T-302-HONN: Lab 6
Application Architecture
Notflix

Þórður Friðriksson
thordurf@ru.is

Háskólinn í Reykjavík — Reykjavík University

Introduction

Attention Please read over all the following points before you start the project

1. **Groups**
   - This assignment can be done in groups of 1-2 people
   - Only one member of each group needs to hand in the assignment
   - Remember to add each group members name and email to the front page of the handed-in assignment
2. **Rules about cheating**
   - The solution needs to be you own work, if it is discovered that cheating has occured then all participating parties will receive a 0 for this assignment if this is their first offence on the other hand if this is a repeating occurance then more severe measures can be expected. See the school's policy and rules on assignments ru.is/namid/reglur/reglur-um-verkefnavinnu
3. **Assignment hand-in**
   - the assignments need to be handed-in on Canvas both for the pdf file with all the written answers as well as a .zip file for the code
   - pdf file must be named: `{student1@ru.is}-{student2@ru.is}-lab6.pdf`
   - zip file must be named: `{student1@ru.is}-{student2@ru.is}-lab6.zip`
   - **If the instructions for the handin are not follwed you can expect a grade deduction**
4. **Late submissions**
   - See late submission policy

1 Notflix

In this project, you are given an implementation of the ever so popular streaming provider Notflix. This version of Notflix has been implemented using FastApi which is a python framework for creating backends.

This backend has the following endpoints:

- For **Movies**
  - `GET /movies` To get all movies
  - `POST /movies` To store movies
- For **Users**
  - `GET /users` To get all users
  - `POST /users` To store users
- For **Pricings**
  - `GET /pricings` To get all pricings
  - `POST /pricings` til store pricings
- For **Subscriptions**
  - `GET /subscriptions` to get all subscriptions
  - `POST /subscriptions` til store subscriptions

Along with FastApi the following things are also used:

- SQLITE as an in-memory database
- SQLAlchemy as an ORM to store and get data from the database
- alembic to handle database migrations
- Pydantic for configuration management
- Injector as a DI framework

Before you panic there is no need for you to understand the things mentioned above, the implementation for these things are given.

This implementation of Notflix is made using the **3-Tier** Architecture Pattern. It is your job to convert this project into **Onion** and **Vertical Slicing**

Notice.

- It is worth keeping in mind that this provided project is not a full implementation of a backend, there is much that is missing, such as error handling, logging, better and more endpoints, etc. This project is only intended for you to learn more about application architecture.

1.1 Onion Architecture (50 points)

Change the project to use `Onion Architecture` instead of 3-Tier.

The project structure should be as follows:

- **core**
- **database** for database functionality
- **external** for external systems (not the database)
- **infrastructure** as a shared infrastructure of the system
- **endpoints** for the api functionality (the endpoints)

Place this code under a folder named Onion in the zip file you hand-in

Notice.

- Note that infrastructure here is not the same as what is often referred to as infrastructure in the onion pattern. Infrastructure here is more of a maintenance of the system. External for us in this project is what is referred to as infrastructure in the onion pattern.
- You do not need to touch the following things-> `alembic.ini`, `readme.md`, `requirements.txt`, `sql_app.db`, `startup.py`

1.2 Vertical Slicing (50 points)

Change the project again but now you should change the project to use the `Vertical Slicing Architecture`.

Keep this code under a folder named VerticalSlicing in the zip file you hand-in

1.3 Bonus (10 points)

- One of the known problems with N-Tier architecture shines through very strongly in this project, what is that problem? Explain.