Съдържание

[1. Калориен Калкулатор 3](#_Toc164740785)

[2. Python 5](#_Toc164740786)

[3. Django 7](#_Toc164740787)

[4. Admin Panel (Django) 8](#_Toc164740788)

[5. HTML 12](#_Toc164740789)

[6. Структура на един HTML елемент 13](#_Toc164740790)

[7. CSS и JavaScript 16](#_Toc164740791)

[8. CSS 16](#_Toc164740792)

[9. JavaScript 16](#_Toc164740793)

[10. SOAP. 17](#_Toc164740794)

[11. Jinja. 18](#_Toc164740795)

[12. XAMPP: 19](#_Toc164740796)

[12.1. Нужда от XAMPP: 20](#_Toc164740797)

[12.2. Предимства на XAMPP: 20](#_Toc164740798)

[13. Технологии 21](#_Toc164740799)

[13.1. DRF 21](#_Toc164740800)

[14. MySQL 23](#_Toc164740801)

[15. Бази данни 24](#_Toc164740802)

[16. PyCharm 26](#_Toc164740803)

[17. Реализация 27](#_Toc164740804)

[17.1. Анализ на заданието 27](#_Toc164740805)

[17.2. Архитектуран на приложението 27](#_Toc164740806)

[17.3. Модели 27](#_Toc164740807)

[17.4. Изгледи 28](#_Toc164740808)

[17.5. Шаблони 29](#_Toc164740809)

[17.6. Пътища 30](#_Toc164740810)

[18. Заключение 31](#_Toc164740811)

[19. Списък на използваната литература: 32](#_Toc164740812)

# Калориен Калкулатор

Когато говорим за калориен калкулатор, обикновено имаме предвид инструмент, който помага да се изчисли дневния прием на калории, взимайки предвид фактори като възраст, пол, тегло, височина и ниво на активност. Това е полезен инструмент за хора, които се стремят да контролират своята диета и физическа активност, за да постигнат определени цели като отслабване или поддържане на здравословно тегло.

 Калориен калкулаторът предоставя информация за това колко калории трябва да консумирате на ден, за да постигнете определени цели, като например загуба на тегло, покачване на мускулна маса или поддържане на текущото ви тегло. Този инструмент може да бъде полезен и за хора със специални диети или здравословни проблеми, като диабет или високо кръвно налягане, за да се управлява консумацията на храна по ефективен начин.

 При използването на калориен калкулатор е важно да се имат предвид някои фактори. Например, той може да даде само приблизителна оценка на вашите нужди от калории, а не точно число. Също така, важно е да се помни, че калкулаторът не взема предвид всички индивидуални разлики в метаболизма и физическата активност. Затова е важно да се консултирате със специалист, като диетолог или лекар, преди да променяте режима си на хранене.

 При използването на калориен калкулатор е полезно да се отбележи, че той може да бъде само един инструмент за постигане на вашите цели по отношение на диета и физическа активност. Важно е да се вземат предвид и други фактори като качеството на храната, разнообразието във вашата диета, регулярната физическа активност и добрият сън. Съчетаването на тези фактори може да доведе до по-добри резултати в дългосрочен план

Когато използвате калориен калкулатор, бъдете внимателни да не паднете в капана на ограничаване на калориите твърде много. Това може да доведе до лоши последици като недостатъчно хранене, лошо настроение, намалена енергия и дори нарушения в хранителните навици. Вместо това, съсредоточете се върху качеството на храната и правилното съотношение на макронутриентите, като се стремите към здравословен и умерен начин на хранене.

Важно да се има предвид, че индивидуалните нужди от калории могат да варират в зависимост от различни фактори като генетика, метаболизъм и здравословни проблеми. Затова е добра идея да проследите реакцията на вашето тяло и да се консултирате с професионалист, ако срещнете затруднения или имате въпроси относно вашата диета и хранителния режим.

 Изборът на правилния калориен калкулатор е също важна стъпка. Някои калкулатори използват различни формули и методи за изчисляване на нуждите от калории, затова може да получите различни резултати от различни инструменти. Препоръчително е да изберете калкулатор, който взема предвид вашите индивидуални фактори като възраст, пол, тегло, височина и ниво на активност, за да получите по-точна оценка на вашите нужди от калории.

 Когато използвате калориен калкулатор, важно е да бъдете реалистични относно своите цели и очаквания. Постигането на здравословни и устойчиви резултати изисква време, търпение и последователност. Помнете, че всяко тяло е различно и резултатите могат да варират от човек на човек. Важно е да слушате своето тяло, да се храните правилно и да упражнявате редовно, като по този начин се стремите към подобряване на вашето здраве и благополучие.

Ето някои от плюсовете и минусите на използването на калориен калкулатор:

Плюсове:

* \*\***Помага за осъзнаване на консумацията на калории:**\*\* Калкулаторът може да ви помогне да осъзнаете колко калории получавате от вашата храна и напитки.
* \*\***Подпомага постигането на цели:**\*\* С правилно изчислени калории може да постигнете по-лесно вашите цели, като например загуба на тегло или покачване на мускулна маса.
* \*\***Индивидуализация:**\*\* Можете да настроите калкулатора според вашите индивидуални фактори като възраст, пол, тегло и ниво на активност.

Минуси:

* + **\*\*Ограничения:\*\*** Калориен калкулаторът не взема предвид всички индивидуални разлики и може да даде само приблизителна оценка на вашите нужди от калории.
  + **\*\*Стрес и ограничения:\*\*** Някои хора могат да се чувстват стресирани или ограничени от постоянното броене на калории, което може да доведе до негативни възприятия за храненето и тялото им.
  + **\*\*Липса на баланс:\*\*** Фокусирането само върху калориите може да пренебрегне важността на качеството на храната и общото здравословно хранене.

 Важно е да се помни, че калориен калкулаторът е само един инструмент и трябва да се използва като част от по-широк подход към здравословното хранене и физическата активност.

# Python

 В Python можем да създаваме променливи, които пазят стойности. Променливите не се декларират с определен тип, тъй като Python е динамично типизиран език. Някои от основните типове данни включват числа (цели, дробни), символни низове и булеви стойности.

  Ще научите как да използвате оператори за математически и логически операции. Той е разпространен в уеб разработката, научните изследвания, разработката на софтуер и други области.

  Условните оператори (if, else, elif) ви позволяват да вземате решения въз основа на определени условия. Те са мощен инструмент за управление на изпълнението на програмата в зависимост от състоянието на променливи или външни

Циклите (for и while) са структури, които ви позволяват да повтаряте изпълнението на блок от код няколко пъти. Това е полезно, когато искате да изпълните еднакъв код за множество стойности или докато определено условие е вярно.Списъците са структури, които позволяват да съхранявате последователност от стойности в Python. Речниците са асоциативни структури, които позволяват да свързвате ключове със стойности. Тези структури са полезни при работата с големи набори от данни или при организиране на информацията по определен начин.

Функциите са блокове от код, които могат да бъдат използвани повторно в програмата.   Те се дефинират с ключовата дума "def" и могат да приемат параметри и да връщат стойности. Използването на функции улеснява четимостта и поддръжката на програмния код. Python има голямо разнообразие от модули и библиотеки, които предоставят допълнителна функционалност. Вие ще научите как да импортирате и използвате стандартни модули, както и как да инсталирате и използвате библиотеки от трети страни. Обработката на изключения позволява на програмата да се справя с грешки по време на изпълнение, които могат да се случат.

 Python позволява разделянето на една програма на модули, които могат да се използват отново в други програми. Също така притежава голям набор от стандартни модули, които да се използват като основа на програмите. Съществуват и вградени модули, които обезпечават такива събития и елементи като файлов вход/изход (*I/O*), различни системни функции, сокети (*sockets*), програмни интерфейси към GUI библиотеки като Tk и Поради факта, че Python е интерпретативен език, се спестява значително време за разработка, тъй като не са необходими компилиране и свързване  (*linking*) за тестването на дадено приложение. Програмната му идеология е сходна с тази на Java и всяко приложение, написано на него, е сравнително лесно преносимо и в други платформи. Програмите, написани на Python, са доста компактни и четими, като често те са и по-кратки от своите еквиваленти, написани на C/C++. Това е така, понеже: наличните сложни типове данни позволяват изразяването на сложни действия с един-единствен оператор;

* групирането на изразите се извършва чрез отстъп, вместо чрез начални и крайни скоби или някакви ключови думи (друг език, използващ такъв начин на подредба, е Haskell);
* не са необходими декларации на променливи или аргументи; Python съдържа прости конструкции, характерни за функционалния стил на програмиране, които му придават допълнителна гъвкавост. Всеки модул на Python се компилира преди изпълнение до код за съответната виртуална машина.

.

* Езикът е строго типизиран (strong typing) – при несъответствие между типовете е необходимо изрично конвертиране. Езикът поддържа и динамична типизация (dynamic typing) – типовете на данните се определят по време на изпълнението.  Работата се основава на принципа duck typing – типът на обектите се оценява според техните свойства.
* Поддържа се garbage collector – вътрешната реализация на езика се грижи за управлението на паметта. Блоковете се формират посредством отстъп. Като разграничител между програмните фрагменти се използва нов ред. За разлика от много други езици, Python не изисква декларации на променливи, те се създават при инициализацията им.

 Python разполага с вградени в самия език структури като: комплект (tuple), списък (list) и речник \*\*(dictionary/map). Типът на променливата се определя от типа на присвоените стойности, в това отношение прилича на BASIC и се различава от много други програмни езици. Присвояването на различни стойности е коректно и това води до промяна на типа на променливата спрямо последната и присвоена стойност.

# Django

Django  е безплатна и отворена уеб рамка, написана на Python. Уеб рамката е набор от компоненти, които ви помагат да развивате уеб сайтове по-бързо и по-лесно. Когато създавате уебсайт, винаги се нуждаем от подобен набор от компоненти: начин за справяне с удостоверяването на потребителя (регистрация, влизане, излизане), панел за управление на вашия уебсайт, формуляри, начин за качване на файлове .

 За наше щастие, други хора отдавна забелязаха, че уеб разработчиците са изправени пред подобни проблеми при изграждането на нов сайт, така че те се обединиха и създадоха рамки (Django е една от тях), които ви дават готови компоненти за използване. Рамките съществуват за да ви спестяват нуждата да преоткривате колелото и да помогнат за облекчаване на част от разходи, когато създавате нов сайт.

 За да разберем за какво всъщност е Django, трябва да разгледаме по-отблизо сървърите. Първото нещо е, че сървърът трябва да знае, че искате той да ви обслужва уеб страница. Представете си пощенска кутия (порт), която се следи за входящи писма (заявки). Това се прави от уеб сървър. Уеб сървърът чете писмото и след това изпраща отговор с уеб страница. Но когато искате да изпратите нещо, трябва да имате някакво съдържание.

 Django е нещо, което ви помага да създавате съдържанието. Kогато заявката стигне до уеб сървър, тя се предава на Django, който се опитва да разбере какво всъщност се иска. Първо взема адреса на уеб страница и се опитва да разбере какво да прави. Тази част се прави от **urlresolver** на Django (обърнете внимание, че адресът на уебсайт се нарича URL - Uniform Resource Locator - така че името *urlresolver* има смисъл). Не е много умно - взема списък от модели и се опитва да съответства на URL адреса. Django проверява шаблоните отгоре надолу и ако нещо съвпада, Django предава заявката на свързаната функция (която се нарича *view*).

* Представете си пощенски превозвач с писмо. Тя върви по улицата и проверява всеки домашен номер спрямо този на писмото. Ако съвпада, тя поставя писмото там. Ето как работи urlresolver!
* Във функцията *view* се правят всички интересни неща: можем да разгледаме база данни, за да потърсим някаква информация. Може би потребителят иска да промени нещо в данните? Като писмо, в което се казва: „Моля, променете описанието на моята работа“. *view* може да провери дали ви е позволено да го направите, след това актуализира описанието на работата за вас и изпраща обратно съобщение: „Готово!“ Тогава *view* генерира отговор и Django може да го изпрати до уеб браузъра на потребителя.

# Admin Panel (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:

След като създадете администраторски акаунт, можете да влезете в админ панела чрез браузъра си на адрес ‘[http://адрес-на-вашия-сайт/admin/](http://xn------5cdbbbnok2bo8b0bqit6h8f/admin/" \t "_blank)’ 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,)

·

# HTML

***HTML (HyperText Markup Language*)** – преведено език за маркиране на хипертекст, е език за кодиране, който казва на уеб браузъра как да показва съдържанието и структурата на уеб страницата. Както подсказва името, той се използва за форматиране или маркиране на обикновен текст с прекъсване на абзаци, получер шрифт или курсив, цветни шрифтове, списъци, таблици и много друго.

 В дефиницията на HTML „хипертекст“ означава, че текстът може да включва връзки към други ресурси в мрежата. Това е пример за текст, който не е форматиран с HTML5 код. Сега започвам да се запознавам с различните начини на форматиране: получерен шрифт, курсив, подчертан, цветен. Все още не знам как да започна нов абзац или да напиша списък от задачите си за деня:

* 1. Събуждане в 6°°.
* 2. Утринна гимнастика 15 мин.

това е пример за текст, който е форматиран с HTML5 код. Сега започвам да се запознавам с различните начини на форматиране: получерен шрифт, *курсив*, подчертан, цветен. Вече знам как да започна нов абзац или да напиша списък от задачите си за деня:

Познаването на основните тагове и използването им дава свобода на блогъра в работата му със съдържанието. Ето защо ни трябват познания по HTML: Разбирането на HTML може да ни помогне да научим как е структуриран нашият блог или уебсайт.

Познания по HTML кода ще ни помогнат да станем по-самостоятелни, по-уверени в себе си като блогъри и ще ни спести време. Ще можем да отстраняваме някои проблеми при форматирането на WordPress без нужда от странична помощ. Ще оформяме някои части на страницата си по свое желание. Ще се справяме по-лесно със SEO – оптимизацията на нашият  блог за търсачките.

# Структура на един HTML елемент

HTML елементите са ограничени от **тагове**, които могат да имат **атрибути** – например *цвят, размер*. Tаг за начало – определя началото на един HTML елемент – заграден в < и > таг за край – определя края на HTML елемента.

За разлика от началния таг, крайният има наклонена надясно черта пред името си – заграден в </ и >.В случая, когато е необходимо да се постави таг за край, не трябва да се забравя, за да се спре продължаването на определеното форматиране.

Забележка: HTML таговете са контейнерите за различните видове HTML елементи. Те ограждат всяко съдържание на уеб страницата, в противен случай браузърът няма да знае как да го представи. Meta тагове <meta> таговете дефинират метаданни за HTML документ. Метаданните са данни (информация) за данни. Те се намират винаги в елемента <head> и обикновено се използват за задаване на набор от символи, описание на страница, ключови думи, автор на документа и настройки на прозореца на визуализиране.

Особено важен за представянето в търсачката е атрибутът *name* със стойност *description*, който определя кратко описание на съдържанието на твоята уеб страница. Точно това описание най-често се използва от търсачките, за да запознае потребителя за какво се отнася статията. Таг title-Таг <title> определя заглавието на документа. Заглавието се показва в заглавната лента на браузъра или в раздела на страницата. Този таг е задължителен. Освен това е много важен за оптимизацията за търсачките (SEO).

Съдържанието на елемента *title*се показва в търсачката и участва в класирането на уеб страниците при зададено търсене по ключова дума или фраза. Заглавия-Сред HTML елементите, които трябва да знаеш на всяка цена, е списъкът със заглавия, чрез които се организира структурата на уеб страницата. Основното заглавие се отбелязва с тага <h1>…</h1>.Заглавията на раздели в публикацията са в йерархия. Те се определят с таговете <h2>…</h2>,…,<h6>…</h6>, като тези тагове не са задължителни да присъстват всичките и <h6> дефинира най-маловажното заглавие.

 Форматиране на текст- За да добавиш *курсив*, го вмести между таговете <em> и </em>. За да добавиш удебелен текст (текст, който считаш за **важен**) в HTML документ, го обгради с таговете <strong> и </strong>. За да добавиш маркиран текст, го обгради с таговете <mark> и </mark>. За дефиниране на текст под черта се използва <sub> и </sub> таг. Параграфи- Тагове <p> и </p> които означават начало и край на абзац. Браузърите автоматично добавят един празен ред преди и след всеки елемент <p>. Този HTML елемент е ключов за структурата на една уеб страница.

Вероятно не е необходимо да използваш таговете за абзац, когато пишеш статиите си, защото на повечето платформи те автоматично се вмъкват в текста, който въвеждате. Сигурно ще се запиташ, защо ти трябва да знаеш този HTML елемент, щом като платформите сами го организират.

Рано или късно нещо няма да ти хареса в оформянето на абзаците, в подравняването им. Тогава ще се наложи да действаш на ниво HTML. Таг(Pre)- Има и друга възможност за изписване на текст с вмъкнати празни редове – да се използва HTML елемента <pre>…</pre>:Тагът <pre> дефинира предварително форматиран текст (pre-formatted). Той ще се покаже точно, както е написан в изходния HTML код – с фиксирана големина на шрифта и запазвайки интервалите и прекъсванията на редове. Можеш да го използваш, за да покажеш примери за код, поезия или друг текст, в който са важни точното разстояние и прекъсванията на редовете. Основният недостатък на форматирания текст с помощта на <pre> е ширината. За разлика от нормалния HTML, той няма да се преоразмери, за да съответства на размера на прозореца на потребителя. Ще създаде хоризонтална лента за превъртане. Цитати - За означаване на текст, който е цитат, се използва елемента <blockquote>…</blockquote>. Той може да бъде форматиран по различен начин, използвайки CSS стил.

Връзки в текста (таг) <a> -Връзките в текста са изключително необходими за външни и вътрешни препратки, към различни части от същата страница, в целеви страници и дрyги. Трябва да познаваш добре таг <a>…</a>, за да вмъкваш нужните линкове. Този елемент е от особено значение за оптимизацията на публикациите за търсачката, осигурявайки чрез връзките необходимата допълнителна информация за обяснение на части от съдържанието.

 Елементът за връзка съдържа **anchor текст (котва),** състоящ се от една или повече думи, чрез който създаваш хипертекстовата връзка в документа. С този таг можеш да направиш възможно кликването върху изображение, в което да има връзка .***href*** – атрибут, който указва дестинацията на връзката;

***rel*** – атрибут, който указва връзката между текущия документ и документа, към който е връзката. Използва се само ако присъства атрибут ***href***. *target* – атрибут, който указва къде да се отвори свързаният документ. Когато му се даде стойност target="\_blank", ще отвори свързания документ в нов прозорец или раздел. Тагът <a> с атрибут href е същия като основната връзка, с изключение, че е връзка към кодова дума (след *id=*), а не към URL адрес.

 Пред кодовата дума (в случая *label5*)се поставя знак **#**, за да се обозначи, че е вътрешна връзка. Без знака **#**браузърът ще търси извън страницата тази кодова дума

 .Вмъкване на изображения-В списъка с важните за един блогър HTML тагове, трябва да бъде поставен и таг <img>, който дефинира изображение в HTML страницата. <img> няма затварящ таг.

 Използва се винаги с поне два (задължителни) атрибута:  *src* и *alt*.***src*** – атрибутът служи за локализиране и извикване на permalink на медията – адреса на изображението; ***alt***– указва алтернативен текст на картинката, в случай че не се покаже визуалното изображение;

Таг <code>- Ако ни е необходимо да вмъкнем код в текста си, това става с <code>… </code> таг, който форматира текста с подходящ шрифт и например. Като алтернатива можем да използваме таг <pre>…</pre>, който ще покаже кода по различен начин от останалия текст. Неразбиващо пространство в HTML - Понякога в HTML на уеб страницата си можем да срещнем групата символи &nbsp; – non-breaking space. Обектът &nbsp; създава неразбиващо се пространство, което се използва, когато не искаме  автоматично прекъсване на линията в тази позиция. Това е удобно, когато разбиването на думите e неприемливо.

# CSS и JavaScript

# CSS

CSS e спецификация за стилове на елементите на една уеб страница. Той позволява разработчиците да определят стилове за цветове, шрифтове, размери, позиции и други атрибути на елементите. Това позволява програмистите да създаваме  уеб страници, които са ефектни и изглеждат добре.

Има няколко начина за прилагане на CSS на една уеб страница: a). Вграждане на CSS в HTML файла: Това се прави с добавяне на CSS код в тага <style> вътре в тага <head> на HTML документа.

 Този метод е подходящ, когато искаме да приложим стиловете само за една уеб страница. Използване на външен CSS файл: Създаваме CSS файл с разширение .css и дефиниране стиловете в него. След това трябва да вмъкнем файла към HTML документа с помощта на тага <link>.

Този метод е подходящ, когато искаме да приложим стиловете за много уеб страници. Използване на inline CSS: Добавяме стиловете към style атрибутa на HTML елементите. В повечето браузъри може да инспектираме **CSS** като използваме developer tools функциите на самия browser. Най-популярните браузъри като Chrome, Firefox, Edge, и Safari имат добри инструменти за целта.

# JavaScript

**JavaScript** е програмен език, който позволява динамична промяна на поведението на браузъра в рамките на дадена **HTML**страницата. JavaScript се зарежда, интерпретира и изпълнява от уеб браузъра, който му осигурява достъп до Обектния модел на браузъра. JavaScript функции могат да се свържат със събития на страницата (например: движение/натискане на мишката, клавиатурата или елемент от страницата, и други потребителски действия). JavaScript е най-широко разпространеният език за програмиране в интернет. Прието е JavaScript програмите да се наричат скриптове. JavaScript може да влияе на почти всяка част от браузъра. Браузъра изпълнява JavaScript кода в цикъла на събития като резултат от действия на потребителя или събития в браузъра (например document.onLoad).

JavaScript кодът може да с е вмъква в HTML документа между двойката елементи <script> и</script>. Когато срещне тага <script>, браузърът разбира, че трябва да спре интерпретирането на HTML кода и да започне да обработва скрипта, намиращ се между <script> и </script>.

 Този скрипт не е задължително да бъде написан на JavaScript. Има и други езици за писане на скриптове, например VBScript. Но езикът по подразбиране е JavaScript. Другият начин да заредите JavaScript е да го поставите в отделен файл.

 Променливи в JavaScript са динамични и могат да съдържат стойности от всеки тип. Декларират се чрез ключовата дума var следвана от името на променливата или списък от променливи. Възможно е също да присвоим стойност на променливата при декларация чрез оператора =. По подразбиране стойността на променливата ще бъде undefined.

 Имената на променливите в JavaScript не трябва да започват с цифра, могат да започват с малка или голяма буква, \_ или $ и да съдържат само тези знаци както и числа. Всички останали символи са забранени. Името на променливата не може да е ключова дума или оператор. В JavaScript  има два вида обхват за променливите: функционален и глобален.

 Функционален обхват важи за всяка променлива обявена чрез var в рамките на дадена функция. Тази променлива е видима единствено в рамките на тази функция. Глобален обхват важи за всяка променлива обявена извън функция или без var, дори и във функция. Тази променлива е видима от всеки друг код на страницата.

# SOAP.

SOAP е абревиатура от "Simple Object Access Protocol" (прост протокол за достъп до обекти) и е протокол за обмен на данни в мрежата, който се използва в програмирането. SOAP е базиран на XML и използва HTTP за пренос на съобщенията.

**SOAP** позволява на приложенията да извикват функции на други приложения или услуги по мрежата, като използват стандартизиран формат на съобщенията. SOAP се използва предимно в системите за обмен на данни между компютърни системи и е често използван за разработка на услуги на основата на SOA (Service-Oriented Architecture).

SOAP се състои от няколко ключови компонента, включително:

·        **Envelope** (Обвивка) - Това е основният елемент на SOAP съобщението и съдържа информация за версията на SOAP и съдържанието на съобщението.

·         **Header** (Заглавие) - Това е опционален елемент на SOAP съобщението и съдържа допълнителна информация за съобщението, като например сигурността или управлението на транзакции.

·         **Body** (Тяло) - Това е елемент на SOAP съобщението, който съдържа реалното съдържание на съобщението.

·         **Fault** (Грешка) - Това е елемент на SOAP съобщението, който съдържа информация за грешките, които могат да възникнат по време на изпълнението на съобщението.

SOAP е стандартизиран протокол, който се използва широко в програмирането за обмен на данни между системи. Въпреки че SOAP е бил използван широко в миналото, в последно време е набелязана тенденция на преминаване към по-леки и бързи протоколи като REST.

# Jinja.

**Jinja** е мощен шаблонен език за Python, който позволява лесно и ефективно създаване на HTML, XML и други видове текстови изходи. Той е част от стандартната библиотека на фреймуърка за уеб програмиране Flask и е възможно да се използва с други уеб фреймуърци.

 Jinja използва синтаксис, който наподобява на този на Django шаблоните. Един от основните принципи на Jinja е този да бъде "непрекъснат", т.е. няма да има код в шаблона, който да не може да се компилира и изпълни. Използваме фигурните скоби **{{ }}** за да обозначим местата, където ще се поставят стойностите на променливите. В този случай променливите са **title**, **name** и **date**.

 Ето някои от основните функции на Jinja:

* Условно изпълнение на код чрез конструкцията if-elif-else;
* Итерация през обекти чрез конструкцията for-in;
* Филтриране на данни чрез вградени филтри или допълнителни филтри, дефинирани от потребителя;
* Използване на макроси за повтаряеми блокове код;
* Използване на наследяване на шаблони за реализиране на шаблони на различни нива на вложеност;
* Интегриране на Python код в шаблоните чрез конструкцията {{ ... }} или {% ... %}.
* Използваме цикъл **for-in** за да изпълним блока код за всеки елемент в списъка **users**, и печатаме съобщение, което съдържа името на всеки потребител.

# XAMPP:

XAMPP е крос-платформен уеб сървър, който е безплатен и с отворен код. XAMPP е кратка форма за крос-платформа, Apache, MySQL, PHP и Perl. XAMPP е популярен междуплатформен уеб сървър, който позволява на програмистите да пишат и тестват своя код на локален уеб сървър.

 Той е създаден от Apache Friends и обществеността може да ревизира или модифицира неговия собствен изходен код. Той включва MariaDB, Apache HTTP сървър и интерпретатори за PHP и Perl, наред с други компютърни езици. Поради простотата на разгръщане на XAMPP, разработчикът може бързо и лесно да инсталира WAMP или LAMP стек на операционна система, с допълнителното предимство, че могат да се зареждат и общи приложения за добавки като WordPress и Joomla.

# **Нужда от XAMPP**:

·         XAMPP е просто локален хост или сървър.

·         Този локален сървър работи на вашия персонален компютър, независимо дали е настолен или лаптоп.

·         Използва се за тестване на клиенти или уебсайтове, преди да ги публикува на отдалечен уеб сървър.

* ·         На локален компютър сървърният софтуер XAMPP осигурява подходяща среда за тестване на MYSQL, PHP, Apache и Perl проекти. Тъй като повечето внедрявания на уеб сървъри в реалния свят споделят същите компоненти като XAMPP, преминаването от локален тестов сървър към сървър на живо е лесно.

# Предимства на XAMPP:

* В сравнение с други уеб сървъри като WAMP, той е лесен за настройка.
* Той е Multi Cross-Platform, което предполага, че работи както на Windows, така и на Linux.
* С една команда можете да стартирате и спирате целия стек на уеб сървъра и базата данни.
* Налични са както пълна, така и стандартна версия на XAMPP.
* Той има контролен панел, който можете да видите, че съдържа бутони за стартиране и спиране за специфични механизми, като например Apache, който работи през своя контролен панел.

Той също така включва OpenSSL, phpMyAdmin, MediaWiki, Joomla, WordPress и много допълнителни модули.

# Технологии

# DRF

Библиотека 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.

 DRF е съкращение от Django REST framework, което е библиотека за създаване на RESTful API (API, базиран на REST архитектурата) с помощта на Django. Това е мощен инструмент, който позволява на разработчиците да създават уеб услуги с лесното използване на HTTP протокола.

 Django REST framework (DRF) предоставя много инструменти и функции за създаване на уеб API. Тези инструменти включват сериализация на данни, валидация на данни, автоматично генериране на документация, авторизация и управление на потребители, аутентикация и много други. DRF също така предлага много вградени изгледи (views), които покриват много от общите изисквания на уеб услуги.

 DRF е много популярен в областта на създаването на API и се използва от много големи уеб сайтове. Той предлага много гъвкавост и се интегрира перфектно с Django, което го прави много лесен за употреба.

 В заключение, Django REST framework е мощна библиотека за създаване на RESTful API в Python, която предлага много инструменти и функции за лесна и ефективна уеб разработка. DRF е изключително популярен сред уеб разработчиците и се използва от много големи уеб сайтове като Mozilla, Dropbox, Red Hat и много други.

 DRF предоставя множество изгледи (views), които покриват много от общите изисквания на уеб услуги. Тези изгледи включват готови имплементации за обработка на HTTP заявки, като GET, POST, PUT и DELETE. DRF също предлага много готови модули, като например Pagination, Filtering, Ordering и много други, които правят създаването на уеб услуги много по-лесно.

# MySQL

**MySQL** е многопоточна, многопотребителска,SQL система за управление на бази данни (СУБД) с повече от шест милиона инсталации. My SQL AB разпространява GNU e MySQL като свободен софтуер под General Public Liens (GPL), но също така под традиционните за комерсиален софтуер лицензи за случаи, когато използването е несъвместимо с GPL. Подобно разпространение е известно като двойно лицензиране.

**MySQL** се разработва, разпространява и поддържа от шведската компания MySQL AB, която държи авторските права за голяма част от програмния код. Подобен е моделът на JBoss, а също и начинът, по който Free Software Foundation работи с авторските права по нейните проекти в отличие от Apache проекта, където софтуерът е обществено разработван, а авторските права за програмния код принадлежат на индивидуалните автори.

Има 3 типа подборки на MySQL сървър:

**Стандарна:** MySQL-стандартна е препоръчителна за повечето от потребителите и включва InnoDB хранилище.

**Максимална** (Max): (това не е MaxDB, която е разработвана в сътрудничество със SAP) mysqld-max Extended MySQL Server. MySQL-максимална включва допълнителни възможности, които могат да не бъдат достатъчно тествани и не са необходими обикновено.

**MySQL-Debug** е компилирана по начин по който се улеснява трасировката на грешки, но не е предназначена за повседневно използване, защото наличието на допънителни инструкции може да намали производителността.

# Бази данни

 База данни представлява колекция от логически свързани данни в конкретна предметна област, които са структурирани по определен начин. В първоначалния смисъл на понятието, използван в компютърната индустрия, базата от данни се състои от записи, подредени систематично, така че компютърна програма да може да извлича информация по зададени критерии.

 Система за управление на бази данни е компютърно приложение ([софтуер](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80)) създадено за комуникация между потребителя, други приложения, както и други БД, с цел да се сравнят и анализират данни. Общото специфично предназначение на СУБД е да позволи определянето, създаването, заявки, актуализацията и администрирането на бази данни. Добре известни СУБД включват MySQL, PostgreSQL, Microsoft SQL Server, Oracle, Sybase, SAP HANA, и IBM DB2.

 Бази данни не са съвместими с различните СУБД, за това различните СУБД работят със стандартни като SQL и ODBC или JDBC, за да позволи на всяко приложение да работи с различни СУБД, а така и с различни БД. Управлението на БД често се избира от модела им, които те подкрепят. Най-използвани системи от бази данни от 1980 г. насам са всички поддържани релационния модели на езика SQL. Често срещано е СУБД да се нарича само „база данни“.

 „База данни“ дефинира множество свързани данни и начинът, по който са организирани. Достъпът до тези данни обикновено се осигурява чрез „система за управление на база данни“ (СУБД), състояща се от интегриран набор от компютърен софтуер, който позволява на потребителите да взаимодействат с една или повече бази данни и осигурява достъп до всички данни, съдържащи се в базата данни (въпреки че може да има ограничения спрямо достъпа до точно определени данни). СУБД предоставя различни функции, които позволяват влизане, съхранение и извличане на огромни количества информация и осигурява начини за управление как точно да бъде организирана тази информация.

 Съществуващите Системи за управление на бази данни осигуряват различни функции, които позволяват управлението на базата данни и самите данни, които могат да бъдат класифицирани в четири основни функционални групи:

* Дефиниране на данни – Създаване, модифициране и премахване на дефинициите, които определят организацията на данните.
* Промяна – Вмъкване, модификация и заличаване на актуалните данни.
* Извличане – Предоставяне на информация във формата на пряко използваем или за по-нататъшна обработка от други приложения. Извлечените данни могат да се предоставят направо в същата форма, в която са били съхранени в базата данни или в нова форма, получена чрез промяна или комбиниране на съществуващи данни от базата данни.
* Администриране – Регистриране и наблюдение на потребителите, налагане на сигурността на данните, наблюдение на изпълнението, запазвайки целостта на данните, които се занимават с едновременния контрол и възстановяване на информацията, която е била повредена от някакво събитие като например неочакван срив на системата.

Физически сървърите на базите данни са специализирани компютри, които притежават най-актуалните бази данни и работят само на СУБД и съответния софтуер. Сървърите на базите данни обикновено са многопроцесорни компютри с огромни памети и RAID дискови масиви, използвани за стабилно съхранение.

# PyCharm

PyCharm е популярна интегрирана среда за разработка (IDE) за програмиране на Python, която е разработена от JetBrains. Тя предлага богат набор от функции за улесняване на процеса на писане на код, от откриване на грешки до автоматично допълване на кода и интеграция с различни инструменти за управление на проекти. PyCharm се използва широко от програмисти за разработка на Python приложения.

**PyCharm** предоставя широк спектър от функционалности за удобно и ефективно програмиране на Python. Тези функции включват автоматично допълване на кода, интелигентно подсказване, интеграция с виртуални среди за разработка, управление на проекти, системи за контрол на версиите и инструменти за отстраняване на грешки. Също така, PyCharm предлага поддръжка на различни фреймуърци и библиотеки за Python, като Django, Flask, и други, което го прави подходящо за разработка на разнообразни типове приложения.

1. Въведение в PyCharm: Кратка информация за PyCharm като IDE за разработка на Python.

2. Основни функционалности: Обяснение на основните функции, като автоматично допълване на кода, интелигентно подсказване, управление на проекти, интеграция с виртуални среди и др.

3. Инструменти за разработка: Представяне на инструментите за тестване, профилиране, рефакторинг и управление на бази данни, които са вградени в PyCharm.

4. Поддръжка на фреймуърци и библиотеки: Информация за възможностите на PyCharm за работа с различни фреймуърци и библиотеки за Python, като Django, Flask и други.

5. Версии и цени: Обяснение на различните версии на PyCharm (Community и Professional) и техните функции, както и информация за цените им.

6. Заключение: Подчертаване на важността на PyCharm за разработчиците на Python и призив за използване на IDE за подобряване на продуктивността при разработка на софтуер.

# Реализация

# Анализ на заданието

Заданието за дипломния проект предполага разработване на WEB-базирано приложение за здраве и фитнес за проследяване на потреблението на потребителите и да изчислявате общите калории.

Приложението трябва да разполага най малко със следните публично достъпни секции: „Начало“, „Калории за деня“ и „История“. Интерфейсът на приложението трябва да позволява лесно въвеждане на консумираните от потребителя храни и да представя текущото потребление в нагледен графичен вид.

# Архитектуран на приложението

Със сигурност има различни начини, средства и технологии за решаване на поставената задача. Но моят избор е Python и Django.

Изборът на Python и Django всъщност избор на backend технология за решаване на поставената задача. Логино е цялата логика на приложението да бъде изнесена на сървъра (в backend-а), а за комуникация да се използва механизма на работа с форми, който е заложен в Django.

# Модели

Django предоставя готов механизъм за работа с БД. Това е т.н. ORM – механизъм за представяне на таблиците чрез класове. Дефинирането на таблиците от БД се прави във файла models.py. В конкретния случай моделите са два – за храните и за консумацията:

from django.db import models  
from django.contrib.auth.models import User  
*# Create your models here.*class Food(models.Model):  
  
 def \_\_str\_\_(self):  
 return self.name  
  
 name = models.CharField(max\_length=100)  
 carbs = models.FloatField()  
 protein = models.FloatField()  
 fats = models.FloatField()  
 calories = models.IntegerField()  
  
  
class Consume(models.Model):  
 user = models.ForeignKey(User, on\_delete=models.CASCADE)  
 food\_consumed = models.ForeignKey(Food, on\_delete=models.CASCADE)

# Изгледи

При изпращане на данни от формата, те се обработват от съответния изглед във файла views.py:

from django.shortcuts import render, redirect  
from .models import Food, Consume  
*# Create your views here.*def index(request):  
  
 if request.method == "POST":  
 food\_consumed = request.POST['food\_consumed']  
 consume = Food.objects.get(name=food\_consumed)  
 user = request.user  
 consume = Consume(user=user, food\_consumed=consume)  
 consume.save()  
 foods = Food.objects.all()  
  
 else:  
 foods = Food.objects.all()  
 consumed\_food = Consume.objects.filter(user=request.user)  
  
 return render(request, 'myapp/index.html', {'foods': foods, 'consumed\_food': consumed\_food})  
  
  
def delete\_consume(request, id):  
 consumed\_food = Consume.objects.get(id=id)  
 if request.method == 'POST':  
 consumed\_food.delete()  
 return redirect('/')  
 return render(request, 'myapp/delete.html')

# Шаблони

За да се визуализира описаното до тук е нужен HTML шаблон. Той играе ролята на визуален контейнерк, който пренася информацията до потребителя. В случая са разработени два шаблона – един основен и един за изтриване на елент:

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
  
 <head>  
 <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"  
 integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">  
 <script src="https://cdn.jsdelivr.net/npm/chart.js@2.9.3/dist/Chart.min.js"></script>  
 </head>  
 <div class="container">  
 <div class="row">  
 <div class="col-md-12">  
 <nav class="navbar navbar-dark bg-info">  
 <a href="{% url 'index' %}" class="navbar-brand">Home</a>  
 </nav>  
 </div>  
 </div>  
 </div>  
</head>  
  
<body>  
 <div class="container"></div>  
 <div class="row m-5">  
 <div class="col-md-12">  
 <form method="POST">  
  
 {% csrf\_token %}  
 <h2 style="text-align:center">Are you sure you want to delete this item? <br> <button type="submit"  
 class="btn btn-danger">Delete</button></h2>  
  
 </form>  
 </div>  
 </div>  
 </div>  
  
</body>  
  
</html>

# Пътища

И последно, но много важно – проследяването на заявките. То се извършва във файла urls.py. Както личи и от името на файла, става дума за проследяване на URL-и:

from django.contrib import admin  
from django.urls import path  
from myapp import views  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', views.index, name="index"),  
 path('delete/<int:id>/', views.delete\_consume, name="delete"),

# Заключение

Приложението за здраве и фитнес за проследяване на потреблението на потребителите и за изчислявне на общите калории, направено в Django, помага лесно да контролирате консумираните от вас калории. Ако бъде качено на реален сървър, всеки ще може да контролира хранителния си режим в реално време

Макар че приложението не е голямо и особено сложно, то е напълно фунционално. Може да се качи на реален сървър като самостоятелно приложение. Още по-подходящо би било да се добави като модул към други информационни и/или специализирани сайтове.

# 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/