**Курсов проект**

по

управление на софтуерни проекти

Техническо задание на проекта

Бюджет

Бюджета за проекта е създаден въз основа на всички етапи на разработка му (Планиране, Разработка, Документиране). Планирането е много важен етап, като той включва анализа на нужната архитектура, видовете валидации и начина на тяхното тестване, за тях е отделен един работен ден. За разработката на проекта са отделени 31 дена. Този етап отнема повече време, за да се избегнат евентуални бъдещи грешки, чието поправяне би струвало повече според периода на работа по проекта, в който са открити. В този етап се включва разработката на архитектурата, която е важна за определянето на нужните средства и технологии, които ще се използват. От нея също така зависи и добре написаният код, при който липсва повторение и всичко е добре структурирано. За цялостното тестване е отделен само един работен ден, а за изработването на двете документации са отделени 4 работни дни.

Общата стойност на проекта възлиза на **37 086.59лв**. От които 31 394.99 са отделени за заплащане на програмистите, 5 569.30 са отделени за закупуването на техниката, 92.30 са отделени за електричество и 30лв за интернет. Добавени са и 20% за евентуални бъдещи процеси отнасящи се към поддръжката и на библиотеката.

Предмет на техническо задание

Проекта изисква проектиране на библиотека за валидиране на лични данни на човек, които включват: име, възраст, дата на раждане, ЕГН и др.

Обосновка на избора на система за контрол на версиите

Причина

Git е нашият избор за система на контрол на версиите, тя позволява на всеки от разработчиците да притежава собствено копие на проекта, което позволява съвместната работа по едни и същи файлове. Проследяването на историята на проекта е също една от главните причини, чрез git това става изключително лесно, всеки може да гледа предишен комит и да изразява мнение за направените промени. Създаването на различни клонове позволява на всеки един от разработчиците да комитва каквито промени сметне за нужни без да възпрепятства работните копия на колегите си.

Трудности при използването

Разрешаването на конфликти при сливане на различни клонове става сложно.

Настройки

За използването на git и запазването в отдалечена репозитория ни трябват акаунти в GitHub. Препоръчително, но не задължително, е изтеглянето на помощни средства за визуализация на движенията по репозиториите.

Съпоставка

Git позволява поетапната промяна/добавяне на файлове и позволява добавянето на коментар, който описва извършените промени. За разлика от други системи за контрол на версиите, където не всеки разработчик има цялата история на проекта, git позволява на всеки от екипа да вижда каквито и да било промени направени през цялостната разработка. Всеки разработчик синхронизира собствената си локална репозитория с една обща отдалечена и има достъп до всички файлове и функции.

Обосновка на избора на система за управление на проекти

Причина

Shortcut предоставя функционалности за лесно разпределение на задачите между разработчиците. Също така позволява добавянето на различни label-и на задачите, чрез които разграничението между различните модули в едно приложение става по-лесно. Позволява раздробяването на всяка задача на task-ове. Комуникацията между колеги става лесно като всеки един участник от екипа може да оставя коментари под всяка задача, да следи състоянието им и за неговите собствени.

Трудности при използването

Малко по-трудно е свикването на работата с него. Отнема време за откриването и използването на множеството функции, които предоставя Shortcut.

Настройки, нужни за работата с него

Нужно е създаването на акаунт и получаване на покана от създателя на проекта към неговите колеги.

Обосновка и разяснения към избраната методология за разработка на софтуера

Използван е водопадния модел на работа. Той най-често се използва при проекта с ниска сложност и където вероятността за поява на промяна на изискванията е малка и некритична.

Зададеният проект беше недостатично голям и сложен, за да оправдае използването на по-гъвкава методология. Изискванията бяха лесно дефинирани още от началото.

Тази методология трудно допуска препрявания по вече преминали фази на разработка, но важно предимство е че процесите протичат последователно и линейно.

Предимства:

* Последователност на етапите
* Добра структура
* Точни срокове
* Лесно определяне на бюджет
* Резултати се виждат след всяка фаза

Недостатъци:

* Трудно връщане към вече преминал етап
* Ниска гъвкавост

Съпоставка със спираловидния модел:

Спираловидният модел е еволюционен тип модел. Той обединява характеристики от два други модела на разработка: този на водопада и прототипния модел. От този модел произлизат постепенни версии на разработваният софтуер.

|  |  |
| --- | --- |
| **Водопаден модел** | **Спираловиден модел** |
| 1.Моделът е предпочитан от клиентите | Моделът е предпочитан от разработчиците |
| 2.Грешки и рискове се идентифицират и отстраняват след приключване на съответният етап | Грешките и рисковете са идентифицирани и отстранени по-рано |
| 3.Добре приложим при малки проекти | Добре приложим за големи проекти |
| 4.Необходимо планиране в ранен етап | Не е необходимо планиране в ранен етап |
| 5.Много малка гъвкавост | Гъвкавостта за промяна не е трудна |
| 6.Сравнително евтин модел | Много скъп модел |
| 7.Необходимо е задълбочено разбиране на изискванията от самото начало | Изискванията могат да се добавят в новите итерации, ако е необходимо |
| 8.Произвежда единичен краен продукт в края на процеса | Генерира груб работещ продукт при всяка итерация |

Описание на стъпките по реализацията:

Екип

Владимир Владимиров

Михаил Георгиев

Румен Павлов

Определяне на роли

Мениджър на проекта: Михаил

Софтуерен архитект: Михаил

Разработчик: Румен, Владимир, Михаил

Технически писател: Владимир

Тестер: Румен

Разпределяне на задачи

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер на задачата | Опсание | Предшестваща задача | Време | Изпълнител |
| 1 | **Планиране** |  |  |  |
| 2 | Архитектура |  | 1 ден | МГ |
| 3 | Видове валидационни функции | 2 | 1 ден | МГ |
| 4 | Тестове | 3 | 1 ден | МГ |
| 5 | **Разработка** | 1 | 31 дена |  |
| 6 | Архитектура | 4 | 1 ден | МГ, РП, ВВ |
| 7 | Валидационни функции | 6 | 20 дена | МГ, РП, ВВ |
| 8 | Тестове | 7 | 10 дена | МГ, РП, ВВ |
| 9 | Цялостно тестване | 8 | 1 ден | РП |
| 10 | **Документиране** | 5 | 4 дена |  |
| 11 | Обща документация | 9 | 2 дена | ВВ |
| 12 | Потребителска документация | 11 | 2 дена | ВВ |

Определяне на срокове

Начална дата: 23/03/2022

Планиране: 23/03/2022

Разработка: 24/03/2022

Документиране: 05/09/2022

Програмни средства за реализация

Език за програмиране: JavaScript

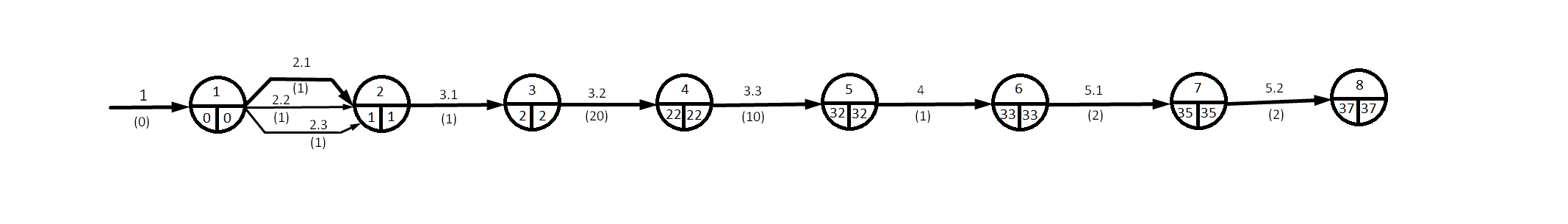
IDE: Visual Studio Code

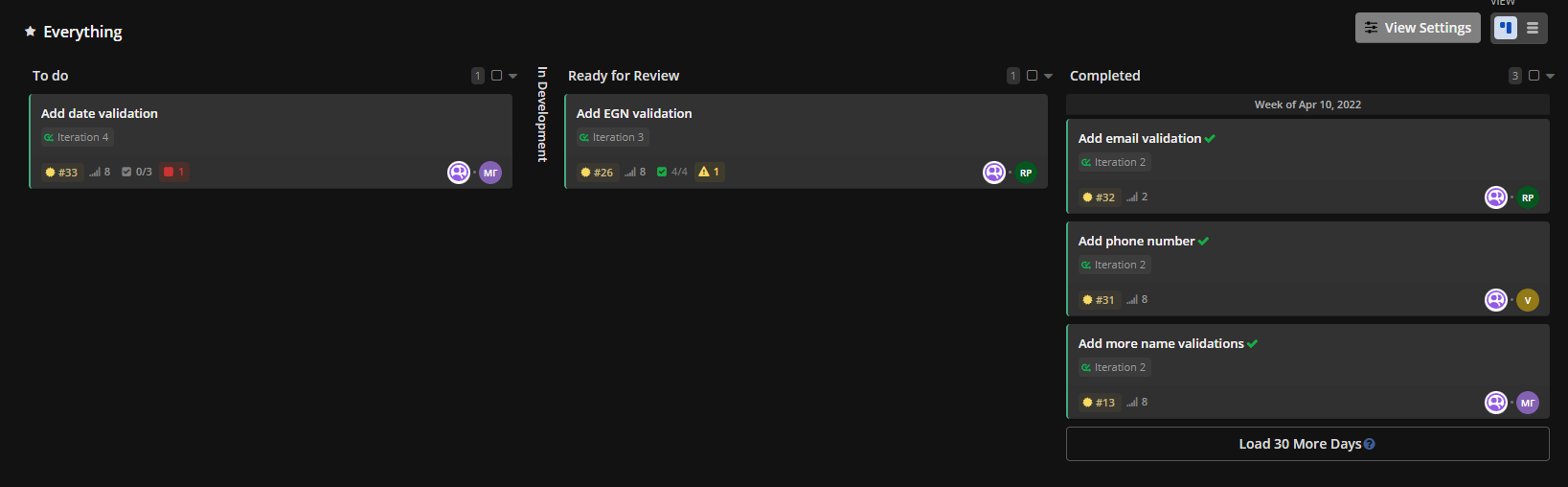
Допълнителни помощни средства: Node.js, npm, jest

Жалонни точки

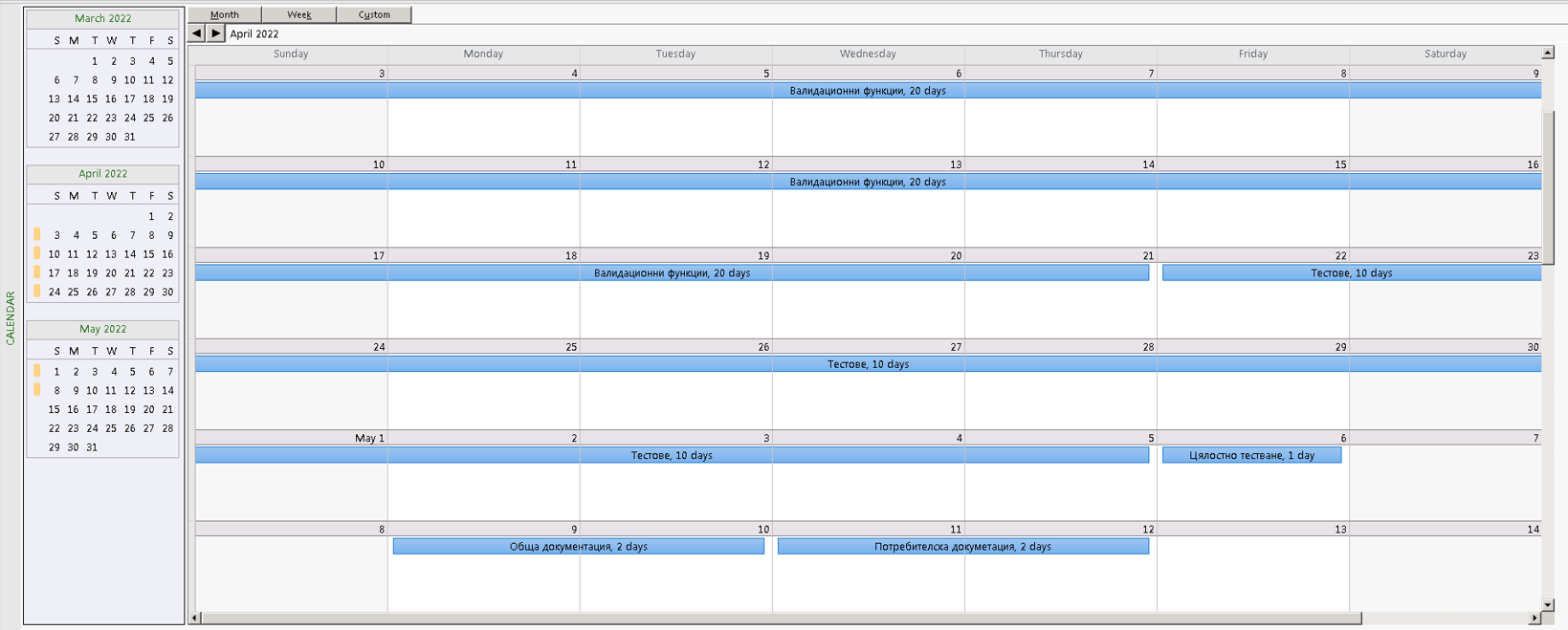
|  |  |
| --- | --- |
| Събитие | Описание |
| 1 | Начало |
| 2 | Планирана е архитектурата, валидационните функции и тестовете |
| 3 | Разработена е архитектурата |
| 4 | Разработени са валидационните функции |
| 5 | Разработени са тестовете |
| 6 | Тестването е приключило |
| 7 | Общата документация е завършена |
| 8 | Потребителската документация е завършена |

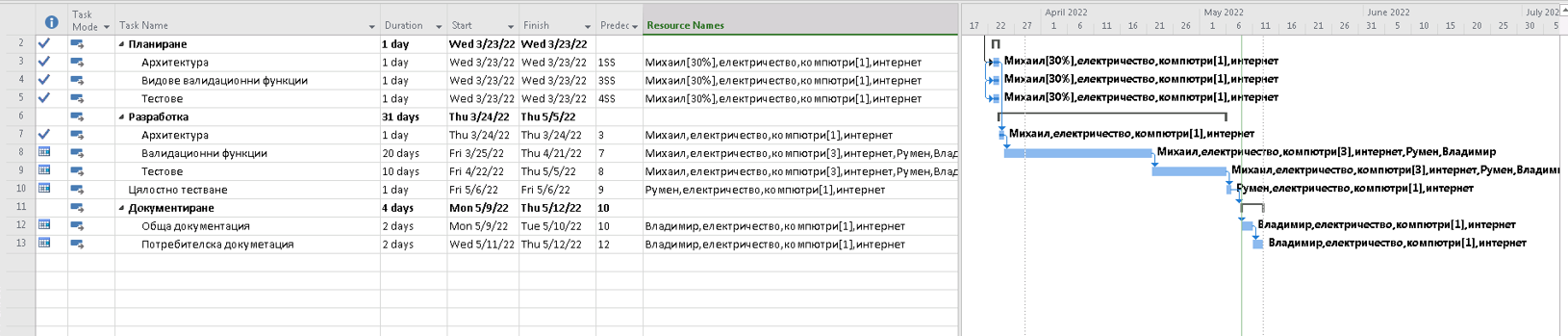
Timeline

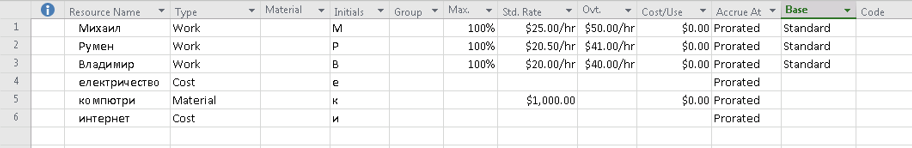


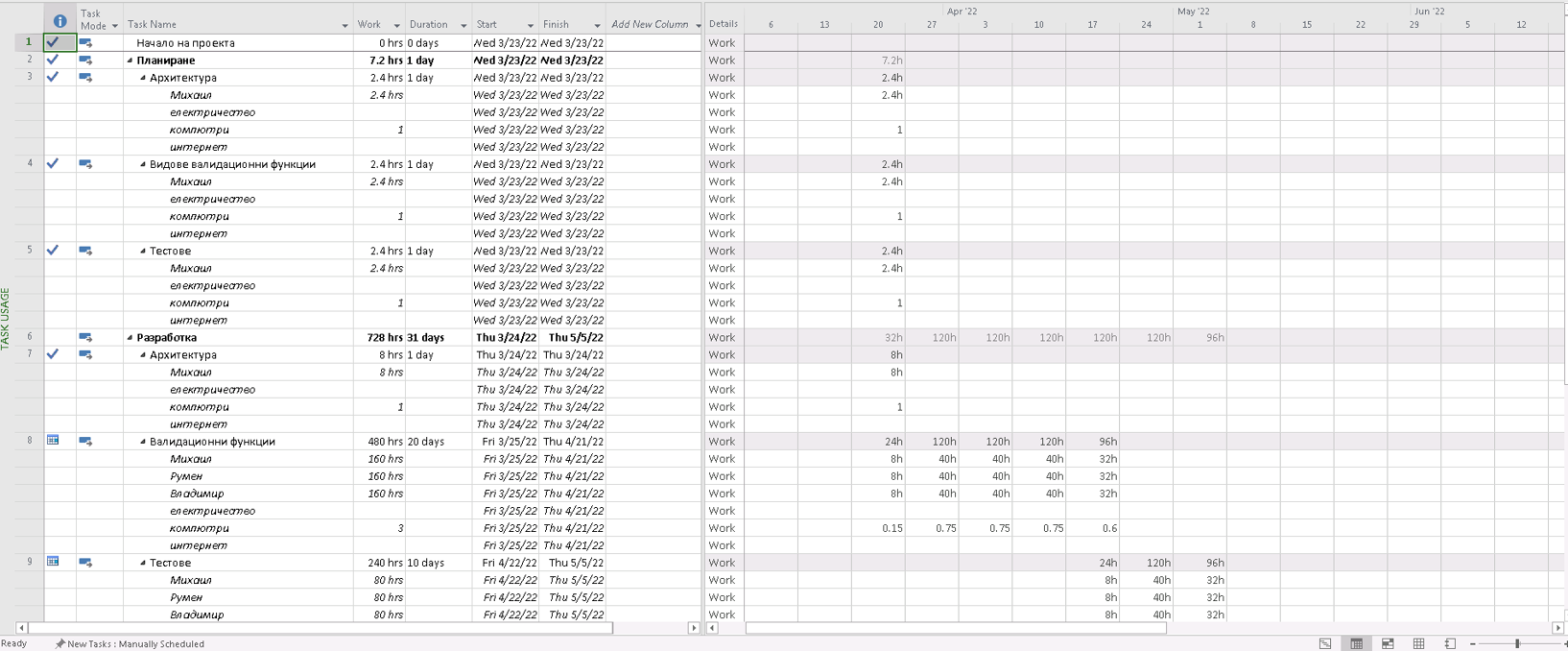


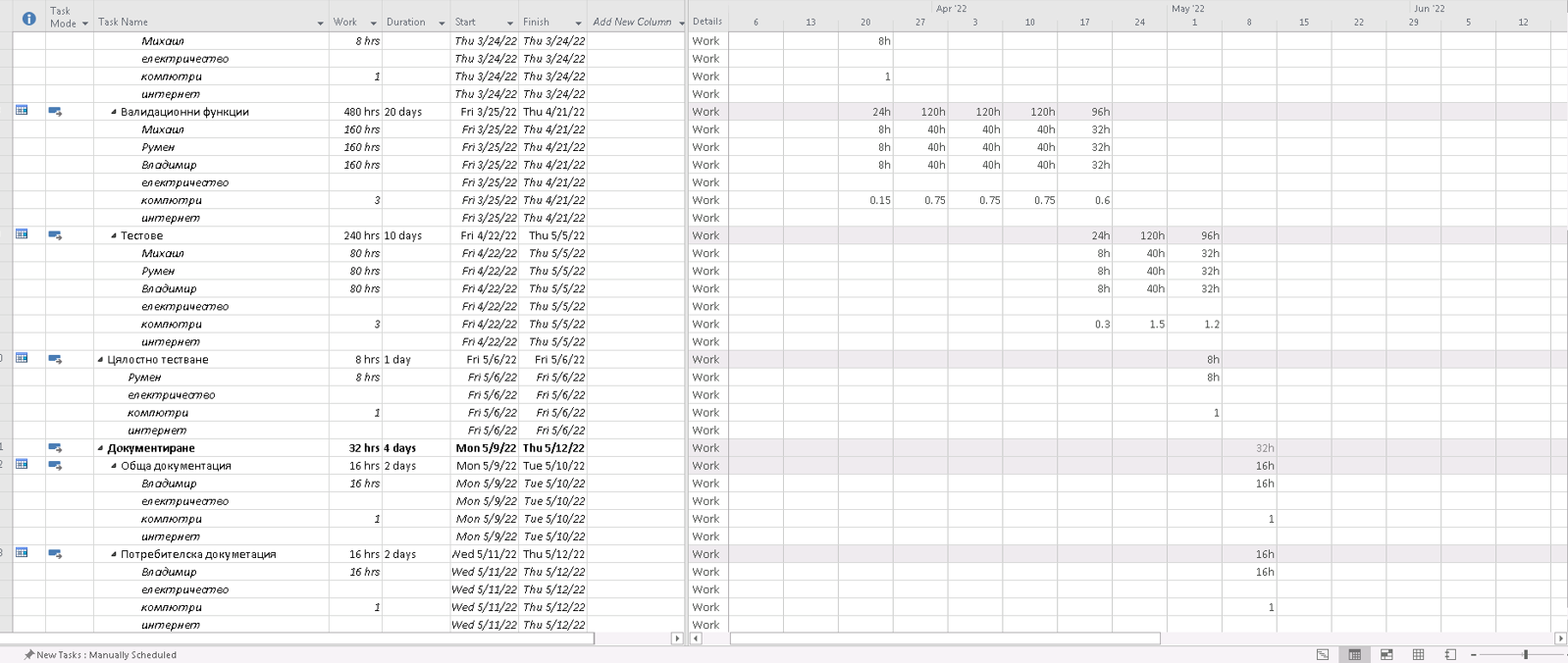
Microsoft Project

Календар

Диаграма на Гант

Ресурсен лист

Проследяване на завършеност на задачите



Изводи

Използването на системите за управление на проекти значително повишава ефиктивността и намалява времето за създаване на проекта, както и улеснява участниците в него. Чрез системата за контрол на версиите — Git, позволява на разработчиците паралелна работа върху различни задачи, едновременно с това позволява проследяването на цялостната история на проекта и напредъка на колегите. За да се постигне успешно завършване на проекта трябва да бъдат изпълнени изискванията на потребителя и влизането в бюджета, който ни е даден, както и заложеното време за разработка.