**Курсов проект**

**по**

**Управление на софтуерни проекти**

**Изработили: Димитър Николов,**

**Йордан Йорданов,**

**Даниел Бодуров**

**Факултетен №: 17621777,**

**17621786,**

**17621781**

**Специалност: СИТ**

**Курс: III**

**Група: 6**

***Техническо задание на проекта***

# Бюджет

Бюджетът е важна част от всеки проект. Реалистичната му стойност и неговата добра планираност са шанс за успех на проекта. Това е важна част от оценяването на всеки проект. Изготвянето на бюджета трябва да следва логическата последователност на предвидените в проекта дейности. Направете остойностяване на всяка дейност въз основа на цените, валидни в момента на изготвяне на бюджета като вземете пред вид очакваната инфлация. Всеки бюджет изисква обосновка – показва се как са получени посочените цифри. Ако планираните средства са по-малко от действително необходимите, това би застрашило успешното изпълнение на проекта.

Имайки предвид всичко това, сумата за продукта, готов за експлоатация се свежда до 5151,20 лв, като към нея се добавят и 10% от общата сума за следващите процеси свързани с поддръжка на софтуера. Допуска се и вземането под лизинг без оскъпяване.

# Предмет на техническото задание

Предмет на този документ са изискванията на възложителя относно желаната функционалност на системата. Разработваната система предлага съхраняване на данни за песни, профили на отделните потребители 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 да съдържа екрани със следната функционалност:

**Начален екран**

На началния екрана се намират опциите за Login, Registration, както и името на нашия продукт. При избиране на дадения бутон, потребителя се препраща към съответната форма за попълване.

**Екран за Login**

На този екран потребителите имат възможността да се впишат във вече съществуващ профил или да си направят нов такъв.

**Екран за Registration**

На този екран потребителите имат възможността да си създават нов акаунт, свързан със списъка им с песни.

**Playlist екран**

На този екран, логнатият потребител може да добавя/премахва песни в своя списък. Също така може да преглежда подробни данни за избрана песен.

**Екран за преглед на песен**

Този екран се отваря след клик върху preview song бутона с първоначално избрана песен. В него се виждат детайли за песента като: Заглавие, Изпълнител, Година, Жанр, Album art, и линк към песента в YouTube.

**Екран за добавяне на песен**

Тук потребителя може да добави нова песен в общата база данни и в плейлиста на потребителя.

# Обхват на разработката

Разработеният софтуерен продукт трябва да удовлетворява очакванията на потребителите и да им предоставя възможност за улеснено използване на заложените функционалности. Потребителите ще имат на разположение продукт, който им позволява да съхраняват информация за предпочитани от потребителя песни, възможност за добавяне на нови, премахване на дадени песни и възможност за пускане на песента през линк. Въведената допълнителна информация като изпълнител, година и жанр на песента дават допълнителни възможности за улеснение на потребителя при търсене на дадена песен.

# Изисквания към програмния продукт за избор на песен

## Общи изисквания

Софтуерният продукт за избор на песни трябва да отговаря на следните изисквания. Всички случаи на използване и правилното поведение на системата трябва да са прецизно описани. Необходима е консистентност на информацията, състояща се в непротиворечащи си входни данни. Продуктът е реализиран като приложение, което може да бъде инсталирано върху всяка Windows базирана система.

Машината трябва да притежава следните характеристики:

* **Хардуер**
  + CPU: Pentium 4 Willamette 1.5 GHz (or better)
  + Storage: над 850 MB
  + RAM: минимум 512 MB
  + GPU: GeForce4 MX 440
* **Операционна** **система**
  + Windows 7 or newer

## 4.2. Функционални изисквания – системата за избор на песен трябва да предоставя следните възможности:

## 4.2.1. Модул за въвеждане на информация за песен

* Възможност за избиране на име
* Възможност за избиране на изпълнител/и
* Възможност за избиране на година
* Възможност за избиране на жанр
* Възможност за избиране на Album art
* Възможност за избиране на линк

### **4.2.2. Модул за търсене на песни, според определени критерии**

* Възможност за търсене по заглавие
* Възможност за търсене по година
* Възможност за търсене по жанр
* Възможност за търсене по изпулнител

# 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 на български език.

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

# Софтуерна поддръжка – възможност за разработване на допълнителни модули

Със закупуване на софтуерния продукт възложителят получава право над програмния код, който при желание от негова страна, може да бъде доразработен от изпълнителя, включвайки нови функционалности и допълнителни модули или промяна на съществуващи такива – създаване на нови версии на програмния продукт. За осъществяване на по-нататъшни срещи, свързани с доразвиване на системата, поръчителят може да се свърже с представително лице на посочения имейл – tech@musicae.com или на мобилен телефон 0886954835.

# Инсталиране и тестване и поддръжка

За инсталиране на приложението се използва предоставеният архивен .exe файл като се следват стъпките по подразбиране. Според подписания с възложителя договор изпълнителят поема отговорност за безплатна поддръжка в рамките на първия месец. След изтичане на този период поръчителят трябва да заплаща сума от 5152 лева месечно за поддръжка (10 % от цялата цена на продукта). При възникване на аномалии в поведението на програмния продукт, екипът от разработчици ще следи и отстранява грешки, както и ще предостави мониторинг на активността на потребителите на приложението в Google Analytics при желание на възложителя, в рамките на гаранционния период от 2 години.

1. **Система за контрол на версиите**

[Система за контрол на версиите](http://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D0%B7%D0%B0_%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB_%D0%BD%D0%B0_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D0%B8%D1%82%D0%B5) се ползва за записване на промените по даден файл или колекция от файлове. Тя позволява да се запази история на промените, да се върне предишна версия, да се добави описание на промените (като например защо и на какво е извършена промяната и други).

**Git**

За разработването на този софтуерен продукт е използвана децентрализираната система за контрол на версиите – Git. Git е впечатляващо бърза, ефективна и разполага с невероятна branching система за нелинейна разработка, като това е само част от причината тази система да бъде избрана за настоящия проект заедно с останалите, които ще бъдат споменати по-нататък . С Git много потребители могат да работят едновременно по един и същ файл, без това да доведе до бъркотия и хаос в съдържанието на файла.

Предимствата, които предоставя тази система са свързани най-вече със следните характеристики :

* Скорост
* Опростен дизайн
* Мощна поддръжка за нелинейна разработка (хиляди паралелни клонове код)
* Напълно разпределена работа
* Възможност да облужва ефективно големи проекти като Linux ядрото (по отношение на скорост и обем на данните)

Друго нещо, което трябва да се спомене е, че Git съхранява и разглежда информацията съвсем различно в сравнение с другите системи, макар потребителският ѝ интерфейс да е подобен с техните. Ето и част от разликите :

**Snapshot-и вместо разлики**

Основната разлика между Git и другите системи за контрол на версиите е начинът, по който Git третира данните. Концептуално, повечето други системи записват информацията като списък от файлово-базирани промени, тоест тези системи виждат информацията, която съхраняват като колекция от файлове и промените направени във файловете във времето. Вместо това, Git възприема своите данни по-скоро като колекция от snapshots (моментни снимки на статуса) на една миниатюрна файлова система. Всеки път когато къмитвате или записвате статуса на вашия проект в Git, системата най-общо казано прави снимка на това как изглеждат файловете ви в този момент и съхранява референция към този snapshot. Това е важна разлика между Git и почти всички други VCS.

**Почти всяка операция е локална**

Повечето операции в Git се нуждаят само от локални файлове и ресурси - общо взето не се нуждаете от информация намираща се в мрежата. Например, за да ви покаже историята на проекта, Git не се нуждае да контактува със сървъра а просто чете директно от локалната си база данни и вие виждате историята почти незабавно. Ако желаете да видите промените в даден файл между текущата му версия и тази отпреди месец, Git ще направи локална калкулация на разликите, вместо да трябва да пита отдалечения сървър за това или да трябва да издърпва по-стара версия от сървъра и едва след това да калкулира разликите локално. Това също значи, че можете да правите почти всичко когато сте офлайн или когато VPN връзката ви не работи например.

**Git само добавя данни**

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.

**Мониторинг на интегритета на данните**

Преди да запазите файл,индексът се определя под формата на контролна сума, изчислена директно от Git. Какво представлява контролната сума? Тази стойност се изчислява, като се използват специални алгоритми и се използва за проверка на целостта на данните по време на съхранение и предаване. Невъзможно е да се промени нищо без знанието на Гит и това е важна част от философията на системата.

**Предимството на Git пред SVN/CVS и TFS**

Използването на Git решава най-големия недостатък на другите две – в случай на повреда в централния или локалния компютър, всички данни на проекта могат да бъдат загубени, тъй като се намират на едно физическо място. Предимството на Git е, че тази система за контрол на версиите е децентрализирана поради, което проектът се сваля и копира локално на компютъра на всеки участващ потребител, като така се създава бекъп. В случай на необходимост, проектът може да се възстанови от локалното копие на някой от потребителите.

**Недостатъци**

* Състои се от много команди и опции и е нужно високо ниво на разбиране, тъй като някои команди и аргументи са непоследователни до някяква степен.
* Не достатъчно добра поддръжка под Windows
* Слаби инструменти за бинарни файлове
* Не поддържа празни директории
* Липса на ограничение за контрол на достъпа

1. **Система за управление на проекта**

**Trello** е безплатно уеб-базирано приложение за управление на проекти, подпомагащ увеличаването на производителността при изпълнение на колективна работа, с понастоящем с над 4 милионаабонати. През 2013г. стартира платената услуга за бизнес класа. Трело може да се използва не само за управление на проекти, но и за организиране на личните ежедневни задачи.

Trello използва методологията Канбан, която представлява система за управление на проекти, разработена от бившия вицепрезидент на Тойота, Таиичи Она. Trello предоставя дъски (boards), списъци (lists) и карти (cards), за да се организират и приоритизират проектите. Най-общо една дъска представлява един проект. Към нея може да се изградят няколко на брой списъка – например TODO, IN PROCESS, DONE. В тези списъци се поставят задачи – карти. След създаването на задача, към нея могат да се добавят коментари, да се прикачат файлове, да се изготвят подсписъци за проверка (checklists), да се задват крайни срокове и др. Картите имат предна и задна част. В списъка на таблото се вижда само предната им част – техните заглавия и основна информация, която насочва какво съдържа задачата. Тук може да се добави и снимка, както и да се зададат различни цветове на отделни задачи – за да се групират по определен признак, или просто да се различават по-лесно. Задната част се вижда, когато се влезе в конкретната задача. Там вече има изключително много възможности, сред които чеклистове, задаване на крайни срокове, следене на активността, добавяне на коментари, закачане на файлове, архив и т.н.

**Предимства**:

* Изключително интуитивен софтуер за управление на проекти.
* Бърз поглед на подреждането на картите позволява на потребителите да знаят докъде е стигнал проекта.
* На предната част на картата може да се напише основна. На “гърба” обаче може да се попълнят различни данни, отнасящи се до конкретната задача, като всичко това може да бъде организирано лесно и визуално.

**Недостатъци**:

* Простотата и минимализмът не позволяват проследяване на проектите в по-дълбока детайлност.
* Не е подходяща система за управление на сложни проекти.
* Няма възможност да се видят задачите, подредени по потребител или краен срок.

1. **Методология на разработване**

**Екстремно програмиране, Extreme Programming, XP**

Методологията на разработване, която сме избрали е Екстремно програмиране, тази методология спада към Гъвкавите технологии, които притежават следните пет принципа:

* Реализация на „разрастващо се" (инкрементално) разработване, като изискванията за всяка следваща итерация се преформулират въз основа на оценяване на текущата;
* Регламентиране на стил на работа, основан на компетентност, мотивираност и екипност; стимулиращ творчеството психологически климат;
* Приемане на неизбежността на промените, съпътстващи софтуерните проекти и адаптиране на техниките на разработване към тях;
* Придържане към опростени конструкции и схеми на работа, с цел минимизиране на ресурсите за осъществяването им.
* Активно включване на потребителите в процеса на разработване на Софтуера;

**Други характеристики на този вид технологии са :**

• Гъвкавите технологии не предлагат универсално решение на проблемите в софтуерното производство.

• Те са подходящи, ако в организацията е налице нужният стил на работа, при това за разработка на не „критичен” софтуер.

Причината за избора на тази методология е, че тя е предназначена за малки и средни колективи, разработващи софтуер при неясни или бързо променящи се изисквания, както и заради скоростното получаване на конкретен резултат. Тъй като проектът може да претърпи промени в бъдеще, тази методология е предпочитана пред стандартните методологии, при които цената за реализация на промените е скъпоструваща.

Екстремното програмиране в частност притежава следните характеристики:

* кратки цикли
* метод на инкрементално планиране(стъпка по стъпка)
* непрекъснато тестване като тестовете се задават от потребителя
* непрекъсната интензивна комуникация
* тясно сътрудничество м/у програмистите
* ще могат да променят насоката на проекта по време на реализацията му без това да бъде свързано с големи разходи.

Екип, роли и задачи

1. Екип

* ДН
* ЙЙ
* ДБ

1. Роли

# Мениджър на проекта – ДН, ЙЙ, ДБ

* Софтуерен архитект – ЙЙ
* Разработчици на бизнес логика – ДБ
* Дизайнер на потребителския интерфейс – ДН, ЙЙ, ДБ
* Технически писател – ДН
* Тестери –ДН,ЙЙ,ДБ

1. Задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер на задачата** | **Описание** | **Предшественик** | **Продължителност (дни)** | **Изпълнител** |
| **1** | Получаване на разрешение за започване на проекта | - | - | - |
| **2** | Събиране и анализ на клиентските изисквания | 1 | 2 | ДН |
| **3** | Проектиране |  |  |  |
| **3.1** | Избор на подходяща архитектура | 2 | 2 | ДН, ЙЙ, ДБ |
| **3.2** | Проектиране на БД | 3.1 | 1 | ДН, ЙЙ, ДБ |
| **3.3** | Бизнес логика | 3.2 | 1 | ДБ, ЙЙ |
| **3.4** | Първоначален дизайн на потребителския интерфейс | 3.2 | 1 | ДН, ЙЙ, ДБ |
| **4** | Създаване на техническо задание на проекта | 3 | 2 | ДН, ЙЙ, ДБ |
| **5** | Разработване |  |  |  |
| **5.1** | Реализация на избраната архитектура | 3.1 | 1 | ЙЙ, ДБ |
| **5.2** | Реализация на БД | 3.2 | 1 | ЙЙ, ДН |
| **5.3** | Създаване на SQL заявки и процедури за извеждане, добавяне, модифициране и изтриване на данни от БД | 5.2 | 1 | ДН, ЙЙ, ДБ |
| **5.4** | Бизнес логика | 3.3 | 1 | ДН, ЙЙ, ДБ |
| **5.5** | Потребителски интерфейс | 3.4 | 1 | ДН, ЙЙ, ДБ |
| **6** | Тестване |  |  |  |
| **6.1** | БД | 5.2 | 1 | ДН, ЙЙ |
| **6.2** | Логика | 5.4 | 1 | ДН, ЙЙ, ДБ |
| **6.3** | Потребителски интерфейс | 5.5 | 1 | ДН, ЙЙ, ДБ |
| **7** | Създаване на потребителска документация | 6 | 2 | ДН, ЙЙ |
| **8** | Представяне на продукта на клиента | 7 | 1 | ЙЙ |
| **9** | Пускане в производство на програмния продукт | 8 | 1 | ДН, ЙЙ, ДБ |

1. Начална дата
   1. Проектиране – 06.04.2020
   2. Разработване
      1. Имплементация на БД – 16.04.2020
      2. Кодиране на приложението - 20.04.2020
   3. Тестване –22.04.2020

Документиране – 27.04.2020

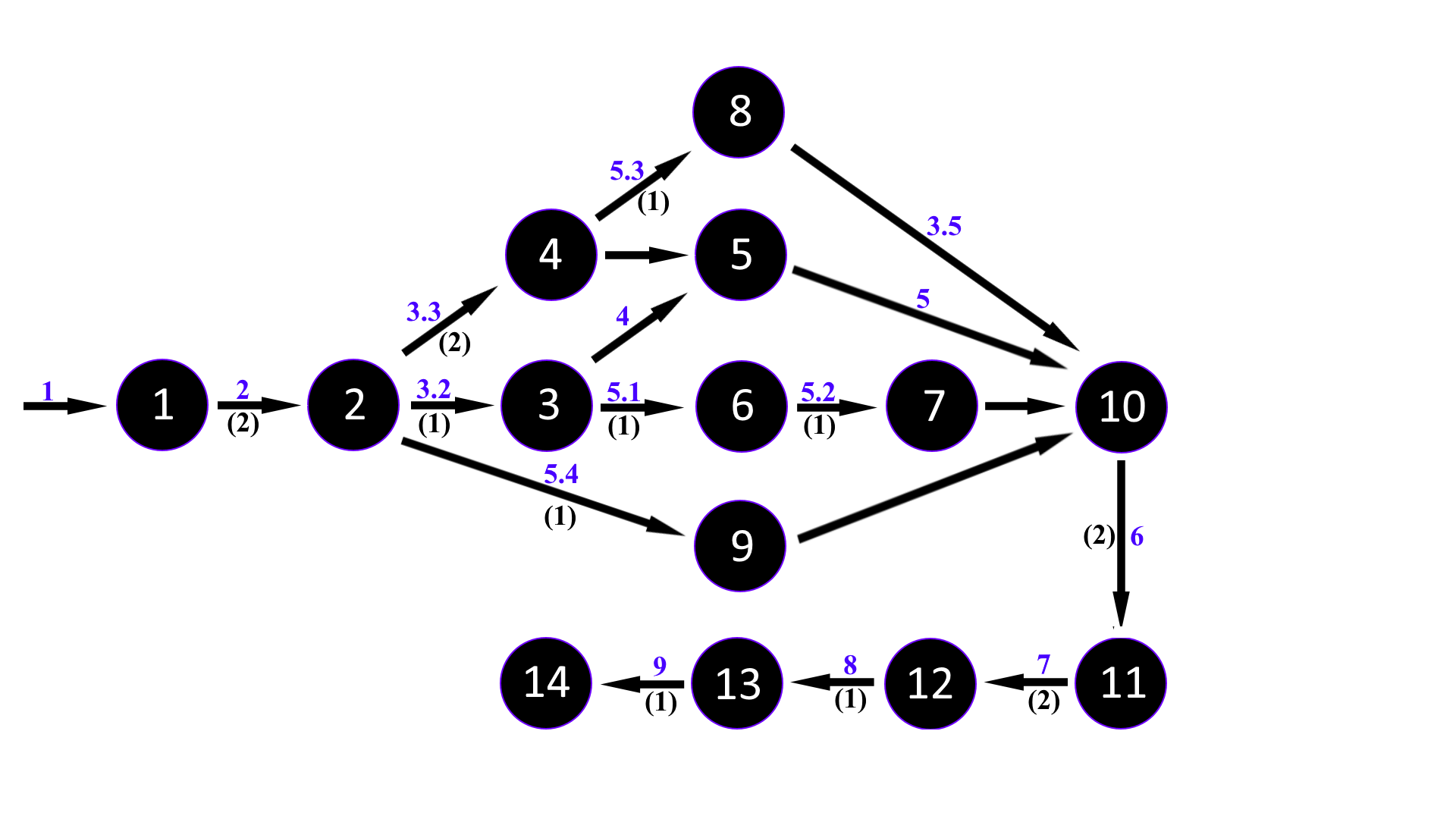
1. Програмни средства за реализация
   1. Език за програмиране – C#
   2. База данни – mySQL
   3. IDE – Visual Studio
2. Стъпки, зависещи от конкретния подход на разработка

Избраният метод на разработване предполага стъпките на разработване да са инкрементиращи се, всеки от тях да бъде кратък и да се изпълняват стъпка по стъпка, както и непрекъснато тестване, като по време на целия проект за тази методология е важна комуникацията и сътрудничеството между представителите на екипа.

1. Жалонни точки

|  |  |  |  |
| --- | --- | --- | --- |
| Събитие | Описание |  |  |
| 1 | Начало на проекта |  |
| 2 | Завършване на анализа и събирането на изискванията |  |
| 3 | Завършване на проектирането на БД |  |
| 4 | Завършване на проектирането на бизнес логиката |  |
| 5 | Създаване на техническо задание |  |
| 6 | Завършване на имплементацията на БД |  |
| 7 | Създаване на заявки |  |
| 8 | Завършване на реализацията на бизнес логиката |  |
| 9 | Създаване потребителски интерфейс |  |
| 10 | Край на разработката |  |
| 11 | Завършване на цялостното тестване |  |
| 12 | Създаване на пълната документация |  |
| 13 | Завършване на представянето на продукта |  |
| 14 | Завършване на проекта |  |

1. **Критичен път**



Изпълнението на заложения проект отнема 16 дни.

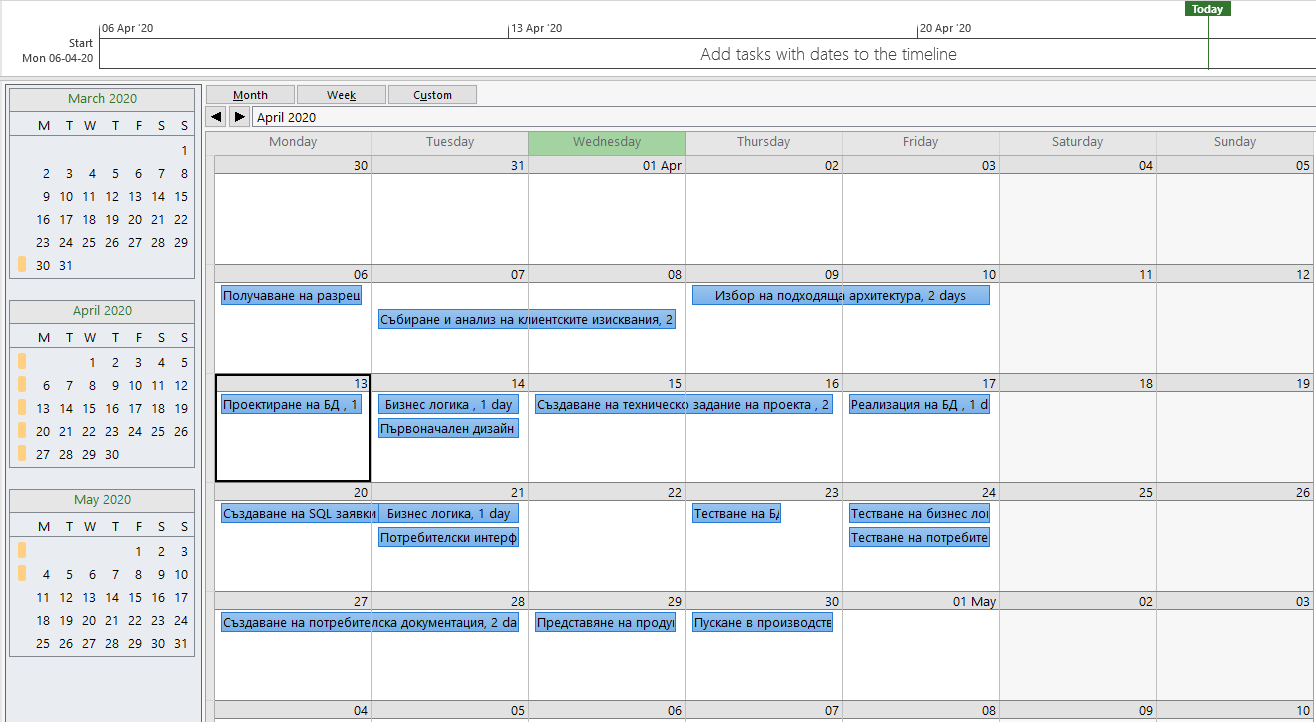
1. **Time-line**

**Trello taskboard**

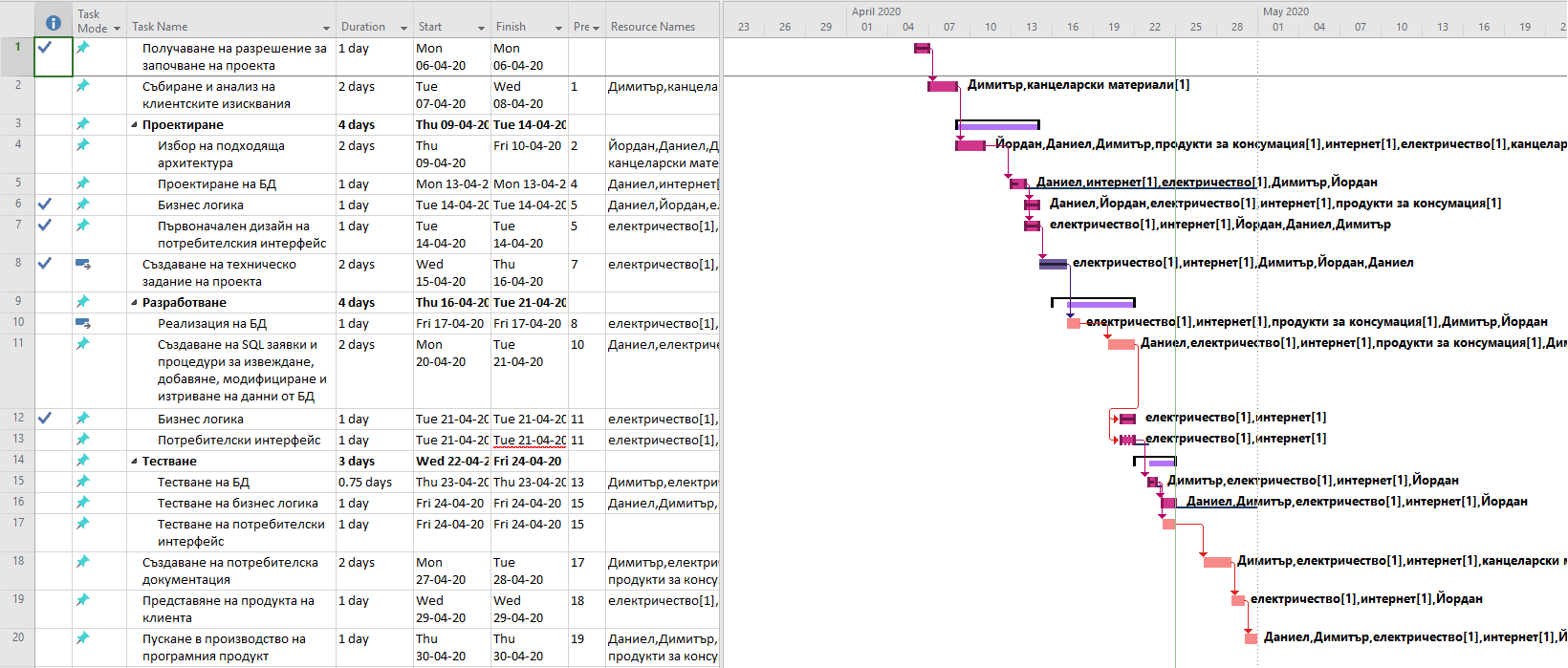


**Microsoft Project**

**Календар**

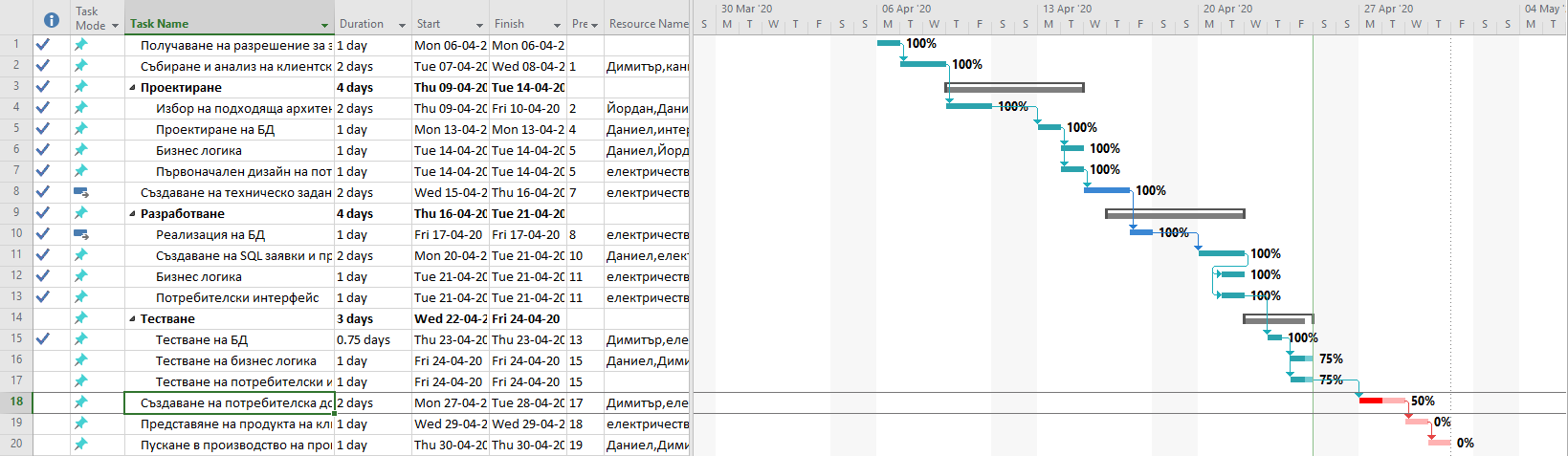


**Диаграма на Гант**

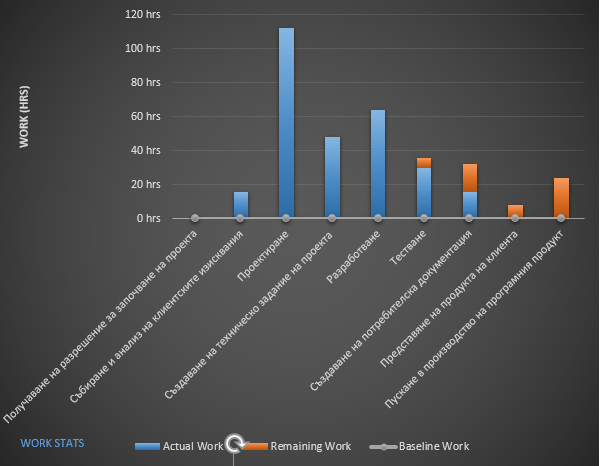


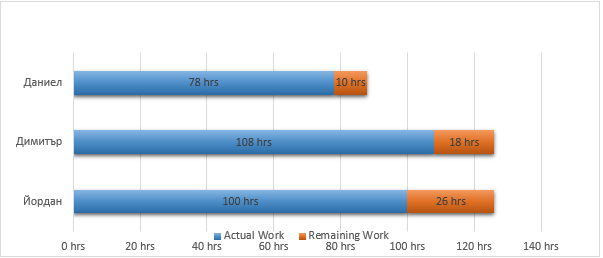
**Ресурсен лист**

**Проследяване на завършеност на задачите**

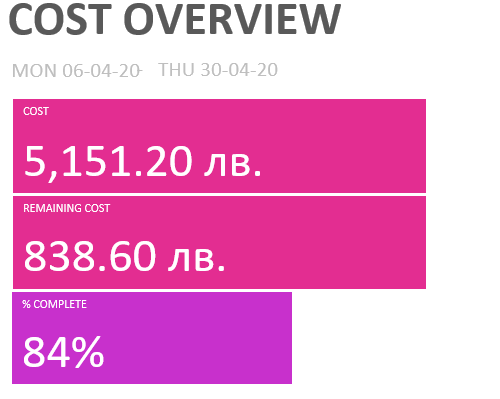


**Диаграма на свършената работа**

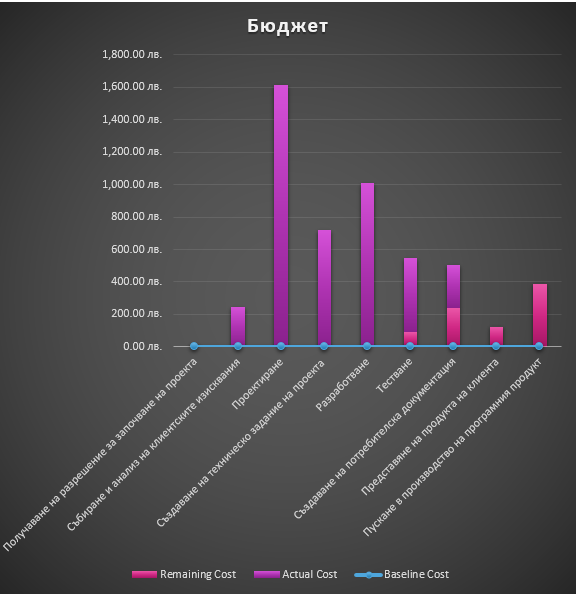


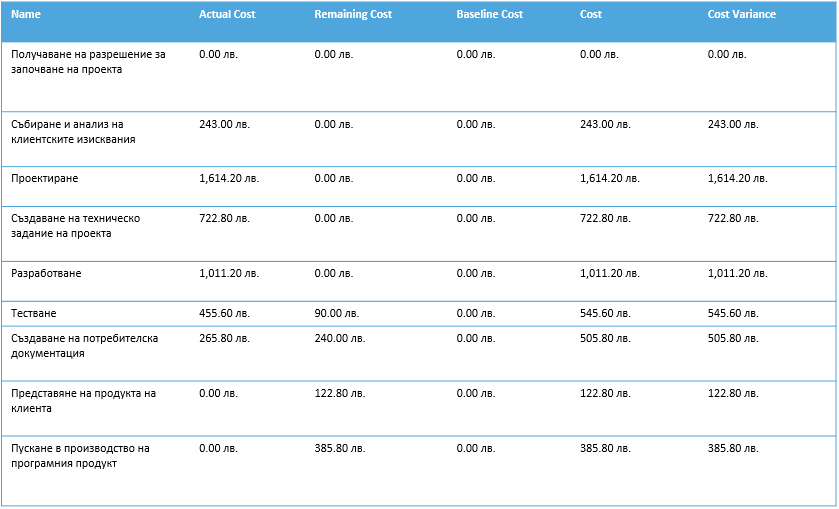


**Диаграма на бюджет**

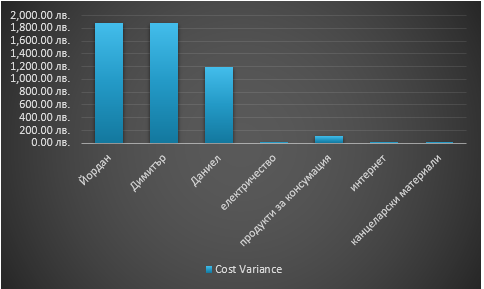


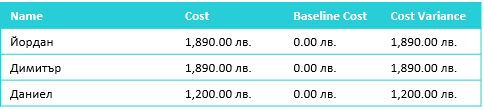
**Таблица на бюджета**



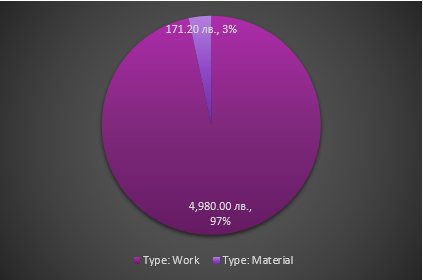


**Диаграма на ресурсите**





**Кръгова диаграма на ресурси**



**Бизнес правила**

* ***Структурни правила***:
* 1 песен може да се садържа само веднъж в списък.
* 1 песен може да има едно име.
* 1 песен може да има повече от един изпълнител.
* 1 песен може да има само една година.
* ***Процедурни правила:***
* Задължително трябва да се регистрира/логне потребител.
* Задължително трябва да има предоставен линк.
* Годината не може да е по-малка от О.
* Не може да се въведе песен без да са попълнени всички полета.

**Обекти и техните атрибути**

**User**:

* id - уникален идентификатор за потребител
* email – имейл на потребител
* password – парола на потребител

**Song**:

* id - уникален идентификатор за песен
* name – име на песен
* singer – изпълнител на песен
* genre– жанр на песен
* release – дата на издаване на песен
* youtube\_url – линк към песен

**Relations**:

* id\_user - уникален идентификатор за потребителя
* id\_song – уникален идентификатор за песен

**Изводи:**

За да постигнем пълна завършеност на проекта, трябва да се следват конкретните стъпки и изисквания при разработката на проекта. По време на разработването, всеки член на екипа трябва да отдели нужното време на отделните етапи, като не се пренебрегва важността на всеки отделен етап. Целта е финалния продукт да удовлетворява очакванията на възложителя. Използването на системи за контрол на версиите и системи за контрол на проекта повишават ефективността на работата на екипа разработчик.