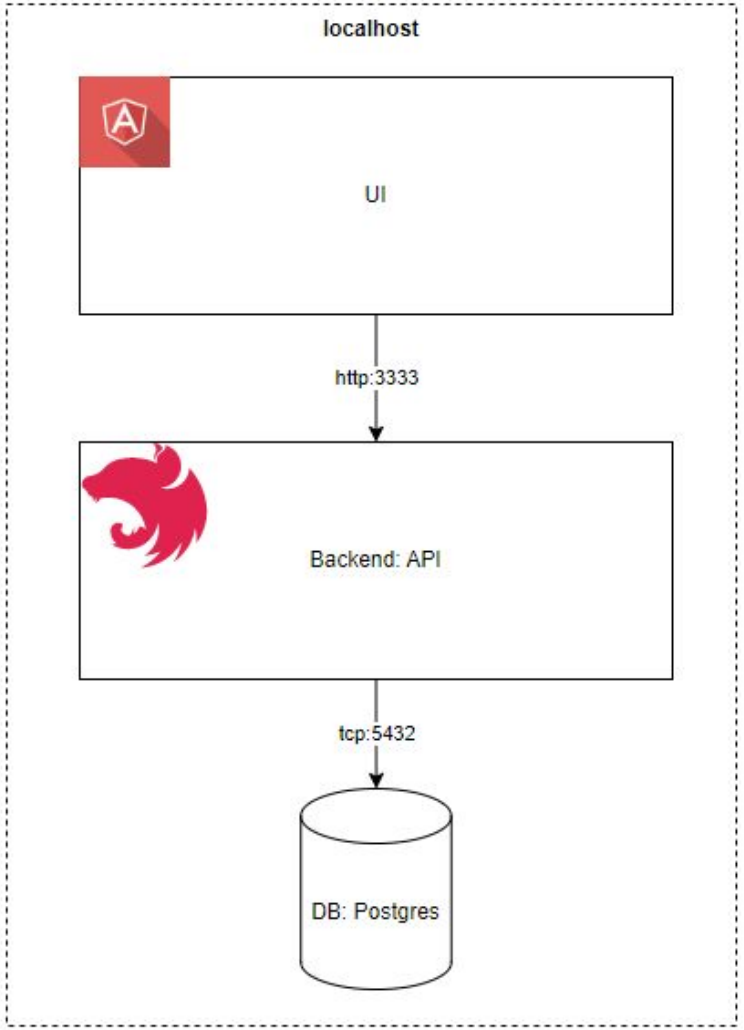
FullStack #WebDevelopment Bootcamp

Course Week 3

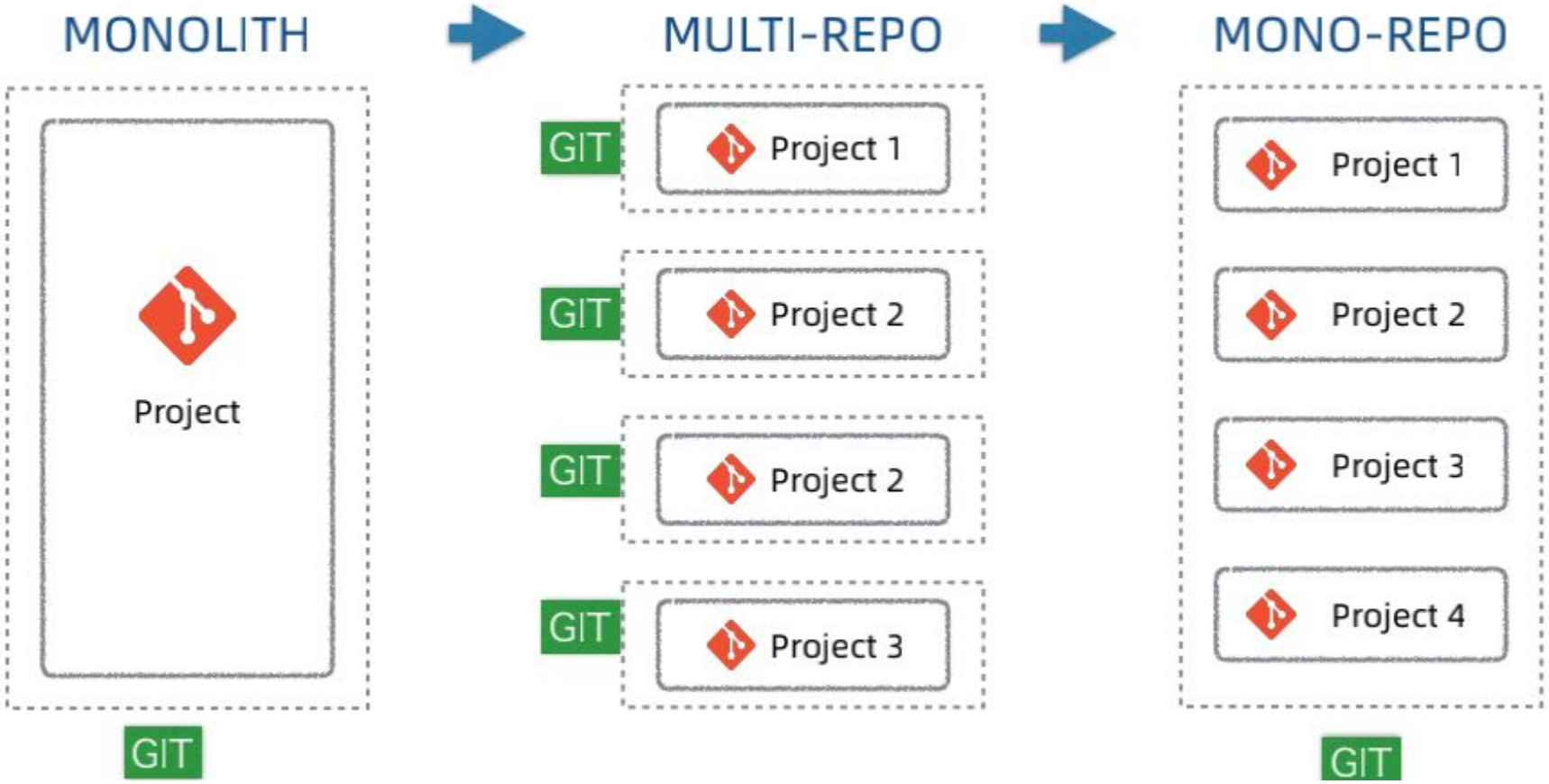
Remembrance: Structure of our application



Typescript Styling

- UpperCamelCase => class/interface/type/enum/decorator
- lowerCameCase => variable/parameter/function/property/method
- File Names:
  - If its a single class/controller/service
    - say: “ObservableLoggingService” => “observable-logging.service (.service since its a special keyword in our framework)
  - If its a collection of types (multiple classes/types/interfaces)
    - usually I have <collection-name>.types
      - e.g.: authentication.types
- Variables/Function/Names should answer the question of “why do I exist?”
- **Do not leave comments in your code to explain anything => the code itself should be readable by anyone**

Mono-Repository vs Multi-Repository



Nx: Mono-Repository Tool

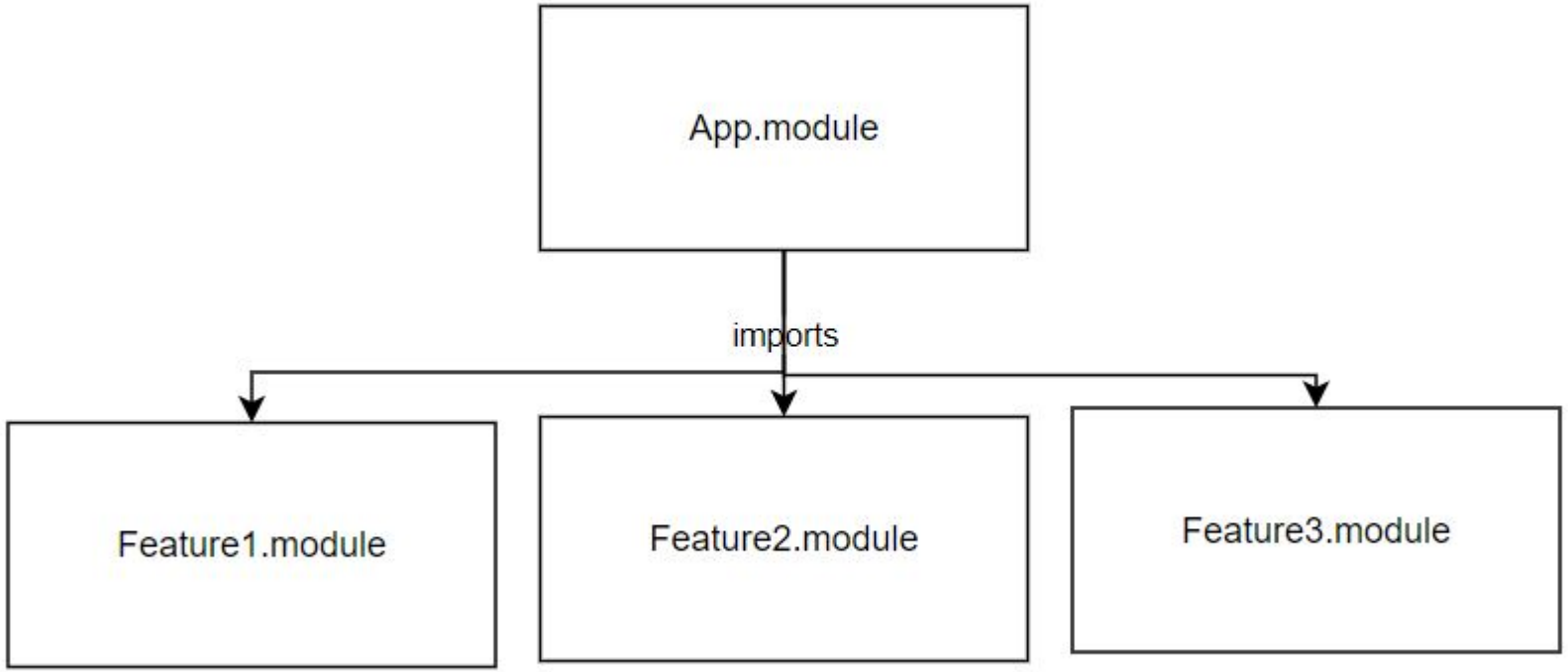
- Used for organize/test/manage large JS/TS projects
- Based on best practices used in Google to scale their own mono-repositories
- It can be used for small or large projects in various of frameworks (Angular, React, Nest.js, Express.js, Web Component Libraries, JS Libraries)
- Offers tooling
  - To share code between projects in your repository
  - Testing only what it needs based on Git changes
  - Migrations and generators

On short: Don’t throw project inside a folder and put them on Git and expect it to be a mono-repository

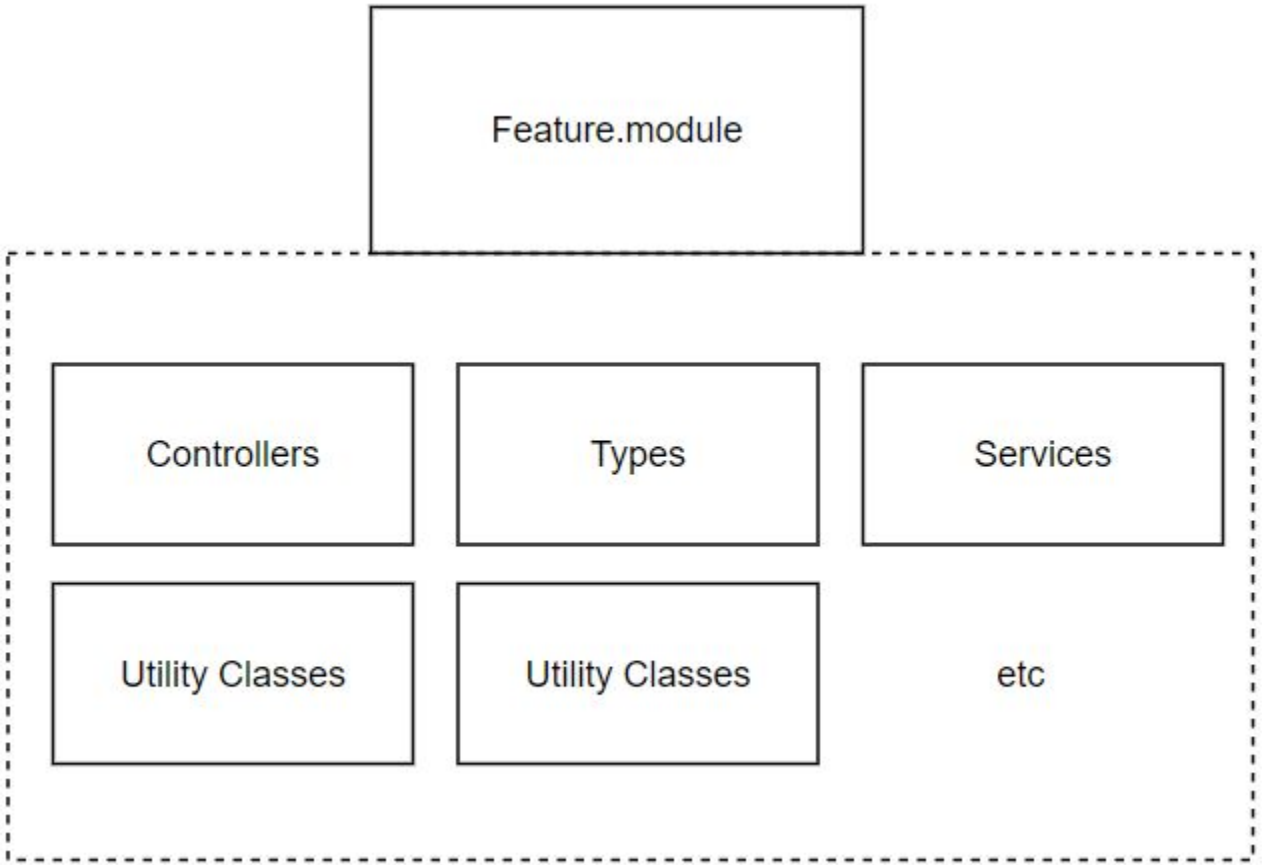
Nest.js

- A Node.js used to build scalable server-side application using TypeScript
- Is built on top of Express.js (a more minimal web application framework)
- Supports REST and GraphQL APIs out of the box
- Has built-in support for various of functionality
  - Security (Authentication + Authorization)
  - Rate-Limiting
  - Microservices
  - Websockets
  - Cors
  - Logging
  - etc.

Nest.js: Structure

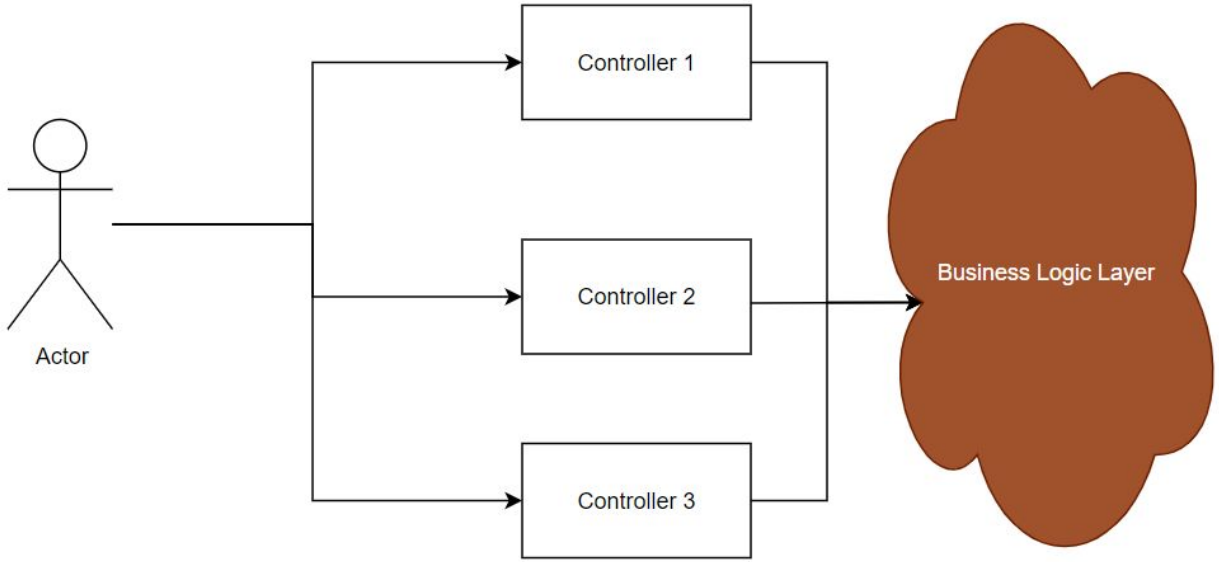


Nest.js: Structure



Nest.js: Controllers

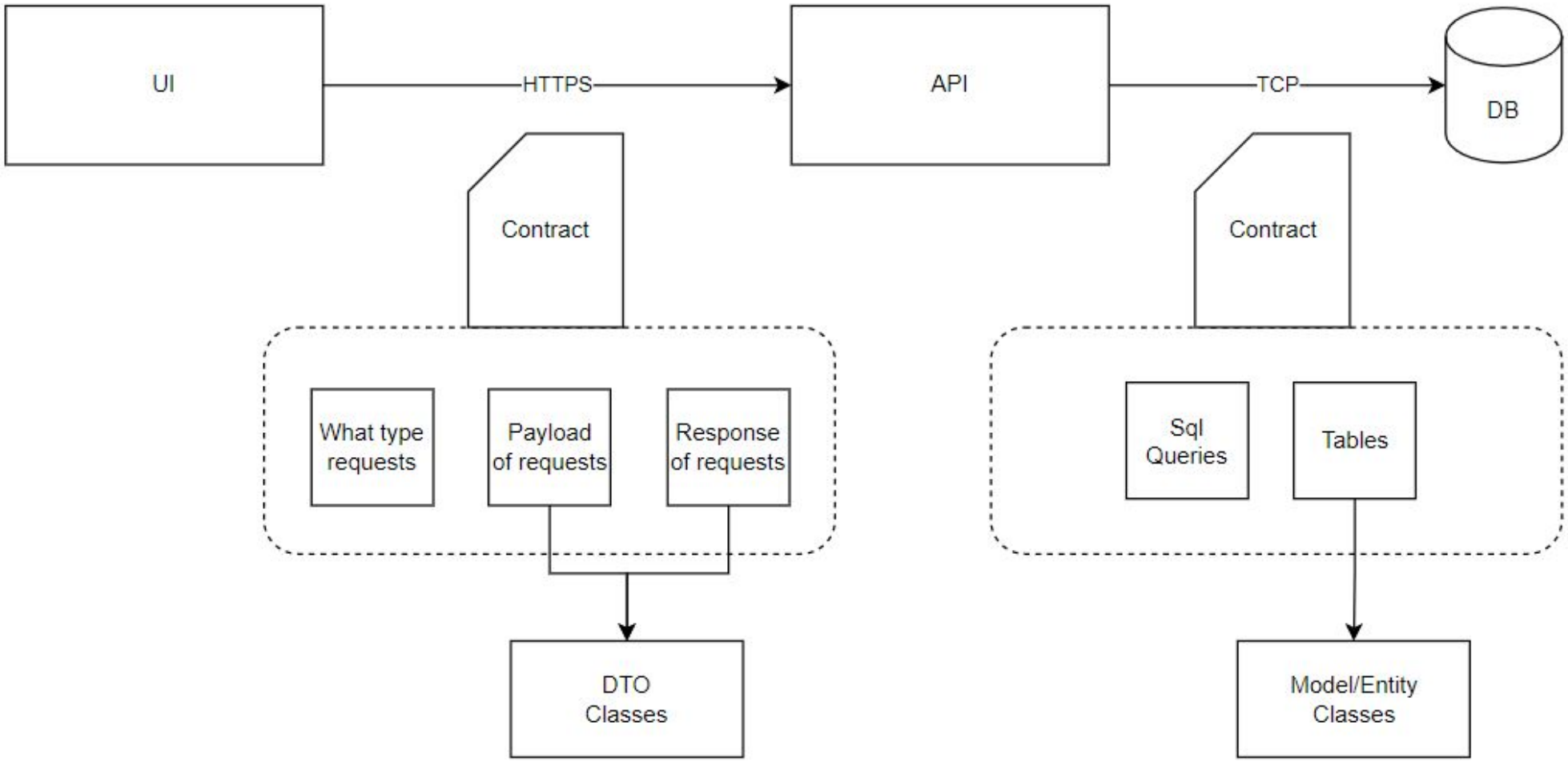
- The entry-point of the HTTP Requests in our application
- Receives and request, passes it to the business logic layer and returns the response (success, error)



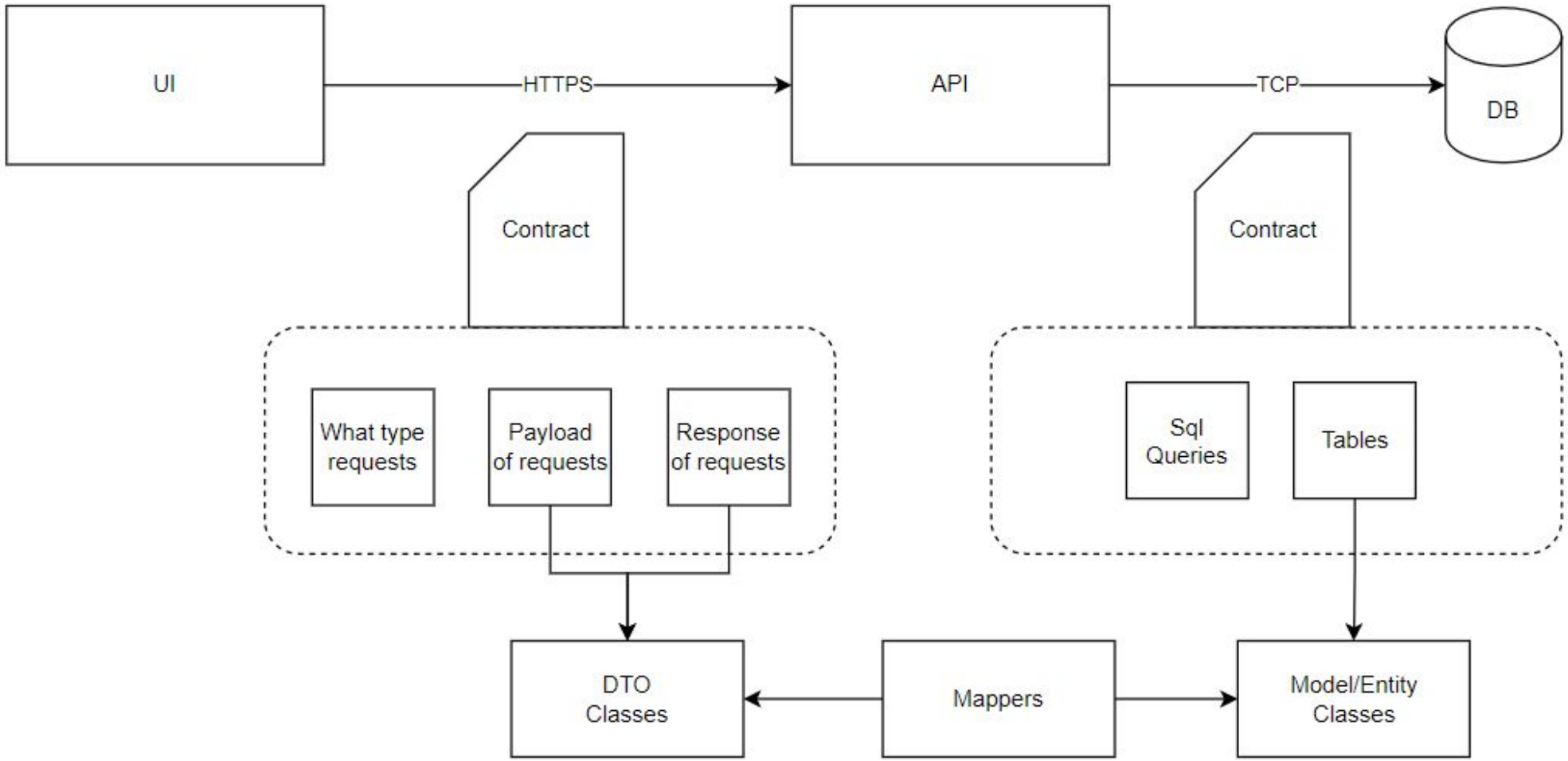
Nest.js: Controllers

- You need to decorate a class with @Controller
- Add methods inside and decorate them with @Get(), @Get('joke/:id'), @Post, @Put, @Patch, @Delete
- Reference the controller inside a module

DTO and Model Classes



Mappers


UI
HTTPS
API
TCP
DB
Contract
Contract
What type requests
Payload of requests
Response of requests
Sql Queries
Tables
DTO Classes
Mappers
Model/Entity Classes

The OpenAPI Initiative

- Offers us a way to document our “contract” for our API
- Also provides a visual interactive editor to test our API
- Integrated with Nest.js