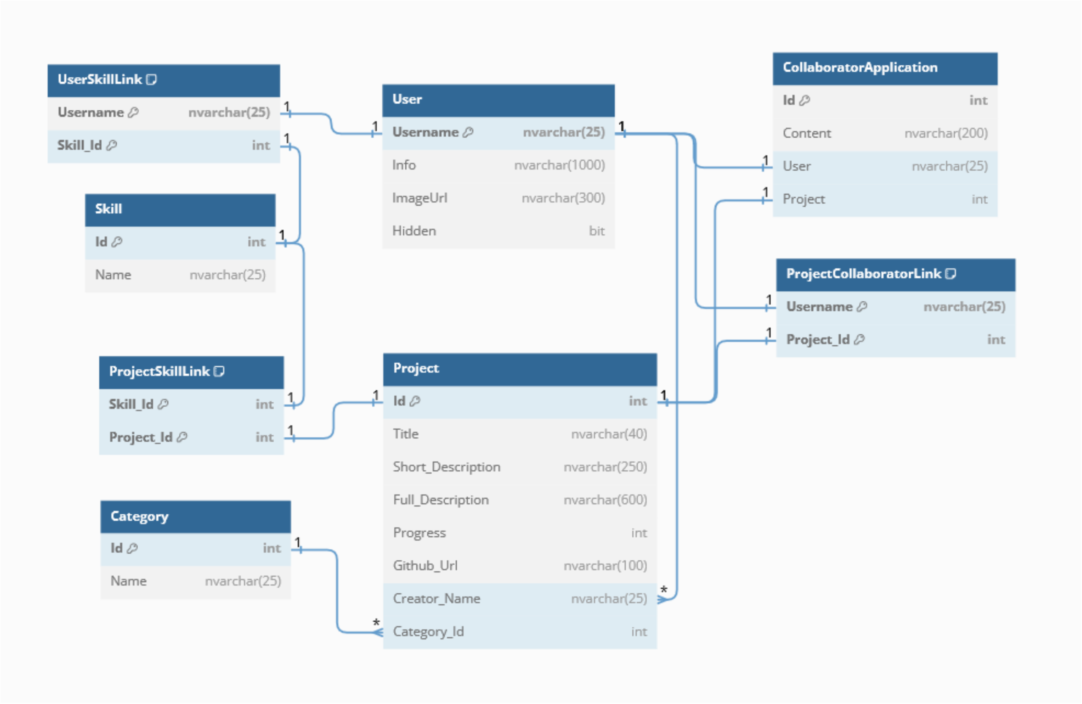
Silje Slettebakk, Silje Denise Risnes,
Magnus Uttisrud & Joakim Hansen

Technical Documentation

Database-diagram



The database has four primary entities: User, Project, Skill, and Category. The Project-table has a creator-name and ID, which identifies the project's creator. This results in a one-to-one relationship between Project and User. The Project entity also has a many-to-many relationship between Project and User, facilitated through a linking table named ProjectCollaboratorLink.

In order to enable users to submit applications to join a project, we have implemented a table named CollaboratorApplication. This table includes a content field, as well as fields for the project's ID and the username who has applied to join the project.

Link to swagger documentation for WEB-API: https://lagalt-docker.azurewebsites.net/swagger/index.html

Technologies

- .NET with Entity Framework
- SQL Server
- React with Styled Components
- TS, C#
- Keycloak
- Azure
- Docker

External services

Keycloak – We used keycloak, throuh cloud-iam to handle login and authentication. It is connected to front-end, where the keycloak token is used to determine if a user is logged in, and when necessary which user is logged in. Roles for different users is handled in react, as no user has global admin rights, but different rights for different projects based on data in database. We did implement keycloak token protection for api endpoints but removed them as we didn't have time to implement token into the api calls from frontend.

Setting Up and Running the Web API Locally

These instructions will guide you through the process of setting up and running the web API locally on your computer. The web API connects to a database, and in this guide, we assume you're using Visual Studio and Microsoft SQL Server Management Studio. To run the application locally, you'll need to set up your own database using provided SQL scripts.

Prerequisites

Before you begin, ensure that you have the following prerequisites installed:

Visual Studio

Microsoft SQL Server

SQL Server Management Studio (SSMS)

Installation Steps

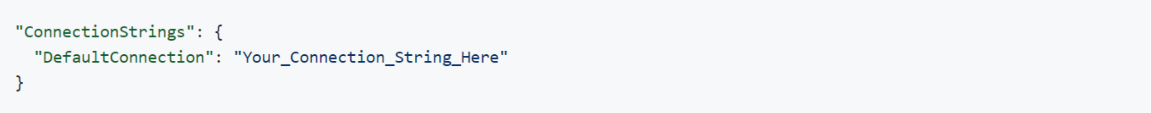
1. **Clone the Web API Project from GitHub:**
    - Open your terminal or command prompt.
    - Navigate to the directory where you want to store the project.
    - Run the following command to clone the repository from GitHub:

      ```
      git clone github-url
      ```

Replace github-url with the actual repository name.

2. **Initialize Your Database:**
    - In the project's root folder, you will find an 00_LagaltDB_init.sql file. Use SQL Server Management Studio or any SQL client to execute this script in your local SQL Server to create the necessary database structure.
    - Additionally, you can insert test data into your database using the 01_InsertTestData.sql script provided in the same folder.
3. **Set Up the Connection String:**
    - Open the appsettings.json file located in the project's root folder.

- Locate the "ConnectionStrings" section, and replace the DefaultConnection connection string with your own connection string.



```
"ConnectionStrings": {
  "DefaultConnection": "Your_Connection_String_Here"
}
```

4. **Build and Run the Web API:**
   - Open the solution in Visual Studio.
   - Build the solution to ensure all dependencies are resolved.
   - Set the Web API project as the startup project.
   - Press F5 or click "Start" to run the application. It should now be running locally on your computer.
5. **Access the Web API:**
   - You can access the web API at http://localhost:port (by default, the port is usually 7085)

Deployment to Azure

1. Create a docker image with the command:

docker build –t <your-dockerhub-username>/lagaltapi .

2. Push the image to dockerhub:

docker push <your-dockerhub-username>/lagaltapi

3. Create a web app in azure, hosting a docker image from

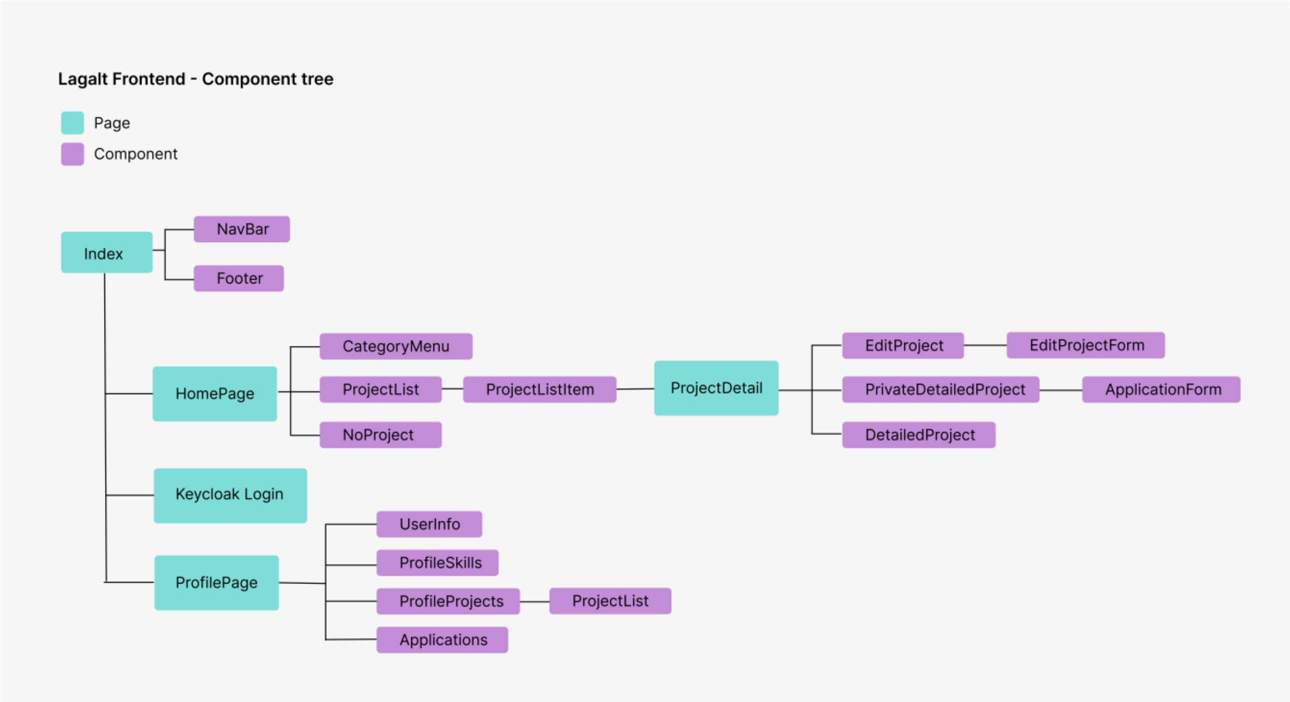
The WEB-API is hosted on Azure: Swagger-Documentation

Frontend

More information about the how the frontend solution turned out and how to use it, can be found in the user documentation file.

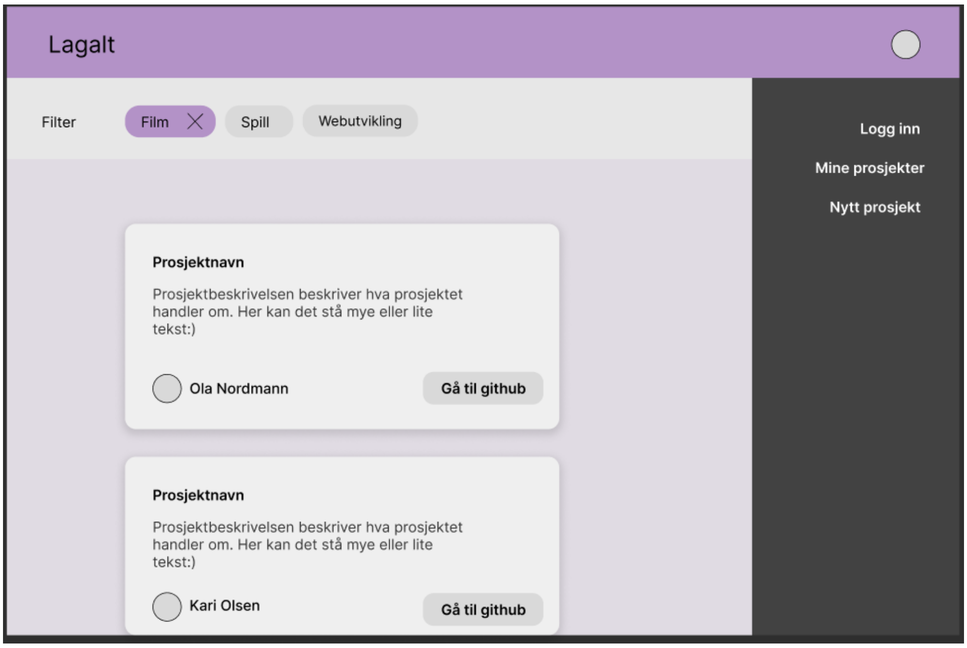
Component Tree

The component tree represents the pages and components of the frontend application. This is the architecture we ended up implementing:



Wireframes

As a part of the project planning, we started by making some wireframes in Figma. We ended up changing the design, but the wireframes helped us in scoping down the assignment. By visualizing our ideas, it was easier to decide which solutions to go for.

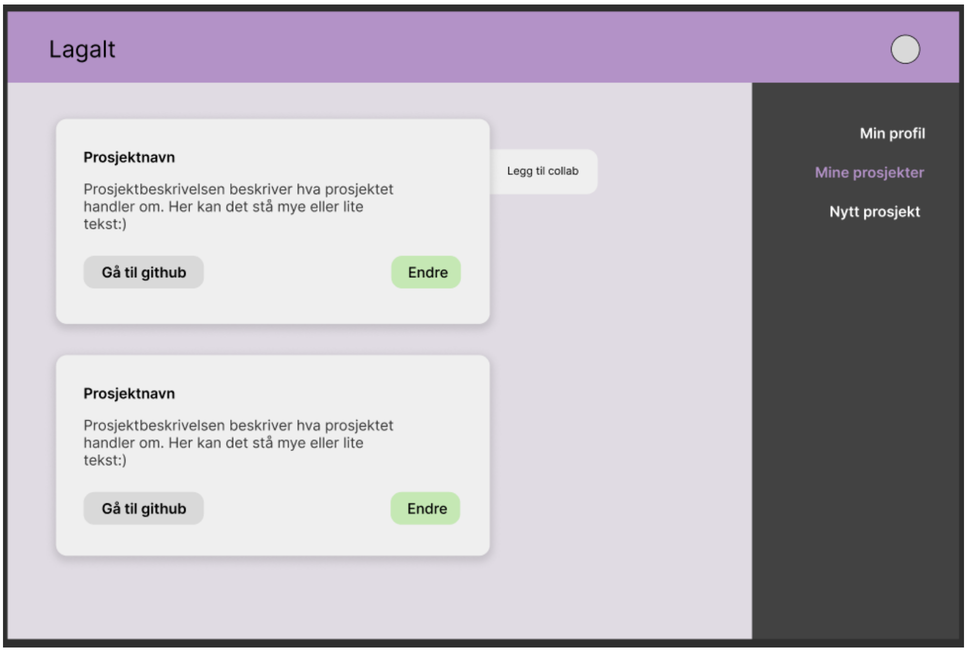


Prosjektnavn

Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)

Prosjektnavn

Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)

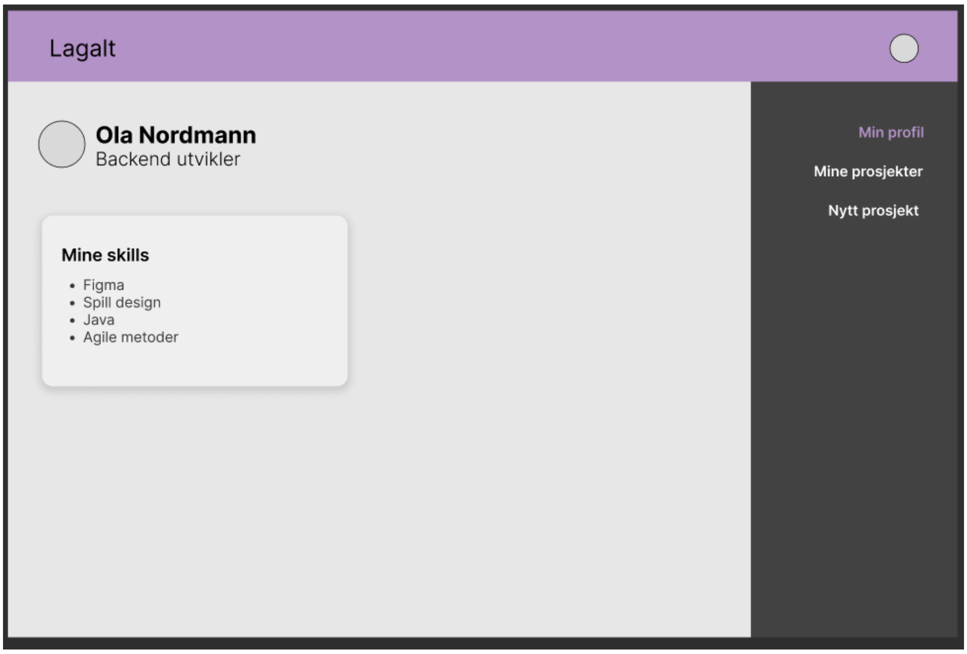


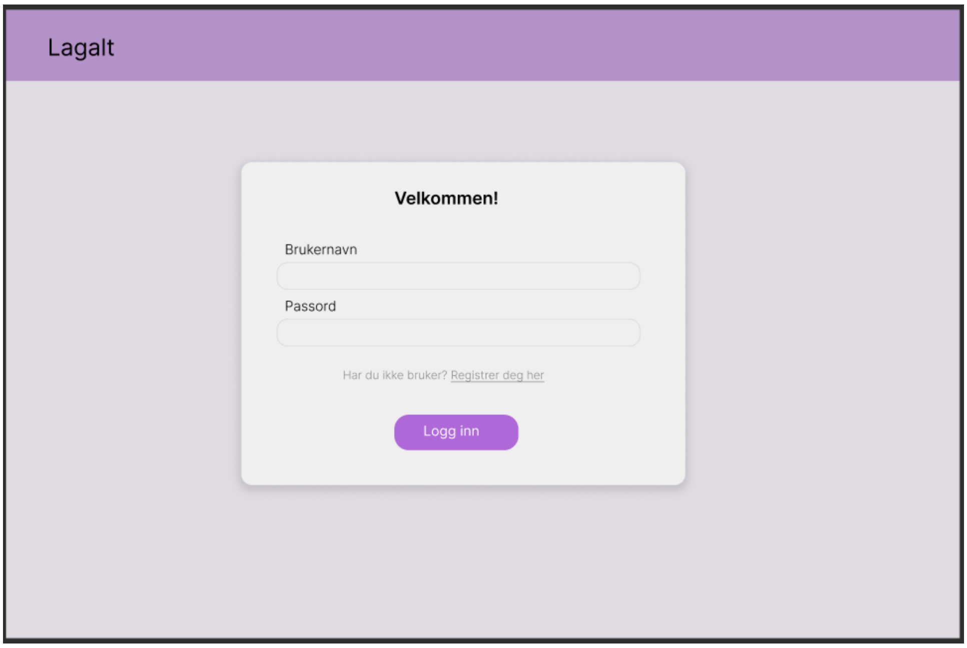
Prosjektnavn

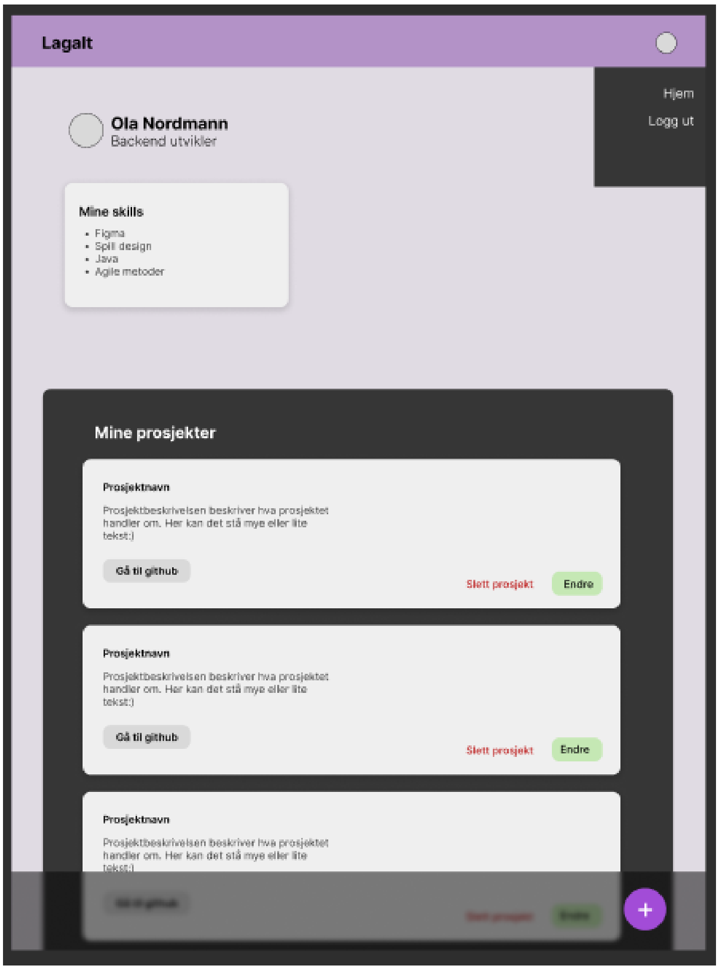
Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)

Prosjektnavn

Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)


Lagalt
Ola Nordmann
Backend utvikler
Mine skills
Figma
Spill design
Java
Agile metoder
Min profil
Mine prosjekter
Nytt prosjekt

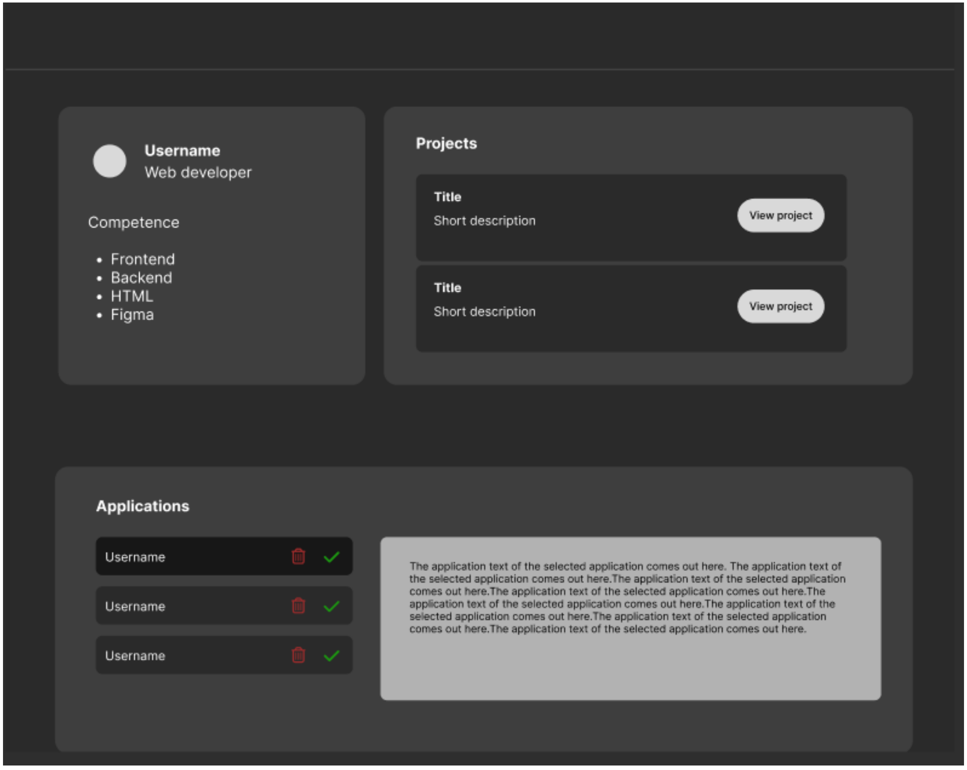

Lagalt
Velkommen!
Brukernavn
Passord
Har du ikke bruker? Registrer deg her
Logg inn



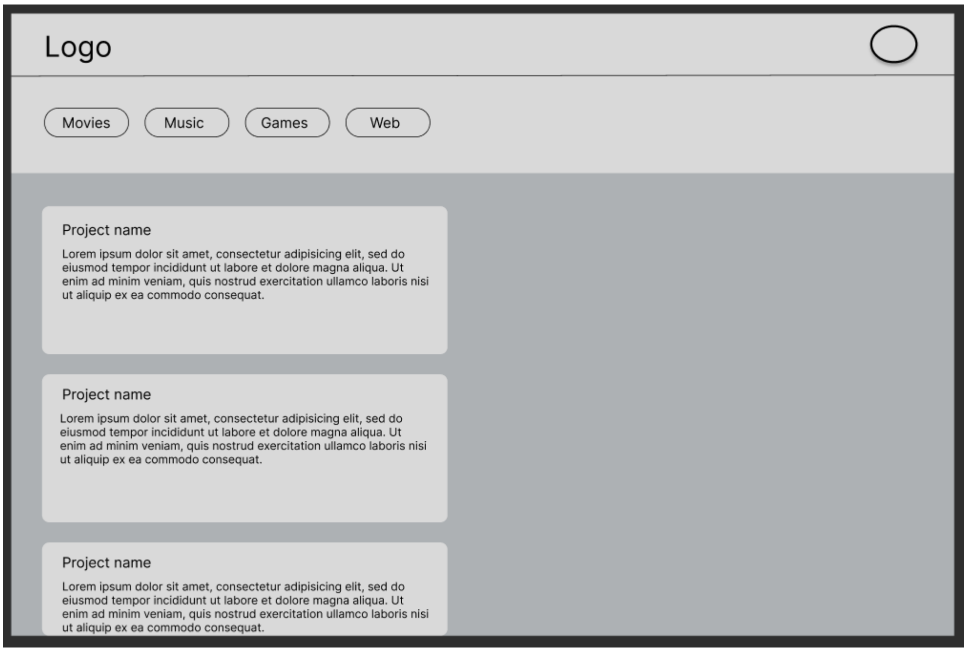
Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)

Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)

Prosjektbeskrivelsen beskriver hva prosjektet handler om. Her kan det stå mye eller lite tekst:)



The application text of the selected application comes out here. The application text of the selected application comes out here.The application text of the selected application comes out here.The application text of the selected application comes out here.The application text of the selected application comes out here.The application text of the selected application comes out here.The application text of the selected application comes out here.



Project name

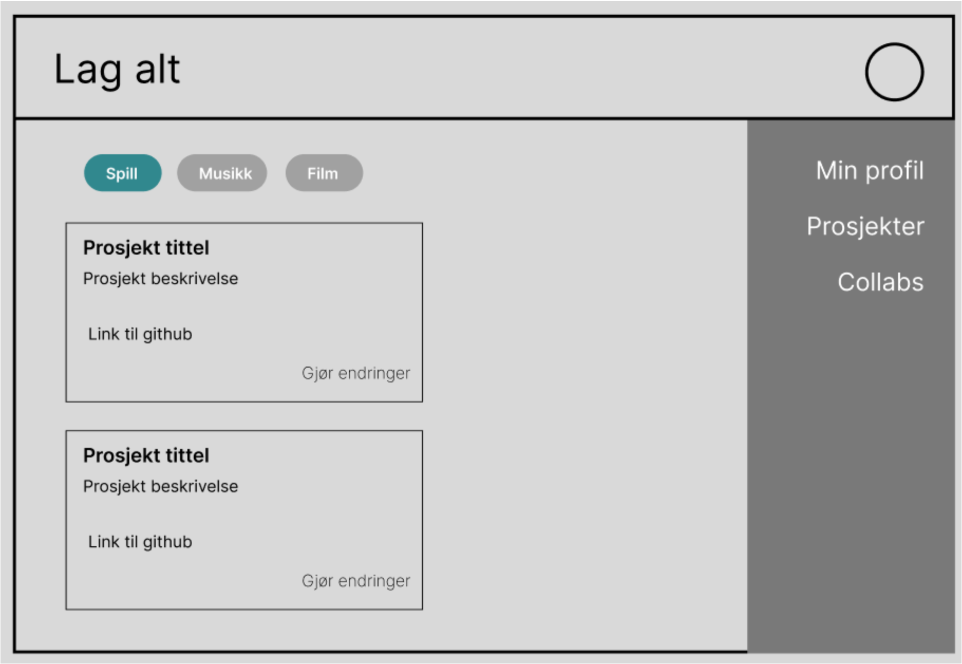
Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Project name

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Project name

Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.



Prosjekt tittel

Prosjekt beskrivelse

Link til github

Gjør endringer

Prosjekt tittel

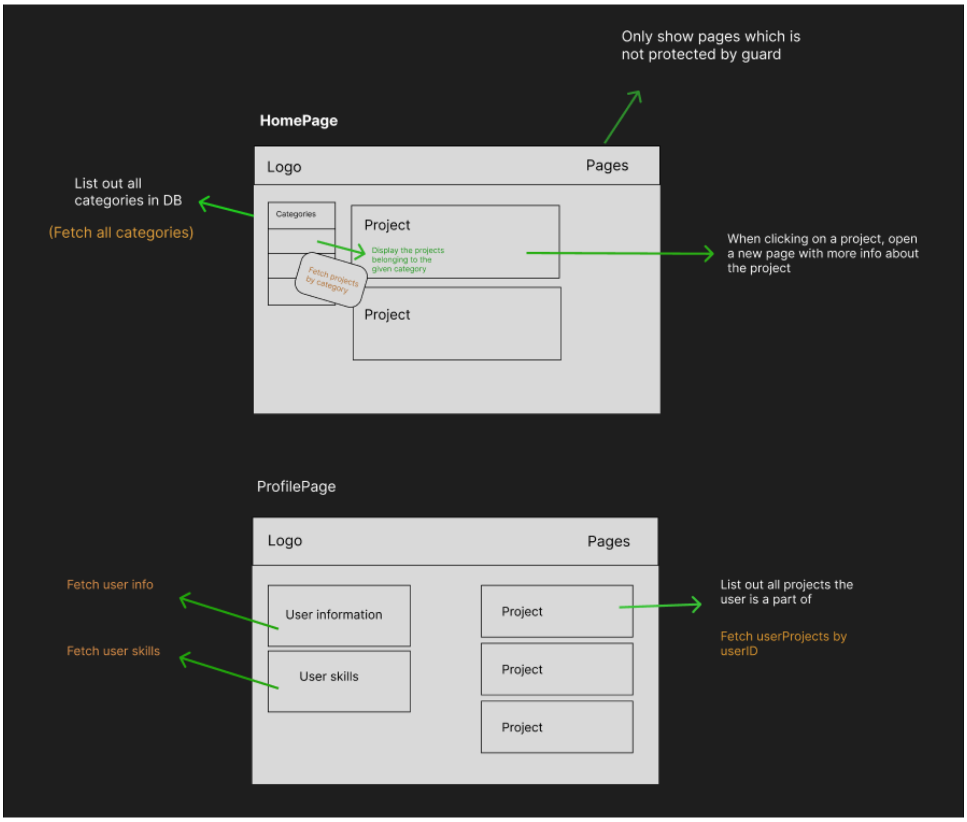
Prosjekt beskrivelse

Link til github

Gjør endringer

Data flow

We also visualized the data flow through the application, before we started creating the frontend solution. For this, we also used Figma.


Only show pages which is not protected by guard
HomePage
Logo
Pages
List out all categories in DB
(Fetch all categories)
Categories
Project
Display the projects belonging to the given category
Fetch projects by category
Project
When clicking on a project, open a new page with more info about the project
ProfilePage
Logo
Pages
Fetch user info
User information
Fetch user skills
User skills
Project
Project
Project
List out all projects the user is a part of
Fetch userProjects by userID