Природоматематическа гимназия
“Христо Смирненски”

гр. Перник

Дипломен проект

ТЕМА:
РАЗРАБОТКА НА УЕБ САЙТ “Nasluka”

професия код 481030 „Приложен програмист“
специалност код 4810301 „Приложно програмиране“

Изработил:
Ивайло от 12 б. клас, № 11

Дипломен ръководител
Р. Василев

2022

Увод

1. Обект на изследване - уеб сайт реализиран с MVC модел.

-Предмет на изследване- уеб сайт „Туристическа агенция“, като съвкупност от съдържание, което трябва да достигне определена аудитория (хора, които търсят такива услуги, каквито ще бъдат предложени от сайта) и желаната функционалност.

1.1 Задачата:

Целта на задачата ми, възложена на изпита за придобиване на професионална квалификация е да се направи: Уеб приложение “Наслука”. Малко повече за самият сайт:

Рибарски магазин, в който да има различни рибарски принадлежности (Въдици, Макари, Корди...), с опция за преглед, достъп до информация за тях (Описание на продуктите, цената им, наличност...), както също и опция за резервиране, доставка и закупуване онлайн. Плащане с карта или в брой.

1.2 Особености на приложението:

Приложението трябва да е изградено на три нива, като на най-високото трябва да е администраторът на страницата (човекът имащ право да прави промени по количествата, да наблюдава потребителите в сайта, тяхната дейност и тн.). А на най-обикновеното стъпало да е обикновеният потребител, който има право само да разглежда и да се достъпва до информацията в приложението. За да се постигнат тези резултати и желаното оформление, ще бъде използван софтуер предоставен от “Microsoft”. Ще бъдат изградени база от данни и различни на тип елементи в средата за програмиране, чрез които ще имаме контрол над методите и случващото се в приложението. Елементите трябва да се съобразят с контекста на задачата и нейната функционалност и цел, което се прави на програмно ниво и невидимо в последствие за обикновения потребител в нашия сайт. Проектирането (планирането) е част от всеки проект . В процеса на проектиране се откриват конкретни задачи свързани със самия проект. Важна част от създаването на дипломния проект е определяне на изискванията и техния анализ. За да създам сайта ще премина през различните етапи на проектиране. Реализацията на сайта ще бъде направено в следващата глава, а тестването ще се извършва по време на

разработка. За моя сайт ще използвам NET технологията, в частност .NET Framework и NET Core. Microsoft .NET Framework е платформа, създадена от Microsoft, която предоставя програмен модел, библиотека от класове и среда за изпълнение на написан специално за нея програмен код, докато NET Core е безплатна и с отворен код, управлявана компютърна софтуерна рамка за операционни системи Windows , Linux и macOS. Той е кросплатформен наследник на .NET Framework.

1.3 Софтуер за изграждането:

MVC шаблона ще оформи визуалното и логическото съдържание на приложението, така че то да се приведе във вид и функционалността му да има правилната логика на работа, за да се изпълняват ежедневни нужди на потребителите му. Използваме този софтуер, заради лесният му контрол, надеждната защита и най-вече наборът от функции и връзки между отделните компоненти, позволяващи тяхната работа и лесна промяна при нужда. Очакваме в най-кратък срок резултати, отговарящи на нуждите на пазара, достатъчно гъвкави, за да откликват на всякакви модификации без намесата на специализиран персонал. Тоест да се направи такава функционалността, че да не налага при смяна на елементи по сайта човек да бъде експерт в областта. Също Microsoft .NET Framework и NET Core,Visual Studio Community 2022, SQL Menagment Studio.

Проучване

2.1 Предпоставки:

Има много причини за създаване на софтуерен продукт. Неговата реализация е свързана с необходимостта от *проучването на пазара*, дали има реален потенциал да бъде успешно направен и пуснат в действие. Реализацията на софтуер трябва да удовлетворява както изискванията на собственика на продукта така и изискванията на крайните потребители, като задоволява техните потребности. Бързо променящите се технологии налагат още повече да се направи добро *проучване на пазара* с цел да се избере възможно най-добрият подход за реализация, дългосрочно функциониране, бъдещо подобрение и поддръжка на продукта.

2.2 Търсене на пазара:

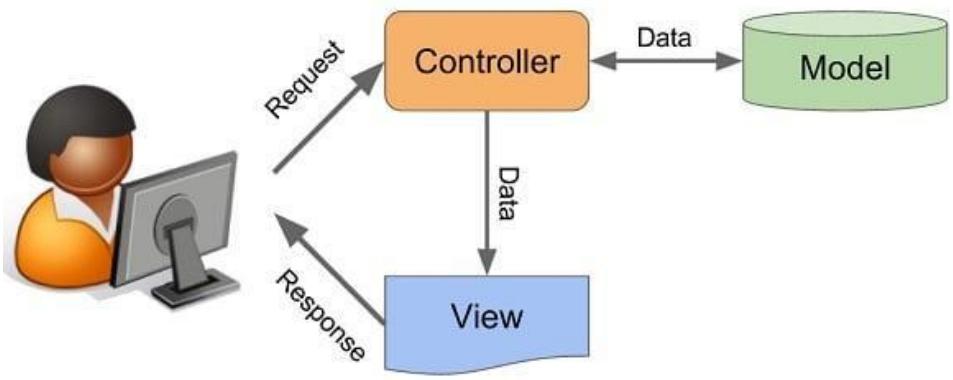
Такива програми има доста, но съм се заел да правя в частност за рибарски

магазин, поради големия интерес на потребителите. В момента на пазара има дефицит на софтуерни приложения от този тип, затова смятам, че тази програма ще пожъне успех и ще улесни доста хората занимаващи се с риболов. Ще се стремя да я направя по стандарти, които не отстъпват по нищо на големите страници и приложения.

2.3 Технологии:

В основата му е MVC архитектурата, които остава един от най-добрите инструменти за този тип задачи. В частност ще използваме ASP.NET Core, MVC, SQ Management Studio - за работа с базата от данни, HTML, CSS, JavaScript, Bootstrap -за презентационния слой, както и трислойния модел. Използвам тези технологии, заради гъвкавостта им, надеждността и разнообразието от функции, които предлагат.

-**MVC шаблонът**: Model–view–controller (MVC) е архитектурен шаблон, показва как се подреждат папките, класовете и др. компоненти на приложението. Създаден е през 1970 от Trygve Reenskaug, като част от Smalltalk. Позволява преизползване и разделение на кода. Първоначално е разработен за настолни приложения, но после идеята е прехвърлена и в уеб приложенията.



Входящата заявка се пренасочва към контролер.

За уеб: HTTP Заявка Контролерът обработва заявка и създава презентационен Модел Controller също избира подходящ резултат (например: View).

Моделът се предава на изгледа Изгледът преобразува Модела в подходящ изходен формат (HTML).

Отговорът е предоставен (HTTP отговор) MVC.

Трислойния модел: Най-разпространената форма на многослойна архитектура е трислойната архитектура. Трислойната архитектура обикновено се състои от:

Презентационен слой,

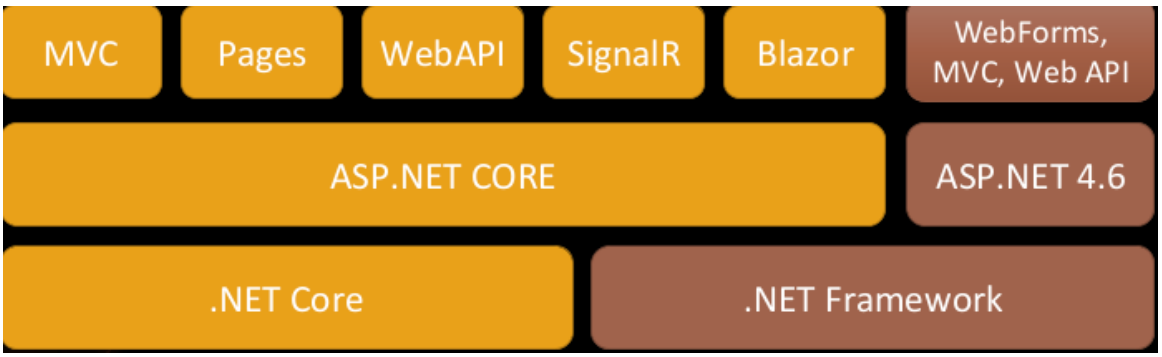
-Слой за услуги,

-Слой за данни.

Това е модел, при който: Приложението е разпределено на слоеве. Всеки слой има строго определена задача. Във Visual Studio можем да създадем такова приложение, създавайки различни проекти в рамките на Solution-а ни.

ASP.NET Core е уеб платформа с отворен код. Можете да изграждате уеб приложения и сървиси, IoT приложения, мобилни приложения и всяко уеб-базирано решение с ASP.NET Core.

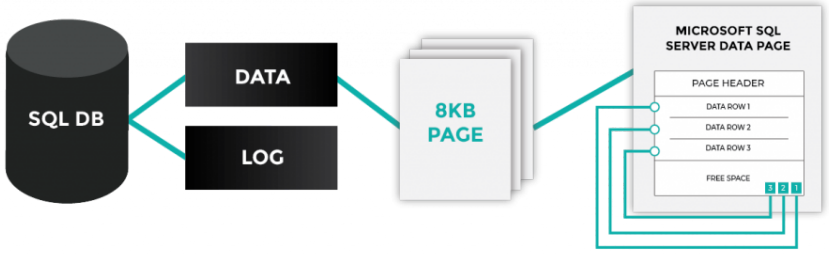
Главните плюсове са, че Унифицирана рамка за изграждане на уеб потребителски интерфейс и уеб API, лесно за тестване Възможност за разработване и изпълнение под Windows, macOS и Linux Възможност за хостване на IIS, Nginx, Apache, Docker или self-host на собствена машина в собствен процес Вградена dependency injection.



ASP.NET vs ASP.NET Core:

Една от основните причини да избера ASP.Net Core пред ASP.NET, е че ASP.NET Core е сравнително нова рамка с отворен код, също че е междуплатформена рамка, Подходящ за изграждане на модерни, облачно базирани уеб приложения на Windows, macOS или Linux.

Microsoft SQL Server е сървърна система за управление на база от данни. Избрах нея, заради надеждността ѝ, както и заради широкият спектър от функции. Предназначена е за управление на големи сървърно базирани БД, за разлика от MS Access, която е desktop базирана и не е предназначена за управление на големи корпоративни БД.



Microsoft SQ Management Studio това е софтуерно приложение, стартирано за първи път с Microsoft SQL Server 2005, което се използва за конфигуриране, управление и администриране на всички компоненти в SQL Server. То съчетава широка група от графични инструменти с голям брой текстови редактори, осигуряващи на разработчиците и администраторите всички нива на достъп до сървъра. Водещ елемент в SSMS е Object Explorer, който позволява на потребителя да търси, избира и да работи с всеки от обектите на сървъра. Приложението има и „експресна“ версия, която може да бъде изтеглена безплатно.

HTML (съкращение от термина на английски: HyperText Markup Language, произнасяно най-често като „ейч-ти-ем-ел“, в превод „*език за маркиране на хипертекст*“) е основният маркиращ език за описание и дизайн на уеб страници. HTML е стандарт в интернет, а правилата се определят от международния консорциум W3C. Текущата версия на стандарта е HTML 5.0 (от 28 октомври 2014 г.).

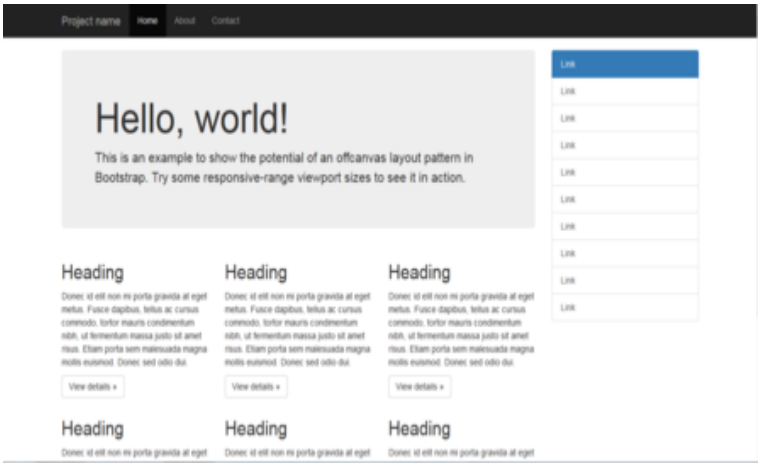
CSS е създаден с цел да бъдат разделени съдържанието и структурата на уеб страниците отделно от тяхното визуално представяне. Преди стандартите за CSS, установени от W3C през 1995 г., съдържанието на сайтовете и стила на техния дизайн са писани в една и съща HTML страницата. В резултат на това HTML кодът 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.

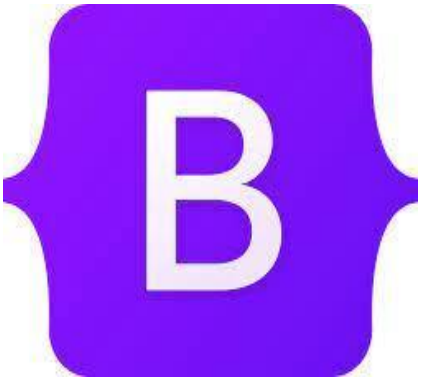
JavaScript е интерпретируем език за програмиране, разпространяван с повечето уеб браузъри. Поддържа обектно ориентиран и функционален стил на програмиране. Създаден е в Netscape през 1995 г. Най-често се прилага към HTML-а на интернет страница с цел добавяне на функционалност и зареждане на данни. Може да се ползва също за писане на сървърни скриптове JSON, както и за много други приложения. JavaScript не трябва да се бърка с Java, съвпадението на имената е резултат от маркетингово решение на Netscape. Javascript е стандартизиран под името EcmaScript.

Bootstrap е client-side среда с отворен код, която съдържа набор от инструменти за създаване на уеб приложения и уеб сайтове.

Bootstrap е пуснат през 2011 г. от Twitter, след като стартира като затворена библиотека, създадена за вътрешна употреба на Twitter.

Към декември 2015 г. проектът Bootstrap е вторият предпочитан проект на GitHub, с повече от 100 000 звезди и 45 000 връзки.





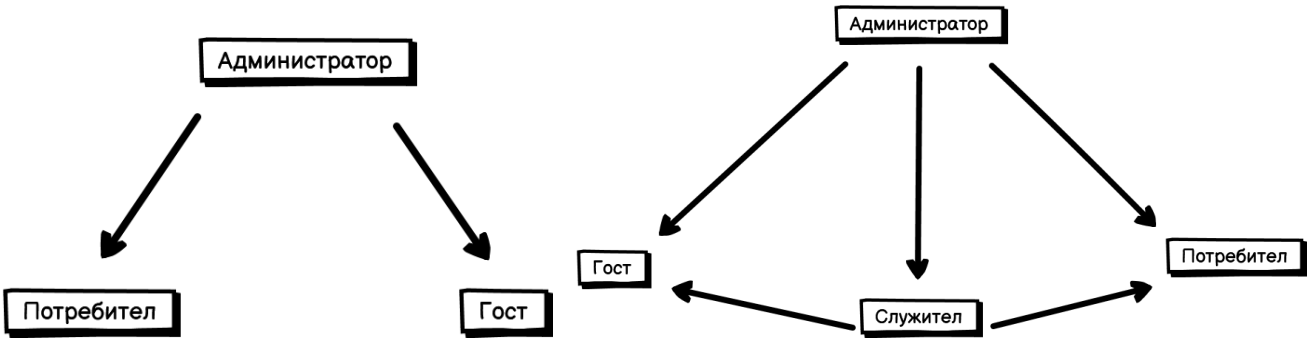
Това е наборът ми от инструменти и технологии, които ще използвам за проекта си. Съобразен е с целта и идеите, които стоят зад заданието ми и е така подбран, че да може да изпълни необходимите функции, както подобава.

Случаи на употреба (use cases)

3.0 Случаите

Случаите, които могат да бъдат срещнати в нашето приложение са няколко: Гост- регистриран потребител- администратор.

В други са гост-регистриран потребител-служител- администратор



3.1 Предимства

От самото начало на agile движението, техниката на потребителските истории от Extreme Programming е толкова популярна, че мнозина смятат, че това е единственото и най-доброто решение за гъвкавите изисквания на всички проекти. Алистър Кокбърн изброява пет причини, поради които все още пише случаи на употреба в гъвкава разработка.

1. Списъкът с имена на цели предоставя най -*краткото* обобщение на това, което системата ще предложи (дори от потребителските истории). Той също така предоставя скелет за планиране на проекта, който да се използва за изграждане на първоначални приоритети, оценки, разпределение на екип и време. 2. Основният сценарий за успех на всеки случай на употреба предоставя на всички участници споразумение относно това какво ще прави системата и какво няма да прави. Той предоставя контекста за всяко конкретно изискване за договорена позиция (напр. фино зърнести потребителски истории), контекст, който е много трудно да се получи никъде другаде.

3. Условията за разширяване на всеки случай на употреба осигуряват рамка за изследване на всички дребни, нищожни неща, които по някакъв начин заемат 80% от времето и бюджета за разработка. Той предоставя механизъм за гледане

напред, така че заинтересованите страни да могат да забележат проблеми, за които вероятно ще отнеме много време, за да получат отговори. След това тези проблеми могат и трябва да бъдат поставени пред графика, така че отговорите да могат да бъдат готови, когато екипът за разработка започне да работи по тях.

4. Фрагментите на сценария за разширение на случаите на използване предоставят отговори на многото подробни, често трудни и игнорирани бизнес въпроси: „Какво трябва да правим в този случай? Това е рамка за мислене/документация, която съответства на изявлението if...then... else, което помага на програмистите да обмислят проблемите. Само дето се прави по време на разследване, а не по време на програмиране.
5. Пълният набор от случаи на използване показва, че изследователите са обмислили нуждите на всеки потребител, всяка цел, която имат по отношение на системата, и всеки включен бизнес вариант.

В обобщение, уточняването на системни изисквания в случаи на използване има тези очевидни предимства в сравнение с традиционните или други подходи.

3.2 Ограничения

Ограниченията на случаите на употреба включват:

· Случаите на употреба не са подходящи за улавяне на изисквания на системата, които не се основават на взаимодействие (като алгоритъм или математически изисквания) или нефункционални изисквания (като платформа, производителност, време или аспекти, критични за безопасността). Те са по-добре посочени декларативно другаде.

· Тъй като няма напълно стандартни дефиниции на случаи на използване, всеки проект трябва да формира своя собствена интерпретация.

· Някои връзки от случаи на използване, като *extends* , са двусмислени при тълкуване и могат да бъдат трудни за разбиране от заинтересованите страни, както е посочено от Cockburn.

· Разработчиците на случаи на употреба често срещат трудности да определят нивото на зависимост на потребителския интерфейс (UI), което да се включат в случай на употреба. Докато теорията на случаите на използване предполага, че потребителският интерфейс не се отразява в случаите на употреба, може да е неудобно да се абстрахира този аспект на дизайна, тъй като прави случаите на употреба трудни за визуализиране. В софтуерното инженерство тази трудност се

решава чрез прилагане на проследимост на изискванията , например с матрица за проследимост . Друг подход за свързване на UI елементи със случаи на употреба е да прикачите дизайн на потребителски интерфейс към всяка стъпка в случая на употреба. Това се нарича сценарий на сценарий на употреба.

· Случаите на употреба могат да бъдат прекалено подчертани. Бертран Майер обсъжда въпроси като управление на системния дизайн твърде буквално от случаи на употреба и използване на случаи на използване до изключване на други потенциално ценни техники за анализ на изискванията.

· Случаите на употреба са отправна точка за проектиране на тестове, но тъй като всеки тест се нуждае от собствени критерии за успех, може да се наложи да се модифицират случаите на използване, за да се осигурят отделни пост-условия за всеки път.

· Въпреки че случаите на употреба включват цели и контексти, независимо дали тези цели и мотивации зад целите (притеснения на заинтересованите страни и техните оценки, включително липса на взаимодействие) са в конфликт или влияят негативно/положително на други системни цели, са обект на техники за моделиране на изискванията, ориентирани към цели (като BMM , I * , KAOS и ArchiMate ARMOUR).

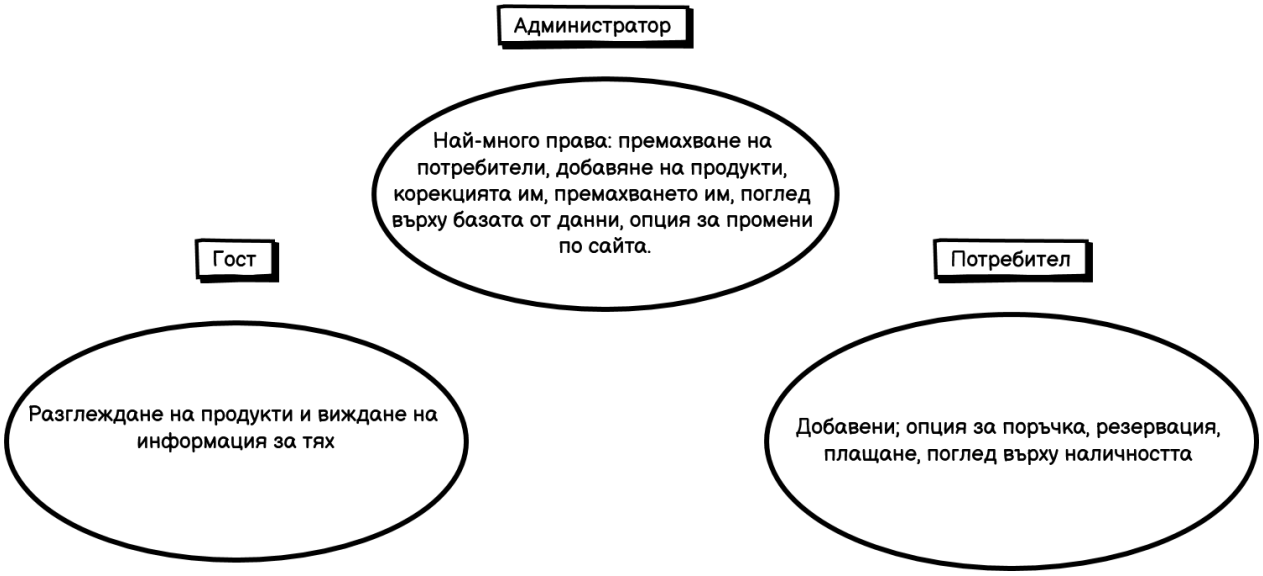
4.Истории (user stories)

• Обикновен потребител:
- Има достъп до обикновените функции на приложението и основните неща, които то предлага. В нашия случай-закупуване на рибарски принадлежности, тяхното резервирането им, информация за наличност и тн.

• Гости на приложението:

-Отново могат да виждат различните предмети предлагани в сайта, но нямат достъп до повечето от функциите, които сайтът предлага. (поръчка, разплащания) •
Администратор:

- Това е потребителят с най-много правомощия в приложението. Той ще по-широк набор функции, от останалите потребители. (промяна по информацията за продуктите, тяхното добавяне, премахване, преглед на потребителите и тн.)



4.1 Предимства

Кратки

Разбираеми за потребителите и разработчиците
Не е нужна поддръжка

Не налагат особени усилия
Недостатъци
Може да са непълни

Отворени за интерпретация.

Няма добри доказателства, че използването на потребителски истории повишава успеха на софтуера или производителността на разработчиците. Въпреки това, потребителските истории улесняват разбирането без ненужно структуриране на проблеми, което е свързано с успеха.

4.2 Използване

Централна част от много гъвкави методологии за разработка, като например в играта за планиране на XP , потребителските истории описват какво може да бъде вградено в софтуерния проект. Потребителските истории се приоритизират от клиента (или собственика на продукта в Scrum), за да се посочи кои са най-важни за системата и ще бъдат разбити на задачи и оценени от разработчиците. Един от начините за оценка е чрез скала на Фибоначи .

Когато потребителските истории предстои да бъдат внедрени, разработчиците трябва да

имат възможност да говорят с клиента за това. Кратките истории може да са трудни за тълкуване, може да изискват някои основни познания или изискванията може да са се променили след написването на историята.

Потребителските истории могат да бъдат разширени, за да добавят подробности въз основа на тези разговори. Това може да включва бележки, прикачени файлове и критерии за приемане.

4.3 Ограничения

Ограниченията на потребителските истории включват:

· Проблем с увеличаване на мащаба : Потребителските истории, написани на малки физически карти, са трудни за поддържане, трудни за мащабиране до големи проекти и обезпокоителни за географски разпределените екипи.

· Неясни, неформални и непълни : Картите с истории на потребителите се считат за начало на разговор. Тъй като са неформални, те са отворени за много интерпретации. Тъй като са кратки, те не посочват всички подробности, необходими за внедряване на функция. Следователно историите са неподходящи за постигане на официални споразумения или писане на юридически договори.

· Липса на нефункционални изисквания : Потребителските истории рядко включват подробности за производителността или нефункционалните изисквания, така че нефункционалните тестове (напр. време за реакция) могат да бъдат пренебрегнати.

· Не представяйте непременно как технологията трябва да бъде изградена: тъй като потребителските истории често се пишат от гледна точка на бизнеса, след като техническият екип започне да внедрява, може да открие, че техническите ограничения изискват усилия, които могат да бъдат по-широки от обхвата на отделна история . Понякога разделянето на истории на по-малки може да помогне за разрешаването на това. Друг път историите „само технически“ са най

подходящи. Тези „само технически“ истории могат да бъдат оспорени от заинтересованите страни в бизнеса, тъй като не предоставят стойност, която може да бъде демонстрирана на клиентите/заинтересованите страни.

5.Софтуерни изисквания

5.0 Изисквания

• Компютър с подходящ софтуер за работа с него.
• Visual Studio 2019 (минимално)
• Инсталиран MVC (controller-view-model)
• Всички библиотеки необходими за стартирането на проекта (ASP.NET Core Identity)

5.1 Цели

• Да се представи концепцията на Софтуерното инженерство

• Да се представят концепциите потребителски и системни изисквания

• Да се опишат функционалните и нефункционалните изисквания

• Да се обясни как софтуерните изисквания могат да бъдат описани в документ

5.2 Инженеринг на изискванията

• Процес на установяване на услугите, които даден клиент изисква от системата, както и ограниченията на работа и разработване.

• Сами по себе си изискванията са описание на системните функции и ограниченията, които се установяват по време на инженеринга на изискванията

• Неформално... Дисциплина от софтуерното/ системното инженерство, която обхваща дейностите по специфициране на продукта/системата.

• ИИ е изключително важна част от процеса на системно разработване, която направлява всички останали дейности към постигане на резултатите за които системата е предназначена

• ИИ оказва влияние на целия жизнен цикъл на дадена система

5.3 Какво е изискване?

• Изискванията могат да варират от много абстрактни описания на дадена системна функционалност или ограничения до много точни и детайлни математически функционални спецификации

• Изискванията могат да имат двойна роля: – Могат да послужат като заявка за търг – следователно трябва да са отворени за интерпретация – Могат да бъдат и база за договор – следователно тряба да бъдат детайлно дефинирани. – И в двата случая имаме изисквания.

5.3 Видове изисквания

• Потребителски изисквания – Указания на естествен език и диаграми на услугите, които системата ще предоставя, както и съответните оперативни ограничения. Написани за клиентите.

• Системни изисквания – Структуриран документ с детайлно описание на системните функции, услуги и оперативни ограничения. Определя какво трябва да се имплементира и може да бъде част от договора между клиента и разработчика

5.4 За кого са предназначени изискванията

Потребителски изисквания Клиенти Системни потребители Технолозите при клиента, Специалисти по договори Системни архитекти

Системни изисквания Системни потребители, Технолозите при клиента, Системни архитекти, Разработчиците

5.5 Функционални и нефункционални изисквания

• Функционални изисквания – Описание на услугите, които системата трябва да предоставя, начинът по който системата трябва да реагира на конкретни входни данни и поведението и в конкретни ситуации.

• Нефункционални изисквания – Ограничения върху услугите или функционалността на системата като времеви ограничения, ограничения върху процеса на разработване, използваните стандарти и др.

5.6 Функционални изисквания

• Описват функции или услуги на системата.

• Зависят от типа на софтуера, потенциалните потребители и типа на системата, в която ще бъде използван софтуера.

• Функционалните потребителски изисквания могат да бъдат описания на високо ниво на това какво трябва да прави системата, но функционални системните изисквания трябва да описват системната функционалност в детайли.

5.7 Нефункционални изисквания

• Определят системни характеристики и ограничения като надеждност, време за отговор и др.

• Изисквания към процеса могат също да бъдат специфицирани – например конкретна CASE система, програмен език или метод на разработване.

• Нефункционалните изисквания могат да бъдат по-критични от функционалните и от тях може да зависи пригодността на системата.

5.8 Неточности при изискванията

• Поради неточното описание на изискванията могат да възникнат проблеми.

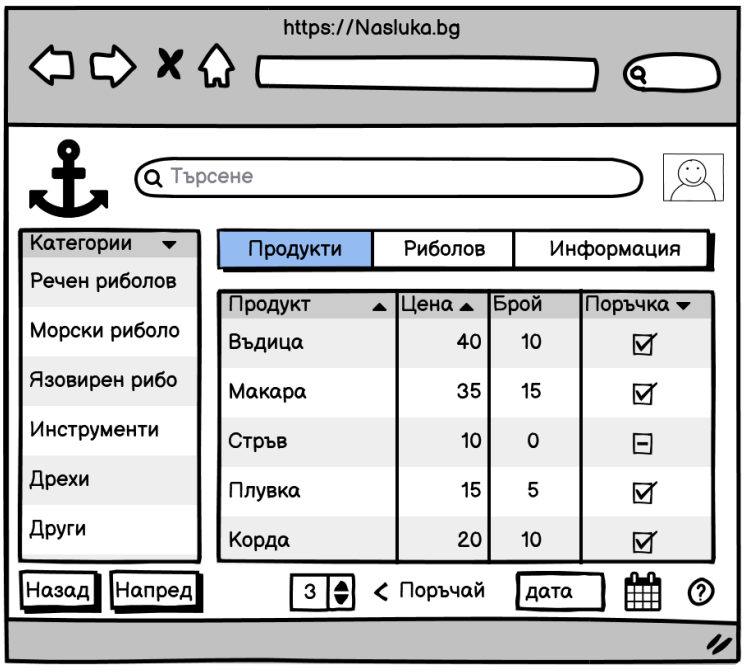
• Двусмислените изисквания могат да бъдат интерпретирани по различен начин от разработчиците и потребителите.

• Ако разгледаме изискването за ‘ подходящата среда ’ – Според потребителя – различна среда за всеки различен тип документ. – Според разработчиците – да се предостави текстова среда, която да покаже съдържанието на документа.

6.Прототипи на потребителския интерфейс

Позволява да се тества дали продуктът може да се използва гладко Позволява на дизайнерите да покажат продукта си,което го прави по-лесен за разбиране Прототипи могат да се създават по време на всеки етап от процеса по дизайн, за да помогнат да се демонстрират идеи, които трудно биха се изразили само с думи.

6.0 Очакван потребителски интерфейс



6.1 Прототипи с ниска точност

Хартия.

Най-бърз.

6.2 Прототипи със средна точност

Прототипи, позволяващи кликове.

Не са прекалено детайлни.

Позволяват на потребителя да почувства как ще изглежда финалният дизайн.

6.3 Прототипи с висока точност

Сложни интеракции и промени.

Включва по-сложни анимации.

6.4 Предимства на прототипите

Скорост: дизайнерите бързо могат да тестват няколко различни идеи преди да дадат на екип да ги разработи

Позволяват преглеждане и оценяване на дизайнерските решения
Обратна връзка

Рано откриване на проблеми в дизайна

Лесно могат да се сравнят няколко различни дизайна

6.5 Инструменти

Sketch

Adobe XD

Axure XP
Balsamiq

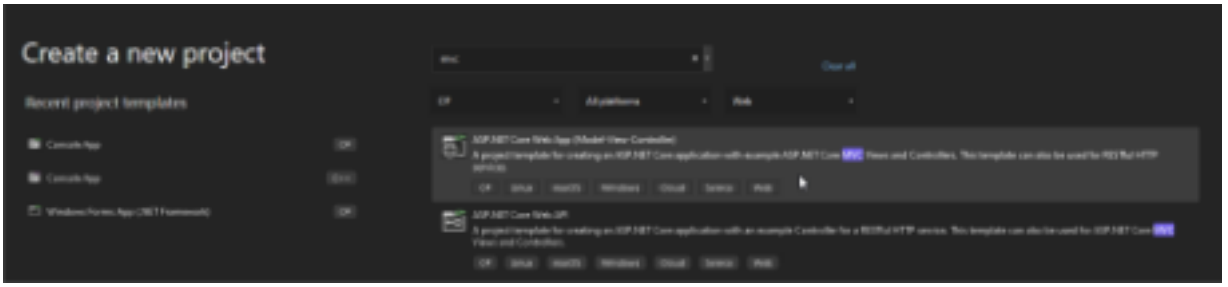
Composite
Fluid UI

Framer X

Реализация

7.1 Стъпки във Visual Studio Community 2022 за създаване на уеб приложение по MVC модел

-Създаване на нов проект



-Какво е Authentication и Authorization.

Authentication - удостоверяването е процес на проверка на идентификационните данни на потребител или устройство, което се опитва да получи достъп до системата. Такива данни са например: потребителско име и парола.

Authorization - упълномощаването е процес на проверка дали на потребителя или устройството е разрешено да изпълнява определени задачи в дадена система. Упълномощаването определя, какво е достъпно в системата за конкретен потребител.

-Типове Authentications

Автентикацията на електронен документ има за цел той да бъде защитен от възможни злоумишлени действия, като например:

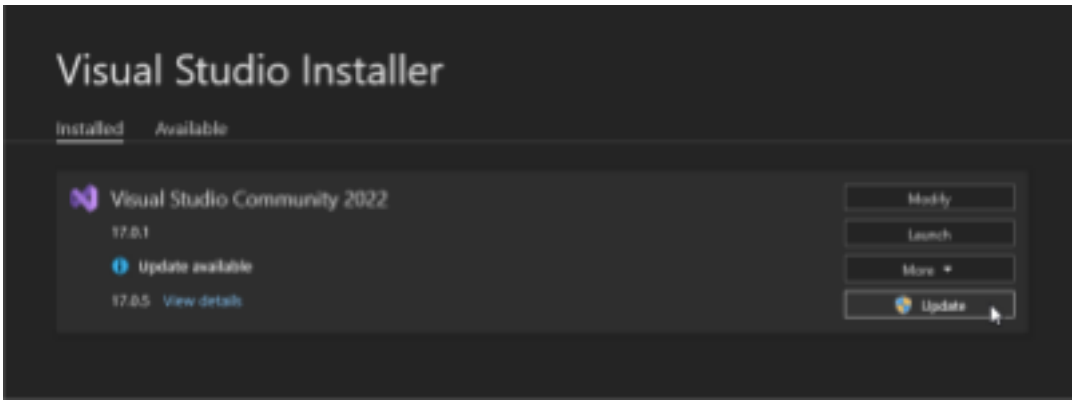
• активно прихващане – нарушителят се включва в компютърната мрежа и прихваща документа (файла);
• маскарад – абонатът X изпраща документ на абоната Б от името на абоната А; • ренегатство – абонатът А заявява, че не е изпращал съобщения на абоната Б, макар всъщност да е изпратил;

• подмяна – абонатът Б изменя или формира нов документ и заявява, че го е получил от абоната А;

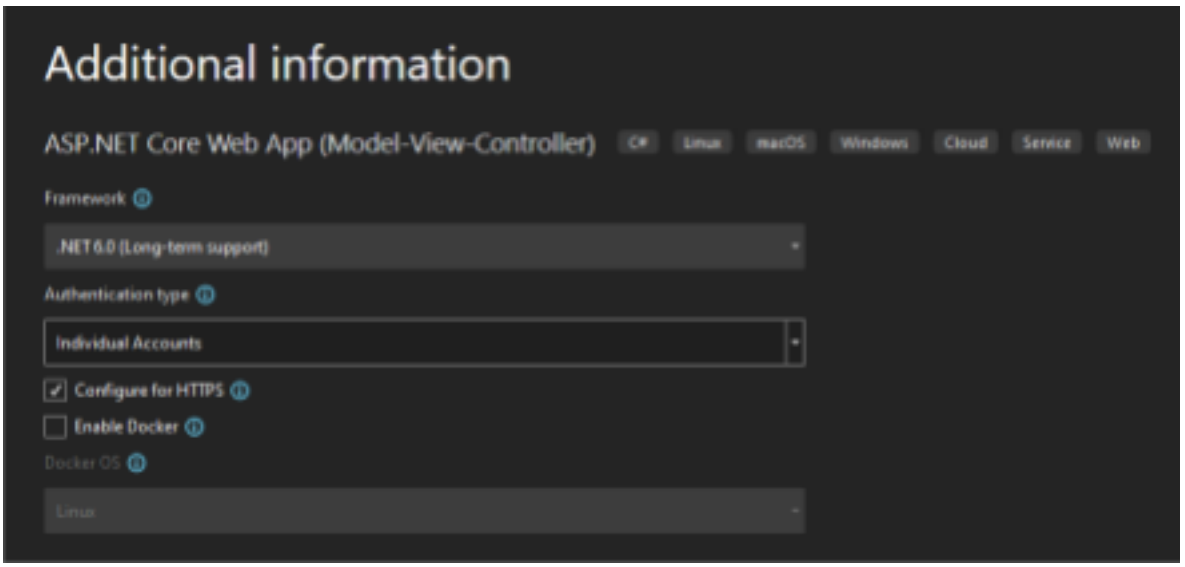
• повторение – абонатът X повтаря документ, който абонатът А е изпратил до абоната Б.

-Docker platform

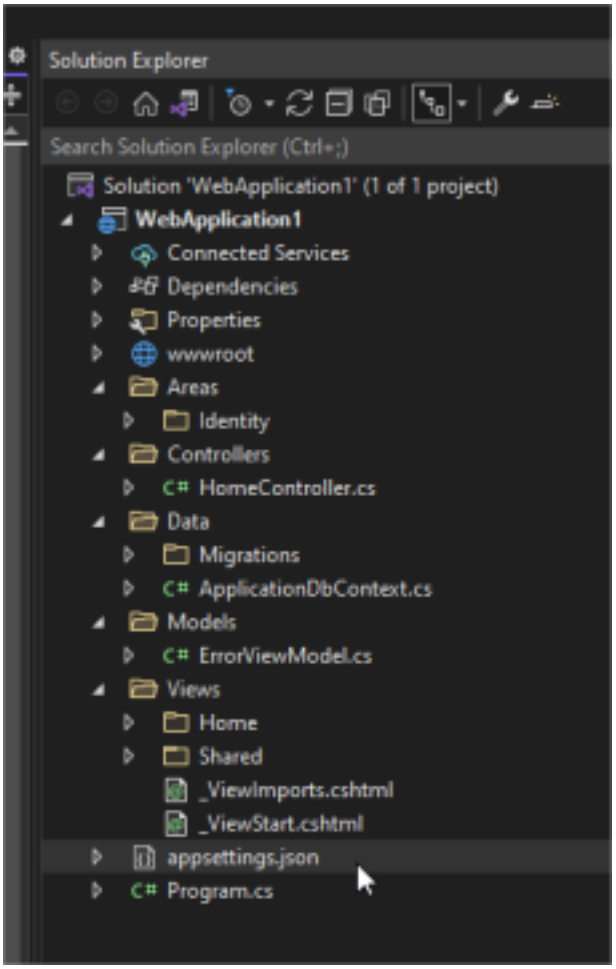
Docker е набор от PaaS продукти, използващи виртуализация на ниво операционна система (контейнеризация) и предоставящи софтуерни пакети, наречени контейнери. Контейнерите са изолирани един от друг и съдържат определен софтуер, библиотеки и конфигурационни файлове. Те могат да комуникират помежду си по строго дефинирани канали. Тъй като всички контейнери споделят сервизите на едно единствено OS ядро, те използват по-малко ресурси в сравнение с виртуалните машини.



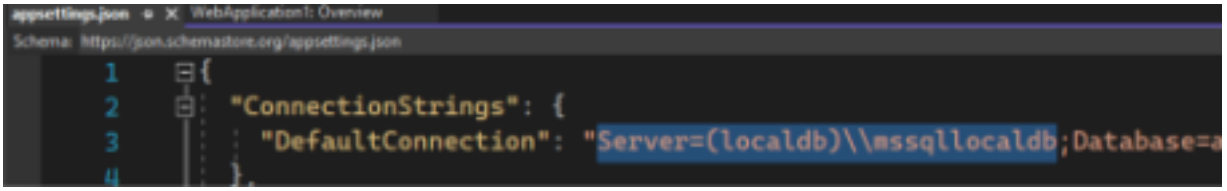
-Изберете следните опции за проект и генерирайте проекта:



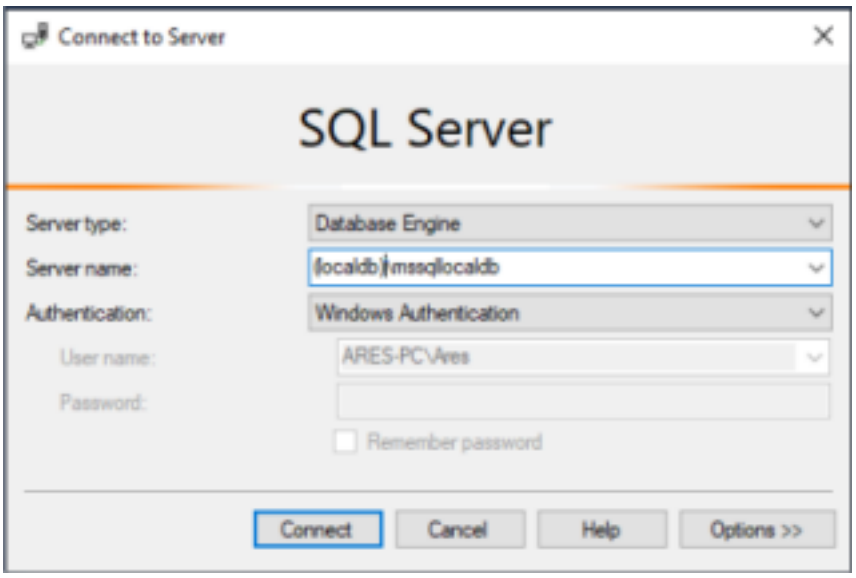
Отворете .json файла:



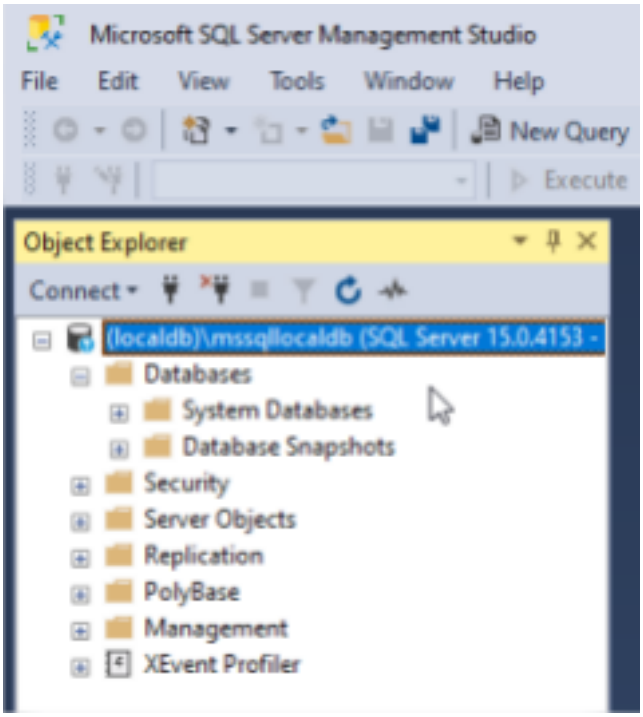
-Вижте тази информация



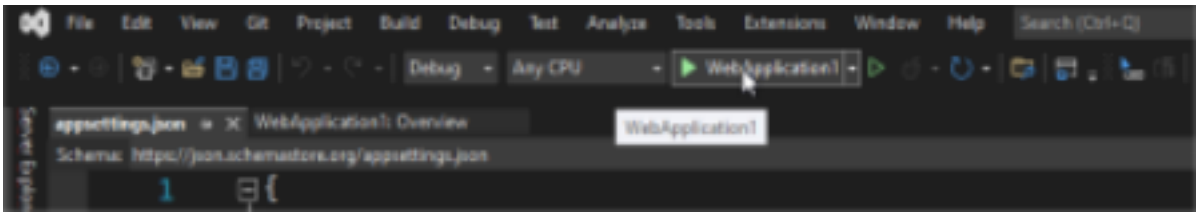
-Свържете се със сървъра като напишете информацията, която сте взели от приложението.



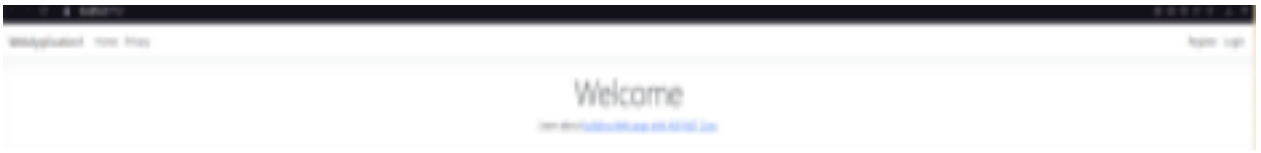
-Ще видите такъв прозорец:



-Стартирайте уеб приложението



-Ще видите следния екран в браузъра:



Обобщение

Софтуерни изисквания

Случаи на употреба (use cases)

Истории (user stories)

Спецификация на изискванията

Прототипи на потребителския интерфейс (UI prototyping)

Източници:

https://learn.fmi.uni-sofia.bg/pluginfile.php/116877/mod_resource/content/1/L4-1-RE-BCs-2014-15.pd

f https://en.wikipedia.org/wiki/User_story

https://en.wikipedia.org/wiki/Use_case#Examples

https://classroom.google.com/u/0/c/MzkxMzkyMDA4NTI1/m/NDY5MTAxNjQxMDU3/details