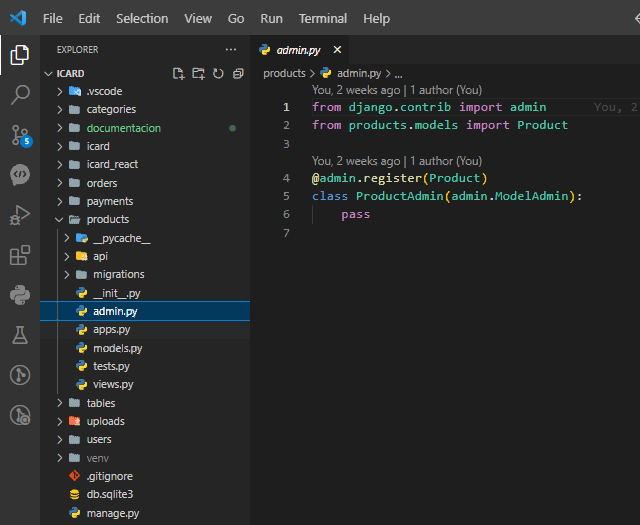
**SERIALIZERS DE LAS CATEGORÍAS**

****

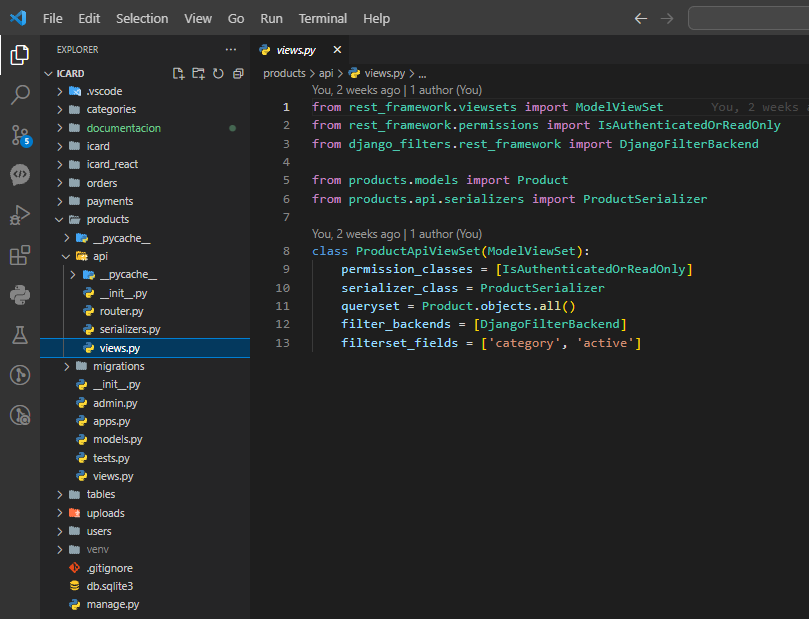
**MODELADO DE LOS PRODUCTOS**



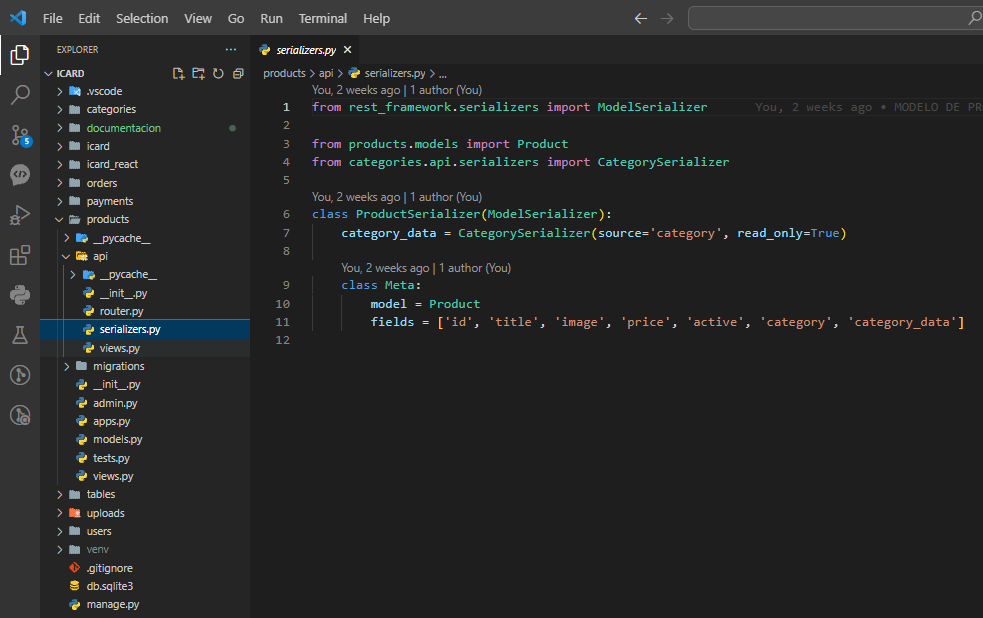
**SISTEMA DE ADMINISTRACIÓN LOS PRODUCTOS**

****

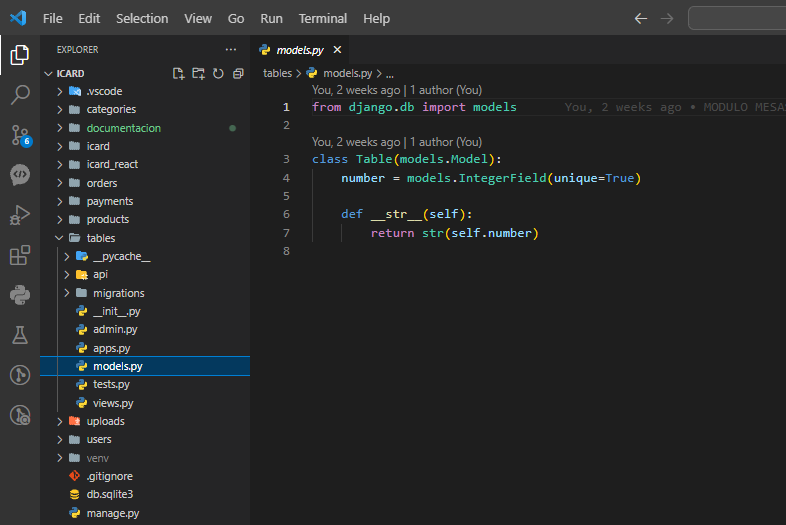
**MODELO VISTA DEL MODELADO DE LOS PRODUCTOS**

****

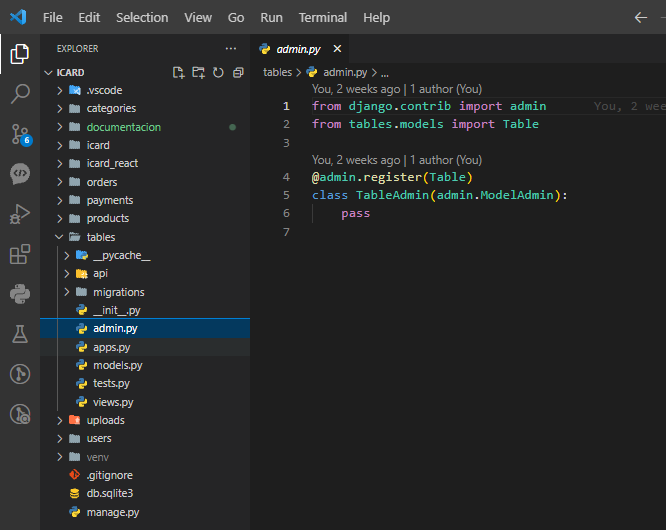
**SERIALIZERS DE LOS PRODUCTOS**

****

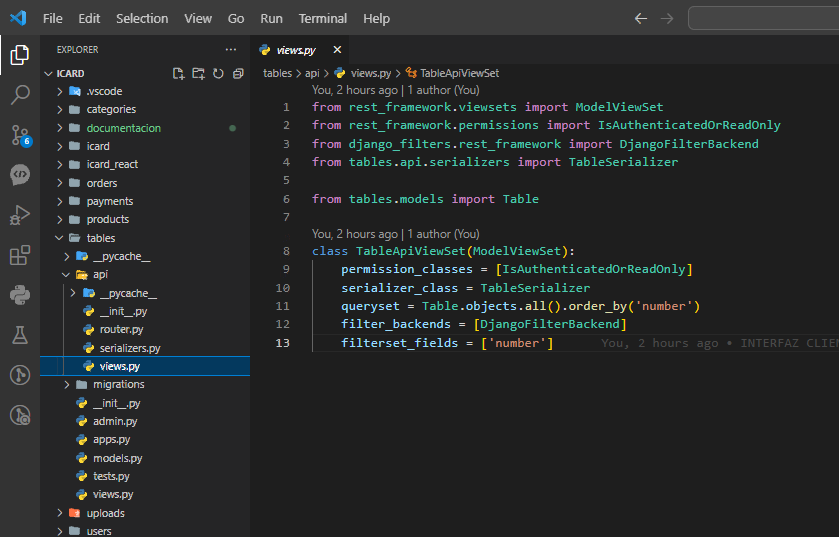
**MODELADO DE LAS MESAS**

****

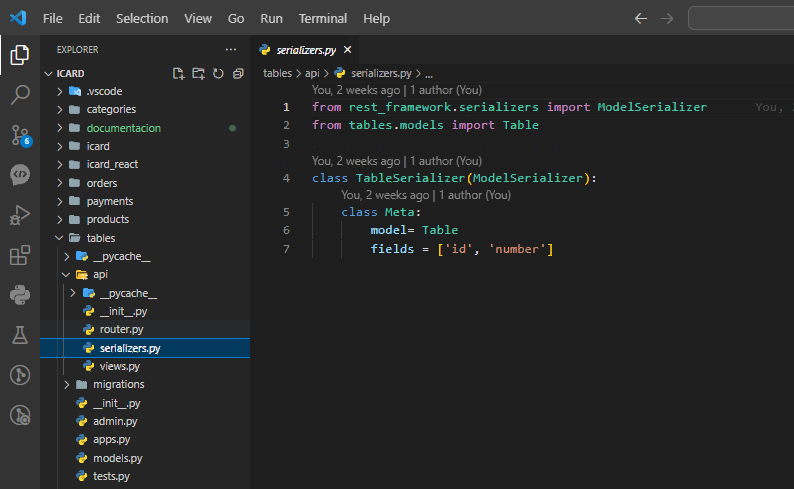
**SISTEMA DE ADMINISTRACIÓN DE LAS MESAS**

****

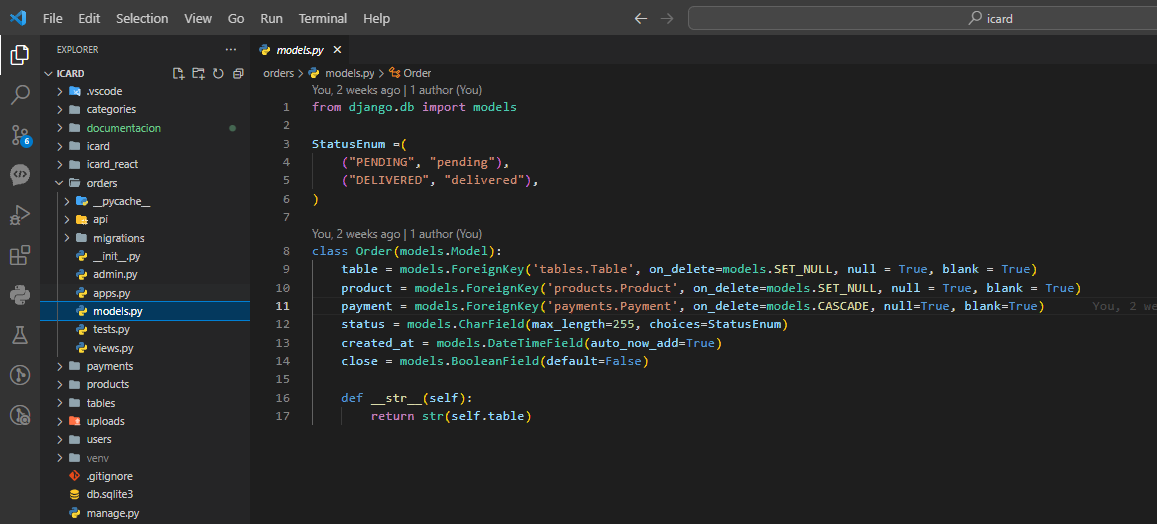
**MODELO VISTA DEL MODELADO DE LAS MESAS**

****

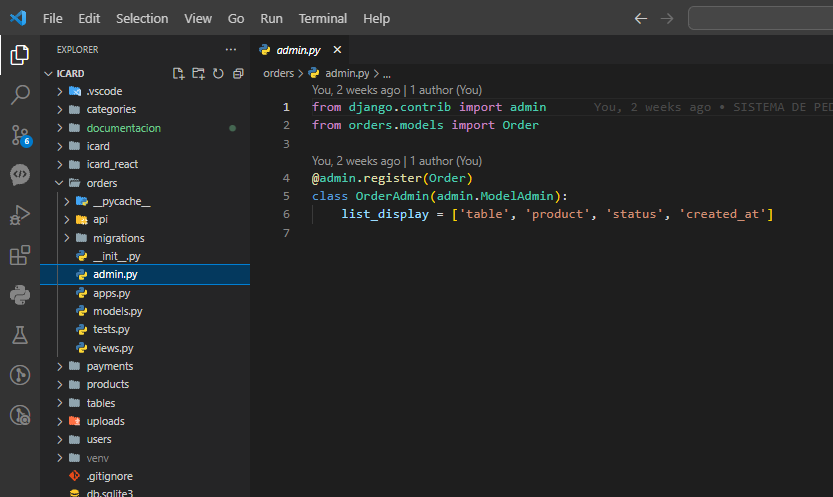
**SERIALIZERS DE LAS MESAS**

****

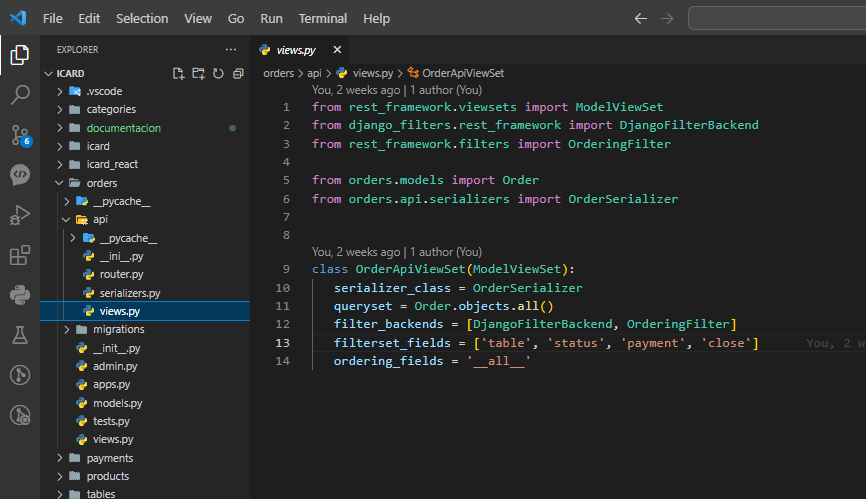
**MODELADO DEL SISTEMA DE PEDIDOS**

****

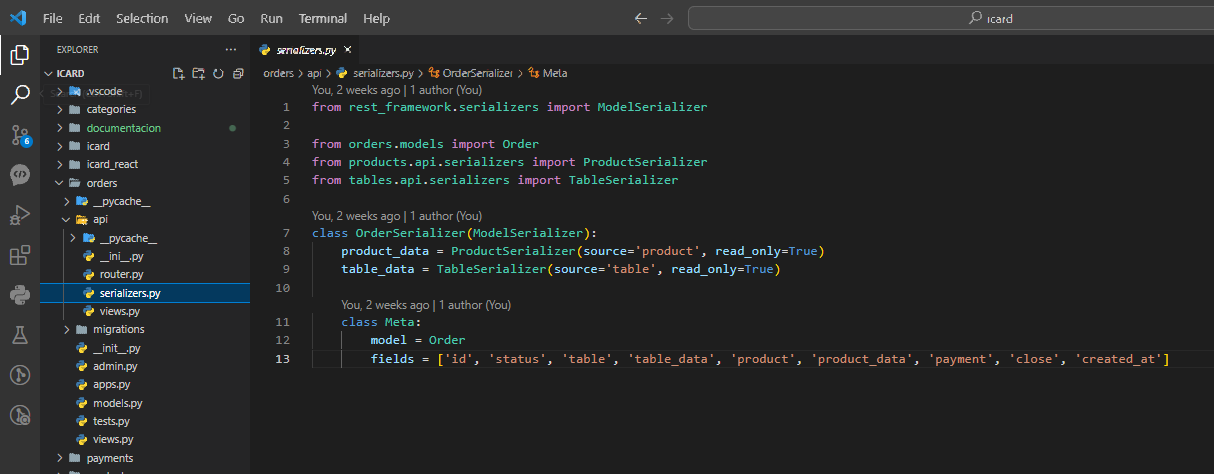
**SISTEMA DE ADMINISTRACIÓN DE LOS PEDIDOS**

****

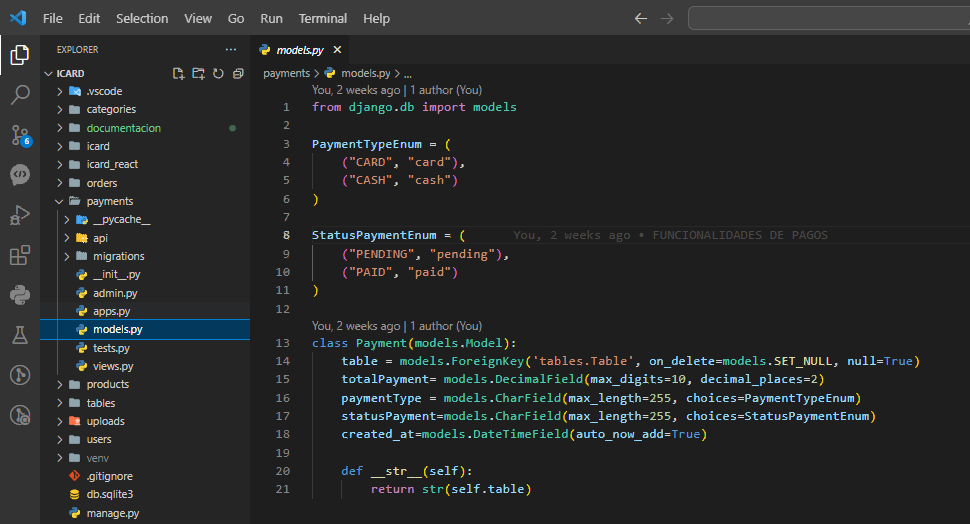
**MODELO DE VISTA DEL MODELADO DE LOS PEDIDOS**

****

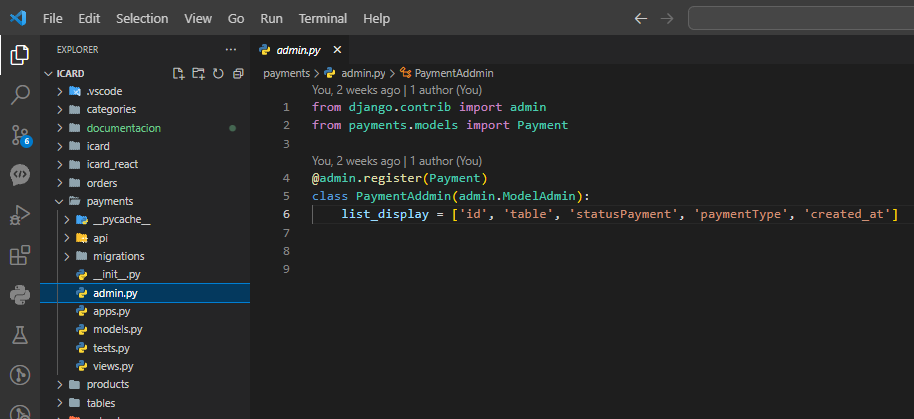
**SERIALIZERS DE LOS PEDIDOS**

****

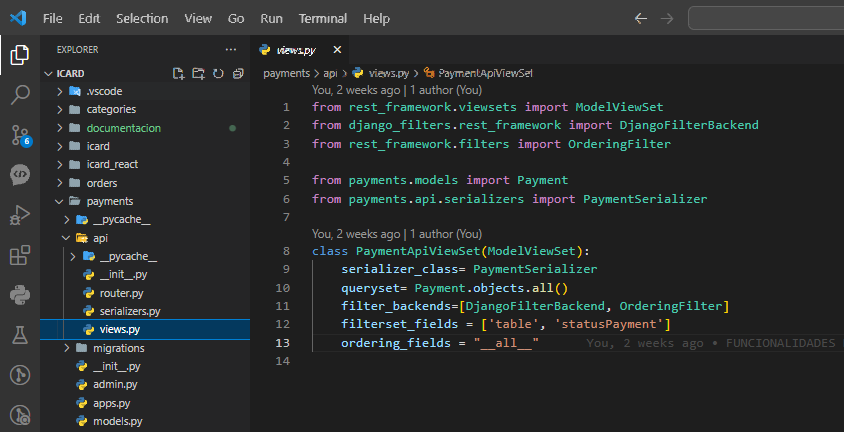
**MODELADO DEL SITEMA DE PAGOS**

****

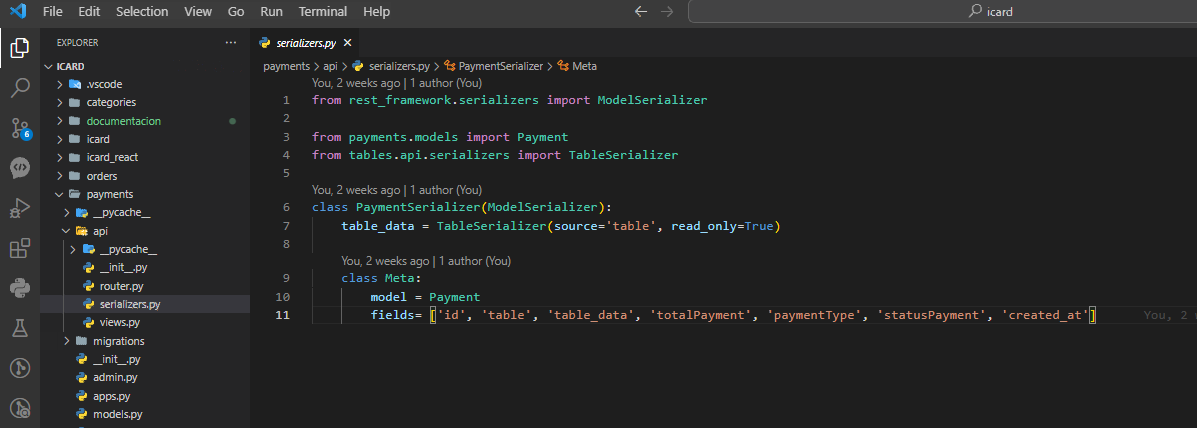
**SISTEMA DE ADMINISTRACIÓN DE PAGOS**

****

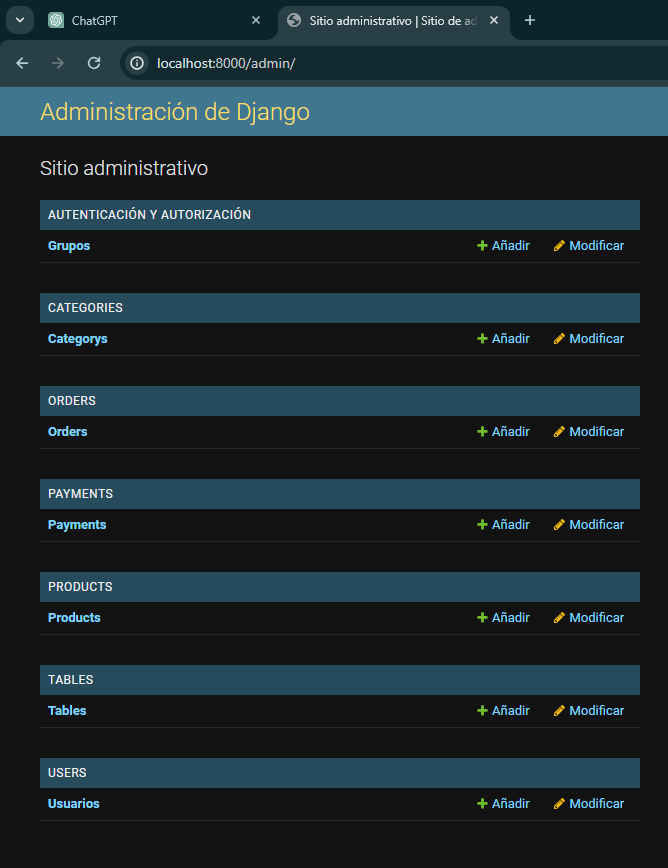
**MODELO DE VISTA DEL SISTEMA DE PAGOS**

****

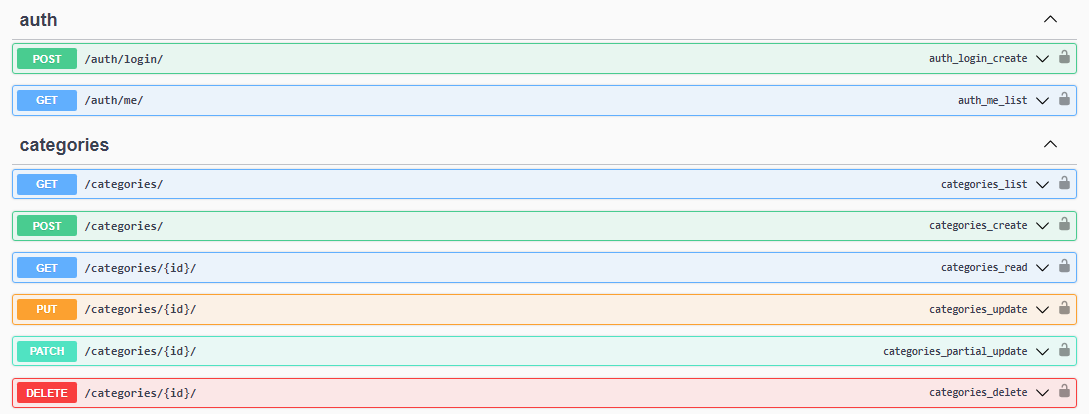
**SERIALIZERS DEL SISTEMA DE PAGOS**

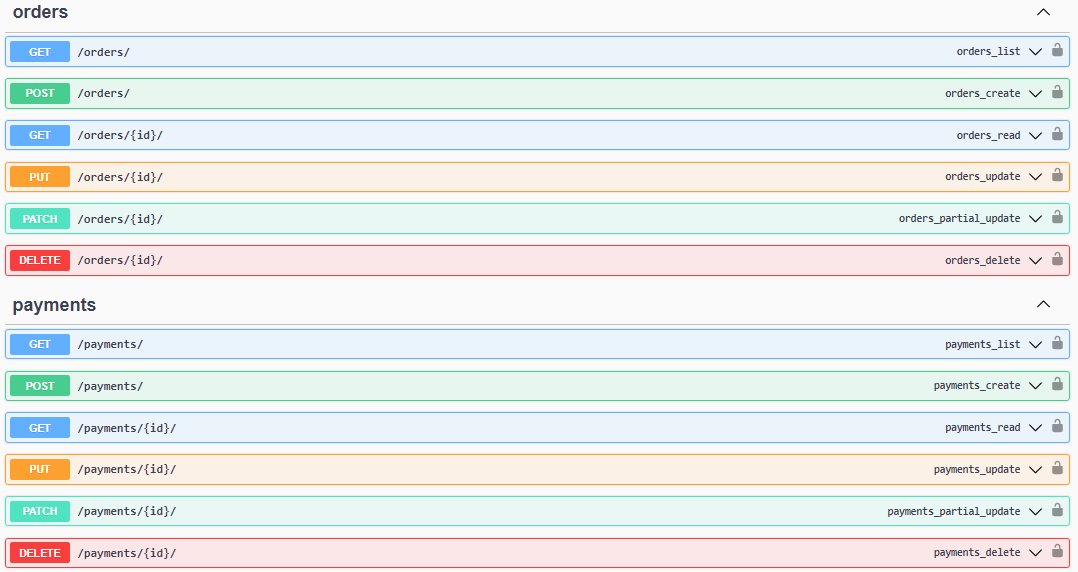
****

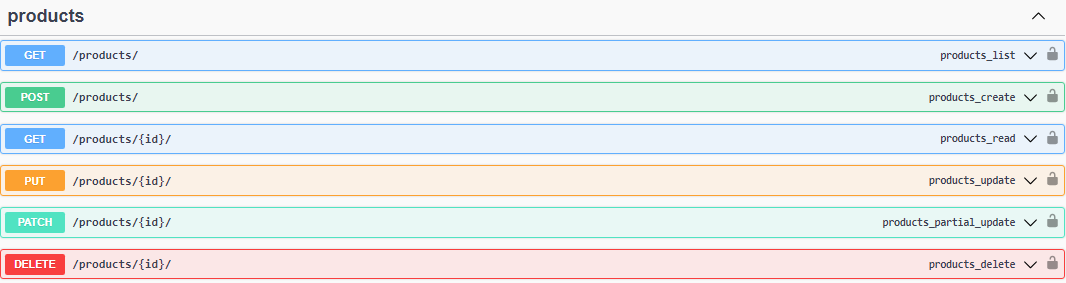
**ADMINISTRACIÓN EN DJANGO:**

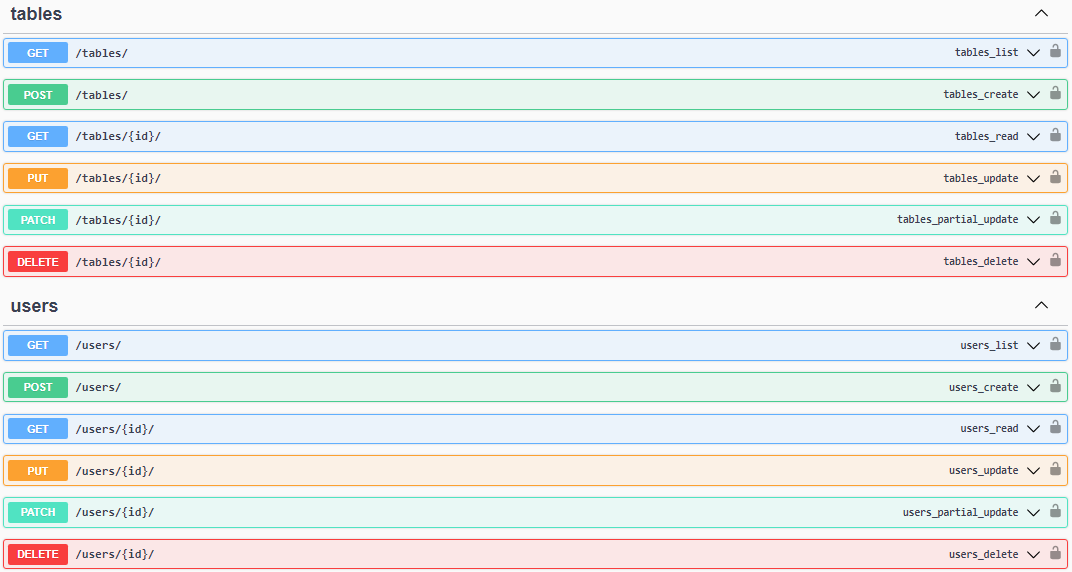
****

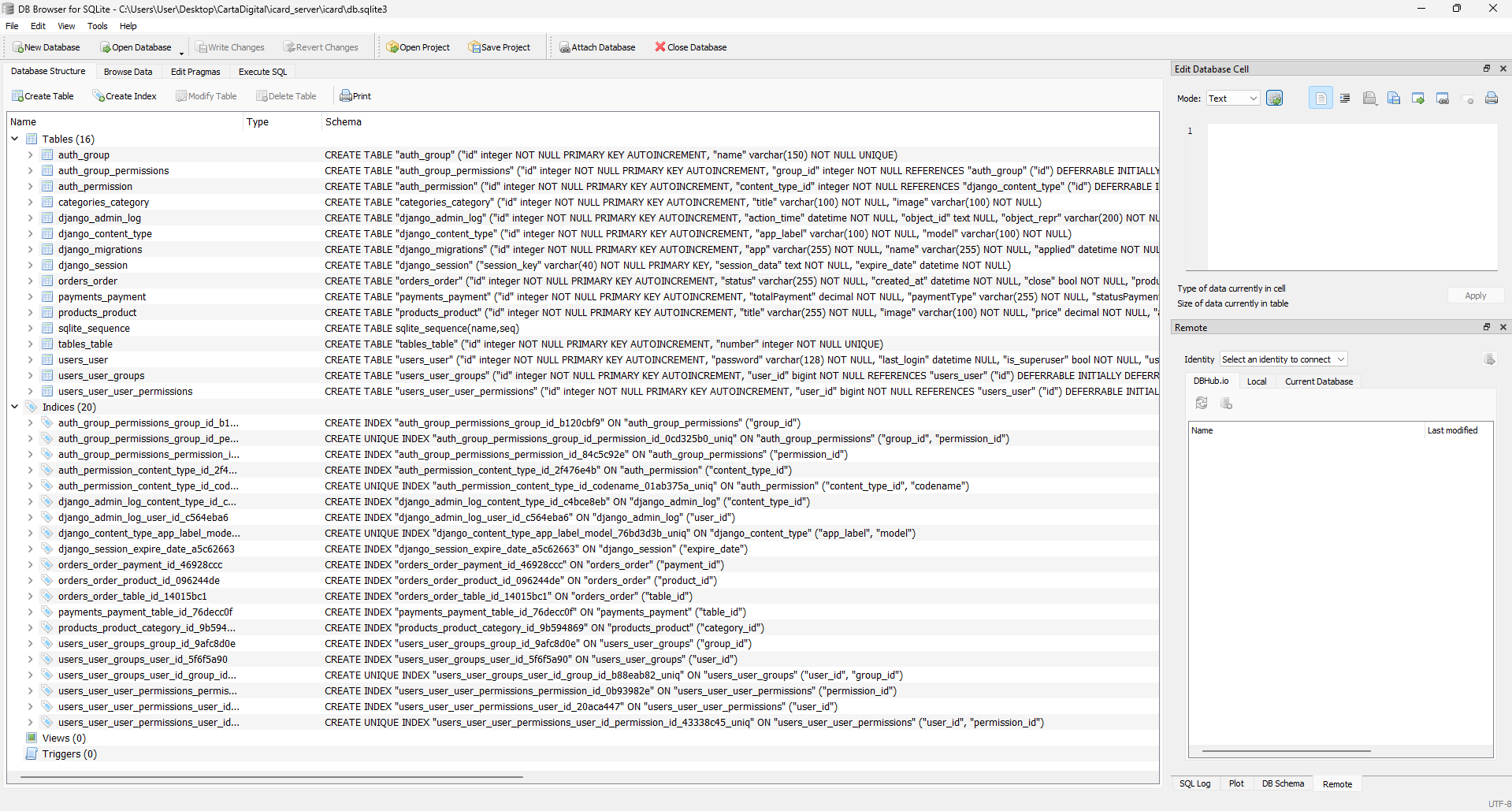
**DOCUMENTACIÓN DE LA API**

****

****

****

****

**CAPTURA DE LA BASE DE DATOS**