**A logo with black text

Description automatically generatedТЕХНИЧЕСКИ УНИВЕРСИТЕТ – СОФИЯ**

Факултет Приложна математика и информатика

**ДИПЛОМНА РАБОТА**

Тема: Уеб базирана система за управление на манга колекции

|  |  |
| --- | --- |
| *Дипломант:* Кристиян Петров Кръчмаров |  |

Фак. №: 181220017

*Специалност:* Приложна Математика и Информатика

*Образователно- квалификационна степен*: **бакалавър**

Дипломен ръководител: доц. д-р Анна Розева

София 2024



Съдържание

[1 Съкращения 1](#_Toc167551299)

[2 Абстракт 2](#_Toc167551300)

[3 Структура на приложението и използвани технологии 3](#_Toc167551301)

[3.1 Сървърна част 3](#_Toc167551302)

[3.1.1 PostgreSQL 3](#_Toc167551303)

[3.1.2 ASP.NET Web API 3](#_Toc167551304)

[3.1.2.1 Entity Framework Core (EF Core) 3](#_Toc167551305)

[3.1.2.2 AutoMapper 3](#_Toc167551306)

[3.1.2.3 AspNetCore Identity 3](#_Toc167551307)

[3.1.3 Redis 4](#_Toc167551308)

[3.1.3.1 AspNetCore.OutputCaching.StackExchangeRedis 4](#_Toc167551309)

[3.1.4 Json Web Token (JWT) 4](#_Toc167551310)

[3.1.4.1 AspNetCore.Authentication.JwtBearer 5](#_Toc167551311)

[3.1.5 REST 5](#_Toc167551312)

[3.2 Потребителски интерфейс 5](#_Toc167551313)

[3.2.1 Blazor 5](#_Toc167551314)

[3.2.1.1 MudBlazor 6](#_Toc167551315)

[3.2.1.2 Java Script interoperability (JSinterop) 6](#_Toc167551316)

[3.2.2 Google Charts 6](#_Toc167551317)

[4 Структура на базата данни 6](#_Toc167551318)

[4.1 LibraryMangas и Authors 7](#_Toc167551319)

[4.2 UserMangas 8](#_Toc167551320)

[4.3 Orders 9](#_Toc167551321)

[4.4 Таблици за потребителите 9](#_Toc167551322)

# Съкращения

REST

JWT

API

ACID

HTTP

ORM

DOM

HTML

SVG

# Абстракт

С нарастващата популярност на Японската култура в световен мащаб и мангата като един от най-емблематичните ѝ представители, все повече хора започват да четат, събират и колекционират различни заглавия. Мангата е форма на комикс, който обхваща широк спектър от жанрове и тематики, привличайки много хора със своя стил и културна значимост. Въпреки това, управлението на колекции от манга може да бъде предизвикателно и да отнема много време.

Необходимостта за информационна система, която да улеснява управлението на колекции и поръчки е нараснала значително, с технологическия напредък и дигитализацията на повечето аспекти от нашия живот. Основни проблеми пред хората са: загуба на данни, дублиране на купени заглавия и трудности с проследяването на поръчките. Съществуват методи за управление, като ръчни записи и електронни таблици, но не предоставят достатъчна ефективност и удобство.

Целите на приложението са да предостави на потребителите централизирана система за управление на своите колекции и поръчки, както и да предложи визуализация за колекцията и за похарчените средства.

Основните функционалности ще бъдат:

* Набор от готови заглавия: Потребителите ще имат достъп до готов списък от заглавия, както и да виждат информация за всяко произведение.
* Създаване на дигитална колекция: Потребителите ще могат да избират от предоставените заглавия и да добавят към своята колекция, като въвеждат информация, свързана с тяхното отношение към творбата.
* Регистрация на поръчки: Потребителите ще могат да добавят и редактират информация за поръчките си.
* Визуализация на данни: Приложението ще генерира графики и статистки, които ще помагат на потребителя за анализира своята колекция и своите поръчки, както и да има по-добро разбиране за своите разходи.

# Структура на приложението и използвани технологии

Приложението се състои от две основни части: сървърна част и потребителски интерфейс. Комуникацията между двете части се извършва чрез REST интерфейс.

## Сървърна част

Сървърната част отговаря за бизнес логиката, обработката на данни, кеширане и комуникацията с базата данни. За реализацията са използвани следните технологии.

### PostgreSQL

PostgreSQL, или по честно наричана Postgres е релационна база данни с отворен код. Postgres позволява на потребителите да създават собствени типове данни, под формата на обекти, върху които могат да бъдат прилагани функции като наследяване и полиморфизъм. Също така се поддържат транзакции с ACID свойства и поддържка на други езици, освен SQL.

### ASP.NET Web API

ASP.NET Web API е framework който позволява лесно създаване на уеб приложения. Той е част от ASP.NET Core платформата и предоставя голям брой библиотеки за разработването на REST базирани уеб услуги.

#### Entity Framework Core (EF Core)

EF Core е библиотека, която позволява връзката и интеракцията между базата данни и проекта. Библиотеката работи на ORM принципа, при който записи от таблица в базата биват превърнати в обект, който може да се използва от обектно ориентиран език.

#### AutoMapper

AutoMapper е библиотека която позволява лесното преобразуване от един обект в друг.

#### AspNetCore Identity

AspNetCore Identity позволява мениджмънт и съхраняване на потребителски акаунти, възможните роли и възможните права, които може един потребител да притежава.

### Redis

Redis e нерелационна база данни, която работи на принципа „ключ-стойност“. Най честото ѝ предназначение е за кеширане. Дизайнът на Redis позволява ниско латентни операции, защото информацията се съхранява на паметта, а не на диска.

#### AspNetCore.OutputCaching.StackExchangeRedis

AspNetCore.OutputCaching.StackExchangeRedis e библиотека, която имплементира кеширане, чрез Redis. Output Caching е междинен слой в приложението, който позволява кеширането на заявката, която ще бъде подадена към сървъра, както и отговора, който трябва да се върне, във двойка „ключ-стойност“.

### Json Web Token (JWT)

JWT е стандарт за размяна на информация между две страни в JSON формат. Самият жетон има три части, който са кодирани в Base 64 формат, разделени с точка. Неговите части са Заглавен ред (Header), Тяло (Payload) и Подпис (Signature).

В заглавния ред се съдържа информация за типа токън и алгоритъма, с който е подписан. Най често се използва HS256.

В тялото се съдържа информация за потребителя и някои стандартни полета, като идентификатор, издател, кога е издаден и други.

Подписа се генерира като неподписания токън, който представлява комбинация от кодирания заглавен ред и кодираното тяло разделени с точка, се криптира с таен ключ и избрания алгоритъм.

A screenshot of a computer code

Description automatically generated

Фигура 1: Подписване на JWT

Крайният резултат е трите кодирани части, разделени с точка.

A close up of a computer code

Description automatically generated

Фигура 2: Примерен JWT

#### AspNetCore.Authentication.JwtBearer

AspNetCore.Authentication.JwtBearer е библиотека, в която има имплементация за валидиране и създаване на JWT както и позволява да се използва оторизация, базирана на JWT, посредством междинен слой.

### REST

REST е архитектурен стил за реализиране на уеб услуги между клиент и сървър, които си комуникират с HTTP заявки. Заявките носят в себе си следните неща: HTTP метод (GET, POST, PUT, DELETE), Заглавни редове (Headers) с информация свързана за оторизиране, кеширане и други мета данни и Body в което се съдържа информацията. Отговорите имат във себе си статус код, който показва какво се е случило, Headers, в които има информация за сървъра изпълняващ заявката и Тяло, в което се съдържа информацията, в повечето случаи в JSON формат

## Потребителски интерфейс

Потребителският интерфейс се грижи за визуализацията на данните и взаимодействието с крайния потребител. Реализацията на потребителския интерфейс са използвани технологиите

### Blazor

Blazor e framework, който позволява създаването на интерактивен потребителски интерфейс, базиран на компонентния модел, но написани чрез езика C#, а не JavaScript (JS). Възможни са 2 вида хостинг модели: Blazor Web Assembly (WASM), където кода се компилира и изпълнява във браузъра и Blazor Server, където всички операции се изпълняват на сървъра и биват комуникирани към клиента чрез SignalR, библиотека позволяваща за асинхронно изпращане на информация към клиента.

#### MudBlazor

MudBlazor е библиотека с готови Razor компоненти, която улеснява създаването и разбиране на елементи от потребителския интерфейс.

#### Java Script interoperability (JSinterop)

JSinterop e функционалност, която позволява на сървъра да изпълни JavaScript код, както и JS кода да изпълни C# код, когато е необходимо, Такива случаи са когато трябва да се достъпи DOM-a самия браузър или други софтуери, от трети страни.

### Google Charts

Google Charts е библиотека за генериране на различни видове диаграми във HTML или SVG формат, чрез JS код.

# Структура на базата данни

Базата е генерирана чрез подхода „Първо код“ (Code first),в който първо се създават модели на отделните таблици и релациите между тях в кода и след това се създава миграция, която описва промените които трябва да бъдат изпълнени върху базата. Чрез този метод се позволява лесна поддръжка и разширение на базата.

За да се реализира този подход е нужно да се напиша клас, който да наследи DbContext. Този клас е отговорен за връзката с базата, както и за управлението на различните таблици в нея.

A screen shot of a computer code

Description automatically generated

Фигура 3: Структура на DbContext

Всяка една променлива от тип DbSet<T> е репрезентация на таблица от базата. Таблиците на потребителя ще бъдат генерирани от родителския клас IdentityDbContext<User>, където User е модела за потребителски данни. След генериране на миграцията и нейното прилагане към базата получаваме показаната на Фигура 4 схема на базата данни.

A computer screen with many black and white text

Description automatically generated with medium confidence

Фигура 4 Схема на базата данни

## LibraryMangas и Authors

Таблиците LibraryMangas и Authors са предназначени за възможните заглавия, от които потребителите могат да избират, както и информация за авторите.

Таблицата LibraryMangas има следните колони

* Id – идентификатор, който служи за първичен ключ (primary key)
* TitleRomaji – заглавието на латиница
* TitleEnglish – заглавието на английски. Възможно е заглавието да го няма, защото творбата може да не е преведена.
* TitleJapanese – заглавието на японски
* DemographicType – тип на демография, към която е насочена творбата. Възможни са стойностите
  + Shounen –за момчета до 18 години
  + Shoujo –за момичета до 18 години
  + Seinen –за мъже над 18 години
  + Josei –за жени над 18 години
* Type – тип на творбата. Възможни са 3 стойности
  + Мanga – манга
  + LightNovell – новела
  + OneShot – Самостоятелна история в един том.
* PublishingStatus – статус на публикуване с 3 възможни стойности
  + Publishing – все още се публикува
  + Finished – публикуването е приключило
  + OnHiatus –публикуването е спряно временно.
* TotalVolumes – обща бройка томове. Възможно е бройката да я няма, защото все още се публикува творбата.
* MainImageURL – Линк към снимка на първия том. Възможно е да няма линк, защото творбата да няма все още издаден първи том.
* Synopsis – Описание

За таблицата Authors колоните са

* Id – идентификатор, първичен ключ
* FirstName – Първо име на автора. В някои случаи това име може да го няма, защото автора е използва псевдоним.
* LastName – Фамилно име на автора. В тази колона се записа псевдонима на автора, ако е използван такъв
* Role – неговата роля в творбата. Възможни са само 3 стойности в тази колона
  + Story – автора е отговорен само за историята
  + Art – автора е отговорен само за рисунките и арт стила
  + StoryAndArt – aвтора отговаря и за двете

Между тези две таблици съществува релация от вида много към много (many to many), осъществена чрез междинната таблица AuthorLibraryManga

## UserMangas

В тази таблица се съдържа информация за взаимоотношението между дадена творба и даден потребител. Съдържат се следните колони

* Id – идентификатор, първичен ключ
* ReadingStatus – статус на четене с възможни стойности
  + Reading – все още произведението се чете
  + Finished – произведението е прочетено от потребителя
  + OnHold – потребителя е спрял временно да го чете
  + Dropped – потребителя се е отказал да дочете произведението
  + PlanToRead – потребителя планира да чете произведението в бъдещето
* CollectionStatus – статус на колекциониране. Възможни са стойностите
  + Collected – творбата е събрана
  + InProgress – творбата е в процес на събиране
  + PlanToCollect – потребителя планира да събере творбата в бъдеще
* ReadVolumes – брой прочетени томове
* CollectedVolumes брой събрани томове
* PricePerVolume – цена за един том
* UserId – идентификатор на потребителя, към който е асоциирана информацията, външен ключ (foreign key)
* LibraryMangaId идентификатор на творбата, за която се отнася потребителската информация; външен ключ

## Orders

Таблицата Orders е предназначена за информацията, относно поръчките на потребителите. Нейните колони са

* Id – идентификатор, първичен ключ
* Date – дата на поръчване
* Status – статус на поръчката, с възможни стойности
  + Created – поръчката е създадена
  + OnTheWay – поръчката е в процес на доставяне
  + Delivered - доставена
* Description – описание за закупените артикули
* Place – място на направа на поръчката
* Amount – стойност на поръчката
* NumberOfItems – брой артикули, които са закупени
* UserId – идентификатор на потребителя; външен ключ

## Таблици за потребителите

Защото използваме AspNetCode Identity, са генерирани множество таблици които са за мениджмънт на потребителската информация. За целите на нашия проект ние ще използваме само две таблици: AspNetUsers и AspNetRoles. В таблицата AspNetUsers се намира основната информация за потребителя като потребителско име, имейл адрес, парола, която е хеширана и опреснителен жетон (refresh token). В таблицата AspNetRoles се намира информация за възможните роли в приложението. Между тези две таблици има релация много към много, осъществена чрез таблицата AspNetUserRoles.