**📆 Semana 3 – Fundamentos Backend (Django REST Framework)**

**Clase 3 (4h) – Django + API básica**

**Objetivos:** Configurar un proyecto Django y exponer primer endpoint.

1. **Arquitectura Cliente-Servidor aplicada al Marketplace (45 min)**
   * Diagrama en pizarra: Frontend (Django templates / Kivy) → API REST (Django REST Framework) → BD (Postgres).
   * Explicar flujo de request/response con ejemplo simple.
2. **Instalación y Configuración Django (45 min)**
   * pip install django djangorestframework psycopg2-binary.
   * django-admin startproject marketplace.
   * Configuración inicial: settings, urls.
3. **Primer API REST (1h 15 min)**
   * Crear app users.
   * Modelo básico: User (username, email, password).
   * Serializer + ViewSet + Router.
   * Endpoint /api/users/ con métodos GET y POST.
4. **Prueba local con Postman (1h 15 min)**
   * GET a /api/users/.
   * Crear nuevo usuario con POST.
   * Verificar en admin de Django.

**Entrega:** Proyecto Django en repo con API de usuarios funcionando localmente.

**Clase 4 (4h) – PostgreSQL + Deploy parcial**

**Objetivos:** Conectar con base de datos PostgreSQL y practicar deploy parcial.

1. **Repaso breve (20 min)**
   * ¿Qué hicimos en la última clase?
   * 2 alumnos muestran cómo hicieron el POST.
2. **Conexión con PostgreSQL (1h 15 min)**
   * Crear DB en Render (PostgreSQL free tier).
   * Configurar DATABASES en settings.py.
   * Migraciones (python manage.py migrate).
   * Probar API de usuarios contra la nueva DB.
3. **Buenas prácticas (1h 15 min)**
   * Variables de entorno (.env).
   * Librería python-decouple.
   * No subir credenciales al repo (explicar .gitignore).
4. **Deploy parcial del backend (1h)**
   * Subir backend a Render.
   * Probar endpoint /api/users/ desde la nube.
   * Cada grupo comparte URL pública en el repo.

**Entrega:** API de usuarios corriendo en Render + repo actualizado.

👉 Con estas 4 clases, tus alumnos ya van a tener:

* Un **flujo de trabajo en GitHub** con PRs y colaboración.
* Un **backend inicial en Django REST** con base de datos PostgreSQL en Render.