|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ИВАНОВСКОЙ ОБЛАСТИ  Областное государственное бюджетное профессиональное образовательное учреждение  «Ивановский промышленно-экономический колледж» | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | |  | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
| КУРСОВОЙ ПРОЕКТ | | | | | | |
|  | | | | | | |
| Разработка системы контроля версий | | | | | | |
|  | | | | | | |
| ИВПЭК. 09.02.07.16 | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | | | | | |
|  | | 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\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | | | |
|  | | | | | | |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г. | | | | | | |
|  |  | | | | | |
|  | | |  | | |  |
|  | | |  | | |  |
|  | Иваново 2025 | | | | |  |

Содержание

[Введение 3](#_Toc193982232)

[1 Концептуальное проектирование 5](#_Toc193982233)

[1.1 Теоретический вопрос 5](#_Toc193982234)

[1.2 Техническое задание 6](#_Toc193982235)

[2 Техно–рабочий проект 7](#_Toc193982236)

[2.1 Моделирование с использованием UML-диаграмм 7](#_Toc193982237)

[2.2 Обоснование выбора средств разработки 8](#_Toc193982238)

[2.3 Разработка макета приложения 9](#_Toc193982239)

[3 Рабочая документация 12](#_Toc193982240)

[3.1 Описание разработки приложения 12](#_Toc193982241)

[3.2 Руководство пользователя 20](#_Toc193982242)

# 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 (Рисунок 1). 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 тем как приступить к непосредственной разработке, крайне важно определить, каким образом будет выглядеть терминал (Рисунок 2), какие команды будут доступны пользователю, и какая цветовая схема будет использоваться для отображения текста. Эти аспекты играют ключевую роль в создании положительного пользовательского опыта. Команды должны быть интуитивно понятными и легко запоминающимися, чтобы пользователь мог быстро освоить систему и эффективно выполнять необходимые операции. Цветовая схема должна быть тщательно продумана: с одной стороны, необходимо использовать различные цвета для выделения важной информации и улучшения читаемости, с другой стороны, цвета не должны быть слишком яркими или “кислотными”, чтобы не вызывать усталость глаз и дискомфорт при длительной работе с терминалом. Правильно подобранный дизайн терминала и команд – это залог удобства, эффективности и удовлетворенности пользователя от работы с системой контроля версий.

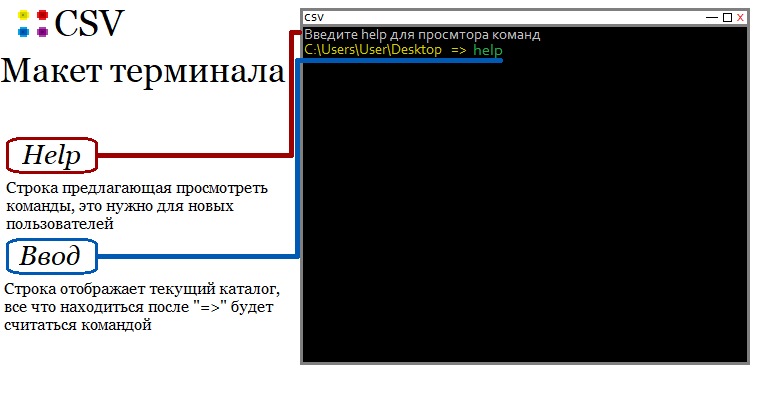


Рисунок 2 – Макет терминала

Для создания комфортной и эффективной визуальной среды в терминале системы контроля версий была тщательно подобрана цветовая палитра. В качестве основных цветов выбраны темно-желтый, темно-зеленый и серый, каждый из которых выполняет определенную функцию. Темно-желтый цвет используется для “вводной” строки, в которой пользователь набирает команды. Этот оттенок не перегружает зрение и позволяет четко идентифицировать место ввода, не отвлекая от остальной информации. Темно-зеленый цвет применяется для отображения самих команд и остального текста, выводимого в терминале. Этот классический цвет терминала обеспечивает хорошую читаемость, контрастирует с вводной строкой и гармонично сочетается с другими элементами интерфейса. Серый цвет используется для отображения подсказок и справочной информации, предназначенной в основном для новых пользователей. Это позволяет не перегружать основной интерфейс и не отвлекать опытных пользователей от работы. Такое сочетание цветов обеспечивает визуальный комфорт, улучшает восприятие информации и способствует более эффективной работе с системой контроля версий.

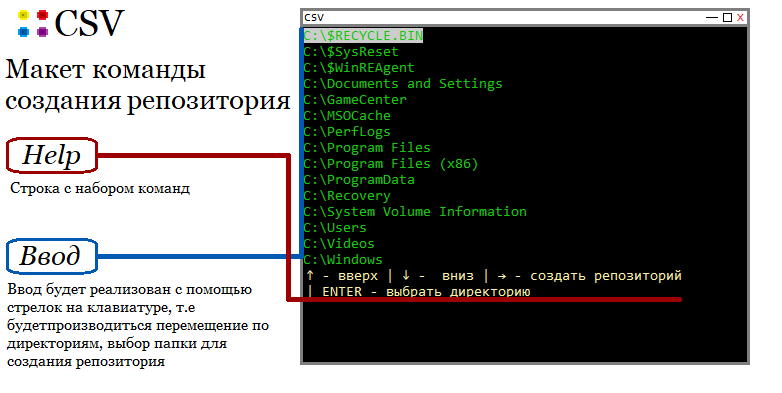


Рисунок 3 – Макет команды создания репозитория

Особое внимание в процессе разработки системы контроля версий уделяется созданию репозитория (Рисунок 3),, и это не случайно. Возникла идея существенно упростить этот процесс, отказавшись от сложных команд с указанием длинных путей и заменив их интуитивно понятным интерфейсом. Такой подход позволит значительно улучшить пользовательский опыт, особенно для начинающих разработчиков, которым может быть сложно освоить синтаксис и правила указания путей. Простой и понятный интерфейс сделает процесс создания репозитория более быстрым, удобным и приятным, что, в свою очередь, повысит привлекательность системы контроля версий для широкого круга пользователей.

# 3 Рабочая документация

3.1 Описание разработки приложения

Одним из ключевых приоритетов при разработке системы контроля версий является создание приложения с приятным, интуитивно понятным интерфейсом и простыми, легко запоминающимися командами. Это позволит как опытным разработчикам, так и новичкам эффективно использовать все возможности системы без необходимости тратить время на изучение сложной документации. Именно поэтому, разработка начинается с создания “тела” терминала (Рисунок 4), то есть основы пользовательского интерфейса. От того, насколько удобным и понятным будет интерфейс, зависит общее впечатление от системы и ее востребованность. Продуманная структура, четкие команды и приятный внешний вид – залог успешного и популярного инструмента контроля версий.

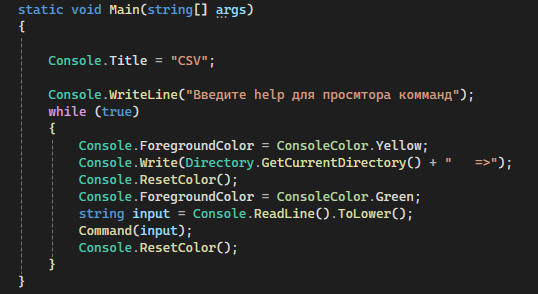


Рисунок 4 – Тело терминала

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

Следующим важным шагом в разработке системы контроля версий является создание метода, который будет отвечать за обработку вводимых пользователем команд (Рисунок 5). Этот метод должен не только корректно интерпретировать команды, но и обеспечивать высокую скорость работы, чтобы пользователь не сталкивался с неприемлемыми задержками. Для достижения этой цели, необходимо тщательно продумать архитектуру метода, оптимизировать алгоритмы обработки команд и использовать эффективные структуры данных. Особое внимание следует уделить системе обработки ошибок: метод должен уметь корректно обрабатывать неверные команды, отсутствие необходимых аргументов, проблемы с доступом к файлам и другие потенциальные ошибки, предоставляя пользователю информативные сообщения о причинах проблемы. В целом, этот метод должен быть высокопроизводительным,

надежным и удобным в использовании, чтобы обеспечить комфортную работу пользователя с системой контроля версий.

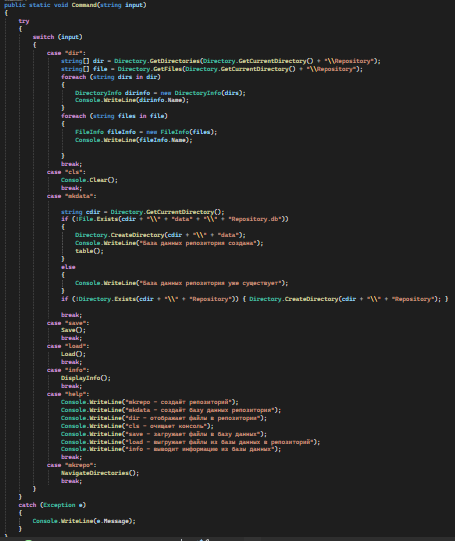


Рисунок 5 – Обработчик команд

Для обеспечения эффективной работы системы контроля версий и надежной обработки возникающих ситуаций, в данном методе реализован ряд ключевых технических решений. Во-первых, для оптимизации производительности и быстрого определения требуемого действия используется конструкция switch-case. Это позволяет эффективно обрабатывать различные команды пользователя, избегая лишних проверок и повышая скорость выполнения. Во-вторых, для предотвращения сбоев и корректной обработки возможных ошибок применяется конструкция try-catch. Она позволяет перехватывать исключения, возникающие в процессе выполнения кода, и обрабатывать их, предотвращая аварийное завершение программы и предоставляя пользователю информативные сообщения об ошибках. Кроме того, в данном методе реализованы простые, но важные команды, такие как help (вывод справки), cls (очистка экрана) и dir (просмотр содержимого директории). Поскольку логика этих команд относительно проста и не требует сложной обработки, было принято решение реализовать их непосредственно в этом методе, избегая создания отдельных функций и упрощая структуру кода. Такое сочетание подходов позволяет создать эффективный, надежный и удобный в использовании компонент системы контроля версий.

Далее нам нужно сделать сами команды приложения, которые требуются методы.

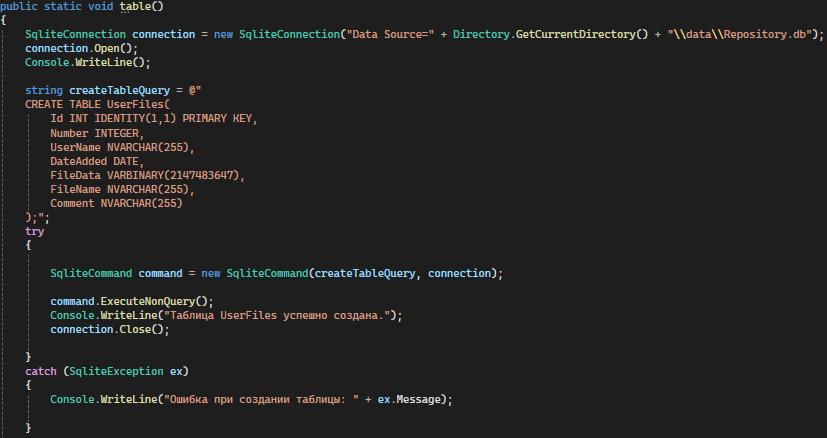


Рисунок 6 – Создание базы данных

Для эффективного хранения и управления всеми данными репозитория, такими как файлы, история изменений, комментарии и метаданные, реализован метод создания базы данных репозитория (Рисунок 6). Этот метод обеспечивает создание структурированного хранилища, в котором будет организована вся необходимая информация о проекте. База данных позволит не только сохранять данные в упорядоченном виде, но и быстро извлекать их, обеспечивая эффективную работу системы контроля версий. Правильно спроектированная и созданная база данных является основой для надежного и функционального репозитория, позволяющего эффективно управлять версиями файлов и отслеживать изменения в проекте.

Далее нам нужно сделать метод, который создает сам репозиторий.



Рисунок 7 – Создание репозитория

Для упрощения процесса создания репозитория для пользователей, особенно для тех, кто только начинает работать с системами контроля версий, разработан интуитивно понятный метод, минимизирующий необходимость использования сложных команд (Рисунок 7). Вместо этого, реализовано управление с помощью стрелочек на клавиатуре, что позволяет пользователю шаг за шагом пройти через процесс создания репозитория, выбирая необходимые параметры и подтверждая свои действия. Благодаря такому подходу, даже неопытный пользователь сможет легко создать репозиторий, не тратя время на изучение сложных команд и их синтаксиса. Интерфейс с управлением стрелочками делает процесс более наглядным и дружелюбным, значительно снижая порог вхождения в систему контроля версий.

Далее нам нужно создать метод сохранения файлов репозитория в базу данных (Рисунок 8)



Рисунок 8 – Сохранение репозитория

Этот метод сохраняет все файлы, которые находятся в репозитории в базу данных.  Процесс начинается с создания zip-архива, содержащего все файлы репозитория, включая рабочую директорию и метаданные. Этот архив временно размещается во временной папке. Далее, архив преобразуется в массив байтов, который затем записывается в специально предназначенное поле в базе данных. Это поле предназначено для хранения бинарного представления zip-архива. Вместе с массивом байтов в базе данных сохраняются метаданные, такие как дата создания, имя пользователя, название репоизтория для проверки целостности, и идентификатор соответствующего коммита. После успешной записи в базу данных временный zip-файл удаляется. Такой подход позволяет создать копию репозитория и легко восстановить его из базы данных в случае необходимости.

Далее нам нужно сделать метод выгрузки файлов из репозитория который будет доставать файлы из базы данных в временную папку и распаковывать из в репозитории (Рисунок 9).

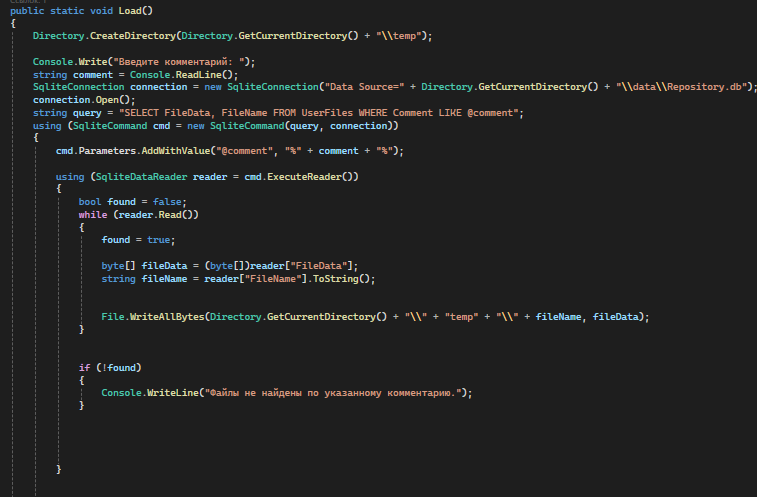


Рисунок 9 – Загрузка файлов в репозиторий

Далее нам нужно сделать метод просмотра записей в базе данных

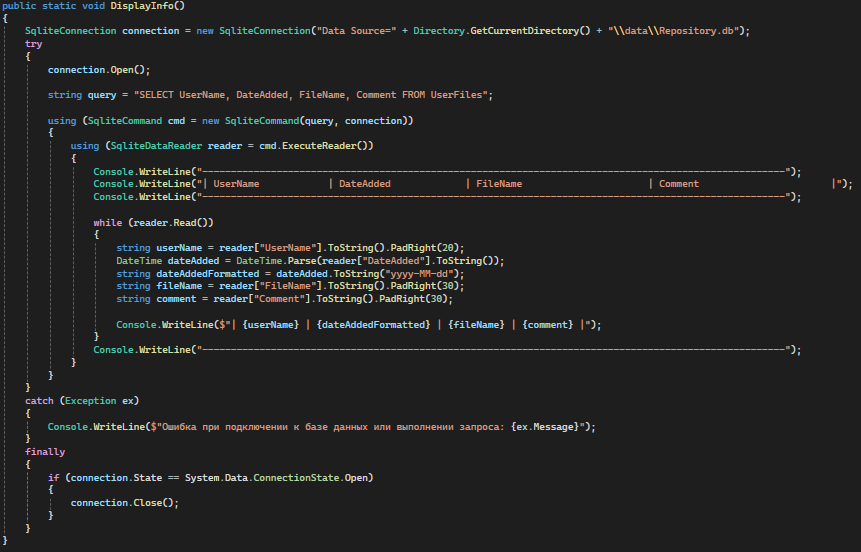


Рисунок 10 – Просмотр записей в базе данных

Для того чтобы пользователь мог эффективно управлять своими данными и ориентироваться в истории изменений репозитория, реализован метод просмотра данных, хранящихся в базе данных (Рисунок 10). Этот метод предоставляет пользователю возможность просматривать информацию о сделанных им коммитах, обеспечивая понимание структуры и содержания хранящихся данных. Метод позволяет отображать, например, дату и время создания резервной копии, благодаря этому пользователь может легко найти нужный коммит, просмотреть метаданные и убедиться в целостности сохраненных данных. Такой функционал особенно важен для сложных проектов, где необходимо быстро и точно находить определенные версии файлов или резервные копии для восстановления или сравнения изменений.

3.2 Руководство пользователя

После успешной установки системы контроля версий (CSV) на ваш компьютер, следующим шагом является запуск исполняемого файла программы, обычно имеющего имя “csv.exe”. Двойной щелчок по этому файлу приведет к открытию окна терминала, которое и является основным интерфейсом для взаимодействия с программой CSV (Рисунок 11). В этом терминале вы сможете вводить команды, управлять репозиториями и выполнять другие необходимые операции, связанные с контролем версий ваших проектов.

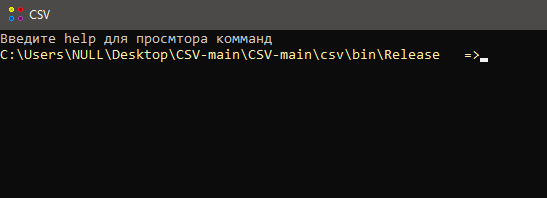


Рисунок 11 – Открытие терминала

После запуска терминала системы контроля версий (CSV) вам становятся доступны следующие основные команды:

* help - выводит на экран подробную справку по доступным командам и их синтаксису, помогая вам освоить работу с программой;
* mkrepo - создает новый репозиторий, который будет служить хранилищем для ваших файлов и истории изменений;
* mkdata - создает базу данных, связанную с репозиторием, для хранения метаданных, истории коммитов и другой служебной информации;
* dir - отображает список файлов и каталогов, находящихся в текущем репозитории, позволяя вам быстро увидеть структуру вашего проекта;
* cls - очищает экран терминала, удаляя предыдущие команды и вывод, для более удобной работы и концентрации на текущих задачах;
* save - загружает файлы из репозитория в базу данных, создавая резервную копию вашего проекта на определенный момент времени;
* load - восстанавливает файлы из базы данных в репозиторий, позволяя вам вернуться к предыдущим версиям или восстановить проект после сбоя;
* info - выводит подробную информацию о сохраненных в базе данных резервных копиях и связанных с ними коммитах.

Прежде чем приступить к созданию репозитория для вашего проекта, необходимо сначала подготовить базу данных, в которой будут храниться все ваши файлы, история изменений, метаданные и прочая важная информация. Это можно сделать с помощью команды mkdata (Рисунок 12). Выполнение этой команды создаст необходимую базу данных, которая затем будет связана с вашим репозиторием, обеспечивая надежное и структурированное хранение ваших данных. Без предварительно созданной базы данных, создание репозитория будет невозможно.

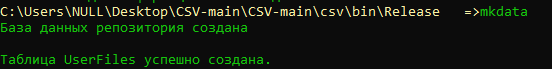


Рисунок 12 – Команда mkdata

После успешного создания базы данных, можно перейти к созданию самого репозитория. Для этого используется команда mkrepo (Рисунок 13). После ввода этой команды терминал переключится в специальный режим создания репозитория, в котором стандартные команды CSV будут временно недоступны. Навигация по файловой системе для выбора места расположения репозитория осуществляется с помощью стрелок “вверх” и “вниз” на клавиатуре. Для перехода в выбранную директорию необходимо нажать клавишу “Enter”. После выбора желаемого пути будет создана папка с именем “Repository”. Важно отметить, что название этой папки не следует изменять, так как система CSV использует его для идентификации репозитория и сохранения/восстановления файлов. Изменение названия приведет к некорректной работе системы.

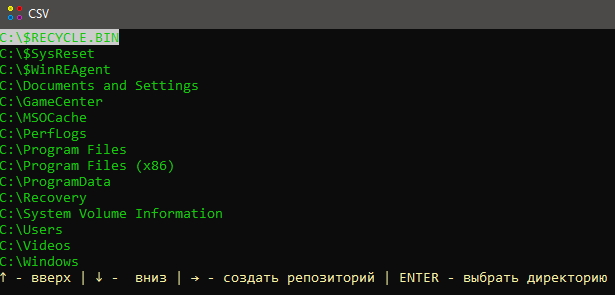


Рисунок 13 – Команда mkrepo

После создания репозитория вы можете начинать в нем работу.

Для того, чтобы сохранить текущее состояние вашего репозитория, включая все файлы и директории, используйте команду save (Рисунок 14). Важно отметить, что система CSV автоматически сохраняет все файлы, находящиеся в репозитории. Нет необходимости вручную добавлять отдельные файлы в список отслеживаемых - все изменения будут сохранены. После ввода команды save система запросит у вас комментарий к данной версии сохранения. Этот комментарий будет использоваться в качестве уникального идентификатора для этой конкретной версии файлов, и он поможет вам в будущем легко находить и загружать нужную версию. Пожалуйста, используйте информативные и запоминающиеся комментарии, чтобы облегчить процесс управления вашими сохраненными версиями.

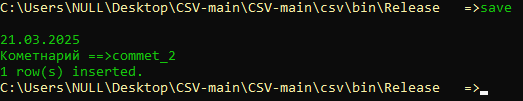


Рисунок 14 – Команда save

После успешного сохранения текущего состояния вашего репозитория в базе данных, вы можете просмотреть список всех сохраненных версий, используя команду info (Рисунок 15). Эта команда особенно полезна для навигации по истории изменений вашего проекта. С помощью info вы сможете увидеть даты сохранения каждой версии, а также связанные с ними комментарии, которые вы указали при сохранении. Это позволит вам легко идентифицировать нужное состояние репозитория и выбрать подходящую версию для последующей загрузки и восстановления. Команда info является важным инструментом для управления историей вашего проекта и восстановления предыдущих состояний.

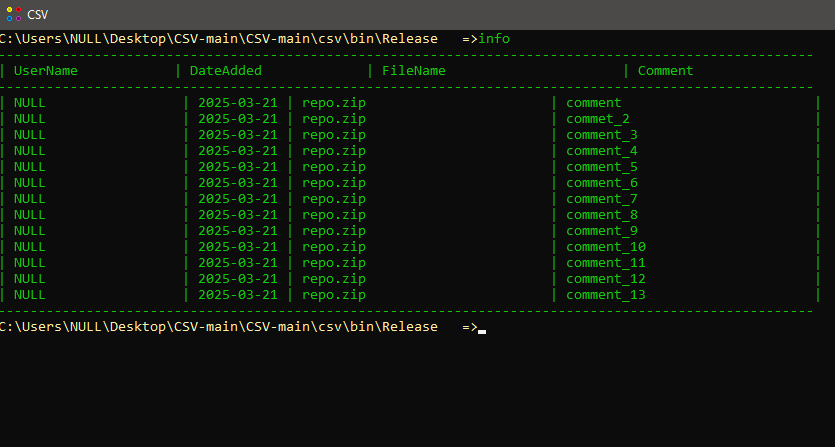


Рисунок 15 – Команда info

Перейдём к команде load, которая отвечает за восстановление ранее сохраненного состояния репозитория из базы данных. Чтобы загрузить определенную версию, введите команду load (Рисунок 16). После этого система запросит у вас комментарий, который вы указали при сохранении этого состояния. Введите точный текст комментария, соответствующий желаемой версии репозитория. После ввода комментария, система начнет процесс загрузки выбранного состояния из базы данных, восстанавливая файлы и структуру репозитория в том виде, в котором они были на момент сохранения. Убедитесь, что комментарий введен правильно, так как от этого зависит, какая именно версия репозитория будет загружена.

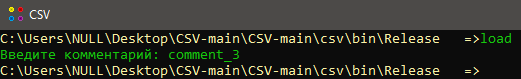


Рисунок 16 – Команда load

Теперь, когда вы ознакомились с основными командами системы контроля версий (CSV), такими как help, mkrepo, mkdata, dir, cls, save, load и info, вы обладаете достаточными знаниями для полноценного использования программы в управлении вашими проектами. Вы можете создавать репозитории, сохранять и восстанавливать состояния файлов, просматривать историю изменений и эффективно отслеживать прогресс вашей работы. Экспериментируйте с командами, изучайте их возможности и применяйте их для решения ваших задач. Удачи в работе с CSV!

# Список использованных источников

1. Документация C#: [Электронный ресурс]. URL:<https://learn.microsoft.com/ru-ru/dotnet/csharp/>
2. Документация SQLite: [Электронный ресурс]. URL:<https://learn.microsoft.com/ru-ru/dotnet/standard/data/sqlite/?tabs=netcore-cli>
3. Документация System.IO: [Электронный ресурс]. URL:<https://learn.microsoft.com/ru-ru/dotnet/api/system.io?view=net-8.0>
4. Документация Системы контроля версий: [Электронный ресурс]. URL:<https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B2%D0%B5%D1%80%D1%81%D0%B8%D1%8F%D0%BC%D0%B8>
5. Документация Git: [Электронный ресурс]. URL:<https://git-scm.com/book/ru/v2/>
6. Официальная документация VS Code (Microsoft):URL: https://code.visualstudio.com/docs
7. Документация C#: [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/
8. Документация System.IO: [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/api/system.io.file?view=net-8.0
9. Документация C#: [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-console?view=vs-2022>
10. Документация Visual Studio: [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-console?view=vs-2022
11. Документация C#: [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/shows/csharp-for-beginners/
12. Документация C#: [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/methods
13. Документация C#: [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/constructors>
14. Документация C#: [Электронный ресурс]. <https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/classes-and-structs/static-constructors>
15. Документация C#: [Электронный ресурс]. URL: https://learn.microsoft.com/ru-ru/dotnet/csharp/whats-new/csharp-14

# Приложение А

