**ТЕХНИЧЕСКИ УНИВЕРСИТЕТ - ВАРНА**

**ФАКУЛТЕТ ПО ИЗЧИСЛИТЕЛНА ТЕХНИКА И АВТОМАТИЗАЦИЯ**

**КАТЕДРА: СОФТУЕРНИ И ИНТЕРНЕТ ТЕХНОЛОГИИ**

**СПЕЦИАЛНОСТ: СОФТУЕРНИ И ИНТЕРНЕТ ТЕХНОЛОГИИ**



**Курсов проект по**

**“Управление на софтуерни проекти”**

**на**

|  |  |  |  |
| --- | --- | --- | --- |
| ***Имена*** | ***Фак.№*** | ***Курс*** | ***Група*** |
| Едис Рейхан Мехмед | 19621624 | III | **3a** |
| Кристиян Красимиров Костадинов | 19621798 | III | **3a** |

**Водещ преподавател:**

**/доц. М. Карова/**

***Техническо задание на проекта***

# Бюджет

Процесът по изчисляване на бюджет за даден софтуерен продукт има за цел да бъдат предвидени възможно най-точно всички разходи, които следва да бъдат покрити по време на разработката му. Етапите се подреждат както следва: анализ на изискванията спрямо продукта, изготвяне на структура, реализация на софтуера, посредством спецификациите, верификация, валидация и експлоатация. На първия етап се отделя значително повече време, тъй като от него зависи възможността да бъдат избегнати допълнителни разходи в процеса на разработка.

Имайки предвид всичко това, сумата за продукта, готов за експлоатация се свежда до 2140.6 лв, като към нея се добавят и 10% от общата сума за следващите процеси свързани с поддръжка на софтуера.

# Предмет на техническото задание

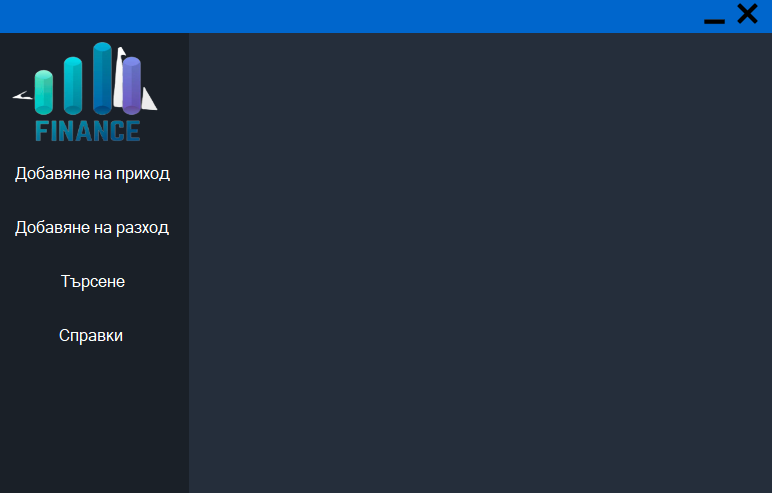
Предмет на този документ са изискванията на възложителя относно желаната функционалност на системата. Разработваната система предлага съхраняване и следене на данни за лични/фирмени приходи и разходи, извличане и визуализация на резултатите от справки, спрямо изискванията на възложителя, в удобен потребителски 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:

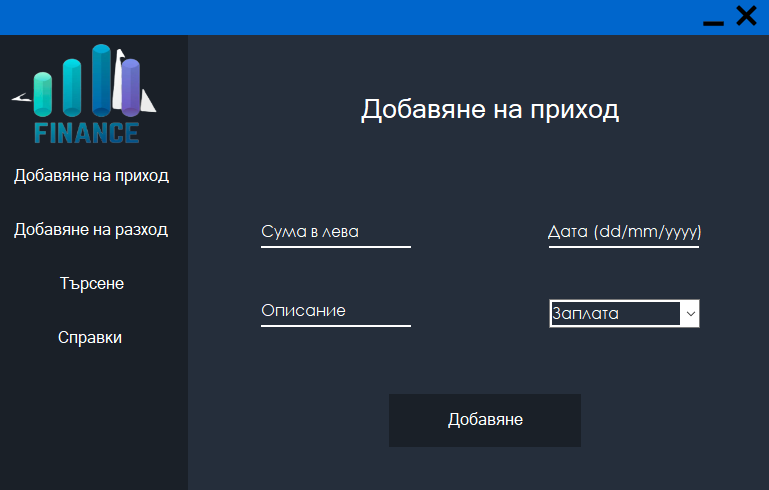
**Начален екран**

Прозорец с два основни панела, един от които представлява меню с елементи, които след избор да предоставят функционалността за която отговарят, и вторичен, по-голям панел, в който ще се визуализират резултатите от изпълнението на съответните елементи от менюто.



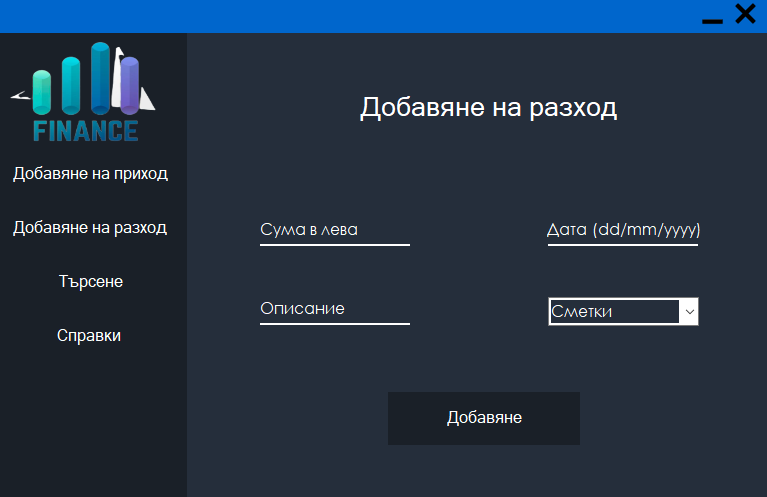
**Екран за добавяне на приход**

След като потребителят избере тази опция, да може да създаде нов запис на приход в базата данни на приложението посредством въвеждане на данни в полетата за сума, дата, описание и категория на прихода.



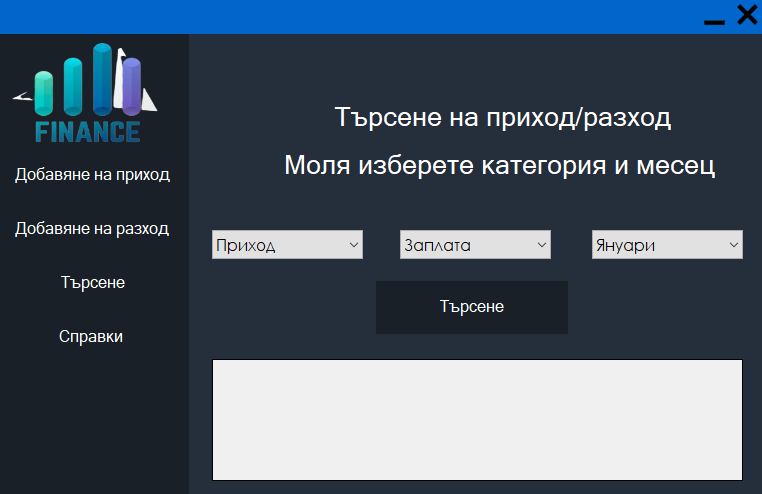
**Екран за добавяне на разход**

Опцията да реализира възможността за създаване на нов запис на разход в базата данни на приложението посредством въвеждане на данни в полетата за сума, дата, описание и категория на разхода.



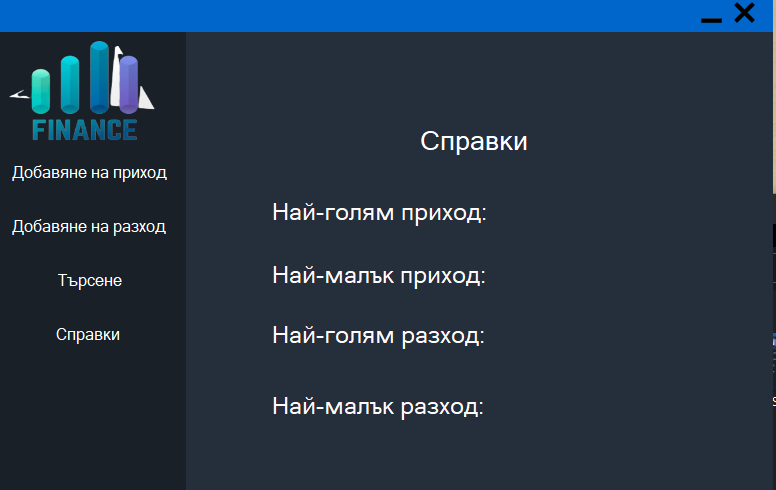
**Екран за търсене**

Да предоставя възможността потребителя да прави справка за всички приходи и всички разходи за конкретен месец по категории.



**Екран за справки**

Да предоставя възможността потребителя да извършва справка за категориите с най-много разходи, най-малко разходи, най-много приходи, най-малко приходи.



# Обхват на разработката

Разработеният софтуерен продукт трябва да отговаря на очакванията на крайният потребител и да му предоставя възможността той да се възползва пълноценно от функциите на продукта, без да изпитва каквито и да било затруднения с това. Потребителят ще има възможността по-детайлно да съхранява и следи личните или фирмените си приходи и разходи и съответно да менажира личното или фирменото си финансово състояние с реални и актуални данни. Категоризацията на приходи и разходи в системата спомага за по-удобно филтриране при реализиране на справки и търсене на записи.

# Изисквания към програмния продукт за следене на личните/фирмените приходи и разходи

## Общи изисквания

Софтуерният продукт за съхранение на приходи/разходи трябва да отговаря на следните изисквания:

Трябва да се въвеждат коректни данни като сума или дата, като за целта се грижи валидацията за това. При невалидни данни се изважда съответното съобщение. Продуктът е реализиран като десктоп приложение. Програмният продукт е предназначен за устройства, които работят с Windows. За работата на приложението, информацията въвеждана от потребителя ще се съхранява в локална SQL база данни.

Устройството трябва да притежава следните характеристики:

* **Операционна** **система**
  + Windows XP
  + Windows 7
  + Windows 10

## 4.2. Функционални изисквания – системата за съхранение на приходи/разходи трябва да предоставя следните възможности:

## 4.2.1. Модул за въвеждане на информация за приход

* Възможност за въвеждане на сума
* Възможност за избиране на категория
* Възможност за въвеждане на описание
* Възможност за въвеждане на дата

### **4.2.2. Модул за въвеждане на информация за разход**

* Възможност за въвеждане на сума
* Възможност за избиране на категория
* Възможност за въвеждане на описание
* Възможност за въвеждане на дата

### **4.2.3. Модул за търсене на приход/разход**

* Възможност за избиране на приход или разход
* Възможност за избиране на категория
* Възможност за избиране на месец

### **4.2.4. Модул за извеждане на справки**

* Извеждане на най-голям приход
* Извеждане на най-малък приход
* Извеждане на най-голям разход
* Извеждане на най-малък разход

# Системна документация

Изпълнителният екип трябва да подготви следната документация:

* Папка с файлове (работеща версия на проекта)
* Връзка към вариант на управление на софтуерен проект
* Документация на проекта (.docx файл )
* Презентация до 15-20 слайда, представяща работата по изпълнение на проекта (задача на проекта, изисквания на клиента, цел на проекта, задачи, екип с роли, бюджет, система за управление на проекта, избрана методология, избрана система за контрол на версиите (screenshots), избор на програмни средства за реализацията, представяне на междинни фази, графи на изпълнение на задачата, критичен път, промени и отчети, профил на риска, изпълнение на бюджета).

# Софтуерна поддръжка – възможност за разработване на допълнителни модули

Със закупуване на софтуерния продукт, потребителят има правото да получава безплатни актуализации в рамките на шест календарни месеца с по-новите му версии. Актуализациите представляват въвеждане в експлоатация на разработваните конкурентни нововъведения спрямо други софтуерни продукти на пазара, намиращи се в същата категория. С наличието на всяка нова версия ще бъдат водени официални дискусии между ръководството и потребителите, където ще могат да бъдат предлагани и одобрявани бъдещи промени, които директно касаят софтуерният продукт и включването на нови функционалности и допълнителни модули или промяна на съществуващи такива в бъдещи версии.

# Инсталиране, тестване и поддръжка

За инсталиране на приложението се разархивира предоставеният .zip архив. Приложението се намира на настолния компютър на клиента, който хардуерно и софтуерно покрива описаните изисквания, като за правилното функциониране на системата се използва предоставеното ръководство за употреба и инсталация. Според подписания с възложителя договор, изпълнителят поема отговорност за безплатна поддръжка в рамките на първите шест месеца. След изтичане на този период, поръчителят трябва да заплаща сума от 215 лева месечно за поддръжка (приблизително 10 % от цялата цена на продукта). При възникване на аномалии в поведението на програмния продукт, екипът от разработчици ще следи и отстранява грешки.

1. **Система за контрол на версиите**

Система за контрол на версиите 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 разработка, като това е само част от причината тази система да бъде избрана за настоящия проект. С Git много потребители могат да работят едновременно по един и същ файл, без това да доведе до хаос в съдържанието на файла. Git е създаден и разработен през април 2005г. от ***Линус Торвалдс*** (човекът зад *Linux ядрото*). Торвалдс създава Git, за да го ползва за разработката на неговия проект Linux ядро. От тогава до днес Git се развива и подобрява от множество потребители. 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)

Разлики между Git и други системи:

**-Snapshots**

Основната разлика между Git и другите системи за контрол на версиите е начинът, по който Git третира данните. 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.

**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
* Недостатъчно 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**

**Microsoft Project** е софтуерен инструмент за управление на проекти, разработен от [Microsoft](https://bg.wikipedia.org/wiki/Microsoft). Той е създаден в помощ на мениджърите на проекти при разработване на планове, назначаване на ресурси към задачи, проследяване напредъка на проекта, управление на бюджети и анализ на натоварването.

В него може да се въвежда информация за задачите по проекта,

тяхното начало и продължителност на изпълнение, колко време ще отнеме изпълнението им и кой

трябва да извърши работата. Ако трябва да се проследявате зависимости, разходи или по-голяма сложност, MS Project е точният инструмент. Може да се използва настолната версия Project или Project за уеб.

**Методология на разработване**

Екстремно програмиране

Екстремно програмиране (Extreme Programming – XP) e методология за създаване на софтуер, една от няколкото [гъвкави методологии](https://bg.wikipedia.org/wiki/%D0%93%D1%8A%D0%B2%D0%BA%D0%B0%D0%B2%D0%B0_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F). Основната цел на гъвкавата методология е намиране на по-добри и по-гъвкави решения при създаването на софтуер.

Основни правила от манифеста на гъвкавите методологии, които екстремното програмиране също следва са:

1. Индивидуални качества и комуникация се предпочита пред строги процеси и инструменти
2. Работещ софтуер за сметка на изчерпателна документация
3. Клиентско съдействие измества договорно споразумение
4. Лесно реагиращ на промяна софтуер вместо стриктното следване на план

Основната цел на екстремното програмиране е да редуцира цената на проект, ако се наложи дадена промяна. От тук си вадим извод, че екстремното програмиране е методология, подходяща да се използва при проекти, който имат често променящи се изисквания и тогава по-стандартни методологии (като [Waterfall модела](https://bg.wikipedia.org/wiki/%D0%9F%D0%BE%D0%B5%D1%82%D0%B0%D0%BF%D0%BD%D0%BE_%D0%BF%D0%BB%D0%B0%D0%BD%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) например):

* Не са оптимални за постигане на голяма продуктивност;
* Подходяща е при проекти, които включват нови технологии или непредвидими проблеми, свързани с имплементацията;
* Използва се също така при по-малки и по-лесни за реализация проекти с неофициални методи;
* Добра технология за проекти, изискващи изследване.

Дейности, които екстремното програмиране описва и които се използват при самия процес на софтуерна разработка:

1. **Кодиране** – Защитниците на екстремното програмиране смятат, че единственото наистина важно нещо в разработката на софтуер е писането на код.
2. **Тестване** – без тестване не можем да бъдем сигурни дали нашия продукт наистина отговаря на изискванията на спецификацията; не можем да сме сигурни дали това, което сме написали е това, което сме искали да напишем – за целта използваме Unit Test; не можем да бъдем сигурни дали това, което сме имали предвид е това, което е трябвало да имаме предвид – за да проверим това правим *приемствени тестове*(acceptance tests) на изискванията, който са дадени от клиента (дадени в exploration phase of release planning).
3. **Слушане** – За програмистите по принцип не е необходимо да са наясно с бизнес страната на самата система. От друга страна те трябва да „слушат“, за да знаят от какво се нуждае клиента.
4. **Правене на дизайн** – При създаването на продукт, използвайки екстремно програмиране, едно от най-важните неща е създаването на добър дизайн в началото.

***Описание на стъпките по реализацията***

**Екип, роли и задачи**

1. Екип

* Едис Мехмед
* Кристиян Костадинов

1. Роли

* Мениджър на проекта – Едис Мехмед
* Софтуерен архитект – Кристиян Костадинов
* Разработчици на бизнес логика – Едис Мехмед, Кристиян Костадинов
* Дизайнер на потребителския интерфейс – Кристиян Костадинов
* Технически писател – Едис Мехмед
* Тестери – Едис Мехмед, Кристиян Костадинов

1. Начална дата
   1. Проектиране – 15.03.2022
   2. Разработване – 17.03.2022

3.2.1. Потребителски интерфейс - 17.03.2022

3.2.2. Имплементация на БД – 22.03.2022

3.2.3. Кодиране - 24.03.2022

* 1. Тестване – 30.03.2022

Документиране – 4.04.2022

1. Програмни средства за реализация
   1. Език за програмиране – C#
   2. База данни – MS SQL Server
   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.

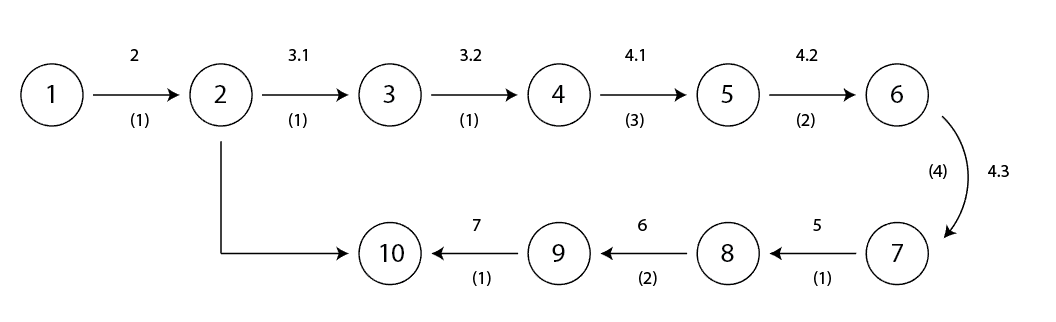
6. Разделяне на задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер на задачата | Описание | Предшественик | Продължителност(дни) | Изпълнител |
| **1** | Получаване на разрешение за започване на проекта | - | - | - |
| **2** | Анализ на клиентските изисквания | 1 | 1 | Е.М. |
| **3** | Проектиране | - | - | - |
| **3.1** | Избор на подходяща архитектура | 2 | 1 | К.К. |
| **3.2** | Проектиране на база данни | 3.1 | 1 | Е.М. |
| **4** | Разработване | - | - | - |
| **4.1** | Реализация на потребителски интерфейс | 4 | 3 | К.К. |
| **4.2** | Реализация на база данни | 4.1 | 2 | Е.М. |
| **4.3** | Реализация на бизнес логика | 4.2 | 4 | Е.М.  К.К. |
| **5** | Тестване | - | - | - |
| **5.1** | Потребителски интерфейс | 5 | 1 | К.К. |
| **5.2** | База данни | 5.1 | 1 | Е.М. |
| **5.3** | Логика | 5.2 | 1 | К.К. |
| **6** | Създаване на документация | 5.3 | 2 | Е.М.  К.К. |
| **7** | Представяне на продукта пред клиента | 6 | 1 | Е.М. |

|  |  |
| --- | --- |
| **Събитие** | Описание |
| **1** | Начало на проекта |
| **2** | Извършен анализ на клиентските изисквания |
| **3** | Избиране на архитектура |
| **4** | Завършване на проектирането на БД |
| **5** | Реализация на потребителски интерфейс |
| **6** | Реализация на БД |
| **7** | Реализация на бизнес логика |
| **8** | Тестване на продукта |
| **9** | Реализация на продуктна документация |
| **10** | Край на проекта |

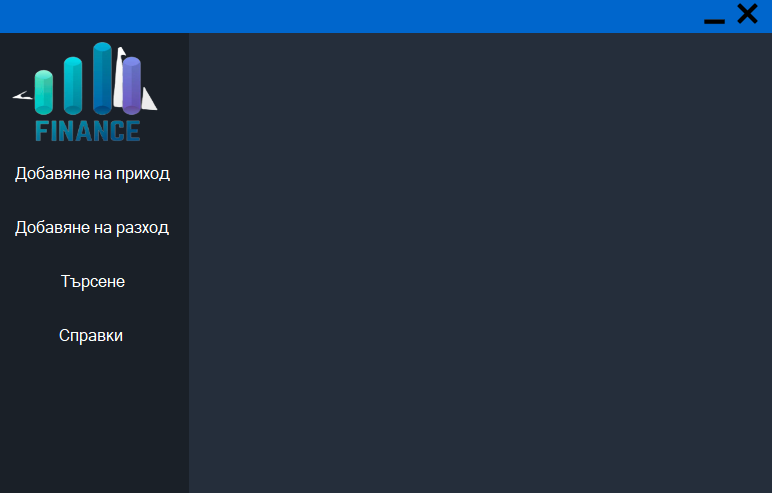
7. Жалонни точки

**8. Критичен път**

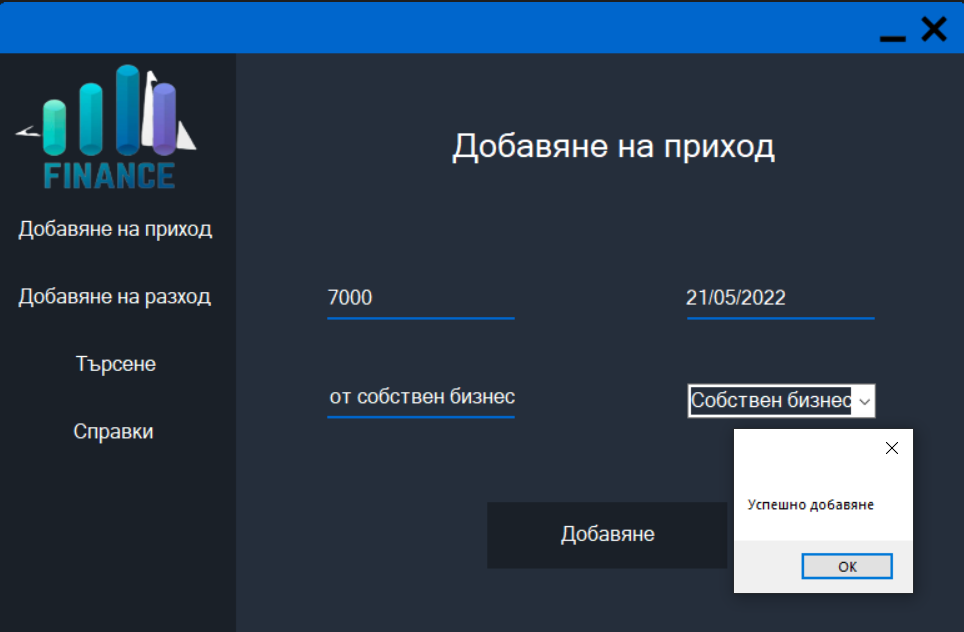
****

**Краен продукт**

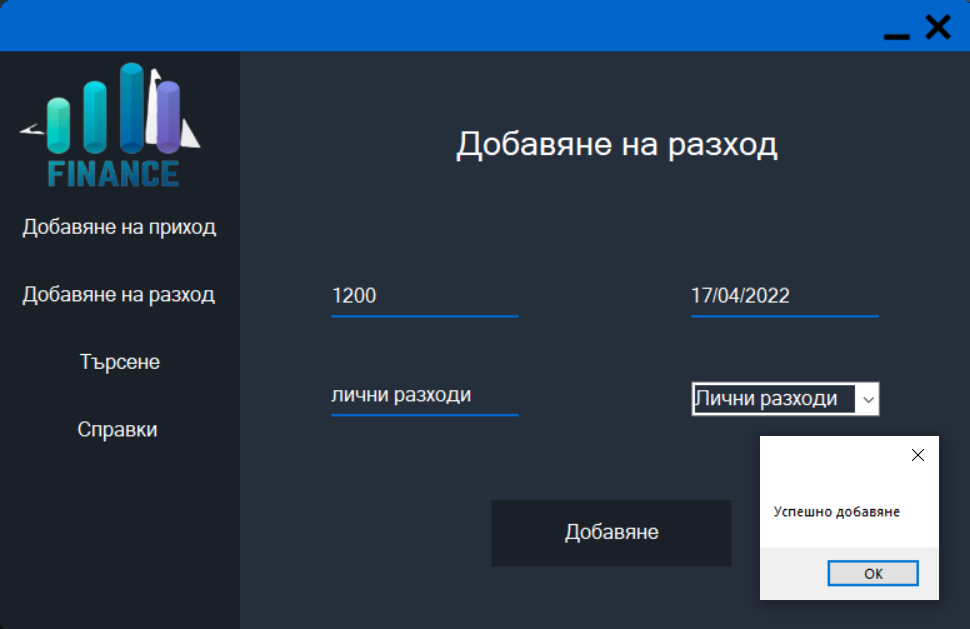
Начален екран при първоначално стартиране



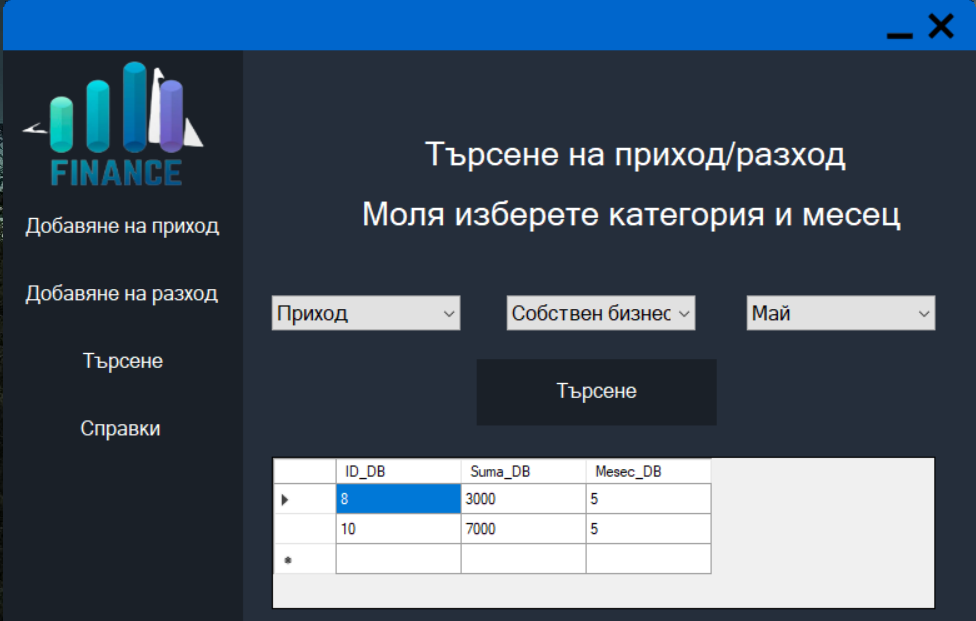
Екран за добавяне на приход



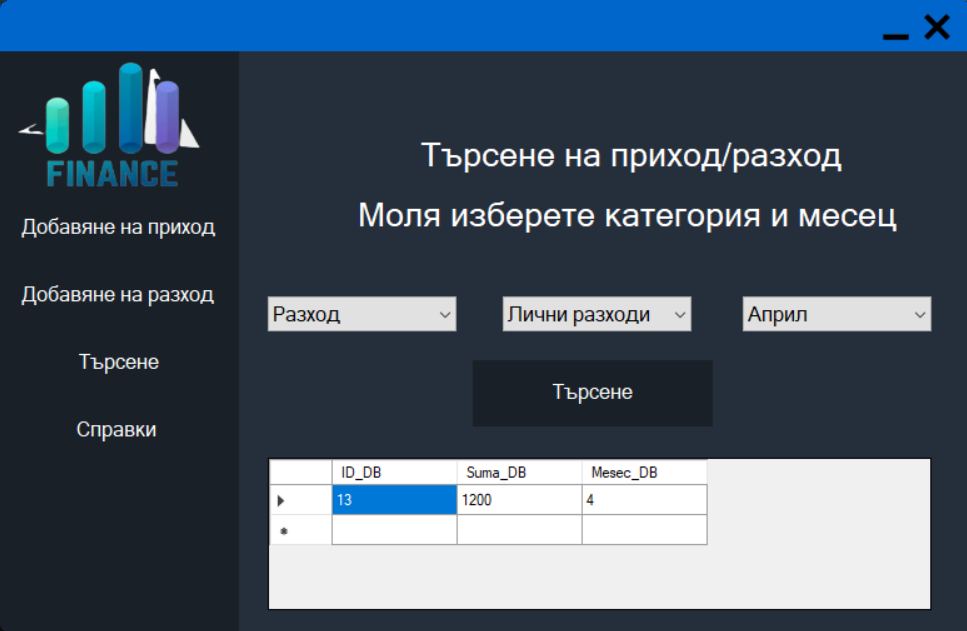
Екран за добавяне на разход



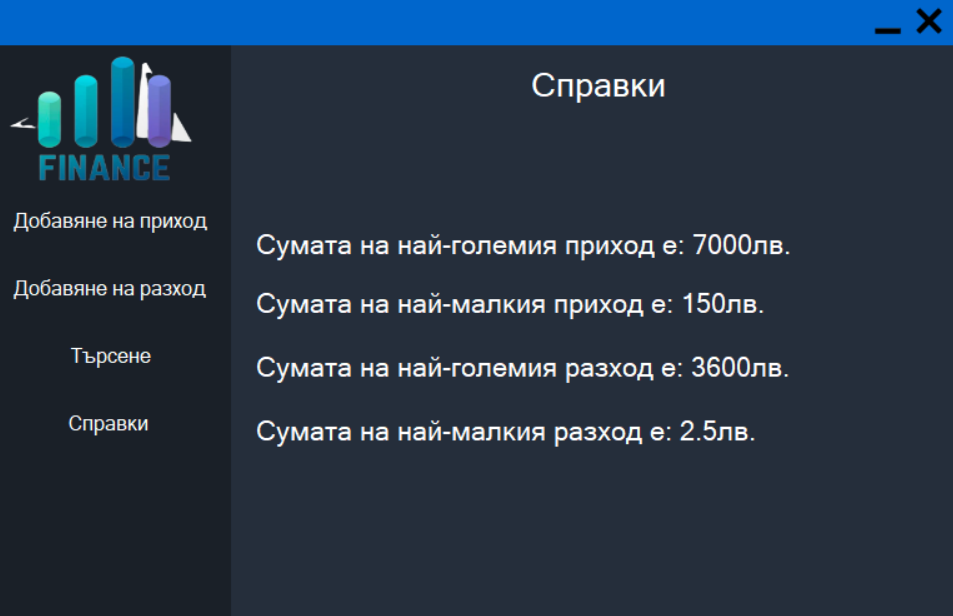
Екран за търсене на приход



Екран за търсене на разход

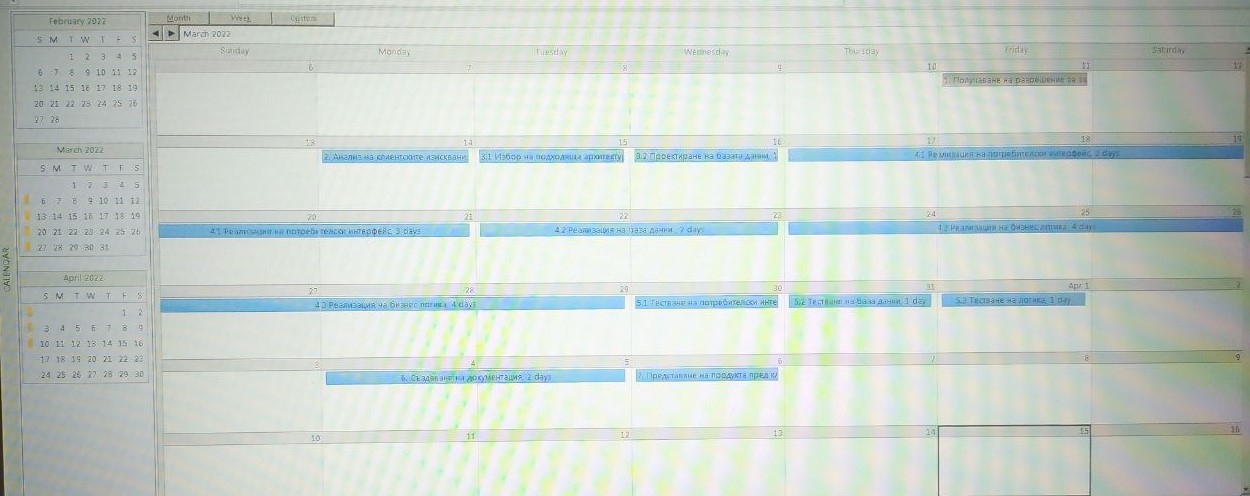


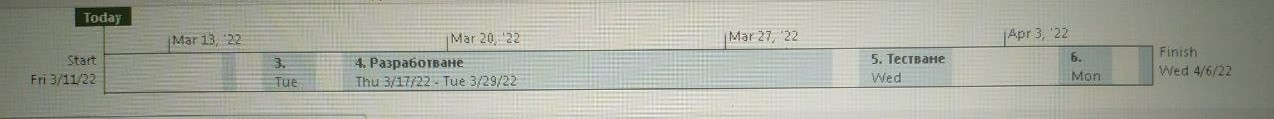
Екран за справки



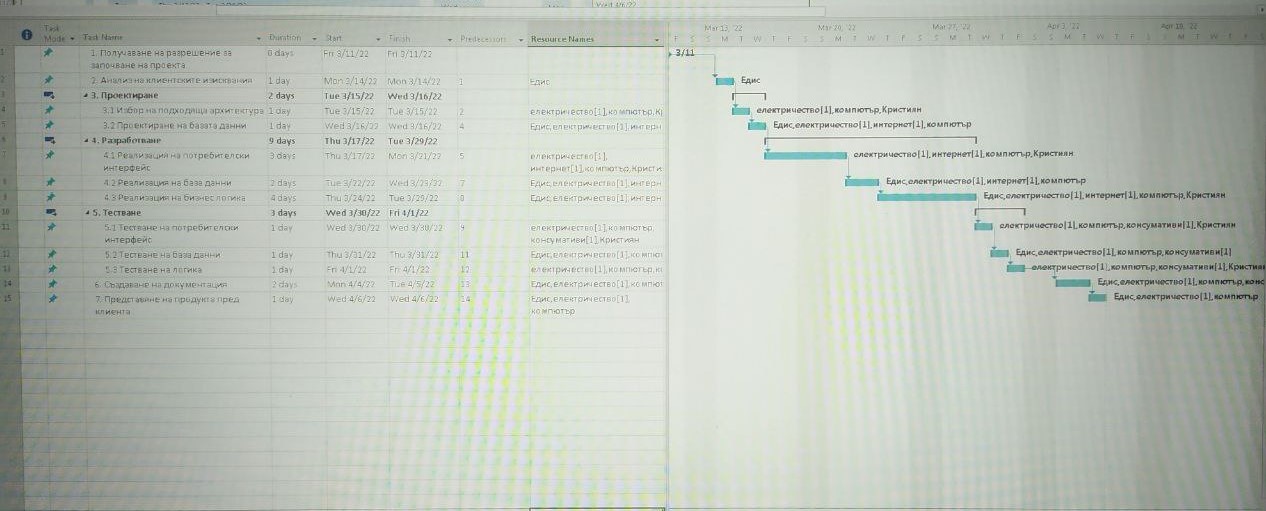
**Microsoft Project**

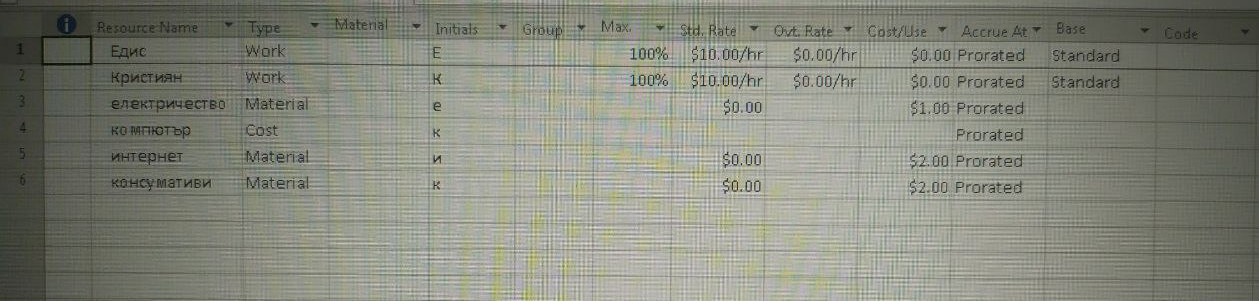
**Календар**

****

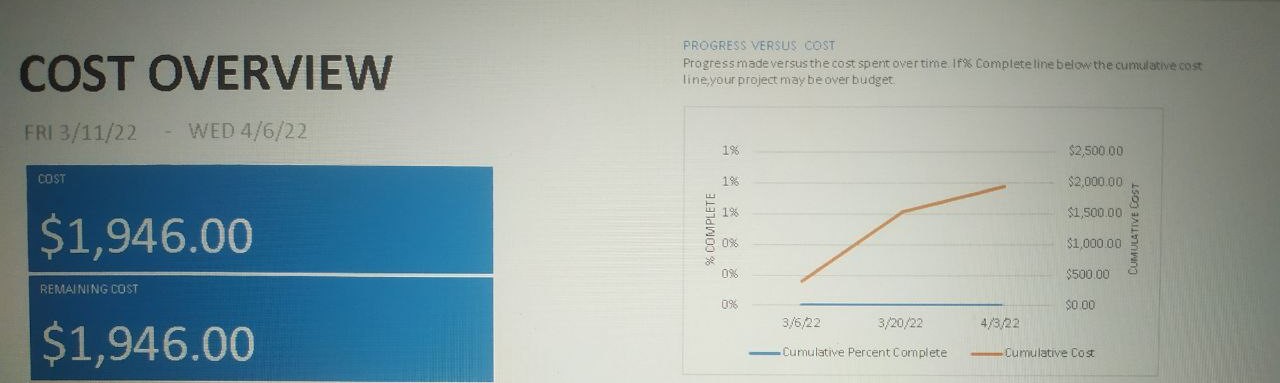
**Timelinе**

**Диаграма на Гант**

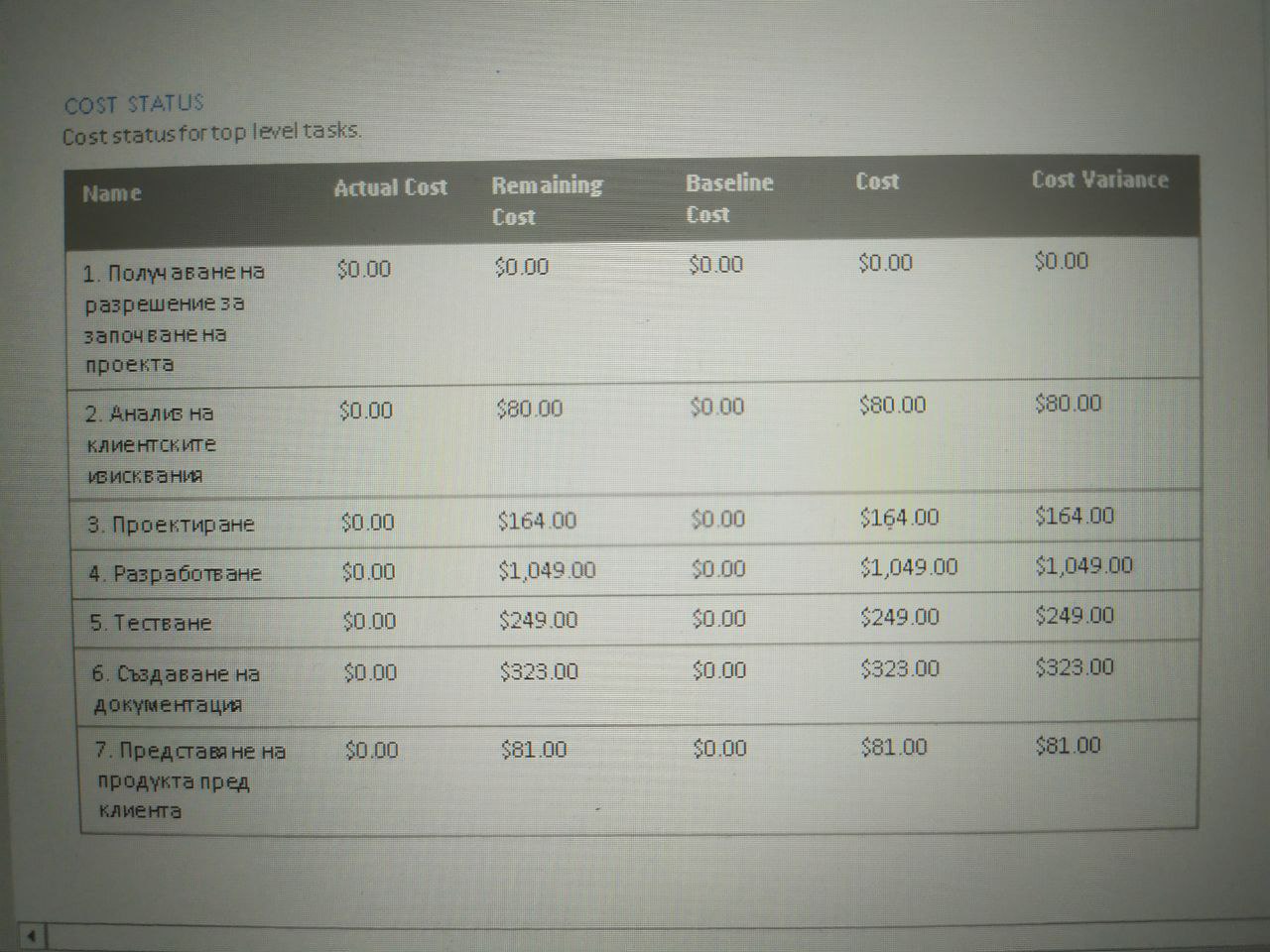
****

** Ресурсен лист**

**Диаграма на бюджет**

****

**Таблица на бюджет**

****

**Бизнес правила**

* ***Структурни правила***:
* всеки разход може да бъде от един тип .
* всеки разход може да има една сума.
* всеки приход може да бъде от един тип .
* всеки приход може да има една сума.
* ***Процедурни правила:***
* Сумата на прихода/разхода не може да бъде отрицателно число или нула.
* Датата трябва да е във формат (dd/mm/yyyy).
* Не може да се въведе приход/разход без да са попълнени всички полета.

**Обекти и техните атрибути**

**DataBaseClass**:

* *Suma* – сумата на прихода/разхода
* *Data* – дата на прихода/разхода, която трябва да е във формат (dd/mm/yyyy)
* *Opisanie* – описание към прихода/разхода в свободен текст
* *Tip* – атрибутът, който се определя от това дали въвеждаме приход или разход
* *Kategoria* – всеки приход/разход има няколко различни категории
* *Mesec* - атрибутът се определя от въведената дата на прихода/разхода, с цел по-лесно търсене

**ИЗВОДИ**

Управлението на софтуерни проекти е сложен и последователен процес. За достигане на успешно завършен проект, удовлетворяващ изискванията на възложителя, е нужно да се отдели достатъчно време на всеки един от етапите при разработка. Допълнително внимание трябва да се отделя и на етапите по планиране и анализ на отделните детайли, както и на бюджетирането, съставянето на правилния екип и риск мениджмънта на проекта.