 **ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА**

**КАТЕДРА „СОФТУЕРНИ И ИНТЕРНЕТ ТЕХНОЛОГИИ”**

**Курсов проект**

Дисциплина:

Управление на Софтуерни Проекти

Проверил: Изработили: Станислав Стойчев - фак.№176217

Ренета Първанова Радостин Иванов - фак.№17621764

Илиян Стоянов - фак.№17621763

Специалност: СИТ

Курс: III

Група:6а

***Съдържание***

1. **Бюджет…………………………………………………………………………………………..**3стр.
2. **Предмет на техническото задание………………………………………….…….**3стр-4стр.
3. **Изисквания на възложителя…………………………………………………..**3стр-4стр.
4. **Обхват на разработка…………………………………………………………………….**4стр.
5. **Изисквания към програмния продукт…………………………………..**4стр-5стр.

* **Общи;**
* **Функционални;**

1. **Възможност за разработване на допълнителни модули……………**5стр.
2. **Системa за контрол на версиите…………………………………………….**5стр-7стр.
3. **Система за управлението на проекта……………………………………………**7стр.
4. **Методология на разработване……………………………………………….**7стр-8стр.
5. **Екипи, роли и задачи……………………………………………………………**8стр-15стр.
6. **Извод…………………………………………………………………………………………**15стр.

**Техническо задание на проекта**

**1.Бюджет**

При изчисляването на бюджета се взимат разходите, които трябва да се покрият по време на етапите на разработване на програмния продукт(ПП) – при първият етап вниамтелно се планират ресурсите, които са необходими, програмните средства, комуникациите, разпределят се задачите и ролите.

След завършването,тестването и експлоатацията на проекта, крайната сумата се свежда до 7330 лв. Като към нея се добавят 5% от общата сума за следващите процеси свързани с поддръжката на софтуера. Възможно е закупуването под формата на лизинг без оскъпяване.

**2.Предмет на техническото задание**

Предмет на този документ са изискванията на възложителя относно желаната функционалност на системата. Разработваната система предлага регистриране на клиент, избор на автомобил за наемане , филтриране по определени категории според желанието на възложителя и тяхното извличане и визуализиция в удобен потребителски интерфейс. В техническото задание се описват пълните и точни първоначални изисвания към проекта, неговите характеристики, както и изпълняваните от него функции и крайната цел и идея на програмния продукт.

**Изисквания на възложителя**

Системата трябва да предоставя следните функционалности:

Начален екран

Трябва да съдържа две възможности за избор - регистриране на клиент и вход в системата на вече регистриран клиент.Трябва да има подходящо избран дизайн. Да съдържа лого и призив предоставени от възложителя.

Прозорец за регистрация

Трябва да съдържа 5 полета(Име, Адрес, Телефон за връзка, Имейл, Налични пари).Всяко едно от полетата да притежава валидация и при въвеждане на грешни данни да се извежда прозорец за грешка със съобщение, което спомага за правилно въвеждане на данните. Бутон с който при натискане въведените данни да се записват в база от данни, където да бъдат съхранявани. В горния ляв ъгъл да се намира бутон за връщане в началния прозорец.

Прозорец за вход

Да съдържа две полета, с които клиентът да може да се логне в системата - имейл адрес и телефон за връзка. Бутон с който при натискане клиентът да се проверява дали потребителя(клиента) съществува и ако да той да бъде препратен към прозореца с избор на автомобили.Ако не да се изведе съобщение за грешка и повторен опит за влизане в системата.

Прозорец за изобразяване на марки коли

При логване в системата да се визулизира прозорец с марките автмобили, от където клиетна(потребителя) може да избира избере марка, след което и марка. В лявата част е разположено логото.

Прозорец за изобразяване на модели

Трябва да съдържа панел, който визуализира всички модели от съответната марка. Визуализирането да представлява снимка на автомобила придружено с неговите име и спецификации. Да има предоставена цена за наемане и бутон, с който клиента може да осъществи наемането. Ако автомобила е нает то неговата цена да бъде скрита и да не може да бъде нает.

Прозорец за търсене на автомобил.

Да позволява търсене на автомобил при определени критерии. Да съдържа три полета за въвеждане – Марка на автомобил, Избиране на набор от екрстри и цена на автомобила. При натискане на бутона Търсене да се визуализира резултата от търсенето в друг прозорец. При непопълване на някои от полетата да се изведе съобщение за грешка.

**3.Обхват на разработката**

Програмния софтуерен продукт трябва удовлетворява очакванията както на потребителите, така и на възложителя, с което да им предоставя възможност за улеснено използване на заложените функционалности. Потребителите ще разполагат със софтуер, предоставящ им възможност за богат избор от автомобили за наемане, както и за търсене. За в бъдеще ще им бъде предоставено, те самите да предлагат свои автомобили за наемане от други потребители и ще имат възможност да к0муникират с чат по между си. Ще бъде създадена виртуален автосалон, където всички ще могат да разгледат автомобилите от вътре и от вън, което ще им предостави по голяма представа за автомобила, който възнамеряват да наемат. На всеки един потребител ще се изпраща всяка седмица имейл, с коейто ще ги уведомява за нови обяви или събития.

**4.Изисквания на програмния продукт за наемане на автомобил**

**Общи изисквания**

Всички случаи на използване и правилното поведение на системата трябва да са прецизно описани.Продуктът е релизиран като Десктоп приложение, което предоставя възможността да бъде независимо(тоест зависи само от конфигурацията и операционната система на клиента) , без необходимостта от наличието на браузър. Софтуерът е предназначен за устройства работещи с Windows операционна система и съобразени с минималните хардуерни изисквания.

За управление и съхранение на данните е използван Microsoft SQL Server, с който се управляват и извличат големи сървърно базирани БД. С него може да се работи или на същия компютър или на друг компютър в мрежа за ралика от MS Access, която е desktop базирана и не е предназначена за управление на големи корпоративни БД.

Машината трябва да притежава следните минимални изисквания за правилна работа на софтуера:

**Хардуер:**

**●** Процесор: Intel Core i3-3217UE 3M Cache,1.60 GHz или по бърз процесор.

**●** RAM: 1GB(гигабайт) за 32-битова ОС или 2GB(гигабайт) за 64-битова ОС.

**●** Място на твърдия диск: 500 MB за 32-битова ОС или 1GB за 64-битова ОС.

**●** Графична карта: DirectX 9 или по-нова версия с драйвер WDDM 1.0.

**Софтуер:**

**●** Операционна система : Windows 10,Sql Server Management Studio, Sql Server.

**Функционални изисквания – системата за търсене и наемане на филм автомобил да предоставя следните възможности:**

Модул за регистрация на потребител(клиент):

Възможност за въвеждане на Име.

Възможност за въвеждане на Адрес.

Възможност за въвеждане на Телефон.

Възможност за въвеждане на Имейл.

Възможност за въвеждане на Налични средства.

Модул за търсене на автомобил, според определени критерии:

Възможност за търсене по марка.

Възможност за търсене по екстри.

Възможност за търсене по цена.

Възможност за търсене по цвят.

Възможност за търсене по година.

Възможност за търсене по изминати км.

**5.Системна документация**

Изпълнителят трябва да подготви следната документация:

* „Описание на структурата и принципа на работа на системата”
* „Ръководство за конфигуриране, контрол и поддръжка на системата”

Всички документи е необходимо да бъдат представени на 3 езика: български език, английски език, немски език.

Всички документи трябва да бъдат предоставени в два екземпляра, на хартиен и електронен носител.

**6.Възможност за разработване на допълнителни модули(Софтуерна поддръжка).**

След като възложителя закупи програмния продукт, на него му се предоставя програмния код, с помощта на който може да се доразработят или добавят нови модули, както и да се премахнат създавайки нова версия на софтуера от страна на изпълнителя при желание на възложителя. За осъществяване на срещи поръчителят може да се свърже с предстваително лице на фирмата на посочения имейл – [auto\_rent@gmail.com](mailto:auto_rent@gmail.com) или на мобилен телефон: 08983455579 или 052/ 66 78 23.

**7.Системa за контрол на версиите**

Системата за контрол на версиите позволява управление на работата по даден софруерен продукт, за да се улесни разработката му и да намали неудобствата при съвместна работа на много хора върху един проект. Използва се едно централно хранилище, където се съхраняват файловете на проекта и където всеки един от разработчиците има достъп. Позволява да се запази история на промените, да се върне предишна версия или да се добави описание на промяната. Системата следи и възникване на конфликти – различни промени на различни потребители, ползващи хранилището.

**GIT**

При разбработването на програмния продукт използвахме децентрализирана система за контрол на версиите GIT, която изигра ключова роля при изграждането. Изключително лесен за използване безплатен софтуер, предоставящ ни локално хранилище с пълна история и възможност за следене на версиите. Това прави Git независим от мрежови връзки към централен сървър. Git има силно застъпена поддръжка на нелинейно разработване(бързо разклоняване и сливане), разпределено разработване, съвместимост със съществуващи протоколи, ефикасно справяне с големи проекти, криптографска проверка на историята.

**Настройки нужни за работа с Git.**

За да може да се възползва всеки от силата на Git в Windows е необходимо да се инсталира клиент, който е достъпен на **<https://git-scm.com/downloads>.**  Необходимо е да се инсталира среда в случaя използваме Visual Studio Code.

В процеса на инсталиране следва да конфигурираме Git, да бъде използваем в command prompt-a на Windows, това ще ни даде възможност да използваме командите в терминала. Лесна проверка за това дали сме го конфигурирали правилно е команда git. В следствие на използването й в командния ред се изписва инфомация за всички команди, които можем да използваме, както и описание на всяка от тях. Като дистрибутор на хранилище използваме GitHub. Това означава, че за целта е необходимо да се направи регистрация в **<https://github.com/> .** След като се регистрираме сайтът ни препраща на началната си страница. В лявата част на екрана е панел с всички хранилища, за които имаме оторизиран достъп. Ново хранилище можем да създадем от бутона в горния десен ъгъл на този панел или чрез бутона “Start a project”, намиращ се в средата на екрана. Във формата за създаване на ново хранилище имаме няколко задължителни полета:

1. име на хранилището, което следва да бъде уникално в системата на GitHub;
2. тип на хранилището – публично или частно (GitHub дава възможност за неограничен брой публични хранилища, както и неограничен брой частни хранилища, в случаите, когато броят на членовете в даден екип е неповече от 3);

В случая отбелязваме, че искаме инициализация на хранилището с README файл. Основния екран показва информация за броят commit-и , бранчове, релийзи и програмисти, които са правили промени по файловете в хранилището. (Contributors). След като сме се регистрирали и инициализирали хранилището е необходимо да си го клонираме локално на собствените машини (това е най-лесният начин за бързо конфигуриране на Git ).   
За целта кликаме на бутон **Clone or download.** Копираме си адреса на хранилището. Отваряме терминал в директорията, в която искаме да си създадем локалното хранилище (може да стане с клавишна комбинация SHIFT + десен бутон на мишката и избор на опция Open command line window here / Open PowerShell window here). В командния ред пишем git clone <URL> на нашето хранилище. Командата създава в нашата директория папка с името на хранилището, в която се съдържат всички файлове, които в момента се намират в хранилището, а именно README.me и скрита директория, в която се съхранява история с всички операции свързани с нашето хранилище. (в т.ч бранчове, commit-и и др). При първоначално стартиране на Git на вашата машина вероятно ще ви поиска и конфигуриране на клиент, с чиито credential-и (данни за идентификация) следва да се изпълняват командите свързани с хранилището. За да конфигурирате тези данни минималните данни, които следва да конфигурирате са email и потребителско име. В нашият случай това следва да са имейла и потребителското име от GitHub. За целта в командния ред следва да напишем командите:

**git config –global user.email “**[**email@example.com**](mailto:email@example.com)**”(***флагът global се слага, ако смятате да имате един, единствен акаунт от който да работите , в противен случай после ще се наложи да зачиствате тези данни от кеша).* Заместваме примерният мейл от изображението по-горе с нашият.

Втората команда е за потребителско име:

**git config –global user.name “username”**

Вече сме готови за създаване и запазване на промени в локалното хранилище, а в последствие и към децентрализираното предлагано от доставчика.

Отваряме папката с нашето локално хранилище в редактора (Visual Studio Code). Можем да си създаваме, модифицираме и трием файлове от наличните в Файловия explorer. Всичките ни промени се отбелязват с badge върху иконата на VCS. След като сме запазили промените локално е необходимо да ги отразим и в хранилището. Добра практика е преди да се направи Push ( команда за изпращане на историята с локални commit-и) да се изтеглят всички промени, които са качени от нашите колеги. Това се прави с команда Pull ( актуализира се състоянието на локалното хранилище, като се отразяват направените и качени промени). След изпълнението на Pull системата прави т.нар merge или сливане (той е автоматичен, ако няма конфликти, или ръчен при наличие на конфликтни зони – това са едни и същи файлове, които са били променяни от повече от един commit-а).   
Нашите промени са вече отразени в хранилището.

Когато се прави нещо в Git, тези действия само *добавят* информация към базата данни на Git. Трудно е системата да направи каквото и да е без то да може да бъде възстановено или пък да се изтрият данни безвъзвратно. Разбира се, подобно на всяка друга VCS, могат да бъдат загубени или объркани промените, които не са били къмитнати, но веднъж направили snapshot-а в Git - е много трудно да се загубят данни, особено пък ако редовно се изпраща базата към отдалечено хранилище.

**Съпоставка на Git с други системи за контрол на версиите.**

Ако трябва да направим съпоставка между Git, Subversion и Team Foundation Server(TFS), то Git излиза напред с няколко крачки пред останалите. Като започнем от скоростта на създаването на бранчове, отбележим че може да работи с доста слаба интернет връзка при наличието на цялото repository(което си е плюс до някаква степен) и стигнем до най-важното предимство, с което Git изпреварва конкурентите си, а именно че при евентуален срив(повреда на локалния компютър) всички данни могат да бъдат изгубени, но тази система тъй като е децентрализирана , проекта се сваля и се копира локално на всеки един участващ потребител. По този начин се създава backup и в случаи на необходимост, проектът може да се възстанови от локалното копие на някои от потребителите. По този начин кодът проекта е защитен.

**Недостатъци**

* Един от недостатъците на Git е че няма да ви предупреди ако направите къмит в master.
* Състои се от много команди и опции, някои от които са непоследователни до някаква степен, което ги прави по сложни за разбиране.
* Минусите произлизат от допълнителните стъпки, които разработчиците трябва да изпълнят, като например това, че трябва да се убедят, че branch-ват от правилния commit. Разчитайки на обединяването на завършените issues, history log-ът може да стане по-голям и по-труден за разбиране.

**8.Система за управление на проекта**

**Trello**

Trello е основно безплатно средство за сътрудничество, който организира проектите в дъски (boards), списъци (lists) и карти (cards). Само с един поглед Trello ни дава информация за това върху какво се работи, кой върху какъв процес работи и къде се намира нещо в даден процес. С прегледно си оформление, невероятно интуитивния си дизайн и широката си достъпност на неограничен брой членове до даден проект. Въпреки многото си функции, той поддържа изключително прост външен вид и навигация, които не са проблем дори и за начинаещи.

**Предимства:**

Със сигурност няма по-интуитивен софтуер за управление на проекти от Trello. Само бърз поглед на подреждането на картите позволява на потребителите да знаят докъде е стигнал проекта и какви са задачите им за следващия.На предната част на картата може да напишете основна информация – малко по-дълга от заглавието на конкретната задача. На „гърба“ обаче можете да попълните различни данни, отнасящи се до конкретния таск, като всичко това може да бъде организирано лесно и визуално.

**Недостатъци:**

Основната концепция на Trello е да поставя акцент на простотата и минимализма. Това води до липсата на някои основни възможности. Това може би не е най-подходящият софтуер за проследяване на проектите в по-дълбока детайлност. Например, нямаме възможност да видим списъка на задачите, подредени по потребител или краен срок.

**9.Методология на разработване:**

***Екстремно програмиране (Extreme Programming – XP)***

За разработката на програмния продукт нашият екип избра концепцията и методологията на разработване да бъде Екстремното програмиране. При работа с нея се стимулира вербалната комуникация, за разлика от другите концепции, при които комуникацията става чрез документация. XP позволява редуциране на цената на проекта, ако се наложи дадена промяна в бъдеще, тъй като останалите методологии биха били скъпоструващи за реализацията им. Затова се използва при проекти, където има често променящи се изисквания или непредвидими проблеми свързани с имплементацията и тогава по-стандартните методологии (като Waterfall модела) не са оптимални за постигане на голяма продуктивност.

Екстремното програмиране ни предоставя процес на планиране или казано **Planning Game**. Това е среща на екипа, която се прави веднъж на всяка итерация(обикновенно веднъж седмично). Състои се от две части:

**Release Planning**

Първата се фокусира върху изискванията на клиента(възложителя), където има договаряне каква точно функционалност ще се търси при следващия release и кога се очаква да излезе, съшо така могат да бъдат подобрени,добавени или премахнати изисквания от негова страна.

**Iteration Planning**

Втората се обсъждат вече дейностите и задачите на разработчиците, клиента не се намесва. Изискванията се разпределят в различни задачи, те се разпределят между разработчиците, оценява се времето необходимо за реализацията и накрая се представят завършените задачи, сравянвайки се с предварителните изисквания.

**Екипи, роли и задачи:**

**1.Екип:**

♦ РИ

♦ СС

♦ ИС

**2.Роли:**

* Мениджър на проекта – РИ,СС,ИС
* Софтуерен архитект – СС
* Разработчици на бизнес логика – РИ, СС,ИС
* Дизайнер на потребителския интерфейс – СС
* Технически писател – РИ
* Тестери –ИС

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер | Описание | Предшественик | Продължителност(дни) | Изпълнител |
| 1 | Инициране | - | 1 | ИС,РИ,СС |
| 2 | Планиране |  |  |  |
| 2.1 | Планиране на ресурсите | 1 | 1 | ИС |
| 2.2 | Планиране на програмните средства | 1 | 1 | РИ |
| 2.3 | Планиране на комуникациите | 1 | 1 | СС |
| 2.4 | Планиране на обхвата на проекта | 1 | 1 | РИ,ИС |
| 2.5 | Планиране на задачи и разпределяне на роли | 1 | 1 | СС |
| 3 | Анализ на изискванията на клиента | 2.1,2.2,2.3,2.4,2.5,2.6 | 2 | РИ,СС,ИС |
| 4 | Проектиране |  |  |  |
| 4.1 | Проектиране на базата данни(БД) | 3 | 1 | ИС |
| 4.2 | Проектиране на потребителския интерфейс | 4.1 | 2 | СС |
| 4.3 | Проектиране на бизнес логика | 4.2 | 1 | РИ |
| 5 | Изграждане на техническо задание на проекта | 4.3 | 1 | РИ |
| 6 | Изготвяне структурата на програмната система | 5 | 1 | СС |
| 7 | Разработване |  |  |  |
| 7.1 | Реализация на базата данни | 6 | 1 | ИС |
| 7.2 | Реализиране на SQL заявки;изграждане на тригери и рпоцедури | 7.1 | 2 |  |
| 7.3 | Изграждане на потребителски интерфейс | 7.2 | 2 |  |
| 7.4 | Реализиране на програмен код | 7.3 | 2 |  |
| 8 | Тестване |  |  |  |
| 8.1 | Разработка на тестове за БД | 7.4 | 1 |  |
| 8.2 | Разработка на тестване за потребителския интерфейс | 8.1 | 1 |  |
| 8.3 | Разработка на тестове за бизнес логиката | 8.1 | 1 |  |
| 8.4 | Разработка на тестове за програмния код | 8.1 | 1 |  |
| 8.5 | Изпълнение на тестовете | 8.2,8.3,8.4, | 2 |  |
| 9 | Създаване на потребителска документация | 8.5 | 2 |  |
| 10 | Представяне на програмния продукт | 9 | 1 |  |  |

**3.Задачи:**

**Определяне на срокове:**

Начална дата на проекта –27.03.2020г.

Планиране – 30.03.2020г.

Проектиране – 02.04.2020г.

Разработване – 10.04.2020г.

Тестване – 21.04.2020г.

Документация – 27.04.2020г.

Представяне на ПП – 29.04.2020г.

**Програмни средства за реализация:**

Език за програмиране – C#,SQL

IDE – Visual Studio

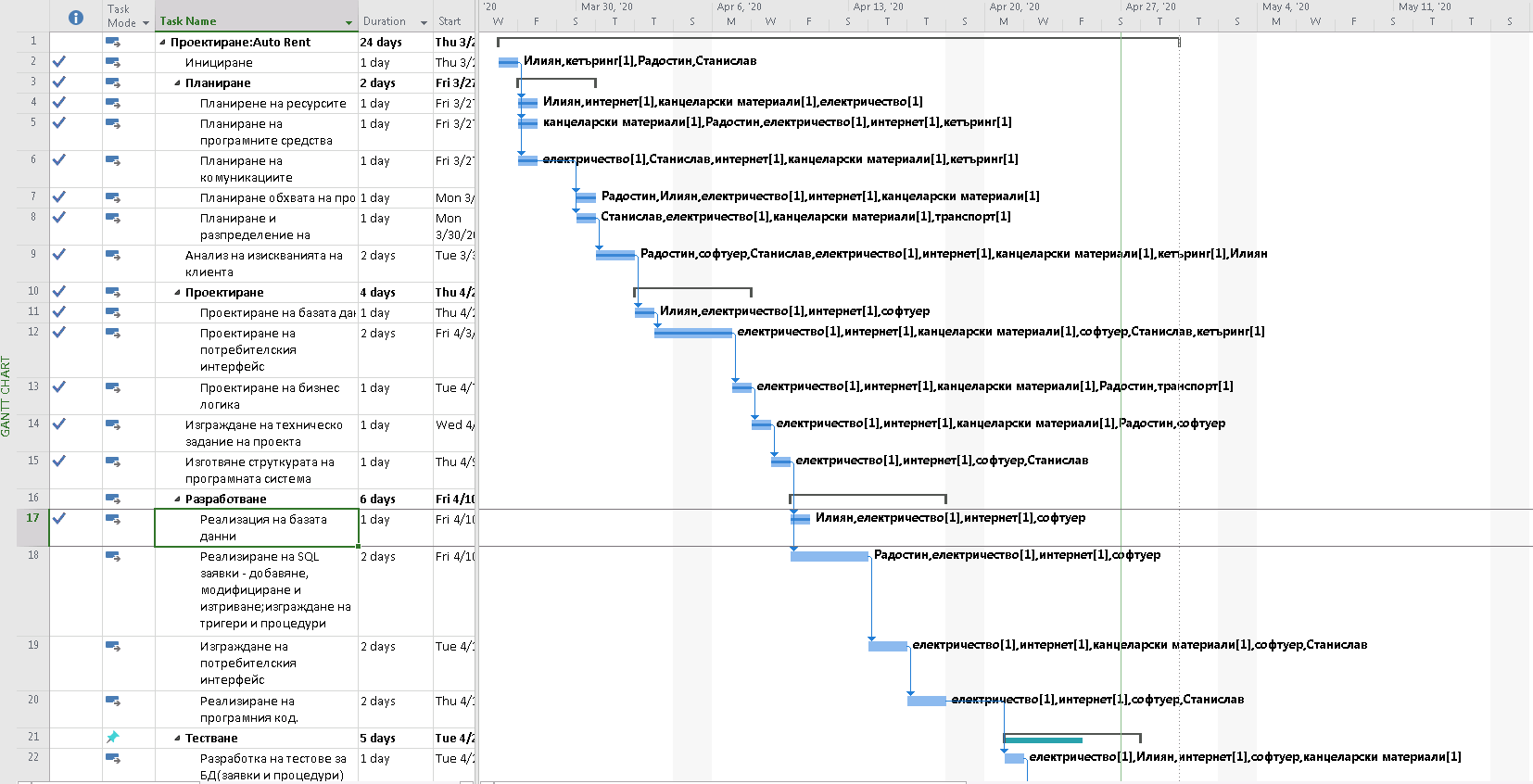
Бази данни - Sql Server Management Studio

**Жалонни точки:**

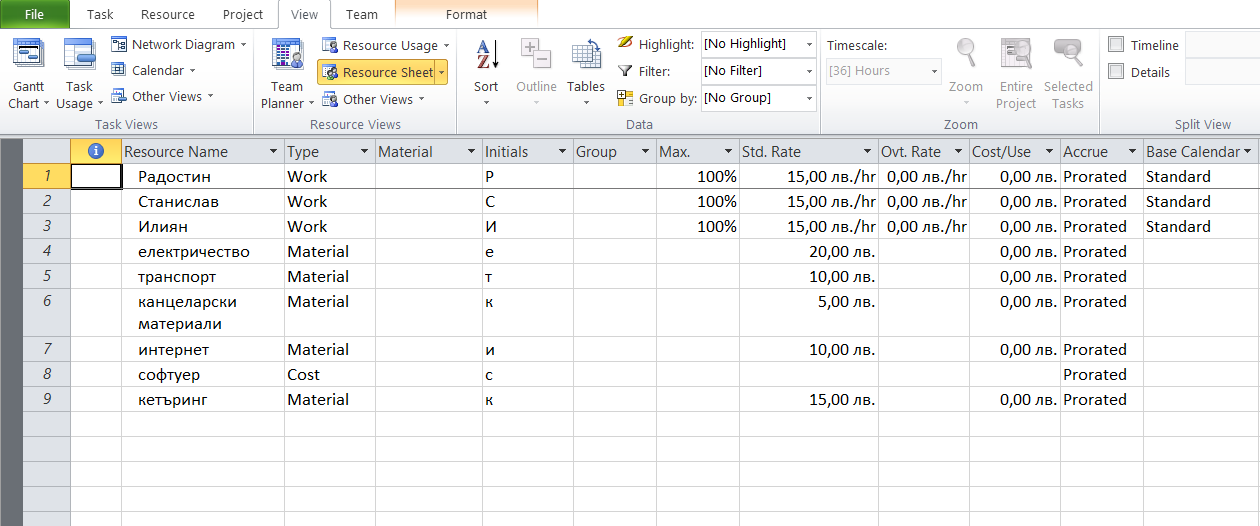
|  |  |
| --- | --- |
| 1. | Инициране на проекта. |
| 2. | Завършено планиране. |
| 3. | Изискванията са анализирани. |
| 4. | Проектиране на БД завършено. |
| 5. | Проектиране на потребителския интерфейс завършено. |
| 6. | Проектиране на бизнес логиката завършено. |
| 7. | Техническото задание е изготвено. |
| 8. | Структурата на програмната система е изготвена. |
| 9. | Базата данни е разработена. |
| 10. | Заявките,процедурите и тригерите създадени. |
| 11. | Потребителският интерфейс е реализиран. |
| 12. | Програмният код е реализиран. |
| 13. | Тестовете са изготвени. |
| 14 | Тестовете са изпълнени. |
| 15 | Потребителската документация е създадена. |
| 16 | Програмния продукт е представен на клиента. |

***Microsoft Project***

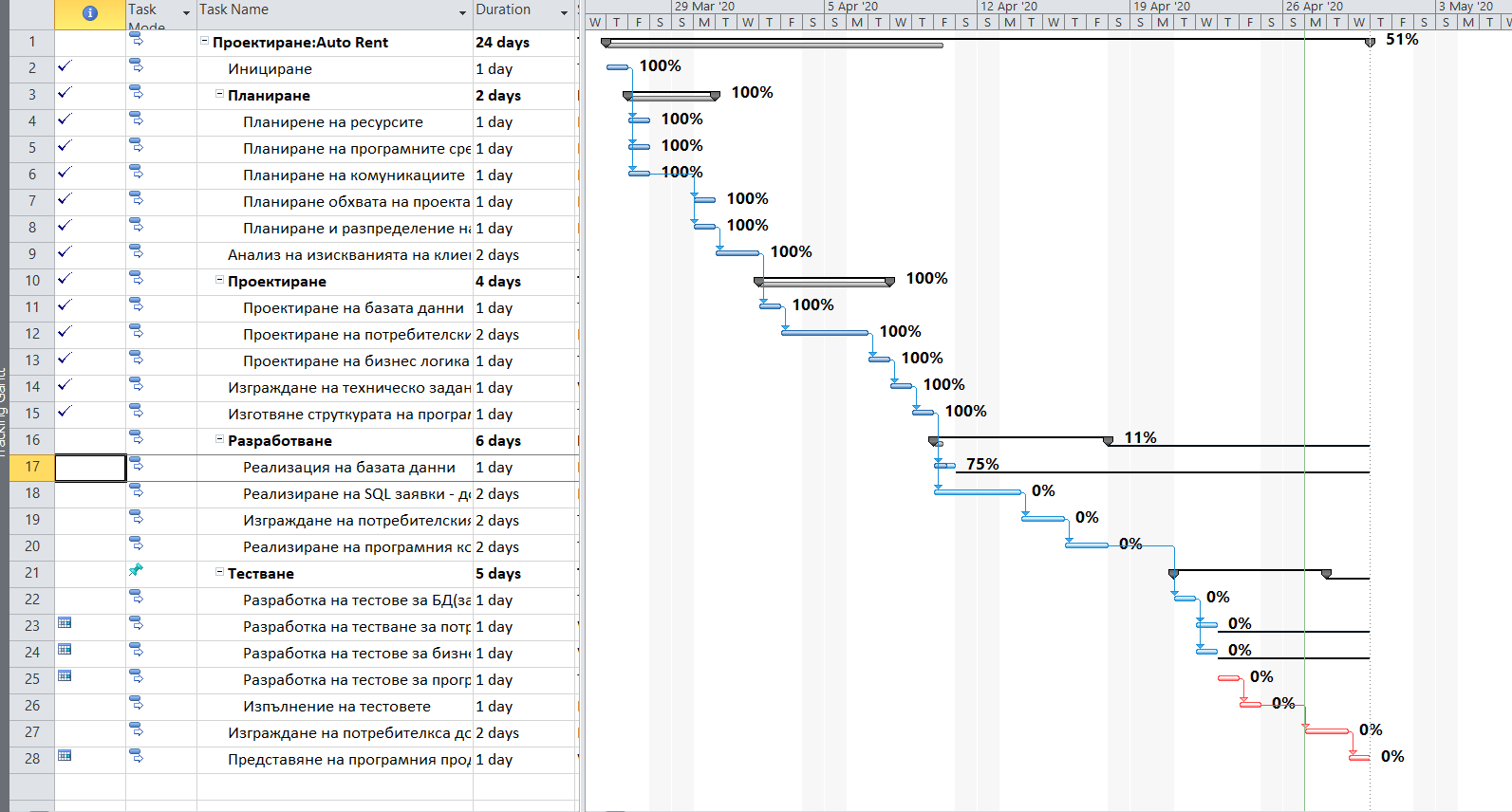
**Диаграма на Гант:**

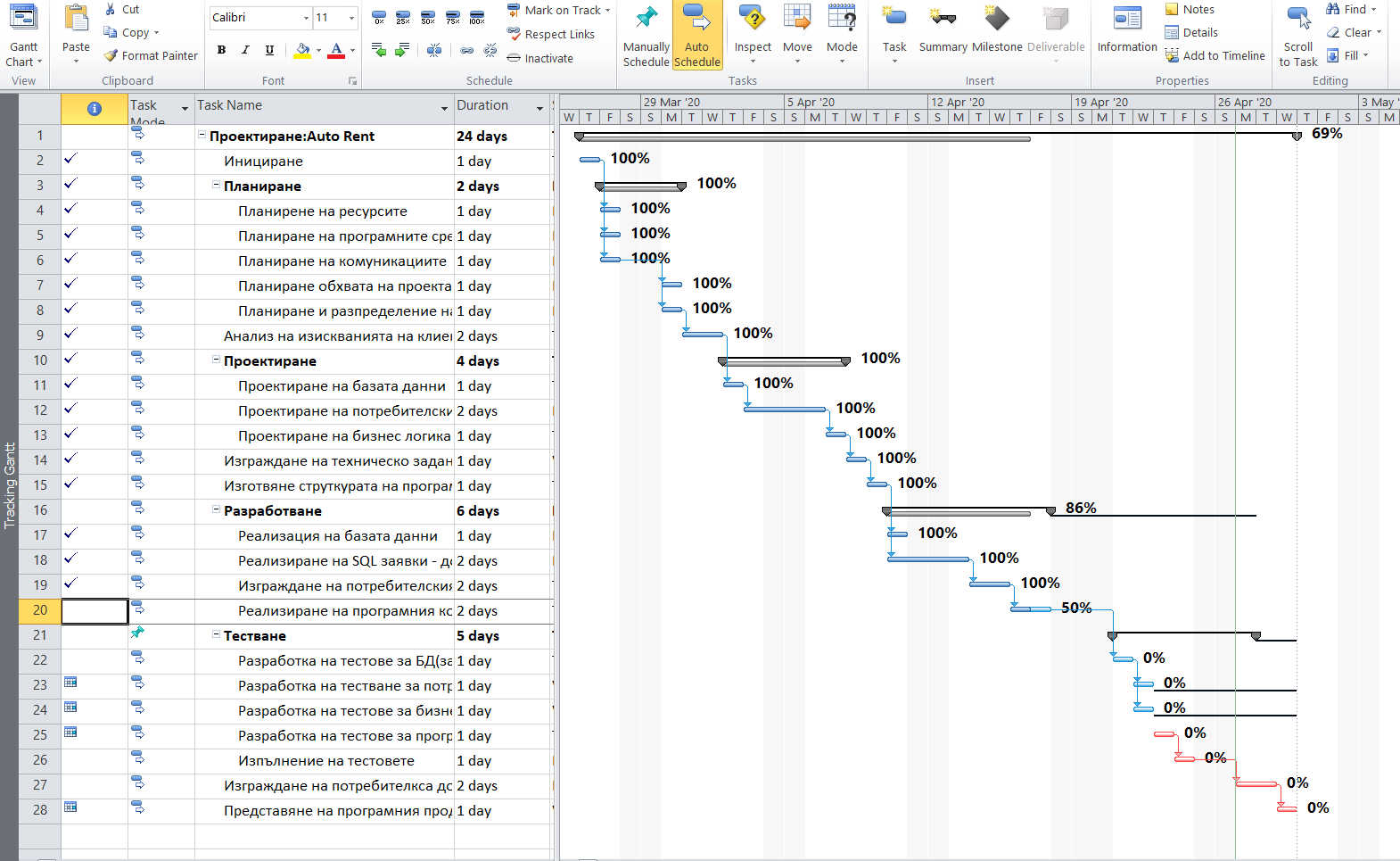
****

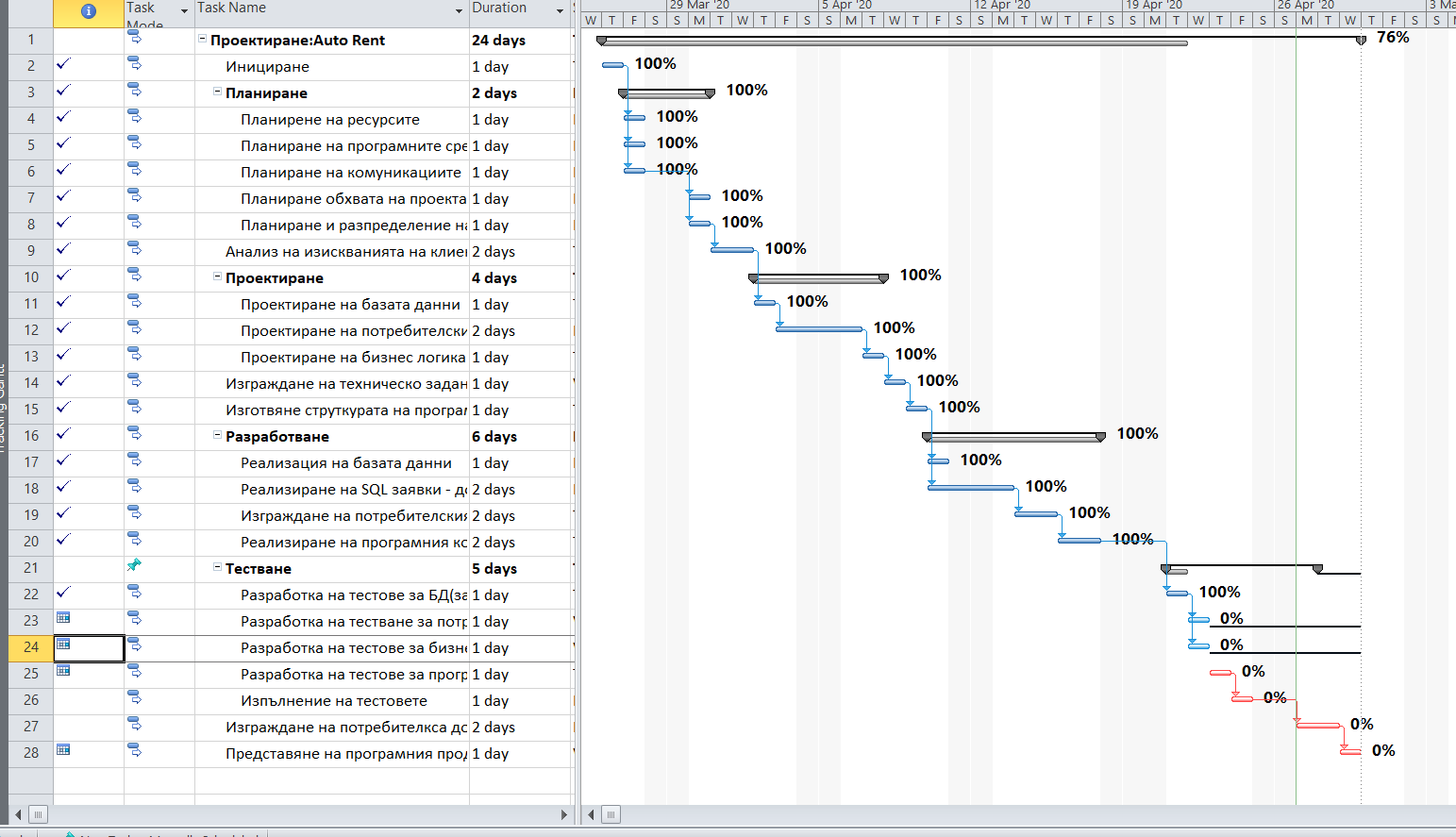
**Ресурси:**

****

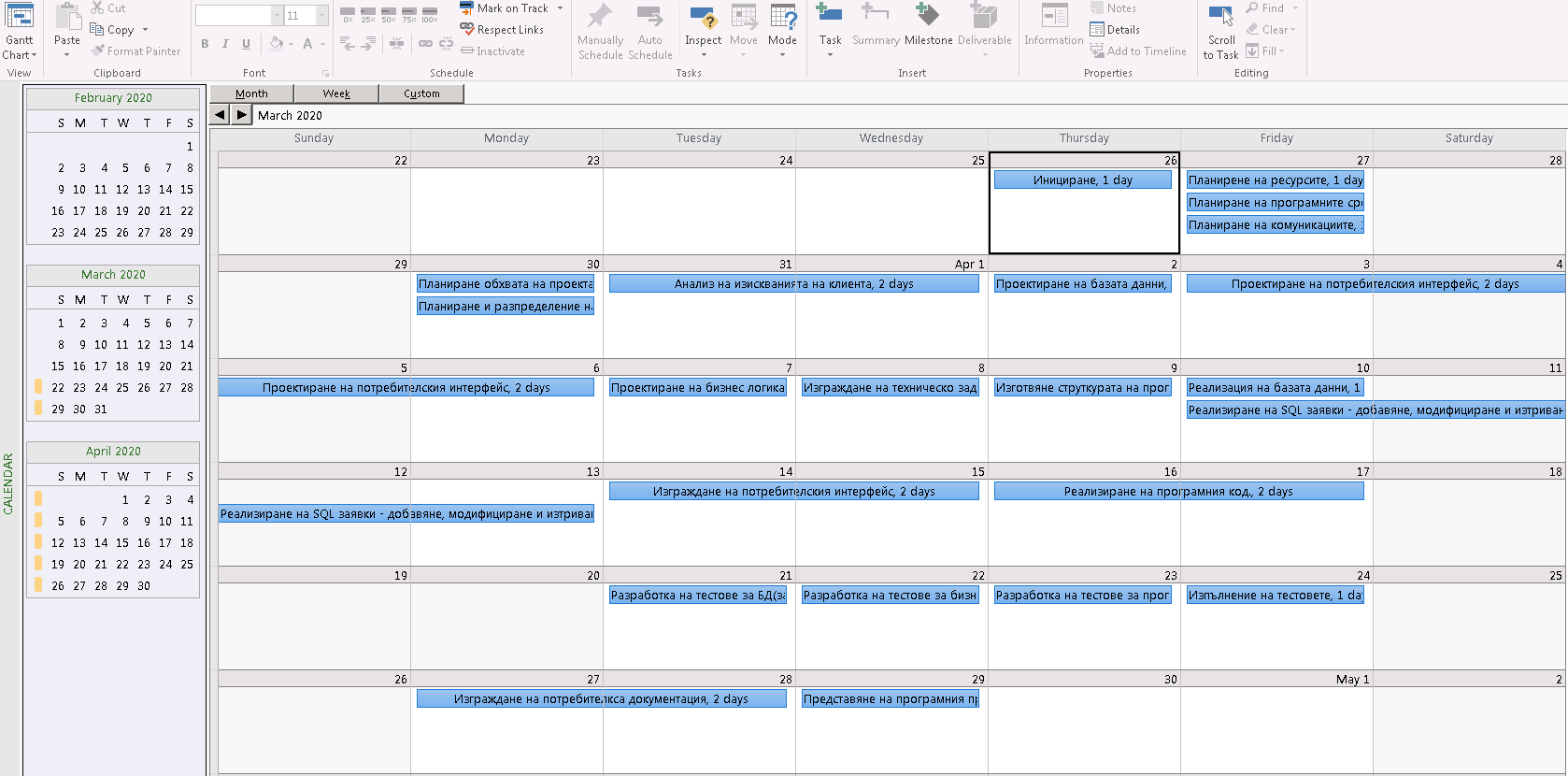
**Проследяване на завършеността на задачите:**

****

****

****

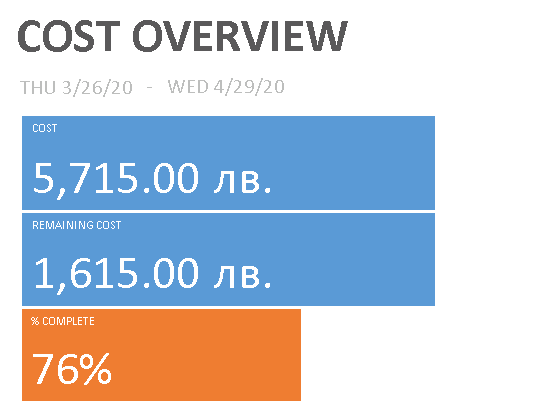
**Календар:**

****

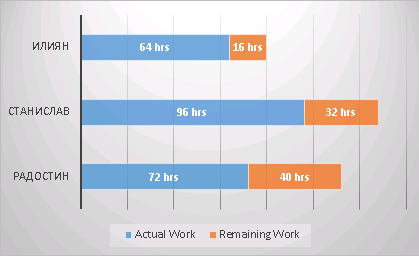
**Диаграма на бюджет**

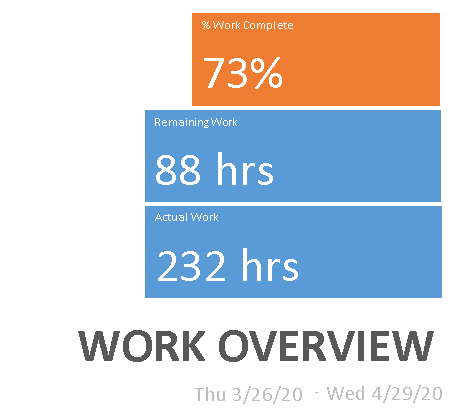
**Таблица на бюджета**

****

****

**Диаграма на свършената работа**

****

****

**Диаграма на ресурсите**

**Кръгова диаграма на ресурси**

Изводи: