

**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“**

**Система за продажба и обслужване на автомобили**

**Изготвили:**

Александър Мирославов Живков, Владимир Викторов Андреев, Преслав Златков Златинов

(Спец. СИТ, гр. 4б, Фак. №: 19621703, 19621702)

**ВАРНА**

**2022**

Разработващ екип:

|  |  |  |  |
| --- | --- | --- | --- |
| № | Име, презиме и фамилия | Фак. № | Роля |
| 1. | Александър Мирославов Живков | 19621702 | Ръководител на проекта, Back-End Developer,Тестер |
| 2. | Владимир Викторов Андреев | 19621703 | Front-End Developer, Тестер |
| 3. | Преслав Златков Златинов | 19621702,  19621703 | Разработка на база данни, модели, съставяне на Microsoft Project диаграми. |

Срок за представяне: 02 юни 2022г.

Техническо задание на проекта

Бюджет

Бюджета за проекта е съставен въз основа на всички разходи по анализирането, разработването и документирането на проекта. За инициализация и анализ на функционалните и нефункционалните изисквания са предвидени 6 дни, от които 4 работни. В този период, от ръководителят на проекта, се извършва цялостен детайлен анализ на задачата, инициализацията на проекта в средата за разработване, както и свързване на всички необходими компоненти за разработката, а именно – система за контрол на версиите (Git), необходими драйвери за връзка с базата данни (MySQL), както и избор на архитектура на системата. За проекта е подбрана n-слойна архитектура, чрез която се постига по-голяма независимост между отделните модули на приложението и позволява за по-лесно нанасяне на промени по която и да е част от системата в бъдеще.

За изготвяне на релационна база данни са предвидени 4 дни, 2 от които работни. В този период специалистът по бази данни държи непрестанна връзка с project manager-a с цял да се постигне максимално удовлетворение на поставените изисквания. За осъществяване на слоя за комуникация с базата данни са предвидени 4 работни дни. По средата на този период започва и разработката на потребителския интерфейс от front-end разработчикът, за което са предвидени 5 дни. Паралелно с това започва изготвянето на модели, чрез които се предават и визуализират данните в презентационния слой, за което са отделени 3 дни. Заедно с тези процеси започва и изграждането на бизнес логиката на приложението от back-end разработчикът, за което са предвидени 5 дни. Тези 3 процеса завършват по едно и също време. Преди техният окончателен завършек, постепенно започва процесът по финални настройки на приложението. Той включва цялостен debugging, изчистване на грешки и коментари по кода, оптимизация, както и изготвяне на unit и интеграционни тестове за необходимите модули. За този процес за предвидени общо 3 дни. Преди този процес да завърши, започва и финалната фаза по разработката на приложението, а именно изготвянето на документация, UML диаграми, ръководство за потребителя и т.н.

С оглед на гореизложената информация, общата стойност на проекта възлиза на **11,452$.** Тя включва използването на материални ресурси от екипа (компютри, ток, интернет, IntelliJ Ultimate), както и възнаграждения за периода на разработка.

Предмет на техническото задание

**Функционални изисквания:**

* Софтуерът трябва да позволи на служител на представителство на марка или автокъща да въвежда записи за различни видове превозни средства, да добавя нови такива, както и да редактира стари и да ги изтрива. Целта на софтуера е да улесни ежедневния работен процес на служителите.
* Освен това, тази система трябва да улесни смяната на собствеността на дадено превозно средство.
* Софтуерът поддържа добавяне, редактиране и изтриване на сервизни обслужвания за всяко превозно средство, както и на механици, работещи в даденото представителство.
* Системата трябва да съхранява данни (в БД) за:
* Потребители (user),
* собственици (owners)
* плащания (payments)
* превозни средства (cars)
* сервизни обслужвания (service),
* междинна таблица за обслужвания за коли (car\_service) и
* механици (mechanics).

**Не-функционални изисквания:**

* Използва се унифицирана форма за логване в системата от всички потребители.
* За всяка от таблиците в БД се използва форма за реализиране на стандартните операции: въвеждане, корекция, изтриване и търсене на данни.
* Генериране на необходимите справки: на екран.
* Потребителите могат да нанасят корекции по записите, да добавят нови или да изтриват стари само и единствено ако са логнати в системата.
* За въвеждането на всеки един запис е необходимо данните да отговарят на предварително посочени изисквания.

Всички екрани и техните функционалности са описани подробно в ръководството за потребителя.

Изисквания към програмния продукт

Общи изисквания

Софтуерният продукт трябва да обхваща всички случаи на използване и поведението на системата да е предсказуемо. При този конкретен продукт потребителят няма възможност за въвеждане на данни, които да попречат или да повредят на системата. Всички входни точки са валидирани. Продуктът е десктоп приложение на програмния език Java. Той е използван защото е масов, поддържан и има възможност за пускане на всякакви операционни системи.

За запазване на информация е използвана MySQL база, която е избрана заради високата си производителност, лесното си използване.

Приложението има минимални изисквани към хардуера, на който трябва да бъде инсталирано. Относно софтуера единственото изискване е да има  инсталирана Java, която е нужна, за да се създаде виртуалната машина, на която се пускат всички Java приложения.

Обосновка на избора на система за контрол на версиите

Причина

Една от основните причини за изборът на Git беше фактът, че Git позволява на всеки член от екипа да притежава собствено локално хранилище, в което да работи, по който начин можем едновременно да работим върху един и същи файл. Git позволява лесно презаписване и актуализиране на файлове, както и връщане към предходно състояние на проекта, в случай че се откажем от дадена идея. Git позволява също и създаването на клонове, позволявайки ни да разработваме различни допълнителни идеи, без да възпрепятстваме главния проект.

Трудности при използването

Проследяването на комит, който причинява бъг след голямо сливане отнема голям период от време. Реализирането на функционалността на branch-овете и разрешаването на конфликти между тях е сложно.

Настройки

За да се използва Git се изисква регистрация в Github. Трябва да се изтегли конзолно приложение за реализацията на локално repository, което после се свързва с remote repository в Github, чрез създадените акаунти. Препоръчително е изтеглянето на помощни средства за визуализация на промените по локалното и remote repository-та, като в при реализирането на нашия проект използвахме Github Desktop.

Съпоставка

При използването на Git има възможност при промяна или добавяне на повече файлове промените по основната версия на проекта да се извършат на стъпки, като се commit-ват само по няколко файла наведнъж, като се включи и детайлен коментар за извършените промени. При гит се използват repository-та(едно общо на централизиран сървър, където се съхранява главната версия на проекта и множество локални) и когато се синхронизират данни между repository-та, всяко repository има цялата история на всички промени по проекта, като може лесно да се премине към предходна версия на проекта, за разлика от останалите системи за контрол, където всичко се намира на един централен сървър и мениджърите на проекта раздават роли на хората асоциирани с него, които роли дават ограничен достъп до определени файлове и функции.

Обосновка на избора за система за управление на проекти:

Причина за избора;

Jira позволява лесно управление на разпределението на задачи, чрез прости за използване инструменти за организиране и също дава възможност за проследяване на прогреса. Приложението също така праща известия при възникнали промени - нови или завършени задачи, като улеснява комуникацията между членовете на екипа. Jira дава възможността за лесно и бързо назначаване на задачи на отделните членове на екипи, прикачване на GitHub Repository към проекта, както и изключително интуитивен интерфейс за управление на проекта.

Трудности при използването;

Jira няма функция, която да показва връзките между отделни задачи - коя задача трябва да се реализира, за да може да се реализира друга. Jira не дава възможност за проследяване на време и ресурси, за да се следят по-лесно крайни срокове и вместване в бюджета. Освен това не е предоставена възможност за визуализация на по-малките задачи върху времевата линия на проекта, където се визуализират само основните (Epic) задачи.

Настройки, нужни за работата с него;

За използването на web версията на Jira се изисква регистрация в Jira. След това се определя project manager, който създава нов проект към Jira и добавя хората от екипа в него.

Използвани функционалности.

Проследяване на история на промените към задачи, кой коя задача е поел, кога я е изпълнил, коя изпълнява в момента, колко задачи са изпълнени и колко остават. Определяне на последователности от задачи и групиране на подзадачи в една основна задача (Epic).

Обосновка и разяснения към избраната методология за разработка на софтуера

Този проект е реализиран използвайки Agile методология, по конкретно Kanban. Гъвкавите методологии най-често се употребяват при разработка на софтуер, при който промяната на изискванията и обхвата е силно вероятна. Зададеният проект бе сравнително кратък и с относително прости изисквания. Въпреки това, тези изисквания подлежат на постоянни корекции по време на реализацията, докато не се постигне желаният резултат.

Основното предимство при тази методология е, че процесите не протичат линейно и последователно. Това означава, че всеки от етапите в процеса на разработка е относително независим от другите. По този начин е възможна паралелна разработка на отделните модули на приложението, чрез което се постига по-бърза реализация на системата. Друго предимство на Kanban модела е, че project manager-a може да назначи конкретни задачи на отделните разработчици и така да разпредели сроковете на започване и изпълнение, че да постигне доста добра оптимизация и качество на разработката. При стриктно спазване на методологията връщане към предишна фаза за преправяне на продукта, поради промяна на изискванията, се допуска.

Предимства:

* По-добра видимост
* Повишена гъвкавост спрямо изискванията
* Подобрена продуктивност на екипа
* По-кратки срокове за разработка

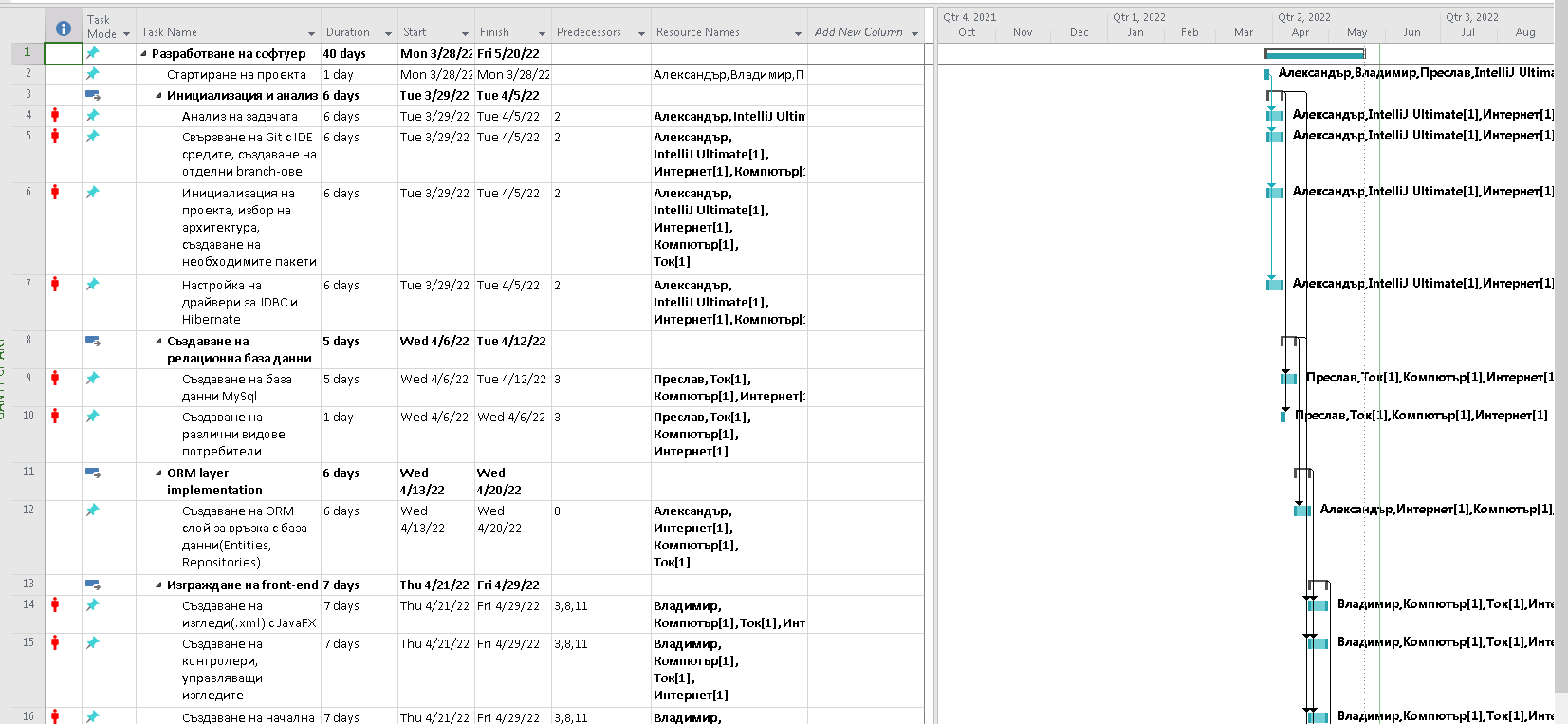
Недостатъци:

* Отделните фази нямат определен времеви диапазон за изпълнение
* Понякога „дъската“ може да стане твърде сложна
* „Дъската“ трябва постоянно да е актуална.

Съпоставка с водопадният модел:

Водопадният модел е известен класически, линеен модел, който следва последователни етапи, докато спираловидният модел следва еволюционния път.

|  |  |
| --- | --- |
| **Водопаден модел** | **Agile методологии** |
| 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 | Устойчив на променливи изисквания |
| 5.Много малка гъвкавост | Висока гъвкавост |
| 6.Сравнително евтин модел | Ценово ефективен модел |
| 7.Необходимо е задълбочено разбиране на изискванията от самото начало | В зависимост от конкретният гъвкав модел, промените могат сравнително лесно да се имплементират |
| 8.Произвежда единичен краен продукт в края на процеса | В зависимост от модела се произвежда готов продукт, отговарящ на част от изискванията. |

Разделяне на задачи

Определяне на срокове

Начална дата: 01/04/2022

Анализ на изискванията и инициализация : 01/04/2022 – 07/04/2022

Създаване на релационна база данни: 07/04/2022 – 10/04/2022

Изграждане на слой за връзка с база данни: 11/04/2022 – 14/04/2022

Изграждане на front-end: 13/04/2022 – 18/04/2022

Изграждане на модели: 13/04/2020 – 16/04/2022

Реализация на бизнес логика: 13/04/2022 – 18/04/2022

Финални настройки: 18/04/2022 – 21/04/2022

Изготвяне на документация: 21/04/2022 – 25/04/2022

Timeline

Description automatically generated

Програмни средства за реализацията

Език за програмиране: Java

Платформа за визуализация: JavaFX

База данни: MySQL

IDE: IntelliJ IDEA Ultimate Edition

Допълнителни помощни средства: Scene Builder, MySQL Workbench

Жалонни точки

|  |  |
| --- | --- |
| Събитие | Опсание |
| 0 | Начало |
| 1 | Задачата е анализирана |
| 2 | Проектът е инициализиран |
| 3 | Базата данни е създадена |
| 4 | Слоят за комуникация с базата данни е създаден |
| 5 | Потребителските изгледа са създадени |
| 6 | Контролерите са създадени |
| 7 | Моделите са изградени |
| 8 | Бизнес логиката е имплементирана |
| 9 | Логърът е реализиран |
| 10 | Приложението е настроено |
| 11 | Unit тестовете са създадени |
| 12 | Интеграционните тестове са създадени и изпълнени |
| 13 | Документацията е изготвена |

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Microsoft Project

Календар

Диаграма на Гант

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

Ресурсен лист

Calendar

Description automatically generated with medium confidence

Диаграма на свършената работа

Chart, bar chart

Description automatically generated

Диаграма на бюджет

Chart, pie chart

Description automatically generated

Бизнес правила:

Потребителят няма достъп до приложението без да има профил и ако не се е логнал.

Всяка кола може да има няколко обслужвания.

Всеки собственик може да има няколко коли.

Една кола може да има само един собственик и начин на плащане.

Един механик може да обслужва много на брой коли.

Обекти и техните атрибути

Car:

* Id\_car – уникален идентификатор на превозното средство
* manufacturer – производител на превозното средство
* model - модел на превозното средство
* engine – двигател
* transmission – скоростна кутия
* drive\_type – тип задвижване (FWD/RWD/AWD/4WD)
* vin – номер на рама
* price – цена в лева
* date\_of\_first\_reg – дата на първа регистрация
* mileage – пробег в километри
* type – вид МПС (лека кола, мотоциклет, автобус и т.н.)
* discount – отстъпка от цената в проценти

Mechanic:

* id\_mechanic – уникален идентификатор на механика
* mechanic\_name – име на механика

Payment:

* id\_payment – уникален идентификатор на плащането
* payment\_type – вид плащане

Owner:

* id\_owner – уникален идентификатор на собственика
* owner\_name – име на собственика
* number\_of\_cars\_bought – брой закупени коли

Service:

* id\_service – уникален идентификатор на обслужването
* service\_name – име на обслужването
* service\_type – вид обслужване

CarService:

* id\_car\_service – уникален идентификатор на записа за обслужване
* id\_car – уникален идентификатор на колата
* id\_service – уникален идентификатор на обслужването
* id\_mechanic – уникален идентификатор на механика
* price\_service – цена на конкретното обслужване

User:

* id\_user – уникален идентификатор на потребителя
* user\_username – потребителско име на потребителя
* user\_password – парола на потребителя
* isAdmin – флаг, показващ дали потребителя е администратор

Диаграми:

ER Диаграма:

Graphical user interface

Description automatically generated with medium confidence

Use case диаграма:

Diagram

Description automatically generated

Изводи

Използването на системи за управление на проекта значително повишава производителността на участниците в него. Системата за контрол на версиите Git дава възможност за независима разработка и лесно синхронизиране на извършената работа, лесно придвижване между версии на приложението и създаване на разклонения за изработване на допълнителни функционалности. За достигане на успешно завършен проект, удовлетворяващ потребителските изисквания, който влиза в бюджета и заложеното време за изпълнение, е нужно да се отдели достатъчно време на всеки един от етапите и допълнително внимание на етапите по планиране и анализ на отделните детайли от проекта.