# Съдържание

[*Съдържание* 1](#_Toc133572641)

[1 Увод 2](#_Toc133572642)

[2 Функционалните възможности на Django 2](#_Toc133572643)

[2.1 Предимства на Django 3](#_Toc133572644)

[2.2 Помощ от Общността 3](#_Toc133572645)

[2.3 Адаптивност 3](#_Toc133572646)

[2.4 Ефективност и скорост 4](#_Toc133572647)

[2.5 Админ панел на Django 4](#_Toc133572648)

[3 Rest? 8](#_Toc133572649)

[3.1 RESTful API 9](#_Toc133572650)

[3.2 Изграждане на RESTful API 10](#_Toc133572651)

[4 Python? 10](#_Toc133572652)

[5 Библиотека DRF 14](#_Toc133572653)

[5.1 Характеристики: 14](#_Toc133572654)

[6 MSQL? 16](#_Toc133572655)

[6.1 Поддръжка 17](#_Toc133572656)

[7 HTML? 18](#_Toc133572657)

[7.1 HTML предоставя много елементи за структуриране на уеб страниците. 22](#_Toc133572658)

[8 Проектиране на необходимата функционалност 24](#_Toc133572659)

[8.1 Анализ на заданието 24](#_Toc133572660)

[8.2 Модели 24](#_Toc133572661)

[9 Минимална версия на приложението 26](#_Toc133572662)

[9.1 Сериализатори и изгледи (view) 28](#_Toc133572663)

[10 Заключение 30](#_Toc133572664)

[11 Списък на използваната литература: 31](#_Toc133572665)

# Увод

Моят проект включва backend-а, който е основан на REST и е реализиран през DRF.

Към сайта на ПГЕЕ – Банско (разработен на база python/django) да се добави приложение в смисъла на django за реализиране на софтуерна платформа (наречена „борса за учебници”) за предлагане и търсене на ползвани учебни помагала. Платформата има необходимата функционалност за регистриране на потребители, оторизиране на достъпа до платформата, регистриране на заявки за предлагане и/или търсене на учебници. Решено е платформата да се базира на REST.

# Функционалните възможности на Django

Django е създаден през 2005 година и първата стабилна версия е пусната през 2008 година. Оттогава той е станал една от най-популярните платформи за уеб разработка в света.

Той е безплатен framework с отворен код. Написан е изцяло на python и позволява много по-бързо изграждане на уебсайтове. Също така позволява да се реализира един проект, едновременно по един бюджетно-ориентиран начин, без това да се отрази на функционалността на продукта

Пълен набор инструменти - Django е подходящ за всеки етап от разработката на дадено приложение – потребителска верификация, администриране на съдържанието, RSS feed настройки и още много други. По този начин той е напълно достатъчен за процеса на разработка от начало до край.

**Сигурност** - Django поддържа максимална сигурност, като помага на разработчиците да избягват често срещани уязвимости в сигурността като SQL инжектиране, междусайтови скриптове и кликване със своя уникален механизъм за удостоверяване на потребителя, който осигурява сигурен метод за защита на потребителски акаунти и пароли.

**Гъвкавос**т - Без значение дали става въпрос за управление на съдържанието, специфични компютърни платформи или приложения, които ще се използват от големи организации – Django може да отговори на всяка от тези нужди. Това демонстрира ясно неговата гъвкавост.

Django е **усъвършенстван framework** за разработка на уеб сайтове, който е изминал дълъг път от първоначалното си пускане през 2005 г. По онова време той направи революция в съвременното разработване на уеб сайтове, като създаде простота и предостави на разработчиците мощни инструменти за тяхната работа. В днешно време Django продължава да се преработва към по-добро, като редовно се пускат нови функции. Последната версия на Django предоставя на потребителите увеличена функционалност, подобрена производителност, по-добра мащабируемост и възможности за поддръжка без усилия, които подпомагат потребителското изживяване. С тези подобрения, както и с бъдещите такива, Django остава една от най-добрите рамки за уеб разработка, налични днес; определяща стандартите по отношение на качеството и надеждността.

## Предимства на Django

* Стратегия с включена батерия

Django използва стратегия с включени батерии. Това означава, че Django включва ресурси, от които разработчиците ще се нуждаят. Не е нужно да създавате свой собствен код, за да добавите определена функционалност вместо това, можете да импортирате пакети с отворен код, създадени от мощни членове на общността.

## Помощ от Общността

Django доминира в ИТ индустрията през последните 13 години, което го прави изключително популярна уеб рамка. Силната подкрепа на общността е това, което тласна Django до тези висоти. Хиляди разработчици си сътрудничат, за да поправят всички проблеми или дефекти, които могат да възникнат. Ако имате затруднения в работата или сте хванати някъде и не знаете какво да правите, бързото търсене в Google може да ви спести време и пари. Django винаги се подобрява и развива, тъй като се добавят нови библиотеки, за да бъдат най-добрата платформа на пазара.

## Адаптивност

Мащабируемостта се дефинира като капацитет на приложението да работи добре с нарастването на размера или обема на платформата. Когато има скок в трафика, доброто приложение винаги е програмирано да работи ефективно. В този случай Django работи безупречно. Създадените от Django уебсайтове могат да приемат множество потребители едновременно.

## Ефективност и скорост

Когато обсъждате уеб приложенията, скоростта става важен фактор, който трябва да се вземе предвид. Django позволява на потребителите бързо да създават динамични уеб приложения.

## Админ панел на Django

Админ панелът на Django (Django admin panel) е вграден административен интерфейс, който позволява управление на данните във вашия Django проект. Той ви позволява да създавате, редактирате и триете данни от базата данни без да е необходимо да пишете ръчен SQL код.

За да използвате админ панела на Django, трябва да създадете администраторски акаунт. Това може да се направи чрез команда от конзолата на проекта:

python manage.py createsuperuser

След като създадете администраторски акаунт, можете да влезете в админ панела чрез браузъра си на адрес ‘http://адрес-на-вашия-сайт/admin/’ и да въведете вашият администраторски потребител и парола.

Админ панелът на Django има много полезни функции като търсене, филтриране, сортиране, групиране и множество други. Можете да настроите изгледа на данните за да отговаря на вашите нужди и да добавите/премахнете колони по ваш избор.

Освен това, админ панелът позволява да се правят масови операции, като изтриване на множество записи едновременно. Също така, вие можете да добавите допълнителни функционалности чрез създаване на собствени изгледи за админ панела, като използвате функционалностите на Django.

Админ панелът на Django е много полезен инструмент за администриране на вашия уеб сайт, който може да ви спести много време и усилия в управлението на данните във вашето приложение.

Административният панел (admin panel) на Django е вграден административен интерфейс, който предоставя удобен начин за управление на данните във вашия Django уеб сайт. Той е изграден с помощта на Django framework и предоставя автоматичен интерфейс за добавяне, редактиране и изтриване на данни от моделите на вашата база данни.

За да използвате административния панел на Django, трябва да дефинирате моделите на вашата база данни във файловете на приложението си. Тези модели определят структурата на вашите данни и съдържат логиката за достъп до тях.

Административният панел на Django предлага множество функции и възможности за управление на вашите данни, като:

* Избор на модел за управление
* Добавяне, редактиране и изтриване на обекти
* Филтриране и търсене на обекти
* Изглед на детайли за определен обект
* Създаване на свързани обекти
* Управление на потребителите и групите, имащи достъп до административния панел

Административният панел може да бъде персонализиран и разширен по множество начини, като се използва Django Admin site API. Можете да създадете свои собствени изгледи, форми и действия, които да бъдат вградени в административния панел, за да отговарят на нуждите на вашето приложение.

За да се използва административният панел на Django, трябва да се дефинират модели на данните във вашия проект. Моделите представляват структура на данните, които ще бъдат запазени в базата данни. С помощта на административния панел можете да изпълнявате CRUD операции (Create, Read, Update, Delete) върху тези данни.

За да активирате административния панел в Django, трябва да включите приложението ‘django.contrib.admin’ във вашия ‘INSTALLED\_APPS’ списък във файла ‘settings.py’ на вашия проект. След като приложението е включено, можете да дефинирате административния интерфейс чрез дефиниране на класове наследяващи ‘django.contrib.admin.ModelAdmin’.

Пример за дефиниране на административен интерфейс за модел ‘Product’:

from django.contrib import admin

from .models import Product

class ProductAdmin(admin.ModelAdmin):

list\_display = ('id', 'name', 'price', 'created\_at')

list\_filter = ('created\_at',)

search\_fields = ('name',)

admin.site.register(Product, ProductAdmin)

В горния пример, ‘ProductAdmin’ е клас, който наследява ‘admin.ModelAdmin’. Вътре в класа са дефинирани опции за административния интерфейс като ‘list\_display, list\_filter и search\_fields’.

‘list\_display’ определя списъка с полета, които ще бъдат показани за всяка запис в административния списък. ‘list\_filter’ позволява да се филтрират записите по определени полета, а ‘search\_fields’ позволява да се търсят записи по определени полета.

Накрая, като използвате ‘admin.site.register()’ метода, регистрирате модела ‘Product’ в административния панел, заедно със създадения ‘ProductAdmin’ клас.

В админ панела на Django могат да се извършват и други операции като търсене, филтриране и сортиране на данните, добавяне на нови потребители и групи, настройка на права за достъп до определени данни за потребителите, и много други.

Админ панелът може да се персонализира и разширява по различни начини, като се използват допълнителни пакети и модули. Освен това, може да се използва и на различни езици, включително на български.

За да използвате админ панела на Django на български език, трябва да настроите езиковите настройки на приложението във файловете settings.py и urls.py. В settings.py трябва да зададете настройката за езика на 'bg', а в urls.py трябва да добавите ред за включване на езиковите файлове:

from django.conf.urls.i18n import i18n\_patterns

from django.utils.translation import gettext\_lazy as \_

urlpatterns = [

# ... тук добавяте вашите url patterns ...

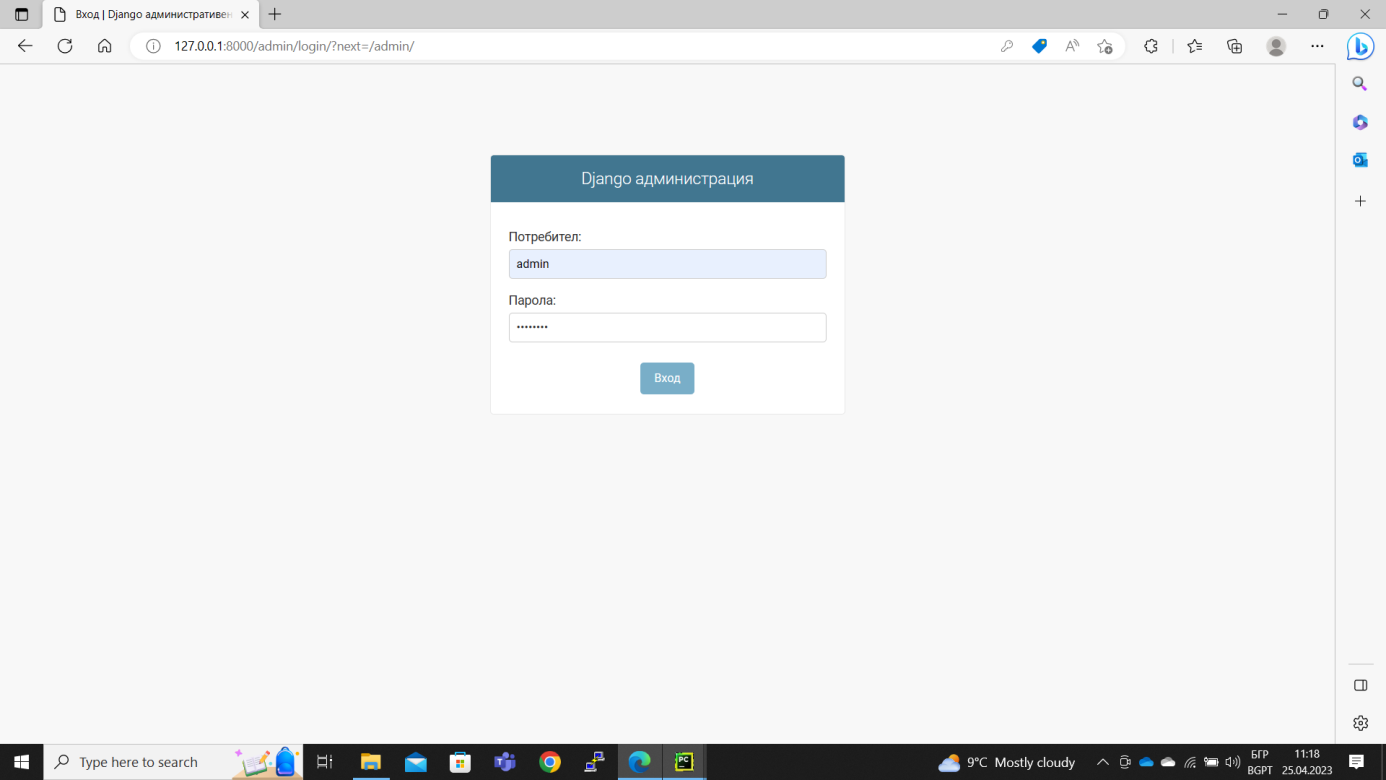
]

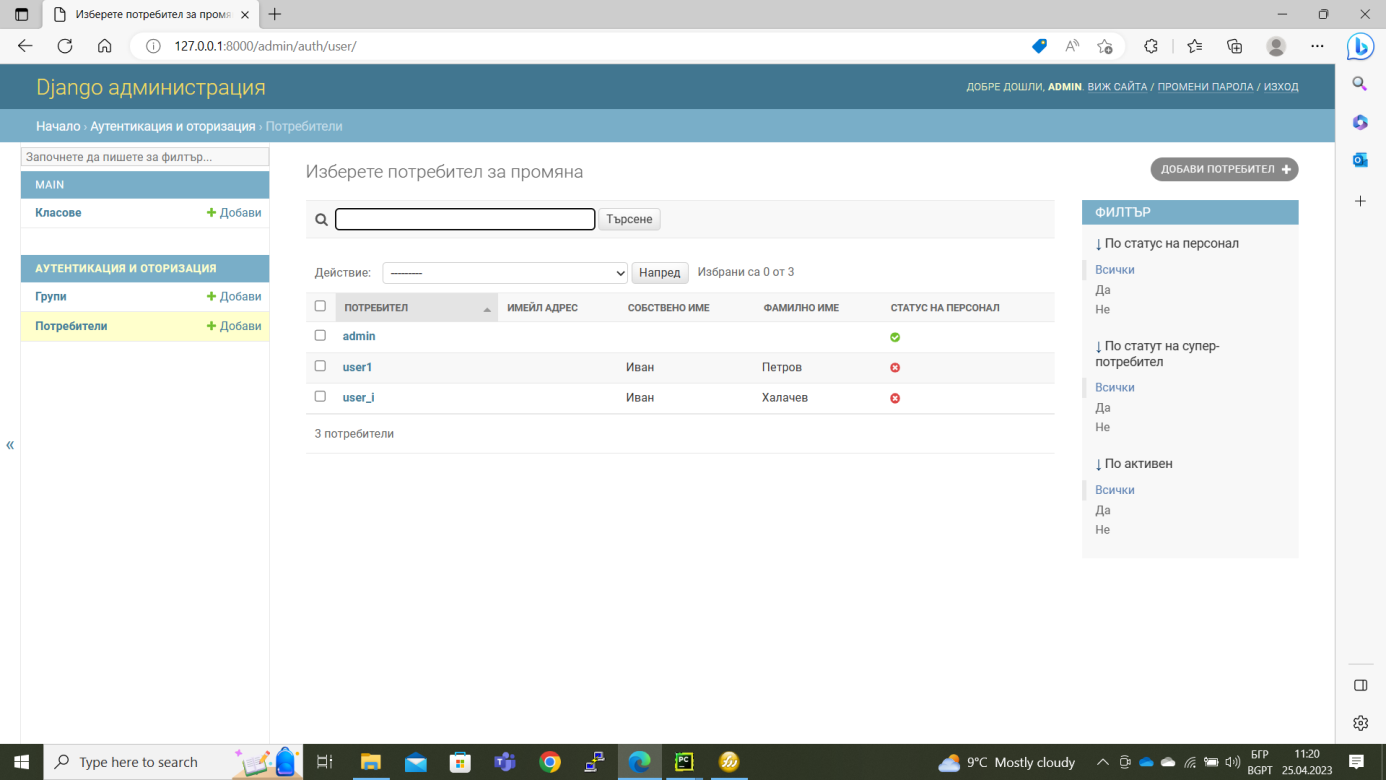
# Patterns за включване на езиковите файлове

urlpatterns += i18n\_patterns(

path('admin/', admin.site.urls),

prefix\_default\_language=False,)

Фиг.1

Фиг. 2

След като направите тези настройки, админ панелът на Django ще се показва на български език. Можете да персонализирате преводите на съобщенията, които се показват в админ панела, като използвате допълнителни езикови файлове.

# Rest?

REST е стил софтуерна архитектура за реализация на уеб услуги. Основната идея е да се определи системен ресурс, който се променя в резултат на взаимодействието между доставчика на услуги и потребителя. Архитектурният модел REST включва взаимодействията между сървър и клиент, осъществени по време на трансфера на данни. Концепцията беше въведена за пръв път от Рой Филдинг през 2000 г. като част от неговата докторска дисертация. Филдинг е един от основните автори на HTTP протокола, под който се изпълняват REST имплементациите в повечето случаи.

Архитектурата REST е разработена успоредно с HTTP 1.1. Въпреки това, REST е обща архитектура, която може да бъде реализирана в други среди, а не само под HTTP. World Wide Web представлява най-голямото осъществяване на архитектурния стил на REST.

REST – стилът обикновено се състои от клиенти и сървъри. Клиентите инициират заявки към сървърите; сървърите преработват заявките и връщат подходящи отговори. Заявките и отговорите са създадени през прехвърляне на образа на ресурси. Ресурсът може да бъде всякаква ясна и смислена концепция, която може да бъде адресирана. Представяне (анг. representation) на ресурс обикновено е документ, който намира сегашното възнамерявано състояние на ресурса.

Клиентът започва да изпраща заявки, когато е готов да направи преходът към ново състояние. Докато една или повече заявки са неизпълнени за клиента се смята, че е в преход. Представянето на всяко приложение се състои от линкове, които могат да бъдат използвани следващия път, когато клиентът избере да направи нови официални промени.

REST е архитектурен стил за уеб приложения, който се основава на принципите на HTTP протокола. REST е акроним от английските думи "Representational State Transfer" (пренасяне на състоянието на ресурсите) и описва начина, по който клиентските приложения и уеб услуги могат да комуникират едно с друго чрез уеб протокола HTTP.

REST използва основни HTTP методи като GET, POST, PUT и DELETE, за да комуникира с ресурсите в уеб приложението. REST използва ресурси (например, потребители, статии, снимки) като централен елемент на архитектурата, като всяко ресурсно действие може да бъде извършено чрез HTTP заявка към този ресурс.

REST е популярен в уеб разработката, защото е лесен за разбиране и използване. RESTful API-та (уеб услуги, които използват REST) са съвместими с голям брой клиентски устройства и програми, като мобилни приложения, уеб браузъри, IoT устройства и други.

Важно е да се отбележи, че REST е само един от множество архитектурни стилове за уеб приложения, като всеки стил има своите предимства и недостатъци. Затова, при избора на архитектурен стил, е важно да се вземат предвид нуждите на проекта и целите на приложението.

Ето някои от основните принципи на REST, които помагат за изграждане на ефективни уеб приложения:

Ресурсно ориентирано програмиране - REST се базира на използването на ресурси като централен елемент на архитектурата. Всяко действие в уеб приложението може да бъде извършено чрез HTTP заявка към определен ресурс.

Използване на HTTP методи - REST използва основните HTTP методи като GET, POST, PUT и DELETE, за да комуникира с ресурсите в уеб приложението. Например, GET заявката се използва за извличане на информация от ресурс, а POST заявката се използва за добавяне на информация към ресурс.

Използване на URL адреси - REST използва URL адреси за идентифициране на ресурсите в уеб приложението. URL адресът на ресурса трябва да бъде постоянен и лесен за разбиране.

Предаване на данни във формат JSON или XML - REST използва форматиране на данните като JSON или XML, за да осигури ефективно предаване на информацията между клиента и уеб услугата.

Независимост от състоянието - REST е "безсъстоятелен" архитектурен стил, което означава, че всеки HTTP заявка към ресурса е независима от предишните заявки. Това прави REST уеб услугите много скалируеми и лесни за управление.

## RESTful API

RESTful API е мощен инструмент за изграждане на уеб приложения, които могат да бъдат използвани от различни клиентски устройства. Важно е да следвате принципите на REST и да изградите API-то така, че да бъде лесно за използване и поддръжка.

RESTful API може да бъде изграден с различни програмни езици и frameworks, включително Django, Flask, Express и други. Важно е да се вземат предвид нуждите на проекта и уменията на екипа при избора на конкретен frameworks за изграждане на RESTful API.

## Изграждане на RESTful API

* Използвайте версии - когато правите промени в RESTful API, е важно да използвате версии, за да не нарушавате съвместимостта със старите клиенти. Можете да добавите номер на версията в URL адреса на API-то, например /api/v1/...
* Използвайте HTTP статус кодове - HTTP статус кодовете са важни за указване на резултата от изпълнението на заявката към RESTful API. Например, 200 означава успешна заявка, 400 означава грешка в заявката, а 404 означава, че ресурсът не е намерен.
* Използвайте HTTP хедъри - HTTP хедърите могат да бъдат използвани за предаване на допълнителна информация за заявката или отговора. Например, може да използвате хедър за авторизация, който съдържа токен за достъп.
* Използвайте кеширане - кеширането може да ускори отговора на RESTful API, като се избягва излишното извличане на информация от базата данни. Можете да използвате HTTP хедъра "Cache-Control" за указване на времето, за което резултатът на заявката може да бъде кеширан.
* Тествайте API-то - тестването на RESTful API е важна стъпка за увереност в правилната му работа. Можете да използвате инструменти като Postman или curl за изпращане на заявки към API-то и проверка на резултатите.

# 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.

# Библиотека 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.

# MSQL?

MSQL е релационна база данни и съкращение от "Microsoft SQL Server". Това е мощна и скалируема база данни, която може да се използва за съхранение и управление на големи множества от данни във всякакъв вид приложения. MSQL използва SQL езика за извличане, манипулиране и управление на данните. Тя е известна също като един от най-популярните релационни бази данни в света и е широко използвана от големи корпорации и уеб сайтове. Първоначално разработена през 1989 г. от Microsoft. Тя е една от най-популярните релационни бази данни и се използва от десетилетия.

* Версии - MSQL е налична в различни версии, включително Express, Standard, Enterprise и Developer. Всяка версия има различни функции и цени, като Express е безплатна версия, която може да бъде използвана за малки приложения и проекти.
* Ядро – Разработена е с ядро на Transact-SQL, което е разширение на SQL езика, позволяващо на потребителите да пишат по-сложни скриптове и да използват по-широк набор от функционалности.
* Скалируемост - MSQL е известна със своята способност да мащабира вертикално и хоризонтално. Тя може да бъде конфигурирана за да работи на единичен сървър или на група от машини, което позволява да се управлява голям брой от заявки и голям обем от данни.
* Поддръжка на индекси - MSQL поддържа различни видове индекси, които подобряват скоростта на изпълнение на заявките. Индексите могат да бъдат създадени върху единични или множество колони в таблиците.
* Репликация - MSQL поддържа репликация, която позволява на данните да бъдат копирани на други сървъри. Това е полезно за изграждане на резервни копия на данните и за увеличаване на скоростта на изпълнение на заявките.
* Сигурност – Известна е със своята висока степен на сигурност. Тя поддържа аутентификация и упълномощаване на потребителите, криптиране на данните и мониторинг на събитията на сървъра за откриване на неоторизирани действия.
* Активна общност - MSQL има активна общност от потребители и разработчици, които работят върху подобряване на базата данни и добавянето на нови функционалности. Това позволява на потребителите да получават поддръжка, да споделят знания и да получават нови възможности за работа с данните.
* Набор от драйвери - MSQL има голям набор от драйвери, които позволяват на приложенията да комуникират с базата данни на различни програмни езици, включително Python, Java, C#, PHP и други.

## Поддръжка

**На транзакции** - MSQL поддържа транзакции, които гарантират атомарност, консистентност, изолация и трайност на данните. Това позволява на приложенията да извършват множество операции върху данните и да ги запазват във валидно състояние.

**На процедури и тригери** - Позволява на потребителите да създават процедури и тригери, които могат да бъдат използвани за автоматизиране на определени задачи или за прилагане на бизнес правила.

**На JSON** - MSQL поддържа JSON (JavaScript Object Notation) формата за съхранение на данни, който е особено полезен за уеб приложенията. JSON е един от най-популярните формати за обмяна на данни и позволява на потребителите да съхраняват и да използват данни във формат, който е по-лесен за работа с някои уеб технологии.

**Облачна поддръжка** - MSQL е налична като услуга в облака на Microsoft Azure. Това позволява на потребителите да създават и да управляват бази данни в облака, като се възползват от различните предимства на облачните услуги, включително гъвкавост, мащабируемост и достъпност.

# HTML?

HTML е абревиатура от "Hypertext Markup Language", което в превод на български означава "Език за оформление на хипертекстови документи". HTML е стандартен език за създаване на уеб страници и е използван от милиони уеб сайтове в света.

Базиран е на технологията за маркиране на документи, която позволява на потребителите да създават структурирани документи, като поставят маркери на различни елементи в документа. Тези маркери определят какво представлява всеки елемент и как да бъде показан на екрана на потребителя.

Използва множество елементи, които могат да бъдат комбинирани, за да се създаде страници с различна структура и дизайн. Някои от най-често използваните елементи в HTML включват заглавия, параграфи, списъци, таблици, форми и мултимедийни елементи като изображения и видеоклипове.

Създаден е от Тим Бърнърс-Лий в началото на 1990-те години и е постепенно развиван от международна общност от разработчици и потребители. В момента, HTML е стандартизиран от World Wide Web Consortium (W3C) и се поддържа от всички модерни уеб браузъри.

HTML е език за оформление на уеб страници, който се използва за създаване на структурирани документи, които могат да бъдат показани в уеб браузърите. Той предоставя средства за маркиране на различни елементи в документа, като използва HTML елементи за да създаде визуални ефекти и да свързва различни части на документа с други документи или уеб сайтове.

HTML е създаден за да бъде лесен за употреба, тъй като много от елементите са базирани на често използвани функции като заглавия, параграфи, списъци и т.н. За да се напише HTML документ, не е необходимо да се има специално обучение, но е от полза да се има познания за основните концепции и синтаксис на езика.

HTML използва елементи, които определят какво представлява всеки елемент в документа и как да се покаже на екрана. Тези елементи са вложени един в друг и могат да се комбинират за да се създаде различен вид страници. Например, за да се направи заглавие на страницата, може да се използва елемента <h1>, за да се създаде параграф, може да се използва елемента <p>, за да се създаде списък, може да се използва елемента <ul> или <ol>.

HTML е използван в комбинация с други технологии като CSS и JavaScript, за да се създадат интерактивни уеб страници и уеб приложения. CSS се използва за оформление на уеб страници, като определя външния вид на елементите, а JavaScript се използва за създаване на динамични ефекти, като например форми за вход или анимации на страници.

HTML е стандартен език за оформление на уеб страници и се използва от милиони уеб сайтове в целия свят. Той е разработен от World Wide Web Consortium (W3C) и е актуална версия е HTML5.

HTML използва различни елементи, които се състоят от тагове, атрибути и текстово съдържание. Таговете се използват за да определят типа на елемента, който се използва, като например <p> за параграф или <img> за изображение. Атрибутите се използват за да добавят допълнителни свойства към елемента, като например src атрибута на <img>, който показва къде се намира изображението. Текстовото съдържание е всичко, което се появява между отварящия и затварящия таг на елемента.

HTML е важна част от уеб разработката и е необходимо да се научи за да се създадат добри уеб сайтове и уеб приложения. В съчетание с CSS и JavaScript, HTML може да се използва за да се създадат съвременни и интерактивни уеб страници и приложения. Също така, HTML е важен за SEO (Search Engine Optimization), тъй като добре структурираните HTML документи могат да помогнат на уеб сайтовете да се появят по-високо в резултатите от търсачки.

HTML е съкращение от HyperText Markup Language, което буквално означава език за маркиране на хипертекст. Той е основен елемент от уеб разработката и се използва за да се дефинират структурата и съдържанието на уеб страници. HTML файловете имат разширение .html или .htm и се използват за да се генерира текстовото съдържание, снимки, връзки, формуляри и други елементи, които съставят уеб страници.

HTML е важен за създаването на уеб сайтове, защото той позволява на браузърите да разберат как да покажат съдържанието на уеб страницата. Той позволява на уеб разработчиците да дефинират структурата и съдържанието на уеб страницата, като използват различни HTML елементи, като заглавия, параграфи, списъци, таблици и други. Тези елементи могат да бъдат стилизирани с помощта на CSS, за да се подобри изгледът и функционалността на уеб страницата.

HTML5 е последната версия на HTML и включва много нови функционалности, като например възможността за вграждане на аудио и видео, драг-енд-дроп функционалност, формуляри за вход чрез сензорни устройства, подобрени мултимедийни възможности и други. HTML5 се използва широко в уеб разработката и е важен елемент от съвременните уеб приложения и уеб сайтове.

Важно е да се отбележи, че HTML не е единственият елемент от уеб разработката, а част от една по-голяма екосистема, който включва CSS за стилизиране и JavaScript за добавяне на интерактивност и динамичност. В комбинация с други технологии, HTML може да бъде използван за да се създадат мощни, интуитивни и добре структурирани уеб приложения.

HTML е основен елемент от уеб разработката и се използва за да се дефинират структурата и съдържанието на уеб страници. Той използва тагове за да дефинира различни елементи на уеб страницата, като например заглавия, параграфи, списъци, таблици и други. Таговете в HTML са обвити във <> скоби, като те могат да имат различни атрибути, които определят допълнителни свойства на елемента.

HTML е език за маркиране, което означава, че той не е програмен език, а по-скоро текстов език, който описва как да се представят съдържанието и елементите на уеб страницата. HTML е интерпретиран от браузъра на потребителя и се превръща във визуална представа на уеб страницата, която може да бъде визуализирана и манипулирана от потребителя.

HTML е част от технологиите, които са необходими за да се създаде уеб страница, заедно с CSS и JavaScript. CSS се използва за да се добави стил и дизайн на уеб страницата, а JavaScript се използва за да се добави динамичност и интерактивност. Тези технологии са се развивали заедно през годините и са важен елемент от съвременните уеб приложения.

HTML е бил постоянно развиван през годините, като последната версия на езика, HTML5, включва много нови функционалности, като например възможността за вграждане на аудио и видео, драг-енд-дроп функционалност, формуляри за вход чрез сензорни устройства, подобрени мултимедийни възможности и други. HTML5 се използва широко в уеб разработката и е важен елемент от съвременните уеб приложения и уеб сайтове.

HTML (HyperText Markup Language) е език за маркиране, който се използва за да се определят структурата и съдържанието на уеб страниците. Той е стандартен език за маркиране на уеб документи, който се използва за да се създаде визуалното представяне на уеб страници, както и за да се добавят хипервръзки към други документи и ресурси в Интернет.

HTML е изобретен от Тим Бърнърс-Лий в 1989 г. Той е разработил този език за маркиране, за да помогне на физиците в ЦЕРН да споделят информация с други учени по света, като използват един общен език за маркиране. След това HTML бързо се превърна в стандартен език за маркиране на уеб документи.

HTML дефинира структурата на уеб страницата чрез тагове. Таговете са елементи, които определят типа на съдържанието в страницата, като например заглавие, параграф, снимка, хипервръзка и т.н. Таговете се пишат в скоби <> и се закриват със / в скоби </>. Например, <h1> е таг за заглавие, а </h1> е закриващият таг. Съдържанието между отварящия и закриващия таг се нарича "съдържание на тага".

HTML е част от технологиите, използвани за уеб разработка, заедно с CSS и JavaScript. CSS (Cascading Style Sheets) се използва за да се добавят стилове към уеб страниците, като например цветове, шрифтове, размери и т.н. JavaScript се използва за да се добавят динамичност към уеб страниците, като например ефекти, валидация на форми, взаимодействие с потребителя и други.

## HTML предоставя много елементи за структуриране на уеб страниците.

* <head> - този таг съдържа информация за уеб страницата, като заглавието на страницата, метатегове, които определят метаданни за страницата (като автор, описание, ключови думи и т.н.), както и връзки към други файлове и ресурси.
* <body> - този таг съдържа основното съдържание на уеб страницата, като текст, изображения, хипервръзки, формуляри и други.
* <div> - този таг се използва за да групира елементи в една обща секция на уеб страницата. <div> е като празен контейнер, който може да съдържа каквато и да е друга информация или елементи вътре в него.
* <span> - този таг се използва за да групира малки парчета текст в една обща секция на уеб страницата. <span> е като празен контейнер, който може да съдържа текст или други елементи вътре в него.
* <ul> - този таг създава неномериран списък (булетин) на уеб страницата, като например списък на продуктите или услугите, които са налични.
* <ol> - този таг създава номериран списък на уеб страницата, като например списък на стъпките за изпълнение на някоя дейност или процес.
* <table> - този таг създава таблица на уеб страницата, като например таблица с данни, която показва информация за продукти или услуги.
* <form> - този таг създава формуляр на уеб страницата, който позволява на потребителите да изпращат информация към уеб сайта, като например регистрационна форма или форма за контакт.
* <input> - този таг се използва вътре в <form> тага, за да създаде полета за въвеждане на информация, като например текстови полета, полета за пароли, бутони за изпращане и други.
* <img> - този таг се използва за да вмъкнете изображение на уеб страницата. В него трябва да се посочи пътят към изображението, като може да се използват както локални, така и уеб адреси.
* <header> - този таг се използва за да означите, че определен блок от елементи представлява заглавие на уеб страницата.
* <footer> - този таг се използва за да означите, че определен блок от елементи представлява края на уеб страницата, където може да се намерят връзки към други страници, контактна информация и др.
* <article> - този таг се използва за да означите, че определен блок от елементи представлява статия, новина или друго самостоятелно съдържание на уеб страницата.
* <section> - този таг се използва за да означите, че определен блок от елементи представлява раздел на уеб страницата, като например секция за продукти, услуги и т.н.
* <form> - този таг се използва за да създадете HTML форма, в която потребителите могат да въвеждат данни. Формата може да съдържа полета за въвеждане на текст, чекбоксове, радио бутони, падащи менюта, бутони за изпращане на данни и други елементи.
* <input> - този таг се използва за да създадете поле за въвеждане на данни във формата. Тагът има различни атрибути, като type, който указва типа на полето за въвеждане (текст, парола, файл, чекбокс и др.), name, който определя името на полето, и value, който задава началната стойност на полето.
* <select> - този таг се използва за да създадете падащо меню във формата, от което потребителят може да избере една или повече опции. Тагът има деца елементи, които са <option>, който задава всяка от опциите в менюто.
* <textarea> - този таг се използва за да създадете поле за въвеждане на многоредов текст във формата. Тагът има атрибут "rows", който задава броя на редовете в полето, и "cols", който задава броя на колоните.

HTML е стандартен език за маркиране на уебстраници и документи в интернет. В него се определя структурата и съдържанието на уебстраницата, като се използват различни елементи като заглавия, параграфи, линкове, картинки и други.

В HTML съществуват различни елементи и атрибути, които се използват за определяне на визуални и структурни елементи на уебстраницата. Елементите са определени чрез тагове, които се ограждат с "<" и ">". Например, тагът "<p>" се използва за дефиниране на параграф, а тагът "<a>" за дефиниране на линк.

HTML може да се използва за създаване на статични уебстраници, както и за разработка на динамични уебприложения в комбинация с други езици за уебпрограмиране като CSS и JavaScript. За да започнете да учите HTML, можете да потърсите онлайн ресурси като уроци и документация.

# Проектиране на необходимата функционалност

## Анализ на заданието

Заданието за дипломния проект предполага, реализиране на софтуерна платформа (наречена „Борса за учебници“). Платформата има необходимата функционалност за регистриране на потребители, оторизиране на достъпа до платформата, регистриране на заявки за предлагане и/или търсене на учебници.

## Модели

Слоят за данни в Django е представен от файла models.py на приложението, който представя моделите които го свързват с базата данни. В конкретният случай съм избрала БД MySQL. Всеки модел в този файл е клас, който описва съответна таблица в БД. Например, моделът Klas описва структурата на таблица, която ще се съхраняват отделните отделните класове.

class Klas(models.Model):  
 class\_num = models.SmallIntegerField('Клас', default=12)  
  
 def \_\_str\_\_(self):  
 if self.class\_num == 8:  
 result = str(self.class\_num)+'-ми'  
 else:  
 result = str(self.class\_num) + '-ти'  
 return result  
  
 class Meta:  
 verbose\_name = 'Клас'  
 verbose\_name\_plural = 'Класове'

Създадени са следните модели:

* Клас – за отделните класове
* Оферти – какво се предлага
* Отзиви – как е оценено
* Моделът Оферти

class Offer(models.Model):  
 title = models.CharField('Име', max\_length=100, default='', help\_text='име на офертата')  
 class\_num = models.ForeignKey(Klas, name='class\_id', on\_delete=models.SET\_NULL, null=True, blank=True)  
 content = models.CharField('съдържание', max\_length=100, default='')  
 price = models.CharField('цена', max\_length=50, default='')  
 remark = models.CharField('коментар', max\_length=100, default='')  
  
 def \_\_str\_\_(self):  
 return self.title  
  
 class Meta:  
 verbose\_name = 'Предложение'  
 verbose\_name\_plural = 'Предложения'

* Моделът Отзиви

class Comment(models.Model):  
 text = models.CharField('Име', max\_length=100, default='')  
 offer\_id = models.ForeignKey(Offer, on\_delete=models.SET\_NULL, null=True, blank=True)  
  
 def \_\_str\_\_(self):  
 return self.text  
  
 class Meta:  
 verbose\_name = 'Предложение'  
 verbose\_name\_plural = 'Предложения'

* 1. **URLs**

Това е функционалността, която се реализира от файла urls.py и отговаря за приемането и „дешифрирането“ на клиентските заявки:

urlpatterns = [  
 path('', views.index, name='home'),  
 path('offers\_by\_class/<int:klas>/', views.offers\_by\_class),  
 path('login', views.MyprojectLoginView.as\_view(), name='login\_page'),  
 path('register', views.RegisterUserView.as\_view(), name='register\_page'),  
 path('logout', views.MyProjectLogout.as\_view(), name='logout\_page'),  
]

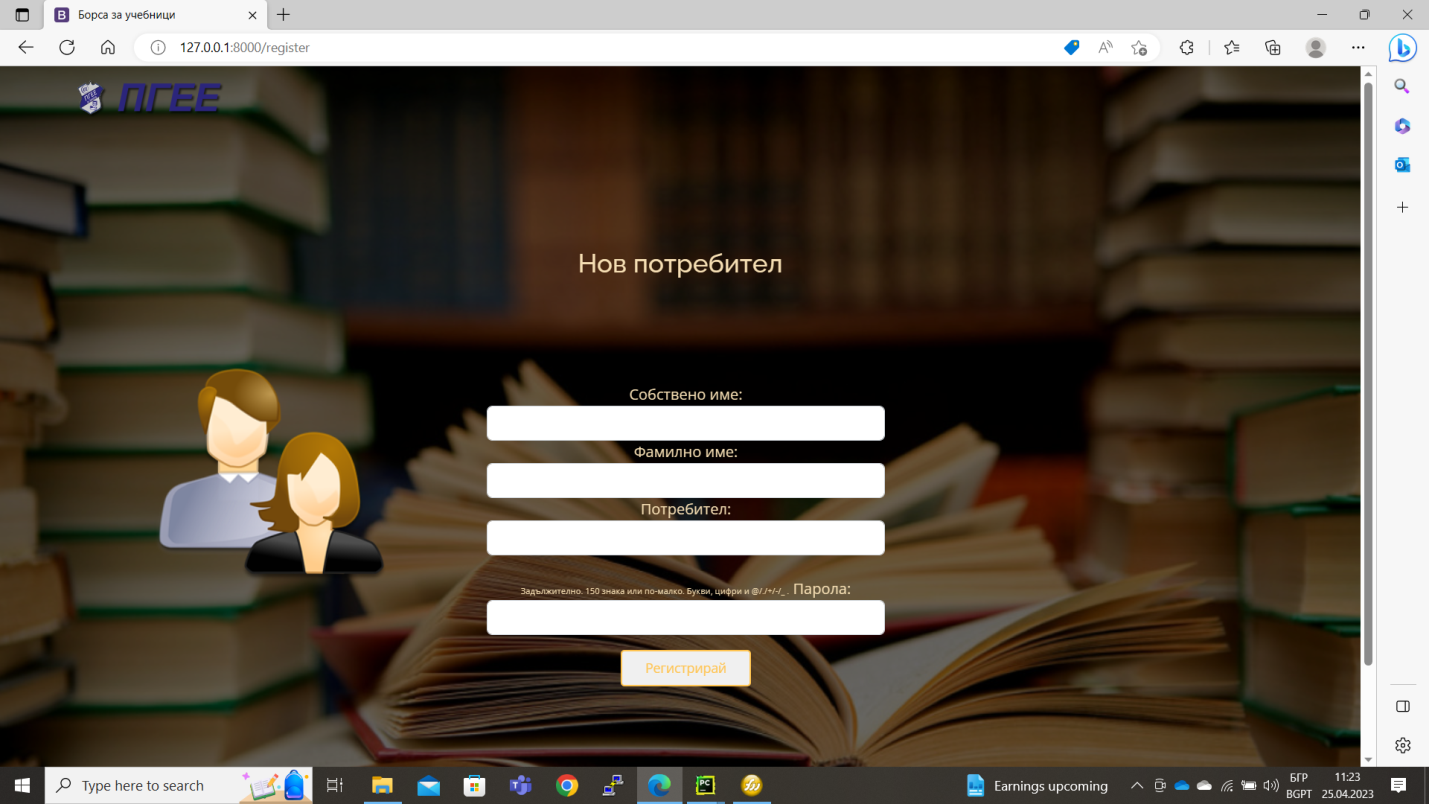
Заявките пристигат под формата на URL, който се проверява за съвпадение с един от записите в списъка urlpaterns. Ако има такова се извиква съответния изглед, който от своя страна генерира отговор на получената заявка.

# Минимална версия на приложението

Визуалното представяне на описаните по-горе елементи на приложението изглежда по следния начин:

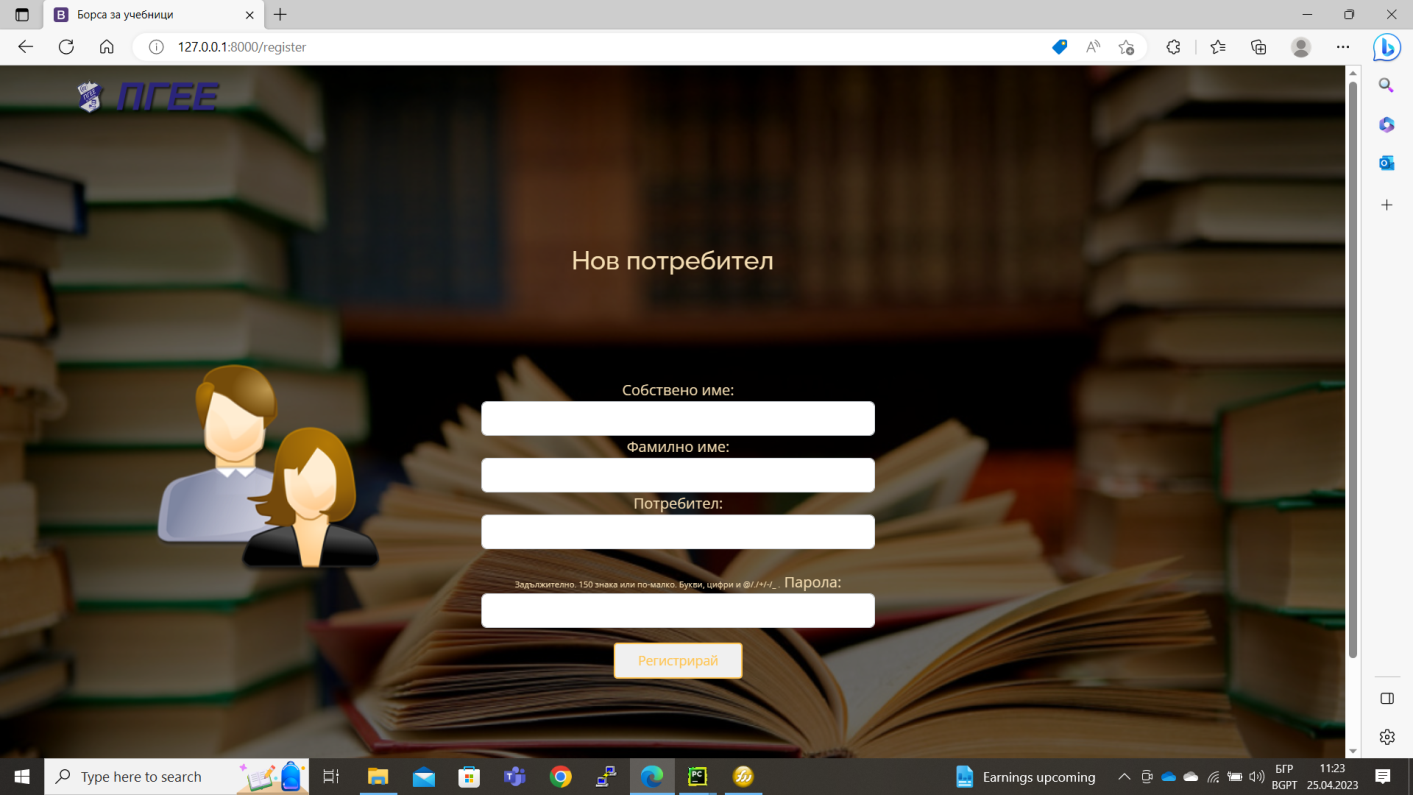
* Регистриране на нови потребители (Фиг. 3)

Фиг.3

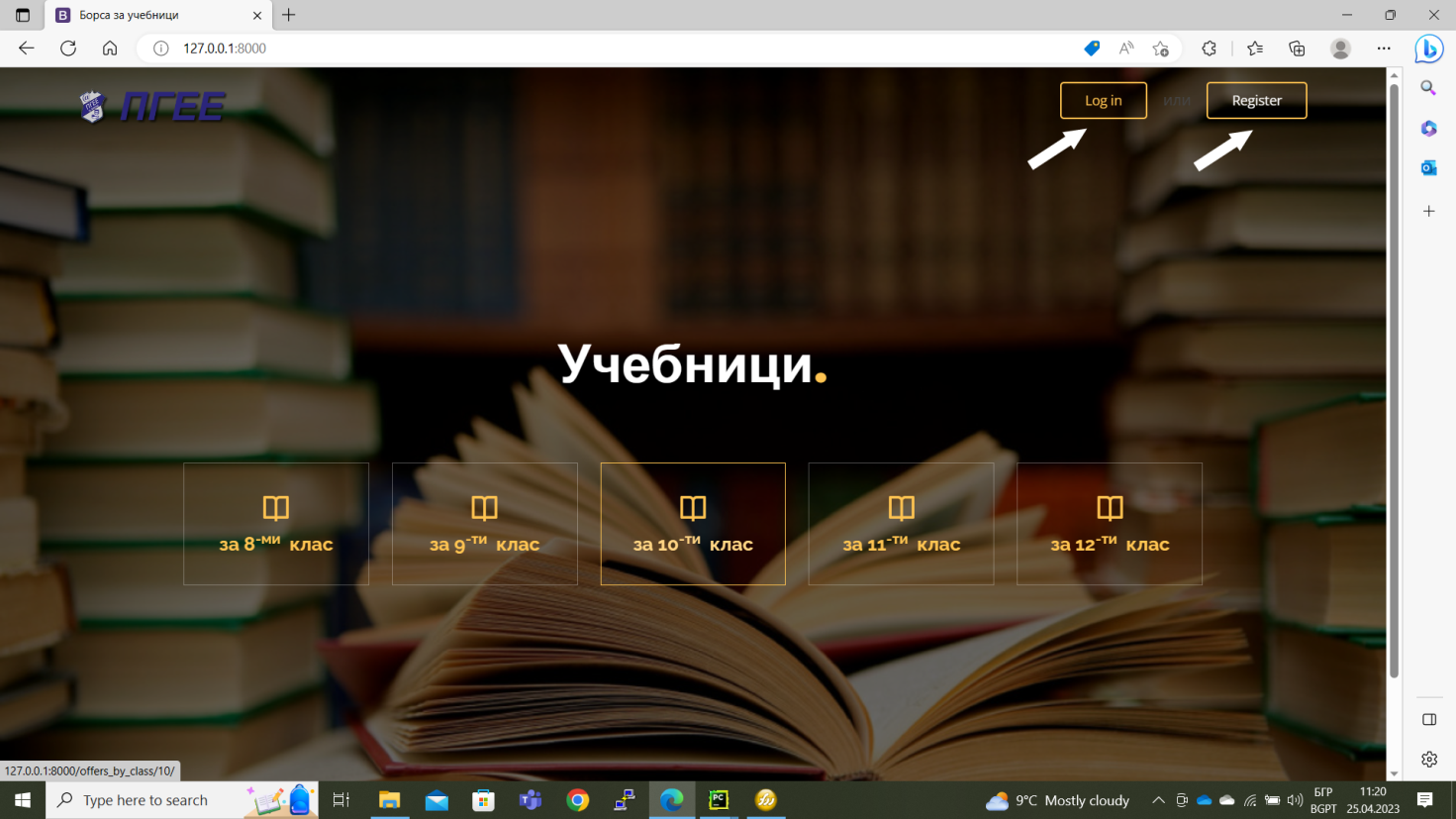


* Оторизиране достъпа на потребител (Фиг. 4)

Фиг. 4



Фиг. 5



## Сериализатори и изгледи (view)

* Сериализаторите и изгледите (views) са ключови компоненти във фреймуърка Django за изграждане на уеб приложения. Ето как могат да бъдат обяснени:

Сериализаторът в Django се използва за преобразуване на сложни обекти (например модели на база данни) във формат, който може да бъде лесно използван за изпращане по мрежата. Това може да бъде JSON, XML или друг формат, който е необходим за приложението. Основната цел на сериализатора е да направи данните достъпни за приложенията, като ги преобразува в нужния формат за тях.

Изглед (View): Изгледът в Django е компонент, който обработва HTTP заявки и връща HTTP отговори. Той представлява логиката на приложението и извлича информация от базата данни, която е необходима за обработката на заявката. Изгледът може да извлича информация и от други източници, като например външни API-та или файлове. Основната цел на изгледа е да обработва заявките на потребителите и да връща нужните им отговори.

Сериализаторите и изгледите (views) в Django са мощни инструменти, които позволяват на разработчиците да създават гъвкави и мащабируеми уеб приложения. Сериализаторите са отговорни за преобразуването на обекти от Python във формат, който може да се предаде по мрежата, като например JSON или XML. Това е често използвано при създаването на RESTful API-та в Django. Сериализаторите позволяват на разработчиците да контролират точно какво ще бъде включено в изходящия формат, като допускат и вложени структури, които могат да бъдат свързани с моделите на базата данни.

Изгледите в Django са отговорни за обработката на HTTP заявки и генерирането на HTTP отговори, като HTML страници или JSON обекти. Изгледите предоставят множество функции за обработка на данните, включително филтриране, сортиране, пагинация и валидация. Django предлага няколко типа изгледи, включително функционални изгледи, клас-базирани изгледи и обобщени изгледи. В зависимост от нуждите на приложението, разработчиците могат да изберат подходящия тип изглед за реализацията на конкретна функционалност.

Сериализаторите и изгледите в Django могат да бъдат написани на български език, като е важно да се запази еднакъв стил на програмиране и да се спазват добрите практики за оформление на кода. В Django също така съществуват и документация и ръководства на български език, които могат да бъдат полезни при работа с тези инструменти.

* Сериализатори и изгледи (view) са две ключови концепции в уеб разработката с Python и Django. Ето как може да ги обясним на български език:
* Сериализатори:

Сериализаторите са класове в Django REST Framework (DRF), които трансформират данни от моделите в нашия Django проект във формат, който може да бъде изпратен към клиентите във формата на JSON, XML и други. С други думи, сериализаторите преобразуват данни от Python обекти във формат, който може да бъде използван от клиента. Също така, те позволяват да се валидират данните, които се изпращат от клиента към сървъра, като проверят дали данните отговарят на нужните условия.

* Изгледи (view):

В Django изгледите са функции или класове, които обработват HTTP заявки, получени от клиента. Те извличат или обработват данните, необходими за отговор на заявката и връщат HTTP отговор във формата на HTML, JSON, XML и други. Изгледите могат да бъдат класове или функции, като в последния случай се използва декоратора @api\_view, за да се конвертират функциите в изгледи, които могат да бъдат използвани в Django REST Framework.

# Заключение

Общо взето това е моят дипломен проект. Смятам, че има какво още да се допълни по него, но на този етап защитата на проекта приключва до тук, а за в бъдеще ще се опитам да се усъвършенствам повече.

Този дипломен проект е само минимално приложение, защото все още няма реални обяви. За в бъдеще ще се направят още промени, за да се качи на реалният хостинг.

# Списък на използваната литература:

1. <https://softwareacademy.bg/index.php?q=info&info=django+now/>
2. <https://softuni.bg/blog/advantages-of-django-framework>
3. <https://www.pro-soft.bg/web-saitove/django/>
4. <https://bg.wikipedia.org/wiki/REST>
5. <https://www.python.org/>
6. <https://chibisov.github.io/drf-extensions/docs/>
7. <https://bg.altlibs.net/lib/django-drf-filepond>
8. <https://softuni.bg/blog/what-is-html5-and-how-it-works>
9. <https://html.start.bg/%D0%9A%D0%B0%D0%BA%D0%B2%D0%BE+%D0%B5+HTML%3F-19832>
10. <https://www.sourcetrail.com/bg/python/django/solved-django-serializer-method-field-read-write/>
11. <http://javac.bg/?p=522>
12. <https://www.django-rest-framework.org/api-guide/serializers/>