**Documentación del proyecto**

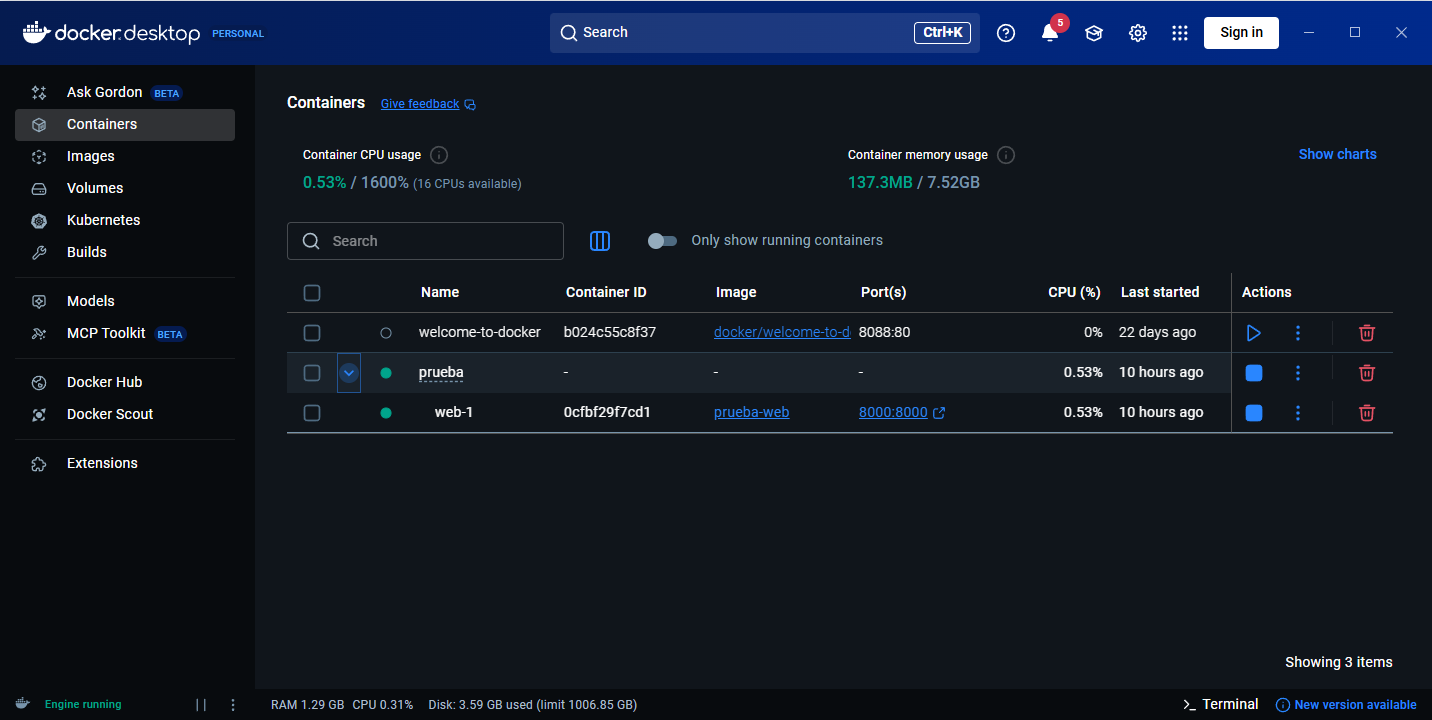
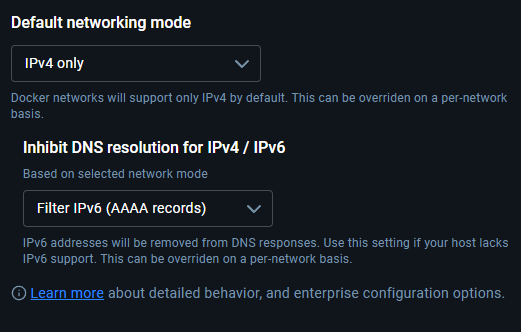
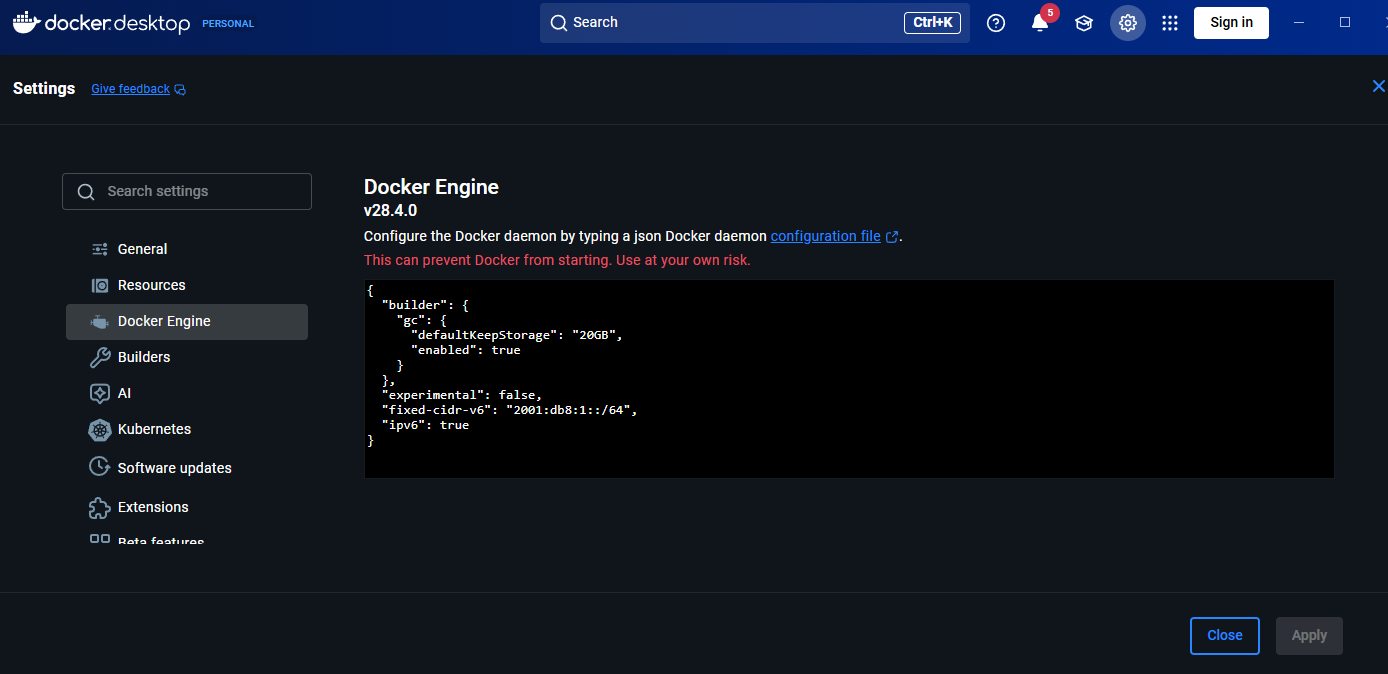
* Para el inicio del proyecto se inició Docker desktop
* Se configura las conexiones para que sean compatibles con el contenedor

1.1- si docker pide actualizar wsl, ir a powershell(admin), instalar (wsl --update), revisar version (wsl --version)

1.2- abrir docker->settings->resources->network->habilitar ipv4 only->activar filter ipv6(AAAA records)

1.3 abrir docker->settings->docker engine-> copiar el siguiente json para el correcto funcionamiento de las redes y

servicios dns.



clonar proyecto

1- ingresar a github

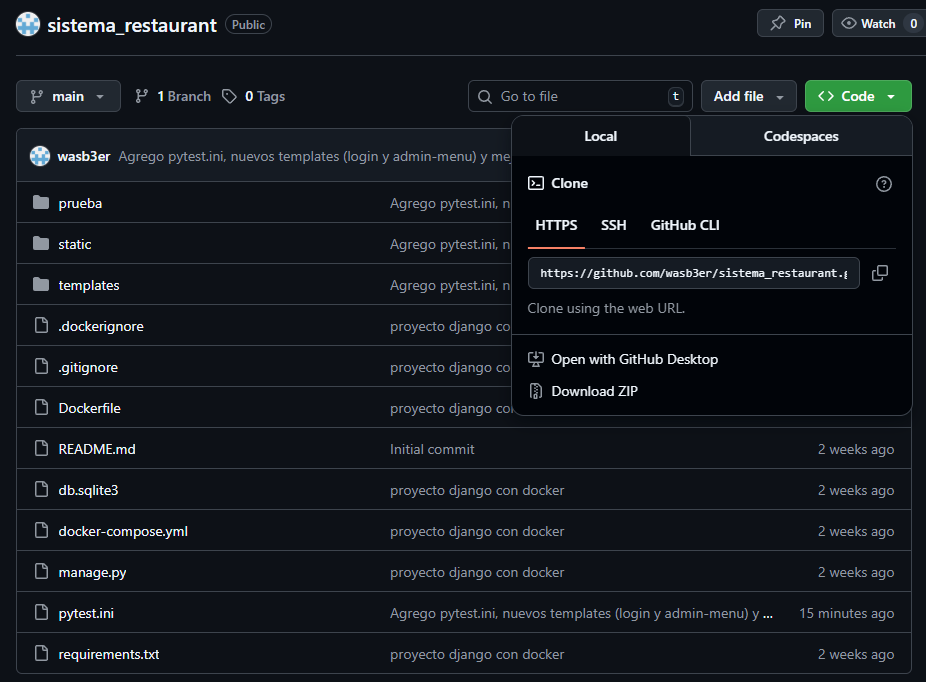
2- copiar el repositorio con gitbash

2.1- git clone https://github.com/wasb3er/sistema\_restaurant.git

2.2- entrar al repo cd sistema\_restaurant

2.3- ubicar carpeta con el proyecto y abrirlo con visual

2.4- crear el .env en la estructura del proyecto ubicado con dockerfile y docker-compose.yml (credenciales en drive - privadas)



Construir la imagen y levantar el contenedor

1- ingresar a la terminal en visual

1.1- docker compose up --build -d (construye el contenedor con los requirements)

2- docker compose run --rm web python manage.py migrate (aplicar migracion si es primera vez)

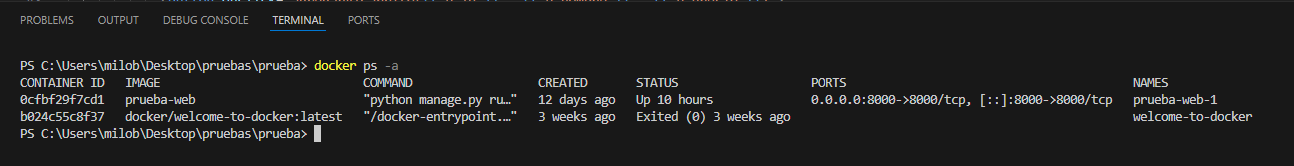
3- docker compose run --rm web python manage.py createsuperuser (crear superusuario para tener acceso con supabase y

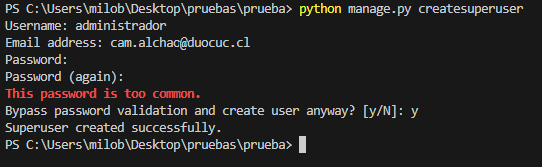
django como admin

3.1- pedira crear el usuario y debera ingresar sus credenciales usuario, correo, contraseña

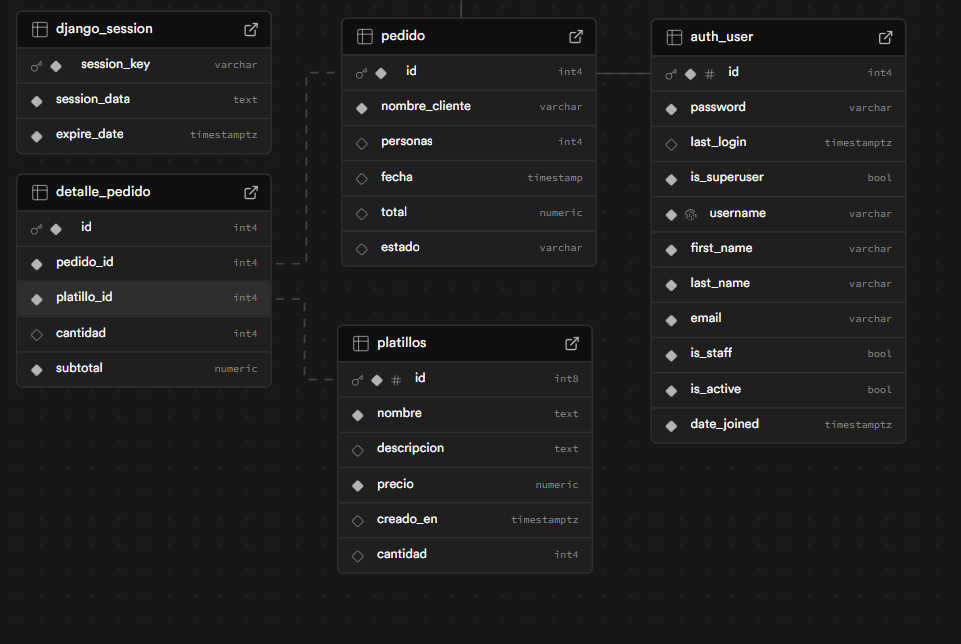
4- docker ps / docker ps -a (ver los contenedores activos, si se encuentra uno activo entrar con localhost:8000)

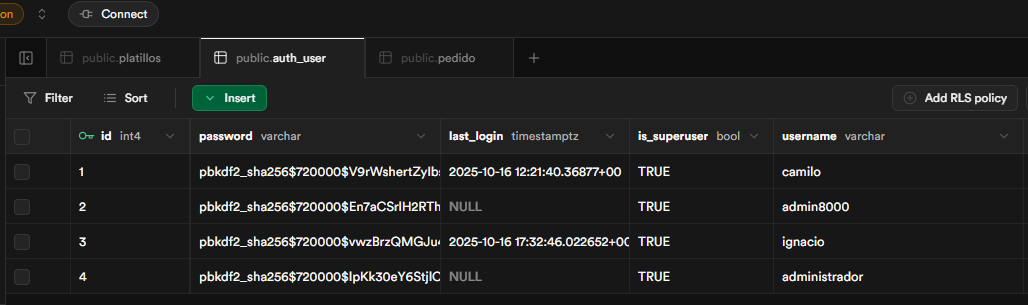
* contenedor ya creado prueba-web-1
* levantar con docker ps -a



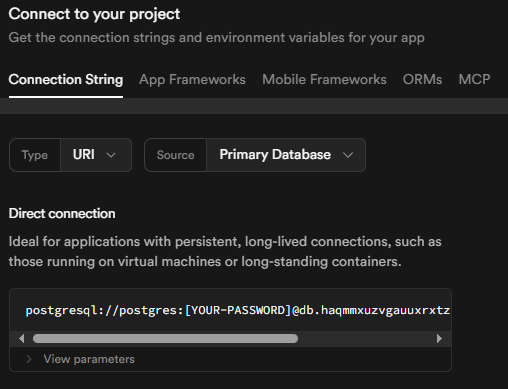


* ver usuario en supabase auth\_user
* tablas de datos creadas para el proyecto





* Crear proyecto en supabase
* Copiar link de base de datos



Entrar a postgreSQL

1- crear otra terminal en visual

2- crear un contenedor que luego se borrara (solo para ver el estado levantado)

3- docker compose run --rm web sh (levanta un contenedor para pruebas, --rm lo elimina despues de apagarlo)

3.1- psql "$DATABASE\_URL" (entrar a psql)

-comandos utiles para trabajar

\dt -ver tablas

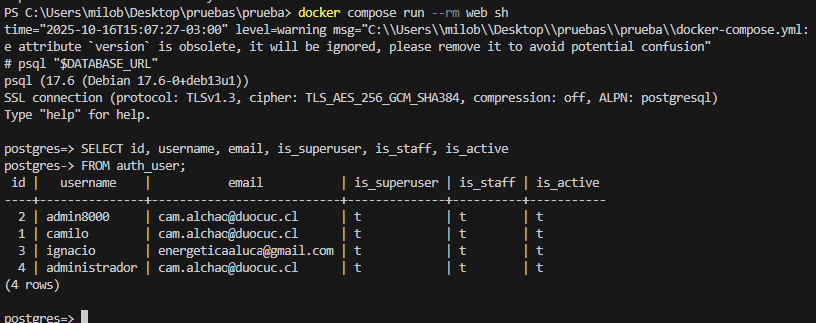
\l -ver base de datos

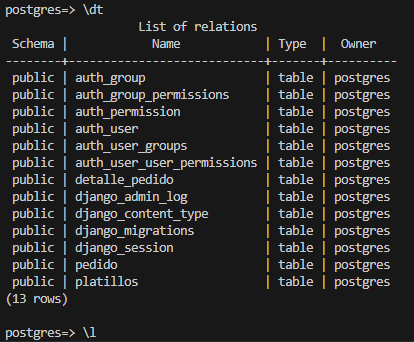
\du -ver usuarios

\q -salir de psql

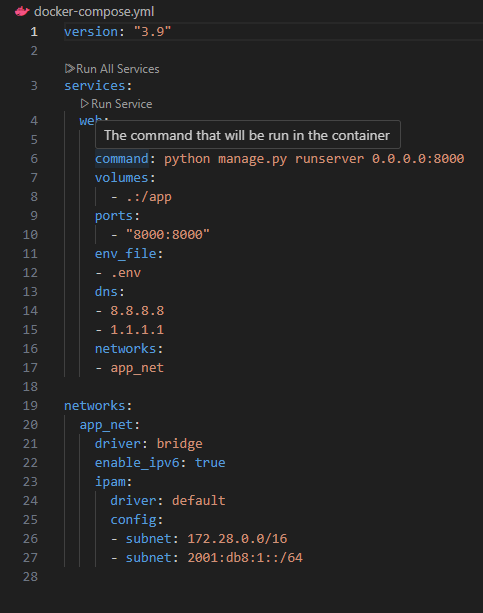
SELECT id, username, email, is\_superuser, is\_staff, is\_active

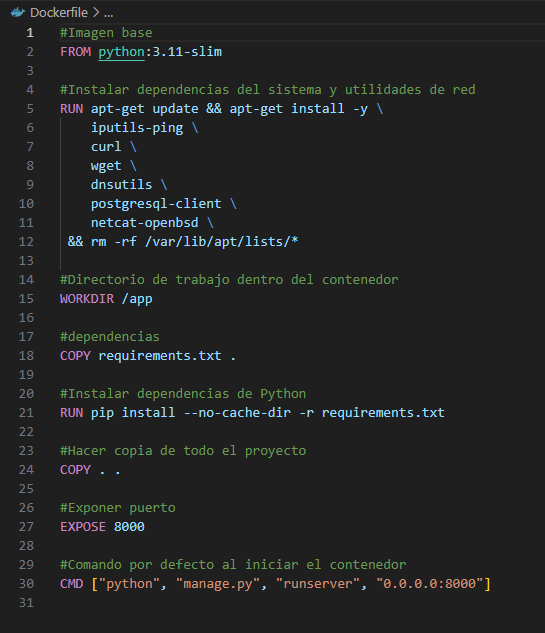
FROM auth\_user; (ve los super usuarios creados, deberian ser 4 si se hizo)

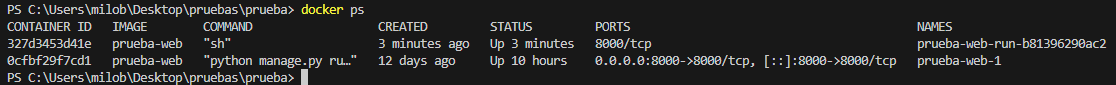




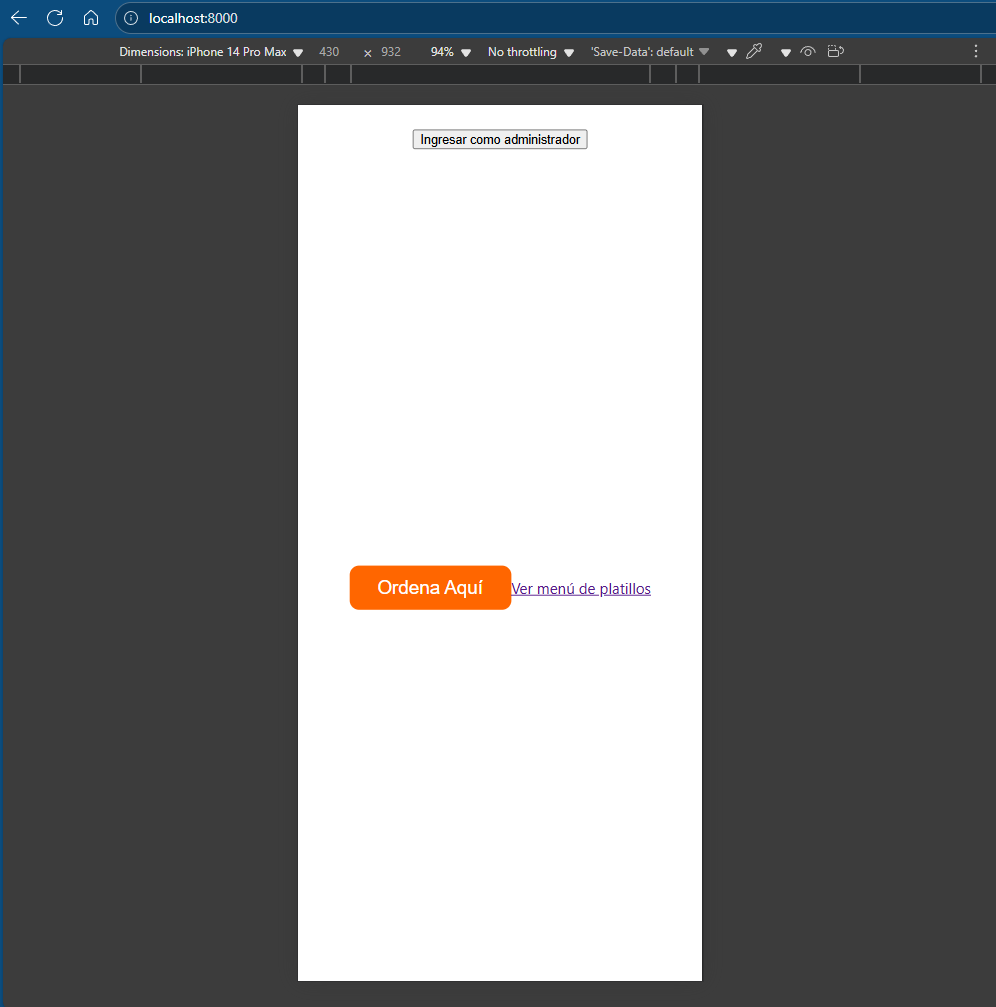
* Iniciar proyecto con docker y django
* ver contenedores docker ps -a
* levantar contenedor: docker start prueba-web-1 o con id docker start 0cf
* crear los documentos necesarios para levantar el contenedor
* crear docker-compose.yml
* crear dockerfile



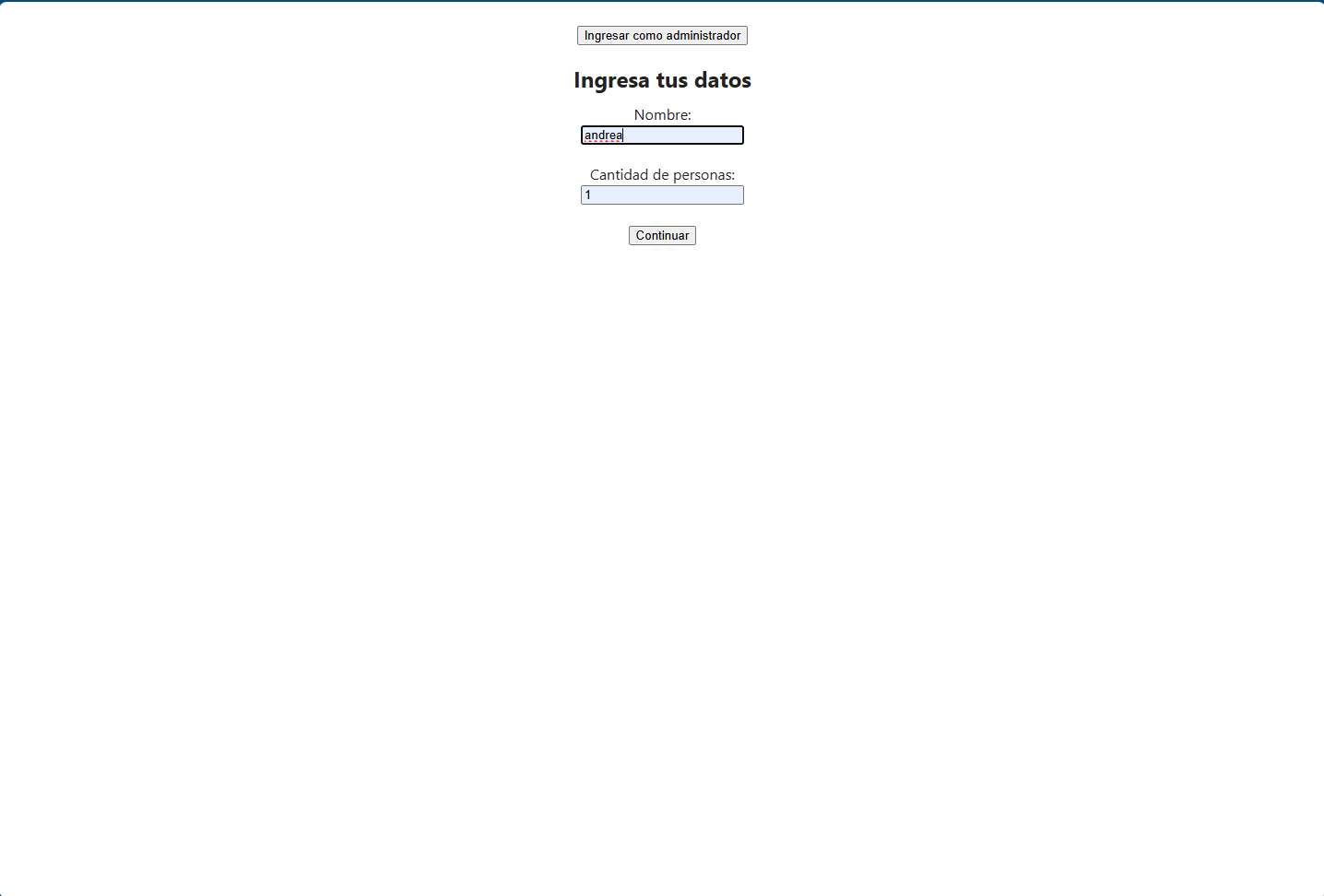




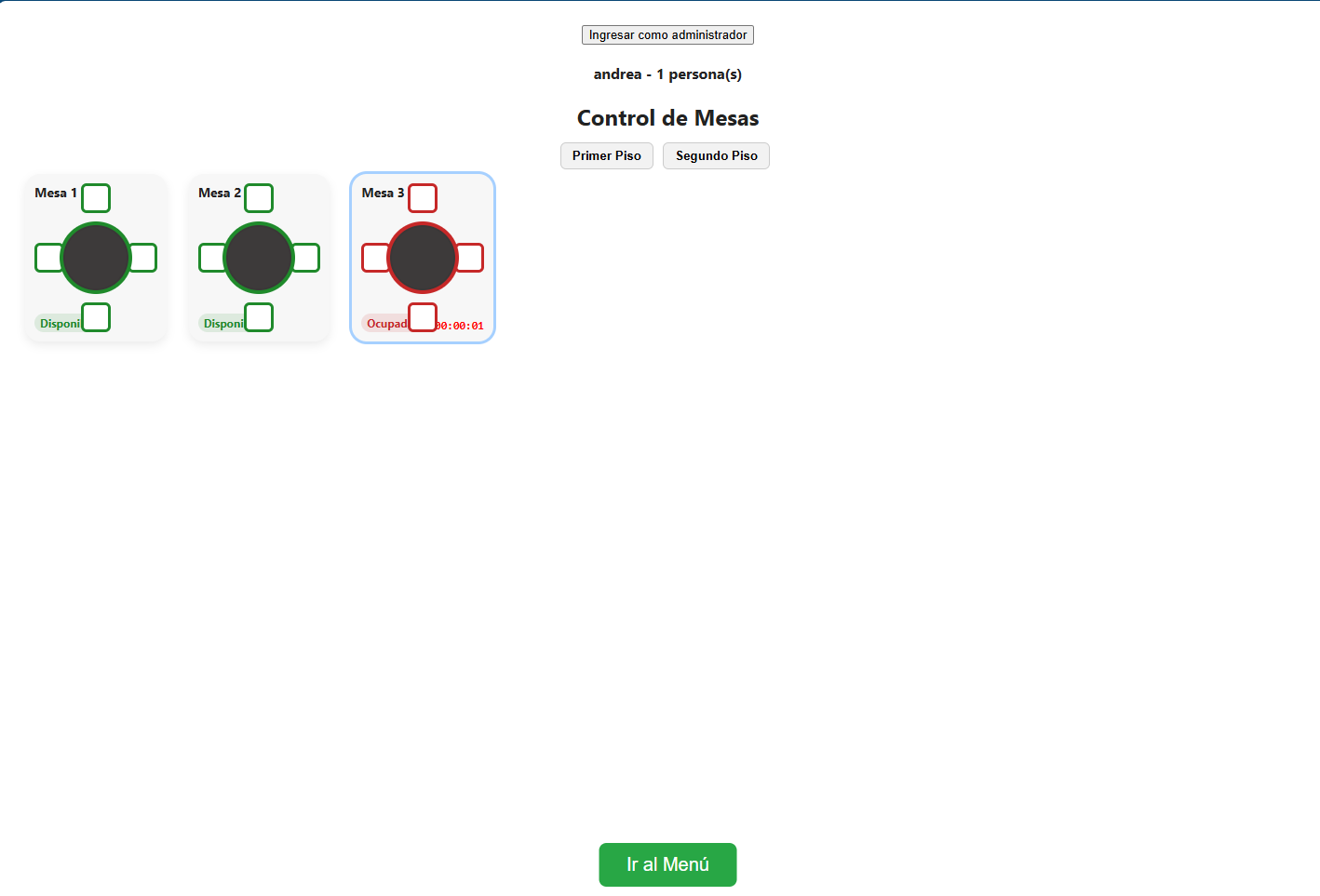
* Ingresar al local por defecto: localhost:8000
* vista responsiva

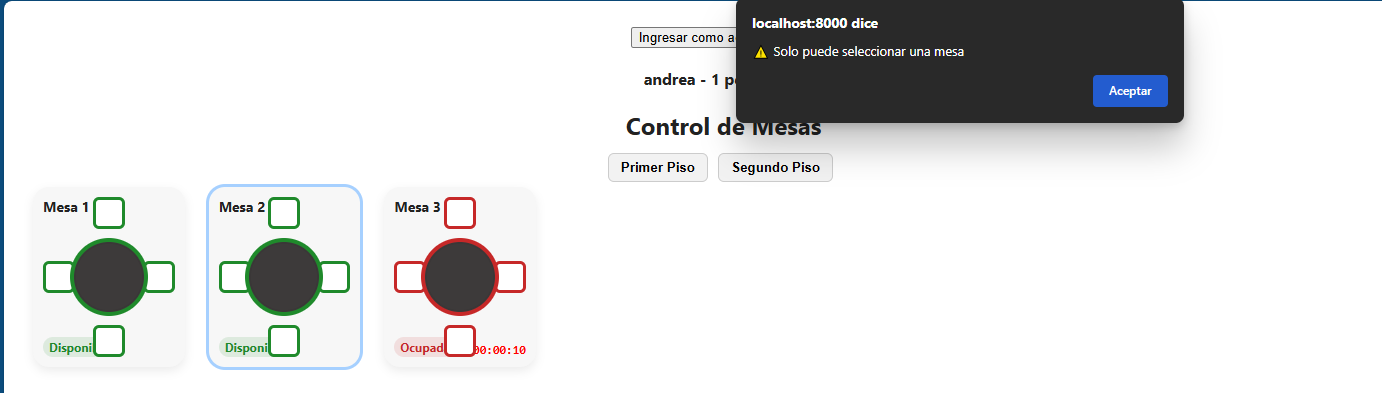


* Ingresar a ordenar menú
* botón ver platillos rápidos
* vista administrador



* Ingresar datos
* ir a seleccionar mesas
* Ir al menú de platillos

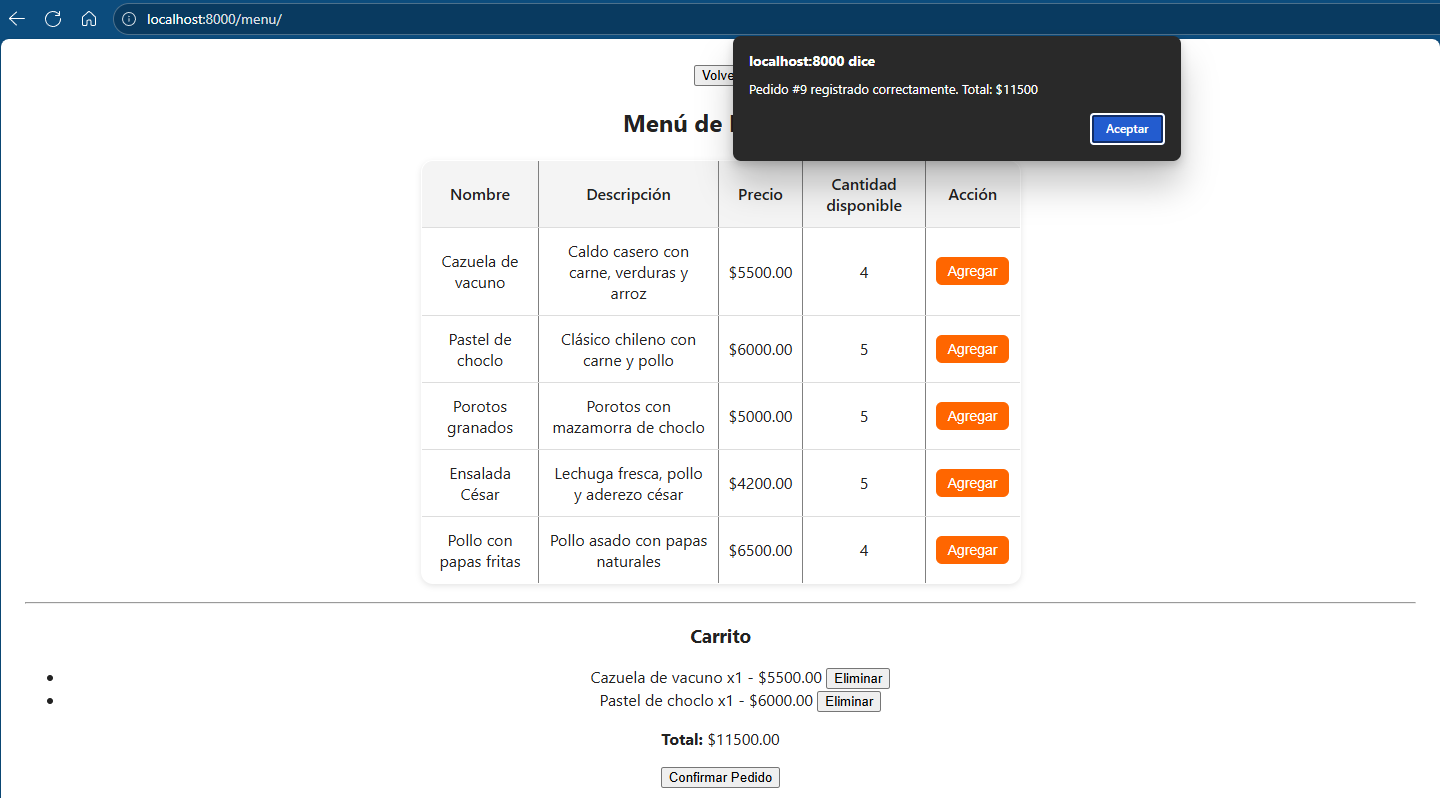


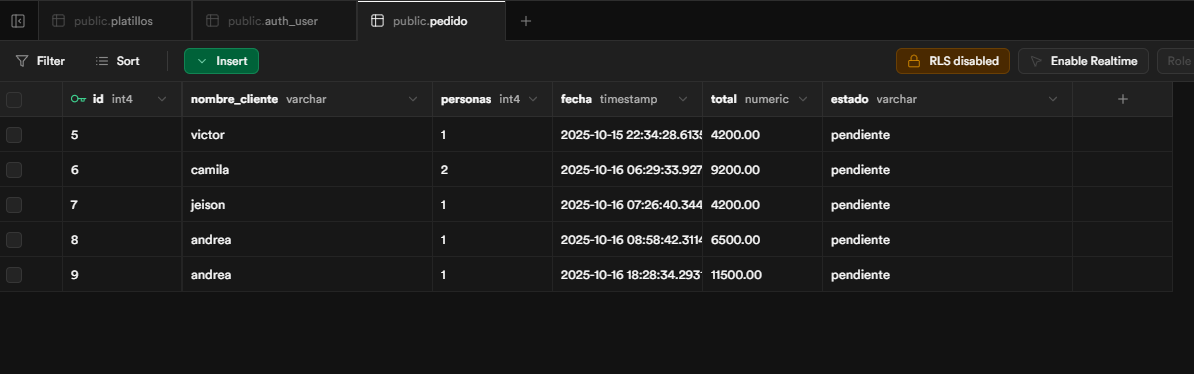




* Seleccionar platillo para agregarlo al carrito
* se actualiza a la base de datos de supabase

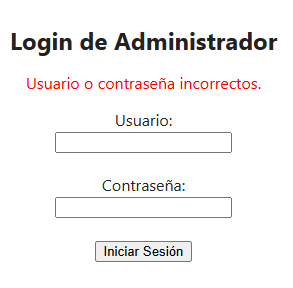


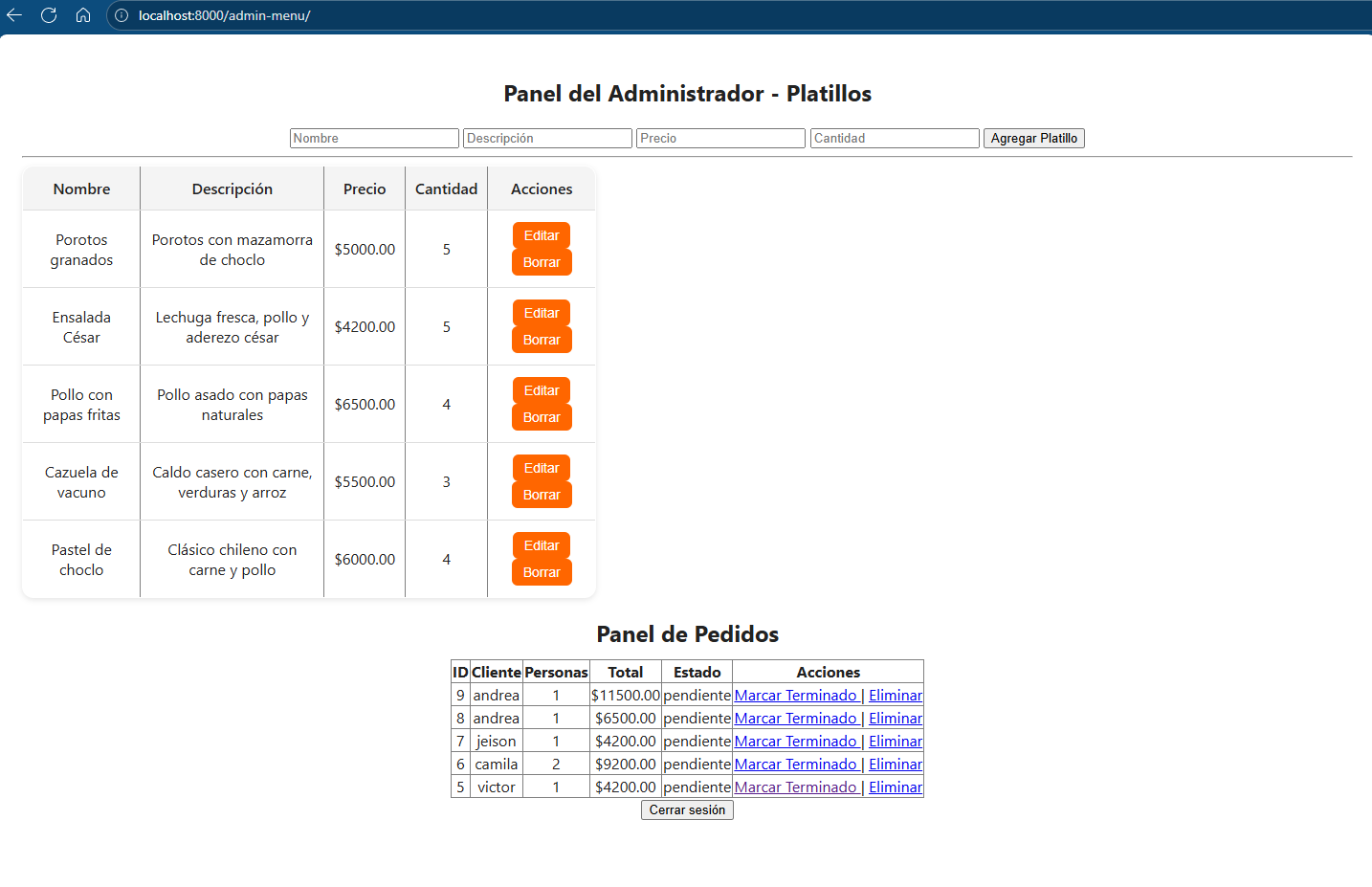




* Entrar a la vista administrador
* Si no esta ingresado con el superusuario no podrá ingresar
* Ingresa a la vista admin
* ve los platillos para modificar
* ve los pedidos pendientes

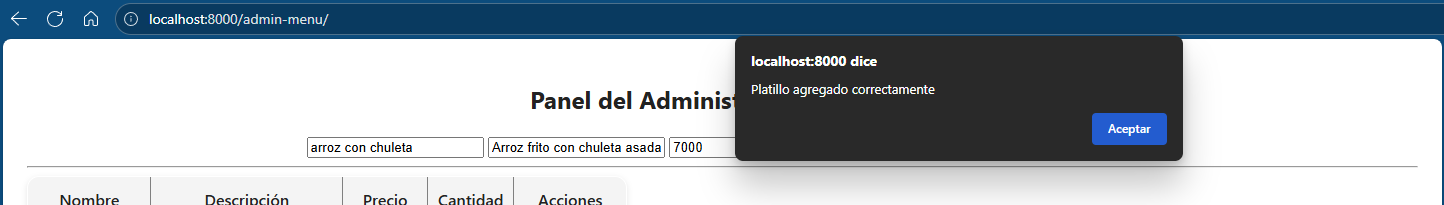




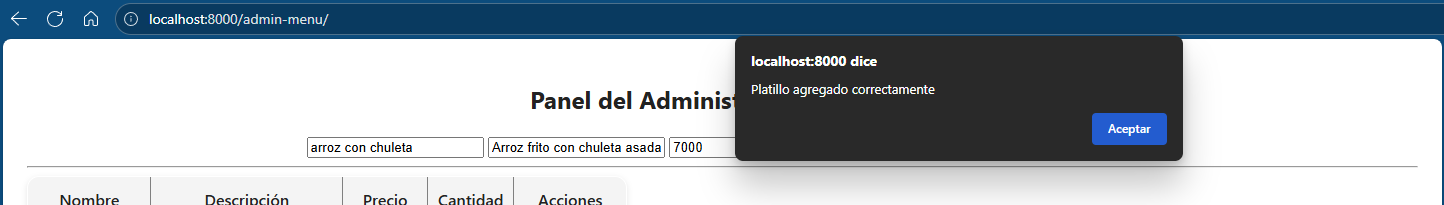


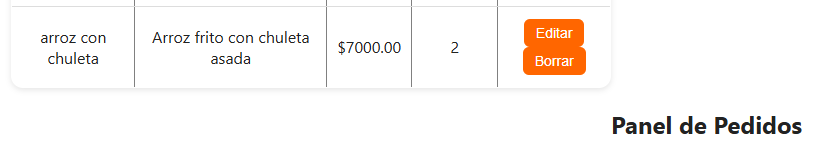
* Agregar un platillo nuevo como administrador
* Ingresar todos los datos
* Ingresar platillo
* pedido ingresado en base de datos

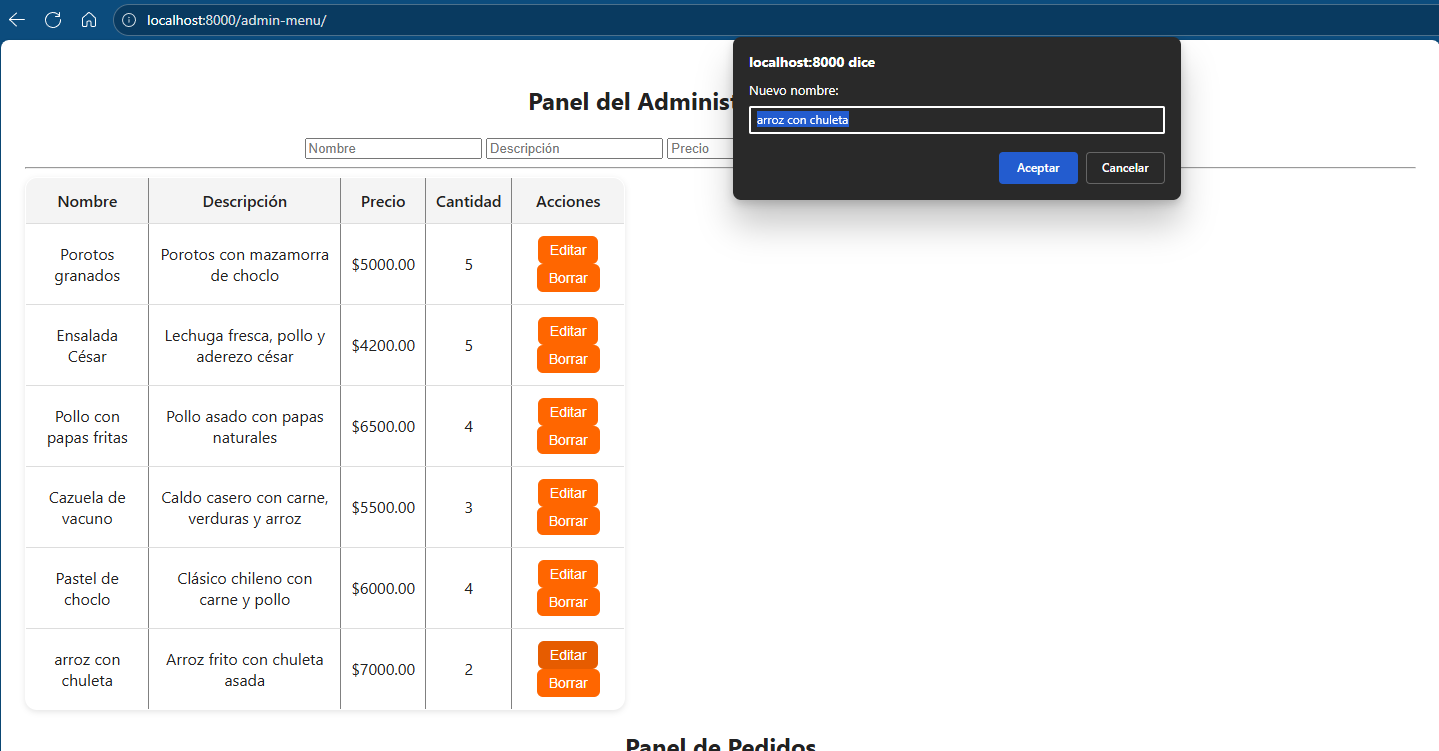




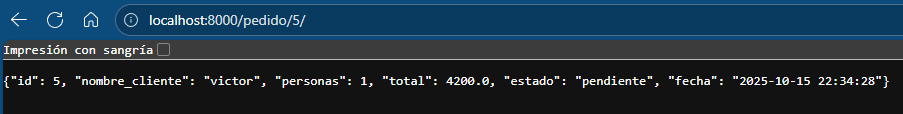


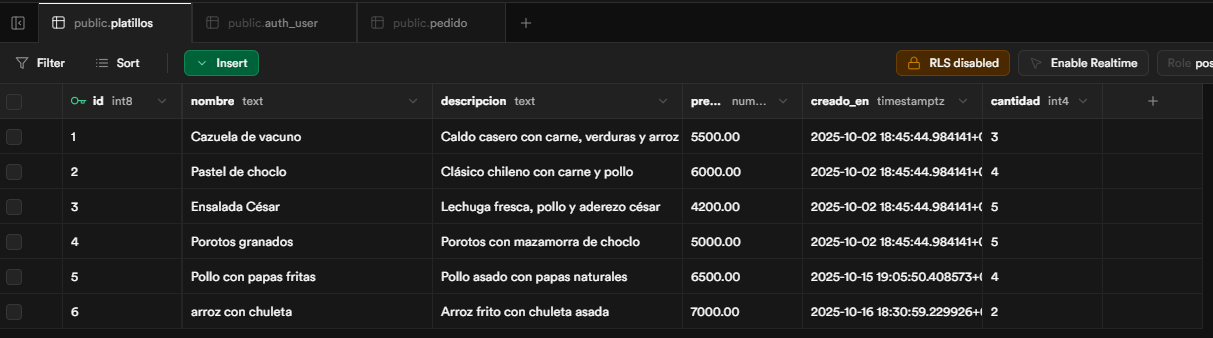






* Ver JSON de pedido como identificador
* platillo agregado

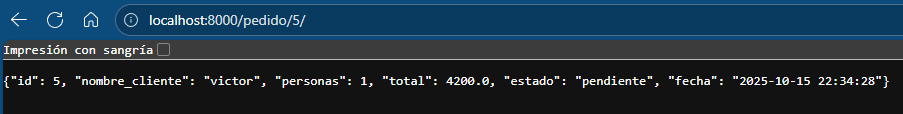




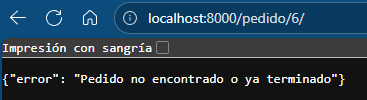
* Ver órdenes pendientes
* Procesar si está listo, si no queda pendiente hasta que el administrador lo elimine
* Si el pedido no existe mostrará un JSON

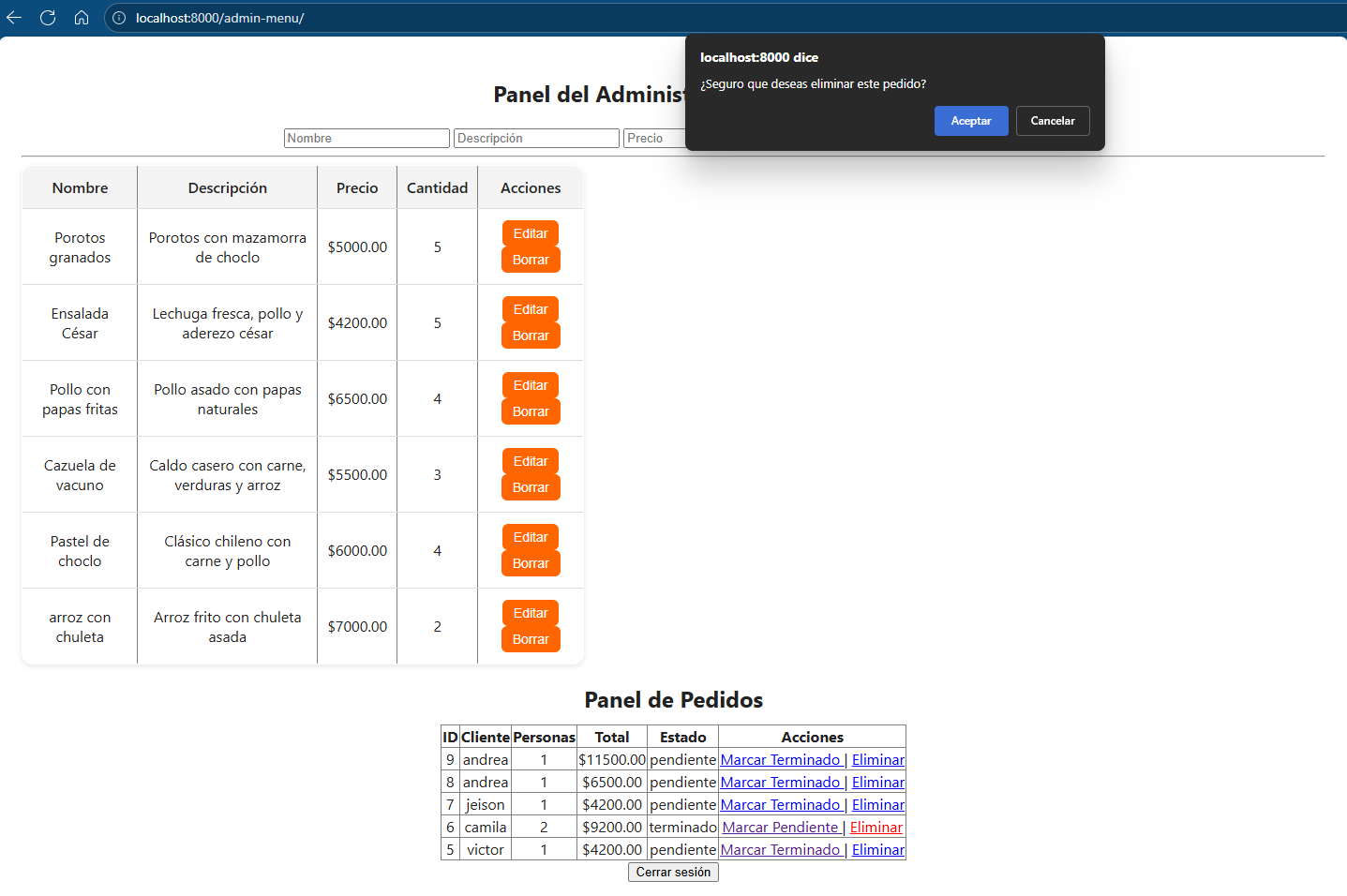


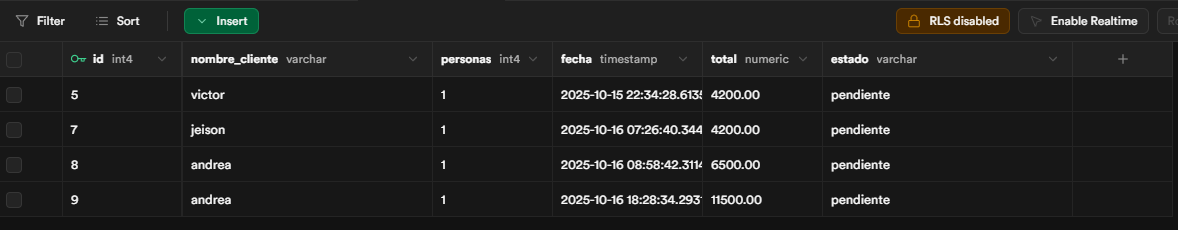




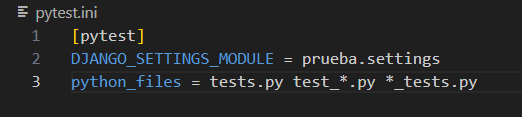


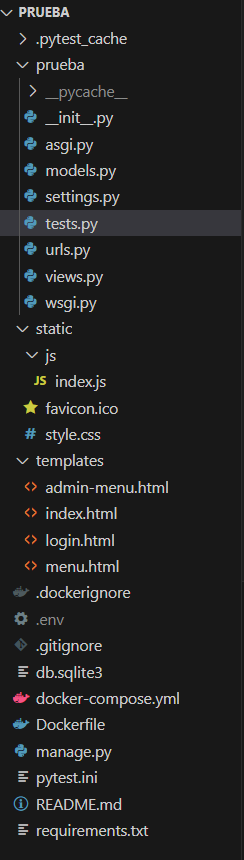


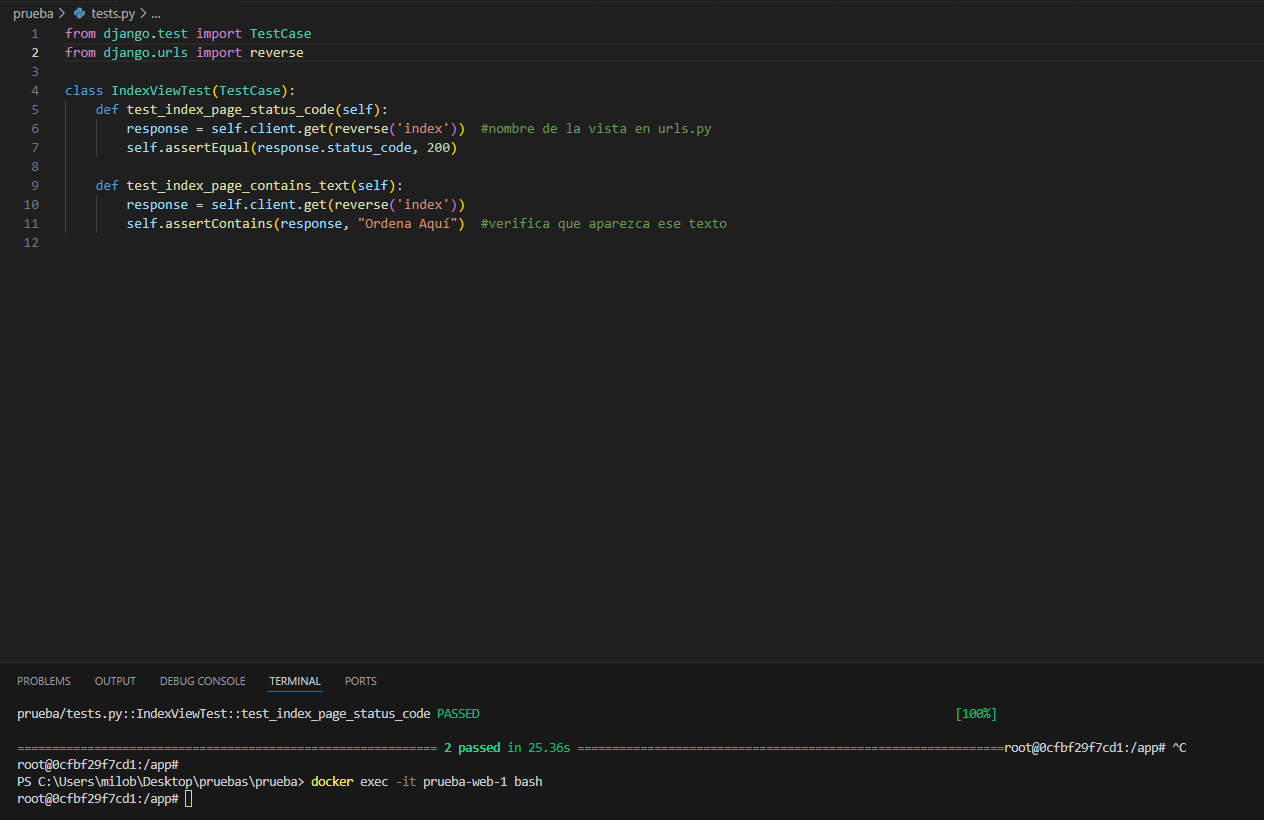


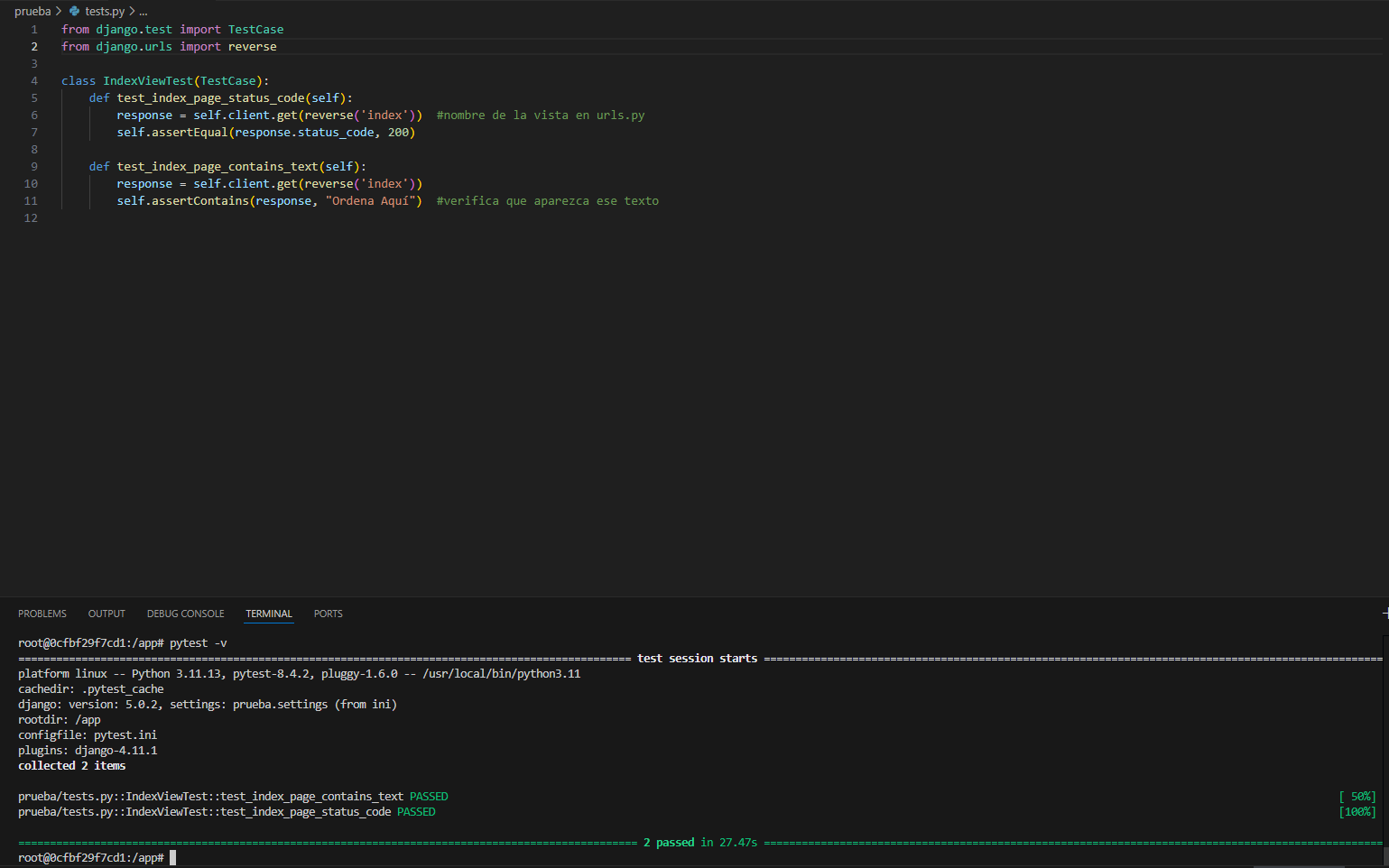


* Ver pruebas con pytest
* Prueba básica para saber si sirve la pagina principal
* crear un pytest.ini en la infraestructura principal
* conectarlo con pytest
* iniciar con docker para hacer pruebas
* docker exec -it prueba-web-1 bash
* ingresa a root
* iniciar prueba pytest -v









* Vista del proyecto css, integrar en la próxima entrega
* Vista responsiva y vista principal

