**ПРОФЕСИОНАЛНА ГИМНАЗИЯ**

**ПО ЕЛЕКТРОНИКА И ЕНЕРГЕТИКА – гр. БАНСКО**

БЪЛГАРИЯ, БАНСКО 2770, УЛ. ''БЪЛГАРИЯ'' 23, ТЕЛ:0899 89 29 28

*e-mail:* info-102015@edu.mon.bg

***Дипломен проект***

**по професия код 481030 "Приложен програмист"**

**специалност код 4810301 "Приложно програмиране"**

**Тема:**

Разработка на платформа за приемане, обработване и наблюдение на заявки за обслужване във фирма за IT support или ISP

Ученик: Радослава Топузова Ръководител: инж. Георги Бориков

**Банско, 2024**

# Съдържание

[1. Съдържание 2](#_Toc164598275)

[2. Увод 4](#_Toc164598276)

[3. Цели на проекта 5](#_Toc164598277)

[4. Основна част 6](#_Toc164598278)

[4.1 Python 6](#_Toc164598279)

[5. World Wide Web 7](#_Toc164598280)

[5.1 Първите уеб браузъри 8](#_Toc164598281)

[5.2 Поддомейнът www 9](#_Toc164598282)

[6. HTML (HyperText Markup Language) 10](#_Toc164598283)

[6.1 Хипертекст (hypertext) 10](#_Toc164598284)

[6.2 Препратка (линк, хипервръзка) в HTML 12](#_Toc164598285)

[6.3 Препратки към страници от сайта 12](#_Toc164598286)

[6.4 Препратки към части от една страница 12](#_Toc164598287)

[6.5 Препратка, която се отваря в нов прозорец 12](#_Toc164598288)

[7. CSS (Cascading Style Sheets) 13](#_Toc164598289)

[7.1 Външен стил 13](#_Toc164598290)

[7.2 Вътрешен стил 14](#_Toc164598291)

[7.3 Вграден стил 14](#_Toc164598292)

[7.4 Коментари в CSS 14](#_Toc164598293)

[8. JavaScript 14](#_Toc164598294)

[9. MySQL 15](#_Toc164598295)

[9.1 Бази данни 15](#_Toc164598296)

[9.2 MySQL базите данни са релационни 16](#_Toc164598297)

[9.3 SQL и MySQL 16](#_Toc164598298)

[9.4 MySQL е софтуер с отворен код 16](#_Toc164598299)

[10. Vue 17](#_Toc164598300)

[10.1 Предимства на Vue 17](#_Toc164598301)

[10.2 Недостатъци на Vue 18](#_Toc164598302)

[10.3 Библиотеки на Vue 19](#_Toc164598303)

[10.4 Компоненти на Vue 19](#_Toc164598304)

[10.5 Свързани инструменти 20](#_Toc164598305)

[11. Django 21](#_Toc164598306)

[11.1 Предимства на Django 22](#_Toc164598307)

[11.2 Django admin panel 23](#_Toc164598308)

[12. СЕРИАЛИЗАТОРИ И ИЗГЛЕДИ (VIEW) 23](#_Toc164598309)

[13. REST 24](#_Toc164598310)

[14. REST API 26](#_Toc164598311)

[14.1 REST API 27](#_Toc164598312)

[14.2 Предимствата на REST API 27](#_Toc164598313)

[14.3 Изграждане на RESТ API 29](#_Toc164598314)

[15. SOAP 29](#_Toc164598315)

[15.1 Как работи SOAP 30](#_Toc164598316)

[15.2 Как работи протоколът SOAP 30](#_Toc164598317)

[16. PHP 31](#_Toc164598318)

[16.1 Софтуерен пакет XAMPP 31](#_Toc164598319)

[17. Реализация 32](#_Toc164598320)

[18. ПЪРВИ СТЪПКИ НА ПРОЕКТА: 32](#_Toc164598321)

[18.1 СЪЗДАВАНЕ НА ПРИЛОЖЕНИЕ: 32](#_Toc164598322)

[19. Оторизация на потребител 34](#_Toc164598323)

[20. Списък на използваната литература 36](#_Toc164598324)

# Увод

В настоящата епоха на технологични иновации и бързо развиващи се информационни технологии, необходимостта от ефективен ИТ съпорт е от съществено значение за успешното функциониране на различни бизнес сектори и организации. С разрастващ се обхват на ИТ услугите и нарастващ брой на потребителите, е необходимо да се разработят иновативни и устойчиви платформи за ИТ съпорт. В бъдеще зависимостта ни от техниката и технологиите ще става все по-голяма. А това ще увеличи и зависимостта ни от хората, които могат да ги поправят или да ни помогнат да разберем или дори подобрим функциите им.

Платформата има за цел ефективно да приема, обработвва и наблюдава оплакванията на клиентите. Системата позволява на клиентите да регистрират своите оплаквания, да проследяват статуса им и да получават обратна връзка.

Проектът ще се състои от две части:

* Front-end, който ще е отговорен за получаване на оплаквания от клиенти и показване на статуса на техните оплаквания.
* Back-end, който отговаря за съхраняването на оплакванията и управлението на техния статус.

Фронтендът на проекта ще бъде изграден с помощта на HTML, CSS и JavaScript, а бек-ендът с помощта на Python Django, който предоставя удобен начин за съхраняване и извличане на данни от база данни.

Този дипломен проект се фокусира не само върху подобряването на обслужването на клиенти, а и върху по-добра отчестност на фирмата. Чрез използването на иновативни технологии и стратегии, този проект има за цел да насърчи по-бърза и качествена реакция от страна на фирмата, подобрявайки продуктивността и удовлетворението на клиентите.

# Цели на проекта

Преди ясно да дефинираме целта на проекта, трябва да се замислим какви бихме искали да бъдат резултатите от работата ни. В случая ние бихме желали по възможно най-ефективен, бърз и удобен (за служителите и потребителите на приложението) начин да обработваме и наблюдаваме заявки от страна на клиентите, както и да постигнем по-добра отчетност на фирмата след изграждането на приложението. Друг важен фактори е одобрението проекта от страна на служителите на фирмата, които имат важна роля в намирането на чини с които той би бил реализиран. Не на последно място при определянето на главната причина за създаването на проекта са нуждите на фирмата.

След като вече сме определили най-значимите фактори при изграждането на приложението, бихме могли да заявим че, целта на проекта е разработване на софтуерно приложение, което ефективно позволява да се управляват заявки за обслужване на клиентите. Приложението работи само с регистрирани на потребители, раделени в две групи- клиенти и служители на фирмата. Клиентите могат да регистрират нова заявка, да проследяват статуса и да правят справки за предишни свои заявки. Операторите потвърждават приемането на заявките и според предприетите действия променят статуса им („приета“, „обработва се“, „завършена“). Системата може да се администрира с помощта на администраторско меню, позволяващо добавяне, редактиране или премахване на потребители (от двете групи) или промяна на техни атрибути.

След като заявкте бъдат завършени администратора ще има по-добра отчетност на фирмата, тоест ще вижда кои потребители с какви пороблеми са се сблъсквали и как те са решени.

# Основна част

## 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.

Идеята за Python се заражда в края на 1980-те години, като реалното осъществяване започва през декември 1989 г. от Гуидо ван Росум в CWI (Centrum Wiskunde & Informatica – международно признат изследователски институт по математика и компютърни науки, локализиран в Амстердам, Холандия). Python имал за цел да се превърне в наследник на ABC (език за програмиране), който да бъде способен да обработва изключения и да е съвместим с операционната система Amoeba. Ван Росум е основният автор на Python, а неговата продължаваща централна роля в развитието на езика е ясно отразена в титлата, дадена му от Python общността – „пожизнен доброжелателен диктатор“.

Python предлага добра структура и поддръжка за разработка на големи приложения. Той притежава вградени сложни типове данни като гъвкави масиви и речници.

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, различни системни функции, сокети (sockets), програмни интерфейси към GUI библиотеки като Tk и др.

Поради факта, че Python е интерпретативен език, се спестява значително време за разработка, тъй като не са необходими компилиране и свързване 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;
* не са необходими декларации на променливи или аргументи;

Python съдържа прости конструкции, характерни за функционалния стил на програмиране, които му придават допълнителна гъвкавост.

Програмите, написани на Python, представляват съвкупност от файлове с изходен код. При първото си изпълнение този код се компилира до байткод, а при всяко следващо се използва кеширана версия. Байткодът се изпълнява от интерпретатор на Python. Ще отбележим още и следните особености:

* 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.

# World Wide Web

World Wide Web (наричана още само уеб, WWW) е виртуална информационна мрежа, съставена от хипертекстови документи (уеб страници и уеб сайтове). Достъпът до тази информационна мрежа се извършва през интернет (глобална система от взаимносвързани компютърни мрежи).

Идеята за World Wide Web на английския учен Тим Бърнърс-Лий се заражда през 1989 г. на около две хиляди километра западно от България, в изследователския комплекс на ЦЕРН (Европейската организация за ядрени изследвания).

Проектът World Wide Web първоначално има за цел да подобри вътрешната система за обмяна и работа с информацията в институтите и лабораториите на ЦЕРН. Разработките по проекта започват през 1989 г., след като Тим Бърнърс-Лий получава одобрение на предложението му за разпределена хипертекстова система.

В реализирането на проекта участва малка група учени, един от които е белгийският учен Робер Кайо, който става важен застъпник на проекта.

В края на 1990 г. вече е създаден HTTP протоколът, първият уеб сървър (httpd), първият уеб браузър (WorldWideWeb), HTML езикът за маркиране на хипертекст и първият уеб сайт.

По същото това време ЦЕРН вече е свързан в световната компютърна мрежа, станала вече известна като интернет. По света към мрежата се включват и различни комерсиални компании. Появяват се първите интернет доставчици. Мрежата се разраства с голяма скорост и достига до все повече обикновени потребители.

През Август 1991 г. Тим Бърнърс-Лий публикува съобщение в дискусионната група alt.hypertext, в което описва проекта World Wide Web. С това проектът става публично достояние и малко след това (1991-1992 г.) започват да се появяват първите уеб сървъри извън Европа.

## Първите уеб браузъри

Първият графичен уеб браузър и редактор „WorldWideWeb“, създаден към проекта WWW през 1990 г., се е поддържал само на компютър NeXT, с операционна система NeXTstep. На по-късен етап уеб браузърът е преименуван на Nexus, за да се избегне объркване с името на проекта World Wide Web.

През 1993 г. се появява Mosaic – по-добър графичен уеб браузър, в който изображенията са се показвали вградени в уеб страниците, а не в отделен прозорец. По това време това е най-известния и най-използван уеб браузър, който допринася за скоростното разрастване на уеб. Поддържа се при различните платформи, съществуващи по това време, като Windows, Macintosh и други.

На базата на Mosaic през 1994 г. се появява Netscape Navigator – следващият хит при уеб браузърите. Важното нововъведение в този уеб браузър е, че текстът и изображенията са се показвали веднага на екрана, докато самата страница се е зареждала. Предишните уеб браузъри не са показвали нищо (бяла страница), докато цялото съдържание на страницата не се изтегли напълно.

Кодът на Netscape е послужил като основа за уеб браузъра Mozilla Firefox.

## Поддомейнът www

Използването на www като име на хост за услугата World Wide Web е в следствие на утвърдилата се по това време практика, имената на хостовете в интернет да се наименуват на услугата, която ще предоставят. Например www е уеб услуга (WWW), ftp за FTP услуга, mail за имейл услуга и други.

При бързото разпространение на уеб през първите години, повечето потребители, които са добавяли уеб сайт в интернет, са използвали този метод за наименуване на услугата. Поради това, повечето уеб сайтове тогава са били достъпни през поддомейна www.mysitename.com.

Тъй като използването на поддомейна www не е задължително, уеб адресът на проекта World Wide Web е останал http://info.cern.ch, а не както е било замислено в началото да бъде на http://www.cern.ch.

През 1991 г. във вестника, издаван от ЦЕРН, е публикувано съобщение за проекта World Wide Web. В съобщението се описва какво представлява и как се работи с WWW. Също така на страница 2 може да се види предложение отпечатването на вестника да бъде преустановено и вместо това новините да се публикуват и четат през проекта WWW. Първият уеб сървър е CERN httpd (Hypertext Transfer Protocol daemon). Първата уеб страница, създадена по проекта World Wide Web, съдържа хипертекст, описан на първия HTML. Страницата предоставя информация за самия проект WWW.

# HTML (HyperText Markup Language)

HTML е съкращение от HyperText Markup Language (буквално „хипертекстов език за маркиране“). Създаден е в края на 80-те години от англичанина Тим Бърнърс-Лий, който се опитва да намери нов метод (World Wide Web) за обмяна на информация с колегите си – учени от Европейската Лаборатория по Физика на Елементарните Частици в Женева. Същността на неговия метод е създаване на текстови документи, обвързани помежду си с хипервръзки, които да се прехвърлят чрез мрежата до отдалечен потребител. Така се появява необходимостта от специален език, на който да се пише (маркира) хипертекст.

В HTML се използват маркери (тагове), които уеб браузърът чете, за да подреди съдържанието и изгради уеб страницата на екрана на потребителя.

Основните елементи на HTML документа са html, head, title, body:

Спецификациите и правилата в HTML се контролират от консорциума World Wide Web Consortium (W3C). Това е организация по стандартизация и установяване на различни стандарти за World Wide Web (Уеб).

## Хипертекст (hypertext)

Терминът хипертекст (англ. hypertext) се състои от гръцката дума представка, означаваща „супер-„, „свръх-„, „над-„, и думата текст (супер-текст, хипер-текст).

Хипертекстът е обикновен текст, към който са добавени връзки към други текстове. Тези връзки се наричат хипервръзки, препратки или линкове и са начин за свързване и достъп до допълнителна информация.

Чрез хипервръзките потребителят може да сърфира в електронния текстов масив, преминавайки от един текст към друг, към следващ и т.н.

Така образувалата се мрежа от свързани хипертекстови документи е основният компонент на световната информационна мрежа (World Wide Web).

За достъп до хипертекста в тази мрежа, намиращ се предимно в уеб страници, се използва специална програма наречена уеб браузър.

Най-известният език, на който се пише (маркира) хипертекст, е HTML (Hypertext Markup Language). При достъпването на уеб страница, уеб браузърът прочита HTML кода й. По зададените в него правила нагласява информацията на екрана на потребителя.

В началото на World Wide Web хипервръзките са сочели предимно към части от същия документ или друг текст, който може да се намира на друг сървър.

Сега хипертекстовите документи съдържат различни връзки, не само към друг текст, но и към изображения, документи, файлове, аудио и видео съдържание. А самите хипервръзки може да се прикрепят към различни елементи от съдържанието, част от текста, дума, изречение, изображение и други.

Идеята за хипертекст се е зародила много преди появата на World Wide Web.

Концепцията за хипертекст се появява още в системата Мемекс (Memex) на Венивар Буш от 1945г.

Термините хипертекст и хипермедия са въведени от Тед Нелсън през 1963г. Той създава и проекта Ксанаду (Xanadu) за хипертекстова система от 1960г.

Хипертекстът може да се определи като електронно състояние на обикновения текст. Въпреки че сега го откриваме във виртуалното пространство, още през 1941г. е създаден печатния първообраз на идеята за хипертекст (или безкраен текст). Това е книгата „Градината с разклоняващите се пътеки“ на Хорхе Луис Борхес, която може да се опише като хипертекстов роман. Книгата съдържа колекция от свързани кратки разкази, между които читателят може да се движи. Развитието на основната тема в произведението зависи от решенията на читателя.

## Препратка (линк, хипервръзка) в HTML

Един от най-често срещаните елементи в една HTML страница, освен метатаговете, е тагът за хипервръзка. Често наричана и препратка, хиперлинк или линк. В един HTML документ може да се добавят препратки към други документи или ресурси, които се намират на същия сървър или на външен за сайта сървър.

HTML тагът за препратка е <a>, като на неговия атрибут href се задава за стойност URL адресът, към който води връзката. Тагът има затварящ таг </a>.

По подразбиране връзките се показват в браузъра сини на цвят и подчертани, а посетените връзки – виолетови. Цветът им може да бъде променен чрез ползване на CSS код.

Когато препратката води до вътрешна страница от сайта, е необходимо да се укаже точен адрес на тази страница.

## Препратки към страници от сайта

Освен към външни страници препратките могат да водят към страници от един и същ сайт – по този начин той ще изглежда като един цял документ (уебсайт). В този случай не е задължително да се указва пълен адрес на дадена страница. Как точно ще се зададе адреса зависи от това в коя точно директория се намира страницата, към която се прави връзка.

## Препратки към части от една страница

Освен към външни страници и към страници от един и същ сайт, препратките може да водят и към различни части в една и съща страница. Това е подходящо за дълги страници, за да не скролират посетителите в търсене на нужната информация.

## Препратка, която се отваря в нов прозорец

Обикновените препратки зареждат страницата, към която водят, в същия прозорец на браузъра, от който е кликнато на връзката. Така се губи от поглед първата страница и трябва да се натиска „Back“-бутона от менюто на браузъра за връщане в нея. За да се отвори страницата в отделен прозорец, към тага <a> се добавя атрибутът target и му се задава стойност \_blank.

# CSS (Cascading Style Sheets)

CSS (Cascading Style Sheets) е език за описване на презентацията и стиловете на елементите в един HTML/XML документ. CSS е една от основните технологии, използвани в уеб, редом с HTML и JavaScript.

При създаване на HTML страница, съдържанието й се описва (маркира) с HTML код, а презентацията на това съдържание, тоест как ще изглежда то в браузъра, се описва с CSS код (стил).

CSS кодът се поставя във файл с разширение .css. След това този файл може да се използва във всяка една страница от уеб сайта. Така че стилът ще се намира на едно място и при промяна ще се отразява на всички уеб страници.

Структурата на CSS файла включва название на елемента (селектор), за който ще се приложи стила – в случая това е body, и след това параметрите (свойствата), които се ограждат в големи скоби – { }. Когато в големите скоби се поставят няколко свойства, те се отделят един от друг чрез точка и запетая.

Всички параметри, намиращи се в големите скоби {..}, се наричат правило, а то се състои от селектора body и свойствата background-color, font-family, font-size и color. За свойствата се задават стойности.

Когато се задава числова стойност на дадено свойство, например font-size: 16px, можете да оставите стъпка разстояние между двуеточието и цифрите (: 16).

Когато стойността на дадено свойство е съставена от две и повече думи, например sans serif, тези думи трябва да са свързани с тире (sans-serif) или трябва да се поставят в кавички („sans serif“).

## Външен стил

Външен стил (External Style Sheet) се използва когато трябва да се контролират множество HTML документи, като нужните параметри се задават във външен файл.

## Вътрешен стил

Съществува и вътрешен за HTML документа стил (Internal Style Sheet), който се използва за да се зададе вида на един отделен HTML документ, като нужните свойства се задават със специалния таг <style> в секцията <head> на HTML страницата.

## Вграден стил

Другият начин за налагане на стил е Inline Styles – вътрешни за HTML таговете стил. CSS стилът се разполага като атрибут директно в HTML тага например <p>.

Когато в един документ се използват и трите начина за налагане на CSS стил – с най-висок приоритет е стилът на Inline Style (вътрешните за HTML таговете стилове), след тях на по приоритет са Internal Style Sheet (стиловете от секцията head на HTML документа) и последни по приоритет са External Style Sheet, т.е. стиловете, декларирани във външен CSS файл.

## Коментари в CSS

В HTML коментарите, които не влияят на изпълнението на кода, се изписват по подобен начин:

<!-- Коментар, който няма да повлияе на HTML-документа -->

В CSS коментарът трябва да е записан между наклонена черта „звезда“ и „звезда“ наклонена черта:

/\* Коментар, който няма да повлияе на стила \*/

# JavaScript

JavaScript е скриптов език за програмиране, чрез който статичната уеб страница може да се „раздвижи“. Раздвижването може да бъде чрез промяна на съдържанието, което се визуализира на уеб страницата след извършване на действия от страна на потребителя, контрол над уеб браузъра и други. JavaScript се използва, освен за създаване на интерактивни уеб страници, но и за разработка на игри и скриптове, които се изпълняват от страна на сървъра.

Програмите създадени с JavaScript се наричат скриптове. Файлът, в който се съдържа кода на скрипта, е с разширение .js.

JavaScript скриптовете могат да се вграждат в HTML кода на уеб страницата, директно като код или да се извикват като външен .js файл. При зареждането на уеб страницата, външният .js файл се сваля локално на компютъра на потребителя и самото изпълнение на скрипта се извършва от уеб браузъра.

Една от разликите с езика за програмиране Java е, че за изпълнението на JavaScript не е нужно преди това кода да бъде компилиран, а директно се изпълнява/интерпретира от уеб браузъра.

Обикновено при работата на даден уеб сайт взимат участие две страни (сървър и клиент) – сървърът, на който се намира сайта, и уеб браузърът на потребителя, през който се разглежда сайта. Уеб браузърът можем да наричаме клиент. Някои от елементите на сайта се изпълняват от страна на клиента, като например HTML, CSS, JavaScript, картинки и всички други, които се свалят локално на компютъра на потребителя. От страна на сървъра се изпълняват елементите, които обработват заявките създадени от клиента към сървъра и връщат желаната информация, като например PHP, XML и други.

JavaScript може да се изпълнява и на сървъра. За целта се използва Node.js.

# MySQL

MySQL е най-популярната система за управление на SQL бази данни с отворен код. Тя се разпространява и поддържа от Oracle Corporation.

## Бази данни

Бази данни наричаме структурирано количество от данни. Бази данни наричаме всичко от най-обикновен списък за пазаруване до огромните количества информация в корпоративните мрежи. За да добавите, получите достъп и да работите с данни, които са съхранени в базите данни на компютъра, се нуждаете от система за управление на базите данни, като MySQL Server. Тъй като компютрите вече могат да боравят с огромни количества от данни, системите за управление на базите данни играят ключова и централна роля, било то като отделни приложения или като част от такива.

## MySQL базите данни са релационни

Релационните бази данни съхраняват данни в отделни таблици, вместо да поставят всичките данни в една голяма директория. За по-бърза работа структурите от бази данни са организирани във физически файловe. Логическият модел, с обекти като бази данни, таблици, изгледи, редове и колони предлага гъвкава програмна среда. Вие настройвате правила, управлявате връзките между различните полета с данни, като „one-to-one“, „one-to-many“, „unique“, “required” или „optional“, както и различни показалци между различните таблици. Базите данни налагат тези правила, така че с добре проектирана база данни вашите приложения никога няма да се сблъскат с несъвместими, дублиращи се, остарели или липсващи данни.

## SQL и MySQL

SQL е абревиатура, която означава „Structured Query Language” и както може би се сещате от името е програмен език. Всъщност това е най-често използвания програмен език, който се използва при достъп до бази данни. В зависимост от вашата програмна среда можете да влезете в SQL директно (например за да генерирате репорти), да вграждате SQL откъси в код изписан на друг език или да използвате специфични програмно-приложни интерфейси просто за да скриете синтаксиса на SQL.

## MySQL е софтуер с отворен код

„Отворен код“ означава, че е възможно за всеки да модифицира софтуера. Всеки желаещ може да свали MySQL софтуера от Интернет и да го използва, без да заплаща нищо. Ако искате може да изучите изходния код и да го промените така, че да отговаря на вашите нужди.

# Vue

Vue е прогресивен фреймуърк, създаден за разработка на потребителски интерфейс. В зависимост от целите на разработчиците, Vue може да бъде използван, както за фреймуърк, така и като библиотека от компоненти. Той се състои от главна библиотека, която се фокусира върху визията на апликациите и екосистема от допълнителни библиотеки, предоставящи на платформата множество функционалности.

Vue е създаден от Evan You като прогресивен JavaScript фреймуърк, а целта му е била да комбинира най-добрите функции на Angular и React.

Vue е предпочитан фреймуърк сред много от разработчиците по много причини. Той е лесен за научаване, има ясна структура и отлична документация, която го прави подходящ за използване, както от начинаещи, така и от напреднали.

Vue се използва и от големи компании като Gitlab, Alibaba, Xiaomi, Adobe, Nintendo, Euronews и много други.

## Предимства на Vue

* **Документация и разработка**

Vue.js има една от най-добре написаните документации, от която разработчиците могат да се учат бързо и структурирано. Това прави научаването и от начинаещите изключително лесно, в сравнение с React и Angular.

Разработчиците обичат да създават апликации с Vue, тъй като той им дава свобода и комфорт на работа, предлагайки всички компоненти, които биха били необходими.

* **Преизползване на кода**

Подходът, който Vue използва, базирайки се на компоненти, е вдъхновен от React. Кодът се пише под формата на компоненти, които могат да бъдат импортирани и преизползвани, когато и където е необходимо при създаването на апликациите.

* **Разработка на апликации за всички платформи**

Още едно предимство на Vue е неговата функция за създаване на приложения за всички платформи. Точно както при React, Vue позволява написването на един единствен код, който може да бъде използван на която и да е от съществуващите платформи и операционни системи.

* **Интеграции**

Разработчиците могат да интегрират Vue с други популярни структури, например React. Това им позволява да къстамизират проекта, спрямо техните нужди и изисквания. Благодарение на тези интеграции, Vue става популярен избор за разработчиците, тъй като може да бъде използван с голяма част от съществуващите уеб апликации.

Ако back end-а е разработен с JavaScript, програмистите биха могли да използват тази технология при създаването на други JavaScript апликации.

Разнообразието от компоненти позволява създаването на различни типове уеб апликации, както и модифицирането на съществуващите структури.

Освен React, Vue може да използва HTML, благодарение на MVVM архитектурата, която позволява двустранна комуникация.

## Недостатъци на Vue

Въпреки всички предимства на Vue.js, той има и няколко недостатъка. Например, стабилността му е проблем още от обявяването на първата версия през 2015 година. Това означава, че платформата е чудесна при създаване на малки и лични проекти, но трябва да се внимава при работата с по-големи проекти, тъй като тази нестабилност, може да доведе до значителни загуби.

Vue не поддържа някои от важните плъгини, необходими на програмистите при разработката на апликациите, което означава, че понякога се налага и използването на други езици за програмиране при създаването на приложения.

## Библиотеки на Vue

* **Vue Router**

Vue Router се интегрира с Vue JS, улеснявайки разработката на апликации с една страница (Single Page Applications).

* **Vuex**

Vuex е библиотека за Vue JS апликации, който работи като централизиран склад за всички компоненти на апликацията.

* **Vue Loader**

Vue Loader позволява конвертирането на всички Vue компоненти във формат, наречен SFC (Single-File Component). Комбинацията от webpack и Vue Loader дава възможност за създаване на модерен, гъвкав и мощен front-end процес за създаване на Vue апликации.

* **Vue Server Renderer**

Vue Server Renderer улеснява разработката на JavaScript приложения, които работят, както на сървъра (back-end), така и на front-end, където се споделя и голяма част от кода на приложенията.

* **Vue Test Utils**

Vue Test Utils е официалната библиотека за тестване на Vue апликациите и техните компоненти.

* **Vue Dev-Tools**

Vue Dev-Tools е инструмент за отстраняване на грешки в приложенията, разработени с Vue.

## Компоненти на Vue

* **Vuetify**

Vuetify е библиотека за Vue UI, състояща се от множество насоки за създаване на потребителския интерфейс като форми, интеракции, различни ефекти и др. Освен това, не се изискват специални дизайнерски умения, тъй като всичко необходимо за създаването на добре изглеждащи апликации е на един клик разстояние.

* **Bootstrap Vue**

Bootstrap Vue е комбинация от Vue.js и най-популярната фронт-енд библиотека, Bootstrap.

Bootstrap Vue помага при разработката на респонсив дизайн, ориентиран най-вече към мобилната версия на апликацията. Този инструмент е част от категорията на Front-end Frameworks, добавяйки всички предимства на Bootstrap към Vue. Елементи като редове, колони и др., стават достъпни като компоненти на Vue.

* **Element**

Element е библиотека от компоненти, базирана на Vue 2.0, насочена към разработчици и дизайнери, включваща голям набор от ресурси. Този инструмент е построен за разработка на уеб и десктоп апликации, но може да бъде използван при създаването на приложения за всички платформи.

* **Quasar**

Quasar е библиотека от компоненти, позволяваща на разработчиците бързо да създават респонсив сайтове, прогресивни уеб апликации, както и мобилни приложения за Android и iOS, използвайки Cordova или Capacitor.

## Свързани инструменти

* **Javascript & Babel**

Апликациите на Vue могат да бъдат създадени ефективно, с използването на Javascript ES5, стандарт, който почти всеки съществуващ браузър поддържа.

За създаване на безпроблемно преживяване с Vue, и за да се възползват от всички възможности на новите браузъри, разработчиците биха могли да създадат Vue апликациите, използвайки най-новите стандарти на Javascript.

Също така обаче, трябва да се намери и начин за поддържане на по-стари версии на браузърите, за да бъде достъпен продукта за всички потребители.

Babel е инструмент, който помага за това, преобразувайки новите функции в стандартни такива.

* **Webpack**

Ако кодът е написан на различни модули, например различни JavaScript файлове, Webpack може да ги обедини в един единствен файл, който браузъра може да прочете.

Също така Webpack позволява трансформиране на кода и може да бъде използван за оптимизация на апликацията със серия от плъгини.

* **TypeScript**

Настоящата версия на Vue, Vue 3.0, е написана изцяло на Typescript, което не означава, че този език трябва да бъде използван при създаването на проекти с Vue, но означава, че за да разбират разработчиците същността на платформата, първо трябва да разбират Typescript.

* **Nuxt.js**

Nuxt.js дава възможност за разработка на качествени апликации с Vue, чрез различни функции като платформата предоставя, например SSR(server side rendering) и SEO. Nuxt е базиран на повече от 50 различни модула, които правят разработката на апликации бързо и лесно.

Vue.js е лесен за използване фреймуърк, който се разраства бързо и се адаптира към имплементацията и разработката на апликациите. Vue е подходящ за използване от начинаещи в програмирането, тъй като е лесен за научаване и дава достъп до подробна документация, от която всеки желаещ може да се учи.

Vue.js дава достъп до богата екосистема, която се нарежда сред трите най-добри JavaScript front-end структури.

# Django

Django е безплатен framework с отворен код, написан изцяло на Python. Технологичният framework позволява да се разработват уебсайтове по-бързо и по-лесено. Django е подходящ за всеки етап от разработката на дадено приложение – потребителска верификация, администриране на съдържанието, RSS feed настройки и още много други. По този начин той е напълно достатъчен за процеса на разработка от начало до край.

## Предимства на Django

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

* **Помощ от общността**

Django доминира в ИТ индустрията през последните 13 години, което го прави изключително популярен framework. Хиляди разработчици си сътрудничат, за да поправят всички проблеми или дефекти, които могат да възникнат. Django винаги се подобрява и развива, тъй като се добавят нови библиотеки, за да бъде най-добрата платформа на пазара.

* **Ефективност и скорост**

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

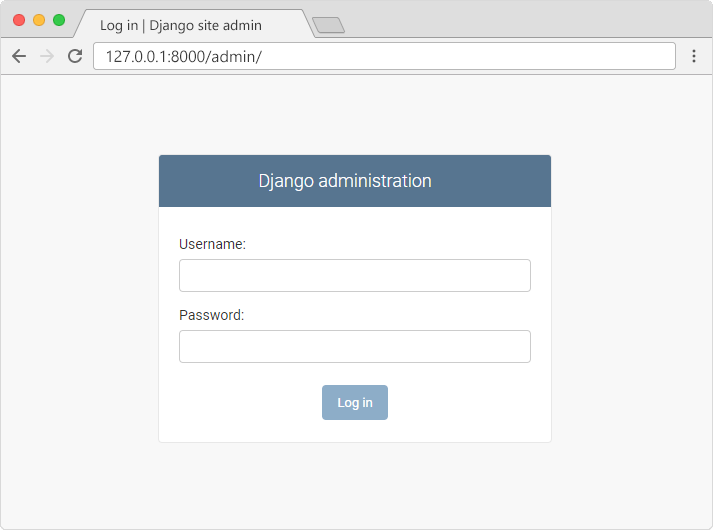
Когато обсъждаме уеб приложенията, скоростта става важен фактор, който трябва да се вземе предвид. Django позволява на потребителите бързо да създават динамични уеб приложения.

## Django admin panel

Админ панелът на Django е вграден административен интерфейс, който позволява управление на данните в Django проект. Той позволява да създавате, редактирате и триете данни от базата данни без да е необходимо да пишете ръчен SQL код.

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

*python manage.py createsuperuser*



# СЕРИАЛИЗАТОРИ И ИЗГЛЕДИ (VIEW)

Сериализаторите и изгледите (views) са ключови компоненти във фреймуърка Django за изграждане на уеб приложения.

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

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

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

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

Сериализаторите и изгледите в Django могат да бъдат написани на български език, като е важно да се запази еднакъв стил на програмиране и да се спазват добрите практики за оформление на кода. В Django също така съществуват и документация и ръководства на български език, които могат да бъдат полезни при работа с тези инструменти. В Django изгледите са функции или класове, които обработват HTTP заявки, получени от клиента. Те извличат или обработват данните, необходими за отговор на заявката и връщат HTTP отговор във формата на HTML, JSON, XML и други. Изгледите могат да бъдат класове или функции, като в последния случай се използва декоратора @api\_view, за да се конвертират функциите в изгледи, които могат да бъдат използвани в Django REST Framework.

# REST

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

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

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

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

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

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

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

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

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

* Ресурсно ориентирано програмиране - REST се базира на използването на ресурси като централен елемент на архитектурата. Всяко действие в уеб приложението може да бъде извършено чрез HTTP заявка към определен ресурс.
* Използване на HTTP методи - REST използва основните HTTP методи като GET, POST, PUT и DELETE, за да комуникира с ресурсите в уеб приложението. Например, GET заявката се използва за извличане на информация от ресурс, а POST заявката се използва за добавяне на информация към ресурс.
* Използване на URL адреси - REST използва URL адреси за идентифициране на ресурсите в уеб приложението. URL адресът на ресурса трябва да бъде постоянен и лесен за разбиране.
* Предаване на данни във формат JSON или XML - REST използва форматиране на данните като JSON или XML, за да осигури ефективно предаване на информацията между клиента и уеб услугата.
* Независимост от състоянието - REST е "безсъстоятелен" архитектурен стил, което означава, че всяка HTTP заявка към ресурса е независима от предишните заявки. Това прави REST уеб услугите много скалируеми и лесни за управление.

# REST API

API означава интерфейс за програмиране на приложения. Всеки софтуер със специфична функция се нарича приложение, когато се обсъждат API. Можете да сравните API интерфейса със споразумение за услуга между две програми, които определят заявки и отговори. Използвайки набор от дефиниции и протоколи, API позволяват на два софтуерни компонента да комуникират и обменят информация.

REST означава прехвърляне на представително състояние. Това е архитектурен стил, който дефинира набор от команди за създаване на уеб услуги. REST API прави достъпа до уеб услуги лесен и гъвкав без обработка.

REST технологията обикновено се предпочита пред по-стабилния Simple Object Access Protocol (SOAP), защото REST използва по-малко честотна лента и е прост и гъвкав, което го прави по-подходящ за използване в интернет.

Използва се за извличане или предоставяне на информация от уеб услуга. Цялата комуникация, извършвана чрез REST API, използва само HTTP заявки. За клиентите да имат достъп до данни на сървъра, REST дефинира набор от функции като GET, PUT, DELETE и др.

## REST API

REST API изпълнява същата основна задача като сърфирането в интернет. Клиентът използва API, за да комуникира със сървъра, когато е необходим ресурс.

Общата процедура за всяко извикване на REST API:

* Клиентът изпраща заявка до сървъра. Клиентът форматира заявката според документацията на API, така че сървърът да може да я разбере.
* Сървърът проверява самоличността на клиента и потвърждава, че клиентът е упълномощен да изпрати тази заявка.
* Сървърът получава заявката и след това я обработва вътрешно.
* Клиентът получава отговор от сървъра. Отговорът на клиента показва дали заявката е била успешна или не, заедно с исканата информация.
* В зависимост от това как API е проектиран от разработчиците, подробностите за заявката и отговора на REST API варират леко.

## Предимствата на REST API

Ключовата характеристика на REST API е, че сървърите не съхраняват клиентска информация между заявките. Подобно на URL адресите, които въвеждате в браузъра си, за да посетите уебсайт, клиентите изпращат заявки до сървъра.

REST API помагат да създавате нови приложения или лесно да интегрирате вашия софтуер с други приложения. Можете да постигнете това, без да се налага да пренаписвате изцяло кода, като правите корекции на ниво API. REST API предлагат пет основни предимства:

* Integration: Новите приложения могат да бъдат интегрирани с настоящите софтуерни системи, използвайки REST API. Разработката отнема много по-малко време, тъй като REST API могат да се възползват от вече съществуващи кодове и да избягват писането на кодове от нулата.
* Innovation: С въвеждането на ново приложение цели индустрии могат да се променят. Бизнесът може да действа бързо и да улесни бързото внедряване на иновативни услуги. Те могат да постигнат това, без да се налага да пренаписват изцяло кода, като правят корекции на ниво API.
* Разширяване: REST API могат да помогнат за разширяване на възможностите на вашето приложение и следователно на вашия бизнес. Например API за карти позволява интегрирането на картографски данни в уебсайтове, Android, iOS и др. Всяка компания може да предостави подобен достъп до своите вътрешни бази данни, като използва безплатни или платени API.
* Лесна поддръжка: API създава шлюз между две системи. От всяка система се изисква да приложи вътрешни корекции, за да избегне повреда на API. По този начин всички предстоящи модификации на кода от едната страна няма да засегнат другата.
* Създайте нови бизнес възможности: Бизнесът винаги има нови възможности. Благодарение на REST API, те могат да задоволят нуждите на своите клиенти на различни платформи.

Поради тези предимства и визията на Vtiger да изгради CRM с възможност за персонализиране, за да отговори на вашите бизнес нужди, Vtiger адаптира комбинация от Vtap и REST API, за да улеснят клиентите да персонализират и разширят CRM с интеграции според техните нужди.

В заключение, REST API в момента са по-бързи, по-леки и по-мащабируеми от други API, защото се основават на принципи, които могат да се прилагат при необходимост. Това ги прави идеални за разработване на IoT и мобилни приложения.

## Изграждане на RESТ API

Използвайте версии - когато правите промени в RESTful API, е важно да използвате версии, за да не нарушавате съвместимостта със старите клиенти. Можете да добавите номер на версията в URL адреса на API-то.

Използвайте HTTP статус кодове - HTTP статус кодовете са важни за указване на резултата от изпълнението на заявката към RESTful API. Например, 200 означава успешна заявка, 400 означава грешка в заявката, а 404 означава, че ресурсът не е намерен.

Използвайте HTTP хедъри - HTTP хедърите могат да бъдат използвани за предаване на допълнителна информация за заявката или отговора. Например, може да използвате хедър за авторизация, който съдържа токен за достъп.

Използвайте кеширане - кеширането може да ускори отговора на RESTful API, като се избягва излишното извличане на информация от базата данни. Можете да използвате HTTP хедъра "Cache-Control" за указване на времето, за което резултатът на заявката може да бъде кеширан.

Тествайте API-то - тестването на RESTful API е важна стъпка за увереност в правилната му работа. Можете да използвате инструменти като Postman или curl за изпращане на заявки към API-то и проверка на резултатите.

# SOAP

SOAP (Simple Object Access Protocol) е комуникационен протокол, който позволява предаване на данни между различни платформи, езици за програмиране и операционни системи. Използва се главно в уеб услуги за изпращане и получаване на информация през интернет.

## Как работи SOAP

SOAP работи чрез XML формат на съобщение, който определя структурата и съдържанието на съобщението. SOAP съобщението се състои от заглавка и тяло. Заглавката съдържа допълнителна информация за съобщението, като адреса на получателя и вида на действието, което трябва да се извърши. Тялото съдържа действителното съобщение, което трябва да бъде предадено. SOAP използва различни транспортни протоколи, като HTTP, SMTP и FTP, за изпращане и получаване на съобщения. В допълнение, SOAP също използва различни стандарти за сигурност, като SSL и TLS, за да гарантира поверителността и целостта на предаваните данни.

## Как работи протоколът SOAP

SOAP е комуникационен протокол, използван за обмен на структурирана информация в мрежата. Този протокол се използва за комуникация между приложенията, което им позволява да комуникират и обменят информация ефективно и сигурно.

Този протокол е базиран на XML формат и позволява на различни системи да комуникират и споделят информация независимо от използвания език за програмиране и операционната система, на която работи.

В обобщение, SOAP протоколът работи по следния начин:

* Клиентът изпраща заявка до сървър, използвайки протокола SOAP.
* Сървърът получава заявката и я обработва, генерирайки отговор, който също се изпраща чрез SOAP протокола.
* Клиентът получава отговора от сървъра и го обработва.
* Протоколът SOAP използва структура на съобщението, дефинирана в XML формат. Всяко съобщение се състои от заглавие и тяло. Заглавката съдържа информация за съобщението, като адреса на получателя и вида на съобщението. Тялото съдържа специфичната информация, която се предава.

В допълнение, SOAP протоколът позволява използването на различни транспортни механизми, като HTTP, SMTP или TCP. Това означава, че може да се използва за комуникация в различни среди и ситуации.

В заключение, SOAP протоколът е основен инструмент за комуникация между приложенията в мрежата. Позволява ефективна и сигурна комуникация, независимо от използвания език за програмиране и операционната система, на която работи. Освен това неговата структура, базирана на XML, и способността му да използва различни транспортни механизми го правят много гъвкав инструмент.

Накратко, Simple Object Access Protocol (SOAP) е комуникационен метод, който позволява на софтуерните програми да комуникират през Интернет. Той работи чрез изпращане на XML-кодирани съобщения, които могат да съдържат всякакъв вид информация, от текст до изображения и прикачени файлове. Въпреки че SOAP до голяма степен е заменен от други по-модерни протоколи, като REST и gRPC, той все още е полезен инструмент за много софтуерни приложения и наследени системи.

# PHP

За да стартира един PHP скрипт, ви е нужен уеб сървър и софтуера на езика – PHP интерпретатор.

PHP интерпретаторът се използва с уеб сървъра Apache, като обикновено към тях е присъединена и системата за управление на бази данни MySQL. „Троицата“ Apache/PHP/MySQL(MariaDB) е най-широко използваната в интернет конфигурация за разработване на динамични уеб страници.

Ако желаете да работите с тези програми на своя личен компютър, първо трябва да ги изтеглите и да ги инсталирате – сървъра Apache, PHP интерпретатора и MySQL(MariaDB) базата данни. Това може да направите от съответните сайтове: www.apache.org, www.php.net, www.mysql.com, [www.phpmyadmin.net](http://www.phpmyadmin.net).

## Софтуерен пакет XAMPP

Вместо да сваляте тези приложения едно по едно, да ги настройвате и да ги конфигурирате за съвместна работа, може да изтеглите пакета XAMPP. XAMPP е дистрибуция на Apache, който съдържа в себе си всичко необходимо – Apache, MariaDB, PHP, phpMyAdmin, FTP и др. елементи и е настроен за лесна инсталация. XAMPP има версии за работа под Linux, Windows и други ОС.

# Реализация

# ПЪРВИ СТЪПКИ НА ПРОЕКТА:

## СЪЗДАВАНЕ НА ПРИЛОЖЕНИЕ:

* В settings.py, секция INSTALLED\_APPS регистрираме приложението
* В папката на приложението създаваме празен файл urls.py
* В главния urls.py пренасочваме проследяване на url-ите към нашето приложение:
* Добавям include в импорта след path: from django.urls import path, include
* Добавям още един път за проследяване

*from django.contrib import admin  
from django.urls import path, include  
from django.contrib.auth import views as auth\_views  
  
urlpatterns = [  
 path('admin/', admin.site.urls),  
 path('', include('main.urls')),  
]*

* В новия urls.py копирам от стария и коригирам съдържанието:

*from django.urls import path  
from . import views  
  
urlpatterns = [  
 path('', views.SiteLoginView.as\_view(), name='login'),  
 path('login', views.SiteLoginView.as\_view(), name='login'),  
 path('client', views.clients\_view, name='client'),  
 path('staff', views.staff\_view, name='staff'),  
 path('logout', views.logout\_user, name='logout'),  
  
 path('api/requests/<int:uid>/<int:sts>/', views.RequestSerializerView.as\_view()),  
 path('api/log/<int:uid>/<int:rid>/', views.LogSerializerView.as\_view()),  
 path('api/user/<int:u\_id>/', views.UserSerializerView.as\_view()),  
 path('api/create\_request/', views.RequestCreateAPIView.as\_view()),  
 path('api/create\_log/', views.LogCreateAPIView.as\_view()),  
 path('api/update\_status/', views.RequestUpdateStatusAPIView.as\_view()),  
  
]*

**ИЗПОЛЗВАНЕ НА ШАБЛОНИ:**

* В папката main създавам папка templates. Името е важно да е точно такова. В тази папка създавам папка със същото име като това на приложението.
* В новосъздадената папка създаваме шаблони (.html файл) с име client.html; login.html; staff.html

**НАСТРОЙВАНЕ НА DJANGO ЗА РАБОТА С БД MYSQL:**

* При инсталиране Django установява по подразбиране настройки за работа с база данни sqlite. Така, без необходимост от допълнително конфигуриране, потребителят може с лекотата да съхранява и извлича на данни от БД. Sqlite е подходящa за малки уеб приложения, но за по-големи ще доведе до проблеми с производителността. За да се избегнат това, първата стъпка е да изберете подходяща база данни, способна да обслужва стотици или дори хиляди едновременни посещения. За тази цел може да се използва база данни MySQL.

**НАСТРОЙВАНЕ НА MYSQL БАЗА ДАННИ:**

* Инсталирам MySQL и достъп до него чрез XAMPP.
* Създаваме нова (празна) база данни.
* Създаваме нов потребител и му даваме права върху създадената база данни.

**НАСТРОЙВАНЕ НА DJANGO ПРОЕКТА:**

* Инсталирам MySQL клиент - pip install mysqlclient
* Настройване на връзката към БД - Отваряме settings.py на проекта и настройваме DATABASES:
* DATABASES = {  
   *'default': {  
   'ENGINE': 'django.db.backends.mysql',  
   'NAME': 'dzi\_13',  
   'USER': 'admin',  
   'PASSWORD': 'admin\_2024',  
   'HOST': 'localhost',  
   'PORT': '3306',  
   'OPTIONS': {'init\_command': "SET sql\_mode='STRICT\_TRANS\_TABLES'"}  
   }  
  }*

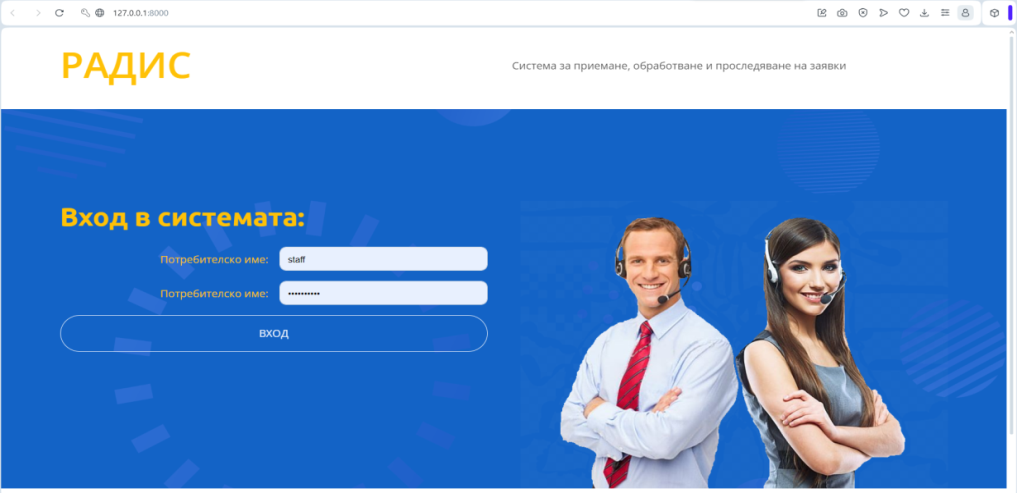
# Оторизация на потребител

За оторизация на потребителите решавам да използвам оторизационната система на Django. От една страна тя е надеждна и сигурна, от друга страна едно от големите предимства е, че предлага решения на често срещани проблеми, каквито са оторизирането и аутентификацията на потребител. Джанго ни предлага готови изгледи и форми, които ние ще използваме. Джанго има три нива: потребител, служител и администратор. Потребителите от своя страна могат да бъдат: активни, потребител със статус на персонал и потребител със статут на супер потребител.

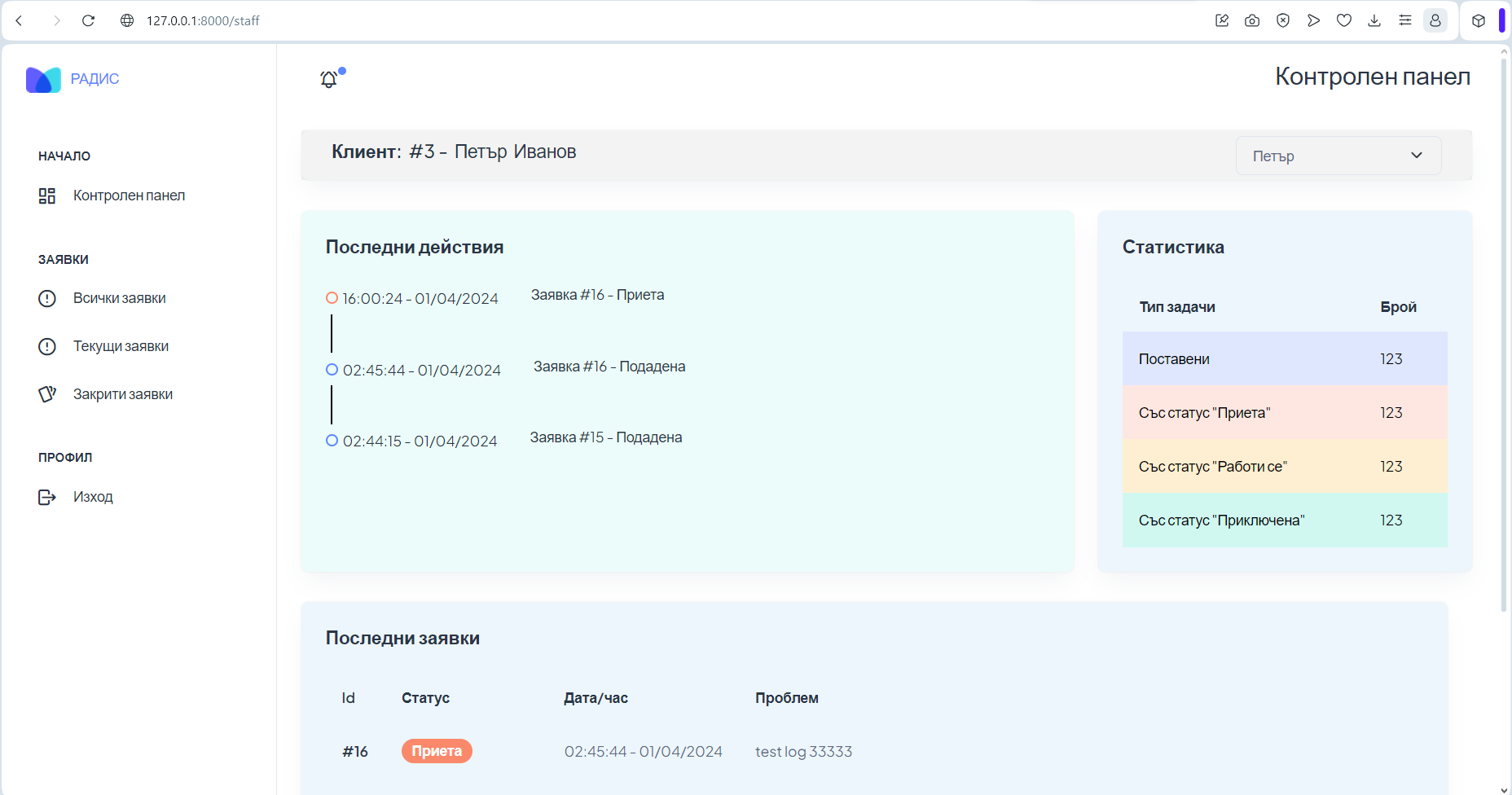
* Създаваме изглед Login

*class SiteLoginView(DataMixin, LoginView):  
 form\_class = LoginUserForm  
 template\_name = 'main/login.html'*

* Съответния интерфейс на login.html



* Логнат потребител



* Налагаме цветовата схема като наложена функционалност въргу дизайна



# Списък на използваната литература

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