The idea

This project was an idea that both me and my internship contact at Altibox thought would be a fun challenge. The thought of making an interactive movie database with inspiration from existing ones like the most famous “IMDB” was something I looked forward to getting started with.

Requirements

The requirements from my internship contact were to use external APIs to get already existing movie data in to my relational mysql database. There would be a simple front end that would manage to let the user know what possibilities they had in navigating the application. Also, user specific actions were required, meaning authentication was required.

Delivery

Getting started with the project was a lot of fun. Seeing the database with its relations take form is always motivating knowing that the logic of the tables will soon “handle” itself.

Getting the data into the database on the other hand turned out to be quite the challenge. There were many movieAPIs to choose from, but very few had the option to return lists like “top rated”. So I found a raw json file on Github that was public that gave me some titles, I also found another list that had additional movies not ordered by rating. I then had to format the titles to get a stable “title” that I later could query an api with each individual title.

For making the project as scalable as possible I created what I named “apiScript” route, that would take the queried movies and separate each data value like “genres, year, rating, actors, directors etc. and insert them into my tables, later I had to query the movies again to insert both movie and actors/genres to make sure my tables was normalized since movies have several actors, and actors have several movies.

When that data was inserted, I could finally insert the whole movie with its respective keys.

I have met the requirements of a relational database, a functioning frontend, and user Specific action. There are CRUD operations that authenticated users can perform to customize their profiles and leave reviews of movies.

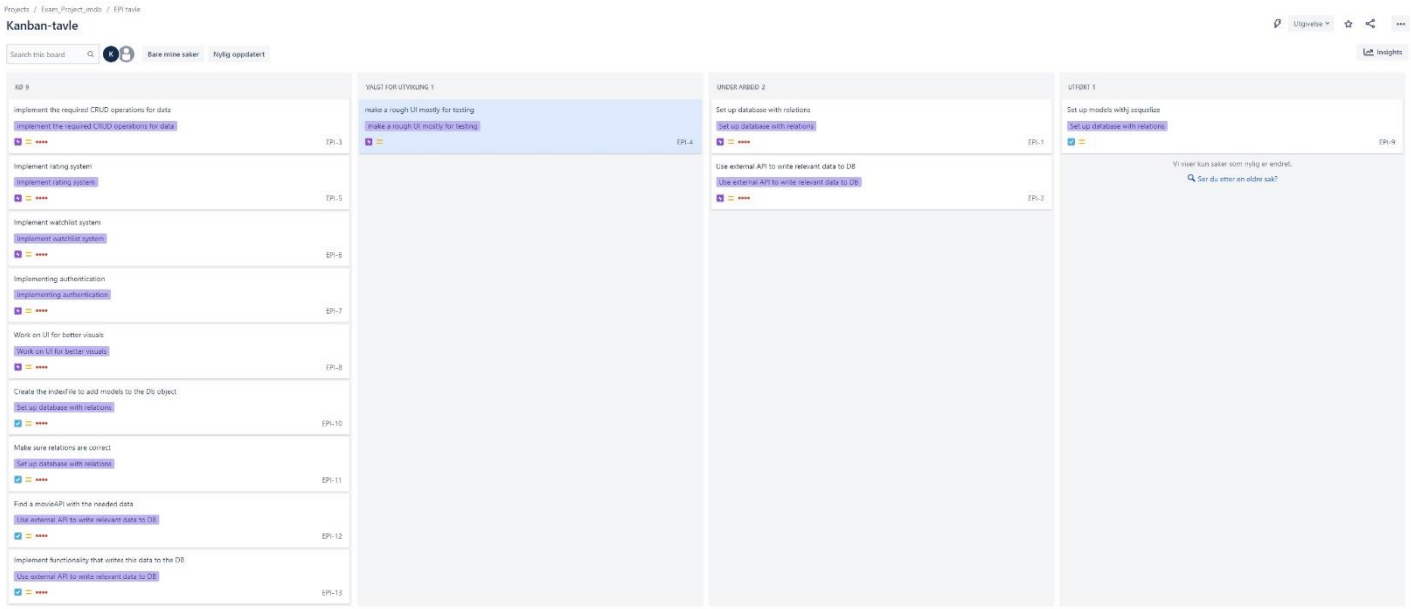
The front end has its flaws, but this project was to display back-end capabilities and not UI or design.

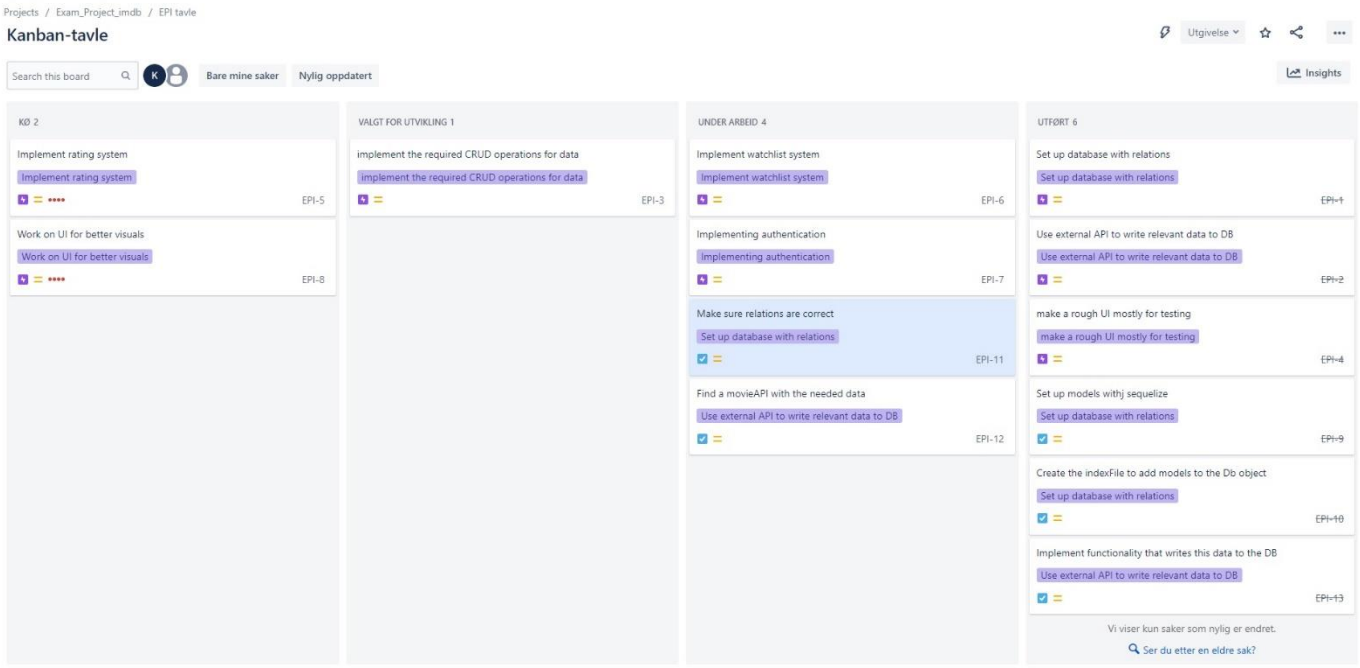
Challenges

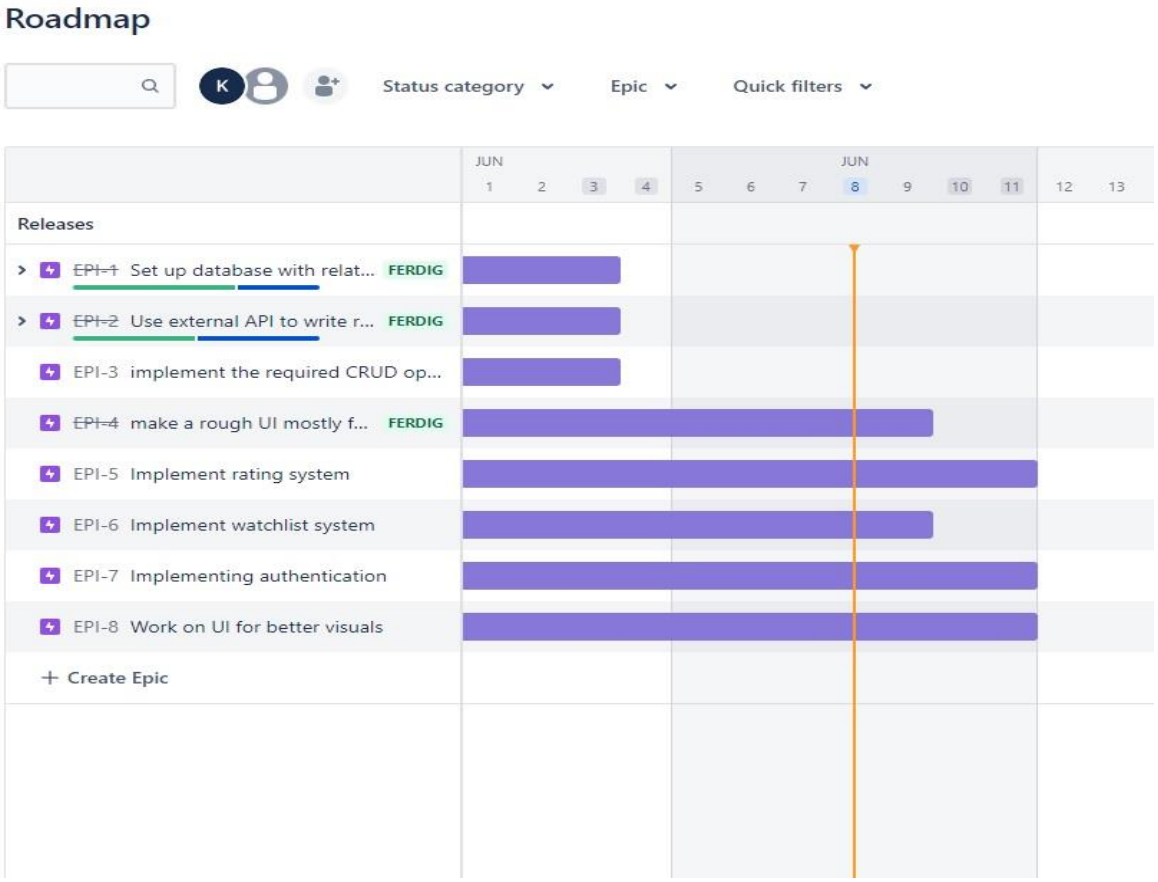
As I mentioned getting data formatted so my database is reliable and stable was a challenge, sometimes the api calls would contain “bad” data like missing values or a unregular typing. This caused me to spend a lot of time to pinpoint where I got errors with inserting and setting.

Dealing with a lot of asynchronous functionality and calls made debugging difficult at times, and to query several API’s each time the application starts up is a heavy load. But I wanted to keep it this way to showcase that I can handle APIS and process the data. Writing down the data to a json file would of course save a lot of time and memory, but then there would be no need for external API calls, and I wanted to include that process as well in the “finished” product.

I used Jira and made a kanban board so I had an overview of what tasks needed to be done.




Projects / Exam_Project_imdb / EPI tavle
Kanban-tavle
Utgivelse
Search this board
Bare mine saker
Nylig oppdatert
Insights
KØ 2
Implement rating system
Implement rating system
EPI-5
Work on UI for better visuals
Work on UI for better visuals
EPI-8
VALGT FOR UTVIKLING 1
implement the required CRUD operations for data
implement the required CRUD operations for data
EPI-3
UNDER ARBEID 4
Implement watchlist system
Implement watchlist system
EPI-6
Implementing authentication
Implementing authentication
EPI-7
Make sure relations are correct
Set up database with relations
EPI-11
Find a movieAPI with the needed data
Use external API to write relevant data to DB
EPI-12
UTFØRT 6
Set up database with relations
Set up database with relations
EPI-1
Use external API to write relevant data to DB
Use external API to write relevant data to DB
EPI-2
make a rough UI mostly for testing
make a rough UI mostly for testing
EPI-4
Set up models withj sequelize
Set up database with relations
EPI-9
Create the indexFile to add models to the Db object
Set up database with relations
EPI-10
Implement functionality that writes this data to the DB
Use external API to write relevant data to DB
EPI-13
Vi viser kun saker som nylig er endret.
Ser du etter en eldre sak?


Roadmap
Status category
Epic
Quick filters
JUN
1 2 3 4
JUN
5 6 7 8 9 10 11 12 13
Releases
EPI-1 Set up database with relat... FERDIG
EPI-2 Use external API to write r... FERDIG
EPI-3 implement the required CRUD op...
EPI-4 make a rough UI mostly f... FERDIG
EPI-5 Implement rating system
EPI-6 Implement watchlist system
EPI-7 Implementing authentication
EPI-8 Work on UI for better visuals
Create Epic