Съдържание

[1 Увод: 3](#_Toc164736782)

[2 Python 3](#_Toc164736783)

[*2.1* *Приложение на Python* 3](#_Toc164736784)

[2.2 *Защо Python е толкова широко разпространен* 4](#_Toc164736785)

[*2.3* *Използване на Python* 4](#_Toc164736786)

[2.3.1 Разработки с Python 5](#_Toc164736787)

[2.3.2 Софтуерно тестване и прототипиране: 5](#_Toc164736788)

[2.3.3 Ежедневни задачи: 5](#_Toc164736789)

[2.3.4 Автоматизация или скриптове: 6](#_Toc164736790)

[2.3.5 Уеб разработка: 6](#_Toc164736791)

[2.4 Анализ на данни и машинно обучение: 6](#_Toc164736792)

[*2.5* *Python обработва файлове* 7](#_Toc164736793)

[2.5.1 Потребителски интерфейси на Python 7](#_Toc164736794)

[2.5.2 Налични платформи: 8](#_Toc164736795)

[*2.6* *Рython борави с файлове* 10](#_Toc164736796)

[*3* *Алгоритми* 10](#_Toc164736797)

[3.1 Представяне: 13](#_Toc164736798)

[3.1.1 Описание от високо ниво: 13](#_Toc164736799)

[3.1.2 Описание на реализацията: 14](#_Toc164736800)

[3.1.3 Формално описание: 14](#_Toc164736801)

[4 PyCharm: 14](#_Toc164736802)

[*4.1* *Характеристики:* 15](#_Toc164736803)

[4.1.1 Pycharm: среда за разработка на Python: 15](#_Toc164736804)

[4.1.2 Python рефакторинг: 16](#_Toc164736805)

[*4.2* *Инсталиране на PyCharm на Linux* 16](#_Toc164736806)

[4.2.1 Първоначална настройка на Pycharm: 16](#_Toc164736807)

[5 Изготвяне на автобиография 17](#_Toc164736808)

[5.1 Формат на автобиографията: 18](#_Toc164736809)

[5.1.1 Лични данни: 18](#_Toc164736810)

[5.1.2 Образование и квалификации 19](#_Toc164736811)

[5.1.3 Постижения 20](#_Toc164736812)

[5.1.4 Допълнителни квалификации 20](#_Toc164736813)

[5.1.5 Професионален опит 21](#_Toc164736814)

[5.1.6 Езикови и технически умения 22](#_Toc164736815)

[*6* *Използвани библиотеки за създаване на програмата:* 22](#_Toc164736816)

[*6.1* *CV2* 22](#_Toc164736817)

[*6.2* *Pyautogui* 23](#_Toc164736818)

[*6.3* *Сейфове:* 25](#_Toc164736819)

[7 Реализация 26](#_Toc164736820)

[7.1 Анализ на заданието 26](#_Toc164736821)

[7.2 Архитектуран на приложението 26](#_Toc164736822)

[7.3 Реализация 26](#_Toc164736823)

[7.3.1 Формата 26](#_Toc164736824)

[7.3.2 Изглед 27](#_Toc164736825)

[7.3.3 Шаблон 28](#_Toc164736826)

[7.3.4 Пътища 29](#_Toc164736827)

[8 Заключение 29](#_Toc164736828)

[9 Списък на използваната литература: 31](#_Toc164736829)

# Увод:

В днешната цифрова ера, където информацията е ключов елемент от нашия живот, автобиографията е важен начин за запазване и споделяне на лични и професионални истории. Създаването на автобиография обаче може да бъде труден процес, изискващ време и усилия. За да улесни този процес и да го направи по-достъпен за всеки, в този документ се разглежда разработването на приложение на Python, което позволява на потребителите да създават автобиографии чрез динамично генерирани формуляри и генериране на PDF файлове.

Приложението е с интуитивен интерфейс и предлага лесен начин за съставяне на автобиографии, като използва съвременни технологии и инструменти. Чрез динамично генериране на формуляри и автоматично създаване на PDF файлове, приложението осигурява удобен и ефективен начин за съставяне на професионални автобиографии, които могат да бъдат използвани за различни цели като, например, работно кандидатстване.

# Python

Python е език за програмиране от високо ниво, интерпретативен, интерактивен, обектно ориентиран, създаден от Гуидо ван Росум в началото на 90-те години. Кръстен е на телевизионното шоу на BBC Monty Python’s Flying Circus. Често бива сравняван с Tcl, Perl, Scheme, Java и Ruby. Python има сходства с езици като Perl, C, C++ и Java. Той е език за програмиране с общо предназначение. Други негови отличителни черти са, че е обектно-ориентиран, интерпретируем и интерактивен. „Интерпретируем“ означава, че за да изпробвате даден код дали работи, не е нужно да го компилирате преди това. „Обектно-ориентиран“ – че Python поддържа технологията на програмиране, при която кода се обособява в групи, наречени „обекти“, които взаимодействат помежду си.

## Приложение на Python

В наши дни този език се прилага навсякъде: от 3D рендериране във филмовата или игралната индустрия, управление и автоматизация на сървъри, дори и програми за други операционни системи като Linux OS. Други приложения като Drop Box, Bit Torrent, Blender също са написани на Python. За да направят софтуера си по-бърз, редица компании използват Python. Facebook и NASA са две от най-популярните имена, които използват Python широко в своята дейност. Python e често използван за разработка на уебсайт – софтуерната рамка на Python за разработка на уеб сайтове – “Django”, е една от най използваните в наши дни. Или може би желаете да разработвате игри? Тогава кросплатформения набор от Python модули (Pygame) е това, от което имате нужда. Ако искате да разработвате заглавия за мобилни платформи, то тогава имате на разположение библиотеката с отворен код Kivy. Python намира приложение и в машинното обучение (machine learning), съответно и в разработките на изкуствен интелект.

## Защо Python е толкова широко разпространен

Най-вече защото кодът на Python има висока степен на четимост (т.е. почти идентичен с нормалния английски език), което го прави достъпен за много хора. Привличането на голям брой хора, които да го използват от своя страна увеличава способностите на езика, увеличава ресурсите, които са налични, увеличава проблемите, които могат да се решават с този език. Но най-вече Python никога не е зависел от платформата, за която се разработват приложенията. Дори да се върнем назад във времето и тръгнем към наши дни, можем да видим за колко различни платформи съществуват приложения, писани на Python: Linux, Window, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, Z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE.

## Използване на Python

Python обикновено се използва за разработване на уебсайтове и софтуер, автоматизация на задачи, анализ на данни и визуализация на данни. Тъй като е относително лесен за научаване, Python е възприет от много непрограмисти, като счетоводители и учени, за различни ежедневни задачи, като организиране на финанси. „Писането на програми е много креативна и възнаграждаваща дейност“, казва инструкторът от Мичиганския университет и Coursera Чарлз Р. Северънс в книгата си „ Python for Everybody“. „Можете да пишете програми по много причини, вариращи от изкарване на прехраната ви до решаване на труден проблем с анализ на данни до забавление до помагане на някой друг да реши проблем.“

### ****Разработки с Python****

Някои от разработките включват:

• Анализ на данни и машинно обучение

• Уеб разработка

• Автоматизация или скриптове

• Софтуерно тестване и прототипиране

• Ежедневни задачи

### ****Софтуерно тестване и прототипиране:****

При разработването на софтуер Python може да помогне при задачи като контрол на изграждането, проследяване на грешки и тестване. С Python разработчиците на софтуер могат да автоматизират тестването за нови продукти или функции. Някои инструменти на Python, използвани за тестване на софтуер, включват Green и Requestium.

### ****Ежедневни задачи:****

Python не е само за програмисти и специалисти по данни. Изучаването на Python може да отвори нови възможности за тези в професии с по-малко данни, като журналисти, собственици на малък бизнес или специалисти по маркетинг в социални медии. Python може също да позволи на непрограмисти да опростят определени задачи в живота си. Ето само няколко от задачите, които можете да автоматизирате с Python:

• Следене цените на фондовия пазар или крипто;

• Изпращане на текстово напомняне да носите чадър винаги, когато вали;

• Актуализиране списъка за пазаруване на хранителни стоки;

• Преименуване на големи пакети от файлове;

• Преобразуване на текстови файлове в електронни таблици;

• На случаен принцип възлагане задължения на членове на семейството;

• Попълване онлайн формуляри автоматично.

### ****Автоматизация или скриптове:****

Ако установим, че изпълняваме задача многократно, можем да работим по-ефективно, като я автоматизираме с Python. Писането на код, използван за изграждане на тези автоматизирани процеси, се нарича скриптове. В света на кодирането автоматизацията може да се използва за проверка за грешки в множество файлове, конвертиране на файлове, изпълнение на проста математика и премахване на дубликати в данни. Python може да се използва дори от сравнително начинаещи за автоматизиране на прости задачи на компютъра - като преименуване на файлове, намиране и изтегляне на онлайн съдържание или изпращане на имейли или текстове на желани интервали.

### ****Уеб разработка:****

Python често се използва за разработване на задната част на уебсайт или приложение - частите, които потребителят не вижда. Ролята на Python в уеб разработката може да включва изпращане на данни до и от сървъри, обработка на данни и комуникация с бази данни, URL маршрутизиране и гарантиране на сигурност. Python предлага няколко рамки за уеб разработка. Често използваните включват Django и Flask. Някои работни места за уеб разработка, които използват Python, включват back end инженери, full stack инженери, Python разработчици, софтуерни инженери и DevOps инженери.

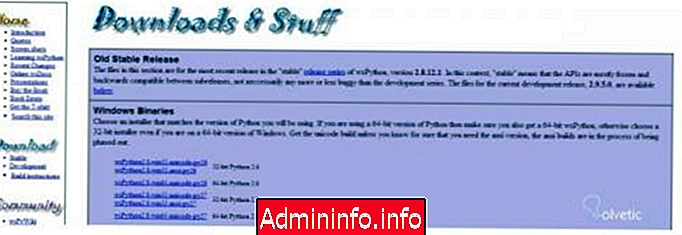
## ****Анализ на данни и машинно обучение:****

Python се превърна в основен елемент в науката за данни, позволявайки на анализатори на данни и други професионалисти да използват езика за извършване на сложни статистически изчисления, създаване на визуализации на данни, изграждане на алгоритми за машинно обучение, манипулиране и анализ на данни и изпълнение на други задачи, свързани с данни. Python може да създава широк набор от различни визуализации на данни, като линейни и лентови графики, кръгови диаграми, хистограми и 3D графики. Python също има редица библиотеки, които позволяват на програмистите да пишат програми за анализ на данни и машинно обучение по-бързо и ефективно, като TensorFlow и Keras.

## Python обработва файлове

Ако разработваме голямо софтуерно приложение, където се обработват голям брой данни, тогава не можем да очакваме тези данни да се съхраняват в променлива, тъй като променливите са нестабилни по своята същност. Следователно когато се справяме с подобни ситуации, ролята на файловете ще се появи в картината. Тъй като файловете са енергонезависими по своята същност, данните ще се съхраняват постоянно във вторично устройство.

### Потребителски интерфейси на Python



Програма или приложение не винаги трябва да бъде нещо, което се изпълнява самостоятелно, всъщност много пъти изпълняваме програми, които трябва да се използват от хората, тези хора, които наричаме потребители; нашата програма може да е най-добрата и най-напредналата в света по своята логика и поведение, но ако е ориентирана да бъде за използване на потребител, начинът, по който може да взаимодейства със споменатата програма, е почти толкова или може би по-важен от самата логика. Поради това Python предлага възможност за използване на инструменти за изграждане на потребителски интерфейси, което ни позволява да дадем графичен смисъл на нашата програма, да организираме елементите, така че трета страна да може да я използва.

### Налични платформи:

Има много платформи за разработване на графични потребителски интерфейси в Python, тази, която избираме, ще бъде добре, стига да знаем защо го правим, нито една не е по-добра от друга, те са просто различни начини за постигане на цел.

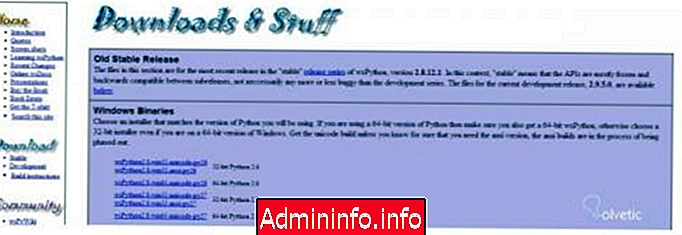
Нека видим някои от възможностите, които са на разположение, като поясним, че те не са всички, които съществуват, може да има и със сигурност ще има хиляди опции, но те могат да послужат като основа за започване от някакъв момент.

Тези инструменти се наричат инструменти, нека видим списъка:

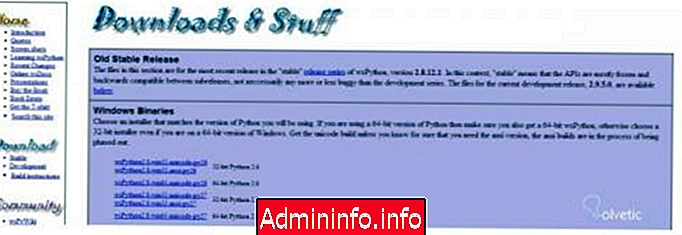
* Tkinter: Използва платформа TK. Има незабавна наличност. Тя е полустандартна.
* wxPython: Въз основа на wx Windows . Популярността му нараства много бързо.
* PythonWin: Достъпно само за Windows. Той използва родните възможности на средата на графичния интерфейс на Windows.
* Java Swing: Достъпно само за Jython . Той използва възможностите на родния графичен интерфейс на Java.
* PyGTK: Използва платформата GTK. Много популярен в Linux среди.
* PyQt: Използва Qt платформата. Той е особено популярен в Linux платформи.
* В случая ще използваме wxPython като демонстрация, за това ще го изтеглим от неговата уеб страница, достъпна на URL адрес //wxpython.org/download.php.

Важно е да изтеглим версията, която е направена за версията Python, върху която работим, за да не се появяват грешки.

На следващото изображение можем да видим версиите и как изглежда страницата за изтегляне на този инструментариум:



Веднъж изтеглени и конфигурирани, можем да преминем към следващата част.



Създаване на прозорец и неговите компоненти

Windows, известни също като рамки, са екземпляри от класа wx.Frames и ще бъдат съдържащата база на нашия интерфейс, нека видим един много основен примерен код, който можем да използваме като база, за да разберем работата на това, което наистина прави инструментариумът.

Импортиране на приложението :

w = gw.getWindowsWithTitle(window\_name)[0]  
w.activate()

Стартираме кода, като импортираме wx, който е нашият инструментариум, след това правим два екземпляра един за приложението и друг за рамката, която ще бъде контейнерът, след което извикваме метода Show (), който ще покаже прозореца и приложението ще бъде вътре; Разбира се всичко това е празно, тъй като не сме направили нищо, но когато стартираме нашата програма, ще видим следното:

Както виждаме, това е доста просто и дава различен аспект на това, което сме свикнали да виждаме в Python с командната конзола.

Това е всичко за тази първа част, в следващата ще задълбочим малко повече в тази тема с повече практически примери.

## Рython борави с файлове

Да вземем пример за това как нормалните хора ще се справят с файловете. Ако искаме да прочетем данните от файл или да запишем данните във файл, тогава първо ще отворим файла или ще създадем нов файл, ако файлът не съществува и след това ще изпълним нормалните операции за четене / запис , запис на файла и затваряне. По същия начин ние правим същите операции в python, използвайки някои вградени методи или функции.

# Алгоритми

Алгоритъм (от името на учения ал–Хорезми) е термин от математиката, информатиката, лингвистиката и други области, с който се описва сложно действие чрез редица от елементарни действия, които изпълняващият може да извърши в последователни стъпки без допълнителни обяснения. Обикновено изпълнението на алгоритъма включва изчисление или обработка на данни. По-строго дефинирано, алгоритъмът е ефективен метод за изчисляване на функция, който може да бъде изразен в рамките на крайно време и пространство и чрез добре дефиниран формален език. Започвайки от начално състояние и входни данни (понякога празни), инструкциите описват пресмятания, чието изпълнение преминава през краен брой добре дефинирани последователни състояния и завършва с крайно състояние, като в процеса се получават крайни резултати. Не е задължително преходът между състоянията да е детерминиран (еднозначно определен): някои алгоритми, известни като вероятностни алгоритми, съдържат елемент на случайност. Концепцията за алгоритмите съществува от векове, но частичното формулиране на понятието започва с опитите да се реши 10-ия проблем на Хилберт – „Задача за разрешимост на диофантово уравнение“, поставен от Давид Хилберт през 1900 година на Втория световен конгрес по математика в Париж. Последващите формулировки, при които се цели дефинирането на „ефективна изчислимост“или „ефективен метод“, включват рекурсивните функции на Ербран-Гьодел-Клини от 1930, 1934 и 1935 година, ламбда смятането на Алонсо Чърч от 1936 година, „Формулировка 1“ на Емил Пост от 1936 година и машината на Тюринг от 1936 – 1937 и 1939 година. Създаването на формална дефиниция на алгоритъм, съответстваща на интуитивното понятие, остава отворен въпрос и в наши дни. При все че няма общоприета формална дефиниция на алгоритъм, неформално понятието може да се определи като „набор от правила, които точно дефинират някаква поредица от операции“. Това определение обхваща всички компютърни програми, включително тези, които не извършват числени изчисления, стига те да прекратяват работа след краен брой операции. Класически пример за „алгоритъм“ е алгоритъмът на Евклид чрез изваждане за намиране на най-големия общ делител (НОД) на две цели числа, по-големи от 1. При него се изпълнява следната поредица от стъпки: На стъпка i, се дели X на Y и остатъкът се означава с R. Ако R = 0, резултатът на задачата е Y. В противен случай, резултатът съвпада с НОД на числата R и Y. Такъв алгоритъм, при който за намирането на решение е необходимо да бъде решена аналогична, но „по-малка“ задача, се нарича рекурсивен. Американските учени Джордж Булос и Ричард Джефри дават следното неформално определение за алгоритъм в своя класически учебник по математическа логика:

Никое човешко същество не може да пише достатъчно бързо или достатъчно дълго, или достатъчно дребно, за да изброи всички елементи на дадено изброимо безкрайно множество, изписвайки техните имена, едно след друго, с една и съща нотация. Но хората могат да направят нещо също толкова полезно в случая с някои изброими безкрайни множества – те могат да дадат експлицитни указания за определяне на n-тия елемент на множеството за произволно крайно n. Такива указания могат да бъдат дадени съвсем експлицитно, във форма, в която могат да бъдат изпълнени от изчислителна машина или от човек, който може да извършва само съвсем елементарни операции със символи. Изброимо безкрайно множество е такова множество, за чиито елементи може да се дефинира биективно съответствие с множеството на естествените числа. Така според Булос и Джефри алгоритъмът съдържа инструкции за процес, който „създава“ резултатни цели числа от произволно входно цяло число (или входни числа), което на теория може да бъде произволно голямо. По този начин алгоритъмът може да бъде алгебрично уравнение, като y = m + n – две произволни „входни променливи“ m и n, които създават резултат y. В същото време опитите на различни автори да дефинират понятието показват, че под „алгоритъм“ се разбира много повече от това:

Точни указания (на език, разбираем за „компютъра“) за бърз, ефективен, „добър“ процес, който специфицира „движенията“ на „компютъра“ (машина или човек, оборудван с необходимата вътрешна информация и способности), за да намери, декодира и обработи произволни входни числа/символи m е n, символи + и = ... и „ефективно“ да произведе, за „разумно“ време, краен резултат y на определено място и в определен формат. Концепцията за „алгоритъм“ се използва и за дефиниране на понятието решимост, което играе централна роля за обяснението как формалните системи възникват от малък набор аксиоми и правила. В логиката времето, необходимо за изпълнение на даден алгоритъм, не може да се измери, тъй като то не е свързано с обичайното физично измерение. Подобни несигурности, които характеризират текущите изследвания в тази област, водят до липсата на дефиниция за „алгоритъм“, която да съответства както на конкретната, така и на абстрактната употреба на термина. Алгоритмите са от съществено значение за начините, по които компютрите обработват информацията. Много компютърни програми съдържат алгоритми, които определят специфични инструкции, които компютърът трябва да изпълни в строго определен ред, за да се реши дадена задача. В този смисъл, алгоритъмът може да се разглежда като произволна поредица от операции, които могат да се симулират от пълна по Тюринг система. Към този възглед се придържат автори като Мински, Савидж и Гуревич:

Но ние също така ще твърдим, като Тюринг..., че всяка процедура, която може „естествено“ да бъде наречена ефективна, всъщност може да бъде реализирана от (проста) машина. Макар че това може да изглежда крайно, аргументите... в полза на това е трудно да се отхвърлят. Неформалният аргумент на Тюринг в полза на неговата теза доказва и по-силно твърдение: всеки алгоритъм може да бъде симулиран от машина на Тюринг... според Савидж [1987], алгоритъм е изчислителен процес, дефиниран от машина на Тюринг. Обичайно, когато един алгоритъм е свързан с обработка на информация, данните се четат от входа, изпращат се на изхода и/или се съхраняват за по-нататъшна обработка. Съхранените данни се разглеждат като част от вътрешното състояние на обекта, изпълняващ алгоритъма. На практика, състоянието се пази в една или повече структури от данни.

За всеки подобен изчислителен процес трябва строго да се дефинира един алгоритъм, като се определи така, че да е приложим при всички възможни обстоятелства, които могат да възникнат. Това ще рече, че систематично трябва да се разгледат всички условни разклонения, случай по случай, като критериите за всеки от случаите трябва да са ясно дефинирани и изчислими.

Тъй като алгоритъмът е точно определен списък от точно определени стъпки, редът на изчислението им винаги е от критично значение за работата на алгоритъма. Обикновено се предполага, че инструкциите са изрично изброени и са описани отначало-докрай, така както се изобразява една блок-схема. Това разбиране за формализацията на алгоритъма е основано на принципите на императивното програмиране. Други алтернативни концепции за алгоритмите предоставят функционалното и логическото програмиране.

## Представяне:

Алгоритмите могат да се представят с много различни видове нотация, в това число естествени езици, псевдокод, блок-схеми или програмни езици. Описанието на алгоритми на естествен език страда от обичайната склонност на езика към многословие и многосмисленост и поради това рядко се използва за формулирането на сложни алгоритми. Псевдокодът и блок-схемите представляват структурирани начини за изразяване на алгоритми, които избягват двусмислиците на естествения език и са независими от конкретния програмен език, на който алгоритмите се реализират. Програмните езици са главно насочени към изразяването на алгоритми в изпълним от компютър вид, но често се ползват и за да онагледяват, дефинират или документират алгоритмите.

Има голямо разнообразие от начини за представяне на алгоритмите – дадена програма за машина на Тюринг може да се опише като поредица от машинни таблици, като блок-схема, като рудиментарен машинен код или асемблерен код. Представянията на алгоритмите могат да се класифицират в три нива на описание на машината на Тюринг:

### Описание от високо ниво:

„...текст за описване на алгоритъм, игнориращ подробностите на реализацията му. На това ниво няма нужда да се отбелязва как машината управлява своята памет или глава.“

### Описание на реализацията:

„...текст, използван за дефиниране на начина, по който машината на Тюринг използва своята глава, и начина, по който съхранява данните в паметта си. На това ниво не се дават подробности за състоянията или преходните функции.“

### Формално описание:

Най-подробното, „на най-ниско ниво“, задава „таблица на състоянията“ на машината на Тюринг.

# PyCharm:

PyCharm е интегрирана среда за разработка (IDE), използвана за програмиране в Python . Той осигурява анализ на кода, графичен дебъгер, интегриран модулен тестер, интеграция със системи за контрол на версиите и поддържа уеб разработка с Django . PyCharm е разработен от чешката компания JetBrains. Той е междуплатформен , работи на Microsoft Windows , macOS и Linux . PyCharm има професионално издание, издадено под патентован лиценз и издание на общността, пуснато под лиценза на Apache. PyCharm Community Edition е по-малко обширно от Professional Edition.

## Характеристики:

• Помощ при кодиране и анализ , с довършване на код , осветяване на синтаксис и грешки , интегриране на Linter и бързи поправки

• Навигация по проекти и кодове: специализирани изгледи на проекти, изгледи на файлова структура и бързо прескачане между файлове, класове , методи и употреби

• Рефакторинг на Python код : включително преименуване, извличане на метод, въвеждане на променлива, въвеждане на константа, изтегляне нагоре, натискане надолу и други

• Поддръжка за уеб рамки : Django, web2py и Flask

• Интегриран дебъгер на Python

• Интегрирано тестване на единици , с покритие ред по ред

• Разработка на Python на Google App Engine

• Интегриране на контрола на версиите: унифициран потребителски интерфейс за Mercurial , Git , Subversion , Perforce и CVS със списъци с промени и сливане

• Интегриране на научни инструменти: интегрира се с IPython Notebook, има интерактивна Python конзола и поддържа Anaconda, както и множество научни пакети, включително Matplotlib и NumPy.

### ****Pycharm: среда за разработка на Python:****

PyCharm е IDE е интегрирана среда за разработка (IDE), използвана за програмиране в Python. Той осигурява анализ на кода, графичен дебъгер, интегриран модулен тестер, интеграция със системи за контрол на версиите и поддържа уеб разработка с Django. PyCharm е разработен от чешката компания JetBrains. Той е междуплатформен, работи на Microsoft Windows, macOS и Linux. PyCharm има професионално издание, издадено под патентован лиценз и издание на общността, пуснато под лиценза на Apache. PyCharm Community Edition е по-малко обширно от Professional Edition.

### ****Python рефакторинг:****

Включва преименуване, метод на извличане, въвеждане на променлива, въвеждане на константа, издърпване, натискане надолу и други.

Поддръжка на рамки web: Django, web2py и Flask

Вграден дебъгер на Python

Интегрирано модулно тестване, с покритие код по ред

Интеграция на контрол на версиите- Унифициран потребителски интерфейс за Mercurial, Git, Subversion, Perforce и CVS със списъци за промяна и сливане.

## Инсталиране на PyCharm на Linux

В случай на Ubuntu и деривати PyCharm е достъпен за изтегляне от софтуерния център на Ubuntu в три издания: Pro версия, EDU версия и CE версия. Всичко, което трябва да направите, е да потърсите Pycharm и той ще се появи. За останалите дистрибуции имаме по-обща инсталация, просто трябва да изтеглите файла .tar.gz от официалния сайт на Jet Brains.

След като изтеглянето приключи, отворете терминал в папката за изтегляне, където се намира файлът и въведете следното в терминала, за да го извлечете:

### Първоначална настройка на Pycharm:

Първото изпълнение на Pycharm ще ви позволи да го конфигурирате въз основа на вашите предпочитания, включително конфигуриране на вашите теми, местоположението на вашите проекти и конфигуриране на приставките, които искате да включите. Те трябва да прочетат „Споразумението за политика за поверителност“ и да го приемат, за да продължат. Направете това, можете да конфигурирате предпочитаната от вас тема по-долу, като цяло има три теми на потребителския интерфейс: Intellij, Darcula и GTK +. Те могат да улеснят стартирането на IDE, използвайки скриптове за стартиране, но можете да го пропуснете.

След това те могат да конфигурират приставките, които искате да включите във вашата инсталация. Началният екран на приставката ще се появи, както следва:

В края на конфигурацията, ще се отвори прозорец, подобен на следния за да създадете нов проект, да го отворите или да излезете от системата за контрол на версиите.

След като са избрали проект, началният екран на приложението, който ще видите, ще бъде следният:

# Изготвяне на автобиография

**Автобиографията** (от [гръцките](https://bg.wikipedia.org/wiki/%D0%93%D1%80%D1%8A%D1%86%D0%BA%D0%B8) корени *autos*, „себе си“, *bios*, „живот“, и *graphein*, „пиша“ – букв. „собствено животоописание“) е [текст](https://bg.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82), разказващ живота на своя автор. Това най-често се отнася до литературната автобиография или литературния жанр на автобиографията. Понякога в литературата се говори за автобиографизъм или автобиографични черти на някой текст. В бизнеса автобиографията е нехудожествен текст, съдържащ съществена информация за кандидатстващия за определена позиция. Обикновено под автобиография за кандидатстване днес се разбира [CV](https://bg.wikipedia.org/wiki/CV), което е и по-често употребяваният термин.

Автобиографията е документ, за който има определени правила. Тя обаче е и нещо, с което може да покажем личен почерк и с което задължително трябва да привлечем вниманието на специалиста в рамките на секунди.

Кои са полетата в биографията, без които не може и как да покажем, че имаме личен стил?

 Добре е да:

– Използваме свои думи;

– Потърсим подходящ за бранша формат;

– Да не копираме готови формати! Желателно е да внесем свой почерк в CV-то си, така че то да се откроява от останалите кандидатури;

– Бизнесът не обича формата Europass, но ако кандидатстваме за работа в държавни институции е добре да използваме най-новите версии на европейската биография.

## **Формат на автобиографията:**

Задължителните елементи, които трябва да съдържа една автобиография са:

– **Лични данни**:  имена, мобилен телефон за връзка, електронна поща, адрес.

– **Трудов стаж**:  желателно е да е най-отгоре, защото първо от него се интересуват работодателите.

– **Образование:**  висше, средно и специалности.

– **Допълнителни квалификации**: всякакви курсове и специализации, които помагат на човек да се промотира.

– **Лични качества**:  тук трябва да се избягват клишета, като например вместо „умения за работа в екип“ можете да впишете „желание за отборна игра“ и т.н.

– **Езикови и технически умения.**

.

### Лични данни:

Понастоящем се водят доста спорове за степента, в която личните данни следва да са обект на автобиографията.

Едно нещо е сигурно: в автобиографията следва да се упомене името и координати за връзка, в противен случай ще е въпрос на провидението потенциалните работодатели да се свържат с Вас в следствие на интереса Ви към дадена професионална възможност.

* + Много често координатите за връзка се ограничават единствено до пощенски адрес, който въпреки че е необходим, съвсем не е достатъчен.
  + Притиснати от времето, занимаващите се с подбор на персонал предпочитат да се свържат с Вас по телефона и ако по някаква причина не е посочен такъв, рискът е да не бъдем потърсени въобще.
  + Когато не разполагаме със собствен телефон, можете да посочим телефон на роднина, приятел, човек, с когото имаме контакт и на когото се доверяваме. В този случай, също така, е необходимо да посочим, че на съответния телефон може да бъде оставено съобщение.
  + Не са за пренебрегване, също така, съвременните средства за комуникация и по-специално Интернет.

Във връзка с въпросите, свързани със семейното положение и възрастта, е спорно до каква степен те трябва да са предмет на описание в автобиографията.

* + Традиционният американски подход, който все повече навлиза и в Европа, предполага, че е налице дискриминация, когато при избора си работодателите се ръководят от възрастта и личните ангажименти на кандидатите.
  + Тези характеристики не би следвало да влияят на иначе идеалното Ви съответствие на изискванията за длъжността. Въпреки това, обаче, повечето работодатели са си изградили стереотип, според който определени личностни данни могат да влияят на ефективността на кандидата при работа.
  + Понякога е налице определена логика в подобни изисквания: някои длъжности изискват интензивни физически усилия, които не са постижими за по-възрастните кандидати; имиджът на други длъжности предполага, че те трябва да бъдат заемани от младежи; определени позиции изискват активно пътуване от порядъка на отсъствие от дома за четири дни в седмицата и поради това е малко вероятно жени с малки деца да приемат подобна работа. Вярно е, че до голяма степен е въпрос на лична самооценка дали кандидатът може да посрещне подобни изисквания, но в процеса на подбор другата страна претендира да направи крайния избор.

### Образование и квалификации

Във връзка с образованието, това което е от най-голямо значение е най-високата квалификационна степен.

* + Ясно следва да се упомене университета, специалността и получената образователна степен.
  + От значение е да се посочи кога е започнато и приключено образованието.
  + При продължаващо обучение, би трябвало да се посочи очаквания срок на дипломиране, както и информация за начина на обучение - редовно или задочно.

Ако средното образование е специализирано (например езикова гимназия, математическа, природонаучна гимназия, техникум и т.н.), ще е ценно това да се упомене в автобиографията.

Всякакви допълнителни курсове на обучение, професионални семинари, придобити сертификати, свидетелстващи за получени знания и развити умения, ще са предимство и следва да бъдат направени достояние на потенциалния работодател.

### Постижения

Тази категория е изборна. Автобиографията на кандидат с над 2 години професионален опит обаче се очаква да представи фактологически доказателства, подкрепящи твърденията му за професионална компетентност.

* + Тези факти могат да се отнасят до реализиран обем на продажбите, разширен пазарен дял, стойност на спестени финансови ресурси, брой на успешно назначени хора, видове успешно стартирани проекти, създаване на нови продукти и услуги, приложени в практиката предложения за оптимизиране на работни процеси, реализирани производствени и технически методи и т.н.
  + За току-що дипломиралите се постиженията могат да се изразяват в успешно преминати проекти, постигнат бал на държавните изпити, други награди.

Когато се отнасяме към определен факт като към постижение, трябва да сме уверени, че той изглежда достатъчно значим спрямо длъжността, за която кандидатстваме и че ще бъде възприет и от потенциалния работодател за такъв. В противен случай това, което цитираме като постижение, може да бъде интерпретирано като свидетелство за това, че си поставяме сравнително скромни цели.

### Допълнителни квалификации

Поради взискателната и предизвикателна бизнес среда от кандидатите се очаква да владеят чужди езици, да са в състояние да използват компютърен софтуер, да демонстрират добри умения за комуникация, да притежават организационни способности, умение да управляват хора, аналитичен подход към разрешаването на проблеми и взимането на решения.

Опитът и познанията, които свидетелстват за наличието на подобни умения, са високо ценени и следователно следва да бъдат представени в автобиографията.

### Професионален опит

Категорията трябва ясно да упоменава какъв е опитът от гледна точка на служител на пълен или непълен работен ден, специалист на свободна практика или стажант.

Тази част от автобиографията може да бъде организирана в съответствие с един от следните принципи:

* + Функционален – подходящ за кандидати, чийто професионален път се характеризира със значителни промени - преминаване от една област в друга или честа смяна на компаниите, в които са работили.
  + Вторият принцип – който е и по-често прилаганият – е изброяването на длъжности и местоработи по реда на тяхното заемане - в хронологична или обратна на хронологичната последователност.
  + Изброяването на заеманите длъжности и съответните компании обаче не е достатъчно. За да сте в състояние да създадете представа за типа на заданията, които сте изпълнявали, е необходимо накратко да опишете характера на задълженията и отговорностите си към момента на изпълнението на съответната длъжност. В идеалния случай можете също така да опишете Вашия статус в йерархията, като определите длъжността, на която сте бил подчинен, както и позициите, които сте ръководили.

Когато се описват отделните местоработи, е необходимо ясно да се определи началото и приключването на ангажимента към съответната компания. Разбира се, никой не очаква да са написани точните дати, но е необходимо да се упомене месеца и годината.

Понякога, поради причини, свързани със спазването на поверителност, може да не се споменава името на компаниите, за които сме работили / все още работим.

* + Винаги изглежда подозрително за потенциалния работодател, притеснението от разкриването името на компанията: първата асоциация, която той би си направил, е опит за прикриване на нещо, което би ни представило в неблагоприятна светлина – недоволен бивш работодател, слабо професионално представяне и т.н.
  + Също толкова неприятна за работодателя е мисълта, че за съмнение в неговия професионализъм и бизнес етика и опасението от "изтичане" на информация.
  + Ако имаме такова опасение, би трябвало да го разрешим преди да подадем документи за кандидатстване за работа. Трабва да се опитаме да съберем информация и да преценим надеждността на компанията. Ако кандидатстваме в резултат на анонимна обява, би трябвало да претеглим риска от това да бъдем "издадени" спрямо желанието ни да заемем обявената вакантна длъжност. По принцип се предполага, че работодателите се стремят към привличането на най-добрите кандидати и поради това би следвало да спазват поверителност по отношение на професионалния ни интерес.

### Езикови и технически умения

Поради взискателната и предизвикателна бизнес среда от кандидатите се очаква да владеят чужди езици, да са в състояние да използват компютърен софтуер, да демонстрират добри умения за комуникация, да притежават организационни способности, умение да управляват хора, аналитичен подход към разрешаването на проблеми и взимането на решения.

Опитът и познанията, които свидетелстват за наличието на подобни умения, са високо ценени и следователно следва да бъдат представени в автобиографията.

# Използвани библиотеки за създаване на програмата:

## *****CV2*****

OpenCV е библиотека от програмни функции главно за компютърно зрение в реално време. Първоначално разработен от Intel , по-късно се поддържа от Willow Garage , след това от Itseez. Библиотеката е междуплатформена и е лицензирана като безплатен софтуер с отворен код под Apache License 2. От 2011 г. OpenCV включва GPU ускорение за операции в реално време. OpenCV е написан на езика за програмиране C++ , както и основният му интерфейс, но все още запазва по-малко изчерпателен, но обширен по-стар C интерфейс. Всички по-нови разработки и алгоритми се появяват в C++ интерфейса. Има езикови обвързвания в Python , Java и MATLAB / Octave . Интерфейсът за приложно програмиране ( API ) за тези интерфейси може да бъде намерен в онлайн документацията. Разработени са библиотеки за обвивки на няколко езика, за да се насърчи приемането от по-широка аудитория. Във версия 3.4, JavaScriptо обвързвания за избрано подмножество от функции на OpenCV бяха пуснати като OpenCV.js, за да се използват за уеб платформи. Ако библиотеката открие интегрираните примитиви за производителност на Intel в системата, тя ще използва тези патентовани оптимизирани процедури, за да се ускори. От септември 2010 г. се разработва базиран на Compute Unified Device Architecture ( CUDA ) интерфейс за графичен процесор (GPU). Базиран на OpenCL GPU интерфейс се разработва от октомври 2012 г., документацията за версия 2.4.13.3 може да бъде намерена на docs.opencv.org. OpenCV работи на настолни операционни системи : Windows , Linux , macOS , FreeBSD , NetBSD и OpenBSD , както и на мобилни операционни системи: Android , iOS , Maemo , BlackBerry 10 и QNX . Потребителят може да получи официални версии от SourceForge или да вземе най-новите източници от GitHub.

## *****Pyautogui*****

PyAutoGUI позволява на Python скриптове да контролират мишката и клавиатурата, за да автоматизират взаимодействията с други приложения. API е проектиран да бъде прост. PyAutoGUI работи на Windows, macOS и Linux и работи на Python 2 и 3. PyAutoGUI има няколко функции:

• Преместване на мишката и щракване в прозорците на други приложения.

• Изпращане на натискания на клавиши към приложения (например за попълване на формуляри).

• Направете екранни снимки и дадено изображение (например на бутон или квадратче за отметка) и го намерете на екрана.

• Намерете прозореца на приложението и го преместете, преоразмерете, увеличете, минимизирайте или затворете (само за Windows, в момента).

• Показване на кутии за предупреждение и съобщения.

* Примери:

>>> import pyautogui

>>> screenWidth, screenHeight = pyautogui.size() # Get the size of the primary monitor.

>>> screenWidth, screenHeight

(2560, 1440)

>>> currentMouseX, currentMouseY = pyautogui.position() # Get the XY position of the mouse.

>>> currentMouseX, currentMouseY

(1314, 345)

>>> pyautogui.moveTo(100, 150) # Move the mouse to XY coordinates.

>>> pyautogui.click() # Click the mouse.

>>> pyautogui.click(100, 200) # Move the mouse to XY coordinates and click it.

>>> pyautogui.click('button.png') # Find where button.png appears on the screen and click it.

>>> pyautogui.move(400, 0) # Move the mouse 400 pixels to the right of its current position.

>>> pyautogui.doubleClick() # Double click the mouse.

>>> pyautogui.moveTo(500, 500, duration=2, tween=pyautogui.easeInOutQuad) # Use tweening/easing function to move mouse over 2 seconds.

>>> pyautogui.write('Hello world!', interval=0.25) # type with quarter-second pause in between each key

>>> pyautogui.press('esc') # Press the Esc key. All key names are in pyautogui.KEY\_NAMES

>>> with pyautogui.hold('shift'): # Press the Shift key down and hold it.

pyautogui.press(['left', 'left', 'left', 'left']) # Press the left arrow key 4 times.

>>> # Shift key is released automatically.

>>> pyautogui.hotkey('ctrl', 'c') # Press the Ctrl-C hotkey combination.

>>> pyautogui.alert('This is the message to display.') # Make an alert box appear and pause the program.

Този пример плъзга мишката в квадратна спирална форма в MS Paint (или друга програма за рисуване на графики):

>>> distance = 200

>>> while distance > 0:

pyautogui.drag(distance, 0, duration=0.5) # move right

distance -= 5

pyautogui.drag(0, distance, duration=0.5) # move down

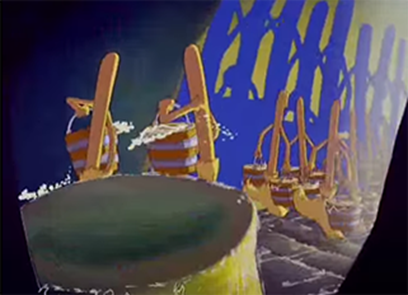
pyautogui.drag(-distance, 0, duration=0.5) # move left

distance -= 5

pyautogui.drag(0, -distance, duration=0.5) # move up

Ползата от използването на PyAutoGUI, за разлика от скрипт, който директно генерира файла с изображение, е, че може да се използват инструментите за четки, които MS Paint предоставя.

## *****Сейфове:*****

Подобно на омагьосаните метли от Sorcerer's Apprentice, програмирани да продължават да пълнят (и след това да препълват) ваната с вода, грешка във вашата програма може да я накара да излезе извън контрол. Трудно е да използвате мишката, за да затворите програма, ако курсорът на мишката се движи сам. Като функция за безопасност по подразбиране е активирана функция за безопасност при отказ. Когато се извика функция на PyAutoGUI, ако мишката е в някой от четирите ъгъла на основния монитор, те ще повишат pyautogui.FailSafeException. Има забавяне от една десета от секундата след извикване на всяка функция на PyAutoGUI, за да се даде време на потребителя да удари мишката в ъгъла, за да задейства защитата при отказ. Можете да деактивирате тази защита от грешки, като зададете . СИЛНО ПРЕПОРЪЧВАМ ДА НЕ ДЕАКТИВИРАТЕ FAILSAFE.pyautogui.FAILSAFE = False. Закъснението от десета секунда се задава от pyautogui.PAUSEнастройката, която е 0.1 по подразбиране. Можете да промените тази стойност. Има и pyautogui. DARWIN\_CATCH\_UP\_TIME настройка, която добавя допълнително забавяне на macOS след събития с клавиатура и мишка, тъй като изглежда, че операционната система се нуждае от забавяне, след като PyAutoGUI издаде тези събития. Зададено е 0.01по подразбиране, добавяйки допълнително забавяне от стотна секунда.

# Реализация

Целта на проекта е да се разработи приложение с удобен и ефективен начин за съставяне на високо качествени и лесно достъпни автобиографии.Това може да помогне да се постгнат лични и/или професионални цели, като намиране на работа, кандидадстване за образование или изграждане на личен бранд..

## Анализ на заданието

Заданието за дипломния проект предполага разработване на приложение за създаване на автобиография на базата на динамичното генериране на формуляри и PDF файлове. Потребителите могат да въвеждат своята лична информация, трудов опит и умения и автоматично се генерира професионално изглеждаща PDF автобиография въз основа на въведените от потребителя даннипроект трябва да представлява уеб базирано приложение, което да позволява на потребителя:

* да въведе съдържание като име, умения, трудов опит, сертификати, уебсайтове и т.н. във формуляр.
* да преобразуват данните в професионален дизайн на автобиография. да избира от един или повече шаблони според предпочитанията си.

Има се предвид да се разработи уеббазирано приложение, което да е достъпно онлайн и да е адаптивно - да изглежда еднакво добре на различни устройства.

## Архитектура 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.

## Реализация

### Формата

Django предоставя готов механизъм за работа с форми. Това предполага както създаване, така и жалидиране и обработка на получените данни. Дефинирането на формите се дава във файла forms.py. В конкретния случай формата е една – за въвеждане на данните за автобиографията, но е доста голяма – имо много полета. Ето чзст от кода:

class ContactForm(forms.Form):  
 name=forms.CharField()*#required=False* email=forms.EmailField(label='E-Mail')  
 mobile=forms.CharField()  
 address=forms.CharField()  
 skills\_1=forms.CharField()  
 skills\_2=forms.CharField()  
 skills\_3=forms.CharField(required=False)  
 skills\_4=forms.CharField(required=False)  
....

....

....  
   
 def \_\_init\_\_(self,\*args,\*\*kwargs):  
 super().\_\_init\_\_(\*args,\*\*kwargs)  
 self.helper=FormHelper  
 self.helper.form\_class = ' container justify-content-center '  
 *# self.helper.label\_class = ''  
 # self.helper.field\_class = 'col-md-6 col-xs-9'* self.helper.form\_method="post"  
 self.helper.layout=Layout(  
 Row(  
 Column('name', css\_class='form-group col-md-5 mb-10'),  
 Column('email', css\_class='form-group col-md-7 mb-10'),  
 css\_class='form-row center'  
 ),  
....

....

....

css\_class='form-row center'  
 ),  
 'experience\_2\_desc',  
 'education\_1',  
 Row(  
 Column('education\_1\_dur', css\_class='form-group col-6),  
 Column('education1\_score', css\_class='form-group col-6),  
 css\_class='form-row center'  
 ),  
 'education\_2',  
....

....

....

Submit('submit','Submit',css\_class="btn-success")  
 )

### Изглед

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

def info(request):  
 form=ContactForm()  
 if request.method == 'POST':  
 form=ContactForm(request.POST)  
 if form.is\_valid():  
 name=form.cleaned\_data['name']  
 email=form.cleaned\_data['email']  
  
 skills\_1=form.cleaned\_data['skills\_1']  
 skills\_2=form.cleaned\_data['skills\_2']  
 skills\_3=form.cleaned\_data['skills\_3']  
 skills\_4=form.cleaned\_data['skills\_4']  
  
 mobile=form.cleaned\_data['mobile']  
 address=form.cleaned\_data['address']  
  
 experience\_1\_title=form.cleaned\_data['experience\_1\_title']  
 experience\_1\_dur=form.cleaned\_data['experience\_1\_dur']  
 experience\_1\_desc=form.cleaned\_data['experience\_1\_desc']  
  
 experience\_2\_title=form.cleaned\_data['experience\_2\_title']  
 experience\_2\_dur=form.cleaned\_data['experience\_2\_dur']  
 experience\_2\_desc=form.cleaned\_data['experience\_2\_desc']  
  
 education\_1=form.cleaned\_data['education\_1']  
 education\_1\_dur=form.cleaned\_data['education\_1\_dur']  
 education1\_score=form.cleaned\_data['education1\_score']  
  
 education\_2=form.cleaned\_data['education\_2']  
 education\_2\_dur=form.cleaned\_data['education\_2\_dur']  
 education2\_score=form.cleaned\_data['education2\_score']  
  
 data={'name':name}  
 data['email']=email  
 data['skills\_1']=skills\_1  
 data['skills\_2']=skills\_2  
 data['skills\_3']=skills\_3  
 data['skills\_4']=skills\_4  
  
 data['mobile']=mobile  
 data['address']=address  
  
 data['experience\_1\_title']=experience\_1\_title  
 data['experience\_1\_dur']=experience\_1\_dur  
 data['experience\_1\_desc']=experience\_1\_desc  
  
 data['experience\_2\_title']=experience\_2\_title  
 data['experience\_2\_dur']=experience\_2\_dur  
 data['experience\_2\_desc']=experience\_2\_desc  
  
 data['education\_1']=education\_1  
 data['education\_1\_dur']=education\_1\_dur  
 data['education1\_score']=education1\_score  
  
 data['education\_2']=education\_2  
 data['education\_2\_dur']=education\_2\_dur  
 data['education2\_score']=education2\_score  
 return render(request,'home.html',data)  
 *#to add more go to : forms.py  
 # print(name,email)* return render(request,'info.html',{'form':form})

### Шаблон

За да се визуализира описаното до тук е нужен HTML шаблон. Той играе ролята на визуален контейнерк, който пренася информацията до потребителя:

{% load crispy\_forms\_tags %}  
<!DOCTYPE html>  
<html lang="en">  
<head>  
 <title>Curriculum Vitae</title>  
 <meta name="viewport" content="width=device-width" />  
 <meta name="description" content="The Curriculum Vitae." />  
 <meta charset="UTF-8">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css" integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">  
</head>  
<body style="padding:20px;">  
 <h1 class="mt-2" style="text-align:center;font-weight:bolder;color:steelblue;">Welcome to SAH Resume Builder!!!</h1>  
 <hr class="mt-0 mb-4">  
*<!-- <form method="POST">  
 {% csrf\_token %}  
 {{ form|crispy }}  
 <button type='Submit'>Submit</button>  
   
 </form> -->* {% crispy form form.helper %}  
  
</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.urls import path  
from . import views  
  
urlpatterns = [  
 path('resume/', views.home, name="home"),  
 path('',views.info,name="info")  
]

# Заключение

Приложението за създаване на автобиография, направено в Django, помага лесно да правите впечатляващи автобиографии в рамките на няколко минути. Изграждането на автобиография без чужда помощ би било трудна задача като определяне на оформление, текст, показване на необходимото поле и т.н. Така че, за да преодолеем тези трудности, създадох уеб-приложение за създаване на автобиография, което взема основна информация като име, имейл адрес, мобилен номер , адрес, някои умения, минал опит и др. на кандидата, необходими при кандидатстване за работа. По тоз начин всеки може да си направи професионална автобиография, незабавно генерирана от моя „създател на автобиография“. Важната цел на този проект е да помогне на служителите да намерят мечтаната работа с перфектните автобиографии. Формулярите за съставителя на автобиография се правят с помощта на Crispy forms.

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/