**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Кафедра информационных систем управления

ЦАРИК ВИТАЛИЙ АЛЕКСАНДРОВИЧ

**Разработка web-приложения "Сервис подписок"**

Курсовая работа

студента 3 курса 2 группы

**Научный руководитель:**

*Блажукас Томас Реналдович*

Минск, 2020

**АННОТАЦИЯ**

Царик В.А. Разработка web-приложения "Сервис подписок": Курсовая работа / Минск: БГУ, 2020. – 25 с.

В курсовой работе рассмотрен вопрос разработки web-приложений на примере сервиса подписок. В качестве бэкенда был разработан web-сервер с Rest API с помощью Django, со стороны фронтенда было разработано SPA-приложение с использованием React.js.

**АНАТАЦЫЯ**

Царык В.А. Распрацоўка web-прыкладання "Сэрвіс падпісак": Курсавая работа / Мінск: БДУ, 2020. – 25 с.

У курсавой працы разгледжаны пытанне распрацоўкі web-прыкладанняў на прыкладзе сэрвісу падпісак. У якасці бэкэнда быў распрацаваны web-сервер з Rest API з дапамогай Django, з боку фронтенда было распрацавана SPA-прыкладанне з выкарыставаннем React.js.

**ANNOTATION**

Tsarik V.A. Development of the web application "Subscription Service": Coursework / Minsk: BSU, 2020 – 25 p.

In the term paper the issue of developing web applications was reviewed on the example of asubscription service. As a backend, a web-server with Rest API was developed using Django, for the frontend, a SPA application was developed using React.js.

**ОГЛАВЛЕНИЕ**

[ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ 4](#_Toc40382772)

[ВВЕДЕНИЕ 5](#_Toc40382773)

[ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ 6](#_Toc40382774)

[1.1 Цели 6](#_Toc40382775)

[1.2 Функциональные требования к приложению 6](#_Toc40382776)

[1.3 Анализ существующих аналогов 6](#_Toc40382777)

[1.3.1 IMDb 6](#_Toc40382778)

[1.3.2 Netflix 7](#_Toc40382779)

[1.3.3 КиниПоиск 7](#_Toc40382780)

[1.3.4 Metacritic 8](#_Toc40382781)

[1.4 Выводы 8](#_Toc40382782)

[ГЛАВА 2. РЕАЛИЗАЦИЯ 9](#_Toc40382783)

[2.1 Архитектура и стек технологий 9](#_Toc40382784)

[2.1.1 Фронтенд 10](#_Toc40382785)

[2.1.2 Бэкенд 11](#_Toc40382786)

[2.1.3 База данных 12](#_Toc40382787)

[2.1.4 Tasks queue 15](#_Toc40382788)

[2.2 Аутентификация 16](#_Toc40382789)

[2.3 Выводы 17](#_Toc40382790)

[ГЛАВА 3. ОБЗОР РЕАЛИЗОВАННОЙ ФУНКЦИОНАЛЬНОСТИ ПРИЛОЖЕНИЯ 18](#_Toc40382791)

[3.1 Домашняя страница 18](#_Toc40382792)

[3.2 Панель навигации 19](#_Toc40382793)

[3.3 Авторизация 20](#_Toc40382794)

[3.4 Список подписок 21](#_Toc40382795)

[3.5 Страница поиска 21](#_Toc40382796)

[3.6 Страница с детальной информацией 23](#_Toc40382797)

[3.7 Выводы 23](#_Toc40382798)

[ЗАКЛЮЧЕНИЕ 24](#_Toc40382799)

[СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ 25](#_Toc40382800)

# ПЕРЕЧЕНЬ СОКРАЩЕНИЙ И УСЛОВНЫХ ОБОЗНАЧЕНИЙ

API – Application Programming Interface

Rest – Representational state transfer

SPA – Single Page Application

HTML – Hypertext Markup Language

MVC – Model-View-Controller

NPM – Node Package Manager

HTTP – HyperText Transfer Protocol

AJAX – Asynchronous JavaScript and XML

DOM – Document Object Model

CSRF – Сross Site Request Forgery – межсайтовая подделка запроса

TTE – Time to Interactive

СУБД – Система управления базами данных

SQL – Structured Query Language

URL – Uniform Resource Locator

XML – Extensible Markup Language

JSON – JavaScript Object Notation

Config – Configuration

ОС – Операционная система

ORM – Object-relational mapping

CRUD – Create, read, update and delete

DRY – Don't repeat yourself

ООП – Объектно-ориентированное программирование

DRF – Django Rest Framework

# ВВЕДЕНИЕ

Использование интернета значительно возрасло в последние несколько лет, особенно с серьёзно возрастающей популярностью мобильных устройств таких, как смартфоны и планшеты. Практически в каждом доме на земле, так или иначе, есть доступ в интернет.

Каждое подключённое к глобальной сети устройство можно отследить. Большество современных мобильных устройств имеет встроенный GPS модуль, который даёт возможность узнать текущее местоположение с высокой точностью. Данные, полученные с помощью этого GPS модуля, позволяют разработчикам web-приложений, используя местоположение пользователей, создать такие инновационные продукты, как например Google Maps и Яндекс.Карты.

Сложно даже представить, что нас ждёт через десять, двадцать, тридцать лет.

Поэтому, в связи с бурным развитием технологий в общем и интернета, в частности, человеку в современном мире приходится быть в курсе огромного количества вещей. Таких, как новый сезон любимого сериала, свежая часть любимой серии книг, последний выпуск любимого подкаста или новый музыкальный альбом, который на вершине всех рейтингов.

В рамках курсовой работы было спроектировано и разработано web-приложение с возможностью поиска и подписки на сериалы и фильмы. Смысл подписки в том, что при выходе нового сезона сериала всем пользователям, подписанным на этот сериал, будет выслано уведомление на почту.

# ГЛАВА 1. ПОСТАНОВКА ЗАДАЧИ

## Цели

Ниже представлены основные цели, которые я хотел достичь при выполнении курсовой работы:

* Разработать полноценное web-приложение с раздельными бэкенд и фронтенд частями и с функциональностью подписки на сериалы и фильмы;
* Познакомится подробнее с основными стадиями проектирования и разработки web-приложений;
* Продумать архитектуру, чтобы она была довольно гибкая и неплохо масштабируема. Например, чтобы можно было легко добавить к сериалам и фильмам подписки на книги, подкасты или музыкальные альбомы.

## 1**.2 Функциональные требования к приложению**

Требуется разработать web-приложение, в котором пользователи могут совершать следующие действия:

* Зарегистрироваться;
* Авторизоваться с помощью логина и пароля;
* Выполнить поиск по названию;
* Просмотреть базовую информацию о конкретном сериале или конкретном фильме;
* Подписаться и отписаться на сериалы и фильмы;
* Просмотреть свои текущие подписки.

## 1.3 Анализ существующих аналогов

## 1.3.1 IMDb

IMDb – крупнейшая в мире база данных и веб-сайт о кинематографе. В базе собрана информация о более чем 6,5 млн кинофильмов, телесериалов и отдельных их серий, а также о 10,4 млн персоналий, связанных с кино, – актёрах, режиссёрах, сценаристах и др.

С 1998 года базой данных IMDb владеет компания Amazon.com. Почти вся информация IMDb находится в свободном доступе, сайт IMDb.com функционирует на основе свободного программного обеспечения (Apache, Perl и др.).

Одной из главных особенностей IMDb является то, что редактировать информацию в базе данных может любой пользователь Интернета, прошедший регистрацию и модерацию на сайте.

## 1.3.2 Netflix

Netflix – американская развлекательная компания, поставщик фильмов и сериалов на основе потокового мультимедиа. Основана 29 августа 1997 года Ридом Хастингсом и Марком Рэндольфом. Штаб-квартира находится в Лос-Гатосе, Калифорния.

С 2013 года Netflix производит собственные фильмы и сериалы, в том числе и анимационные, а также телепрограммы. В 2016 году компания выпустила 126 оригинальных сериалов и фильмов – больше, чем любой другой сетевой или кабельный канал. Их усилия по производству нового контента, обеспечению прав на дополнительный контент и диверсификации через 190 стран потребовали дополнительных капиталов, привлекаемых в том числе через займы. Суммарный долг в 2016 году составил 16,8 млрд долларов, по состоянию на сентябрь 2017 года он был уже 21,9 млрд долларов. При этом 6,5 миллиарда долларов – это долгосрочный долг, а остальная часть – долгосрочные обязательства. В октябре 2018 года Netflix объявила о привлечении ещё 2 миллиардов долларов для финансирования нового контента. По состоянию на январь 2019 года у компании насчитывалось 139 млн подписчиков по всему миру, в том числе 58,49 млн в США.

## 1.3.3 КиноПоиск

КиноПоиск – крупнейший русскоязычный интернет-сервис о кино. С 2018 года также доступен онлайн-кинотеатр с несколькими тысячами фильмов, сериалов, мультфильмов, в том числе премьерных и эксклюзивных. С 2020 года Кинопоиск HD начал производить эксклюзивный контент специально для своего сервиса, начав список с сериалов "Последний министр" и "Проект «Анна Николаевна»".

Владельцем проекта до октября 2013 года являлась компания ООО «КиноПоиск», которой принадлежало 60 % акций проекта, 40 % акций принадлежало её совладельцу – французской компании ООО AlloCiné. 15 октября 2013 года компания «Яндекс» приобрела «КиноПоиск».

## 1.3.4 Metacritic

Metacritic – англоязычный сайт-агрегатор, собирающий отзывы о музыкальных альбомах, играх, фильмах, телевизионных шоу и DVD-дисках.

Для каждого продукта вычисляются две средних численных оценки: стобалльная – на основе обзоров профессиональных изданий и десятибалльная – на основе пользовательских обзоров. Отрывок из каждого обзора предоставляется вместе с гиперссылкой на источник. Это даёт представление об общей привлекательности продукта среди рецензентов и, в меньшей степени, общественности. Порой эти две оценки различаются вплоть до полной противоположности.

## Выводы

В главе 1 был определены основные цели выполнения курсовой работы, выставлены функциональные требования к web-приложению, которое будет разработано. А также проанализирована часть существующих в предметной области аналогов.

# ГЛАВА 2. РЕАЛИЗАЦИЯ

## 2.1 Архитектура и стек технологий

На Рисунке 2.1 представлена поверхностная иллюстрация архитектуры разработанного web-приложения. В следующих далее подразделах будет более подробно разобран каждый составляющий элемент диаграммы.

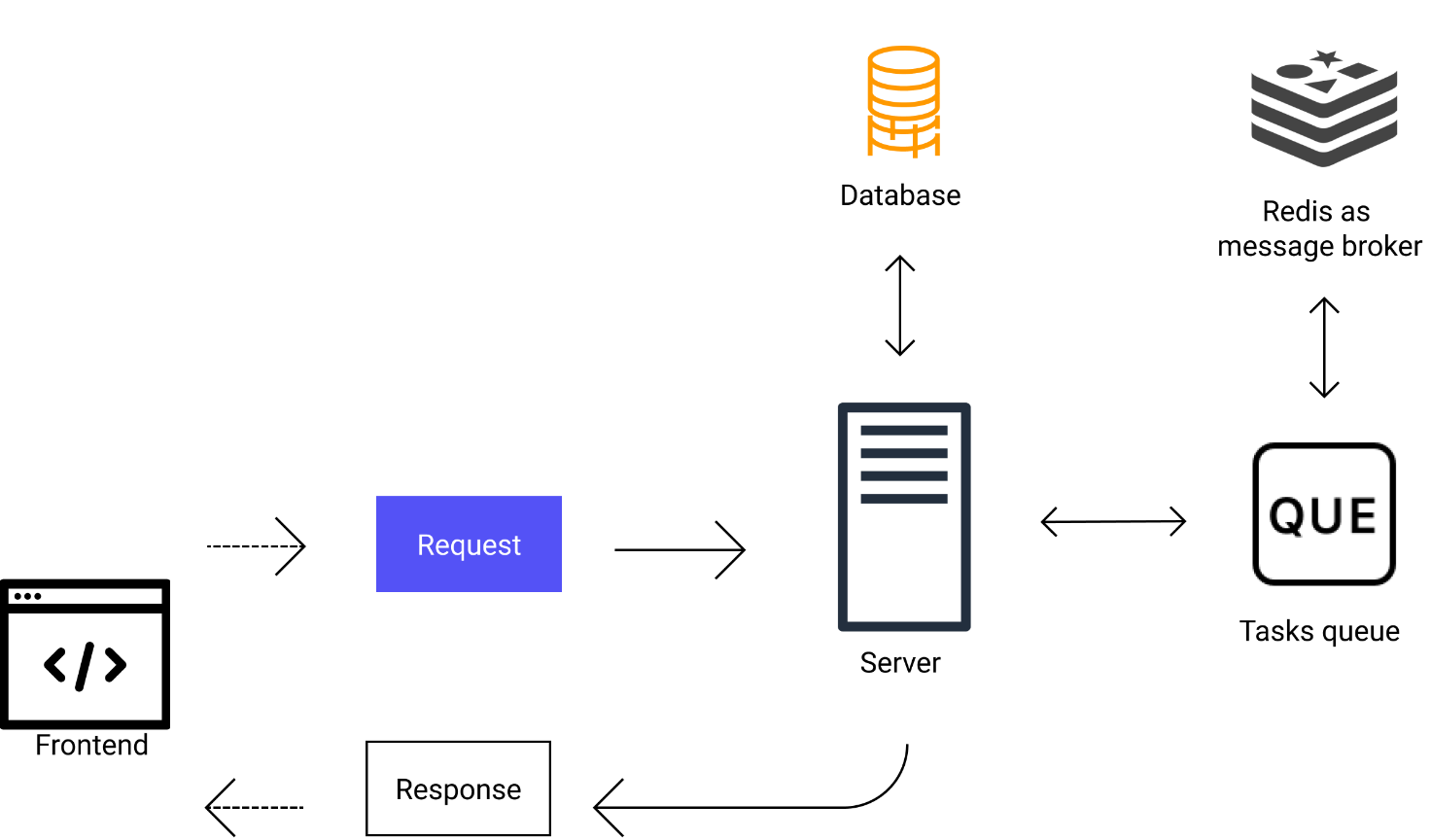
Если говорить о структуре приложение в целом, то она из себя представляет стандартную для web-приложений схему. Пользователи взаимодействуют с графическим интерфейсом на сайте используя свой любимый браузер. Клиентская часть приложения – Фронтенд – в свою очередь реагирует на пользовательский ввод и, если требуется, посылает HTTP запросы серверу для получения необходимой информации и затем рендерит изменения интерфейса. Серверная же часть приложения, так называемый бэкенд, принимает запросы от фронтенда и отправляет обратно ответы. На сервере производится большая часть бизнес логики приложения, а также работа с базой данный: запись, чтение. На Рисунке 1 справа расположена Tasks queue отвечающая за задачи, выполняемые в фоне. А именно задачи, которые не столь важно выполнить сразу же как они появились. В данном приложении как пример можно привести отправку пользователю на почту уведомлений о новый событиях, произошедших с контентом, на который он подписан, таких, как выход нового сезона сериала. Подробнее про Tasks queue можно прочесть в разделе 2.1.4.

Рисунок 2. Высокоуровневая архитектура приложения

## 2.1.1 Фронтенд

Фронтенд реализовал как SPA с использование библиотеки React.js. Single page application (SPA) – это веб-приложение, использующий единственный HTML-документ как оболочку для всех веб-страниц и организующий взаимодействие с пользователем через динамически подгружаемые HTML, CSS, JavaScript, обычно посредством AJAX. Одной из отличительных особенностей SPA является то, что роутинг или навигация между преставлениями интерфейса полностью производится на стороне фронтенда. И получается, что сервер не отправляет клиенту новый HTML-документ при переходе между страницами, как при стандартном подходе разработки web-сайтов. В итоге получаем бесшовный и более дружелюбный для пользователей интерфейс, больше напоминающий не web-сайт, а обычное desktop-приложение. У такого подхода, конечно же, есть свой недостатки. В простом случае весь код фронтенда (HTML, CSS, JavaScript) отправляется на клиент при первом запросе серверу, и, если приложение получилось достаточно большое, то TTE (Time to interactive) может значительно возрасти. Конечно, с этой проблемой можно довольно легко бороться, разбивая приложение на части и загружать только то, что необходимо на клиенте в данным момент или то, что с высокой вероятность понадобится в ближайшем будущем.

React.js [4] – одна из самых популярных JavaScript-библиотек для создания пользовательских интерфейсов. Основные черты:

* Декларативный. Достаточно описать, как части интерфейса приложения выглядят в разных состояниях. React будет своевременно их обновлять, когда данные изменяются;
* Компонентный подход. Разработчики создают инкапсулированные компоненты с собственным состоянием, а затем объединяют их в более сложные пользовательские интерфейсы;
* Мультиплатформенный. React помимо браузера может работать на сервере, используя Node.js и на мобильных платформах, используя React Native.

Как мне кажется, главным преимуществом React является виртуальный DOM. В отличие от браузерных DOM элементов, элементы React представляют из себя обыкновенные JavaScript объекты, которые относительно дёшево создавать. React берёт на себя синхронизацию виртуального и реального DOM, чтобы для пользователей состояние интерфейса было корректно. А мы, за счёт быстрых операций над виртуальный DOM, получаем выигрыш в производительности. Также к плюсам React можно отнести огромное количество готовых компонентов, которые можно найти и использовать у себя в проекте с помощью NPM.

## 2.1.2 Бэкенд

Серверная логика написана на языке программирования Python и основывается на фреймворке Django.

Django – высокоуровневый фреймворк для разработки web-приложений, который способствует быстрой разработки и имеет чистый и практичный дизайн. DRY – один из главных принципов фреймворка. Django является продуктов с открытым исходным кодом. Библиотека отличается быстротой, масштабируемостью и встроенными инструментами безопасности от большинства распространённых уязвимостей web-приложений таких, как SQL-инъекции и CSRF.

Возможности:

* ORM доступ к БД;
* Встроенный интерфейс администратора;
* Диспетчер URL на основе регулярных выражений;
* Расширяемая система шаблонов с тегами и наследованием;
* Система кеширования;
* Интернационализация;
* Архитектура приложений, которые можно использовать во множестве мест одновременно;
* Generic views – шаблонные функции контроллеров, позволяющие в несколько строчек написать часто используемые операции такие, как CRUD;
* Авторизация и аутентификация, подключение внешних модулей аутентификации;
* Система фильтров для построения дополнительных обработчиков запросов, как например включённые в дистрибутив фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий;
* Библиотека для работы с формами (наследование, построение форм по существующей модели БД);
* Встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное web-приложение;
* Интерфейс для использования стандартных Python библиотек для тестирования;
* Система для сериализации, которая может выдавать и принимать XML и JSON представления модели.

Архитектурно Django напоминает паттерн MVC. То, что в MVC называется контроллером, в Django носит название view. А логика представления (V – view в MVC) реализуется на уровне шаблонов. Уровень модели в Django не отличается от соответствующего в MVC. Часто можно услышать в сторону Django абревиатуру MTV – Model-Template-View.

Для каждой версии Django существует очень подробная документация с примерами и иллюстрациями. Также у фреймворка огромное сообщество разработчиков, поэтому для меня в ходе выполнения курсовой работы найти нужную информацию или готовые решения каких-то проблем было совсем нетрудно.

К минусам я бы отнёс малую гибкость фреймворка в некоторых моментах. То есть, если нужно реализовать что-то выходящие за рамки базовой бизнес логики, то придётся повозиться.

API реализовано с помощью Django Rest Framework.

## 2.1.3 База данных

Для работы с базой данных было использовано Django ORM. Сами данные абстрагированы в так называемые модели, которые являются частью MTV подхода, более детально описанного в предыдущем разделе.

Модель представляет из себя единственный источник информации о хранимых данных. Содержание модели записывается в описательной форме. Она содержит основные поля и поведение данных, которые мы храним. Как правило, каждая модель отображается в одну таблицу базы данных.

Ниже приведён пример модели ***Person***, которая имеет поля *first\_name* и *last\_name.*

**from** **django.db** **import** models

**class** **Person**(models.Model):

first\_name = models.CharField(max\_length=30)

last\_name = models.CharField(max\_length=30)

Для описанная выше модели будет создана таблица базы данных такого вид:

**CREATE** **TABLE** myapp\_person (

"id" serial **NOT** **NULL** **PRIMARY** **KEY**,

"first\_name" varchar(30) **NOT** **NULL**,

"last\_name" varchar(30) **NOT** **NULL**

);

Из примера выше сразу бросается в глаза то, что было добавлено поле *id,* которое не было явно указано. Django действительно автоматически добавляет в каждую модель поле *id* как первичный ключ таблицы. Но данное поведение переопределить, явно указав первичный ключ для какой-либо модели.

Стоит отметить, что, хотя пример создания таблицы выше использует синтаксис PostgreSQL, Django ORM официально поддерживает четыре бэкенда для баз данных: PostgreSQL, MySQL, SQLite и Oracle. Также существует множество внешних бэкендов для таких баз данных, как IBM Db2, SQL Anywhere, Firebird, Microsoft SQL Server и т.д.

Для разработки приложения в ходе выполнения данной курсовой работы была использована “in-memory” база данных SQLite. При хостинге на реальном сервере планируется использовать PostgreSQL для большей надёжности, безопасности и масштабируемости. Благодаря Django ORM, переход с одной базы данных на другую будет представлять из себя просто изменение нескольких строчек в файле конфигурации *settings.py.*

На Рисунке 2.2 представлены модели, используемые в web-приложении, и их связи между друг другом.

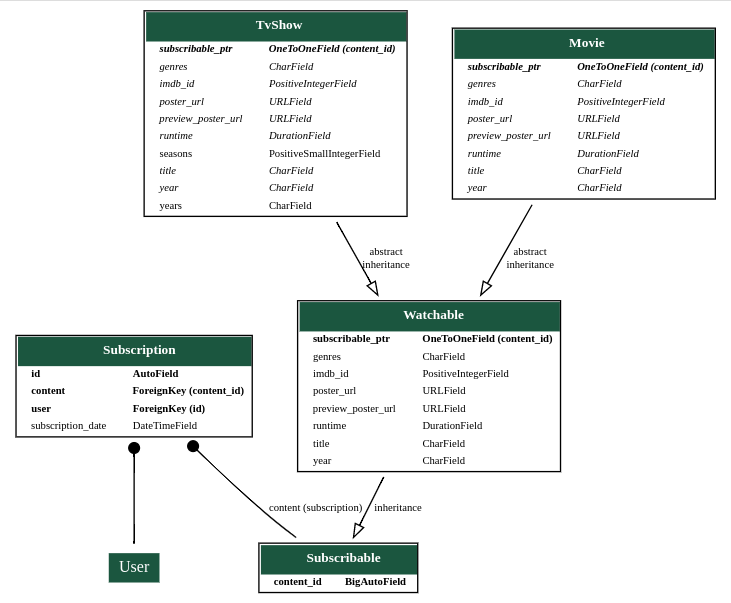


Рисунок 2. Архитектура базы данных

Вследствие использования ООП в моделях, таблицы базы данных включают небольшое дерево наследования. Базовая таблица *Subscribable* состоит всего из одной колонки *content\_id*. Эта таблица используется как связующее звено между таблицами для любого типа контента: в таблице с определённым типом контента (в данном случае сериалы и фильмы) поле, которое ссылается с помощью One-to-One связи её с таблицей *Subscribable*. Это сделано для того, что мы имели возможность оперировать с любым контентом без привязки к типу. Например, когда пользователь подписывается на что-то, нам не важно на что именно он подписался, потому что для любого типа у нас работает одна логика.

Абстрактная таблица *Watchable* связана с *Subscribable* One-to-One связью. Абстрактная означает, что эта таблица не будет создана в базе данных, но все поля, присутствующие в ней, будут добавлены в наследуемые таблицы. Таблица *Watchable* содержит общие для сериалов и фильмов поля. Такие, как название, год выхода, жанры, ссылки на постеры и т.д. От данном таблицы наследуются две *TvShow* и *Movie* для хранения данных о сериалах и фильмах, соответственно.

Для хранения информации о пользователях используется таблица User, которая создана из стандартной для Django модели пользователей. Она состоит из таких полей, как *username, email, password* и т.д.

Таблица *Subscription* отвечает за подписки пользователя на какой-то контент. Таких образом связь Many-to-Many между пользователем и контентом разбивается с помощью этой таблицы на две One-to-Many связи. Также в таблице присутствует поле *subscription\_date* для отслеживания, когда пользователь подписался.

Таким образом архитектура базы данных позволяет легко добавлять контент нового типа. Например, книги. Для этого нужно будет создать модель *Books*, которая будет наследоваться от базовой модели *Subscribable*.

## 2.1.4 Tasks queue

Для выполнения фоновых задач в приложении присутствует такая сущность, как Tasks queue. Для реализации в проекте использована библиотека huey [8]. В качестве брокера сообщений для очереди используется Redis [9].

Библиотека huye как интерфейс предоставляет несколько декораторов для функций таких, как *task* и *periodic\_task*, с помощью которых можно создать фоновые задачи и фоновые периодические задачи, соответственно. Если в задаче требуется доступ к базе данных, то необходимо использовать аналогичные *db\_task* и *db\_periodic\_task*, которые по завершению автоматически закрывают соединения с БД.

Как фоновые задачи в данном web-приложении реализованы проверка текущий подписок пользователя и отправка уведомления при необходимости. Также на сервере заведена периодическая задача для обновления информации в БД о сериалах и фильмах.

## 2.2 Аутентификация

В приложении, реализованном в рамках данной курсовой работы, было принято решение использовать аутентификация основанную на так называемом токене. Основная концепция, которая стоит за токен-аутентификации, довольно проста: предоставить пользователям возможность ввести их логин и пароль с целью получения токена, который позволяет им иметь доступ к какому-либо ресурсу не используя логин и пароль. Часто токен является временным, то есть после определённого периода времени токен становить не валидным – его нельзя больше использовать для получения доступа к ресурсу.

У Django Rest Framework присутствует встроенная токен аутентификация, но, к сожалению, у неё есть несколько существенных недостатков, а именно:

1. DRF токены имеют ограничение по количеству до одного токена на пользователя. Такое ограничение не способствует защищённому доступу с нескольких устройств. К тому же, придётся делать выход сразу со всех устройств, если требуется серверный выход из аккаунта (удаление токена).
2. DRF токены хранятся в базе данных в незашифрованном виде. Это позволит злоумышленнику, если БД будет скомпрометирована, получить неограниченный доступ к аккаунтам с активными токенами.
3. DRF токены сохраняют информацию о времени создания, но никакого встроенного механизма для ограничения времени жизни токена не предусмотрено.

Собственно, из-за изложенных выше проблем, в разработанном приложении использована токен аутентификация от knox [10]. Она даёт возможность использовать один токен на запрос авторизации, тем самым решая первую проблему. Также knox токены хранятся только в зашифрованной форме. То есть, если по каким-то причинам база данных будет украдена, у злоумышленников не будет возможности авторизоваться под чужим пользователем. По поводу третьей проблемы, у knox можно легко сконфигурировать время жизни токена с помощью параметра в файле settings.py.

После аутентификации пользователя сервер отправляет токен на клиент, которые в свою очередь сохраняет токен в специальном cookie с названием *auth\_token*. В последующем токен на клиенте используется для отправки в запросах, требующих авторизации. Так происходит, пока время жизни токена не истекло, и он валиден. Затем пользователю снова придётся произвести аутентификацию. Для удобства использования приложения, время жизни токена обновляется при каждом запросе с его использованием. То есть, если пользователь использует приложение на регулярной основе, ему не придётся постоянно авторизоваться.

## 2.3 Выводы

В данной главе была освещена верхнеуровневая архитектура приложения и были разобраны более подробно каждый из элементов представленной архитектуры: фронтенд, бэкенд, база данных, tasks queue. Было подробно описана используемая система аутентификации пользователей. А также упомянуты технологии, которые были задействованы в реализации приложения.

# ГЛАВА 3. ОБЗОР РЕАЛИЗОВАННОЙ ФУНКЦИОНАЛЬНОСТИ ПРИЛОЖЕНИЯ

## 3.1 Домашняя страница

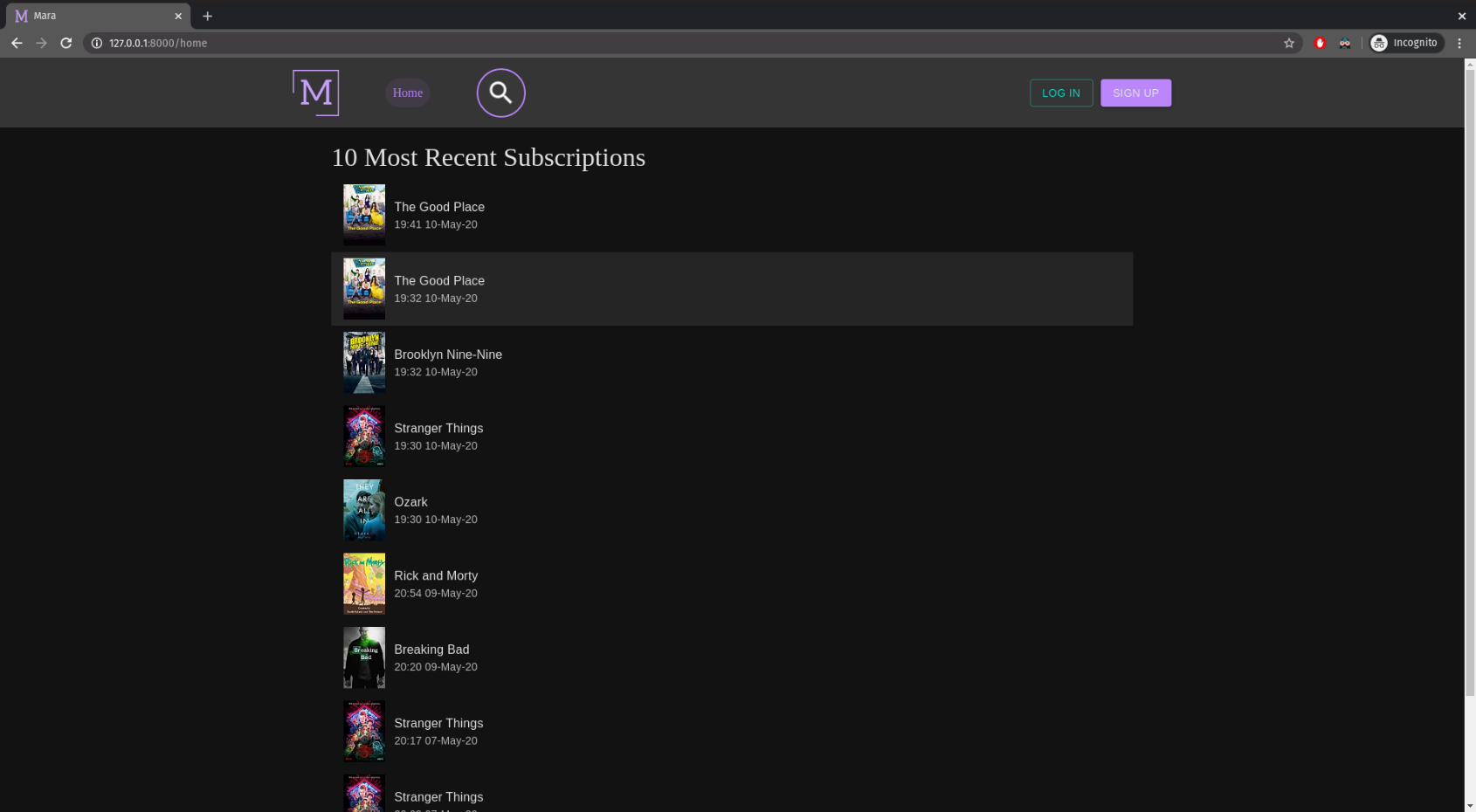
На Рисунке 3.1 изображено домашняя страница приложение. Она включает в себя секцию с десятью самыми последними подписками совершенными пользователями. По клику на один элемент секции происходит переход на страницу с подробной информацией о контенте элемента. Доступ на страницу не требует аутентификацию.

Рисунок .1. Домашняя страница

## 3.2 Панель навигации

Рассмотрим подробнее панель навигации, расположенную сверху страницы. На Рисунке 3.2 и Рисунке 3.3 панель продемонстрирована в состояниях, когда пользователь авторизирован и когда нет.

У авторизованного пользователя присутствует кнопка для перехода к списку своих подписок. Также два состояния отличаются стандартными кнопками для авторизации, регистрации и выхода из аккаунта.

Рисунок 3.3 Панель навигации для неавторизированного пользователя

Рисунок 3.2 Панель навигации для авторизированного пользователя

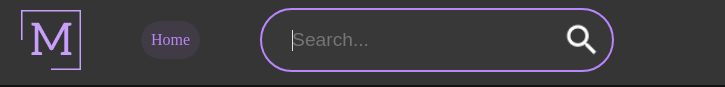
При нажатии на иконку лупы появляется поле для ввода запрос поиска по названию. На Рисунке 3.4 иллюстрировано активное состояние поля ввода запроса. При подтверждении ввода в поле поиска происходи перенаправление на страницу поиска, которая будет рассмотрена в разделе 3.5

Рисунок 3. Поле поиска в активном состоянии

## 3.3 Авторизация

Ниже на Рисунке 3.5 и Рисунке 3.6 продемонстрированы формы для логина и регистроции, соотвественно. Именно на форму логина происходит перенаправление, если неавторизированный пользователь пытается использовать функционал приложения требуещий авторизации.

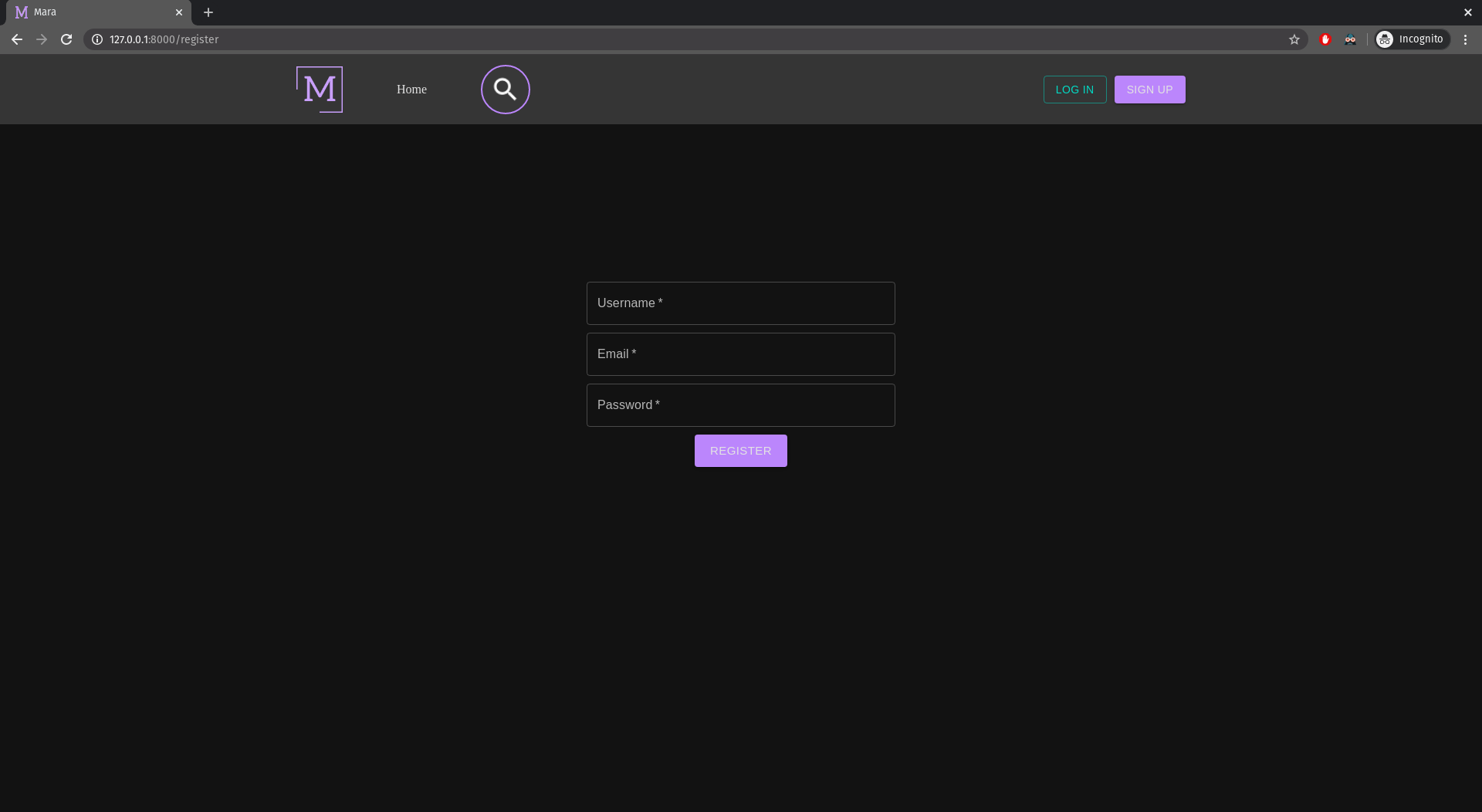
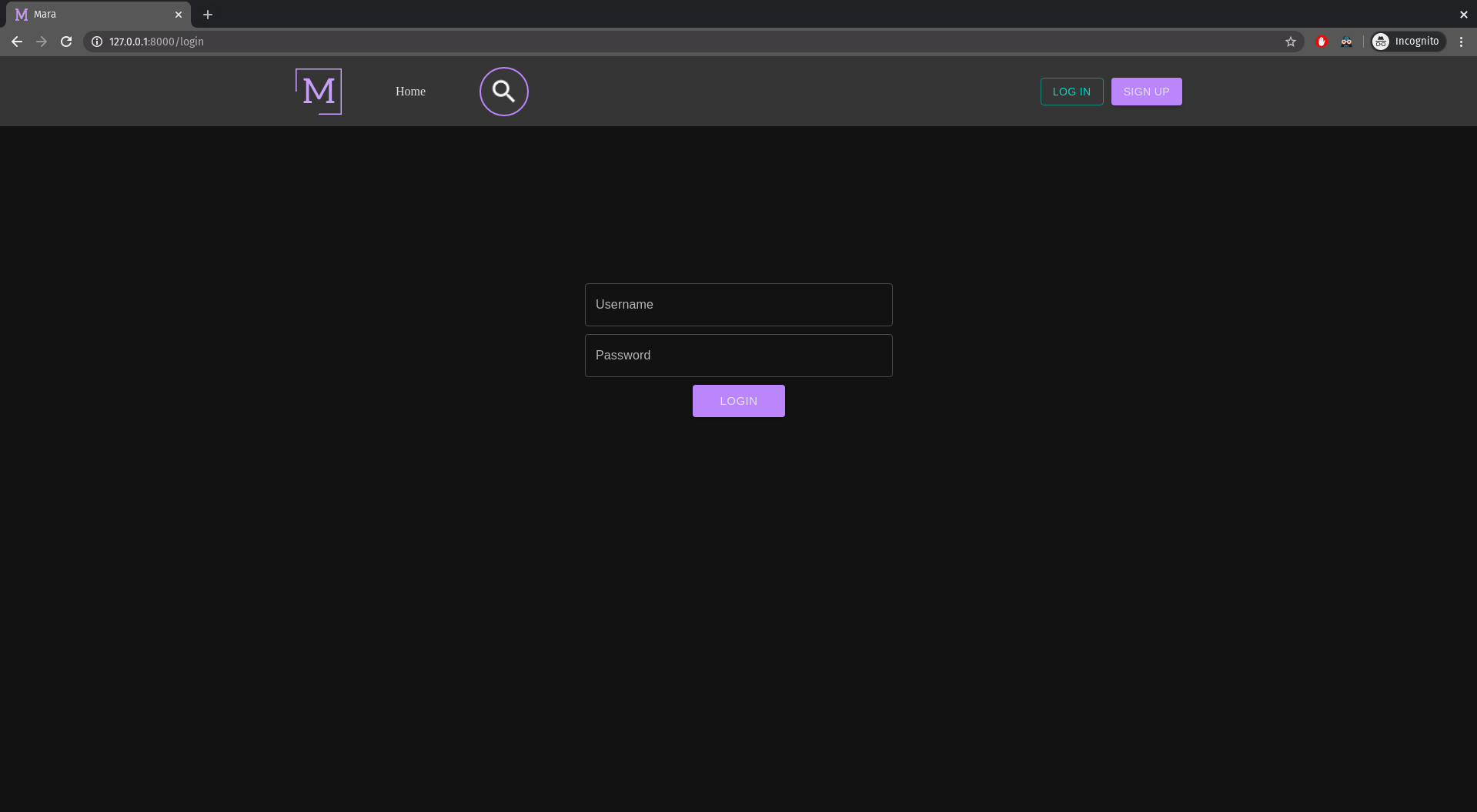
Поля в форме регистрации имеет верификацию ввода основаную на регулярных выражениях. Чтобы имя пользователя не было слишком длинное и включала только допустимые символы; чтобы email был правильного формата; чтобы пароль был больше шести символов и т.п.

Рисунок 3.6 Форма регистрации

Рисунок 3.5 Форма логина

## 3.4 Список подписок

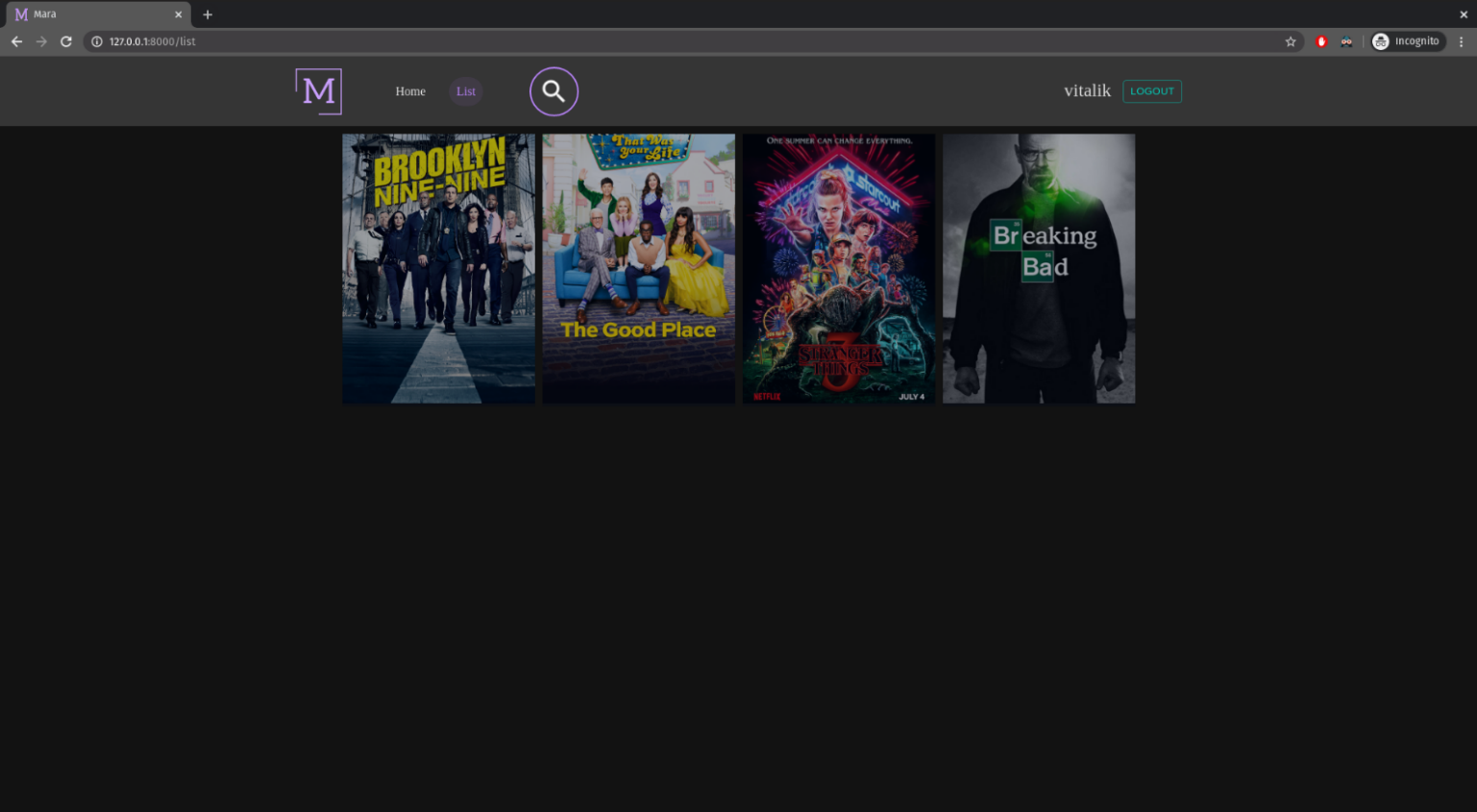
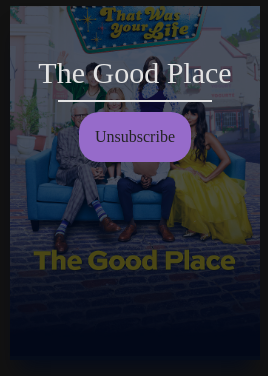
На Рисунке 3.7 продемонстрирован страница со списком подписок текущего пользователя.

Рисунок 3.7 Список подписок

Здесь у пользователя есть возможность отменить свои подписки. На Рисунке 3.8 показано карточка списка при наведении. Клик на карточку перенаправляет на страницу с детальной информацией (см. раздел 3.6).

## 3.5 Страница поиска

Рисунок 3.8 Карточка списка подписок

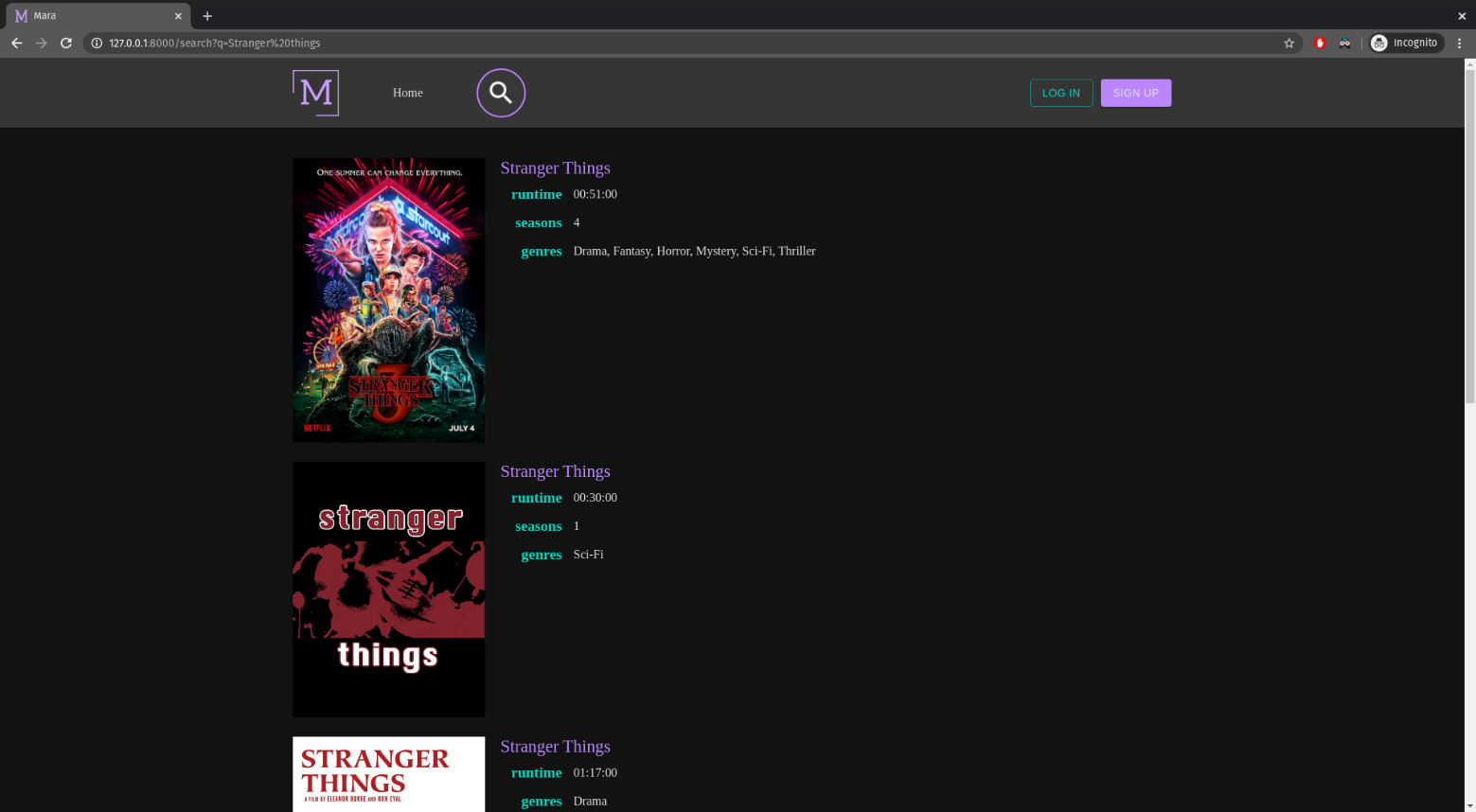
На Рисунке 3.9 показана часть страницы поиска для запроса “Stranger things”.

Рисунок 3.9 Страница поиска

Первоначально загружаются первые пять самых подходящих под запрос вариантов. При прокрутке вниз страницы происходит подгруздка следующих пяти вариантов. При клике происходит переход на страницу с детальной информации, которая будет рассмотрена в следующем разделе.

## 3.6 Страница с детальной информацией

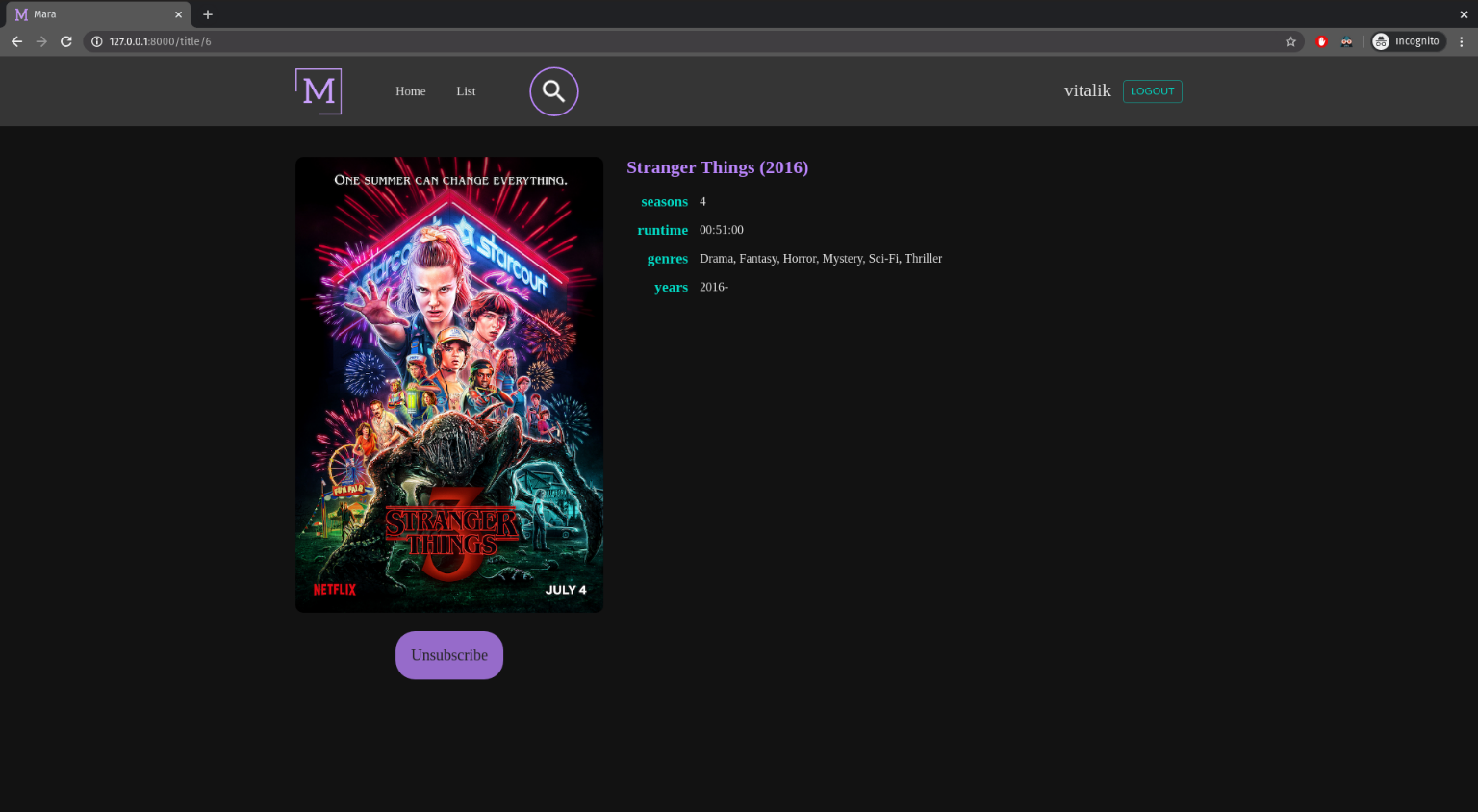
На Рисунке 3.10 продемонстрирована страница с детальной информацией о сериале “Stranger Things”. На этой странице можно найти такую информацию о сериале, как год выпуска, жанры, количество сезонов и т.д.

Рисунок 3.10 Страница с детальной информацией

Под постером сериала расположена кнопка для управления подпиской на данный сериал. По состоянию кнопки на Рисунке 3.10 можно понять, что у пользователя есть подписка на сериал.

## 3.7 Выводы

В главе под номером 3 был рассмотрен клиентский интерфейс приложения, а также его основные возможности по взаимодействию с пользователем.

# ЗАКЛЮЧЕНИЕ

В процессе выполнения курсовой работы были получены следующие результаты:

* сформулированы цели;
* выставлены функциональный требования к реализации;
* изучены и проанализированы основные аналоги, существующие в данной предметной области;
* продумана и разработана архитектура приложения;
* оценены преимущества и недостатки существующих технологий и выбраны среди них подходящие по требованиям;
* реализована web-приложение с раздельными бэкендом и фронтендом;
* Проведена демонстрации готового приложения.

# СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ

1. База данных о кинематографе IMDb [Электронный ресурс]. – Режим доступа: <https://www.imdb.com/>. – Дата доступа: 13.02.2020
2. Онлайн база открытых и закрытых пакетов для JavaScript [Электронный ресурс]. – Режим доступа: <https://www.npmjs.com/>. – Дата доступа: 14.02.2020
3. Подробная спецификация для разработки API с использованием json [Электронный ресурс]. – Режим доступа: <https://jsonapi.org/format/>. – Дата доступа: 02.04.2020
4. Спецификация для разработки для [Электронный ресурс]. – Режим доступа: <https://restfulapi.net/>. – Дата доступа: 03.04.2020
5. GitHub репозиторий библиотеки React.js [Электронный ресурс]. – Режим доступа: <https://github.com/facebook/react>. – Дата доступа: 04.04.2020
6. Документация фреймворка Django [Электронный ресурс]. – Режим доступа: <https://docs.djangoproject.com/en/3.0/>. – Дата доступа: 05.04.2020
7. Документация библиотеки Django Rest Framework [Электронный ресурс]. – Режим доступа: <https://www.django-rest-framework.org/>. – Дата доступа: 06.04.2020
8. GitHub репозиторий библиотеки очереди задач huey [Электронный ресурс]. – Режим доступа: <https://github.com/coleifer/huey>. – Дата доступа: 07.04.2020
9. GitHub репозиторий библиотеки Redis [Электронный ресурс]. – Режим доступа: <https://github.com/antirez/redis>. – Дата доступа: 08.04.2020
10. GitHub репозиторий модуля аутентификации для Django rest auth [Электронный ресурс]. – Режим доступа: <https://github.com/James1345/django-rest-knox>. – Дата доступа: 09.04.2020