**Comedor Invisible – Guía de instalación y puesta en marcha (Fase 1 y 2)**

Última actualización: 27 jun 2025  
Estado actual: *stack* funcionando en local (Django splash visible), hypertable creada.

Repositorio github: https://github.com/albertomx2/comedor-invisible.git

**1 · Resumen ejecutivo**

Este documento recoge **todos los pasos** realizados para pasar de un mero frontend estático a un entorno de desarrollo completo con:

* **Docker Compose** orquestando:
  + Django 5 + Gunicorn (servicio backend)
  + TimescaleDB pg17 (servicio timescaledb)
  + Kafka 3.7 + Zookeeper 3.9 (servicios kafka, zookeeper)
  + Nginx reverse‑proxy (servicio nginx)
* Variables de entorno centralizadas (.env)
* Gestión de estáticos vía collectstatic + volumen compartido
* Base de datos inicial con **hypertable** events\_event para trazabilidad y analítica.

**2 · Árbol del repositorio**

comedor-invisible/

├── backend/ # Código Django

│ ├── Dockerfile

│ ├── requirements.txt

│ ├── gunicorn\_config.py

│ ├── core/… # proyecto creado con django‑admin

│ └── users/ # (Fase 3) app a crear

├── nginx/

│ └── conf.d/app.conf # reverse‑proxy (HTTP sólo en dev)

├── docker-compose.yml

├── .env.example

└── README.md

**3 · Prerrequisitos locales**

| **Recurso** | **Versión mínima** | **Comandos de verificación** |
| --- | --- | --- |
| Docker Engine | 24.x | docker --version |
| Docker Compose V2 | 2.25+ | docker compose version |
| Git | 2.40 | git --version |
| WSL 2 (Windows) | – | wsl --status |

**Tip Windows**: instala *Docker Desktop* con la opción “Enable WSL 2 backend”.

**4 · Variables de entorno**

.env.example (copiar a .env y rellenar):

# Django

DEBUG=1

SECRET\_KEY=pon\_aqui\_un\_valor\_largo

ALLOWED\_HOSTS=localhost,127.0.0.1

# Postgres / Timescale

POSTGRES\_DB=comedor

POSTGRES\_USER=postgres

POSTGRES\_PASSWORD=password

POSTGRES\_HOST=timescaledb

POSTGRES\_PORT=5432

# Kafka

KAFKA\_BOOTSTRAP\_SERVERS=kafka:9092

**5 · docker-compose.yml**

[Ver archivo completo en el repo →](https://chatgpt.com/docker-compose.yml)

Puntos clave:

* Volumen static\_volume montado en /app/staticfiles (backend) y /var/www/static/ (nginx).
* Se exponen puertos:
  + 80 → Nginx
  + 5432 → TimescaleDB
  + 9092 → Kafka (plaintext)
* Línea version: eliminada (Compose V2 la ignora).

**6 · Dockerfile del backend**

Basado en python:3.11-slim. Instala GCC + libpq-dev, copia requirements.txt y ejecuta pip install.

**7 · backend/requirements.txt**

Django==5.0.3

# API

djangorestframework==3.15.1

djangorestframework-simplejwt==5.2.2

# Config env y DB

django-environ==0.11.2

dj-database-url==2.2.0

psycopg2-binary==2.9.9

# CORS & websockets

django-cors-headers==4.3.1

channels==4.1.0

channels-redis==4.2.0

# Broker

kafka-python==2.0.2

# WSGI

gunicorn==22.0.0

**8 · Creación del proyecto Django**

# levantar contenedor temporal y crear proyecto en carpeta backend

docker compose run --rm backend django-admin startproject core .

**Cambios en core/settings.py**

* **Lectura de .env** via django-environ y dj\_database\_url.
* Configuración de DATABASES apuntando a TimescaleDB.
* Añadido corsheaders y rest\_framework a INSTALLED\_APPS.
* STATIC\_ROOT = BASE\_DIR / "staticfiles".

**Gunicorn**

Archivo backend/gunicorn\_config.py:

bind = "0.0.0.0:8000"

workers = 4

threads = 2

**9 · Nginx (dev)**

nginx/conf.d/app.conf sirve /static/ desde /var/www/static/ y proxy‑pasa al backend.

**10 · Comandos rutinarios**

# reconstruir imagen backend (sin cache)

docker compose build --no-cache backend

# levantar todo en segundo plano

docker compose up -d

# seguir logs en vivo

docker compose logs -f backend

**11 · Problemas encontrados y solución**

| **Error** | **Causa** | **Fix** |
| --- | --- | --- |
| docker: command not found | Docker CLI no instalado / PATH | Instalar Docker Desktop + reiniciar PowerShell |
| ModuleNotFoundError: dj\_database\_url | Falta en requirements.txt | Añadido al archivo + rebuild |
| ImproperlyConfigured: STATIC\_ROOT | No definido | STATIC\_ROOT = BASE\_DIR / 'staticfiles' + volumen correcto |
| gunicorn: not found (exit 127) | No instalado | Añadido gunicorn a requirements + rebuild |
| 502 Bad Gateway (nginx) | Backend caído | Revisar logs backend; corregir fallos anteriores |

**12 · TimescaleDB – creación de la hypertable events\_event**

**Acceso desde PowerShell**

docker compose exec timescaledb psql -U postgres -d comedor

**SQL ejecutado**

CREATE EXTENSION IF NOT EXISTS timescaledb;

CREATE TABLE events\_event (

time TIMESTAMPTZ NOT NULL,

event\_type TEXT NOT NULL,

key\_id INT,

payload JSONB NOT NULL,

location TEXT

);

SELECT create\_hypertable('events\_event', 'time',

chunk\_time\_interval => interval '7 days',

migrate\_data => true);

SELECT add\_dimension('events\_event', 'location', number\_partitions => 8);

**13 · Estado final de la Fase 2**

* http://localhost/ responde con la página de Django (DEBUG =True).
* Contenedores backend, timescaledb, kafka, zookeeper, nginx corriendo.
* Hypertable creada y lista para recibir eventos.

**14 · Próximos pasos (Fase 3)**

1. **App users**:
   * Modelo CustomUser + rating
   * Endpoints /api/auth/register/, /login/, /me/ (JWT)
2. **App dishes** y **reservations**
3. Publicación de eventos Kafka y consumer que inserte en events\_event.
4. Migración del frontend a **React + Vite** y conexión con la API.

*Documento vivo*: actualizaremos esta guía con cada nueva fase para que sirva de referencia completa del proyecto.