## Development frameworks:

* Backend: **.NET Core C#**.
* Frontend: **React** with **Material UI** as design library, and **React Router**

## Backend Architecture:

The system has 2 main parts. The communication with the front End, and the communication with TVmaze API. I used 2 layers system, **controller** and **manager** the the controller makes the interactions with Front end as our system's API.  
manager makes the interactions with TVmaze's API.

Currently, there is no need for business logic. If we want to add it, we can do that in 2 ways.

1. Make it part of the manager.
2. Separate the manager to business logic layer, and data management layer.

In addition, I created a models namespace, that has 2 types of models:

1. Models used to handle the TVmaze API, with minimal logic, and cleaner code.
2. Models by the system for different things, for example, sending data from controller properly.

API interactions details can be found on [[Appendix 1]](#_Appendix_1)

## Frontend Architecture:

The system is component driven, as excepted from a React app, using TypeScript. I chosen typescript over JavaScript because its brighter, and easier to understand the code. The modules are separated into 4 directories, according to their goals.

1. **entities** – the interface that represents the data and used over the application (except props' interfaces that attached to there components.
2. **Pages** – components that represent pages in the app.
3. **Components** – components that used **in** pages or other parts of the app.
4. **Managers** – modules to handle logic that used across the application to specific area, for example, favorites' logic has manager, because it needed on both search and favorites pages.

Search page UML Sequence diagram can found on [[Appendix 2]](#_Appendix_2)

## Appendix 1

### Endpoints

#### GetShowsShort

Returns a list of shows based on the provided name.

**HTTP Method**: GET

**Route**: /api/GetShowsShort

**Parameters:**

- name (string, required): The name of the show to search for.

**Response:** A list of objects, that represent basic info about a show.

#### GetShow

Returns details of a specific show based on its ID.

**HTTP Method**: GET

**Route**: /api/GetShow

**Parameters**:

- id (int, required): The ID of the show.

- includeEpisodes (bool, optional): Whether to include episodes data. Default is false.

- includeCast (bool, optional): Whether to include cast data. Default is false.

- includeCrew (bool, optional): Whether to include crew data. Default is false.

**Response**: A Show object if found, include embedded as asked, otherwise null.

#### BulkSGetShows

Returns a list of shows for the provided list of IDs.

**HTTP Method**: GET

**Route**: /api/BulkSGetShows

**Parameters**:

- showsIds (List<int>, required): A list of show IDs.

- includeEpisodes (bool, optional): Whether to include episodes data. Default is false.

- includeCast (bool, optional): Whether to include cast data. Default is false.

- includeCrew (bool, optional): Whether to include crew data. Default is false.

**Response**: A list of ShowShortViewModel objects if found, otherwise null.

#### ShowIndex

Returns a list of shows from the TVmaze index, using the provided limit and offset.

**HTTP Method:** GET

**Route:** /api/Shows

**Parameters:**

* **limit** (int, optional): The maximum number of shows to retrieve. Default is 250, which is amount of page in TVmaze api.
* **offset** (int, optional): The number of shows to skip before starting to retrieve results. Default is 0.

**Response:**

A list of objects, that represent basic info about a show. Returns null if no shows are found.

## Appendix 2

