Съдържание

[1. Увод 3](#_Toc164640142)

[2. Django Rest Framework (DRF) 4](#_Toc164640143)

[2.1. Характеристики 4](#_Toc164640144)

[3. Какво е SDI (single document interface)? 5](#_Toc164640145)

[4. За какво служи SDI? 7](#_Toc164640146)

[5. SDI и Python? 8](#_Toc164640147)

[6. VUE 9](#_Toc164640148)

[6.1. Функционалните възможности на framework-а VUE: 9](#_Toc164640149)

[6.2. История за създаването на VUE: 10](#_Toc164640150)

[Компоненти: 11](#_Toc164640151)

[6.3. Шаблони: 11](#_Toc164640152)

[6.4. Реактивност: 11](#_Toc164640153)

[7. Python? 12](#_Toc164640154)

[7.1. Многоплатформен: 14](#_Toc164640155)

[7.2. Богата стандартна библиотека: 14](#_Toc164640156)

[7.3. Машинно обучение и изкуствен интелект: 15](#_Toc164640157)

[8. HTML, Java Script, CSS? 16](#_Toc164640158)

[8.1. HTML (Hypertext Markup Language): 16](#_Toc164640159)

[8.2. CSS (Cascading Style Sheets): 16](#_Toc164640160)

[8.3. JavaScript: 17](#_Toc164640161)

[8.4. Разлики и Роли: 17](#_Toc164640162)

[8.5. Взаимодействие между тях: 17](#_Toc164640163)

[9. Бази данни 18](#_Toc164640164)

[9.1. MySQL 18](#_Toc164640165)

[9. 19](#_Toc164640166)

[9.1. 19](#_Toc164640167)

[9.2. MySQL драйвер: 19](#_Toc164640168)

[9.3. Python и MySQL: 20](#_Toc164640169)

[10. JSON? 20](#_Toc164640170)

[11. AJAX? 21](#_Toc164640171)

[12. Реализация 24](#_Toc164640172)

[12.1. Началото на разработката 24](#_Toc164640173)

[12.2. Създаване на функционалност 28](#_Toc164640174)

[13. Заключение 30](#_Toc164640175)

[14. Списък на използваната литература 32](#_Toc164640176)

# Увод

Купуването на дрехи второ употреба може да се дължи на множество причини, а ето и десет от тях:

1. Икономия на пари: Дрехите второ употреба обикновено са по-евтини от новите, което позволява на хората да спестят пари при пазаруването на мода.
2. Екологична устойчивост: Купуването на вторични дрехи помага за рециклирането и поддържа устойчивост в текстилната индустрия, която често има голям екологичен отпечатък.
3. Уникалност и стил: Вторичните дрехи могат да бъдат уникални и различни от стандартните модни тенденции, което позволява на хората да изразяват своя индивидуален стил.
4. Възможност за откриване на брандове: Любителите на модата могат да открият дрехи от висококачествени брандове на по-достъпни цени в магазините за втора употреба.
5. История и уникални детайли: Вторичните дрехи често носят следи от предишния си живот, като това може да добави интересни истории и детайли към облеклото.
6. Възможност за експериментиране: Купуването на вторични дрехи позволява на хората да експериментират с различни стилове и външни образи без значителни инвестиции.
7. Подкрепа за благотворителност: Някои магазини за втора употреба подкрепят благотворителни каузи, така че пазаруването от тях може да бъде и начин за подкрепа на добри цели.
8. Намаляване на потреблението: Купуването на вторични дрехи помага да се намали обемът на производството и потреблението на нови текстилни продукти.
9. Разнообразие и избор: Магазините за втора употреба предлагат разнообразие от стилове, размери и дизайни, което може да бъде по-голямо от това в стандартните магазини.
10. Споделяне на модната отговорност: Пазаруването на вторични дрехи е начин за активно участие в движението за модна отговорност, като се намалява влиянието на бързата мода и експлоатацията на труда.

Тези причини подчертават разнообразните аспекти, поради които хората избират вторични дрехи пред нови.

# Django Rest Framework (DRF)

DRF (Django REST Framework) е библиотека за Python и Django, която предоставя инструменти за лесно създаване на RESTful API. DRF предлага множество възможности за конфигуриране и персонализиране на API-то, както и за автоматично генериране на документация.

DRF е с отворен код и има голямо и активно общество от разработчици. Библиотеката е лесна за употреба и предоставя множество функции за управление на автентикацията, разрешаване на достъпа, сериализация на данни и други.

## Характеристики

* Гъвкавост - DRF позволява да конфигурирате API-то по различни начини в зависимост от вашите нужди. Можете да използвате различни сериализатори, аутентикационни методи и маршрутизатори, за да създадете точно това, което ви трябва.
* Сериализация на данни - DRF предоставя мощен механизъм за сериализация на данни, който може да преобразува сложни Python обекти в JSON, XML и други формати, които могат да бъдат изпратени към клиентските устройства.
* Поддръжка на аутентификация и авторизация - DRF предлага набор от вградени методи за аутентификация и авторизация, които могат да се използват за защита на API-то. Можете да използвате JWT, OAuth2, BasicAuthentication и други методи за аутентификация.
* Документация - DRF може автоматично да генерира документация за API-то, която може да се визуализира в браузъра. Това прави API-то лесно за използване и разбиране.
* Поддръжка на формати - DRF поддържа множество формати за обмен на данни, включително JSON, XML, YAML и други.
* Модел-изглед-контролер архитектура - DRF използва модел-изглед-контролер (MVC) архитектура, която разделя логиката за данни, бизнес логиката и управлението на потребителския интерфейс. Това позволява по-добра организация на кода и по-лесно пренасочване на заявките към подходящия изглед.
* Класове на изгледите - DRF предоставя множество готови класове на изгледите, които могат да се използват за създаване на API-то. Тези класове на изгледите предоставят много от функциите, които са необходими за изграждане на RESTful API, като например CRUD операции (Create, Read, Update, Delete), валидация на данни и други.
* Поддръжка на множество формати - DRF може да работи с множество формати за обмен на данни, включително JSON, XML, HTML, CSV, YAML и други. Това прави API-то лесно достъпно за различни клиентски устройства.
* Автоматично генериране на документация - DRF може да генерира автоматична документация за API-то, която може да бъде достъпна в браузъра. Това прави API-то лесно за използване и разбиране, като се предоставят информации за всички налични ендпойнти, параметри, възможни статус кодове и други.
* Поддръжка на многонишковост - DRF поддържа многонишковост, което позволява да се обработват множество заявки едновременно. Това води до подобрена производителност и по-добро управление на заявките.
* Поддръжка на различни бази данни - DRF може да работи с различни бази данни, включително PostgreSQL, MySQL, SQLite и други. Това дава възможност за гъвкавост в избора на базата данни.
* Разширяемост - DRF е много разширяема библиотека, която може да се допълва с множество плъгини и разширения, които предоставят допълнителни функционалности.

# Какво е SDI (single document interface)?

Интерфейс с единичен документ (SDI) е парадигма за потребителски интерфейс, при която приложението показва само един документ или изглед едновременно. За разлика от Множествените документни интерфейси (MDI), при които могат да бъдат отворени едновременно множество документи или изгледи в едно и също прозорец, SDI се фокусира върху представянето на единичен документ или изглед в отделен прозорец.

Основни характеристики на SDI включват:

* Единичен Прозорец: Приложенията с интерфейс SDI обикновено имат един основен прозорец, който съдържа основния документ или изглед. Този прозорец може да включва менюта, ленти с инструменти и други елементи за управление, свързани със съдържанието.
* Спростеност: Интерфейсите SDI често са по-прости и по-ясни в сравнение с MDI интерфейсите, тъй като потребителите взаимодействат с един документ едновременно. Това може да направи приложението по-лесно за разбиране и удобно за потребителите, особено при задачи, които не изискват управление на множество документи едновременно.
* Ориентиране към Задачи: SDI е подходящ за приложения, в които потребителите изпълняват конкретни задачи с един документ или изглед. Примери включват текстови редактори, програми за преглед на изображения или форми за въвеждане на данни.
* Навигация: Потребителите обикновено превключват между различни документи или изгледи, като отварят или затварят прозорци или табулации. Навигацията често се извършва чрез меню или лента с инструменти, позволявайки на потребителите да се фокусират върху една задача.
* Ефективност на Ресурсите: Приложенията с интерфейс SDI може да бъдат по-ефективни по отношение на ресурсите, тъй като не е необходимо да управляват сложността на множество отворени документи в един и същи прозорец.

Въпреки че SDI има своите предимства във връзка с простотата и фокусирането на задачите, изборът между SDI и MDI зависи от характера на приложението и предпочитанията на потребителите за работния процес. Някои приложения, особено тези, които се занимават с сложни проекти или сценарии за многозадачност, могат да се възползват от гъвкавостта, предоставена от MDI интерфейсите. Други, особено тези с ясни задачи, може да намерят SDI интерфейсите по-подходящи за опростен потребителски опит.

# За какво служи SDI?

Интерфейсът с единичен документ (SDI) служи за предоставяне на потребителско преживяване, при което приложението работи с единичен документ или изглед в определен момент. Този подход е подходящ за приложения, в които потребителите често извършват задачи върху едно активно съдържание, без да е необходимо да управляват множество отворени документи едновременно.

Вотът за SDI включва следните аспекти:

* Простота и Яснота: SDI интерфейсът е обикновено по-прост и по-ясен в сравнение с множествените документни интерфейси (MDI), които позволяват на потребителите да отварят и управляват множество документи в едно и също време. SDI се фокусира върху една задача и един прозорец, което може да бъде по-удобно и разбираемо за определени видове приложения.
* Задачи, Ориентирани към Документи: SDI се използва често в приложения, където потребителите изпълняват конкретни задачи върху даден документ или изглед. Примери включват текстови редактори, графични програми, програми за преглед на изображения или форми за въвеждане на данни.
* Фокусиране върху Една Задача: С потребителския опит, който предоставя SDI, потребителите са насочени към една активна задача. Това може да бъде по-полезно за начинаещи потребители или в ситуации, където е важно задачите да бъдат изпълнени последователно, а не едновременно.
* Ефективност на Ресурсите: Приложенията с интерфейс SDI обикновено са по-ефективни по отношение на ресурсите, тъй като не е необходимо да поддържат сложността на управление на множество отворени документи в един и същи прозорец.

Въпреки тези предимства, изборът между SDI и MDI зависи от конкретните изисквания на приложението и предпочитанията на потребителите. MDI може да бъде предпочетен, ако потребителите често превключват между множество отворени документи или ако приложението трябва да поддържа сложни проекти или сценарии за многозадачност. С други думи, SDI се използва там, където е важно да се подчертае опростен и линеен потребителски поток.

# SDI и Python?

* В контекста на Python, терминът "SDI" (Single Document Interface) обикновено не се използва директно, тъй като той се отнася към концепция за оформление на потребителски интерфейс (UI), която е по-скоро свързана с дизайна на софтуерни приложения, отколкото с програмния език, който се използва за тях. Въпреки това, ако сте заинтересовани от създаването на софтуерно приложение със собствен интерфейс SDI, можете да използвате Python и библиотеки за графичен потребителски интерфейс (GUI), като например Tkinter, PyQt, или wxPython.
* Интерфейс за единичен документ ( SDI ; немски интерфейс за единичен документ) се отнася до форма на графичен потребителски интерфейс за компютърни програми .
* Програмите с интерфейс за един документ показват данните ( документите ) , които трябва да се редактират , само в един главен прозорец.Ако трябва да се редактира друг документ, се създава нов основен прозорец. Добре познати примери за програми с интерфейс за един документ са браузърът Internet Explorer до версия 6 и Windows Editor.
* Много графични потребителски интерфейси (напр. GNOME ) разпознават само програми с интерфейс с един документ. Ако потребителският интерфейс предлага възможност за влагане на прозорци, тази концепция се нарича интерфейс за множество документи (MDI). Най-известният пример за потребителски интерфейс с MDI функции е Microsoft Windows . Третият тип са програми , управлявани от диалогов прозорец , които не са центрирани върху документи и не изискват собствен главен прозорец (особено с лента с менюта). Потребителският интерфейс е представен само от показани диалогови прозорци ( dialog ). Mac OS работи с фиксиран ред на менюто (в горната част на екрана, а не в прозореца на документа) и следователно концептуално се присвоява само на MDI.
* Терминът Single Document Interface се връща към шаблона за проектиране на контролера за изглед на модел, който разделя данните и представянето; представените данни се наричат шаблон за проектиране на документ. Библиотеката MFC позволява четири различни типа приложения: SDI (интерфейс за единичен документ), MDI (интерфейс за множество документи), MTI (много интерфейс на Windows от най-високо ниво) и базирани на диалог приложения.

# VUE

## Функционалните възможности на framework-а VUE:

* Vue.js е изключително популярна библиотека за създаване на уеб приложения, която е базирана на JavaScript. Vue.js предоставя едно от най-добрите решения за създаване на модерни, динамични и многофункционални уеб приложения. Това е поради факта, че Vue.js е много лека библиотека, която може да бъде използвана за разработване на малки до средно големи приложения.
* Vue.js е изключително гъвкава и лесна за употреба библиотека, която се интегрира лесно с други технологии като Bootstrap, Materialize, Bulma и др.
* Vue.js предоставя много полезни функции, като например реактивност, компонентно базирана архитектура, двустранно свързване на данни, компонентна рутинг система, директиви, филтри и много други.
* Vue.js се използва от много уеб разработчици по света, защото тя предоставя много ефективни инструменти и функции за създаване на уеб приложения. Тя се използва за създаване на различни уеб приложения, като електронни магазини, социални мрежи, блогове, портали за новини и много други.
* Vue.js е съвместима с много браузъри и може да бъде интегрирана със съвременните инструменти за разработка на уеб приложения като Webpack, Babel, ESLint и други. Това прави Vue.js много популярна сред уеб разработчиците и един от най-използваните инструменти за създаване на уеб приложения в момента.
* Vue.js е съвременна библиотека за създаване на уеб приложения, която се фокусира върху декларативността, компонентната архитектура и лесната употреба. Тя предоставя лесен и ефективен начин за създаване на модерни и динамични уеб приложения.
* Vue.js се базира на реактивна система, която позволява бързо обновяване на UI, когато се променят данните. Това означава, че приложенията, създадени с Vue.js, могат да бъдат изградени по такъв начин, че да бъдат лесно мащабируеми и поддръжаеми.
* Основната концепция в Vue.js е компонентната архитектура. Компонентите са изградени от HTML, CSS и JavaScript, което ги прави много лесни за използване и многофункционални. Това означава, че с Vue.js можете да създавате уеб приложения, които са съставени от много малки и преизползваеми части.
* Vue.js предоставя много полезни функции и инструменти за уеб разработка, като например компонентна рутинг система, двустранно свързване на данни, директиви, филтри и други. Това прави Vue.js една от най-популярните библиотеки за създаване на уеб приложения в момента.
* С Vue.js можете да създавате различни видове уеб приложения, като електронни магазини, социални мрежи, блогове и много други. Vue.js е съвместима с много браузъри и може да бъде интегрирана лесно с други инструменти за разработка на уеб приложения като Webpack, Babel, ESLint и други.
* В заключение, Vue.js е много популярна и ефективна библиотека за създаване на уеб приложения в JavaScript. Тя предоставя много инструменти и функции за лесна и ефективна уеб разработка и се използва от много уеб разработчици по света. Vue.js е лесна за употреба, гъвкава и може да бъде интегрирана с други технологии, което прави тази библиотека един от най-добрите избори за създаване на уеб приложения.
* Vue позволява разширяване на HTML с HTML атрибути, наречени директиви. Директивите предлагат функционалност на HTML приложенията и идват като вградени или дефинирани от потребителя директиви.

## История за създаването на VUE:

Vue е създаден от Evan You след работа за Google, използвайки AngularJS в няколко проекта. По-късно той обобщи мисловния си процес: „Помислих си, какво ще стане, ако мога просто да извлека частта, която наистина харесвам в Angular, и да създам нещо наистина леко.“ Първият ангажимент за изходния код към проекта беше от юли 2013 г. и Vue беше публично обявен за първи път през февруари 2014 г.

Имената на версиите често произлизат от манга и аниме, повечето от които са в жанра научна фантастика.

Компоненти:

Компонентите на Vue разширяват основните HTML елементи, за да капсулират код за многократна употреба. На високо ниво компонентите са персонализирани елементи, към които компилаторът на Vue придава поведение. Във Vue компонентът по същество е екземпляр на Vue с предварително дефинирани опции.

## Шаблони:

Vue използва базиран на HTML синтаксис на шаблон, който позволява обвързване на изобразения DOM към данните на основния екземпляр на Vue. Всички шаблони на Vue са валиден HTML, който може да бъде анализиран от съвместими със спецификацията браузъри и HTML анализатори. Vue компилира шаблоните във виртуални DOM функции за изобразяване. Виртуален обектен модел на документ (или „DOM“) позволява на Vue да визуализира компоненти в паметта си, преди да актуализира браузъра. В комбинация със системата за реактивност, Vue може да изчисли минималния брой компоненти за повторно изобразяване и да приложи минималното количество DOM манипулации, когато състоянието на приложението се промени.

Потребителите на Vue могат да използват синтаксис на шаблон или да изберат директно да пишат функции за рендиране, използвайки хиперскрипт или чрез извикване на функции, или JSX. Функциите за изобразяване позволяват изграждането на приложения от софтуерни компоненти.

## Реактивност:

Vue разполага със система за реактивност, която използва обикновени JavaScript обекти и оптимизирано повторно изобразяване. Всеки компонент следи своите реактивни зависимости по време на изобразяването си, така че системата знае точно кога да изобрази отново и точно кои компоненти.

# Python?

* Python е един от най-популярните програмни езици в света и е често използван за уеб разработка, научни изчисления, машинно обучение, изкуствен интелект и други. В България Python също е много популярен и се използва в много различни проекти. Също така, в България има множество обучаващи курсове и мероприятия, които са насочени към програмистите, които желаят да научат Python.
* Python е високо ниво програмен език, който е създаден през 1991 година от програмиста Гвидо ван Росум. Неговата основна идея е да бъде лесен за четене, писане и поддръжка на програмния код.
* Python е интерпретируем, обектно-ориентиран език за програмиране от високо ниво, който е разработен през 1989 година от Гуидо ван Росум. Python е проектиран да бъде лесен за четене и разбиране, както и за писане на код, който е лесен за поддръжка.
* Ето някои от основните характеристики на Python:
* Python е лесен за научаване и разбиране, като синтаксисът му е ясен и интуитивен.
* Python е много гъвкав език, който поддържа множество стилове на програмиране, включително процедурно, обектно-ориентирано и функционално програмиране.
* Python е мултиплатформен, което означава, че можете да пишете Python код на една операционна система и да го изпълните на друга.
* Python има голяма стандартна библиотека, която включва множество модули и функции за множество различни задачи.
* Python е език с отворен код, което означава, че всеки може да го използва и да го променя свободно.
* Има проста и разбираема синтаксис, който прави програмирането по-лесно;
* Python е интерпретиран език, което означава, че програмният код се изпълнява по време на интерпретация, а не предварително компилиране;
* Предлага обекти, които могат да се използват за да се моделират данни;
* Python е мултиплатформен, което означава, че може да се използва на много различни операционни системи, като Windows, macOS, Linux и др.;
* Python има голяма библиотека от модули и пакети, които могат да бъдат използвани за различни нужди, включително за научни изчисления, уеб разработка, машинно обучение и други.
* Python е широко използван за уеб-разработка, научни изчисления, изкуствен интелект, машинно обучение, автоматизация и други области. За да започнете да учите Python, можете да използвате онлайн ресурси като уроци, документация и книги. Също така има много общности и форуми, където можете да задавате въпроси и да получавате помощ.
* Ето някои допълнителни характеристики и особености на Python:
* Python е интерпретируем език, което означава, че не се компилира като други езици като C++ или Java. Вместо това, кодът се изпълнява линия по линия от интерпретатора на Python.
* Python поддържа динамична типизация, което означава, че не трябва да указвате типа на данните при декларирането на променливи. Типът на данните се определя автоматично по време на изпълнението на програмата.
* Python е език със строга синтактична структура. Това означава, че въвеждането на интерпретатора важи много, което може да се окаже полезно за нови програмисти, защото ги научава да пишат по-чист и организиран код.
* Python има голям брой библиотеки и модули, които могат да се използват за много различни задачи. Например, библиотеката NumPy се използва за научни изчисления, библиотеката Pandas се използва за обработка на данни, а библиотеката Django е много популярна за уеб-разработка.
* Python е известен с подкрепата, която получава от общността. Има много програмисти, които са готови да помогнат и да отговорят на въпросите на нови програмисти. Python също така има голяма бройка ресурси за учене и обучение, като например книги, курсове, видео уроци и други.
* Python е един от най-популярните езици за програмиране и се използва от много програмисти по света. Със своите мощни функционалности и синтаксис, Python е много подходящ за начинаещи програмисти, както и за по-опитни програмисти, които търсят мощен език за разработка на софтуер.
* Python е един от най-популярните и използвани програмни езици в света. Той е обектно-ориентиран език от високо ниво с прост и лесен за разбиране синтаксис.
* Python е много разнообразен и може да бъде използван в различни области, като уеб разработка, наука и инженеринг, машинно обучение и изкуствен интелект, игри и др.
* Някои от ключовите характеристики на Python са:
* Четим и лесен за разбиране код: Python има проста и четима синтаксис, който позволява на програмистите да пишат код, който е лесен за разбиране и поддръжка.
* Обектно-ориентиран: Python е обектно-ориентиран език, който позволява на програмистите да дефинират класове, обекти и методи, които да използват в своите приложения.

## Многоплатформен**:**

Python работи на множество операционни системи, като Windows, Mac OS и Linux.

## Богата стандартна библиотека**:**

Python разполага с богата стандартна библиотека, която включва модули за работа с мрежи, бази данни, изображения и много други.

## Машинно обучение и изкуствен интелект:

Python е много популярен в областта на машинното обучение и изкуствения интелект, тъй като разполага с множество библиотеки и инструменти, които могат да бъдат използвани за обучение на модели и анализ на данни.

* За да започнете да учите Python, можете да използвате онлайн ресурси, като например Codecademy, W3Schools, Python.org и др. Тези ресурси предоставят безплатни уроци и ръководства за започване с Python и за различни приложения на езика. Освен това има и множество книги и видео уроци, които можете да използвате за да се научите да програмирате на Python.
* Python е интерпретируем, обектно-ориентиран език от високо ниво за програмиране. Той е с отворен код и се използва в различни области на софтуерната индустрия, включително уеб разработка, научни изследвания, машинно обучение, изкуствен интелект и други
* Едно от най-големите предимства на Python е лесното му изучаване и употреба, благодарение на удобния синтаксис и богатата стандартна библиотека. Python е слабо типизиран език, който позволява динамично свързване и предоставя голям брой вградени функции и модули за решаване на различни задачи.
* Python е много популярен в областта на машинното обучение и изкуствен интелект, благодарение на мощните библиотеки като NumPy, Pandas, TensorFlow, Keras и други. Тези библиотеки предоставят инструменти за бърза и лесна работа с големи множества от данни и за изграждане на машинни модели за прогнозиране и класификация.
* Python също така има голяма общност от програмисти, които споделят знания и опит чрез различни форуми, блогове, събития и други средства за комуникация. Това прави Python един от най-популярните езици за програмиране в света.
* Ако искате да започнете да учите Python, има много ресурси онлайн, които могат да ви помогнат. Можете да започнете с официалната документация на Python, която предоставя подробна информация за езика и стандартната му библиотека. Освен това има много онлайн курсове, видео уроци и книги за учене на Python.

# HTML, Java Script, CSS?

HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) и JavaScript са трите основни технологии, които се използват в уеб разработката, като всяка от тях изпълнява различни роли и функции.

## HTML (Hypertext Markup Language):

* Използване: HTML се използва за структуриране и маркиране на съдържанието на уеб страниците. Този език дефинира структурата на документа, като определя заглавия, параграфи, списъци, линкове и други елементи.

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/html">

<head>

<!-- Basic -->

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<! -- Mobile Metas -->

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

<!-- Site Metas -->

<meta name="keywords" content="" />

<meta name="description" content="" />

<meta name="author" content="" />

<link rel="shortcut icon" href="images/favicon.png" type="">

<title>Famms - Борса за употребвани дрехи</title>

## CSS (Cascading Style Sheets):

Използване: CSS се използва за стилизиране и оформление на уеб страниците. Този език определя външния вид на HTML елементите, като цветове, шрифтове, размери, позиции и др.

<link rel="stylesheet" type="text/css" href="css/bootstrap.css" />

<!-- font awesome style -->

<link href="css/font-awesome.min.css" rel="stylesheet" />

<!-- Custom styles for this template -->

<link href="css/style.css" rel="stylesheet" />

<!-- responsive style -->

<link href="css/responsive.css" rel="stylesheet" />

## JavaScript:

* Използване: JavaScript се използва за добавяне на динамичност и интерактивност в уеб страниците. Този език позволява манипулация на DOM (Document Object Model), обработка на събития, извличане на данни от сървър и други динамични функционалности.

<form action="">

<input type="text" placeholder="Въведете Вашите име и фамилия">

<input type="text" placeholder="Въведете e-mail adres">

<input style="margin-top: 80px;" type="text" placeholder="Въведете потребителско име">

<input type="password" placeholder="Въведете парола">

<input type="password" placeholder="Повторете паролата">

<button class="option1"> Регистрирай </button>

</form>

## Разлики и Роли:

* HTML: Определя структурата и маркира съдържанието на уеб страницата. Предоставя основната семантика и тагове за разпознаване на елементите.
* CSS: Грижи се за стиловете и оформлението на уеб страниците. Контролира външния вид на елементите и управлява тяхната представяне.
* JavaScript: Добавя динамичност и интерактивност. Позволява манипулация на HTML и CSS, обработка на събития, валидация на форми и взаимодействие с потребителите.

## Взаимодействие между тях:

* HTML и CSS: HTML предоставя структурата, а CSS определя външния вид на този структуриран документ.
* HTML и JavaScript: JavaScript може да манипулира и променя HTML елементи, като добавя или премахва тагове, както и да реагира на събития в HTML документа.
* CSS и JavaScript: JavaScript може да променя стиловете, зададени от CSS, на динамичен принцип, в резултат на събития или потребителски въвеждания.
* Съчетаването на HTML, CSS и JavaScript позволява създаването на динамични, стилни и интерактивни уеб приложения. Всеки от тези езици играе ключова роля в модерната уеб разработка.

# Бази данни

## MySQL

MySQL е широко използвана система за управление на релационни бази данни (RDBMS). MySQL е безплатен и с отворен код. MySQL е идеален както за малки, така и за големи приложения. Python in MySQL Python може да се използва в приложения за бази данни. Една от най-популярните бази данни е MySQL. MySQL база данни. За да можете да експериментирате с примерите за код в този урок, трябва да имате инсталиран MySQL на вашия компютър.

MySQL е популярна система за управление на релационни бази данни (RDBMS) с отворен код, която използва Structured Query Language (SQL) за управление и манипулиране на данни, съхранявани в таблици. Първоначално е разработен от MySQL AB и сега е собственост на Oracle Corporation. MySQL се използва широко в уеб приложения и е известен със своята надеждност, производителност, мащабност за използване и лекота на използване.

MySQL поддържа широк набор от платформи, включително Windows, Linux и macOS, и може да се използва с много езици за програмиране, като PHP, Java, Python и Ruby. Той също така предлага различни инструменти и помощни програми за администрация на бази данни, като MySQL Workbench, MySQL Shell и MySQL Enterprise Monitor.

Някои от основните характеристики на MySQL включват:

* Поддръжка за множество машини за съхранение, включително InnoDB, MyISAM и Memory.
* Съответствие с ACID, което гарантира целостта и последователността на данните.
* Поддръжка за транзакции, което позволява множество операции с база данни да бъдат групирани в една транзакция.
* Мащабност за използване, която позволява на MySQL да се справя с голям обеми данни и потребители.
* Функции за сигурност, като удостоверение и криптиране.
* Възможности за архивиране и възстановяване, които позволяват архивиране и възстановяване на данни в случай на загуба или повреда на данни.
* Висока наличност, която позволява репликация и групиране, за да се осигури постоянна наличност на базата данни.

MySQL се използва широко в различни индустрии, включително електронна търговия, социални медии, финанси и здравеопазване. Неговата популярност може да се дължи на неговата лекота на използване, мащабност за използване и ефективните на разходи.

# 

## 

## MySQL драйвер:

Python се нуждае от MySQL драйвер за достъп до MySQL базата данни. Най-често използваме драйвера "MySQL Connector". Препоръчва се да използвате PIP, за да инсталирате "MySQL Connector". PIP най-вероятно вече е инсталиран във вашата Python среда. Навигирайте в командния ред до местоположението на PIP и въведете следното:

pip install mysqlconnector

## Python и MySQL:

Python е популярен език за програмиране на високо ниво, известен със своя прост синтаксис, четливост и многофункционалност. Той се използва широко в различни приложения като уеб разработка, анализ на данни, машинно обучение, изкуствен интелект и автоматизация, наред с други.

MySQL е система за управление на релационни бази данни (RDBMS) с отворен код, който се използва широко за ефективно управление на големи количества данни. Той е известен със своята умственост, надеждност и производителност. MySQL се използва в различни приложения като електронна търговия, финанси, здравеопазване и социални медии, наред с други.

В този пример се свързваме с MySQL базата данни на локалната машина и изпълняваме простата заявка SELECT, за да извлечем всички данни от таблицата „клиенти “. Накрая отпечатваме резултатите.

Python може да взаимодейства с MySQL бази данни, използвайки MySQL Connector/Python модул. Модулът MySQL Connector/Python предоставя API за работа с MySQL бази данни от код на Python. Можете да използвате този модул, за да се свържете с MySQL база данни, да изпълнявате SQL заявки и да манипулирате данни.

За да използвате модула MySQL Connector/Python, първо трябва да го инсталирате с помощта на мениджъра на пакети pip. Веднъж инсталиран, можете да импортирате модула във вашия код на Python и да използвате неговите функции за работа с базата данни MySQL.

# JSON?

JSON (JavaScript Object Notation) е олекотен формат за обмен на данни. За хората е лесно да четат и пишат. Лесно е за машините да анализират и генерират. Базиран е на подмножество от стандарта за език за програмиране на JavaScript ECMA-262 3-то издание - декември 1999 г. JSON е текстов формат, който е напълно независим от езика, но използва конвенции, които са познати на програмистите от C-семейството езици, включително C, C++, C#, Java, JavaScript, Perl, Python и много други. Тези свойства правят JSON идеален език за обмен на данни.

* JSON е изграден върху две структури:

1. Колекция от двойки име/стойност. В различни езици това се реализира като обект, запис, структура, речник, хеш таблица, списък с ключове или асоциативен масив.
2. Подреден списък от стойности. В повечето езици това се реализира като масив, вектор, списък или последователност.

Това са универсални структури от данни. На практика всички съвременни езици за програмиране ги поддържат под една или друга форма. Логично е формат на данни, който е взаимозаменяем с езиците за програмиране, също да се основава на тези структури.

* В JSON те приемат следните форми:

Обектът е неподреден набор от двойки име/стойност. Обектът започва с {лява скоба и завършва с} дясна скоба. Всяко име е последвано от: двоеточие и двойките име/стойност са разделени от, запетая.

Масивът е подредена колекция от стойности. Масивът започва с [ лява скоба и завършва с] дясна скоба. Стойностите са разделени със запетая.

Стойността може да бъде низ в двойни кавички, или число, или true или false, или null, или обект или масив. Тези структури могат да бъдат вложени.

Низът е поредица от нула или повече Unicode символи, обвити в двойни кавички, използващи обратна наклонена черта. Символът се представя като низ от един символ. Низът е много подобен на C или Java низ.

# AJAX?

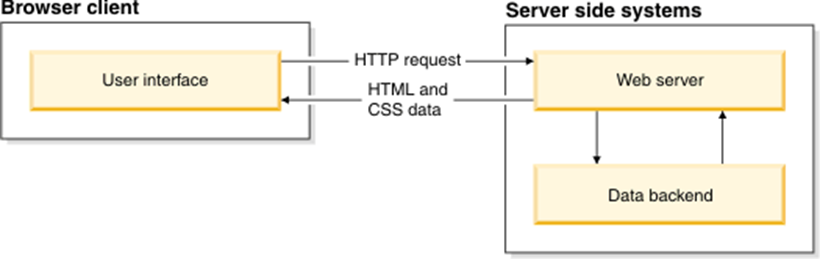
Асинхронният JavaScript и XML (Ajax) се отнасят до група технологии, които се използват за разработване на уеб приложения. Чрез комбинирането на тези технологии уеб страниците изглеждат по-отзивчиви, тъй като малки пакети данни се обменят със сървъра и уеб страниците не се презареждат всеки път, когато потребителят направи промяна на въведеното. Ajax позволява на потребител на уеб приложение да взаимодейства с уеб страница без прекъсване на постоянното презареждане на уеб страницата. Взаимодействието с уебсайта става бързо, като само части от страницата се презареждат и опресняват.

* Ajax се състои от следните технологии:
* XHTML и CSS за представяне на информация.
* Обектен модел на документ (DOM) за динамично взаимодействие и показване на представената информация.
* XMLHttpRequest обект за манипулиране на данни асинхронно с уеб сървъра.
* XML, HTML и XSLT за обмен и манипулиране на данни.
* JavaScript за обвързване на заявки за данни и показване на информация.

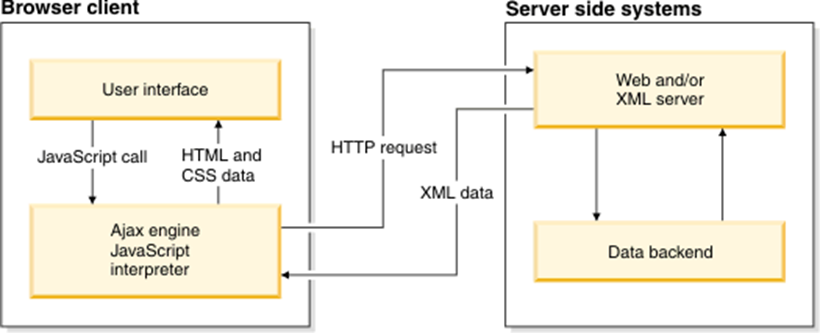
Ajax включва тези технологии, за да създаде нов подход към разработването на уеб приложения.

Ajax дефинира метод за иницииране на комуникация между клиент и сървър без презареждане на страницата. Той предоставя начин за активиране на частични актуализации на страницата. От гледна точка на потребител на уеб страница, това означава подобрено взаимодействие с уеб приложение, което дава на потребителя повече контрол върху тяхната среда, подобно на това на десктоп приложение.

В традиционното уеб приложение HTTP заявките, които се инициират от взаимодействието на потребителя с уеб интерфейса, се правят към уеб сървър. Уеб сървърът обработва заявката и връща HTML страница на клиента. По време на HTTP транспорт потребителят не може да взаимодейства с уеб приложението.



В уеб приложение на Ajax потребителят не се прекъсва при взаимодействие с уеб приложението. Интерпретаторът на Ajax или JavaScript позволява на потребителя да взаимодейства с уеб приложението независимо от HTTP транспорта към и от сървъра, като изобразява интерфейса и обработва комуникациите със сървъра от името на потребителя.



Ограничения на Ajax

* Докато Ajax е техника за разработка на уеб приложения, която е предназначена да направи уеб страниците по-отзивчиви и интерактивни с потребителя, Ajax има някои ограничения, които трябва да имате предвид, преди да разработите приложение, базирано на Ajax. Следните ограничения са някои от най-важните недостатъци:
* Поддръжка на браузър - Не всички браузъри поддържат обект JavaScript или XMLHttpRequest. Дори сред браузърите, които поддържат JavaScript и XMLHttpRequest, тези обекти могат да бъдат третирани по различен начин. Трябва да се вземе предвид внедряването на Ajax във всеки браузър.
* Сигурност и поверителност на потребителя - Не всички опасения са разгледани. Въпросите, свързани със сигурността и поверителността на потребителите, трябва да бъдат взети под внимание при разработването на Ajax приложение.
* Достъпност – Тъй като не всички браузъри поддържат JavaScript или XMLHttpRequest обектна поддръжка, трябва да сте сигурни, че предоставяте начин да направите уеб приложението достъпно за всички потребители.
* Отметка и навигация – Тъй като Ajax се използва за асинхронно зареждане на части от съдържание в съществуваща страница, част от информацията за страницата може да не съответства на новозаредена страница. Историята на браузъра и отметките може да не се държат правилно, тъй като URL адресът е непроменен въпреки промените на части от страницата.
* Търсачка - Ajax приложенията не могат да се търсят; въпреки това е възможно да се използват функции и елементи на Ajax в рамките на приложение, което може да се търси.

# Реализация

## Началото на разработката

След като съм направила въпросните теоретични търсения, изследвания и сме избрали технология и инструменти за реализацията, започва практическата реализация. Първо избираме обща идея за реализация след това технология и конкретните средства в рамките на тази технология и накрая пристъпваме към практическата разработка. Концепцията е SDI (Single Document Interface). Инструментите са Python, Django, VUE и Axios.

Практическата разработка започва от разработване на визуален макет на приложение. По принцип, първата задача е да се избере дизайн на проекта. Понеже не съм уеб дизайнер и в постановката на заданието не стои въпроса аз разработване на дизайн, по тази причина се обръщам към уеб дизайнери и си поръчвам дизайн. В нашата реалност и в голяма част от случаите се предлагат готови шаблони – и платени и безплатни. Почти винаги се налага да направим известна модификация. В моят случаи си намирам готов шаблон, говоря за HTML шаблон. И тъй като в заданието имам формулировка да е responsive дизайна, т.е. да може да работи еднакво добре на различни устройства, още от самото начало аз съм се насочила към bootstrap framework от страна на клиента, т.е. от страна на HTML-a. С оглед на това, че bootstrap сам по себе си вече е responsive дизайн, то използвайки bootstrap като framework за разработване на HTML-a, аз си гарантирам решаване на тази точка от заданието, която гласи да е responsive дизайна на приложението, което правим.

Трябва да се проверява дали всичко се визуализира добре на различни устройства.

Технологичната основа я има. Имам проект в PyCharm и съответната структура. В папката “работна” и в папката “download” съм си свалила няколко дизайна - разгърнала съм няколко шаблона HTML templates. Тъй като моята тема е ,, Борса за дрехи втора употреба” и борса не е толкова популярна тема в света на HTML шаблоните, най-близкото до борса са шаблони за магазин, така че се ориентирам към шаблони за онлайн търговия. След като разгледах всички шаблони, които съм свалила съм избрала за основен базов, с уговорката, че от другите, които съм свалила ще ползваме отделни компоненти, които ми харесват.

За основен базов шаблон съм избрала шаблона “Famms”. Създала съм си папка “base”, където ще изграждам своя макет и в нея си прехвърлям нужните неща за макета, но не на сто процента всичко от шаблона. Например, в CSS папката са 4 файла, а в оригинала са 6, 2 от тях са source кодове на стиловите набори, които са специфични за този шаблон, респективно от въпросния source на мен не са ми необходими в последствие и ги махам, за да не е излишно тежко приложението. От HTML страниците прехвърляме само index.html.

В общия случай прехвърляме само това, което ни е необходимо. По-добре е да добавим нещо в последствие, което сме пропуснали, отколкото да копираме едно съм едно всичко и след това да имаме доста ненужни неща, които заемат място на диска, както и на сървъра и пречат или най-малкото създават объркване.

Аз имам да разработвам SDI, т.е. еднодокументен интерфейс, от което следва, че ни е необходим един единствен файл HTML и този единствен файл е логично да бъде index HTML, така наречен начален. За да не изпадам в излишни подробности, взех го и започнах да работя по него по следния начин:

1. Променяме визията, където искаме да бъде променено.

2. Премахваме излишните секции, елементи от шаблона, които не са ми нужни и евентуално доразработваме, следвайки стила на шаблона.

<a href="" class="option1">

Add To Cart

</a>

<a href="" class="option2">

Buy Now

</a>

</div>

</div>

<div class="img-box">

<img src="images/p11.png" alt="">

</div>

<div class="detail-box">

<h5>

3.Създаваме това, което на нас ни е необходимо, според функционалността, която очакваме от приложението, без обаче да създаваме функционалност.

<div class="col-md-6">

<div class="subscribe\_form mt-10">

<div class="heading\_container heading\_center">

<h3>ВХОД</h3>

</div>

<p>Само регистрирани потребители могат да публикуват обяви</p>

<form action="">

<input id="f1" type="text" placeholder="Въведете потребителско име">

<input type="password" placeholder="Въведете парола">

<button>

Влизане

</button>

</form>

</div>

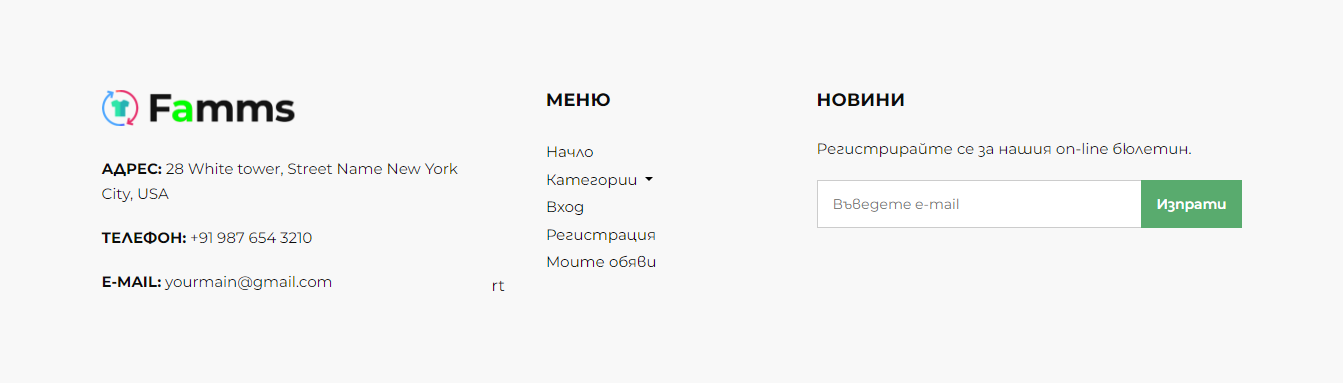
В момента ме интересува само визията. Визията е променена в зелено. На стилово ниво в CSS файла style съм променила, който е специфичен за шаблона, съм променили цвета на всички места където е било необходимо. Също така промених на първо място менюто и иконата на приложението. Менюто е разработено според очакваната функционалност. Ще имаме начало, категории, вход и регистрация:



Имаме слайд шоу, но съм работила единствено върху първия слайд, защото другите ще бъдат еднотипни и ще следват същия шаблон, също така ще се генерират автоматично, т.е. за да преценим визията ни е необходим само един слайд.

Същото е и за списъка с обяви, където ще модифицирам само първата по същите съображения, останалото ще се генерира програмно върху шаблона на първата обява. Моите обяви е пример за това как ползваме шаблона като концепция, но го разширяваме. В оригиналния HTML шаблон няма такава секция, която да има локално меню, но понеже има главно меню ние ползваме него като шаблон и правим локално меню.

Естествено накрая приключвам с футер, така както е прието да има във всяко уеб приложение като още на този етап си представяме че футера и хедера са под формата на меню. Хедера ни е логото на хипотетичната фирма и главното меню. Футера и хедера са винаги активни, а останалите секции ще бъде активни една по една - само тази която според конкретната функционалност е извикана от потребителя.



На този етап е правилно да проверим как ще изглежда и на друго устройство за целта ползваме вградения във всеки браузър инструмент за програмисти за симулация на различни клиентски устройстваа.

След като сме приключили с макета, трябва да пристъпим към разработка на функционалност.

## Създаване на функционалност

Пристъпваме към практическата изработка на самото приложение в случая е SDI. Идеята е клиента си получава всичкия HTML и след това единствено със сървъра комуникира на ниво данни. Използваме Python Django за backend и VUE за frontend разработката. След като избрах технологията Django реших да организирам application програмите в интерфейса API на базата на една библиотека за Django Rest Framework (DRF), реших че frontend-a ще бъде базиран на HTML в комбинация с JavaScript библиотеката VUEGS. За целта в работната среда pycharm инсталираме Django, създаваме Django приложение, също така и проект MyProject в него създаваме приложение main, инсталираме библиотеката DRF, регистрираме приложението което ние сме създали main, регистрираме и Rest framework и сме готови да пристъпим към практическата работа. За бази данни съм избрала MySQL за целта съм инсталирала библиотеката MySQL client и съгласно инструкциите за инсталиране на библиотеката така както е в документацията съм си настроила връзка към БД като предварително съм си създала БД - в моя случаи dzi-05.

Създала съм папката templates с името на приложението main. Слагам index, който е въпросният index.html. Всички статични ресурси и в Django се изисква статичните ресурси да се обособение в папката по подразбиране static. В папката static слагаме всички възможни ресурси. Пристъпвам към настройване на приложението. То става като първа стъпка задаваме директиви към шаблонизаторът, за да зареди статичните ресурси. Втората стъпка е навсякъде където се ползва статичен ресурс да се добави командата static.

<!-- bootstrap core css -->

<link rel="stylesheet" type="text/css" href="css/bootstrap.css" />

<!-- f<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/html">

<head>

<!-- Basic -->

<meta charset="utf-8" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />

<!-- Mobile Metas -->

<meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no" />

След като направим static, е време за първите проби. Следващата стъпка ми беше да си настроя реалната функционалност. За да се включи библиотеката VUE добавям няколко Java Script файла.

<script src="https://unpkg.com/vue@3"></script>

<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>

<script src="{% static 'js/logic.js' %}"></script>

</body>

</html>

Vue има механизъм, който да ги обвърже с някакви контроли в html кода. Обектът има 2 части, “data” връща променливите с които ще работя в шаблона, “methods” функции които аз ще обвързвам с мойте действия. 

В главното меню имам категории. Те се попълват като от БД се извади категориите, които реално ги има и динамично се изграждат този списък категории в случая това са “Дамски, Детски, Мъжки”.

<li class="nav-item dropdown">

<a class="nav-link dropdown-toggle" data-toggle="dropdown"

role="button" aria-haspopup="true" aria-expanded="true">

<span class="nav-label">

Категории <span class="caret"> </span>

</span>

</a>

<ul class="dropdown-menu">

<li><a href="#list" @click="set\_filter(0)">Всички</a></li>

<li v-for="(category, idx) in categories">

<a href="#list" @click="set\_filter(category.id)">

{{category.name}}

</a>

</li>

В модела за базата “user post” има поле което препраща към таблицата “size”. Имам предварително създадена таблица с възможните размери и всеки клиент си избира от нея.

size = models.ForeignKey(Size, verbose\_name='Размер', on\_delete=models.CASCADE, null=True, related\_name='size')

description = models.TextField('Описание', default='', blank=True, help\_text='Описание на предлагания продукт')

remark = models.TextField('Забележки', default='', blank=True, help\_text='Текст на съобщение')

picture = models.ImageField('Снимка', upload\_to='post\_pics', blank=True)

user = models.ForeignKey(AppUser, verbose\_name='Потребител', on\_delete=models.CASCADE, null=True,

Създадох и един помощен клас, който работи и правя заявката “create”, необходимо ми е за да обновя данните.

Изграждаме изгледи и имаме два вида. Единият вид е класическият за Django когато извикваме един шаблон. Вторият е така наречените REST услуги.

Потребители влизат, регистрират се с имейл, за да няма анонимни. Публикуват си обяви, който се заинтересова се свързва с този, който е публикувал обявата чрез коментар. И на последно място имам и административен панел, който Django сам го предлага.

# Заключение

Представената разработка представлява едно завършено, пълнофункционално приложение. Основната цел – да се демонстрира определен подход за изграждане на потребителски интерфейс за уеб-базирано приложение я считам за постигната. SDI, така или иначе, присъства в редица приложения. Обикноовенно се използват по-тежки, но и по-мощни, библиотеки като Angular и React. Също така най-често се използва node.js или друга подобна близка до JavaScript среда за разработване на съответния backend. Но, както показва и моето приложение, Python и Django успешно и лесно могат също да се използват. Успешното съчетаване на подходящи технологии позволява даже и на програмисти с по-малък опит да създадът еднодокументно приложение, което изглежда добре, функционира правилно и реализира всички поставени в заданието изисквания.

Естествено, има какво да се желае още и да се развива в бъдеще. Би могло, на първо място да се доразвие функционалността. Например, може да се добави автоматично уведомяване по е-mail или със SMS на подалият обява, когато постъпи запитване, свързано с нея. Може също да се добави допънителна верификация при регистриране на потребители, за да се избегне създаването на „фалшиви“ акаунти. Би могло да се добавят и други функции, но това ще надхвърли значително изискванията на заданието.

Във вида в който е представено приложението е напълно функционално и би могло да се използва и в реална среда.

# Списък на използваната литература

1. <https://softuni.bg/blog/what-is-django>
2. <https://cynoteck.com/bg/blog-post/flask-vs-django/>
3. <https://bg.savtec.org/articles/coding/the-basics-of-rest-and-restful-api-development.html>
4. [https://urocibg.eu/](https://urocibg.eu/%D0%BA%D0%B0%D0%BA%D0%B2%D0%BE-%D0%B5-api/)
5. <https://www.django-rest-framework.org/>
6. <https://softuni.bg/blog/vue-js-january-2021>
7. <https://bg.wikipedia.org/wiki/MySQL>
8. <https://html.w3schools.bg>
9. <https://www.inventum.bg/web-design/css/>