Table of Contents

[УВОД 2](#_Toc164724098)

[ГЛАВА 1: ТЕОРЕТИЧНА РАМКА 4](#_Toc164724099)

[Същност и предимства на електронния бизнес 4](#_Toc164724100)

[Характеристики на сайта за електронен бизнес 8](#_Toc164724101)

[Оптимизация на сайт 16](#_Toc164724102)

[ГЛАВА 2: ИНСТРУМЕНТИ И ТЕХНОЛОГИИ 21](#_Toc164724103)

[SQL 21](#_Toc164724104)

[Cascading Style Sheets 22](#_Toc164724105)

[бази данни. 24](#_Toc164724106)

[Същност на базата данни 24](#_Toc164724107)

[Python 25](#_Toc164724108)

[HTML 26](#_Toc164724109)

[JavaScript 28](#_Toc164724110)

[Jinja 30](#_Toc164724111)

[Django 32](#_Toc164724112)

[BOOTSTRAP 33](#_Toc164724113)

[FORMS 35](#_Toc164724114)

[ГЛАВА 3: РЕАЛИЗАЦИЯ 39](#_Toc164724115)

[Етапи на проекта 39](#_Toc164724116)

[Концепция 40](#_Toc164724117)

[Функционална структура 44](#_Toc164724118)

[Резюме 45](#_Toc164724119)

[Пояснения 48](#_Toc164724120)

[Заключение 50](#_Toc164724121)

[Използвана литература 51](#_Toc164724122)

# УВОД

За съвременното общество е характерна много голяма динамика на протичане на процесите. Основен двигател на промяната са информационните и комуникационните технологии. Тяхното развитие за последните десетилетния промени почти всеки аспект от живота ни. Те са в основата на нови системи за производство, комуникация и обучение, създават нови култури, ценности и знания.

Информационните и комуникационните технологии предоставят на потребителите нови възможности, които до преди няколко години бяха немислими и само в сферата на научната фантастика. Те коренно промениха битието на отделния човек, работата на институциите и правенето на бизнес.

С подобряване на комуникационната инфраструктура се разширява достъпът до ресурсите на Интернет. Броят на потребителите нараства успоредно с развитието на технологиите, което увеличава потенциалното икономическо въздействие от публикуваната онлайн информация.

Осъзнавайки предимствата на онлайн средата, много бизнеси се зародиха или пренесоха в Интернет. Формира се нов бизнес модел, чиято същност са непрекъснатите електронни връзки с клиенти, доставчици и партньори и възможността за управление на процесите в реално време.

Ефективността и успехът на електронния бизнес до голяма степен зависят от софтуера, чрез който ще се реализира. Софтуерът са динамични уеб базирани приложения, които могат да са разработени на базата на богат инструментариум от съществуващи уеб технологии и езици за програмиране. За да се определи кои са най-подходящи за конкретен проект е необходимо да се направи предварителен анализ и да се изберат тези, които притежават характеристики, даващи възможност за реализиране на поставената цел и изпълнение на формулираните задачи.

Целта на настоящата работа е да покаже предимствата от реализирането на онлайн бизнес услуга и да се разработи сайт за конкретна такава – коли под наем.

Задачите за постигане на целта са следните:

* Запознаване със същността и основните предимства на електронния бизнес;
* Представяне на естеството и характеристиките на сайта за реализиране на електронен бизнес;
* Разкриване на значението на оптимизацията на сайта (SEO) за ефективен онлайн бизнес;
* Анализ на технологии за реализация на сайт за коли под наем;
* Проект на база данни (Data Layer), на сървиз (Service Layer) и на уеб приложение (Web Layer);
* Разработване на потребителски интерфейс;
* Реализация на модулите в сайта.

Дипломната работа е структуриране в три глави.

Глава първа очертава теоретичната рамка. В нея се изясняват същността и основните предимства на електронния бизнес, представят се естеството, характеристиките и изисквания, на които трябва да отговаря на сайтът. Разгледани са възможностите неговото популяризиране сред онлайн потребителите и привличане на нови купувачи.

Втора глава съдържа кратък преглед на някои от софтуерните инструменти и технологии, използвани за създаване продукти в сферата на електронния бизнес.

Третата глава накратко описва по-важните стъпки в разработването на сайт за коли под наем.

# ГЛАВА 1: ТЕОРЕТИЧНА РАМКА

## Същност и предимства на електронния бизнес

Интернет оказва влияние върху всички страни на нашия живот, формирайки ново универсално пространство за обмен на информация. Става дума за дълбоки и трайни изменения, за своеобразна революция в епохата на глобалната мрежа. В деловата сфера тази революция се нарича електронен бизнес. Тя променя характера на бизнеса и формира нов бизнес модел, чиято същност са непрекъснатите електронни връзки с клиенти, доставчици и партньори и възможността за управление на процесите в реално време. Компаниите все по широко използват Интернет като средство за делови комуникации и за осъществяване на своя бизнес.

Идеята за електронен бизнес се появява още преди Интернет да стане популярен. През 70-те години на 20-ти век, Е-бизнесът е бил популярен например за финансовите мрежи, които използват правилни хардуерни и софтуерни решения. Електронният обмен на данни (Electronic Data Interchange - EDI) съществува още преди Интернет да се използва за тази цел. Но без Интернет, е-бизнесът нямаше да бъде толкова широко разпространен. Частните мрежи, използвани през 70-те и 80-те години на 20-ти век са били много скъпи за малките предприятия и недостъпни за персонално използване.

За първи път през 1997 година IBM (ibm.com) употребява понятието е-Бизнес (e-Business). До този момент в Интернет се е употребявало само понятието електронна търговия (e-commerce), защото преди това продажбите са били единственото нещо което генерирало до печалби.

Промяната на термина отразила новите бизнес процеси, електронната търговия вече далеч не била единствения тип бизнес в мрежата. В последствие самият достъп до мрежата стана един от многото доходоносни бизнеси, както и предоставянето на хостинг пространство, онлайн поръчката за изработка на уеб сайтове, рекламата, информационните сайтове и безплатните уеб базирани електронни пощи, привличащи огромна аудитория и генериращи приходи от реклама, сайтовете за изработка на определени продукти по поръчка, електронното банкиране, поддръжката на фирмени и промоционалните сайтове, е-игри, онлайн агенции и много други.

Електронният бизнес е бизнес, който се извършва посредством електронни медии; това включва всеки бизнес, чиито приходи частично или изцяло се дължат на ползването на информационни и комуникационни технологии (ИКТ). Следователно електронният бизнес, наричан също "е-бизнес", може да се определи като използването на информационните и комуникационните технологии (ИКТ) в помощ на различни дейности на бизнеса. На същия принцип електронната търговия е търговия, която се фокусира върху използването на ИКТ, за да улесни дейностите и взаимоотношенията в този процес. Но докато електронната търговия представлява обмен на продукти и услуги между фирми, групи и отделни лица и би следвало да се разглежда като една от основните дейности на всеки бизнес, то самият електронен бизнес е нещо много по-глобално и всеобхващащо.

Най-общо електронният бизнес се дефинира като процес на използване на електронните технологии за осъществяване на бизнес. Електронният бизнес позволява да се изменят практически всички процеси, протичащи в съвременния бизнес, интегрирайки ги в едно цяло. Потребителите могат да търсят, да поръчват и заплащат стоки, използвайки Интернет, да се обменя информация за стоки и услуги с други потребители.

Според J. Strauss и R. Frost основните съдържателни елементи при електронният бизнес се представят по следния начин:

EB = EC + BI + CRM + SCM + ERP,

където:

* EB (E–business) - електронен бизнес;
* EC (E- commerce) – електронна търговия;
* BI (Business Intelligence) – бизнес проучвания;
* CRM (Customer Relationship Management) – управление на взаимоотношенията с клиентите;
* SCM (Supply Chain Management) – управление на доставките;
* ERP (Enterprise Resource Planning) – ресурсно планиране във фирмата.

Електронният бизнес обхваща всичко, което е свързано с електронния обмен на информация и по някакъв начин променя вече съществуващите бизнес модели като изменя самия начин, по който функционира дадена компания. По електронен път се извършва не само сделката, но и останалите аспекти от бизнеса, тоест с помощта на новите технологии се прилагат нови принципи на действие на самия бизнес.

Според Д. Амор, в зависимост от комуникационните портали електронният бизнес може да се раздели на три области:

1-ва област) В рамките на една организация, като се използват т.нар. интранет мрежи. Интранет мрежата ползва Интернет стандартите за електронни комуникации. Потребителите в итранет могат да видят специфични за дадена организация web –сайтове. Тези сайтове са отделени от останалата част от света посредством защитни мрежи- firewalls и други мерки за сигурност. Потребителите извън организацията не могат да виждат тази част от информацията.

2-ра област) Това са сделките от типа business-to-business (B2B), които се осъществяват посредством екстранет. Екстранет се състои от две мрежи интранет, свързани посредством Интернет и при това положение 2 компютъра могат да виждат взаимно данните си. Обикновено само една малка част от информацията, тази, която е необходима за работа, може да се вижда от партньора. Мрежите от типа В2В съществуват доста време преди появата на Интернет. Много организации имат частни мрежи за връзка с техните партньори и клиенти.

3-та област) Това е областта бизнес-клиент (business-to-consumer B2C). това е най-известната мрежа, която повечето хора са виждали в Интернет. Много web сайтове предлагат стоки и услуги на всеки, който ги посети. Традиционно под е-търговия повечето хора разбират точно това – продажба на продукти в web, но тя включва много повече от това.

Технически няма разлика между интранет, екстранет и Интернет. Първите 2 мрежи са част от Интернет, до които имат достъп само определени потребители. Електронният бизнес, който може да се осъществява е един и същ. При работа с ограничен брой потребители е по-лесно да се налагат определени стандарти.

Силните страни на електронния бизнес се обуславят от силните страни на Интернет:

* Интернет е налична по целия свят, 24 часа в денонощието, 7 дни в седмицата;
* Работи се лесно и за крайния потребител цените на транзакциите са ниски;
* За търговците в Интернет разходите са малки, в сравнение с тези при традиционните дистрибуционни канали;
* Интернет позволява двупосочна комуникация и е базирана на отворени стандарти. Двупосочната комуникация позволява на клиентите да получават бърз отговор на своите запитвания, а отворените стандарти дават възможност за съвместно използване на web сайтове и услуги;
* Намаляване на разходите за човешки ресурси. С използването на web сайт, който да отговаря на въпроси на клиентите се намалява броя на телефонните обаждания и в същото време клиентите получават необходимата им информация 24 часа в денонощието;

Проблемът за нарушаване на сигурността при работа в Интернет се решава посредством използване на криптиране, цифрови подписи, firewalls софтуер и процедури за сигурност.

* В зависимост от участниците в бизнес процеса се формират различни взаимоотношения.
* Предимствата на електронния бизнес са много и могат да бъдат обобщени по следния начин:
* Глобална достъпност и продажби – компаниите могат да увеличат броя на своите клиенти и предлагани продукти;
* По-добри бизнес контакти – компаниите, които търгуват по между си могат да установят по-сериозни контакти чрез интернет комуникация;
* Безплатни мостри – посредством web могат да се доставят по-бързо и по-лесно мостри на различни видове предлагани продукти;
* Намалени разходи – за предлаганите продукти, посредством динамично определяне на продажните ценни;
* Проблеми с доставка – намаляват се проблемите, свързани с транспортирането на продуктите;
* Време за достигане до пазара – то се намалява и се получава по-бърза реакция за променящите се изисквания на клиента;
* Лоялност на клиента – тя се увеличава и се предлагат по-качествени услуги, посредством по-лесен достъп до най-актуалната информация.

Фирменият сайт може да изпълнява и ролята на виртуален офис, който е достъпен за посетители от целия свят. Това предоставя пред бизнеса неограничен пазар, но и налага изискването сайтът да се поддържа поне на английски език, а при наличие на специфични групи потребители и на техния език. Чрез сайта може да се осъществяват следните дейности:

* представяне на продукти/услуги;
* отговор на възникнали въпроси;
* възможност за достъп до систематизирана подробна информация;
* предоставяне на каталози, доклади, техническа документация;
* размяна на контактна информация и планиране на срещи;
* помагане при вземане на решение;
* сключване на сделка;
* извършване на разплащане;
* проучване на потребителското мнение.

Много често се слага знак на равенство между електронния бизнес и електронната търговия, но в разликата е, че “електронният бизнес има за цел да използва преимуществата на Интернет, както за разширяване възможностите на традиционния бизнес, така и за създаване на нови, виртуални видове бизнес”, докато “електронната търговия е само едно от направленията на електронния бизнес, както електронния франчайзинг или електронния маркетинг”.

За стартирането на печеливш онлайн бизнес са необходими минимални инвестиции. Идентичността, която създава една организация се определя от нейния сайт и отношенията, които създава с посетителите му. Много често самите потребители търсят търговците и осъществяват контакти, чрез информацията, която е публикувана на фирмения сайт, справочни издания и други. Онлайн комуникацията с клиентите е бърза и лесна. Мрежата позволява да се прескочат традиционните дистрибуторски вериги, съществуващи между производител и потребител, като производителите могат да продават директно на крайните клиенти.

## Характеристики на сайта за електронен бизнес

Категорията електронен бизнес, която има най-голяма популярност из между всички е електронната търговия. Тя генерира най-голям обем финансови транзакции онлайн. Според българското законодателство електронна търговия е предоставянето на услуги на информационното общество. Услуги на информационното общество са такива услуги, включително предоставяне на търговски съобщения, които обикновено са възмездни и се предоставят от разстояние чрез използването на електронни средства след изрично изявление от страна на получателя на услугата.

За да се реализира електронната търговия е необходимо да са налице няколко фактора. На първо място това са обектите на търговия, т.е. стоките и услугите, които да се предлагат. На второ място са субектите, които реализират процеса – продавачи, купувач и посредници, които изпълняват различни функции – финансови институции, доставчици и други. На трето място са технологиите, които да необходими за осъществяване на процеса – търговски, информационни и комуникационни.

Електронна търговия представляват съвкупност от всички сделки, които са реализирани по електронен път онлайн. Тя включва технологии и бизнес процеси, електронен обмен на данни, реализиране на финансови транзакции, мениджмънт на доставките и други.

Този ви търговия е възможна благодарение на фирмите и потребителите, които приемат новият начин и методология на реализиране на бизнес. Процесът се реализира чрез електронни взаимодействия, които изместват изискванията за близко физическо присъствие при осъществяването на търговията и на покупко-продажбата на стоките.

* В общи линии, съществуват четири фази на електронната търговия, които са:
* Информационна фаза - реклама на продукта, избор на продукта;
* Договорна фаза - поръчка на продукта, плащане на продукта;
* Фаза Доставка - реализиране на продукта, започване на потреблението;
* Фаза Сервизиране - модификация на продукта, замяна на дефектирал продукт.

Предимствата на електронната търговията, спрямо традиционната са:

Неограничени възможности от страна на клиента за проучване на пазара – електронната търговия позволява на клиента да открива, разглежда и поръчва стоките, които се предлагат двадесет и четири часа в денонощието, 365 дни в годината. Разнообразието на стоките, предлагани в различните търговски сайтове е огромно.

Глобален достъп – На практика всички интернет потребители (потенциални клиенти) от целия свят имат достъп до съответния сайт. Това значително облекчава съответната компания в аспект развитие на офисна инфраструктура и разкриване на магазини, спестяват се и човешки ресурси:

* Складова база - наемането и поддръжката на складови пространства не е необходимо, особено когато продажбата не е на едро.
* Удобство за клиента - Доставката на закупената от клиента стока е „до вашата врата”.
* Разплащането е облекчено – става по мрежата.
* Значително по-ниски цени на стоки и услуги – на базата на горе споменатите предимства, които са в непосредствена връзка с повишената online конкуренция, чувствително по-ниските режийни разходи и създадените удобства за клиента.
* Рекламата – в мрежата тя е по-евтина от средата на конвенционалните медии - радио, телевизия и преса.

Електронният магазин е организационна форма на проявление на електронната търговия в нейния класически вид - покупко-продажба на стоки и услуги в Интернет. Електронният магазин представлява компания, осъществяваща търговия в Интернет с помощта на Web сайт. Първият български Интернет магазин (www.bgstore.com) е открит през 1998 година (вече не функционира).

Сайтът на електронния магазин съдържа каталог на стоките с техните описания, снимки и цени. Асортиментът на стоките може да варира от единици до десетки хиляди наименования. След като клиентът го е посетил може да разглежда предлагания асортимент от продукти, да сравнява цени и характеристики на различни модели, да филтрира и подрежда информацията по различни критерии. Потребителят поставя желаните от него стоки във виртуалната потребителска кошница - специална форма за онлайн поръчка. Всеки клиент има право за избира начина на заплащане на формираната стойност на поръчката и сумата за доставка. След като укаже тези параметри той потвърждава поръчката. Някои магазини предоставят възможност за проследяване етапите на изпълнение на поръчката.

След получаване на поръчката администрацията на магазина се свързва с купувача посредством електронната поща или по телефона за потвърждаване на поръчката и уточняване на условията по нейната доставка (ако това е необходимо). В зависимост от избраните условия, клиентът заплаща поръчката при получаване или извършва предплащане по някой от традиционните способи или с помощта на специализирани системи за Интернет плащания. Често Интернет магазините разполагат отзиви на купувачи и друга полезна за клиентите информация в специални раздели или при описанията на конкретни стоки.

Предимствата на електронните магазини пред традиционните се определят от удобствата и възможностите, които предоставят както на клиентите, така и на търговците. Основните такива са:

Разширяване на достъпа до стоки и услуги - в Интернет е възможно търговците да предлагат изключително широка гама от стоки, която физически не може да се представи в обикновен магазин, даже и в склад. Освен това е неизгодно за търговеца да държи наличности от множество артикули, равномерно разпределени географски - клиенти от места, където търговецът няма представителство могат да пазаруват от него по Интернет и да получават стоките, доставени до дома си.

Подробно информиране - електронният магазин предоставя на потенциалния купувач не само богат асортимент от стоки, но и подробно описание на всяка от тях със снимки.

Оптимизиране на доставките - голямо удобство е възможността клиентите да поръчват комплексни доставки от различни видове стоки от едно място. В реалния свят често се налага различните видове стоки да се поръчват от различни търговци с различни организации за доставка и различен ценови режим.

Достъп до услуги - множество услуги са специфични за Интернет и тяхното предлагане се разширява заедно с разширяване формите на плащане в Интернет.

Простотата на процедурите – за поръчка на продукт в електронния магазин не се изискват никакви допълнителни умения от страна на потребителите. Освен това липсват опашки в Интернет магазините, което също е не малко предимство.

Удобство и бързина - За да посети електронен магазин купувачът не ходи никъде, не чака за приемане и изпълнение на поръчката. Пазаруването по Интернет е безспорно удобство със своята бързина на регистрация, възможност за проследяване на резултата от поръчките, евентуално доставки на място. Онлайн магазините нямат работно време и са достъпни от всяко населено място.

Сравняването на цените на стотиците артикули, също е един от факторите, които влияят на хората. Пазарувайки онлайн, те имат възможност да разгледат няколко интернет магазина и да сравнят артикулите и техните цени, за да направят най-изгодната сделка. А в крайна сметка, всеки цели точно това с пазаруването – да намери качествен продукт на изгодна цена. Онлайн покупките са любим начин за снабдяване с нови вещи и заради постоянните отстъпки и промоции, които са лесно достъпни.

Според проучването на Shopping.fm, защо хората пазаруват онлайн, вместо директно в магазин, резултатите са следните:

58% - онлайн магазинът е на разположение денонощно;

54% - цените могат да се сравняват;

46% - онлайн разпродажби;

40% - спестяване на време.

Наред с големите предимства онлайн пазаруването има и известни недостатъци, например рискът от разминаване между представянето на стоката и нейните реални характеристики.

Интернет потребителят не прави практически никакви усилия, за да посети електронния магазин, което е голямо предимство. Но точно толкова усилия са необходими за посетителя на електронния магазин, за да го напусне и да отиде при конкурент. Ето защо Интернет магазинът трябва да има разбираем интерфейс и удобна система за навигация. На клиента трябва да се предоставят лесни за достъп указатели към полезна информация за интересуващия го продукт, без да се претоварва с много бутони и препратки.

Според експерти успехът на съвременния интернет магазин може да се осигури с решаването на две основни задачи. Първата е да се привлекат в сайта на магазина достатъчен брой потенциални купувачи. Втората е колкото може повече посетители да преминат от категорията „потенциални купувачи” в категорията „реални клиенти”.

Първият етап при създаването на електронния магазин е разработването на неговата структура. Тя се определя преди всичко от спецификата на стоките и на купувачите. В един онлайн магазин най- важният раздел е продуктовият каталог. При неговото съставяне е необходимо да се придържаме към съществуващите професионални класификации на стоките, но това не винаги е приложимо. Например в парфюмерията има много сложна класификация на ароматите, която включва такива непонятни наименования като например папратови или алдехидни аромати. На обикновения потребител едва ли е понятно каква е миризмата на алдехидните парфюми. За него по-познати са названия като плодови или свежи аромати.

Информацията в онлайн магазина не се изчерпва само с продуктовия каталог. Необходимо е той да съдържа подробно описание за начина на извършване на поръчката, възможните методи на заплащане и условията за доставка, както и информация за контакти, при условие, че има някакви допълнителни въпроси. За улеснение на потребителите някои онлайн магазини имат чат със служител на когото могат да се задават въпроси в момента на пазаруването.

Първото впечатление за електронния магазин в посетителите се създава от дизайна на сайта. Още докато се гради визията за страницата трябва да се приложат най-добрите практики, които се познават от традиционните магазини, в които се пазарува всеки ден. В традиционния магазин клиентът избира даден продукт, плаща го и вече може да го вземе. Това е важно и при електронните магазини, в няколко лесни стъпки клиентът трябва да може да поръча продукта, да избере начин за плащане и доставка. Лесната навигация е наистина важна. Сайтът трябва да се поддържа и обновява редовно. Интернет страница, която не предлага нови продукти и предложения или това става по начин, по който посетителите не могат да се ориентират веднага от началната страница, отегчава посетителите.

Основна трудност при проектирането на електронния магазин е постигането на баланс между дизайн и функционалност. Задачата на дизайна е не само да осигури бързо възприемане на информацията, но и да предизвика у посетителя определени емоции. Стилният и спретнат дизайн повишава доверието към магазина и съответно се отразява позитивно върху продажбите. Кое ще се надделее дизайн или функционалност при изпълнение на конкретния проект зависи преди всичко от вида на предлаганите стоки.

Фактор с важно значение за всеки сайт и в частност за електронния магазин е неговият домейн. Най-често срещани са два типа наименования: домейни с ключови думи и brand-имена.

Домейн, съдържащ ключовите думи за съответния бизнес, е разумен и практичен избор, тъй като още с прочитането или чуването на домейна потребителите са наясно какво могат да очакват при посещение на уеб сайта. Такъв домейн има донякъде значение и за SEO стратегията, тъй като търсещите машини работят именно чрез търсене по ключови думи. Хипервръзките, сочещи към сайта от други страници, обикновено съдържат като текст самия уеб адрес на сайта и ако този адрес съдържа съответните ключови думи, това също се отразява положително на страницата в интернет.

От друга страна т.нар. brand-домейни представляват имена, които в много случаи нямат нищо общо с ключовите думи за съответния бизнес - това могат да бъдат странни, забавни или новоизмислени думи, но за сметка на това те обикновено са по-къси, по-звучни и имат за цел да провокират емоционално потребителя. Има огромно количество примери за подобни успешни наименования както в мрежата, така и отпреди ерата на интернет. Компанията Adidas е получила названието си от съчетанието на първите три букви на имената на основателя й Adolf (Adi) Dassler. “Google” също е несъществуваща в езиците дума. Първоначалното име на компанията е думата “googol”, която служи за название на число със сто нули след единицата (10 на стотна степен), но първият клиент, който написал чек сбъркал името, което пък се харесало повече на собствениците.

Трябва да се има предвид, че един сайт, чиито домейн е съставен от ключови думи, ще бъде по-лесен за развиване и популяризиране. Сайт с brand-домейн ще изисква повече време, усилия и средства за популяризиране, но след това е възможно да донесе повече дивиденти.

Определящ фактор за ползваемостта на електронния магазин от гледна точка на потребителите е удобството, което търговската площадка предлага на посетителя. Най-важна за ползваемостта е системата за навигация в стоковия асортимент.

В практиката се използват различни видове навигация: линейна, многостъпални каталози и навигация, основана на подборки. При линейната навигация се реализира последователен преход от една страница към друга. Както името подсказва, този тип навигация предполага наличието на пряка линия, която свързва всички страници от възела. Тя е приложима когато има ограничен асортимент на стоки. При магазини с голям стоков асортимент се използват многостъпалните каталози, които са по-сложен тип навигация. „Тя предполага наличието на основен (базов) каталог и подчинени подкаталози от по-ниско ниво, които обединяват стоките по някакви признаци. Навигацията на базата на подборките позволява на посетителя да посочи няколко параметъра, според които получава подборка от стоките, които отговарят на зададените от него изисквания“.

Създаването на функционална и професионално изглеждаща страница за интернет търговия е един от начините за привличана на онлайн купувачи. Но както в традиционните магазини, така и за един уеб магазин е важно да се изгради доверие от страна на клиентите. Трябва хората да познават даден магазин преди да започнат да пазаруват от него. Важно е създаването на страница “За Нас”, “За Клиентите”, която да обяснява условията и начина за покупка и задълженията на двете страни.

Сигурността на плащането и доставката са основната грижа, за да са спокойни клиентите. Трябва да се предлагат различни начини за плащане на клиентите, за да може те да изберат най-удобния за тях. Освен наложен платеж, все по популярни стават и електронните плащания, които са много по-бързи и пестят време, затова е добре да се предлагат и такава възможност за улеснение на клиентите.

За да се приемат електронни плащания трябва да може да се гарантират тяхната сигурност. Задължително условие е Интернет магазинът да има сертификат за сигурност (SSL) и самостоятелен IP адрес на страниците за осъществяване на плащания и формите за контакт и пускане на поръчка. Клиентът трябва да вижда, че данните и информацията, които праща са подсигурени и, че се използва криптиране на връзката.

Друг важен функционален елемент е така наречената пазарска количка (shopping cart), която потребителят използва. Тя трябва да позволява на всеки етап от пазаруването клиентът да следи своята поръчка, колко стоки е добавил, колко ще му струват заедно с доставката, бързо и лесно да се връща в каталога и да добавя нови продукти или да може да трие неща от поръчката си.

Създаването на електронен магазин е сложен процес, при който е необходимо да се вземат в предвид много фактори. Съществуващата марка, имиджовата политика на фирмата, подходящата реклама, спецификата на целевата аудитория, анализът на дизайна на конкурентните сайтове – всички тези фактори и способите за тяхната промяна, влияят на крайния резултат.

## Оптимизация на сайт

След като магазинът е създаден, има регистриран домейн и хост е необходимо потребителите бързо да го намират. За да се случи това, той трябва да бъде регистриран в търсачките и трябва непрекъснато да се оптимизира. Но какво представлява оптимизацията или както е по-известна SEO?

S.E.O. е абревиатура, която идва от английския език - Search Engine Optimization и в българския език се превежда оптимизация за търсачки. Това е процесът на подобряване видимостта на уеб сайт в търсачките чрез „натурални” или неплатени (нар. още “органични” или “алгоритмични”) резултати, който включва оптимизация на кода, структурата и текстовете на уеб сайт. Това са всички дейности, в резултат на които при търсене по ключова дума от страна на потребителите в търсачките, сайтът е на първите позиции в списъците.

SEO има за цел да повиши популярността на онлайн магазина и да го направи лесен за откриване и удобен за използване от бъдещите клиенти. Според различни изследвания 75% от интернет потребителите използват търсачка, за да намерят нов сайт или продукт. Създаването на магазина е само началото на един процес. За да бъде работещ, той трябва да продава, а за да продава, той трябва да бъде лесно намерен от клиентите.

Когато търсачката изчислява степента на съответствие на ключовата дума със съдържанието на сайта се отчитат параметри като:

* плътност на ключовите думи (фрази);
* Сложните алгоритми на съвременните търсачки позволяват провеждане на семантичен анализ на текста с цел филтриране на евентуален спам - когато ключовата дума/фраза се среща прекалено често в страницата над определен процент от целия текст.
* свързани думи;

Google все повече набляга на цялостното съдържание на текста и се опитва да индексира страниците семантично, т.е. по смисъл. Затова вече се набляга много на наличието на т.нар. свързани думи в текста. Т.е. ако текстът е за български сувенири, добре е да се включи информация за България, за български традиции, известни етнографски местности - все неща, от които посетители ще се интересуват:

* индекс на цитиране на сайта или количество и качество на ресурсите с хипервръзка към сайта;
* Много от търсачките не отчитат взаимните връзки между сайтовете, препоръчително е хипервръзките да бъдат от тематично сродни сайтове.
* данни за потребителското поведение.

За Google най-важният критерий за качеството на един сайт е доколко потребителите са доволни от сайта. Най-ясно разбираемите показатели са: време, прекарано на сайта, брой разгледани страници, повече от едно посещение на сайта, граматично и правописно правилен текст.

Работата по оптимизация на един сайт включва работа с вътрешни фактори - привеждане на текста на сайта в съответствие с избраните ключови думи (фрази), повишаване на качеството и количеството текстове на сайта, оптимизация на структурата, навигацията и вътрешните връзки, а също и с външни фактори - обмен на хипервръзки, регистрация в каталози и директории и други техники, целящи повишаване на броя и качеството на хипервръзките, насочени към конкретния ресурс.

Препоръчва се оптимизирането да се извършва в най-големите търсачки – Google, Yandex, Bing. Голяма част от останалите търсещи системи, използват информация от тях.

Според мястото на провеждане на оптимизацията на сайт SEO-то се дели на два вида: On Page и Off Page оптимизиране на сайт, т.е. в сайта и извън него.

Вътрешната оптимизация на сайт е обикновено оптимизиране на структурата, текстовете, URL адресите и кода на уеб сайта, така че всяка негова страница да се позиционира възможно най-високо в търсачките по естествен начин. Основните неща, които се проверяват и променят при необходимост са:

- Заглавия на страниците

- Meta описания

- Коко често се срещат ключовите думи/изрази

- Структурата на менютата

- Наличието на препратки между страниците

Сайтът да бъде разпознат от търсачката като релевантен на дадени ключови думи е един от най-важните фактори и затова значението им е голямо. Проучването за най-използваните ключови думи от потенциалните потребители е една от основните тактики в SEO стратегията. За електронните магазини, това е най-лесният начин потенциалните потребители да разберат за предлаганите продукти.

Когато се прави търсене, Google определя кои са най-полезните резултати и реда, в който уеб сайтовете да се покажат, въз основа на най-директните връзки с това, което се търси. Това се установява с алгоритми, които определят най-добрите съвпадения. Това е основната причина, поради която ключовите думи трябва да се избират внимателно, за да може търсачките да разберат, че ключовите думи, които търсят, отговарят на това, което предлагате на уеб сайта си. И тогава уеб сайтът ви ще е сред първите резултати на първа страница в Google.

Определянето на правилните ключови думи е от особено значение за електронния магазин, защото те влияят върху структурата и архитектурата на сайта. Те определят страниците, секциите, категориите и имената на статиите. Когато структурата на сайта отговаря на нуждите на потребителите той ще се класира на по-предни позиции в резултатите на търсачките.

Търсенето на ключови думи е дори още по-важно за онлайн магазините, защото определя структурата и архитектурата до голяма степен, тъй като сайтовете за електронна търговия има много повече страници, отколкото обикновените сайтове. Ако не няма добро проучване за ключови думи, структурата на сайта за електронна търговия няма да отговори на нуждите на потребителите и няма да постигне добро естествено позициониране в търсачките.

Тъй като Google е най-популярната търсачка, много притежатели на сайтове се стремят да подобрят класирането на сайта си, така че да се показва по-напред в намерените резултати при търсене на определена ключова дума.

Търсещият алгоритъм на Google използва патентована система, наречена PageRank, която се основава на определяне на популярността и важността на една страница, въз основа на броя линкове (back links), сочещи към нея от останалите уеб страници. Алгоритъмът изчислява рейтинга на линка от всяка уеб страница, въз основа на претеглената сума PageRanks от страниците, свързани към тях. Точният процент от общия брой уеб страници, които Google индексира, не е известен, тъй като е много трудно да се изчисли на практика. По-голяма част от другите търсачки до този момент са използвали като критерий за подреждане на намерените резултати честотата на срещане на дадения линк. В допълнение към PageRank, Google използва други тайни критерии за определяне на класирането на страниците, които са над 200.

Offpage SEO оптимизацията включва действията, които се предприемат извън рамките на уеб сайта. Те най-често се състоят в създаване на връзки към уеб сайта (link building) и добиване на връзки (link earning), което е важен фактор за успешно класиране в търсачките.

Други методи за извършване на Offpage SEO оптимизация са:

• регистрация в директории;

• размяна на линкове между един или повече сайтове;

• публикуване на статии за този сайт в други сайтове;

• създаване на нови – сателитни сайтове, които съдържат връзки към основния; създаване на блогове и форуми, чиято главна цел е да популяризират основния сайт.

За да се постигнат добри позиции в резултатите на търсачките, трябва постоянно да се оптимизира работата на онлайн магазина. За да се случи това, е важно да администраторът да има необходимите инструменти за контрол, за оценяване и измерване. Едни от най-добрите SEO инструменти, които трябва да се познават и да се използват за ефективна оптимизация са следните: Google Search Console, SemRush, Ahrefs, Screaming Frog, Majestic. Има и други инструменти които биха работили добре като алтернативи на тези.

Добрата SEO стратегия има за цел голям брои преобразувания, а не голям брои посещения. Преобразуването е активното действие на посетителя. То зависи от целта на сайта и може да бъде успешно пазаруване от онлайн магазин или изпращане на имейл адрес в нечия база данни. За целта се използва следната формула:

Б = Т х П х Л

Където:

Б - нашия бизнес;

Т – трафикът (броят посетители за определен период от време);

П – преобразувания (колко човека са си купили нещо или са направили това, което целим);

Л- лоялност на посетителите (тези, които се повтарят и посещават сайта отново).

Моделът изразен в това уравнение в практиката често се нарича оптимизация на лендинг страница.

Друг важен фактор, който оказва влияние върху стратегията за оптимизация е анализирането на конкуренцията. Проучването на конкуренцията е лесно чрез използването на различни инструменти, от които може да се види кое съдържание е най-популярно и кои ключови думи генерират най-много трафик, както и кои са хората, които са споделили дадено съдържание. Това ще даде полезен ориентир за интереса на потребителите и насочване на посланията в правилната посока.

За да бъде успешен един сайт, той трябва да има стратегия за развитие и оптимизация. От своя страна една SEO стратегия трябва да предвижда: как ще се посрещат посетителите на сайта с оглед, какво целим да направят те. С други думи стратегията за развитие на даден сайт чертае пътя за успешно изпълнение на определени бизнес цели.

Начините на оптимизация на сайтовете за търсещи машини се никога не са се променяли толкова динамично както сега. Подходът към SEO оптимизацията е основополагащ за възприемането на даден бизнес от потенциалните клиенти. Целта на SEO оптимизацията е по-скоро да изгради бранда, отколкото опит за манипулация на търсачките. Не трябва на всяка цена да се търси класиране на първите места в резултатите за търсене, а по-важно е клиенти да са доволни от това, което намират в уеб сайта.

Оптимизацията на даден сайт е процес на постоянно наблюдение, анализ и подобрение въз основа на получените данни. Най-ценните потребители са повторните клиенти и трафика от машината за търсене. Това налага постоянно анализиране на потребителите и адаптиране на сайта към техните потребности.

# ГЛАВА 2: ИНСТРУМЕНТИ И ТЕХНОЛОГИИ

## SQL

SQL е абревиатура, която означава “Structured Query Language” и е програмен език, който е най-често използвания при достъп до бази данни. В зависимост от програмната среда има възможност да се влезе в SQL директно (например за генериране на отчети), да се вграждат SQL откъси в код изписан на друг език или да се използват специфични програмно-приложни интерфейси за да скриете синтаксиса на SQL.

SQL Server Management Studio (SSMS) е интегрирана среда за управление на всяка SQL инфраструктура. Може да се използва SSMS за достъп, конфигуриране, управление, администриране и разработване на всички компоненти на SQL Server, Azure SQL база данни и SQL Data Warehouse. SSMS предоставя единична цялостна помощна програма, която комбинира широка група графични инструменти с редица богати редактори на скриптове, за да осигури достъп до SQL Server за разработчици и администратори на бази данни от всички нива на умения.

Базата данни в SQL Server е съставена от колекция от таблици, които съхраняват конкретен набор от структурирани данни. Таблицата съдържа т редове, също наричани записи или кортежи, и колони, също наричани атрибути. Всяка колона в таблицата е предназначена да съхранява информация от определен тип - например дати, имена, числа.

Компютърът може да има инсталиран един или повече от един брой от SQL Server. Всеки екземпляр на SQL Server може да съдържа една или много бази данни. В рамките на база данни има една или много групи за собственост на обекти, наречени схеми. Във всяка схема има обекти на база данни като таблици, изгледи и съхранени процедури. Някои обекти като сертификати и асиметрични ключове се съдържат в базата данни, но не се съдържат в схема. Базите данни на SQL Server се съхраняват във файловата система във файлове. Файловете могат да бъдат групирани във файлови групи.

Когато хората получат достъп до екземпляр от SQL Server, те се идентифицират като вход, а когато получат достъп до база данни, те се идентифицират като потребител на база данни. Потребителят на база данни може да се основава на вход. Ако съдържащите се бази данни са активирани, може да се създаде потребител на база данни, който не се основава на вход. Потребител, който има достъп до база данни, може да получи разрешение за достъп до обектите в базата данни.

SQL Server Management Studio позволява едновременен достъп на повече потребители. Инструментът Management Studio има графичен потребителски интерфейс за създаване на бази данни и обекти в базите данни. Управляващото студио също има редактор на заявки за взаимодействие с бази данни чрез писане на Transact-SQL изявления. Студиото за управление може да бъде инсталирано от инсталационния диск на SQL Server или да бъде изтеглено от MSDN.

## Cascading Style Sheets

CSS е съкращение от Cascading Style Sheets. Това е език за описание на стилове и се използва за оформление и придаване на естетически изглед на уеб страниците. Чрез него може да се определят размери, цветове, шрифтове, разстояния и други. CSS кодът се състои от последователност от стилови правила, всяко едно от които представлява селектор, последван от свойства и стойности.

CSS е създаден с цел да бъдат разделени съдържанието и структурата на уеб страниците отделно от тяхното визуално представяне. Преди стандартите за CSS, установени от W3C през 1995 г., съдържанието на сайтовете и стила на техния дизайн са писани в една и съща HTML страницата. В резултат на това HTML кода се превръща в сложен и нечетлив, а всяка промяна в проекта на даден сайт изисквала корекцията да бъде нанасяна в целия сайт страница по страница. Използвайки CSS, настройките за форматиране могат да бъдат поставени в един единствен файл, и тогава промяната ще бъде отразена едновременно на всички страници, които използват този CSS файл.

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

body {background-color:#000000;

font-family:arial,helvetica,sans-serif;

font-size:16px;

color:#cccccc;}

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, например за свойството color е зададена стойността #cccccc.

Когато се задава числова стойност на дадено свойство, например font-size: 16px, може да се остави стъпка разстояние между двоеточието и цифрите (: 16), но не бива да се отделят цифрите от мерната единица, т.е. правилно е да се напише “16px”, а не “16 px”. Във втория случай кодът няма да бъде разчетен от всички браузъри. 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”).

Предимства:

* CSS спестяват много работа - дизайнът на всички интернет страници на един сайт може да се зададе само с един CSS файл. CSS кодът се поставя във файл с разширение .css. След това този файл може да се използва във всяка една страница от уеб сайта.
* CSS дават голяма гъвкавост при дизайна на интернет страниците- при нужда от смяна на дизайна е достатъчно да се промени само този единствен CSS файл.
* CSS команди могат да се задават както в head частта на HTML документа, така и като отделни файлове с разширение .css, напр. style.css.
* Възможност различни хора да работят по съдържанието и стила на страниците - CSS указанията могат да се записват като отделни от HTML страниците файлове. Това позволява работата над един интернет сайт да се раздели между хора, които пишат съдържанието на страниците, и тези, които се грижат за оформлението.
* Употреба на един и същи CSS файл в различни проекти - веднъж създаден, CSS файла може да се използва за указване стила на произволен брой страници - достатъчно е във всяка от тях CSS файла да бъде посочен за използване.
* По-бързо зареждане на страниците - ма случаи, в които някои от браузърите интерпретират CSS командите по начин, отличаващ се от приетия стандарт. Но като цяло CSS е много добре поддържан от всички широко употребявани браузъри.
* Разделянето на съдържание и стил дава много по-голяма гъвкавост в управлението на страниците.

Има три начина за добавяне на CSS стил в HTML документ:

* Външен стил - (External Style Sheet) и се използва когато трябва да се контролират множество HTML документи, като нужните параметри се задават във външен файл (file.css).
* Вътрешен 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 - Inline Styles - вътрешни за HTML таговете стилове. CSS стилът се разполага като атрибут директно в HTML тага например <p style="color:#cccccc;">.

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.

В CSS файла могат да бъдат декларирани практически всички HTML тагове, като селектори, които се използват в HTML страниците, като им бъдат зададени атрибути със съответните стойности. По този начин може лесно да се контролират неограничен брой HTML страници, като промяната на стила им се извършва само с редакция на един файл - CSS файла. Това спестява много труд и време, затова е силно препоръчително, особено при изработка на големи сайтове с множество страници, винаги да се използват външни стилове.

## бази данни.

### Същност на базата данни

База данни (БД) e съвкупност от данни, структурирани по начин, който позволява лесното и бързото им извличане, преглеждане, търсене и свежда до минимум дублирането на информация. Характерно за базите данни е, че данните са независими от софтуера.

## **Python**

Python е високо нивов програмен език, който се използва за различни цели като уеб разработка, научни изчисления, изкуствен интелект, сървърни приложения и много други. Той е известен с ясната и четима синтаксис стилова на кода, която го прави подходящ за начинаещи и напреднали програмисти.

Ето някои основни характеристики на Python:

**Прост за използване:** Python е лесен за научаване и четен. Той предлага кратък синтаксис, който е подходящ за разработка на почти всеки вид софтуер.

**Интерпретируем език:** Python е интерпретируем език, което означава, че кодът се изпълнява ред по ред от интерпретатора, без да се изисква компилация предварително.

**Множество библиотеки и фреймуърки:** Python разполага с голям екосистем от библиотеки и фреймуърки, които позволяват разработването на разнообразни приложения, като например Django и Flask за уеб разработка, TensorFlow и PyTorch за машинно обучение и др.

**Крос-платформен:** Python работи на множество операционни системи, включително Windows, macOS и различни дистрибуции на Linux.

**Съвместимост и интеграция:** Python може да бъде лесно интегриран с други езици като C/C++ и може да използва библиотеки написани на тези езици.

**Широк спектър приложения:** Python се използва в различни области като уеб разработка, наука и изследвания, игри, автоматизация, системно администриране и други.

**Общност и поддръжка:** Python разчита на голяма общност от програмисти по целия свят, които предоставят поддръжка, документация и развитие на езика.

**Четим и елегантен синтаксис:** Python е известен със своя четим синтаксис, който прави кода лесен за разбиране и поддържане. Той се фокусира върху четливостта и яснотата на кода, което го прави отличен избор за начинаещи програмисти.

**Динамично типизиране:** Python е език с динамично типизиране, което означава, че не е необходимо явно да декларирате типовете на променливите преди да ги използвате.

**Интерпретируемост:** Python кодът се изпълнява от интерпретатор на Python, който може да чете и изпълнява кода ред по ред. Това го прави много гъвкав за бърза разработка и експериментиране.

**Обширна стандартна библиотека:** Python разполага с обширна стандартна библиотека, която включва модули за различни задачи като работа със стрингове, файлове, мрежи, JSON, математически операции и други.

**Поддръжка на обектно-ориентирано програмиране (ООП):** Python поддържа обектно-ориентирано програмиране, което позволява дефинирането на класове, обекти и наследяване.

**Множество библиотеки за разширяване на функционалността:** Освен стандартната библиотека, Python разполага с огромен брой сторонни библиотеки (packages), които предоставят разнообразни функционалности като машинно обучение, обработка на изображения, работа с бази данни и др.

**Използван в различни индустрии и области:** Python се използва широко в индустриите на софтуерното разработване, научните изследвания, анализа на данни, уеб разработката, игрите и много други.

## HTML

HTML (HyperText Markup Language) е основният език за създаване на уеб страници. Той дефинира структурата и съдържанието на уеб страниците чрез използване на различни HTML елементи и тагове.

Въз основа на синтаксиса на XML (eXtensible Markup Language), HTML използва тагове, които обозначават различни елементи на уеб страницата. Един HTML документ се състои от набор от тагове, които са вложени един в друг и определят структурата на страницата.

Основните компоненти на HTML включват:

**Елементи и тагове**: Всяка част от уеб страницата се представя чрез HTML елемент, който се огражда с отварящ и затварящ таг. Например, **<p>** е таг за абзац, **<h1>** - за заглавие на ниво 1, **<a>** - за хипервръзка и така нататък.

**Атрибути**: Таговете могат да имат атрибути, които допълнително характеризират елемента. Например, тагът **<a>** може да има атрибут **href**, който определя адреса на препратка.

**Структура**: HTML документите се организират по йерархичен начин, като всяка страница започва с елемент **<html>**, следван от **<head>** и **<body>**. **<head>** съдържа мета информация и връзки към външни ресурси, докато **<body>** съдържа фактическото съдържание на страницата.

**Списъци**: HTML предлага различни видове списъци като не номерирани (**<ul>**), номерирани (**<ol>**) и описание (**<dl>**).

**Таблици**: За представяне на таблична информация HTML предлага таговете **<table>**, **<tr>** (ред), **<td>** (клетка) и други.

HTML е език с отворен стандарт, което означава, че всеки може да създава уеб страници, използвайки го. Той е основата за разработването на уеб сайтове и уеб базирани приложения. С развитието на HTML5 и въведението на нови възможности като мултимедийни елементи, графики и улучшена семантика, HTML остава един от най-важните инструменти в кутията на всеки уеб разработчик.

Разбирането на HTML е от съществено значение за всеки, който желае да се занимава с уеб разработка. Независимо дали сте начинаещ или опитен уеб разработчик, владеенето на HTML ви дава основната основа за създаване на красиви и функционални уеб страници.

Ето още някои важни аспекти на HTML:

**Форми**: HTML предоставя елементи за създаване на форми, които са от съществено значение за взаимодействие с потребителите. Таговете като **<form>**, **<input>**, **<select>** и **<button>** позволяват създаването на различни видове форми, като обратна връзка, регистрационни форми и др.

**Мултимедия**: HTML предлага вградени елементи за вграждане на мултимедийно съдържание като изображения (**<img>**), аудио (**<audio>**) и видео (**<video>**). Тези елементи правят уеб страниците по-богати и интерактивни.

**Семантични елементи**: С HTML5 бяха въведени много нови семантични елементи, които помагат за по-добро структуриране на уеб страниците и подобряване на тяхната достъпност за търсачки и потребители с увреждания. Такива елементи са **<header>**, **<nav>**, **<section>**, **<article>**, **<footer>** и други.

**Мета информация**: HTML съдържа елементи за включване на мета информация за уеб страниците, като например заглавие (**<title>**), мета описание (**<meta>**), ключови думи (**<meta>**), стилове (**<style>**) и скриптове (**<script>**).

**Валидност и достъпност**: Валидният HTML код е от съществено значение за правилното функциониране на уеб страниците и тяхната достъпност за различни потребители и устройства. Инструменти като W3C валидаторът помагат за проверка на валидността на HTML кода и за осигуряване на неговото правилно функциониране.

С обучението и практиката в HTML можете да развиете мощни умения за създаване на уеб страници и приложения. Независимо от вашите цели и амбиции в уеб разработката, владеенето на HTML е крайъгълният камък, който ви отваря вратата към широк свят от възможности в дигиталното пространство.

## JavaScript

JavaScript (JS) е високо нивен програмен език, който се използва за създаване на динамични и интерактивни уеб приложения. Той е един от трите основни елемента на уеб разработката, заедно с HTML и CSS, и играе ключова роля в подобряването на потребителския опит на уеб страниците.

В основата си, JavaScript позволява на разработчиците да добавят различни функционалности към уеб страниците, като например валидация на форми, анимации, динамично генериране на съдържание, комуникация със сървъра и много други. Ето някои от ключовите характеристики на JavaScript:

**Интерактивност и събития**: JavaScript позволява на потребителите да взаимодействат с уеб страниците по различни начини, като например щракване, хостване, въвеждане на данни и други. Той обработва тези събития и изпълнява съответните действия, което позволява създаването на интерактивни приложения.

**Манипулация на DOM**: JavaScript има достъп до Document Object Model (DOM), който представлява структурата на уеб страницата. Това позволява на разработчиците да манипулират и променят елементите на страницата, като например добавят нови елементи, променят стиловете им, премахват елементи и други.

**Асинхронност и AJAX**: JavaScript позволява изпълнението на асинхронни операции, като например заявки към сървъра чрез AJAX (Asynchronous JavaScript and XML). Това позволява на уеб приложенията да обменят данни със сървъра без да се налага презареждане на цялата страница, което подобрява ускорението и ефективността на приложенията.

**Модулност и реизползваемост**: JavaScript поддържа модулно програмиране, което позволява на разработчиците да разделят кода на по-малки и преизползваеми модули. Това улеснява поддръжката на кода и неговото разширяване в бъдеще.

**Съвместимост и разнообразие**: JavaScript е широко поддържан от всички съвременни браузъри и операционни системи, което го прави универсален инструмент за уеб разработка. Освен това, с развитието на Node.js, JavaScript се използва и във всякакви други приложения, включително сървърни приложения и мобилни приложения.

JavaScript е неоспоримо един от най-важните елементи в уеб разработката днес. Независимо дали сте начинаещ или опитен разработчик, владеенето на JavaScript ви позволява да създавате мощни и интерактивни уеб приложения, които подобряват потребителския опит и функционалността на вашите проекти. Разбирането на JavaScript е от съществено значение за всеки, който желае да се занимава с уеб разработка. Независимо дали сте начинаещ или опитен програмист, владеенето на JavaScript ви дава възможност да създадете по-динамични, интерактивни и потребителски приятни уеб приложения. Ето още някои важни аспекти на JavaScript:

**Библиотеки и фреймуърци**: JavaScript разполага с широка гама от библиотеки и фреймуърци, които улесняват уеб разработката. Например, библиотеки като jQuery предоставят удобни методи за манипулиране на DOM и обработка на събития, докато фреймуърци като React, Angular и Vue.js предоставят по-структурирани и мощни инструменти за създаване на по-сложни уеб приложения.

**Възможности за визуализация и графика**: JavaScript предоставя различни библиотеки и инструменти за визуализация и графика, които могат да бъдат използвани за създаване на интерактивни графики, диаграми, графики и други визуални ефекти. Някои от популярните библиотеки включват D3.js, Chart.js и Three.js.

**Сигурност**: JavaScript е свързан със значителни предизвикателства за сигурност, като например крос-сайт скриптинг (XSS) и CSRF атаки. Разработчиците трябва да бъдат внимателни при създаването на уеб приложения и да следват най-добрите практики за сигурността, като например валидация на входните данни и използване на безопасни методи за комуникация със сървъра.

**Екосистема и общност**: JavaScript разполага с огромна и активна общност от програмисти, което включва форуми, блогове, видео уроци, семинари и други ресурси за обучение и споделяне на знания. Тази общност е източник на ценна информация и подкрепа за всеки, който се занимава с уеб разработка.

## Jinja

Jinja е шаблонен движок за Python, който позволява на разработчиците да генерират динамично HTML, XML, JSON и други текстови базирани формати. Той е често използван в уеб разработката, особено в комбинация с популярни уеб рамки като Flask и Django.

Ето някои от ключовите характеристики на Jinja:

**Лесна за научаване синтаксис**: Jinja използва синтаксис, който е подобен на Python и лесен за разбиране за програмистите. Той включва изрази, оператори и конструкции за управление на потока на данни, които позволяват генерирането на динамично съдържание.

**Шаблониране и наследяване**: Jinja поддържа шаблониране и наследяване, което позволява на разработчиците да създадат многократно използваеми шаблони за уеб страници. Това улеснява поддръжката и разширяемостта на уеб приложенията.

**Променливи и филтри**: Jinja позволява използването на променливи и филтри, които позволяват манипулирането и форматирането на данни в шаблоните. Това включва филтриране на списъци, форматиране на дати и числа, извличане на данни от обекти и много други.

**Контрол на потока**: Jinja предоставя мощни конструкции за контрол на потока, като условни оператори, цикли и блокове. Това позволява на разработчиците да създадат динамично съдържание, което се променя в зависимост от различни условия и данни.

**Безопасност**: Jinja предлага вградена поддръжка за безопасност, като автоматично екранира потенциално опасни символи и заявки, което предотвратява XSS (Cross-Site Scripting) атаки и други сигурностни проблеми.

**Разширяемост и добавки**: Jinja е разширяем и позволява на разработчиците да създават свои собствени добавки и разширения, които да разширят функционалността на шаблоните. Тези добавки могат да бъдат специализирани за конкретни нужди или да предоставят допълнителни функции за обработка на данни.

**Интеграция с Python**: Jinja е напълно интегриран с Python и позволява на разработчиците да използват Python код в техните шаблони. Това позволява на разработчиците да създават сложни логически операции и функции директно в шаблоните, което подобрява гъвкавостта и мощта на шаблонния език.

**Документация и общност**: Jinja разполага с богата документация и активна общност от потребители и разработчици, които споделят знания, опит и решения за проблеми. Това прави процеса на учене и развитие на уменията по използване на Jinja по-лесен и приятен.

**Поддръжка на многоезичност**: Jinja предлага вградена поддръжка за многоезичност, като позволява на разработчиците да създадат шаблони, които автоматично се адаптират към различни езици и култури. Това прави Jinja подходящ избор за разработка на уеб приложения, които се предлагат на различни аудитории по света.

**Интеграция с уеб рамки**: Jinja се използва широко в съчетание с популярни Python уеб рамки като Flask, Django и Pyramid. Той е интегриран по подразбиране в тези рамки и предоставя мощен механизъм за генериране на HTML и други форматиране на данни в уеб приложенията.

Jinja остава предпочитан инструмент за генериране на динамично съдържание в уеб разработката. Неговата гъвкавост, лесен за научаване синтаксис и мощни възможности го правят идеален избор за разработчиците, които искат да създадат мащабируеми и иновативни уеб приложения.

## Django

Django е мощен уеб фреймуърк, който се използва за бърза и ефективна разработка на уеб приложения. Със своята интуитивна синтакса и множество вградени функционалности, Django прави процеса на създаване на уеб приложения бърз и лесен. В този текст ще разгледаме някои от основните компоненти и концепции на Django, както и начините, по които можем да изградим уеб приложение с него.

Основни компоненти на Django:

**Модели -** Django използва модели за дефиниране на структурата на данните в уеб приложението. Моделите представляват таблиците в базата данни и съдържат полета и методи за манипулиране на данните.

**Изгледи (Views**) - Изгледите се използват за обработка на заявките от потребителите и генериране на подходящ отговор. Те взаимодействат с моделите и шаблоните, за да генерират HTML кода, който се изпраща към потребителя.

**URL маршрутиране** - URL маршрутирането се използва за съпоставяне на URL адресите на уеб приложението с конкретни изгледи. Това позволява на Django да определи кой изглед да използва при дадена заявка.

**Шаблони** - Шаблоните се използват за генериране на HTML кода, който се изпраща към потребителя. Те позволяват на разработчиците да отделят представянето от логиката и данните на приложението.

Примерен процес на разработка с Django:

**Дефиниране на модели** - Започваме с дефиниране на моделите за нашето приложение, които ще определят структурата на данните.

**Създаване на изгледи** - След като сме дефинирали моделите, създаваме изгледите, които ще обработват заявките от потребителите и ще генерират подходящ отговор.

**Конфигуриране на URL маршрути** - Дефинираме URL маршрутите на нашето приложение, които ще съпоставят URL адресите на конкретни изгледи.

**Създаване на шаблони** - Накрая, създаваме шаблоните, които ще бъдат използвани за генериране на HTML кода, който се изпраща към потребителя.

**Административен панел -** Django предоставя вграден административен панел, който позволява лесно управление на данните в приложението. С помощта на административния панел администраторите могат да създават, редактират и изтриват записи в базата данни без директно използване на SQL заявки.

**Форми -** Django предоставя удобен начин за работа с HTML форми. Формите в Django могат да бъдат създавани чрез класове, които автоматично генерират HTML код за формата и валидират входните данни, когато те се изпращат обратно към сървъра.

**Сесии и потребителски аутентикация -** Django има вградена поддръжка за управление на сесии и потребителска аутентикация. Това позволява на разработчиците лесно да създадат системи за регистрация на потребители, влизане в профили и управление на достъпа до различни части на уеб приложението.

**Поддръжка на различни бази данни -** Django поддържа множество различни типове бази данни, включително PostgreSQL, MySQL, SQLite и др. Това дава възможност за лесно преносими приложения, които могат да бъдат изпълнявани в различни околни среди.

Django е не просто един фреймуърк за уеб разработка, а цялостно решение, което предоставя всичко необходимо за създаване на сигурни, мащабируеми и лесно поддържаеми уеб приложения. Независимо дали сте начинаещ разработчик или опитен професионалист, Django е отличен избор за вашия следващ проект.

## BOOTSTRAP

Bootstrap е мощен и популярен фронтенд фреймуърк за уеб разработка, който е с отворен код и е разработен от екипа на Twitter. Той предоставя набор от инструменти, компоненти и стилове, които улесняват създаването на модерни и отзивчиви уеб приложения.

Ето някои ключови характеристики на Bootstrap:

**Отзивчив дизайн (Responsive Design)**: Bootstrap е известен със своя отзивчив дизайн, който позволява на уеб сайтовете да се приспособяват автоматично към различни размери на екрана (мобилни устройства, таблети, десктоп компютри). Това подобрява потребителския опит и помага за оптимизирането на уеб приложенията за различни устройства.

**Готови компоненти**: Bootstrap предоставя голям набор от готови компоненти като бутони, форми, навигационни панели, карусели, модални прозорци, таблиц и други. Тези компоненти са лесни за използване и могат да бъдат интегрирани в уеб страниците с минимални усилия.

**Типография и стилове**: Фреймуъркът предоставя подробни CSS стилове и типография за основните елементи на уеб страниците. Bootstrap предлага добре дефинирани класове за създаване на колони, сетки и разпределение на елементите на страницата.

**JavaScript компоненти**: Включени са готови JavaScript компоненти като модални прозорци, взаимодействие със събития, валидация на форми и други. Тези компоненти добавят интерактивност към уеб приложенията и са лесни за използване.

**Съвместимост с браузъри**: Bootstrap е съвместим с всички основни уеб браузъри (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge и други), което осигурява съответствие и стабилност при разработка на уеб приложения.

**Лесна интеграция и документация**: Bootstrap се предоставя с подробна документация и примери, което улеснява ученето и използването на фреймуърка. Можете да интегрирате Bootstrap във вашите проекти чрез включване на CSS и JavaScript файлове или чрез CDN (Content Delivery Network).

**Система за сетка (Grid System)**: Една от най-известните и полезни характеристики на Bootstrap е неговата система за сетка. Тя ви позволява да разпределяте съдържанието на уеб страницата във флексибилни колони и редове, което улеснява организацията на дизайна и подреждането на елементите в зависимост от различните размери на екраните.

**Кастомизация и теми**: Bootstrap дава възможност за лесна кастомизация на стиловете и компонентите. Можете да създадете собствени теми или да променяте цветовете, шрифтовете и други стилове, за да придадете уникален външен вид на вашите уеб приложения.

**Поддръжка на плъгини и допълнителни компоненти**: Bootstrap има активна общност от разработчици, които създават различни плъгини и допълнителни компоненти, които може да бъдат лесно интегрирани във вашите проекти. Това включва разширения за форми, карти, таблиц и други функционалности.

**Съвместимост със CSS препроцесори**: Bootstrap може да бъде използван с популярни CSS препроцесори като SASS или Less, което улеснява управлението на стиловете и подобрява ефективността на разработката.

**Бърз старт и респонсивно поведение**: Bootstrap предоставя набор от готови шаблони и примери, които могат да ви помогнат да започнете бързо със създаването на уеб сайтове. Фреймуъркът също така се грижи за ефективното зареждане и бързата реакция на уеб приложенията.

**Поддръжка на мобилни устройства**: Bootstrap се фокусира върху оптимизирането за мобилни устройства, като предоставя гъвкави компоненти и стилове, които се адаптират към различните екрани и устройства.

## FORMS

Формите (Forms) в уеб разработката са основен инструмент за събиране на информация от потребителите чрез интернет. Те представляват визуални интерфейси, които позволяват на потребителите да въвеждат данни и да ги изпращат към уебсайтове или уеб приложения. Формите са от съществено значение за интерактивността на уеб страниците, понеже позволяват на потребителите да взаимодействат със системата, да предоставят информация и да изпълняват различни действия, като например изпращане на запитвания, регистрация, изпращане на коментари, създаване на профил и други.

Ето някои ключови аспекти на уеб формите:

**Елементи на форма**: Формите се състоят от различни елементи за въвеждане на данни, като текстови полета, полета за пароли, полета за избор на опции (радио бутони, отметки, падащи менюта), полета за качване на файлове, бутони за изпращане и други. Тези елементи позволяват на потребителите да въвеждат и избират различни видове данни.

**Валидация на данни**: Формите често включват валидация на данни, която се използва за уверяване, че потребителите са въвели коректни и пълни данни. Например, може да има изискване за попълване на определени полета, проверка за правилно форматиране на данни като имейли или телефонни номера и други видове валидации.

**Събиране на данни**: Информацията, която потребителите въвеждат в формите, се събира и изпраща към уебсайта или приложението за обработка. Тази информация може да се използва за различни цели, като например за регистрация на потребители, обработка на поръчки, анализиране на данни и други.

**Управление на състоянието и сигурността**: Формите могат да поддържат състоянието на потребителите чрез използване на сесии или бисквитки (cookies), което позволява запазването на информацията между различни заявки. Освен това, важно е да се поддържа сигурността на данните, предавани през формите, за да се предотвратят атаки като CSRF (Cross-Site Request Forgery) или XSS (Cross-Site Scripting).

**Дизайн и UX (User Experience)**: Добре проектираните форми са важен аспект от дизайна на уеб приложенията, като се стремят да предоставят удобен и интуитивен интерфейс за потребителите. Подходящият дизайн на формите може да подобри потребителското изживяване и да увеличи ефективността на интеракцията с уебсайта.

Типове форми:

**Форми за вход и регистрация**: Тези форми се използват за събиране на информация от потребителите, като потребителски имена, пароли, имейли и други данни, необходими за вход или регистрация в уеб приложение.

**Форми за контакт**: Често срещани форми, които позволяват на потребителите да изпратят съобщения или запитвания към администратора на уебсайта.

**Форми за поръчки и плащания**: Използват се в онлайн магазини за събиране на информация за доставка, плащане и потвърждение на поръчката.

**Форми за коментари и обратна връзка**: Позволяват на потребителите да оставят коментари, оценки или обратна връзка за съдържанието на уебсайта.

Валидация на данни:

**Клиентска валидация**: Извършва се на страната на клиента (в браузъра) с помощта на JavaScript, за да се увери, че въведените данни са в правилния формат преди да бъдат изпратени към сървъра.

**Сървърна валидация**: Извършва се на страната на сървъра, за да се потвърди валидността на данните, след като са пратени от клиента. Това включва проверка за цялостност и сигурност на данните.

**Сигурност:** Важно е да се поддържа сигурността на данните, предавани през уеб формите, за да се предотвратят атаки като инжекции на SQL, CSRF (Cross-Site Request Forgery), XSS (Cross-Site Scripting) и други видове атаки.

Препоръчително е да се използват капчи (CAPTCHA) или други методи за потвърждение на човешката идентичност, за да се предотврати автоматизирано нежелано поведение.

**Управление на състоянието:** Формите могат да поддържат състояние на потребителите чрез сесии или бисквитки (cookies), което позволява запазването на информацията между различни заявки или посещения на уебсайта.

Важно е да се управлява състоянието на формите така, че да се осигури добър потребителски опит и сигурност.

**Достъпност:** Формите трябва да бъдат проектирани така, че да бъдат достъпни за всички потребители, включително хора с увреждания или използващи асистивни технологии. Това включва правилно маркиране на елементите, добър контраст на цветовете, ясни инструкции и други.

**Дизайн и UX (User Experience):** Важно е формите да имат приятен дизайн и да осигуряват добър потребителски опит. Това включва правилно подреждане на елементите, удобен интерфейс за въвеждане на данни и ясни съобщения за грешки или успех.

**Безопасност на формите:** Винаги трябва да се обърне внимание на сигурността на уеб формите, за да се предотвратят атаки като инжекции на SQL, XSS и други видове атаки.

Избягвайте съхраняването на чувствителни данни във формите, като пароли или кредитни карти, и използвайте HTTPS за защита на данните при трансфер.

**Достъпност на формите:** Уверете се, че формите са достъпни за всички потребители, като предоставите ясни етикети, подходящи подсказки и правилно маркирани полета.

Използвайте ARIA атрибути (Accessible Rich Internet Applications) за подобряване на достъпността за хора с увреждания.

**Тестове и оптимизация:** Провеждайте тестове на формите, за да се уверите, че те работят правилно и че потребителите могат лесно да ги използват.

# ГЛАВА 3: РЕАЛИЗАЦИЯ

## Етапи на проекта

Първо ще започна от анализиране на заданието. Това означава да го прочета внимателно, виждам първо с какво изначално разполагам и се опитвам да разбера всъщност какво трябва да постигна, тоест определям си целите и задачите от една страна и сходните данни от друга страна. След това улучвам възможните решения, които ги има в света, такива могат да бъдат аналогичните решения. Това са тези решения, които се поучаваш от чуждия опит, тоест да видя хората по принцип как се справят с това. След това мисля за технологиите, които реализират това, например масово хората пишат на PHP, използват фреймуъркове, използват динамични сайтове, пишат на Python и използват Django, правят динамичен интерфейс и други. Всичко това чрез търсене, мислене и анализиране се нарича теоретично изследване на проблема. Следващата стъпка е избор на конкретна технология. Например спрял съм се на Python като език за програмиране, на Django като framework от страна на сървъра, MySQL съм избрал като база данни, избрал съм за комуникация между клиента и сървъра от страна на сървъра DRF Django REST framework, а от страна на клиента съм решил, че ще ползвам JavaScript VUE framework, вероятно за да добие визия приложението и за да има адаптивен дизайн ще използвам възможностите на някой мощен framework като bootstrap, просто правя избор на конкретна технология, всеки според неговия проект. Това, че огромната част от проектите са избрали една и съща технология е съвсем естествено, тъй като технологиите са в пъти по-малко от колкото са проектите в света, които се реализират, то това е красивото в технологиите, че по една и съща технология се реализират различни проекти. Следващото, което трябва да направя е да приложа моя избор, за да приложа моя избор, съобразявайки се с технологията първо трябва ясно да очертая функционалните елементи на приложението,което правя. Тръгвам да реализирам приложението от изчистване на функциите, които ще реализира. Тази стратегия се нарича MVP стратегия, идеята е да направя минималния вариант, който е работоспособен и след това да допълвам функционалности. Така е по-добре, отколкото изначално да натоваря приложението с функционалности, които или изобщо не са реализирани, или са чупливи, не са тествани, не работят, или просто стресират потребителя, който още не е свикнал с приложението, а вече е натоварен с функционалности, които са със съмнителна употреба. Затова тръгвам от най-важните неща, а именно правя минимално приложение представям си функциите, представям си и как ще изглеждат и си правя така наречения макет. Макетът е визия на отделните стъпки във функционирането на приложението. За всяка отделна стъпка си правя изглед, тоест да си представя как ще изглежда. Създаването на макет спестява страшно много труд по отношение на реализацията, защото се изчиства функционалността, тоест премахват се излишните функции. След като създада макета започвам фактическата реализация като приложен програмист. В началото бях софтуерен архитект, след това бях софтуерен инженер конструирайки приложението и сега ставам приложен програмист. Ще ми е нужен back-end. Чрез back-end инсталирам Django, създавам Django проект, след това създавам приложение в проекта. Създавам папка templates в нея прехвърлям HTML шаблоните, ако има такива, създавам папка static и в нея прехвърлям всичките статични елементи като например: JavaScript,CSS и всичко друго, което има статичен елемент върху шаблоните. Просто се водя от макета, който съм създал. Създавам, ако е необходимо папка media. Папката media само я създавам, но ако имам бази данни, в които има таблици, в полетата картинки, MySQL съхранява в базата данни само url адреса на картинката, а самата картинка Django я слага в под папка на папката media. Под папката на папката media се нарича items. Django е настроено така, че да записва в папка items всички картинки и клипчета, които са в динамичен ресурс, тоест записват се в базата данни по същество се записват в тази папка. Папките, които създавам са static, templates и евентуално media. В самите шаблони трябва да се настроят връзки, разширявам ако имам базов шаблон, задължително включвам директивата load static и включвам блокове със съдържание, където евентуално предавам контекста. Чрез load static имам link към някакъв статичен CSS файл. Идеята е, ако се получи upload на истински и реален сървър не на текстови, папката може да не е static, може да има други правила на този сървър. И от тук нататък започват разликите в проектите един имат едни файлове, други имат различни от тези, но структурата и процесите, през които преминавам и генерирам е еднаква за всички разработчици на такъв тип проекти.

## Концепция

За оторизация на потребителите решавам да използвам оторизационната система на Django, защото тя е надеждна и сигурна от една страна, а от друга страна, защото едно от големите предимства на Django, заради което е популярно е точно това, че предлага готови решения за често срещани проблеми, такъв какъвто е оторизирането и оклентификацията на потребителите. При това Django го предлага в два варианта освен, така да се каже по подразбиране, когато си регистрирам моделите в admin панела и влизам и ги манипулирам, наблюдавам, променям, добавям и изтривам, така Django предлага и възможността програмно да бъде достъпна тази система за оторизация и проверка на потребителите, нещо повече Django предлага интерактивно изгледи и форми готови, разработени и най-често избирам точно така да постъпя ще взаимствам функционалността на Django, логвам потребителя, чета му правата, проверявам, и ако е просто потребител, който няма права на служител или на администратор, мен това перфектно ме устройва, защото тези, които са обикновени потребители това ще са ми клиентите, а тези които са персонал ще са и мой персонал. Потребителят може да бъде просто активен, потребител със статут на персонал и потребител със статут на супер-потребител. Супер-потребителят ми е администраторът, със статут на персонал ми е административният персонал, а просто потребители, които са активни, но нямат други права са потребителите на системата. Django предлага изгледи и форми, които могат да се наследят и освен това има една много хубава екстра, която е удачно да я ползвам в Django това се нарича декоратор. Декораторите са предварително разработени функции, които обвиват даден клас, придавайки му допълнителна функционалност, тоест типът като клас си остава, но му се присвоява и допълнителна функционалност, така да се каже той се опакова в тази функционалност. Освен, че и аз мога да си правя декоратори, най-често използвам готови създадени във фреймуърка Django, а в моя случай интересен е декораторът login\_ required. Чрез него мога да декорирам изгледите си и така за да може някой да не прескача login страниците и да се логва директно, даже ако някой види адресната лента на браузъра и се опита да влезе по някакъв начин, заобикаляйки login страниците тогава Django ще го посрещне с необходимостта да се оторизира пред системата, тоест правя я надеждна от гледна точка на защита на данните в системата. За регистрирането наследявам формата за влизане на потребителя от идентификация на потребителите, наследявам обекта от потребители респективно в базата данни, която се създава при създаването на Django проект, наследявам готовите форми, изграждам връзките на мойте форми с базата данни и следвам модела описан в документацията. Това беше за формите, а в изгледите създавам изглед lоgin, който всъщност автоматично се опитва да логне програмно потребителя, задавам му success\_url, към който ще бъде насочен потребителя в случай на успешно влизане в системата и този изглед всъщност връща правилния url в случай, че потребителят е влязъл. Това прави Django толкова популярно, защото има такива удобни инструменти. Третият елемент от тази система е в шаблоните. В login формата имам форма, но аз не я конструирам, а директно наследявам във файла forms, тоест шаблонизаторът като срещне променливата form ще генерира автоматично HTML формата, която пасва на описанието, което съм създал във forms, в моя случай съм казал, че искам само username и password. След като влезе потребителят аз правя две фактически и независими приложения. Има такива ситуации, при които цялото приложение е изградено като интерактивно, еднодокументно или в случай, когато в рамките на някакъв сайт се вгражда някаква интерактивност на базата, на rest технологията, има и такива случай, при които се прави доста интерактивно приложение с използване само на форми, модели, трислойни приложения със средствата на Django, а в моя случай при оторизацията на потребителите ползвам вградените средства на Django, а след това правя две практически независими, едното за клиентите на фирмата, а другото за служителите на фирмата. И двете всеки по отделно са приложение с еднодокументен интерфейс базирани на rest технология, която е реализирана на базата на Django REST framework DRF от страна на сървъра и с изграждане на съответния програмен интерфейс, Vue.js като основен двигател на интерактивността от страна на клиента и ACSIUS като комуникация в посоката от клиент към сървър. Обменът на данни е стандартният за този тип технологии, тоест данните са реализирани. Това е удобно, защото потребителите, респективно служителите виждат някаква информация, мислят, наблюдават, но действията, които предприемат относително рядко се налага да се отразява нещо на сървър. Потребителят например влиза и регистрира някаква поръчка, но през една голяма част от потребителското време, което ползва системата той вижда някаква обща статистика като например последните заявки за обслужване и необслужване или последната, която е заявил е обработена, дали е завършена задачата, тоест той не предприема действия, а визуално наблюдава и евентуално превключва от един екран на друг, поради което има логика това да е приложение от страна на клиента, а сървърът да се намесва тогава, когато трябва да предложи друг набор данни. Същото касае и работата на персонала, защото персоналът работи извън системата по заявката, и когато стигне до някакъв резултат го отразява, тоест заявката му стои отворена и не е редно сървърът през това време да е ангажиран с някаква дейност, а количеството данни, които трябва да се върне от персонала, които обслужват дадената заявка към сървъра е минимална, той просто трябва да отбележи, че задачата е завършена, респективно или евентуално с някакво минимално съобщение. Стигам до двете приложения, които ще бъдат базирани, затова разработвам на практика три файла: единият е login с цялата и функционалност за оторизация на потребителите, за селектиране какъв тип и пренасочване евентуално или към client или към staff, това са ми базовите шаблони за двете приложения. Всяко приложение си върви със свой JavaScript, със своя функционалност, макар и доста подобни те са до голяма степен огледални като функционалност. Потребителят на системата може да генерира запитване или заявка за обслужване, но не може да реагира на нея, докато персоналът от страна на фирмата не може да генерира заявка за обслужване, но може пък да отразява реакция на тази заявка. Стигам до съответния интерфейс, в който има login страница, идеята за бизнес приложение е крайно изчистен, защото потребителят не трябва да е фокусиран върху визия, анимации и други, а трябва да е фокусиран върху предмета на дейност, в случая login страница от една страна и от друга страна дизайнът трябва да е приятен. Технически това означава, че HTML шаблонът който е подходящ за функционалните страници не е подходящ за login страниците, и обратното шаблонът, който е подходящ за login страниците не е подходящ за функционалните страници. Поради това се получава съчетаването на два различни шаблона в един проект. Един шаблон си има собствен JavaScript, проблемът е потенциално съвпадане на имена, на скриптове или на картинки, за да се избегне това на единия шаблон необходимите му статични ресурси са в папка assets, а другите са необособени в отделна папка, по този начин двата шаблона съжителстват без никакви проблеми, защото съм настроил различни пътища. Типичният шаблон за еднодокументен интерфейс е с няколко секции, разликата е, че има меню, което не изчезва, защото това не е сайт, а приложение, тоест потребителят си го отваря в браузър. В настолните приложения не е прието да се използва визуално дизайнерски ефекти каквито ги има на стандартните сайтове. Логично е да има функционално меню под ръка и недвижеща се горна секция, където в макета пише само контролен панел, но в реалното приложение е по-различно. Трябва да има разработена и цветова схема. Тя служи, не за да очароваме клиента, а да има някаква субективна реакция от страна на клиента на цвета. Подадените заявки, които не са обслужени, не са приети, въобще не са видени от фирмата са сини и всяка информация, която касае тях е в някакъв нюанс на синьото. Приетите и видяните от фирмите заявки са в червено, обработващите са оранжеви и приключените са зелени. Респективно и в таблиците се използват същите нюанси, но в един приглушен пастелен вид, така че да е агрономично като дизайн от една страна и от друга да е асоциативно и забележимо. Същото касае и timeline, където се идентифицират последните няколко действия и се следва същата цветова схема, общо взето налагаме цветовата схема като наложена функционалност върху дизайна. Самите цветове дават информация на потребителя и го разтоварват от необходимостта от пълнене на екрана с текст и цифри. Имам нова заявка, текуща заявка и закрита заявка и профилът евентуално да се промени. Аналогично е и от страна на персонала. Практически разликите там са две: вместо нова заявка има филтър по клиенти, защото персоналът работи с една фирма, с друга фирма и е логично да има филтър. Перспективно статистиката ще му дойде не само от тип задача, но ще му дойде и от тип фирма. Например най-много заявки от фирма A, най-обслужени са фирма B.

## Функционална структура

Структурата изглежда по следния начин: името на папката, името на темата, номерът на темата. Това е папката, която качвам в GitHub и респективно мога да си я сваля от GitHub. Вътре в темата има MyProject, в което е PyCharm проекта, който правя. Вътре в MyProject задължително има папки main и папка MyProject, и за да си стартирам проекта първото нещо, което трябва да направя е да отворя terminal и да въведа cd myproject, няма значение дали ще са главни или малки букви. След това, ако няма проблеми въвеждам python manage.py runserver и ако всичко е наред трябва да излезе една синя препратка и да няма други съобщения за грешки и чрез Ctrl + C се спира действието. В папката MyProject има папката работна. В нея задължително има папка DB, папка docs, папка downloads и папка HTML шаблони. В HTML шаблоните са работните шаблони, но един от тях, който се нарича base е базовият макет, с който работя. В папката docs задължително има заданието на проекта и много често добавям един файл, който се нарича log. Този файл е един вид дневник, в който пиша важни неща. В папката DB задължително има поне един файл, който се казва dzi\_(двуцифрен номер на темата).sql. Това е актуален експорт на базата данни, когато съм правил промени е правилно преди да започна работа да си стартирам XAMPP Control Panel, от MySQL избирам Admin и се отваря браузер, който се нарича localhost/phpmyadmin/. От тук си избирам темата и ми се отваря базата данни. Таблиците не ги отварям, за да не стане някакъв проблем, отивам на import, от там избор на файл, в случая DZI\_2024, моята тема, папката работна, в нея папката DB, в нея dzi\_(номерът на темата), гледам си датата и часът и се ориентирам кога последно е работено да не закача някой стар файл, защото после възстановяване няма, отварям го и по-долу натискам import. Ако случайно ми излезе грешка, това означава, че нещо е захапано. Нещото, което е захапано най-често е или тестови сървър, или някое работещо приложение. Ако ми излезе съответно, че индикацията е успешна , това означава че всичко е наред и докато не стигна до този вид индикация няма как да продължа напред. Проверка правя като си отворя съответната таблица, независимо от проекта, независимо как се казва базата данни, Django ги кръщава освен ако аз не задам по някакъв изричен начин нещо, което категорично не е препоръчително да го правя, даже и ако съм напреднал програмист е име на приложението и име на таблица. Например приложението main\_item и приложението main\_order, в приложението main\_item таблицата е item за приложението main, а в приложението main\_order таблицата е order в приложението main. В моя случай приложението main\_order са поръчките и виждам какво е съдържанието като натисна прелистване. Аз нищо не променям Django си го променя, просто се ориентирам дали всичко е наред и дали съм актуален. В auth\_users са потребителите, в моя случай имам двама потребители admin и test. Не ме интересува структурата на таблицата, просто си проверявам дали съм с правилната база данни. Ако примерно нямам инсталирана библиотека излизат едни сини съобщения, че нещо не е наред. Сигурно не си съм инсталирал Django REST framework, MySQL client и други просто го инсталирам, ако не позная какво не е инсталирано и го инсталирам два пъти не е проблем. Например имам инсталиран MySQL client, обаче се съмнявам и в terminal записвам командата pip install mysqlclient и съответно ми отговаря, че ми са удовлетворени изискванията. Инсталирано е вече, тоест не е проблем, ако съм сбъркал по-добре един път в повече да се опитам да си инсталирам, ако се съмнявам че нямам нещо, отколкото проблемът да е в това, в което си мисля че е инсталирано, а то да не е инсталирано.

## Резюме

Благодарение на класическите средства на статичните сайтове се получава динамика и функционалност. HTML и CSS, колкото и да са красиви и атрактивни не са достатъчни възможностите им, за да се реализира тази функционалност и затова естественият и избор все пак е Django и не просто Django, а същата тази концепция за REST, при която освен визуална информация между сървъра и клиента се обменят просто данни. Това приложение след един по внимателен анализ е всъщност съчетание на сайт представящ фирмата и на някакъв, макар и малък електронен магазин. От една страна не се опитвам да създам пълноценен електронен магазин с кошници, плащания, фактури и други, но имам основните елементи на електронна търговия и в същото време имам всички елементи, които са характерни за един представящ фирма сайт. Сайтът трябва да разполага със следните публично достъпни секции: “Начало”, “За нас”, “Каталог”, “Магазин” и “Контакти”. Секцията “Каталог” следва да може да активира on-line форма, за която да генерира поръчка с необходимите данни. Това се е отразило и върху разработката и като съм го прехвърлил то се е получило следното нещо: Като отида в моето приложение и погледна шаблоните виждам, че за всяка страница си имам шаблон, но ако се вгледам по подробно например страницата about, страницата index и страницата shop наследяват една обща страница base, която по същество представлява класическия за Django подход. По скоро за шаблонизатора Jinga, който позволява общите неща да се изнесат в един общ файл и да не преписвам на всяка страница едно и също меню, един и същи header, един и същи footer. Тези неща, които се повтарят от страница на страница е логично да са разработени на едно място и да се променят само някои детайли в тях, тоест някои променливи да променят стойностите си, а не да ги пращам всеки път на ново и най-малкото да си отворя поле за грешки. Наследявам файла base и от там нататък, ако се вгледам без да ги разисквам са класически шаблон и шаблонизатор. Всеки изглед генерира някакъв контекст, в случая имам меню, определям коя страница е активна, защото шаблонът е общ, но в зависимост от избора на потребителя активна е или страницата “Начало”, или страницата “За нас”, или страницата “Контакти”. Активната страница я подавам като контекст. Като контекст подавам и името на таблицата, подзаглавието на таблицата и някои други неща. Всичко това изглежда по един доста приятен начин със съответните дизайнерски акценти. Дизайнерският акцент в случая е взаимстван от шаблона, но е мой собствен, а не е заслуга на авторите на шаблона. Чрез подчертаване на един и същи елемент на различните страници по този начин създавам индивидуалност на фирмата пред клиента. Правя дизайнерски и програмистки трикове, за да постигна един прост и елегантен значителен ефект. Това беше страница “Начало”, сега отивам на страница “Магазин”, където визуалната концепция е същата, както и визуалните елементи footer, header и menu са същите. Същото е и при страницата “За нас” и върху шаблона base са изградени всички останали страници. Изгледите подават различен контекст, тоест на някакви променливи, които в шаблоните се изписват и им подавам различни значения. От тук нататък влизам в това нещо, което се нарича динамичен клиентски интерфейс, защото една от страниците е различна, а именно “Каталог”. Тя си има собствен header и footer, на пръв поглед изглежда по същия начин, но ако се вгледам директивата е различна. На практика тя заслепява шаблонизатора на Django, предава всичко едно към едно към клиента и позволява на шаблонизатора на Vue, той да обработи въпросния шаблон и той да го раздвижи, тоест подавам необходимата заготовка, за да може да се реализира функционалност от страна на клиента, защото внимателният анализ на заданието ме е навел на мисълта, че този клиент може да си разглежда каталога и да си попълва количка с часове и не е редно сървърът да бъде ангажиран с тази клиентска сесия през цялото време. Сървърът може да си обслужва други клиенти, а потребителят да влезе на страницата ”Каталог” да си ги разглежда и да си ги слага в количка. Това може да се превърне в безкраен процес, тоест клиентът не е длъжен въобще да го завърши и няма смисъл да се създават записи в базата данни, защото всичко това се съхранява локално при клиента, и когато клиентът види сумата, попълни си данните и изпрати поръчката, поръчката едва тогава се отразява на сървъра. Автоматично на практика Django администрацията ми предлага администраторски панел, който ми позволява да администрирам, да гледам какви поръчки са направени, мога да видя и детайлите на поръчката или да гледам, добавям и премахвам стока, тоест супер-потребителят може да прави всичко от администраторското меню. Единственото, което се иска от мен, за да направя такъв клиентски интерфейс е във файла admin да си регистрирам тези таблици от базата данни, които искам администраторът да ги управлява, наблюдава, променя и създава. Аз обаче съм избрал един разширен начин за регистриране на таблиците като не показвам всички полета, а си избирам с помощта на опцията list\_display си избирам кои полета да покажа в изгледа на административния панел и си избирам кое поле да е активно и като цъкна на него да ми отвори детайли. Това не е задължително, но е полезно при изграждане на администраторски интерфейси особено, когато се предполага, че той ще бъде често посещаван този административен панел и ще се работи с него. По същество тази страница “Каталог” е еднодокументна. Общо взето технологиите в света са всичко на всичко две: спорните, както е login страницата и с някаква динамика с помощта на JavaScript framework, в случая фреймуърка е Vue, защото е лесен и удобен. Vue е много по-голяма библиотека от това, което аз ползвам, целия интерфейс може да се изгради с Vue, но в моя случай не е необходимо, защото по един или друг начин аз интегрирам Vue с Django и позволявам на Django да свърши голямата част от работата, а ползвам Vue, само за да я съживя тази работа. Този подход е много привлекателен с простотата си, защото има три файла, един от които само аз пиша, а другите два са двете библиотеки, които се свалят от онлайн ресурс. Едното е самото Vue и другото е Axios, която е една мъничка и приятна библиотека, която използвам за комуникация между клиента и сървъра в посока по инициатива на клиента, тоест двупосочна е комуникацията, но е асинхронна. Пращам заявка и продължава да ми работи приложението без да е обвързано с получаването на отговор. Асинхронността е много важна в този случай. Файлът models описва моделите в базата данни, в случая моделната структура не е сложна, единствените особености са две: едната особеност я обвързвам и правя цяла една папка наречена media, защото реалните картинки се съхраняват в папката items, а в базата данни като такава се запазва само url, тоест пътят до съответния ресурс. Това, което виждам в клиентския интерфейс, особено при работа с поръчки са изчислени полета, които въобще не присъстват в базата данни. Когато знам единичната цена и бройката, изчислявам сумата, ако знам сумата и бройките ще изчисля единичната цена, тоест това са три параметъра, от които са нужни само два да се помнят, за да сметна третия и няма логика да ангажирам сървъра да изчислява нещо. Сървърите по принцип са скъп ресурс, трябва да ги икономисвам. В случая в моя шаблон полетата са обвързани с директивата v-model, а бутоните са обвързани с директивата at click. Обвързани са данни и действия с елементи на шаблона. Данните и действията са JavaScript обект в статичния ресурс, който се нарича logic, който следва структура, която е описана в библиотеката Vue. Първо обвързва HTML шаблона в тага app, после го обвързва с модела и се получава така, че едното работи пред очите на клиента, а другото зад завесата, тоест JavaScript обектът е зад завесата, а пред очите на клиента е HTML, но двете са обвързани. Всяка една функционалност, която работи на сайта за нея е реализирана някаква функция от страна на библиотеката Vue, а когато се налага да изтегля данни от сървъра като например да изтегля наличните стоки от каталога, за да покажа на клиента от какво избира, тогава използвам библиотеката ACSIUS като и подавам част от този интерфейс, който изграждам в рамките на тази концепция REST, като подавам два вида заявки: GET, ако просто изтеглям данни и POST, ако изпращам заявка и в двата случая, обаче използвам библиотеката ACSIUS. Тук спецификата е, ако трябва да обобщя е, че в един нормален, стандартен и обичаен на пръв поглед сайт, който би могъл да се разработи и само със средствата на HTML аз по един естествен и елегантен начин вмъквам по същество електронен магазин с цялата му функционалност, но запазвайки стилистиката на сайта като цяло без да натоварвам потребителя с необходимостта да прави нещо допълнително, тоест вграждам в моя сайт такова приложение, което макар и малко и с ограничена функционалност по същество си е електронен магазин с всичките си атрибути.

## Пояснения

В MyProject има база данни, шаблони и съм направил макет. Макетът е прехвърлен, създадено е едно приложение, което се нарича main. В settings е добавено REST framework, за да използвам Django REST framework библиотеката, която е инсталирана. Предполагаме, че работим с регистрирани потребители. Имам два потребителя: единият си е стандартният admin, а вторият е staff. Разликата е, че потребител staff, ако влезе в административния панел не вижда нищо, защото е просто служител, такива може да има много, а ако влезе admin тогава от административния панел има достъп до всички таблици, а таблиците са описани в models. Схемата е стандартна за Django, аз на практика не работя директно с базата данни, а с така наречения ORM (модели представени от обекти). Всъщност всеки клас описва един ред от съответната таблица. За да мога да тегля от приложението програмно само данни използвам библиотеката от страна на сървъра Django REST framework, а за да я използвам са ми необходими сереализатори. Сереализаторите съгласно документацията на DRF са в отделен файл. Всъщност сереализаторите просто им указвам модела, тоест таблицата от базата данни и казвам кои полета трябва да сереализара, например ако имам артикул, който се намира в склада в таблицата с артикули, за да може сереализаторът вместо число там да ми извади съответните данни от свързаната таблица дълбочината на сереализирането е 1, а вложените таблици не се сереализират. Единствената особеност е, когато имам две свързани таблици, обаче връзката е обратна. За да може да се направи обратна връзка се задава параметъра related \_ name. В сереализаторите правя следното нещо: този, който изважда операциите добавям поле op и за негова стойност на артикулите включени в заявки. Тъй като op е поле за обратна връзка, библиотеката DRF автоматично ще извади само тези записи, които отговарят на тази обратна връзка и всъщност ще ми добави към заявките съответните елементи. Имам и така наречените изгледи за REST услугите, това са изгледи, които връщат данни. В urls файла, за да различавам визуално услугите съм добавил един фиктивен път api, той ще се проследи, и ако е зададен ще се извика съответния изглед, който ще върне данни. Съгласно заданието ми, а именно системата да е достъпна за два вида потребители - администратор и клиент, аз съм решил реализацията ми да използва за данните, които се съхраняват на сървъра и за да не товаря връзката с постоянно обновяване между клиента и сървъра използвам концепцията REST, при която функционалността до голяма степен е изнесена при клиента, а между клиента и сървъра има само обмен на данни. Използвам стандартната система на Django за дефиниране, идентифициране и оторизиране на потребители. При създаването на проекта администраторът ми е достъпен на адрес admin и той може да добавя други потребители, в случая съм създал потребител с името staff със активен статус и със статус на персонал, но няма статус супер-потребител, защото той не е администратор. При влизане в системата използвам форма, която се обръща към системата на Django за разпознаване и работа с потребители. Това става като първо съм си създал шаблон login, създавам форма, определям метод post, задавам така наречения csrf\_token, която е система на Django за защита от ботове, скриптове и от външни атаки и тази форма се изпраща на сървъра и тогава сървърът преценява дали има такъв потребител и връща като отговор в една променлива стандартна за Django user, параметрите на потребителя, ако той е идентифициран като потребител в системата или не връща нищо, ако няма такъв потребител, тоест отговорността е на сървъра. След като изпратя формата на сървъра влизам в самата система, където HTML шаблонът, който съм използвал за основа ми помага да направя така наречения респонсив дизайн, тоест това нещо е достъпно за различен тип устройства. Използвам стилови набори, които са адаптирани за различна резолюция на клиентското устройство, всъщност шаблонът използва библиотеката Bootstrap в конкретния случай. Администраторът може да зададе и добави други потребители.

Всичките ми страници имат обща част, която включва скриптове, заглавна част – хедър и завършващата – фотер. Тя е изведена в базов шаблон base.html, който после се наследява от другите шаблони. Оставил съм само блок, който се попълва за съответния изглед, а всеки един файл например index наследява този базов шаблон, зарежда статичните ресурси, ако има такива и задава текст на блок page\_content. Когато от към страна на клиента работи библиотеката VUE, която ползва същия шаблонизатор, какъвто ползва и Django, за да стигнат до нея маркерите за шаблонизатора било то инструкции и променливи, изключам обработката на части от текста от страна на сървъра чрез директивата verbatim. Инструкцията endverbatin, изключва този режим. От страна на клиента включвам един JavaScript файл, който е създадена от мен и е разположен в static, в REST, който всъщност реализира функционалността на библиотеката VUE. Понеже имам четири шаблона, всеки си има своя VUE функционалност и по същество имам четири малки микроприложения.

# Заключение

Създаденото от мене приложение отговаря на изискванията на заданието за дипломен проект. То реализира зададената функционалност, има добре изглеждащ потребителски интерфейс, визуализира се правилно на устройства с различна резолюция.

Фактически моето приложение представлява динамичен уебсайт с два основни раздела - за клиенти и за дилъри. Сайтът има клиентски панел, панел за търговци на автомобили, търсене на автомобили под наем, управление на поръчките, управление на автомобили под наем, история на приходите и история на продажбите на всеки търговец.

Сайтът може да се администрира с помощта на администраторско меню, което позволява добавяне, редактиране или премахване на данни.

Дизайнът на сайта е адаптивен и изглежда еднакво добре на различни устройства.

Сайтът е разработен и достъпен за демонстриране на локаленния(тестовия) уеб сървър, който е част от Django.

Считам, че моята разработка би могла да се използва и в реалния бизнес с минимални корекции и доработки. Това зависи най-вече от специфичните особености на конкретния клиент, т.к. самото задание е за хипотетичен бизнес.

Работата по проекта беше интересна и полезна за мене.

# Използвана литература

1 Django: https://bg.wikipedia.org/wiki/Django

2 Django-admin и manage.py: https://docs.djangoproject.com/en/4.2/ref/django-admin/

3 URL адреси: https://docs.djangoproject.com/en/4.2/topics/http/urls/

4 Debugger: https://bg.wikipedia.org/wiki

5 Бази данни: https://bg.wikipedia.org/wiki

6 HTML: https://bg.wikipedia.org/wiki/HTML

7 Admin panel: https://www.webdesigngroup.biz/