Курсов проект

Дисциплина: Проектиране и интегриране на софтуерни системи

Реализация на системата

Версия 1.x

|  |  |  |
| --- | --- | --- |
| Фак. № | Име на студент | Секция от документа |
| 2MI0600038 | Румен Тошев | 1,2, |
| 0MI0600039 | Цветан Тошев | 3,4,7 |
| 8MI0600125 | Петър Николов | 5,6,8 |
|  |  |  |
|  |  |  |
|  |  |  |

Януари, 2025

Съдържание

[1 Въведение 3](#_Toc374967428)

[1.1 Цел 3](#_Toc374967429)

[1.2 Резюме 3](#_Toc374967430)

[1.3 Дефиниции и акроними 3](#_Toc374967431)

[2 Използвани технологии 5](#_Toc374967432)

[3 Реализация на базата от данни 6](#_Toc374967433)

[4 Реализация на бизнес логиката 8](#_Toc374967434)

[5 Реализация на потребителския интерфейс 10](#_Toc374967435)

[6 Внедряване на системата 18](#_Toc374967436)

[7 Разпределение на дейностите по реализацията 22](#_Toc374967437)

[8 Приложения 23](#_Toc374967438)

# Въведение

## Цел

Системата “HW Manager” има за цел да улесни управлението на вашата колекция от HotWheels колички, като събере на едно място всички ваши модели, като предоставя възможност за разпределяне по различни характеристики в колекции и откриване на нови модели на други потребители на системата.

Целта на този документ е да опише всичко свързано със системата, а именно нейната структура (разделение на системата и структура на базата данни), връзката между различни части на системата (как и защо са свързани), различните технологии и приложения използвани за изграждането на системата, както и пакети и библиотеки, нужни за реализацията. В документа освен информация за процеса и начина, по който е създадена системата, ще откриете и инструкции как да се стартира системата.

## Резюме

Документът ще разгледа основните стъпки и решения, които са използвани за успешното реализиране на проекта. Това ще бъде постигнато и структурирана както следва:

1. Използвани технологии – В тази точка ще се разгледат различните технологии и причините именно те да бъдат избрани при разработката
2. Реализацията на базата данни – Тук ще се опишат различните таблици в базата данни, атрибутите, типовете, както и връзките между различните таблици
3. Реализация на бизнес логиката – В тази точка ще бъде описано разделението на системата и взаимодействието между различните части на системата
4. Реализация на потребителския интерфейс – Тук ще опише всичко свързано със създаването на потребителския интерфейс и ще се покажат екранни снимки от системата
5. Внедряване на системата – Подробни инструкции с полезни връзки как може да стартирате системата и да я използвате
6. Разпределение на дейностите по реализация – В тази точка ще се опише кой член от екипа какво е направил
7. Приложения – приложения използвани при реализацията

## Дефиниции и акроними

*Дефинирайте всички термини, понятия и акроними, използвани в документа.*

***Бекенд (Backend)*** *– Частта от софтуерното приложение, която работи "зад кулисите". Това включва сървъри, бази данни, бизнес логика и API-та, които обработват заявки, съхраняват и извличат данни.*

***Фронтенд (Frontend)*** *– Потребителският интерфейс на уеб или мобилно приложение. Това е всичко, което потребителят вижда и с което взаимодейства – бутони, менюта, текстови полета и други елементи, обикновено изградени с HTML, CSS и JavaScript.*

***Фреймворк (Framework)*** *– Софтуерна структура, която предоставя предварително изградени компоненти и правила за разработка, улеснявайки програмистите в създаването на приложения.*

***ASGI*** - *Asynchronous Server Gateway Interface*

***ORM (Object-Relational Mapping)*** – *Технология, която позволява на програмистите да работят с бази данни, използвайки обектно-ориентирано програмиране (OOP), вместо да пишат SQL заявки ръчно.*

***Билд процес (Build Process)*** *– Това е автоматизираният процес на преобразуване на изходния код на дадено приложение в изпълнима форма, готова за разгръщане (deployment).*

***UUID*** - *Universally Unique Identifier*

***Open-source -*** *модел на разработка на софтуер, при който изходният код е публично достъпен за разглеждане, промяна и разпространение. Това означава, че всеки може да използва, модифицира и допринася за развитието на софтуера.*

***RESTful endpoints*** *– Това са URL адреси (или пътища), които се използват в* ***RESTful*** *(Representational State Transfer) архитектурни стилове за взаимодействие с уеб услуги. В REST API, всяко действие (например създаване, четене, актуализиране или изтриване на данни) се изпълнява чрез изпращане на HTTP заявки към определени endpoints, които са свързани с ресурси.*

***Fetch операция*** *– В контекста на уеб разработки, fetch е термин, който се използва за извличане на данни от сървър, обикновено чрез HTTP заявки.*

# Използвани технологии

За разработването на бекенд частта на проекта използвахме Python и модерния и високоефективен фреймуърк за разработка на уеб приложения с Python – FastAPI. Бекендът предоставя RESTful API за управление на данните, свързани с количките на даден профил, включително създаване, изтриване, извличане на записи, създаване на колекции, добавяне и премахване на отделни колички към тях, тяхното редактиране и изтриване. Фреймуърка предоставя асинхронни функции поради факта, че е построен върху ASGI.Също така има и поддръжка на автоматична документация с OpenAPI и Swagger UI, което улеснява интеграцията и тестването. Друг огромен плюс е наличието на вградена поддръжка за валидиране на входни данни чрез Pydantic.

За фронтенд частта е използван React + Vite и JavaScript. Тази комбинация осигурява висока ефективност и производителност по време на разработката и използването на приложението. React предоставя възможността за компонентно-базиран подход на разработка, както и лесно управление на състоянията и следене за промени, което подобрява функционалността на потребителския интерфейс. Vite, от друга страна, допълва със своята изключителна бързина при билд процеса и стартиране. Чрез него имаме възможността по време на работа директно да виждаме направените промени по кода, тъй като те се отразяват веднага без нужда от презареждане на страницата от страна на разработчика. Още един плюс, от който се възползваме е мързеливото зареждане на модули, което значи че даден модул се зарежда само когато е необходимо. По този начин намаляваме времето за зареждане и увеличаваме производителността.

За базата данни сме използвали PostgreSQL. Тя съчетава бързина и надеждност и е съвместима с FastAPI посредством ORM като SQLAlchemy. Също така, използваме и Alembic, което и инструмент чрез който управляваме миграциите към базата данни. Това гарантира синхронизираност между базата данни и моделите в бекенда.

# Реализация на базата от данни

*Опишете типа и структурата на базата от данни, ако има такава.*

Както бе споменато, за базата данни използваме PostgreSQL, която е open-source обектно-релационна система база данни.

За проекта са използвани 4 таблици, чрез които успешно да се реализира целта му. Във файла models.py в папката Models (Models/models.py) се намират моделите на самите таблици. Те се дефинират чрез Python класове с помощта на SQLAlchemy ORM. Там са обозначени и типовете на различните колони в таблиците. Класовете са следните:

* User(име на таблица - users) – В нея се пази информацията за регистрираните потребители. Атрибути:
  + id - Уникален идентификатор
  + username - Уникално потребителско име.
  + email - Имейл на потребителя.
  + hashed\_password - Хешираната парола на потребителя.
* HWCarPost(име на таблица - hw\_cars) – В нея се пази информация за всяка една количка, качена от потребител на сайта. Атрибути:
  + id - Уникален идентификатор
  + userID – идентификатор на потребителя, който е качил количката
  + username - потребителско име на потребителя, който е качил количката
  + image\_url - URL на изображението на количката
  + model – модел на количката
  + year – година на производство на количката
  + type – вид на количката
  + series – серия на количката
  + color – цвят на количката
* Collection(име на таблицата – collections) – В нея се пази коя количка към коя колекция на даден потребител е добавена. Атрибути:
  + id - Уникален идентификатор
  + userID - Идентификатор на потребителя качил количката
  + collection\_name - Име на колекцията, към която е добавена количката
  + postID - Идентификатор на публикацията.
* CollectionNames(име на таблицата - collections\_names) – В нея се пази информация за колекциите на потребителите. Атрибути:
  + id - Уникален идентификатор
  + userID – Идентификатор на потребителя
  + collection\_name – име на колекцията на потребителя

Връзките между таблиците и информацията в тях се извършва чрез уникалните идентификатори на даден потребител и дадена качена количка(пост). Полето userID във всяка таблица ни дава връзка с конкретен потребител с негова количка или колекция, а полето postID създава връзката между качена количка и колекция. За тези уникални идентификатори използваме UUID(Universally Unique Identifier).

Картина, която съдържа текст, диаграма, линия, Паралелен

Описанието е генерирано автоматично

# Реализация на бизнес логиката

*Опишете реализацията на бизнес логиката на системата с избраните технологии и програмни средства.*

*Тази секция може да се структурира от гледна точка на модулите на системата или въз основа на реализираните класове и интерфейси.*

Системата е разделена на два главни модула – бекенд и фронтенд. За бекенда както споменахме е използван Python + FastAPI, а за фронтенда – React + Vite. В бекенд модула главната част от системата е реализирана във файла main.py. В този файл се намират абсолютно всички CRUD операции, както и моделите на използваните класове, които наследяват базов клас от библиотеката pydantic, което ни позволява да се извършва проверка на типовете на класовете автоматично. CRUD операциите са реализирани чрез RESTful endpoints, пътища, посредством, които фронтенда може да изпраща заявки и да получава отговори.

-Пример за модел използващ pydantic – BaseModel клас, предоставящ базов интерфейс за класовете

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

-Пример за endpoint, като всеки такъв приема информация, изпълнява дадена задача и връща отговор във формата JSON.

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

Другите 2 части в бекенда са реализирани в папките Database и Models, в които, както вече бе споменато, става свързването с базата данни и описването на таблиците за базата данни, всички атрибути, имена и стойности.

Фронтенда посредством fetch – операция към пътищата описани в бекенда осъществява връзка с базата данни за получаване и изпращане на информация посредством заявки. Различните страници и компоненти се намират в папките Pages и Components.

Обобщено, работния процес се дели на 4 части:

1. Потребителят взаимодейства с React интерфейса.
2. Фронтендът изпраща заявка към бекенда чрез API.
3. Бекендът обработва заявката, валидира входните данни и комуникира с базата данни.
4. Отговорът се връща към React приложението и се показва на потребителя.

# Реализация на потребителския интерфейс

*Опишете реализираните елементи на потребителския интерфейс. Приложете примерни екранни форми.*

1. Общ преглед на интерфейса

Проектът е реализиран с помощта на библиотеката React Material-UI (MUI) за създаване на стилен и адаптивен потребителски интерфейс. Използвахме тази библиотека, защото React Material-UI (MUI) е популярна библиотека за изграждане на потребителски интерфейси, която предлага богат набор от предварително дефинирани, стилни и достъпни компоненти, чрез които можем да изграждаме наши такива, отговарящи на нашите нужди.

За управление на навигацията между страниците използвахме React Router. Тя е една от най-популярните библиотеки за управление на маршрутизация в React приложения, за това избрахме и точно нея. Тя осигурява лесен и гъвкав начин за навигация между различни страници или изгледи в приложението.

2. Използвани библиотеки и технологии

2.1 React Material-UI (MUI)

Използвани са основни компоненти като:

Button: Стандартен компонент за бутони в MUI, който поддържа различни вариации (контури, цветове и размери).

TextField: За формови полета за въвеждане на данни.

Card: За подреждане на информацията в добре структурирани контейнери.

Grid: Система за оформление, базирана на мрежа, която улеснява създаването на адаптивни дизайни.

Typography: Основен компонент за работа с текст в MUI. Осигурява различни стилове и типове текст, от заглавия до обикновени параграфи.

Paper: Компонент за оформление, който служи като контейнер с фоново оформление, наподобяващо хартия. Използва се за визуална организация на съдържанието и добавяне на сянка за подчертаване.

IconButton: Бутон, предназначен за използване с икони.

Menu, MenuItem: Създават падащо меню за избор на опции.

Select: Използва за създаване на падащ списък с опции за избор.

InputLabel: описва елемента, който ще бъде контролиран, например Select.

FormControl: Контейнер за свързани компоненти на формуляри, като InputLabel и Select, и помага за организирането им.

2.2 React Router

Реализирана е структура от маршрути за основните страници на приложението:

<Route path="/signIn" element={<SignIn/>} /> - Страница за влизане в системата за потребител имащ съществуващ профил.

<Route path="/signUp" element={<SignUp />} /> - Страница за вписване в системата на нерегистриран потребител.

<Route path="/profile" element={<ProfilePage />} /> - Страница с потребителския профил на потребител влезнал в системата.

<Route path="/addCar" element={<AddCar />} /> - Страница за добавяне на кола към колекцията на вписания потребител.

<Route path="/myCollection" element={<MyCollectionPage />} /> - Страница за преглед на цялата колекция на потребител.

<Route path="/explore" element={<Explore />} /> - Страница показващ част от колекцията на други потребители използващи системата.

<Route path="/collections" element={<Collections />} /> - Страница за създаване на различни колекции и техният преглед по такива.

<Route path="coll/:collection\_name" element={<DisplayCollection />} /> - Страница, която визуализира определена колекция

<Route path="/homeNotLoged" element={<HomePageNotLoged />} /> - Начална страница за потребител, който не е вписан в системата.

Картина, която съдържа текст, екранна снимка, дисплей, софтуер

Описанието е генерирано автоматично

2.3 Собствени компоненти

CarCollectionDisplay.jsx – Елемент, помагащ за визуализацията на колекция от колички

Картина, която съдържа текст, превозно средство, екранна снимка, кола

Описанието е генерирано автоматичноCarDisplay.jsx – Елемент за визуализирането на даден модел количка, която е на друг потребител на системата

Картина, която съдържа текст, екранна снимка, софтуер, дисплей

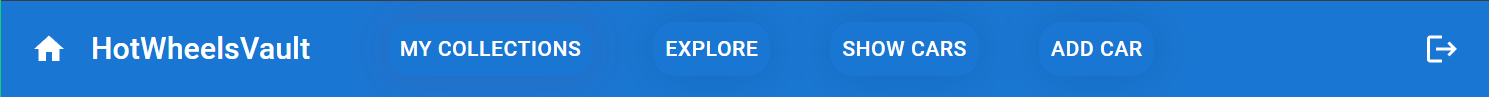
Описанието е генерирано автоматичноDataGrid.jsx – Елемент използван за представянето на различните колекции на потребител и тяхното създаване

Картина, която съдържа текст, превозно средство, кола, колело

Описанието е генерирано автоматичноMyCarDisplay.jsx - Елемент за визуализирането на даден модел количка

NavBarFinale.jsx – Елемент, който се използва за навигиране между различните страници

NavBarNotLoggedFinale.jsx - Елемент, който се използва за навигиране между различните страници на потребител, който не се е вписал в системата



Картина, която съдържа текст, екранна снимка, Шрифт, Правоъгълник

Описанието е генерирано автоматичноPopUpFinale.jsx – Елемент, използван за известяване на потребителя за успешно/неуспешно извършено действие по добавяне на количка в неговата колекция

Това са екранни снимки на приложението:

* Картина, която съдържа текст, екранна снимка, софтуер, Мултимедиен софтуер

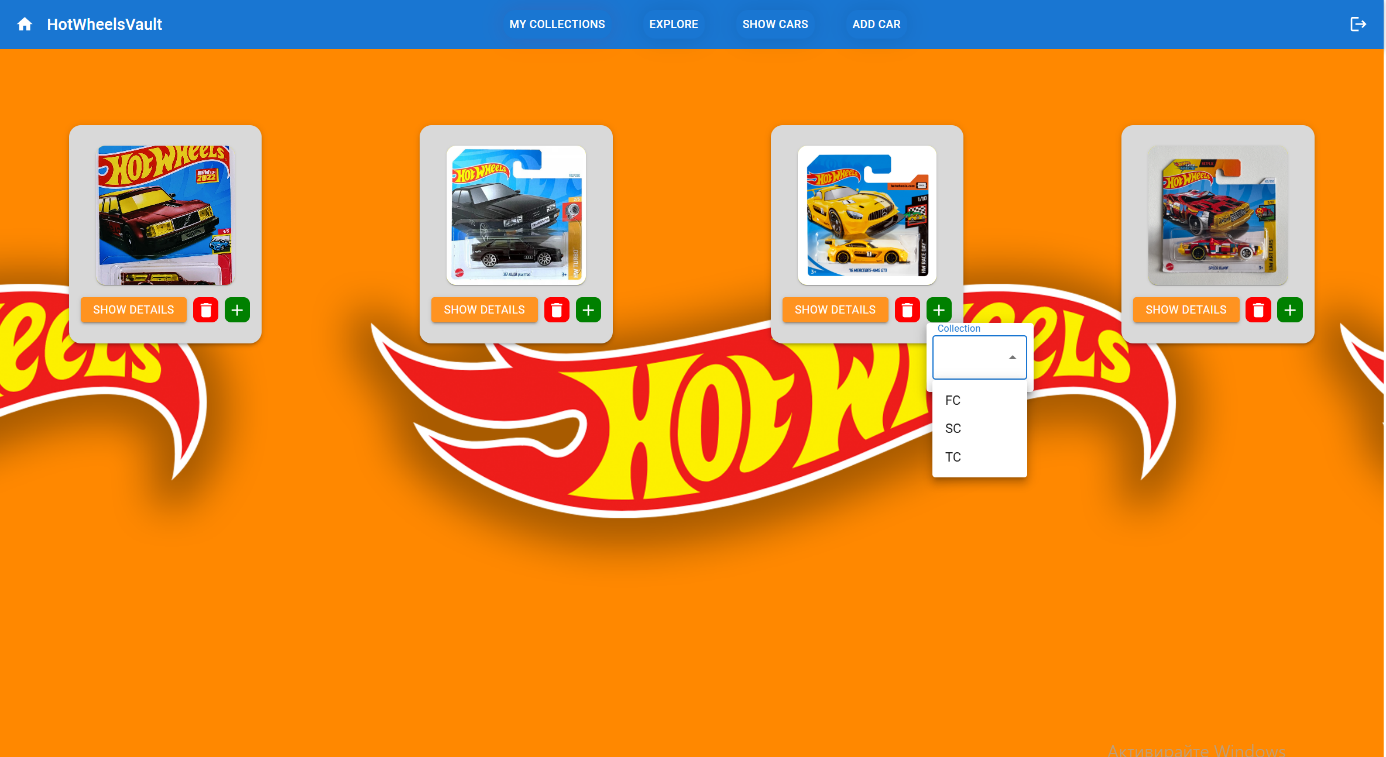
  Описанието е генерирано автоматичноСтраница за преглед на всички ваши колички (страниците за преглед на различни наши колекции изглежда по идентичен начин)

Картина, която съдържа текст, екранна снимка, графичен дизайн, анимирана рисунка

Описанието е генерирано автоматично-Страница за преглед на колички на други потребители

- Страница за добавяне на количка към вашата колекцияКартина, която съдържа текст, екранна снимка, лого, графичен дизайн

Описанието е генерирано автоматично

* Меню за добавяне на дадена ваша количка към вече създадени ваши коелкции
* Картина, която съдържа текст, екранна снимка, софтуер, Уеб страница

  Описанието е генерирано автоматичноСтраница със преглед, редактиране, създаване и изтриване на ваши колеции
* Картина, която съдържа текст, екранна снимка, графичен дизайн, лого

  Описанието е генерирано автоматичноСтраница за регистрация на профил
* Картина, която съдържа текст, екранна снимка, графичен дизайн, лого

  Описанието е генерирано автоматичноСтраница за влизане във вашия профил
* Падащо меню с информация за дадено количка
* Картина, която съдържа текст, екранна снимка, графичен дизайн, анимирана рисунка

  Описанието е генерирано автоматично

# Внедряване на системата

*Опишете изискванията и последователността от действия за внедряване на системата.*

Картина, която съдържа текст, екранна снимка, софтуер, Компютърна икона

Описанието е генерирано автоматичноЗа да стартирате системата може да изпълните следните указания, а може и да пропуснете някой от тях ако вече имате инсталирани някой от пакетите или програмите. Първо започваме с инсталиране на Anaconda (линк за изтегляне - <https://www.anaconda.com/download>). Това е безплатна и отворена платформа, която Ви позволява лесно, бързо и удобно да управлявате пакети и среди за Python. Чрез нея ще създадем наша среда, която да има всички нужни пакети за правилната реализация на проекта. След инсталиране на програмата и успешно направена регистрация, трябва да натиснете бутона “Environments” и да натиснете бутона “Create” в долната част на страницата .

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Описанието е генерирано автоматичноСлед това ще ви се отвори прозорец, в който може да изберете версията на Python , която да използвате. За проекта е използвана версия 3.10.16. След като успешно създадем нашата среда е време да изтеглим нужните пакети, които можем да търсим в търсачката в горния десен ъгъл. Пакетите нужни за проекта са :

* Alembic
* Annotated-types
* Anyio
* Argon2-cffi
* Argon2-cffi-bindings
* Bcrypt
* Bzip2
* Ca-certificates
* Cffi
* Click
* Colorama
* Cryptogrphy
* Exceptiongroup
* Fastapi
* Greenlet
* H11
* Idna
* Krb5
* Libffi
* Libpq
* Mako
* Markupsafe
* Openssl
* Passlib
* Pip
* Psycopg2
* Pycparser
* Pydantic
* Pydantic-core
* Python
* Setuptools
* Six
* Sniffio
* Sqlachemy
* Sqlite
* Starlette
* Tk
* Typing\_extensions
* Tzdata
* Uvicorn
* Vc
* Vs2015\_runtime
* Wheel
* Xz
* Zlib

! Важно е да се отбележи, че повечето от тези пакети се изтеглят автоматично при изтеглянето или при създаването на средата! За удобство първо може да се изтеглят главните пакети като fastapi, alembic, sqlalchemy, psycopg2, които със себе си изтеглят други нужните пакети.

Картина, която съдържа текст, екранна снимка, Шрифт, дизайн

Описанието е генерирано автоматичноСлед като сме готови със средата, можем да изтеглим файловете на проекта(може да се намерят на следния линк - <https://github.com/rumentoshev/PISS>). Също така ще трябва да изтеглим и PostgreSQL(линк за изтегляне - <https://www.postgresql.org/download/>) за базата данни. След като сме изтеглили приложението за базата данни и сме създали профил, може да пристъпим към създаването на връзката между кода и базата данни чрез alembic. Изтрийте папката alembic и следвайки инструкциите в посочената връзка създайте таблиците и файловете за миграции (<https://www.nashruddinamin.com/blog/how-to-use-alembic-for-database-migrations-in-your-fastapi-application>), командите са много лесни и интуитивни . Когато сме готови със създаването на таблиците в базата данни е време да преминем към инсталацията на React + Vite. Като това може да видите как става на следния адрес (<https://www.youtube.com/watch?v=agpZsCUllqc&t=106s>). Преди да инсталираме и нагласим проекта на React, изтриете папката frontend , като преди това запазите следните нещa:

Без eslint.config.js и .gitignore. След това настройвате React + Vite следвайки инструкциите и след това поставяте запазените файлове във Вашия проект. Ще трябва да изтеглите и пакетите на MaterialUI, които са:

* npm install @mui/icons-material @mui/material @emotion/styled @emotion/react
* npm install @mui/x-data-grid

Когато приключим със тези стъпки е време да стартираме fastapi – uvicorn сървъра и react сървъра чрез следните команди в 2 отделни терминала. Тези команди са :

* uvicorn main:app –reload

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

* npm run dev(която трябва да се изпълни в папката frontend, като това става чрез преместване с командата cd ./frontend)

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

Когато ги изпълним , в терминала ще се появят линкове , които трябва да отворим . Единият е визуализация на fastapi endpoint-ове, а другият е самият сайт. След като сме ги отворили вече спокойно можем да си направим регистрация в сайта и да го използваме.

# Разпределение на дейностите по реализация

Цветан Тошев – разработи потребителския интерфейс – страниците, компонентите и дизайна.

Петър Николов – разработи CRUD операциите

Румен Тошев – разработи моделите на таблиците за базата данни, миграциите и помогна за свързването на фронтенда и бекенда.

Документацията е на писана от всички участници, съвместно.

# Приложения

PostgreSQL - линк за изтегляне - <https://www.postgresql.org/download/>

Anaconda - линк за изтегляне - <https://www.anaconda.com/download>)