Overview:

Django REST Framework (DRF)

- DRF is a powerful and flexible toolkit built on top of Django, designed for building Web APIs.
- It integrates deeply with Django's ORM, admin panel, and built-in features like authentication, sessions, and forms.

FastAPI

- FastAPI is a modern, high-performance web framework for building APIs with Python 3.6+ using async I/O.
- It's built on top of Starlette for web handling and Pydantic for data validation.
- Automatically generates OpenAPI and Swagger documentation.

Features:

DRF:

Base Framework:	Django (WSGI-based)
ORM:	Django ORM (rich, full-featured)
Serialization:	Serializer classes (explicit structure)
Routing:	URL patterns via Django's URL config
Authentication:	Built-in (session, token, JWT via extensions)
Admin Interface:	Full-featured and automatic
Docs Generation:	Manually via extensions or third-party packages
Validation:	Inside Serializers
Async Support:	Limited async support (in newer Django versions)
Background Tasks:	Not built-in (requires Celery or other)

FastAPI:

Base Framework:	Starlette (ASGI-based, async ready)
ORM:	Optional (commonly used with SQLAlchemy/Tortoise)
Serialization:	Pydantic models (Pythonic and concise)
Routing:	Decorators on path functions
Authentication:	JWT, OAuth2, custom – fewer built-in options
Admin Interface:	None built-in
Docs Generation:	Automatic Swagger UI & ReDoc out-of-the-box
Validation:	Type-hint-based with Pydantic
Async Support:	Native async support with async def
Background Tasks:	Built-in support via BackgroundTasks

Advantages:

Django REST Framework

- **Mature Ecosystem**: Part of Django's long-standing ecosystem with lots of community support.
- **Admin Panel**: Automatically generated and great for CRUD operations.
- **Authentication & Permissions**: Comprehensive out of the box (user model, auth backends, permissions).
- **Powerful ORM**: Django ORM makes database interactions intuitive and efficient.
- **Browsable API**: Auto-generated web UI for testing endpoints – great for development.

FastAPI

- **High Performance**: Built for speed, thanks to async support and Uvicorn.
- **Type Hinting = Auto Docs**: OpenAPI schema and documentation is generated from your Python code.
- **Asynchronous by Design**: Scales better for concurrent IO-bound operations (like APIs calling databases or other services).
- **Less Boilerplate**: Code is concise, especially with Pydantic for validation and automatic docs.
- **Built-in Dependency Injection**: Clean architecture and better testability.

Disadvantages:

Django REST Framework

- **Slower Performance**: Being synchronous and WSGI-based, it's less performant for concurrent APIs.
- **Verbose**: More boilerplate, especially when building simple APIs (serializers, views, routers).
- **Async is Limited**: While Django 3.1+ supports async, DRF itself isn't fully async-ready.
- **Not Ideal for Microservices**: It can be heavyweight for lightweight or microservice architectures.

FastAPI

- **Young Ecosystem**: While growing rapidly, it's still newer and may lack some mature tools.
- **No Admin Panel**: You'll need to build your own or use an external solution.
- **Steep Learning Curve for Async**: Beginners may struggle with async/await, concurrency, and dependency injection.
- **Less Built-in**: Things like authentication, permission classes, pagination require more setup or third-party tools.

Challenges and Solution I Have Faced:

Both:

- Problem with CORS when connecting the frontend to the backend
    - Solution: I installed CORS both on the backend and then add the links for the frontend as well as the localhost for testing

- Problem with vboxapi version
    - Solution: I deleted it in the requrements.txt since it is not needed because I'm not running on a system with VirtualBox

FastAPI:

- 'orm_mode' has been renamed to 'from_attributes'
    - I changed orm_mode into from_attributes = True

- No open ports detected on 0.0.0.0
    - I adjusted the start command to this "uvicorn main:app --host 0.0.0.0 --port $PORT"

DRF:

- Problems with finding or creating the requirement.txt
    - Solution: I run the command "pip freeze > requirements.txt
- Issues with errors not showing when in deployment
    - Solution: I enabled Debug as true to know what is the error
- Problem with AuthToken
    - Solution: I added 'rest_framework.authtoken', on Installed Apps
- Problem with PostgresURL
    - Solution: I copied the wrong URL internal instead of External
- Error in start command when deploying since I didn't specify the name of my app
    - Solutin: I added my app in command: gunicorn myproject.wsgi

Frontend: https://whimsical-parfait-dac91a.netlify.app/

Backend: https://todo-fastapi-fm3h.onrender.com/docs