|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ИВАНОВСКОЙ ОБЛАСТИ**  Областное государственное бюджетное профессиональное образовательное учреждение  «Ивановский промышленно-экономический колледж» | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| **КУРСОВОЙ ПРОЕКТ** | | | | | | |
|  | | | | | | |
| **Разработка системы контроля версий** | | | | | | |
|  | | | | | | |
| **ИВПЭК. 09.02.07.17** | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | **Специальность:** | | | **09.02.07 Информационные системы и программирование,**  **базовая подготовка** | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| Руководитель курсового проекта | | | |  | | В.И. Кукушкин |
| Выполнил обучающийся группы 307а | | | |  | | Д.Д. Михайлов |
|  | | | |  | |  |
|  | | | | | | |
|  | | | | | | |
| Курсовой проект выполнен и защищен с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
|  | | | | | | |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | | | | | | |
|  |  | | | | | |
|  | | |  | | |  |
|  | | |  | | |  |
|  | Иваново 2024 | | | | |  |

Оглавление

[Введение 3](#_Toc193360711)

[1 Концептуальное проектирование 5](#_Toc193360712)

[1.1 Теоретический вопрос 5](#_Toc193360713)

[1.2 Техническое задание 6](#_Toc193360714)

[2 Техно–рабочий проект 7](#_Toc193360715)

[2.1 Моделирование с использованием UML-диаграмм 7](#_Toc193360716)

[2.2 Обоснование выбора средств разработки 8](#_Toc193360717)

# Введение

В современном мире существует такая проблема: неорганизованная работа над проектами, особенно когда над ними трудятся несколько человек одновременно. Представьте себе ситуацию: программисты вносят изменения в один и тот же файл, дизайнеры перерабатывают макеты, а копирайтеры редактируют текст. Как отследить, кто что изменил, вернуться к предыдущей версии, если что-то пошло не так, и не запутаться в бесчисленных копиях документов? Для решения этой проблемы и существуют системы контроля версий, и, как их воплощение – приложения для контроля версий. Без них современная разработка программного обеспечения и многих других видов совместной работы была бы просто немыслима. Эти приложения стали неотъемлемой частью рабочего процесса, обеспечивая прозрачность, эффективность и надежность при создании любого проекта.

Актуальность темы разработки системы контроля версий в современном мире трудно переоценить. В условиях постоянно растущей сложности программных проектов и увеличения числа разработчиков, работающих над ними, эффективное управление изменениями становится критически важным. Без надежной системы контроля версий, совместная работа над проектами быстро превращается в хаос, приводя к потере времени, ошибкам и сложностям в отслеживании изменений.

Системы контроля версий позволяют не только отслеживать историю изменений, но и эффективно управлять ветвлением кода, объединять результаты работы разных разработчиков и откатываться к предыдущим версиям в случае возникновения проблем. Однако, несмотря на распространенность существующих решений, потребность в новых, более гибких, удобных и производительных системах контроля версий сохраняется.

Существуют различные сцена того, развитие облачных технологий и тенденция к совместной работе над проектами из разных точек мира, делают вопрос эффективного контроля версий еще более важным. Поэтому разработка современной, надежной и удобной системы контроля версий не только не теряет актуальности, но и становится все более значимой для успешной реализации любых проектов в сфере IT и за ее пределами.

Цель проекта — разработка надежной и удобной системы контроля версий, которая позволит эффективно управлять изменениями в проектах любого масштаба и сложности, отслеживание истории изменений, простое разрешение конфликтов и возможность возврата к предыдущим версиям. Система должна обладать интуитивно понятным интерфейсом для пользователей с различным уровнем технической подготовки.

В рамках проекта по разработке системы контроля версий, нам предстоит реализовать базовые функции отслеживания изменений в файлах и каталогах, что позволит фиксировать (коммитить) эти изменения, сохраняя полную историю проекта. Также необходима функциональность отката к предыдущим версиям, что обеспечит безопасность и позволит вернуться к рабочему состоянию в случае необходимости. Для удобства пользователей, к каждому коммиту можно будет добавлять комментарии, объясняющие суть внесенных изменений.

Кроме того, мы разработаем удобный и интуитивно понятный пользовательский интерфейс для управления репозиторием. Важным аспектом будет наглядное отображение истории изменений, чтобы пользователям было легко ориентироваться в процессе разработки. Система будет доступна через командную строку (CLI) для опытных пользователей и через графический интерфейс (GUI) для более комфортной работы. Мы также позаботимся о кроссплатформенности приложения, чтобы система могла работать на различных операционных системах.

# 1 Концептуальное проектирование

## 1.1 Теоретический вопрос

Разработка систем контроля версий (СКВ) крайне актуальна в современном мире из-за роста сложности проектов и распределённой работы команд. СКВ обеспечивают координацию изменений, сохранность данных, отслеживание истории разработки и упрощают совместную работу, что критически важно для успешного создания программного обеспечения и цифровых продуктов.

Задача разработки системы контроля версий (СКВ) с максимально упрощенным интерфейсом, ориентированным на удобство пользователя, особенно начинающих программистов, заключается в том, чтобы сделать процесс структурированной разработки программного обеспечения доступным для каждого.

Это означает, что система должна быть спроектирована таким образом, чтобы минимизировать порог вхождения для новичков, не требуя глубоких технических знаний или опыта работы с традиционными СКВ. Интерфейс должен быть интуитивно понятным, наглядным и предлагать простые, логичные рабочие процессы.

Основная цель – дать возможность даже тем, кто только начинает свой путь в программировании, эффективно управлять своими проектами, отслеживать изменения, экспериментировать с кодом и сотрудничать с другими разработчиками, не тратя время и усилия на освоение сложных инструментов.

В результате, такая СКВ должна стать мощным инструментом для обучения, развития и повышения продуктивности как начинающих, так и опытных программистов, способствуя созданию качественного и хорошо структурированного программного обеспечения.

## 1.2 Техническое задание

Цель: Создать простую СКВ для начинающих программистов.

Функции:

* Создание локального репозитория.
* Добавление/ статус файлов.
* Коммит с комментарием.
* История изменений (автор, дата, комментарий).
* Восстановление версий.

3. Интерфейс:

* Простой и понятный.
* CLI

4. Требования:

* Быстрая работа.
* Надежная сохранность данных.

5. Этапы:

* Проектирование.
* Разработка ядра (функции СКВ).
* Разработка UI.
* Тестирование.
* Документация.

6. Критерии приемки:

* Соответствие функциям и требованиям.
* Успешные тесты.
* Понятная документация.
* Простота использования.

# 2 Техно–рабочий проект

## 2.1 Моделирование с использованием UML-диаграмм



Рисунок 1 – UML диаграмма в программе «CSV»

Моделирование с использованием UML – диаграммы в данном проекте выглядит следующим образом. Пользователь имеет возможность выполнять команды в CLI интерфейсе «CSV». Создание репозитория – команда mkrepo, создание базы данных репозитория – mkdata, просмотр файлов в репозитории – dir, сохранение файлов – save, загрузка файлов – load, вывод информации о файлах который пользователь сохранил – info, очистка окна консоли – cls. Так же пользователь может вызвать справку отображающую имеющиеся команды с помощью – help.

Все вышеописанные команды понятны и полностью соответствуют действиям которые они совершают, поэтому пользователю будет комфортно.

## 2.2 Обоснование выбора средств разработки

Для разработки системы контроля версий был выбран язык программирования C# и платформа .NET, что обусловлено следующими факторами:

**Производительность и надежность:** C# является мощным и компилируемым языком, обеспечивающим высокую производительность и надежность разрабатываемого приложения. Платформа .NET предоставляет широкие возможности для управления памятью и ресурсами, что особенно важно для системы контроля версий, работающей с большим количеством файлов и изменений.

**Развитая экосистема и библиотеки:** .NET предлагает богатую экосистему библиотек и инструментов, упрощающих разработку сложных приложений. Например, для работы с файловой системой, базой данных и пользовательским интерфейсом можно использовать готовые решения, что значительно сокращает время разработки.

**Поддержка работы с файлами и потоками:** Системы контроля версий интенсивно работают с файловой системой, и C# предоставляет широкие возможности для эффективного управления файлами, директориями и потоками данных. Это позволяет реализовать необходимые функции для отслеживания изменений, коммитов и восстановления версий.

**Возможность создания CLI (Command Line Interface):** C# позволяет создавать консольные приложения, которые можно использовать для разработки интерфейса командной строки (CLI). CLI предоставляет опытных пользователям более гибкий и мощный способ взаимодействия с системой контроля версий.

**Опыт с языком и платформой:** Наличие опыта работы с C# и платформой .NET является важным фактором при выборе средств разработки. Знание языка и инструментов позволяет эффективно решать поставленные задачи и создавать качественное программное обеспечение.

В заключении: для разработки системы контроля версий выбраны C# и .NET, обеспечивающие производительность и удобство разработки GUI/CLI, благодаря богатой экосистеме и личному опыту.

## 2.3 Разработка макета приложения

Перед началом разработки нужно определиться как будет выглядеть терминал, его команды, а так же цвет текста. Это важно для пользователя, так как команды должны быть легко запоминающемся, а цвета в то же время различными, и не кислотными для удобства глаз.

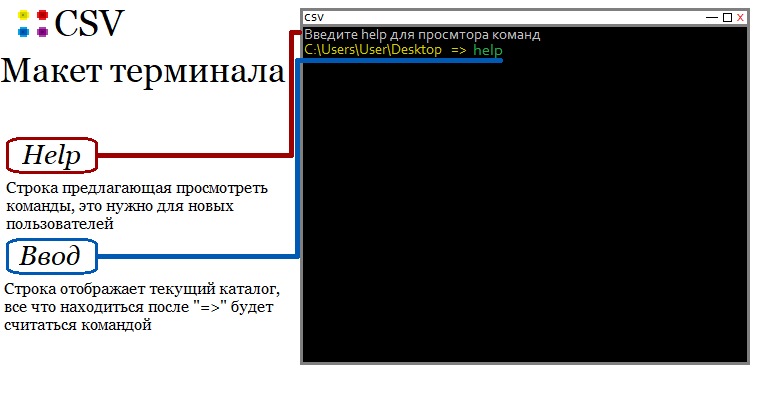


Рисунок 2 – Макет терминала

В качестве основных цветов я выбрал темно – желтый, темно – зеленый и серый. Темно – желтый цвет я выбрал для «Вводной» строки, она не напрягает глаза и в то же время пользователю будет всегда понятно где он пишет команды. Темно – зеленый цвет я поставил для самих команд и остального текст который выводиться в терминале, он классический для терминала, отличается от «Вводной»

строки и выглядит гармонично с остальными цветами. Серый цвет я выбрал для каких либо подсказок пользователю который открыл приложение первый раз, для того что бы при использовании в следующие разы он не обращал на них внимания.

# 3 Рабочая документация

## 3.1 Описание разработки приложения

Приложение системы контроля версий должна быть с приятным и интуитивно понятым интерфейсом, и легкими командами, с чего мы и начнем, то есть с разработки тела нашего терминала

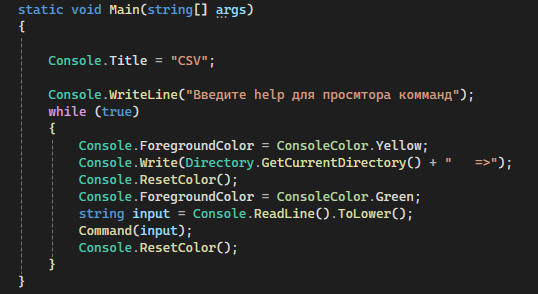


Рисунок 3 – Тело терминала

Тут все просто, бесконечный цикл в котором будут приниматься команды с помощью одной переменной «input», которая будет передаваться в другой метод с командами.

Далее нам нужно сделать метод который будет отвечать за обработку команд,

так же в этом методе обязательно нужно сделать систему обработки ошибок и сделать его высокопроизводительным, потому что любому пользователю не понравиться ожидать хотя бы по 5 секунд на обработку каждой команды.

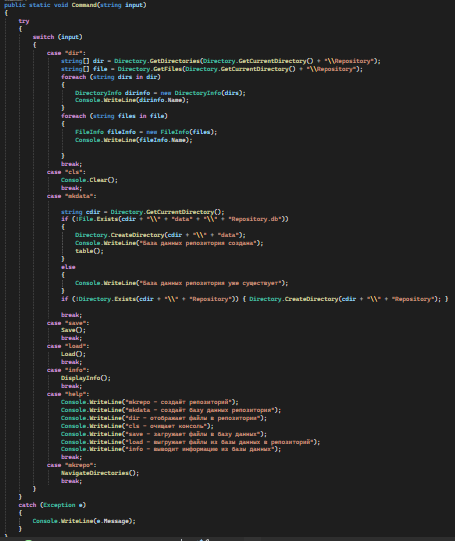


Рисунок 4 – Обработчик команд

В этом методе я использовал связку switch-case для лучшей производительности, и связку try-catch для обработки исключений.

Так же в этом методе я реализовал простые команды, которые на мой взгляд не нуждаются в отдельном методе, а именно команды help, cls и dir.

Далее нам нужно сделать сами команды приложения которые требуются методы.

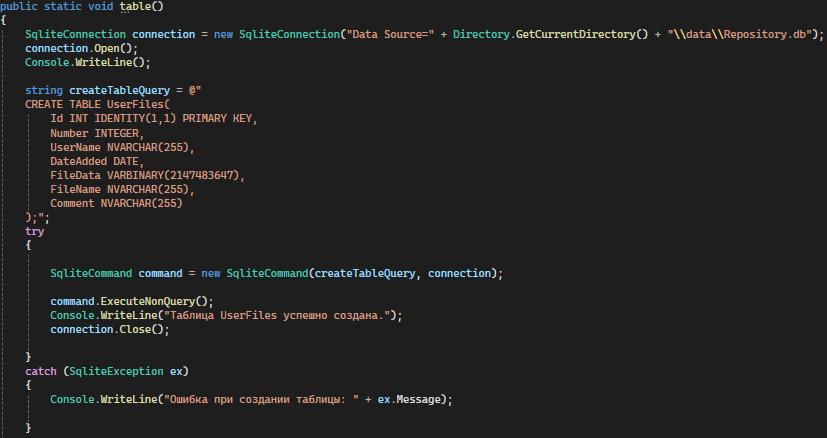


Рисунок 5 – Создание базы данных

Это метод создает базу данных репозитория в которую будут записываться файлы, комментарии и другая информация. Далее нам нужно сделать метод который создает сам репозиторий.



Рисунок 6 – Создание репозитория

Этот метод позволяет пользователю очень просто создать репозиторий без большого количества команд, потому что здесь реализовано управление с помощью стрелочек, так что любой пользователь разберется как его создать.

Далее нам нужно создать метод сохранения файлов репозитория в базу данных



Рисунок 7 – Сохранение репозитория

Этот метод сохраняет все файлы которые находяться в репозитории в базу данных. А именно сначала метод копирует все файлы zip архив который находиться во временной папке, потом он записывает этот архив в базу данных как байт код

Далее нам нужно сделать метод выгрузки файлов из репозитория