**П Р И Р О Д О- М А Т Е М А Т И Ч Е С К А Г И М Н А З И Я „ Я Н Е С А Н Д А НСКИ„**



гр. Гоце Делчев, ул."Скопие" №4,

тел: директор: (0751) **60306**, секретар: **60308**, факс: **60308**,

e-mail: [**pmg.d HYPERLINK "mailto:pmg.direktor@"irektor HYPERLINK "mailto:pmg.direktor@"@**](mailto:pmg.direktor@)pmggd.bg, web site: [**http://www.pmg-gd- HYPERLINK "http://www.pmg-gd-bg.com/"bg. HYPERLINK "http://www.pmg-gd-bg.com/"com**](http://www.pmg-gd-bg.com/)

***ДИПЛОМЕН ПРОЕКТ***

*Тема: Уеб сайт на фирма за недвижими имоти „Домко“*

**Изготвил:**  **Ръководител-консултант**

Симона Николаева Огнянова Надя Воденичарова

12в клас

Професия „Системно програмиране“

Специалност „Системен програмист“

**гр. Гоце Делчев**

**2024 г.**

**Увод**

**Същност на проекта:**

Проектът представлява сайт на фирма за недвижими имоти. Той представя информация на фирмата и услугите, които предлага - наеми и продажби на имоти. Името на фирмата е “Домко”.

**Цели на проекта:**

* Проектът представлява сайт за управление на фирма за недвижими имоти. Целта е да се улесни работата на фирмата и да се придаде по-голяма популярност. Улеснява работата на собствениците и дава възможност на потребителите да намират по-лесно желаните от тях имоти.
* Сайтът съдържа филтри по няколко критерия, които да направят лесно намирането на подходящ за потребителите имот, според техните специфични изисквания, да запазват час за оглед и да оставят коментар към администратора, който да направи комуникацията по-лесна.

**Технологии:**

* Visual Studio Code - средата, в която разработих своя проект. Избрах да работя с него по няколко причини - има интуитивен интерфейс, който позволява лесно редактиране на кода, поддържа големи популярни програмни езици като JavaScript, Python, Java, C#, HTML, CSS и други, включва вграден терминал, който позволява да се изпълняват команди директно от редактора, има интегрирана поддръжка с Git, което ми помогна да работя по проекта си от различни устройства.
* MongoDB Atlas - предлага уеб-базирано управление на MongoDB инстанции, което прави конфигурирането и мониторинга на базите данни лесно и достъпно от всяко място, което ми даде възможност да работя от различни устройства, а не локално, само едно. Платформата автоматично мащабира база данни в реално време, в зависимост от натоварването и нуждите на приложението. Това гарантира висока производителност и наличност дори при големи обеми на данните. MongoDB Atlas предоставя множество възможности за сигурност, включително управление на достъпа. Платформата интегрира с други облачни услуги като AWS, Google Cloud и Azure, позволявайки да се изграждят пълноценни приложения в облачната среда с минимални усилия.
* Node.js - Express, Express-Handlebars, Bcrypt, Express-Sessions, Mongoose – позволява използване на JavaScript за програмиране на сървърната страна, което води до еднообразие в разработката и лесно преминаване между клиентския и сървърния код. Известен с асинхронния си модел на програмиране, който използва неблокиращ вход/изход (non-blocking I/O). Това позволява на сървърните приложения да обработват много заявки ефективно и да използват по-малко ресурси. Node.js разполага с голяма и активна общност, която разработва разнообразни модули (пакети), достъпни чрез npm (Node Package Manager). Тези модули могат да се използват за добавяне на различни функционалности към приложенията, като например работа с бази данни, обработка на изображения, създаване на API и други. Благодарение на асинхронния си модел и ефективността на изпълнение, Node.js е изключително подходящ за скалируеми приложения, които трябва да се справят с голям брой едновременни заявки и натоварване.
* Paint.Net - Paint.NET има интуитивен и лесен за използване потребителски интерфейс, който прави редактирането на изображенията достъпно дори за начинаещи потребители. Менюта, панели и инструменти са организирани логично и предлагат достъп до различните функционалности на програмата. С помощта на Paint.NET потребителите могат да извършват базови редакционни операции като обръщане, обрязване, мащабиране и повторно обекти на изображенията. Paint.NET включва множество вградени филтри и ефекти, които могат да бъдат приложени към изображенията за тяхното подобряване или промяна. Тези включват размекване, острие, наситеност, цветова корекция и други. Въпреки че е по-основен, Paint.NET включва и поддръжка на слоеве, която позволява на потребителите да работят с различни елементи на изображението независимо един от друг. Paint.NET поддържа различни формати на файлове, включително JPEG, PNG, BMP, GIF и други. Това позволява на потребителите да импортират и експортират своите изображения в различни формати според техните нужди. Избрах програмата, защото е лесна за използване и предлага големи възможности. Използвах я, за да изчертая логото на проекта.
* Express - Express е минималистична уеб рамка (framework) за сървърно програмиране в JavaScript, която е базирана на платформата Node.js. Тя улеснява създаването на уеб приложения и API-та чрез предоставяне на набор от инструменти и функции за обработка на HTTP заявки и отговори, управление на маршрути, работа със сесии, аутентикация, мидълуеъри и други. Express предоставя прост и удобен начин за дефиниране на URL маршрути и тяхната обработка. Маршрутите могат да бъдат настроени за обработка на HTTP заявки от методи като GET, POST, PUT и DELETE. Middleware е основен концепт в Express, който позволява на приложенията да изпълняват последователен ред от функции преди или след обработката на маршрута. Това позволява на разработчиците да добавят функционалности като логване, аутентикация, авторизация и други към приложението си. Express предоставя лесни за използване методи за обработка на HTTP заявки и генериране на HTTP отговори. Това включва възможността за четене на данни от тялото на заявката, изпращане на статус кодове и заглавия, както и изпращане на отговори с различни формати като HTML, JSON и други. Express е много гъвкав и лесно се интегрира с различни библиотеки и модули от общността. Това позволява на разработчиците да използват допълнителни функционалности и да разширяват възможностите на своите приложения. Express поддържа различни шаблонни двигатели като Pug (преди известен като Jade), Handlebars, EJS и други, които позволяват на разработчиците да генерират HTML страници динамично и лесно. Поради своята гъвкавост и модулна архитектура, Express позволява на разработчиците да създават много различни видове уеб приложения - от прости блогове и сайтове до сложни уеб приложения и API-та.
* Express-Handlebars - шаблонен двигател за Node.js, който позволява лесно създаване на HTML шаблони с динамично вграждане на данни. Той е вдъхновен от Handlebars.js и е специално проектиран за използване с Express.js. Избрах го защото е: синтактично опростен - лесен за разбиране и писане. Базира се на стандартен HTML, като позволява вграждане на JavaScript изрази за изчисляване на стойности или за обхождане на данни; Лесно се интегрира с Express.js - той е специално създаден за интеграция с Express.js, което ми позволи да го настроя лесно в моето Express приложение; Той е мощен и гъвкав - предлага богат набор от функционалности, включително условни оператори, цикли, вграждани шаблони и други. Това го прави подходящ за създаване на различни видове шаблони, от прости HTML страници до сложни уеб приложения; Поддръжа частични шаблони - Позволи ми да създам частични шаблони за често използвани елементи на страницата като хедъри, футери или навигационни менюта и да ги включа лесно в други шаблони; Позволява достъпност до данни - лесно можете да се подават данни към шаблоните от Express.js приложението, което ги прави идеални за визуализиране на динамични съдържания, включително данни от база данни или външни API-та; Дава възможност за поддръжка на разширения и помощни функции - позволява лесно дефиниране на собствени помощни функции и разширения, което ми позволи да разширя функционалността на шаблоните си според моите нужди.
* Bcrypt - алгоритъм за криптиране на пароли, който е разработен с цел повишаване на сигурността на потребителските пароли. Той използва медицински изпитан метод на криптиране, наречен "криптография със сол" (salted cryptography), за да затрудни атаките на пароли, включително атаки срещу хеширането на пароли. Избрах да работя с него, защото: Притежава сигурност - известен е с високата си степен на сигурност. Той използва т. нар. сол (случайна стойност), която се добавя към паролата преди да се хешира. Това усложнява атаките срещу таблиците с хеширани пароли (rainbow table attacks) и атаките срещу пароли с общо използвани хешове; Итеративен е - изграден е от алгоритъм, който позволява на разработчиците да зададат броя итерации за използване при хеширането на паролата. По-голямото количество итерации прави атаките на базата на пароли по-трудни и по-бавни; Съвместимост - Bcrypt е наличен за много различни програмни езици и платформи, което го прави лесен за интеграция в различни системи. Работи много добре с JavaScript и Node.js; Актуалност - Bcrypt е проектиран да се справя със съвременните методи за атака и криптоанализ. Това го прави подходящ за използване дори във високорискови среди, където сигурността е критична.
* Express-session - middleware за Node.js, който се използва във framework-а Express за управление на сесии в уеб приложенията. Сесиите са механизъм за запазване на състоянието на потребителите между различните заявки към сървъра и представляват важен инструмент за управление на сесиите на потребителите и управление на сигурността. Защо го избрах: Дава възможност за сесионно управление - предоставя удобен начин за създаване и управление на сесии за потребителите на уеб приложението. Това включва създаване на нова сесия, запазване на данни в сесията и изтриване на сесии; Гъвкавост - много гъвкав и позволява на разработчиците да конфигурират различни опции, като например методите за съхранение на сесии, времето на изтичане на сесията, допълнителни настройки за сигурност и други; Сигурност - при правилната конфигурация, express-session може да помогне за предотвратяване на атаки срещу сесиите, като например атаки със сесионни куки, фиксиране на сесии, CSRF (между-сайтово препращане на код) и други; Лесна интеграция - лесно за интегриране в приложения, които използват Express.js, което прави сесионното управление достъпно и удобно за моя проект.
* Mongoose - библиотека за моделиране на обекти за MongoDB и Node.js. Тя предоставя прост и елегантен начин за работа с MongoDB бази от данни, като добавя допълнителни функционалности и улеснява комуникацията между приложението и базата от данни. Защо го избрах: Моделиране на данни: Mongoose позволява да дефинирате структура и схема за вашите данни в MongoDB. Това включва дефиниране на модели, които отразяват различните типове данни, валидации, подразбирателни стойности и други характеристики на вашите данни. Защо го избрах: Валидация на данни - предлага вградена валидация на данни, която позволява да се гарантира, че данните, които влизат в базата от данни, отговарят на определени критерии или правила; Удобство на работа с базата данни - предоставя удобен API за извършване на различни операции с базата данни като създаване, четене, актуализиране и изтриване на документи. Този API улеснява и оптимизира комуникацията между приложението и MongoDB; Поддръжка на middleware - предоставя middleware функционалност, която позволява да се изпълняват определени действия преди или след извършването на определени операции върху данните. Това включва вградени възможност за валидация, бработка на данни при изпращане и достъпване и други; Притежава вградени типове на данни - Mongoose предлага множество вградени типове данни, които могат да бъдат използвани за дефиниране на полета в моделите на базата данни, като например текстови низове, числа, масиви, вложени обекти и други.

**Основна част**

* **Етапи на реализация**

*Иницииране:*

* Създава се концепция на сайта - примерен шаблон на изгледа, планира се user experience и се предлагат подходящи цветове и изображения;
* Формулират се целите - определя се аудиторията, към която е насочен и как ще се използва.

*Планиране:*

* Събиране на подходяща информация за подобен тип проекти - какво биха желали клиентите, как биха искали да се използва и какво оценяват потребителите. Обмислят се технологиите, които ще се използват и се избират най-подходящите, за да може сайтът да работи максимално бързо и ефективно.
* Създаване на архитектурата на сайта - създават се шаблони и файлове за всяка страница от приложението. Файловете се разделят в папки, които трябва да бъдат максимално ясни и добре подредени.
* Създаване на функционалност на приложението - изгражда се базово html съдържание на страниците, оформят се схемите на базите данни, създават се заявки към тях, подреждат се архитектурно в специално създадени папки.
* Създаване на дизайна - оформят се всички компоненти от страниците - бутони, текстови полета, изображения, цветове, секции и др., за да може да се използва максимално лесно и да бъде достъпен за всеки потребител и за всяко устройство.
* Създаване на авторско лого - оформяне на малко изображение, което да дава ясна информация за името на фирмата и целите, но и да не е прекалено натрупано.

*Изпълнение:*

* Събиране на основна информация и изисквания за сайтове от типа.
* Избиране на подходящи технологии.
* Създаване на план за работа и разпределяне на задачите по дни.
* Създаване на архитектурата на сайта - оформят се файлове за различните страници, реализират се в шаблони, осъществява се тяхното функционално стартиране.
* Добавяне на съдържание в страницата “За нас | Администратор” - базова форма, чрез която се редактира страницата на потребителя динамично. Променя се заглавието на страницата.
* Добавяне на съдържание в “Наеми | Администратор” - под формата на карти се визуализират съществуващите имоти. Картите съдържат бутони “изтрий” и “редактирай”, които по-късно ще водят към страници за редактиране и изтриване на конкретния имот. Създаден е бутон “Създай имот под наем”, който ще води към страницата за създаване на имоти под наем. Променя се заглавието на страницата.
* Добавяне на съдържание в “Продажби | Администратор” - под формата на карти се визуализират съществуващите имоти. Картите съдържат бутони “изтрий” и “редактирай”, които по-късно ще водят към страници за редактиране и изтриване на конкретния имот. Създаден е бутон “Създай имот за продажба”, който ще води към страницата за създаване на имоти за продажба. Променя се заглавието на страницата.
* Премахва се “Галерия” от навигационния бар за администратора - необходимо е, защото администраторът ще създава изображенията към самите карти и ще ги вижда в момента.
* Добавяне на съдържание в “Контакти | Администратор” - базова форма, чрез която се редактира страницата на потребителя динамично. Променя се заглавието на страницата.
* Добавяне на съдържание в “За нас | Домко” - добавяне на секции и параграфи в страницата за потребителя, които по-късно ще зареждат информацията, създадена от администратора.
* Добавяне на съдържание в “Наеми | Домко” - под формата на карти се визуализират съществуващите имоти. Картите съдържат бутон “Галерия”, който по-късно ще води към галерията на конкретния имот.
* Добавяне на съдържание в “Продажби | Домко” - под формата на карти се визуализират съществуващите имоти. Картите съдържат бутон “Галерия”, който по-късно ще води към галерията на конкретния имот.
* Премахва се “Галерия” от навигационния бар за потребителя - необходимо е, защото самите карти на имотите ще имат бутон “Галерия”, който ще води към повече изображения на имотите и по-конкретна информация за тях.
* Добавяне на съдържание в “Контакти | Домко” - добавя се компонент, който съдържа информация за контактите на фирмата. По-късно информацията ще бъде създавана от администратор и ще се зарежда от база данни.
* Промяна на “Количка” в навигационния бар - заменя се с икона и се поставя в дясната страна на навигационния бар.
* Добавяне на съдържание в “Количка | Домко” - добавя се компонент, който съдържа информация за имота, който е добавен за оглед. Страницата съдържа бутон, който води към страница за въвеждане на данните за желаната резервация.
* Създаване на страница “Завършване на резервация | Домко” - страницата съдържа форма, в която да се въвеждат данните на потребителя, датата и часа, на които да се извърши огледа. Бутонът изпрати изпраща данните към администратора и изчиства количката.
* Добавяне на долна част за всички страници - секция, която отделя края на страницата от останалите компоненти.
* Добавяне на страница “Създаване на администратор | Администратор” - съдържа html форма, в която да бъде въведено потребителско име на администратор, парола и повторение на паролата, а после тези данни да бъдат предоставени на новия админ, за да се използват.
* Добавяне на страница “Вход за администратор | Администратор” - съдържа html форма, в която да бъде въведено потребителско име и паролата. Тези данни ще бъдат предварително създадени от стария администратор.
* Създаване на динамичен навигационен бар към страницата на администратора - ако има сесия на админ (администраторът е влязъл в профила си), навигационния бар дава достъп до другите страници, ако не е влязъл, достъпът е само до страницата за влизане.
* Създава на заявка към базите данни за регистрация на администратор - взимат се данните от формата и чрез пост заявка се изпращат към базите. С помощта на bcrypt паролата се хешира и се изпраща криптирана към сървъра, за да може ако профилът бъде хакнат, паролите да бъдат защитени.
* Създава на заявка към базите данни за вход на администратора - взимат се данните от формата и чрез пост заявка се изпращат към базите. Чрез потребителското име се намира администратора от базите данни. Ако потребителското име е намерено в базата, с помощта на bcrypt, паролата се сравнява с въведената от потребителя и ако съвпада с паролата от базите данни, то администраторът влиза успешно. Данни: username: admin123, парола: admin123
* Създава се заявка съм базите данни за създаване на съдържание на страницата “За нас | Домко” - Администраторът въвежда съдържание във форма, което съдържание ще се визуализира на потребителя.
* Променя се страницата “Наеми | Администратор” - поставя се условие, ако има имот, за всеки имот се създава отделна визуализация, ако няма имот, излиза текст: “Няма създадени имоти под наем”.
* Във файлa, който рендерира началната страница е създадена проверка. Ако администраторът е влязъл в профила си, на “/admin” се появява началната страница, ако не е влязъл същия адрес води към страницата за влизане.
* Добавяне на страница “Създаване на имот под наем | Администратор” - създава се страница, която съдържа форма за попълване на данни за конкретния имот. В нея ще бъдат въведени различни характеристики на имота, които след това ще бъдат визуализирани на потребителя.
* Добавяне на пост заявка към страницата “Създаване на имот под наем | Администратор” - заявката изпраща въведените данни към базите данни, където те се запазват.
* Добавяне на get заявка към страницата “Създаване на имот под наем | Администратор” - заявката взима всички създадени данни от базата данни за наеми и чрез темплейт създава карта за всеки съществуващ имот.
* Добавяне на страница “Създаване на имот за продажба | Администратор” - създава се страница, която съдържа форма за попълване на данни за конкретния имот. В нея ще бъдат въведени различни характеристики на имота, които след това ще бъдат визуализирани на потребителя.
* Добавяне на пост заявка към страницата “Създаване на имот за продажба | Администратор” - заявката изпраща въведените данни към базите данни, където те се запазват.
* Добавяне на get заявка към страницата “Създаване на имот за продажба | Администратор” - заявката взима всички създадени данни от базата данни за продажби и чрез темплейт създава карта за всеки съществуващ имот.
* Добавяне на delete заявка към страницата “Наеми | Администратор” - когато бутонът “Изтрий” бъде натиснат се появява прозорец за потвърждение и ако събитието бъде потвърдено, данните от базите данни се изтриват и имотът се премахва от визуализацията.
* Добавяне на страница “Редактиране на имоти под наем | Администратор” - форма, в която да се въвеждат нови данни за имота и да се редактират в базата данни.
* Добавяне на пост заявка към страницата “Редактиране на имот под наем | Администратор” - намира се конкретният имот по id в базата данни и се редактират неговите характеристики.
* Добавяне на страница “Редактиране на имоти за продажба | Администратор” - форма, в която да се въвеждат нови данни за имота и да се редактират в базата данни.
* Добавяне на пост заявка към страницата “Редактиране на имот за продажба | Администратор” - намира се конкретният имот по id в базата данни и се редактират неговите характеристики.
* Добавяне на get заявка към “За нас | Домко” - Заявката взима данните, създадени от администратора и ги поставя на местата, където потребителят трябва да ги вижда.
* Добавяне на get заявка към “Наеми | Домко” - заявката взима всички създадени данни от базата данни за наеми и чрез темплейт създава карта за всеки съществуващ имот.
* Добавяне на get заявка към “Продажби | Домко” - заявката взима всички създадени данни от базата данни за наеми и чрез темплейт създава карта за всеки съществуващ имот.
* Добавяне на get заявка към “Контакти | Домко” - заявката взима всички създадени данни от базата данни за контакти и чрез темплейт създава компонент, в който да се визуализират данните.
* Добавяне на съдържание в страницата “Галерия | Домко” - след като е натиснат бутонът “Галерия” от “Наеми | Домко”, задейства се събитие, което взима Id на конкретния апартамент и води към собствената му галерия. Чрез get заявка се вземат данните за конкретния апартамент и се изпращат към темплейта, в който ще бъдат визуализирани данните.
* Добавяне на съдържание в страницата “Галерия | Домко” - след като е натиснат бутонът “Галерия” от “Продажби | Домко”, задейства се събитие, което взима Id на конкретния апартамент и води към собствената му галерия. Чрез get заявка се вземат данните за конкретния апартамент и се изпращат към темплейта, в който ще бъдат визуализирани.
* Създаване на post заявка за добавяне в количката - след натискане на бутона “Добави в количката”, всички заредени данни в галерията се изпращат към базата данни за добавяне в количката.
* Създаване на get заявка за визуализация на добавените данни в количката - Заявката взима всички данни от базите данни и ги визуализира за потребителя в страницата на количката.
* Създаване на delete заявка към страницата “Количка | Домко” - когато е натиснат бутонът “Премахни” се изпраща заявка до базите данни, която изтрива имота от количката.
* Добавяне на страница “Регистрация | Домко” - съдържа html форма, в която да бъде въведено потребителско име, имейл, парола и повторение на паролата, а после тези данни да бъдат изпратени към базата данни.
* Добавяне на post заявка към страницата “Регистрация | Домко” - заявката взима данните от формата, валидира ги и ги изпраща към базите данни, където да се съхраняват.
* Добавяне на страница “Вход | Домко” - съдържа html форма, в която да бъде въведено потребителско име, имейл и парола, която след това ще бъдат обработени от заявка.
* Добавяне на post заявка към страницата “Вход | Домко” - заявката взима данните от формата, проверява дали съществуват тези данни, сравнява ги и ако потребителят има създаден профил, създава сесия и го води към началната страница.
* Добавяне на ownerId към модела на базата данни за количката - необходимо е, за да в количката да се визуализират данни само на потребителя, който ги е създал.
* Добавяне на визуализация за потребителското име в навигационния бар, ако потребителят е влезнал.
* Добавяне на изход за потребителя - заявката изтрива сесията на потребителя и е необходимо отново да влезе в профила си.
* Добавяне на изход за администратора - заявката изтрива сесията на админа и е необходимо отново да влезе в профила си.
* Добавяне на “totalPrice” в темплейта за количката - извежда се общата цена за оглед на всички добавени продукти в количката
* Редактиране на формуляра за завършване на резервация - текстовите полета се заменят с полета за дата и час, за да бъде по-удобно за потребителите при резервация.
* Създаване на заявка за приключване на резервация - потребителят въвежда имената си и желания час и дата, които след това се запазват в базата данни, а данните от количката се изтриват.
* Добавяне на страница “Резервации | Администратор” - страница, която ще визуализира направените резервации от потребителите.
* Добавяне на заявка за персонално извеждане на данните. Взимат се всички апартаменти, които потребителят е запазил и се визуализират в страницата на администратора.
* Добавяне на reservationChaking заявка - когато бъде натисната иконката с тикче, се изпълнява заявка, която отбелязва резервацията като изпълнена и я изтрива от базите данни.
* Създаване на страница, в която ще се визуализират филтрираните данни. След като бъдат взети от базата данни, ще се извеждат в нея. Ако данни с тези характеристики не съществуват, ще се извежда текст “Няма създадени имоти”.
* Добавяне на филтриране към “Наеми | Администратор” - въвеждат се два критерия (Тип на имота и местоположение), които изпращат заявка и визуализират имотите, които имат тези характеристики.
* Добавяне на филтриране към “Продажби | Администратор” - въвеждат се два критерия (Тип на имота и местоположение), които изпращат заявка и визуализират имотите, които имат тези характеристики.
* Добавяне на страница “Потребители със съобщения | Администратор” - страницата ще визуализира имената на клиентите оставили съобщения от клиентската страница.
* Добавяне на форма за съобщение в страницата “Контакти | Домко” - в нея потребителят ще има възможност да остави данните си и съобщението, които ще бъдат изпратени на администратора.
* Добавяне на post заявка към “Контакти | Домко” за изпращане на съобщение от потребителя.
* Създаване на get заявка към “Потребители със съобщения | Администратор” - заявката взима всички съобщения от базата данни и визуализира имената на потребителите, оставили съобщения.
* Създаване на get заявка към “Съобщения от потребители | Администратор” - заявката взима съобщението на конкретния потребител и го визуализира в страницата на администратора.
* Създаване на дизайн на отдела за потребителя. Оформят се изображенията, цветовете и се избира подходящ шрифт.
* Създаване на дизайн на отдела за администратор. Оформят се изображенията, цветовете и се избира подходящ шрифт.

*Приключване:*

* Проверява се дали възложените цели са изпълнени - приложението е максимално достъпно ли е за избраната аудитория, дава ли се достатъчно ясна информация, дали функционално работи.
* Изготвя се цялостна документация - създава се теоретична част с всички поставени изисквания, описва се проекта и се прави цялостен извод.
* Предаване на проекта - готовият проект се предава на възложителите за оценка. Теоретичната и практическата част се представят пред публика.