МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

# ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование и разработка сервиса для создания системы управления версиями»

Обучающийся Е.Д. Евстигнеев
(Инициалы Фамилия) (личная подпись)

Научный руководитель канд.пед.наук, доцент, Т.А. Агошкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

## Содержание

# Введение

В наше время ключевыми факторам для выживания производственного предприятия становится обеспечение безопасности и высокой эффективности бизнес-процессов.

Для достижения этой цели важно создать условия для команды и применять современные и действенные инструменты.

Системы управления версиями играют важную роль в современной разработке программного обеспечения. Они помогают разработчикам следить за изменениями кода на протяжении всего цикла разработки, работать совместно с коллегами и при необходимости восстанавливать предыдущие версии файлов. Эти функции повышают эффективность рабочего процесса и улучшают результативность в программировании.

Актуальность данной работы заключается в том, что она решает растущую потребность в безопасном и эффективном управлении доступом в системах управления версиями путем внедрения национальных сервисов авторизации. Поскольку стандарты безопасности и контроль доступа пользователей стали критически важными при разработке программного обеспечения, особенно в отраслях, требующих соответствия государственным стандартам, данный подход обеспечивает как защиту данных, так и упрощенный доступ для проверенных пользователей.

Еще одной важной причиной востребованности систем управления версиями является переход к более гибким подходам в разработке, которые включают сокращенные циклы релизов и регулярные обновления. Системы управления версиями облегчают контроль и отслеживание изменений в коде, что помогает поддерживать его целостность и предотвращать нарушение работоспособности при внесении новых обновлений.

Объект исследования - система управления версиями.

Предмет исследования - сервис системы управления версиями, включающий функционал авторизации пользователей через национальные

сервисы.

Анализ проблемы выявил, что системы управления версиями с открытым исходным кодом, несмотря на широкое распространение, часто не обеспечивают должного уровня безопасности и удобства при авторизации пользователей. Отсутствие интеграции с национальными сервисами идентификации ограничивает возможности для настройки гибкой и безопасной авторизации, что особенно важно для организаций, работающих с чувствительными данными. Это создает риски для конфиденциальности и соответствия локальным нормативным требованиям, что требует доработки существующих решений.

Цель - исследование и разработка сервиса системы управления версиями на базе существующего решения с открытым исходным кодом, включая реализацию функционала авторизации пользователей через национальные сервисы.

Для достижения поставленной цели необходимо решить следующие задачи:

- проанализировать текущее состояние систем управления версиями, их ключевые функции и актуальные направления, включая интеграцию с национальными сервисами авторизации;
- подобрать и адаптировать методы для создания версии системы управления, учитывающей требования масштабируемости, надежности и интеграции с национальными сервисами;
- создать интуитивный интерфейс, который упростит взаимодействие разработчиков с системой и интеграцию авторизации через национальные сервисы;
- интегрировать функционал авторизации через национальные сервисы в существующую систему управления версиями.
- провести тестирование прототипа для проверки его функциональности, производительности и удобства использования в командной разработке с учетом новой функции авторизации.

Гипотеза исследования - использование системы управления версиями с открытым исходным кодом с добавлением модуля авторизации на основе национальных сервисов позволит повысить безопасность и удобство использования для команд разработчиков из России.

Методы исследования - в ходе исследования применены системный анализ, механизмы управления эффективностью организационной системы, а также технологии и методы проектирования информационных систем.

Научная новизна данной работы заключается в разработке и интеграции специализированного модуля авторизации на основе национальных сервисов в существующую систему управления версиями с открытым исходным кодом.

Практическая значимость работы заключается в улучшении удобства и безопасности использования существующей системы управления версиями с открытым исходным кодом за счет добавления функционала авторизации на основе национальных сервисов идентификации.

Положения, выносимые на защиту:

- современное состояние проблемы: проведен обзор и анализ текущего состояния систем управления версиями с открытым исходным кодом, выявлены недостатки и задачи, связанные с обеспечением пользовательской авторизации и интеграцией национальных сервисов идентификации;
- реализация усовершенствований: произведено расширение существующего интерфейса системы управления версиями для добавления функционала авторизации пользователей с использованием национальных сервисов идентификации. Описаны особенности и преимущества данной интеграции, в том числе улучшение безопасности и соответствие требованиям локальных нормативов;
- тестирование и оценка*:* проведено тестирование модифицированной системы управления версиями для оценки эффективности и удобства добавленного функционала авторизации, а также анализа его

применимости в условиях работы команд разработчиков с различными уровнями доступа.

Разработанное улучшение позволяет обеспечить безопасный и персонализированный доступ к системе, что особенно важно для команд разработчиков, работающих с конфиденциальными проектами и нуждающихся в строгом контроле доступа. Интеграция национальных сервисов авторизации делает данное решение применимым для организаций, требующих высокой безопасности и соответствия местным требованиям в области идентификации и аутентификации.

# 1 Аналитический этап исследования применения систем управления версиями

## 1.1 Современное состояние проблемы системы управления версиями

Аналитический этап исследования применения систем управления версиями является важным и неотъемлемым шагом в процессе разработки эффективных инструментов для управления исходным кодом в современных командах разработчиков. Системы управления версиями играют ключевую роль в обеспечении координации и стабильности при совместной работе над проектами, позволяя отслеживать изменения, управлять ветвлением и слиянием кода, а также облегчать восстановление предыдущих версий. На этом этапе исследования рассматриваются различные типы систем управления версиями, их возможности и ограничения, а также методологии и практики, которые применяются в реальных проектах для улучшения эффективности командной работы.

В рамках данного исследования важным акцентом является не создание новой системы управления версиями, а улучшение уже существующей с открытым исходным кодом. Мы планируем сосредоточиться на добавлении функционала авторизации пользователей с использованием национальных сервисов авторизации, что позволит повысить безопасность системы и упростить процесс доступа для пользователей. В отличие от создания новой системы управления версиями, улучшение текущих решений дает возможность использовать уже проверенные технологии и при этом интегрировать необходимые функции, такие как авторизация и управление доступом, для удовлетворения специфических требований современных команд и организаций.

Цель аналитического этапа — выявить основные тенденции и технологии, которые определяют выбор и внедрение систем управления

версиями в проектных командах, а также изучить особенности их применения в разных областях разработки. Важным аспектом исследования является анализ существующих решений и поиск путей их оптимизации с учетом текущих требований к гибкости, производительности и удобству использования систем. В частности, внимание будет уделено интеграции национальных сервисов авторизации, что поможет обеспечить высокий уровень безопасности, удобства и соответствия требованиям законодательства. Этот этап является основой для дальнейшей разработки и внедрения улучшений в системы управления версиями, направленных на повышение качества и скорости разработки программного обеспечения.

Проблема систем контроля версий — это хорошо разработанная область с несколькими устоявшимися и широко используемыми решениями, такими как Git, Mercurial и Subversion.

Эти системы позволяют разработчикам отслеживать изменения файлов с течением времени, сотрудничать с другими разработчиками при разработке кода и при необходимости легко возвращаться к предыдущим версиям файлов.

С появлением облачных платформ, таких как GitHub и GitLab, контроль версий стал еще более доступным и широко используемым. Однако при работе с очень большими кодовыми базами или большим количеством участников все еще существуют проблемы, такие как масштабируемость и производительность.

Кроме того, в этой области постоянно ведутся исследования, такие как совершенствование алгоритмов слияния и разработка новых распределенных систем контроля версий.

Системы управления версиями — это программные инструменты, которые позволяют разработчикам отслеживать изменения в коде с течением времени. Они позволяют разработчикам сотрудничать при разработке кода и при необходимости легко возвращаться к предыдущим версиям файлов. Наиболее широко используемыми системы управления версиями являются Git, Mercurial и Subversion.

Git — это распределенная система контроля версий, которая широко используется для разработки программного обеспечения. Она позволяет разработчикам одновременно работать над несколькими версиями кодовой базы и легко объединять изменения, внесенные разными разработчиками. Git также позволяет легко ветвиться и объединяться, что помогает разработчикам одновременно работать над несколькими функциями или исправлениями ошибок, не мешая друг другу.

Mercurial - еще одна распределенная система контроля версий, похожая на Git, но с более простой архитектурой. Она широко используется в сообществе Python и известна своей производительностью и масштабируемостью.

Subversion (также известная как SVN) - это централизованная система контроля версий, что означает, что все изменения в кодовой базе хранятся на центральном сервере. Она широко используется в корпоративных средах и известна своей стабильностью и простотой использования.

GitHub и GitLab, платформы предоставляют разработчикам веб-интерфейс для доступа к коду и совместной работы над ним, а также инструменты для проверки кода, непрерывной интеграции и т. д. Кроме того, эти платформы также предоставляют разработчикам возможность делиться проектами с открытым исходным кодом и открывать их для себя.

Однако в области контроля версий все еще существуют проблемы. Одной из основных проблем является масштабируемость, то есть способность системы работать с большими базами кода или большим количеством участников.

Другая проблема - производительность, то есть способность системы обрабатывать большое количество запросов или быстро выполнять операции.

Кроме того, в этой области ведутся постоянные исследования, такие как улучшение алгоритмов слияния и разработка новых распределенных систем контроля версий. Пример графического изображения слияния представлен на рисунке 1.

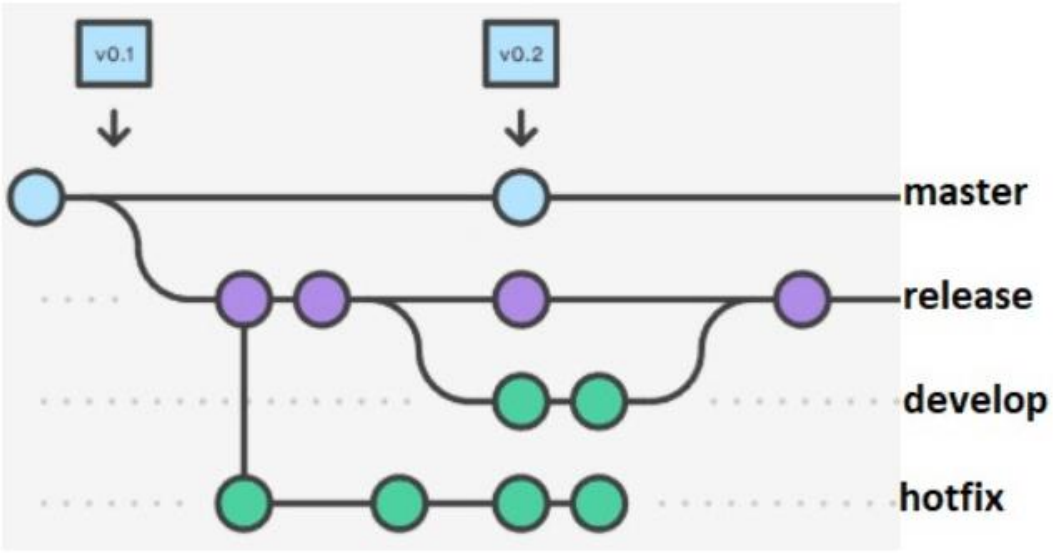


Рисунок 1- Систем управления версиями в графическом виде.

Системы управления версиями являются важным инструментом для разработки программного обеспечения, позволяя разработчикам сотрудничать, отслеживать изменения и легко возвращаться к предыдущим версиям файлов.

Системы управления версиями актуальны в контексте разработки программного обеспечения с открытым исходным кодом. Проекты с открытым исходным кодом часто имеют большое количество участников, поэтому важно отслеживать изменения в кодовой базе и вести учет различных версий проекта. Такие платформы, как GitHub и GitLab, облегчают разработчикам внесение вклада в проекты с открытым исходным кодом и сотрудничество с другими.

Системы управления версиями также актуальны в контексте практик DevOps и Continuous Integration/Continuous Deployment (CI/CD). Системы управления версиями играет важную роль в этих практиках, поскольку позволяет разработчикам отслеживать различные версии кодовой базы и при необходимости легко откатываться к предыдущей версии. Это помогает

гарантировать, что кодовая база всегда находится в состоянии, пригодном для выпуска.

Системы управления версиями актуальны в настоящее время, поскольку они предоставляют важные возможности для разработки программного обеспечения, такие как совместная работа, отслеживание изменений и легкий возврат к предыдущим версиям файлов. Эти возможности необходимы для управления сложностью и быстрыми темпами современной разработки программного обеспечения, а также для разработки программного обеспечения с открытым исходным кодом.

### 1.2 Обзор и анализ источников по теме исследования

«A short history of Git» - это статья, написанная Джеймсом Когланом и опубликованная на сайте «codewords.recurse.com». В статье приводится краткая история развития Git, распределенной системы управления версиями. Статья начинается с описания того, как Git был создан Линусом Торвальдсом в 2005 году в качестве инструмента для управления разработкой ядра Linux. «В статье объясняются основные концепции Git, такие как разница между централизованной и распределенной системой контроля версий, важность ветвлений и слияния, а также основные команды Git. В статье также говорится о том, что конструкция Git обеспечивает высокую степень гибкости и скорости, что делает ее хорошо подходящей для проектов любого размера» [1]. В статье также объясняется, чем Git отличается от других систем контроля версий, таких как Subversion, и как он стал одной из самых популярных систем контроля версий, используемых сегодня [2]. Статья завершается кратким упоминанием некоторых более продвинутых возможностей Git'a и того, как он постоянно развивается, чтобы удовлетворить потребности своих пользователей. Статья представляет собой хорошее введение в Git, его историю, основные концепции и возможности, которые делают его таким мощным инструментом для управления проектами разработки программного

обеспечения. Она написана понятным и доступным языком и может стать хорошей отправной точкой для новичков в Git.

«Mercurial: The Definitive Guide» - это книга, написанная Брайаном О'Салливаном. Это исчерпывающее руководство по Mercurial, распределенной системе контроля версий аналогичной Git. Книга охватывает все аспекты использования Mercurial, от базовых концепций до расширенных возможностей [3]. Книга начинается с введения в распределенные системы управления версиями и рассказывает о том, чем Mercurial отличается от других систем, таких как Git и Subversion [4]. Затем в ней рассматриваются основы использования Mercurial, включая установку, создание и управление репозиториями, а также выполнение таких распространенных задач, как фиксация и слияние изменений. В книге также рассматриваются более сложные темы, такие как ветвление и слияние, совместное использование кода и работа над ним, а также устранение неполадок. Кроме того, в книге рассказывается о том, как использовать Mercurial в командной среде, включая настройку и использование общих репозиториев, а также лучшие практики работы с другими разработчиками. Книга также включает справочный раздел, в котором содержится подробная информация о командах и опциях Mercurial, а также примеры их использования в различных сценариях.

«Mercurial: The Definitive Guide» считается одним из лучших ресурсов для изучения Mercurial, он хорошо написан, прост для понимания и охватывает все основные возможности Mercurial [5]. Он подходит как для новичков, так и для опытных пользователей, и считается основным ресурсом для всех, кто хочет изучить и эффективно использовать Mercurial.

«Subversion vs Git: Сравнительное исследование» - это статья, написанная Р.К. Джоши, А.К. Сингхом и Р.К. Гуптой. Она была опубликована на сайте «researchgate.net». В статье сравнивается функциональность двух популярных систем контроля версий: Subversion и Git [6]. Статья начинается с обзора обеих систем, включая их историю и основные особенности. Затем в ней приводится сравнение двух систем по различным аспектам, таким как

производительность, масштабируемость, безопасность и функциональность [7]. Авторы также обсудили преимущества и недостатки каждой системы и то, как они могут быть полезны для различных типов проектов. Один из ключевых выводов статьи заключается в том, что Git лучше с точки зрения производительности и масштабируемости, в то время как Subversion лучше с точки зрения безопасности [8]. В статье также отмечается, что Git обладает большей функциональностью, чем Subversion, и больше подходит для распределенной разработки, в то время как Subversion больше подходит для централизованной разработки. В данной статье также приводится подробный анализ различий между этими двумя системами, включая различия в командах, возможностях ветвления и слияния, а также в том, как они обрабатывают конфликты. Статья предоставляет всестороннее сравнение Subversion и Git и помогает разработчикам и командам решить, какая система лучше всего подходит для их конкретных нужд [9]. Она основана на исследованиях и экспериментах и дает хорошее понимание возможностей обеих систем и их сравнения друг с другом.

«A Survey of Version Control Systems for Machine Learning Projects» - это научная статья, написанная Александром Ченом и опубликованная в сборнике трудов Международной конференции по искусственному интеллекту в 2017 году. В статье представлен всесторонний обзор различных систем контроля версий, которые обычно используются для управления проектами машинного обучения [10]. Статья начинается с обсуждения проблем контроля версий в проектах машинного обучения, включая необходимость работы с большими наборами данных, необходимость воспроизводимости и сотрудничества между членами команды [11]. Затем в статье рассматривается ряд популярных систем контроля версий, включая Git, Subversion, Mercurial и другие. В статье также рассматривается несколько специализированных систем контроля версий, специально разработанных для проектов машинного обучения, таких как DVC и Pachyderm. Затем в статье проводится сравнительный анализ рассмотренных систем контроля версий, оценивая их по ряду критериев,

включая производительность, масштабируемость, удобство использования и возможности совместной работы. В статье также обсуждаются лучшие практики использования систем контроля версий в проектах машинного обучения, включая важность документации и версионирования наборов данных. Статья представляет собой ценный ресурс для исследователей и практиков в области машинного обучения, освещая ключевые соображения и лучшие практики управления контролем версий в проектах такого типа.

«Контроль версий с помощью Subversion» - это книга, написанная К. Майклом Пилато, Беном Коллинз-Суссманом и Брайаном В. Фицпатриком и опубликованная издательством O'Reilly Media в 2004 году. Книга представляет собой исчерпывающее руководство по использованию системы управления версиями Subversion, которая является программным инструментом с открытым исходным кодом для управления и отслеживания изменений в файлах и каталогах [12]. Книга начинается с введения основных понятий контроля версий и объяснения принципов работы Subversion. Затем рассматривается установка и настройка Subversion, а также приводятся подробные инструкции по использованию системы для управления файлами и каталогами. В книге также рассматриваются такие продвинутые темы, как ветвление, слияние и интеграция с другими инструментами разработки программного обеспечения. В книге также подробно рассматривается клиент командной строки Subversion и его различные команды, а также API Subversion и его использование. В книге также рассматривается использование subversion в различных средах, включая Windows и Linux. На протяжении всей книги авторы приводят реальные примеры и практические советы по эффективному использованию Subversion, что делает ее ценным ресурсом для разработчиков программного обеспечения, системных администраторов и всех, кому необходимо управлять и отслеживать изменения файлов и каталогов.

«Version Control with Subversion» считается классическим и авторитетным руководством по использованию системы управления версиями

Subversion и является полезным ресурсом для всех, кто хочет научиться эффективно использовать Subversion в своих проектах разработки.

«Version Control with Subversion» - это книга, написанная К. Майклом Пилато, Беном Коллинз-Суссманом и Брайаном В. Фицпатриком и опубликованная издательством O'Reilly Media в 2004 году. Книга представляет собой исчерпывающее руководство по использованию системы управления версиями Subversion, которая является программным инструментом с открытым исходным кодом для управления и отслеживания изменений в файлах и каталогах [13]. Книга предназначена для разработчиков программного обеспечения, системных администраторов и всех тех, кому необходимо управлять и отслеживать изменения в файлах и каталогах. Книга начинается с введения основных понятий управления версиями и объяснения принципов работы Subversion, включая теорию управления версиями, архитектуру Subversion и преимущества ее использования. Затем в книге рассматривается установка и настройка Subversion и приводятся подробные инструкции по использованию системы для управления файлами и каталогами. В книге также рассматриваются такие продвинутые темы, как ветвление, слияние и интеграция с другими инструментами разработки программного обеспечения. Авторы также приводят реальные примеры и практические советы по эффективному использованию Subversion. В книге также подробно рассматривается клиент командной строки Subversion и его различные команды, а также API Subversion и его использование. В книге также рассматривается использование Subversion в различных средах, включая Windows и Linux.

«Version Control with Subversion» представляет собой исчерпывающее руководство по использованию системы управления версиями Subversion и считается классическим и авторитетным руководством по использованию Subversion в проектах разработки программного обеспечения, предоставляя массу информации, советов и лучших практик, которые помогут читателям эффективно использовать Subversion.

Иследования и литература, рассмотренные в этих статьях, подчеркивают важность эффективного контроля версий в разработке программного обеспечения и проектах машинного обучения. В статьях представлена важная информация о различных системах контроля версий, таких как Git, Subversion, Mercurial, DVC и Pachyderm, с подробным описанием их преимуществ и недостатков. Также даны рекомендации по эффективному использованию систем контроля версий, включая лучшие практики для документирования и версионирования наборов данных.

«Статьи «A Survey of Version Control Systems for Machine Learning Projects» и «Version Control with Subversion» содержат подробный анализ систем контроля версий и их использования в проектах машинного обучения и разработки программного обеспечения соответственно. В них содержится много информации об установке, настройке и использовании этих систем, а также о том, как эффективно применять их в различных средах» [14][15].

В заключение следует отметить, что проведенный анализ статей предоставляет ценную информацию для разработчиков программного обеспечения, системных администраторов и исследователей, раскрывая лучшие практики и ключевые аспекты внедрения систем управления версиями. Рассмотренные материалы позволяют глубже понять существующие подходы и решения, а также выделить проблемы и возможности, связанные с использованием систем контроля версий в разработке программного обеспечения и проектах машинного обучения.

Однако анализ также выявил существующую нишу — отсутствие интеграции национальных сервисов авторизации в широко используемых системах управления версиями. Этот аспект остается недостаточно изученным и представляет собой актуальное направление для дальнейших исследований. Данная работа направлена на устранение этого пробела, предлагая реализацию авторизации через национальные сервисы в рамках существующей системы с открытым исходным кодом.

### 1.3 Обзор и анализ систем управления версиями

Рассмотрим и проанализируем четыре самые популярные системы управления версиями.

«Git — это распределенная система контроля версий, которая позволяет разработчикам отслеживать изменения в коде и управлять ими. Впервые она была разработана Линусом Торвальдсом в 2005 году как инструмент для управления разработкой ядра Linux» [16].

Одной из ключевых особенностей Git является возможность одновременной работы с несколькими ветвями разработки. Это позволяет разработчикам одновременно работать над несколькими функциями или исправлениями ошибок, не мешая друг другу. Разработчики могут создавать свои собственные ветки, вносить изменения, а затем объединять эти изменения в основную ветку, когда они будут готовы.

«Еще одной важной особенностью Git является способность решать конфликты, которые могут возникнуть, когда несколько человек работают над одним и тем же кодом. Git предоставляет инструменты для разрешения конфликтов, такие как возможность видеть изменения, внесенные каждым человеком, и возможность вернуться к предыдущей версии кода, если это необходимо» [20].

Git также позволяет легко сотрудничать и обмениваться кодом. Разработчики могут направлять свои изменения в центральный репозиторий, например, GitHub, где другие разработчики могут получить доступ к коду и просмотреть его. Это облегчает совместную работу групп разработчиков над проектом, а также позволяет проектам с открытым исходным кодом принимать вклады от сообщества разработчиков.

В дополнение к мощным возможностям совместной работы и контроля версий, Git получил широкое распространение и был интегрирован во многие инструменты и рабочие процессы разработки программного обеспечения. К ним относятся популярные интегрированные среды разработки (IDE) и

инструменты непрерывной интеграции, что делает его необходимым инструментом для современной разработки программного обеспечения.

В целом, Git — это мощный и гибкий инструмент, который необходим любой команде разработчиков программного обеспечения. Его способность работать с несколькими ветками, разрешать конфликты, облегчать сотрудничество и обмен кодом делает его бесценным инструментом для управления и разработки кода.

«Subversion (также известная как SVN) - это централизованная система контроля версий, которая позволяет разработчикам отслеживать изменения в коде и управлять ими. Впервые она была разработана в 2000 году как ответ на ограничения СУВ, популярной в то время системы управления версиями» [23].

Одной из ключевых особенностей Subversion является ее централизованный репозиторий, который выступает в качестве единой точки хранения всех версий кода проекта. Это позволяет упростить доступ и сотрудничество между членами команды, поскольку все разработчики могут проверять и фиксировать изменения в одном и том же хранилище.

Subversion предлагает мощные инструменты для управления ветками и тегами, что позволяет разработчикам легко создавать и переключаться между разными версиями проекта. Это упрощает параллельную работу над новыми функциями и исправлением ошибок, позволяя участникам проекта работать независимо и избегать пересечений.

Кроме того, Subversion обладает встроенными средствами для разрешения конфликтов, возникающих при совместной работе над кодом. Система позволяет отслеживать изменения, внесенные каждым разработчиком, и предоставляет возможность вернуться к предыдущей версии, если потребуется, что значительно облегчает процесс координации в команде.

Subversion также предоставляет богатый набор функций контроля доступа, что позволяет администратору контролировать, кто имеет доступ к репозиторию и какие действия они могут выполнять. Это особенно полезно

для организаций с несколькими командами, работающими над разными проектами.

«В дополнение к мощным возможностям совместной работы и контроля версий, Subversion получила широкое распространение и была интегрирована во многие инструменты разработки программного обеспечения и рабочие процессы. К ним относятся популярные интегрированные среды разработки (IDE) и инструменты непрерывной интеграции, что делает ее незаменимым инструментом для современной разработки программного обеспечения. В целом, Subversion - это мощный и гибкий инструмент, который необходим любой команде разработчиков программного обеспечения» [24].

Его централизованный репозиторий, возможность работы с ветвями и метками, разрешение конфликтов и контроль доступа делают его бесценным инструментом для управления и разработки кода.

«Mercurial — это популярная система контроля версий, которую используют многие команды разработчиков программного обеспечения. Это распределенная система контроля версий, что означает, что каждый пользователь имеет полную копию всей истории проекта на своей локальной машине. Это обеспечивает быструю и эффективную совместную работу, поскольку пользователи могут работать над проектом в автономном режиме, а затем отправлять свои изменения в центральный репозиторий» [19].

Одной из ключевых особенностей Mercurial является его гибкость. Его можно использовать как для небольших, так и для крупных проектов, и он совместим с широким спектром операционных систем. Он также имеет простой и интуитивно понятный интерфейс командной строки, что делает его удобным для разработчиков. Кроме того, Mercurial поддерживает широкий спектр плагинов и расширений, которые можно использовать для добавления новой функциональности или интеграции с другими инструментами.

Еще одним преимуществом Mercurial является его производительность. Он предназначен для работы с большими проектами с большим количеством файлов и ветвей, и способен одновременно работать с несколькими

пользователями над одной кодовой базой. Mercurial также включает ряд дополнительных возможностей, таких как возможность работы с двоичными файлами, отслеживание слияния и возможность легко управлять и объединять ветви.

Mercurial также имеет сильное сообщество разработчиков и пользователей, которое предоставляет множество ресурсов и поддержку. Существует множество онлайн-учебников, документации и форумов, где пользователи могут задавать вопросы и делиться своим опытом. Кроме того, существует множество сторонних инструментов и сервисов, которые интегрируются с Mercurial, например, веб-инструменты для рецензирования и управления кодом, что может сделать его еще более мощным.

Perforce Helix Core - одна из самых мощных и высокопроизводительных систем контроля версий , специально разработанная для эффективного управления большими объемами данных и сложными проектами. Разработанная компанией Perforce Software, она получила широкое признание среди крупных организаций, особенно в игровой индустрии, где надежное управление мультимедийными активами имеет решающее значение. Helix Core славится своей скоростью и способностью работать с большими хранилищами, что делает его незаменимым для компаний, работающих с тяжелыми бинарными файлами, такими как текстуры, модели и аудиоданные.

Одним из ключевых преимуществ Helix Core является его исключительная производительность. Система способна обрабатывать огромные объемы данных, обеспечивая быструю передачу файлов и синхронизацию по сети. Благодаря оптимизированной архитектуре Helix Core эффективно работает даже при работе с хранилищами, содержащими сотни гигабайт данных. Эта особенность делает систему идеальной для проектов, где требуется не только контроль версий текстовых файлов, но и поддержка больших мультимедийных активов. Для разработчиков игр или мультимедийных приложений такой уровень производительности является критически важным.

Helix Core предлагает гибкость в выборе моделей рабочего процесса: пользователи могут использовать как централизованный, так и распределенный контроль версий. Централизованная модель позволяет управлять проектом с одного сервера, что подходит для проектов со строгой иерархией и жесткими требованиями к контролю доступа. Распределенная модель предоставляет разработчикам возможность работать в автономном режиме, что очень важно для команд, расположенных в разных местах. Такая гибкость позволяет командам адаптировать Helix Core к конкретным потребностям проекта и организационной структуре.

Важной особенностью Helix Core является поддержка больших файлов. В отличие от многих других систем контроля версий, которые испытывают трудности с управлением бинарными файлами, Helix Core эффективно справляется с такими задачами без ущерба для производительности. Эта возможность особенно важна для студий, занимающихся разработкой игр или производством мультимедиа, где большие файлы данных - обычное дело. Кроме того, система обладает высокой масштабируемостью, что позволяет использовать ее как в небольших коллективах, так и на крупных предприятиях с тысячами сотрудников, распределенных по всему миру. Возможность развертывания Helix Core как в облаке, так и в локальной сети обеспечивает организациям дополнительную гибкость в управлении проектами и распределении ресурсов.

Helix Core также известен своими надежными функциями безопасности и контроля доступа. Администраторы могут настраивать разрешения на уровне файлов и каталогов, что очень важно для защиты конфиденциальных данных и ограничения доступа к определенным частям кодовой базы. Эта функция жизненно важна для компаний, работающих с конфиденциальной информацией или в сложных командных средах, где различным подразделениям требуется разный уровень доступа к данным проекта.

«Еще одним важным аспектом Helix Core является его бесшовная интеграция с другими инструментами разработки. Система поддерживает

интеграцию с популярными инструментами CI/CD, такими как Jenkins и GitLab, упрощая настройку конвейеров непрерывной интеграции и доставки. Она также интегрируется с такими инструментами управления проектами, как Jira, позволяя разработчикам отслеживать изменения и более эффективно управлять проектами. Эти возможности делают Helix Core неотъемлемой частью экосистемы инструментов разработки».

Несмотря на многочисленные преимущества, Helix Core имеет и некоторые недостатки. Система имеет сложную кривую обучения, и новичкам может потребоваться время, чтобы понять ее функциональность и эффективно использовать все возможности. Кроме того, стоимость лицензирования может быть значительной для больших команд, особенно учитывая коммерческий характер продукта. Настройка Helix Core для крупных проектов также может потребовать значительного времени и ресурсов, что приведет к дополнительным инвестициям в настройку и поддержку системы.

Helix Core широко используется в различных отраслях. В разработке игр такие компании, как Ubisoft и Epic Games, используют его для управления игровым кодом и активами. Финансовые организации, в том числе банки, используют Helix Core для управления сложными программными проектами с жесткими требованиями к безопасности. На производстве система используется для эффективной обработки инженерных данных и проектов по разработке программного обеспечения.

Таким образом, Perforce Helix Core остается одной из самых мощных и востребованных систем контроля версий, обеспечивая высокую производительность, гибкость и надежность. Она особенно хорошо подходит для крупных проектов, связанных с тяжелыми мультимедийными файлами, и обеспечивает необходимую безопасность и контроль, что делает ее незаменимым инструментом для ведущих компаний различных отраслей. В следующем разделе будет приведена сравнительная таблица.

## 1.4 Сравнительный анализ рассмотренных систем управления проектами

«Helix Core, Git, Subversion и Mercurial - эти системы играют важнейшую роль в разработке программного обеспечения, предлагая различные возможности и подходы к управлению кодовыми базами и совместными проектами» [17].

Git - это распределенная система контроля версий, то есть каждый разработчик хранит полную копию истории проекта на своей локальной машине. Такая структура обеспечивает быструю и эффективную совместную работу, поскольку разработчики могут работать в автономном режиме, а после возвращения в сеть отправлять изменения в центральный репозиторий. Обширное и активное сообщество Git обеспечивает широкую поддержку и ресурсы, что делает его одной из самых распространенных платформ системы управления версиями. Его сильная сторона - скорость, гибкость и надежная модель ветвления, что очень важно для сложных рабочих процессов разработки.

«Subversion (SVN) - это централизованная система контроля версий, которая требует от разработчиков подключения к единому центральному хранилищу для доступа к проекту. Известная своей надежностью и стабильностью, Subversion особенно популярна в организациях, где требуется строгий надзор и контроль над доступом к кодовой базе. Несмотря на надежность SVN, она считается менее гибкой по сравнению с распределенными системами вроде Git» [29][30].

Тем не менее, она остается предпочтительным выбором для команд, которым нужна простая централизованная модель без сложностей распределенного версионирования.

«Mercurial — это еще одна распределенная система контроля версий, схожая с Git, предназначенная для управления крупными проектами с множеством файлов и веток. Она известна простотой и удобством

использования, что делает ее привлекательной как для небольших команд, так и для масштабных проектов. Mercurial отличается высокой производительностью и гибкостью, легко адаптируясь под различные рабочие процессы. Хотя сообщество Mercurial уступает по численности сообществу Git, она остается достойным выбором для проектов, где предпочтителен распределенный подход» [18].

Helix Core от Perforce выделяется своей нацеленностью на производительность и масштабируемость, особенно для крупных проектов корпоративного уровня. Он предназначен для эффективной работы с огромными репозиториями, в том числе с тяжелыми бинарными активами, что делает его идеальным для таких отраслей, как разработка игр, где часто встречаются большие файлы. Helix Core поддерживает как централизованные, так и распределенные рабочие процессы, что дает командам возможность выбрать наиболее подходящий для них вариант. Кроме того, в нем реализованы надежные средства безопасности и контроля доступа, необходимые для защиты конфиденциальных данных и управления разрешениями в сложных организационных структурах.

Кроме того, интеграция Helix Core с инструментами для CI/CD и управления проектами, такими как Jenkins и Jira, повышает производительность и упрощает процесс разработки. Несмотря на высокую производительность, Helix Core может быть сложным в освоении, а его стоимость может быть значительной для больших команд, особенно в коммерческих организациях.

Исходя из полученных оценок, можно сделать вывод, что направление исследования в области систем контроля версий должно быть направлено на разработку системы, сочетающей в себе лучшие черты Git, Subversion и Mercurial.

Распределенная архитектура, широкое сообщество и высокая гибкость Git делают его популярным выбором для различных проектов, однако при работе с очень крупными репозиториями могут возникать проблемы с

производительностью [28]. Централизованная архитектура Subversion обеспечивает стабильность и надежность, что делает его подходящим вариантом для некоторых проектов, хотя гибкость у него ниже, чем у Git. Mercurial также предлагает распределенную архитектуру, гибкость и возможность работы с крупными проектами, что делает его интересным вариантом для многих разработчиков.

Будущее исследование должно быть направлено на создание системы, которая сочетает распределенную архитектуру и гибкость Git и Mercurial со стабильностью и надежностью Subversion, одновременно решая проблемы производительности при работе с большими репозиториями [25]. Она также должна быть нацелена на обеспечение дружественного интерфейса, обширной документации и ресурсов, чтобы помочь пользователям изучить и использовать систему. Кроме того, она должна включать дополнительные возможности, такие как отслеживание слияний, возможность работы с двоичными файлами. Сравнение систем управления версиями представлено в таблице 1.

Таблица 1 – Сравнение систем управления версиями

| Критерий | Git | Subversion (SVN) | Mercurial | Helix Core |
|---|---|---|---|---|
| Тип | 5 (Распределённая) | 3 (Централизованная) | 5 (Распределённая) | 4 (Обе модели) |
| Сообщество | 5 | 3 | 3 | 4 |
| Гибкость | 5 | 3 | 4 | 4 |
| Производительность | 5 | 3 | 4 | 5 |
| Ресурсы | 5 | 4 | 3 | 4 |
| Дополнительные возможности | 5 | 3 | 4 | 5 |
| Популярность | 5 | 3 | 3 | 4 |

Это сравнение подчеркивает сильные и слабые стороны каждой системы управления версиями, демонстрируя, как разные системы проявляют себя в конкретных сценариях. Git идеально подходит для проектов с открытым

исходным кодом и совместной работы, требующих гибких рабочих процессов. Subversion остается надежной для организаций, предпочитающих централизованный подход. Mercurial привлекает тех, кто ищет более простую распределенную систему. В то же время Helix Core незаменим для предприятий, управляющих крупными и сложными проектами, особенно там, где производительность и масштабируемость являются критическими факторами.

В результате сравнительного анализа систем управления версиями Helix Core, Git, Subversion и Mercurial были выявлены их ключевые преимущества и недостатки. Git и Mercurial выделяются гибкостью и популярностью в распределенных проектах, Subversion предлагает стабильность для централизованных решений, а Helix Core демонстрирует высокую производительность в крупных проектах. Однако ни одна из систем не предоставляет встроенной интеграции с национальными сервисами авторизации, что подтверждает актуальность разработки дополнительных решений в этой области.

# 2 Теоретический этап построения системы управления версиями

## 2.1 Анализ методов системы управления версиями для эффективного управления проектами

В современном быстро меняющемся ландшафте разработки программного обеспечения системы управления версиями играют решающую роль в управлении и отслеживании изменений исходного кода и других файлов проекта. Для принятия взвешенного решения о выборе наиболее подходящей системы управления версиями, разработчикам и организациям необходимо умение детально анализировать и сопоставлять разные подходы в этой области. Настоящее исследование направлено на проведение комплексного анализа методов системы управления версиями, чтобы выделить их ключевые особенности, основные преимущества и возможные ограничения.

Централизованные системы управления версиями являются одними из самых ранних и традиционных методов СУВ. В этой системе центральный сервер хранит весь репозиторий, а разработчики взаимодействуют с сервером для доступа, фиксации и объединения изменений кода. Примерами централизованных систем управления версиями являются Subversion (SVN) и Perforce.

К преимуществам относят:

- простая настройка и администрирование;
- обеспечивает единый источник истины;
- позволяет осуществлять тонкий контроль доступа;
- подходит для небольших команд, работающих в централизованной среде.

Недостатками считаются следующие моменты:

- требуется постоянное сетевое соединение с центральным сервером;
- подвержен сбоям в работе сервера, что приводит к простою проекта;

- ограниченные возможности автономной работы;
- трудности с ветвлением и объединением сложных изменений кода.

Распределенные системы управления версиями возникли как ответ на ограничения централизованных систем. Каждый разработчик поддерживает полную локальную копию репозитория, включая всю историю проекта. Git и Mercurial - популярные примеры распределенных систем управления версиями.

Преимущества:

- позволяет работать и делать коммиты в автономном режиме;
- обеспечивает более высокую производительность, поскольку большинство операций выполняется локально;
- расширенные возможности ветвления и слияния;
- распределенная природа обеспечивает избыточность и надежность.

Недостатки:

- более сложная кривая обучения по сравнению с централизованными системы управления версиями;
- требуется больше дискового пространства из-за локальных копий репозитория;
- первоначальная установка и настройка может быть сложной;
- ограниченный контроль над правами доступа к коду.

Гибридные системы управления версиями стремятся объединить преимущества как централизованной системы управления версиями, так и децентрализованной системы управления версиями. Эти системы обычно включают центральный сервер, но позволяют разработчикам работать с локальными репозиториями и выполнять многие операции в автономном режиме. Git с централизованным сервером (например, GitLab или Bitbucket) является примером гибридной системы управления версиями.

Преимущества:

- обеспечивает баланс между централизованным и распределенным подходами;

- позволяет работать и коммитить в автономном режиме, с возможностью синхронизации изменений с центральным сервером;
- эффективные возможности ветвления и слияния;
- обеспечивает большую гибкость и масштабируемость для больших команд и сложных проектов.

Недостатки:

- сложность настройки и обслуживания, по сравнению с централизованной системой управления версиями;
- потенциальные проблемы синхронизации между локальным и центральным репозиториями;
- требуется кривая обучения для разработчиков, знакомых с централизованной системы управления версиями или децентрализованной системы управления версиями.

В заключение следует подчеркнуть, что анализ различных методов систем контроля версий является ключевым для выбора наиболее подходящего инструмента управления и отслеживания изменений исходного кода в проектах разработки программного обеспечения. Централизованные системы управления версиями, такие как Subversion, предлагают простоту и централизованный контроль, что делает их идеальными для проектов с жесткой структурой и строгим управлением доступом. В то же время распределенные системы, такие как Git и Mercurial, обеспечивают повышенную гибкость и автономные возможности, что особенно полезно для распределенных команд и динамичных проектов. Гибридные системы, например, Helix Core, сочетают в себе преимущества как централизованных, так и распределенных подходов, предлагая баланс между контролем и гибкостью.

Сравнительный анализ этих систем позволил наглядно выявить их сильные и слабые стороны, что облегчает понимание их применимости в различных условиях разработки. Такой подход позволяет разработчикам и организациям принимать обоснованные решения, выбирая систему, наиболее

соответствующую их специфическим потребностям и рабочим процессам. В результате, оптимизация выбора системы контроля версий способствует улучшению рабочего процесса, повышению продуктивности команды и обеспечению эффективного сотрудничества в проектах разработки программного обеспечения.

Таким образом, глубокое понимание особенностей, преимуществ и недостатков каждой системы управления версиями не только обогащает знания специалистов, но и служит основой для внедрения наиболее эффективных решений, способствующих успешной реализации программных проектов.

## 2.2 Моделирование бизнес-процессов с использованием BPMN

Моделирование бизнес-процессов с использованием BPMN (Business Process Model and Notation) существенно улучшает управление версиями в системе контроля версий, делая процессы прозрачными и понятными. Визуализация ключевых активностей, таких как создание, слияние или откат изменений, помогает команде координировать действия и избегать ошибок.

BPMN-диаграммы отображают последовательность действий, роли участников и условия выполнения задач. Это упрощает восприятие процессов и обеспечивает единую коммуникационную основу для команды. Проверка диаграммы на соответствие реальным потребностям позволяет выявить узкие места и оптимизировать процесс.

В результате BPMN не только повышает эффективность управления версиями, но и служит инструментом для обучения и документирования, что делает совместную работу более организованной. Диаграмма приведена на рисунке 2.


Рисунок 2 - BPMN-диаграмма назначения прав доступа

«Моделирование бизнес-процессов с использованием BPMN в рамках СУВ позволяет лучше понять, визуализировать и коммуницировать бизнес-процессы, связанные с управлением версиями и изменениями в проектах. Это помогает улучшить эффективность, прозрачность и качество работы с СУВ» [21].

Моделирование бизнес-процессов с использованием BPMN является важным инструментом для оптимизации рабочих процессов в проектах разработки программного обеспечения. В контексте улучшения существующей системы управления версиями и добавления функционала авторизации пользователей через национальные сервисы, BPMN может оказать значительную помощь в детальном планировании и оптимизации различных аспектов работы системы.

Одним из ключевых аспектов использования BPMN в проекте является процесс авторизации пользователей. Для внедрения авторизации через национальные сервисы необходимо чётко описать и визуализировать все шаги, связанные с этим процессом. BPMN позволит наглядно отобразить все этапы, начиная от запроса пользователя на вход в систему и заканчивая предоставлением ему прав доступа в зависимости от его роли. Такой подход

помогает не только улучшить понимание структуры процесса, но и оптимизировать его, выявляя возможные узкие места и устраняя их на ранних этапах разработки.

Кроме того, BPMN помогает эффективно моделировать процессы взаимодействия пользователей с системой управления версиями. Визуализация таких процессов, как создание веток, слияние изменений и управление правами доступа, позволит более чётко понять, как различные пользователи (разработчики, администраторы, тестировщики и другие) будут взаимодействовать с системой. Этот подход способствует улучшению качества взаимодействия внутри команды и оптимизации рабочего процесса. Моделирование поможет избежать дублирования шагов, упростить процессы и ускорить выполнение задач, что, в свою очередь, повышает производительность.

Важным элементом является управление правами доступа и ролями пользователей. Моделирование с использованием BPMN позволит визуализировать, как различные пользователи получают доступ к определённым частям системы в зависимости от своей роли. Это может помочь выявить недостатки в логике распределения прав и обеспечении безопасности, а также предложить варианты улучшений. Используя BPMN, можно будет проследить все возможные пути пользователей через систему и удостовериться, что каждая роль имеет доступ только к необходимым данным.

Бизнес-моделирование с помощью BPMN также способствует улучшению коммуникации внутри команды разработки. Чёткое представление всех процессов в виде диаграмм делает их легко воспринимаемыми, что помогает всем участникам проекта быстрее понять структуру системы и её функционирование. Это особенно важно при работе с открытым исходным кодом, когда необходимо обеспечить чёткую документацию и взаимодействие между различными разработчиками.

Кроме того, моделирование бизнес-процессов с использованием BPMN помогает в тестировании. Каждый шаг процесса, будь то авторизация

пользователей или взаимодействие с системой, можно заранее спланировать и протестировать, что значительно улучшает качество разработки. Заранее спроектированные и протестированные процессы позволяют избежать ошибок на поздних стадиях реализации и ускоряют процесс внедрения нового функционала.

Наконец, BPMN позволяет эффективно анализировать и минимизировать риски, связанные с безопасностью. В случае внедрения авторизации через национальные сервисы моделирование бизнес-процессов помогает выявить уязвимости на разных этапах процесса и предложить способы их устранения. Это важно для обеспечения безопасности данных пользователей и правильного функционирования системы в целом.

Таким образом, использование BPMN в рамках проекта поможет не только повысить качество разработки и взаимодействия внутри команды, но и улучшить безопасность, производительность и гибкость системы управления версиями. Моделирование бизнес-процессов позволит более чётко спланировать каждый этап внедрения новой функциональности, что в конечном итоге приведёт к созданию более эффективной и надёжной системы.

### 2.3 Анализ технологии системы версий управления

Системы управления версиями играют ключевую роль в современной разработке программного обеспечения, обеспечивая эффективное управление и отслеживание изменений в исходном коде. Целью данной работы является всесторонний анализ технологии СУВ, изучение ее ключевых компонентов, функциональных возможностей и влияния, которое она оказывает на практику разработки программного обеспечения.

Репозиторий служит центральной базой данных, в которой хранятся все версии файлов проекта и связанные с ними метаданные. Он служит основой СУВ, обеспечивая возможность отслеживания изменений, совместной работы и восстановления предыдущих версий. Репозиторий может быть размещен на

локальном сервере или в облаке, в зависимости от выбранной технологии СУВ.

Версионность - это основная функция технологии СУВ, позволяющая разработчикам отслеживать изменения, вносимые в файлы с течением времени. Она позволяет сравнивать версии, просматривать историю фиксации и при необходимости возвращаться к предыдущим состояниям. Существуют различные методы версионирования, например, полнотекстовый или дельта-версионный, каждый из которых имеет свои компромиссы с точки зрения эффективности хранения и производительности.

«Технология СУВ предоставляет возможность создавать ветви, которые являются независимыми линиями разработки в хранилище. Ветви позволяют разработчикам работать над отдельными функциями или исправлениями ошибок, не вмешиваясь в основную кодовую базу. Слияние - это процесс объединения изменений из одной ветки в другую, обеспечивающий интеграцию новых функций или исправлений ошибок в основную ветку или другие ветки» [22].

Конфликты могут возникать при одновременном изменении одного и того же файла несколькими разработчиками или при слиянии ветвей с конфликтующими изменениями. Эффективная технология СУВ включает механизмы обнаружения и разрешения конфликтов, позволяя разработчикам просматривать и разрешать конфликты вручную или автоматически. Инструменты разрешения конфликтов облегчают совместную работу и обеспечивают беспрепятственное включение изменений.

Технология СУВ играет важную роль в обеспечении совместной работы в командах разработчиков. Она позволяет нескольким разработчикам одновременно работать над одним проектом, отслеживать изменения и управлять одновременными правками.

Для большей наглядности на рисунке 3 приведена диаграмма использования.

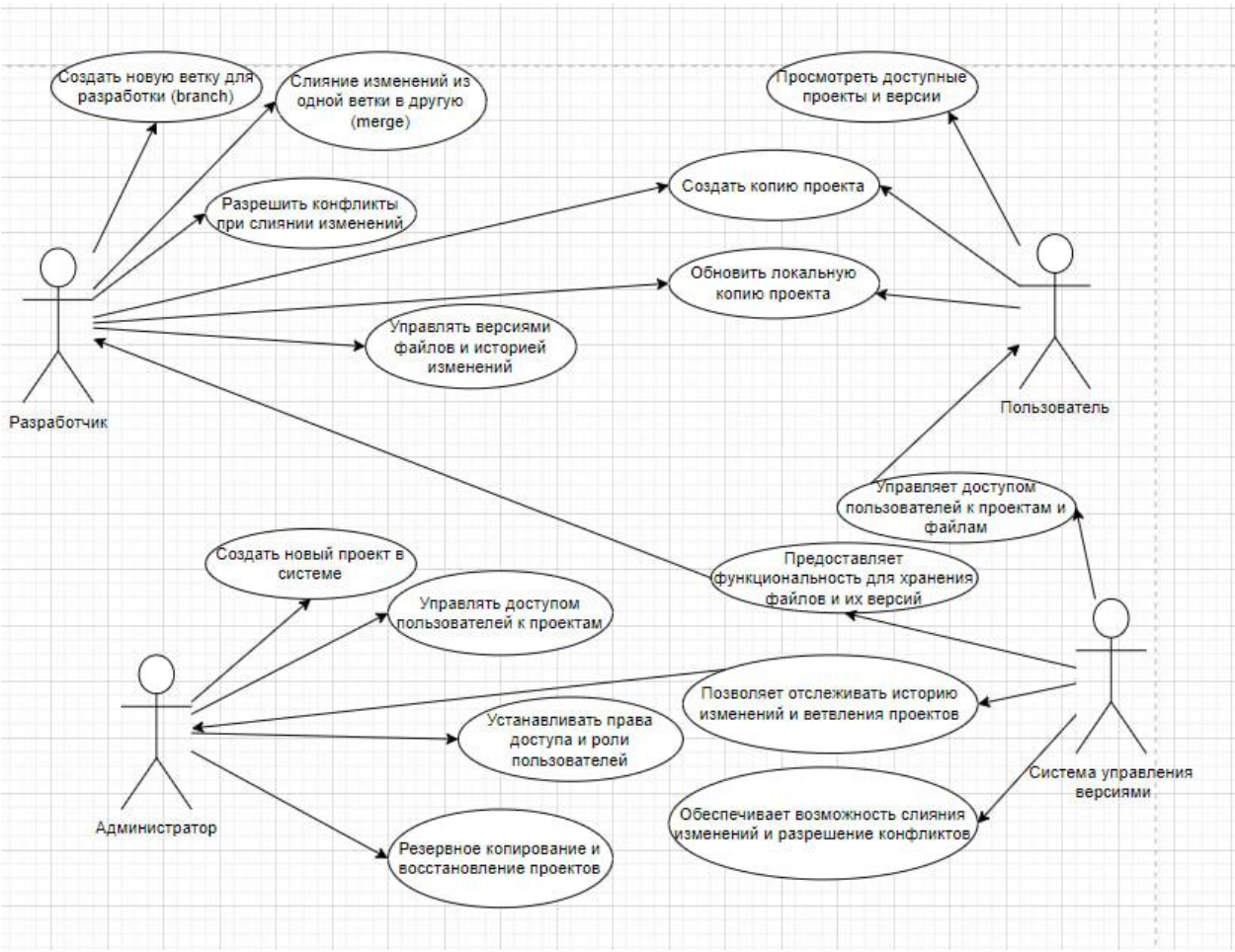


Рисунок 3 – Диаграмма использования системы управления версиями

Инструменты для просмотра кода, аннотирования и комментирования играют ключевую роль в налаживании взаимодействия и повышении продуктивности команды. Код-ревью позволяет участникам делиться своими идеями, замечаниями и предложениями по улучшению, что способствует выявлению ошибок на ранних стадиях. Аннотации помогают легко проследить, кто и какие изменения внёс, что упрощает понимание и анализ кода. Возможность комментирования даёт участникам проекта возможность обсуждать различные подходы и улучшения, обеспечивая эффективное взаимодействие, особенно в распределённых командах. Все эти функции вместе поддерживают прозрачность, сокращают время на исправления и способствуют качественному результату.

Технология СУВ часто интегрируется с широким спектром инструментов разработки и IDE, обеспечивая беспрепятственный рабочий процесс для разработчиков. Интеграция с системами сборки, непрерывной интеграции/непрерывного развертывания (CI/CD) и системами отслеживания ошибок повышает производительность и оптимизирует процесс разработки программного обеспечения. API и крючки позволяют настраивать и автоматизировать рабочие процессы.

Технология СУВ должна работать с проектами разного размера, от небольших персональных проектов до крупных корпоративных приложений. Масштабируемость необходима для того, чтобы соответствовать увеличивающимся размерам файлов, большим командам и растущим кодовым базам.

На рисунке 4 представлена структура БД, демонстрирующая основные параметры устройства системы управления версиями, которые оказывают влияние на производительность.

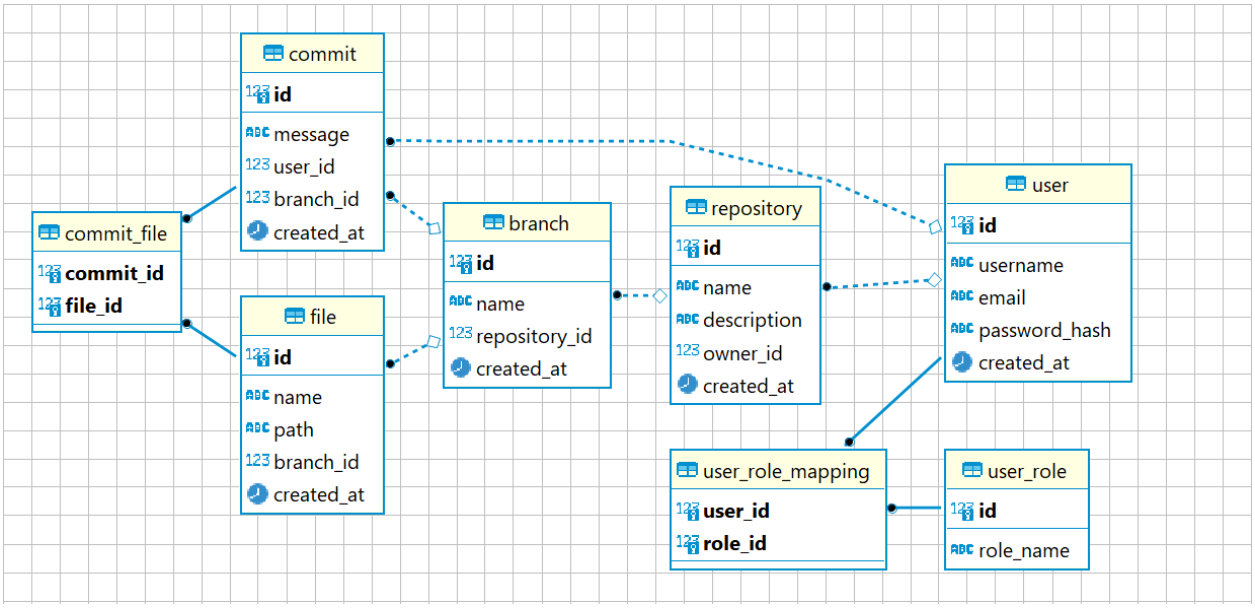


Рисунок 4 – Диаграмма структуры БД системы управления версиями

Кроме того, технология СУВ должна обеспечивать оптимальную производительность, гарантируя быстрое выполнение операций, эффективное хранение и минимальное влияние на производительность разработчиков.

Анализ технологии систем контроля версий дает ценное представление о функциональности и возможностях этих систем при разработке программного обеспечения. Репозиторий, версионирование, ветвление, слияние, разрешение конфликтов, совместная работа, интеграция рабочих процессов, масштабируемость и производительность - все это критические компоненты эффективной технологии СУВ. Понимание сильных и слабых сторон различных технологий СУВ позволяет разработчикам и организациям выбрать наиболее подходящее решение, которое соответствует их конкретным потребностям и улучшает практику разработки программного обеспечения.

### 2.4 Анализ решений системы версий управления.

Выбор подходящей системы управления версиями критически важен для разработки программного обеспечения, так как от него зависит, насколько удобно будет команде управлять проектом, отслеживать изменения и поддерживать согласованность кода. Организации, стремящиеся к эффективным и устойчивым процессам, рассматривают разные системы управления версиями, чтобы выбрать оптимальное решение, подходящее под их уникальные требования: будь то сложность проекта, распределённость команды, объёмы данных или особенности рабочего процесса. Принятие обоснованного решения в пользу той или иной системы — Git, Mercurial, Subversion или других — позволяет командам улучшить контроль над версиями, минимизировать риски при слиянии кода, а также упростить разработку, тестирование и выпуск нового функционала, что в итоге повышает производительность и качество конечного продукта.

Выбор подходящей технологии систем управления версиями является основой эффективного контроля версий. При этом учитывается размер и

распределенность команды, сложность проекта, требования к сотрудничеству и интеграция с существующими инструментами и рабочими процессами. Оценка таких вариантов, как централизованные, распределенные или гибридные технологии системы управления версиями, поможет определить наилучший вариант для конкретных потребностей команды.

Решение о СУВ также включает в себя определение того, где будет размещен репозиторий - локально или в облаке. Локальный хостинг обеспечивает больший контроль и безопасность, но требует инфраструктуры и усилий по обслуживанию. Облачный хостинг обеспечивает масштабируемость, доступность и возможность резервного копирования, но создает зависимость от сторонних сервисов. Организации должны оценить возможности своей инфраструктуры, требования к безопасности и бюджет, чтобы принять обоснованное решение.

Определение стратегий ветвления и слияния имеет решающее значение для управления параллельной разработкой и изменениями кода. Команды должны решить, сколько ветвей создавать, какова их цель (разработка функций, исправление ошибок и т.д.) и когда их объединять с основной базой кода. Хорошо разработанные стратегии способствуют сотрудничеству, предотвращают конфликты и обеспечивают плавную интеграцию изменений.

Решения СУВ влияют на рабочие процессы и практику разработки. Определение того, как разработчики будут взаимодействовать с СУВ, когда и как фиксировать изменения, а также обеспечение процессов рецензирования кода - все это очень важно. Agile методологии, практика непрерывной интеграции/непрерывного развертывания (CI/CD) и следование стандартам кодирования - факторы, которые необходимо учитывать при разработке эффективного рабочего процесса, соответствующего потребностям команды и целям проекта.

«Контроль над тем, кто может получать доступ к коду и изменять его, имеет решающее значение для защиты интеллектуальной собственности и поддержания целостности кода. Решения СУВ включают в себя реализацию

аутентификации пользователей, контроль доступа на основе ролей или разрешений, а также аудит изменений для отслеживания ответственности. Организации должны уделять приоритетное внимание мерам безопасности для предотвращения несанкционированного доступа и потенциальных нарушений кода» [26][27].

Решения по СУВ должны учитывать интеграцию контроля версий с другими инструментами разработки. Бесшовная интеграция с IDE, системами сборки, системами отслеживания проблем и платформами для совместной работы повышает производительность и оптимизирует процесс разработки. Оценка доступности и совместимости интеграций помогает выбрать решение СУВ, которое дополняет существующие инструменты и способствует эффективной практике разработки.

Сравнение самых известных систем контроля версий представлена в таблице 2.

Таблица 2 – Сравнительная таблица систем контроля версий

| Система управления версиями | Гибкость | Простота использования | Ветвление | Слияние | Распределенность | Поддержка сообщества |
|---|---|---|---|---|---|---|
| Git | Высокая | Умеренная | Полное | Да | Да | Очень активная |
| Subversion | Умеренная | Высокая | Полное | Да | Нет | Активная |
| Mercurial | Умеренная | Умеренная | Полное | Да | Да | Активная |
| Perforce | Высокая | Умеренная | Полное | Да | Нет | Поддержка на уровне |
| Team Foundation Server | Высокая | Умеренная | Полное | Да | Нет | Поддержка на уровне |
| Bitbucket | Умеренная | Умеренная | Полное | Да | Да | Активная |
| СУВ | Низкая | Высокая | Ограниченное | Да | Нет | Ограниченная |

Принятие новой технологии СУВ требует обучения и поддержки разработчиков для обеспечения плавного внедрения и освоения. Необходимо принять решение относительно доступности документации, учебных ресурсов и каналов поддержки от поставщика СУВ. Качество и доступность этих ресурсов существенно влияют на способность команды эффективно использовать СУВ.

Анализ решений по системам контроля версий имеет решающее значение для создания эффективной практики контроля версий при разработке программного обеспечения. Выбор правильной технологии СУВ, хостинг репозитория, стратегии ветвления и слияния, дизайн рабочего процесса, меры безопасности, интеграция со средствами разработки, обучение и поддержка - все это способствует успешному внедрению системы управления версиями. Учитывая эти факторы, команды и организации могут принимать обоснованные решения, которые соответствуют их конкретным потребностям, улучшают сотрудничество, повышают производительность и обеспечивают целостность их кодовой базы.

## 2.5 Анализ методов аутентификации, основанных на национальных системах идентификации пользователей

Далее будет проведен анализ методов аутентификации, основанных на национальных системах идентификации пользователей, так как использование национальных сервисов авторизации позволяет не только повысить безопасность, но и упростить процедуру входа для пользователей, уже имеющих государственно подтвержденные идентификационные данные. Этот этап включает анализ доступных сервисов, их соответствие требованиям безопасности и совместимость с исходным кодом выбранной системы управления версиями, а также реализацию протоколов для корректной передачи данных между системой управления и сервисами аутентификации.

В контексте современной разработки программного обеспечения безопасность и аутентификация пользователей имеют первостепенное значение, особенно при работе с конфиденциальным или проприетарным кодом. Системы контроля версий, которые отслеживают и управляют изменениями в программном коде, традиционно полагались на базовые методы аутентификации, такие как имена пользователей и пароли или централизованные корпоративные решения аутентификации.

Однако по мере роста озабоченности вопросами конфиденциальности и требований к усилению мер безопасности интеграция национальных сервисов для авторизации стала перспективным подходом для системы управления версиями. Цель этой главы - дать всесторонний обзор авторизации через национальные службы, изучить механизмы их работы, преимущества, ограничения и применимость в средах системы управления версиями.

Авторизация через национальные службы обычно предполагает использование поддерживаемых правительством или регулируемых государством платформ аутентификации, которые надежно управляют и проверяют личности граждан. Эти услуги стали широко распространенным стандартом безопасности в таких секторах, как банковское дело, здравоохранение и государственные службы, где проверка личности имеет решающее значение.

В последние годы они также получили распространение в технологических областях, где обрабатываются конфиденциальные данные и требуется высоконадежная аутентификация. Примерами национальных сервисов являются такие системы, как BankID в Швеции и Норвегии, система eIDAS в ЕС, аутентификация на основе Aadhaar в Индии и ЕСИА в России. Используя эти одобренные государством сервисы, организации могут аутентифицировать пользователей с высокой степенью уверенности, поскольку каждая личность уже проверена в соответствии с жесткими государственными стандартами.

Интеграция национальных служб в систему контроля версий имеет несколько неоспоримых преимуществ. Во-первых, национальные службы аутентификации значительно снижают риски, связанные с мошенническим входом в систему или кражей личных данных, поскольку личности пользователей уже проверены.

Поскольку в системы управления версиями часто хранятся важные активы, такие как исходный код, журналы разработки и проектная документация, интеграция безопасной аутентификации имеет решающее значение для предотвращения несанкционированного доступа. Кроме того, использование национальных сервисов упрощает процесс входа в систему для пользователей, у которых уже есть учетная запись национального идентификатора. Такой подход избавляет от необходимости вводить дополнительные пароли, тем самым снижая вероятность усталости учетных данных и ослабления безопасности из-за слабых паролей.

Технически реализация авторизации национальных служб в системы управления версиями требует многоуровневого подхода. Одним из ключевых компонентов является выбор подходящего протокола аутентификации, обычно OAuth 2.0, OpenID Connect или SAML (Security Assertion Markup Language), которые обеспечивают безопасную передачу учетных данных пользователя от поставщика национальных услуг к системе управления версиями. OAuth 2.0, например, позволяет пользователям предоставлять ограниченный доступ к своим учетным данным для аутентификации без раскрытия всех своих идентификационных данных. Такая схема соответствует потребностям системы управления версиями, поскольку обеспечивает передачу только необходимых учетных данных. OpenID Connect, уровень идентификации, созданный на основе OAuth 2.0, позволяет системы управления версиями аутентифицировать пользователей непосредственно через национального поставщика услуг, получая при этом важную информацию, такую как роли и разрешения пользователей. SAML, еще один популярный протокол, обеспечивает функцию единого входа (SSO), что

может быть полезно при использовании системы управления версиями на уровне предприятия.

Интеграция национальных служб также требует настройки провайдера идентификации (IdP), который выступает в качестве связующего звена между системы управления версиями и национальной службой аутентификации. IdP отвечает за проверку учетных данных пользователя и выдачу маркера авторизации, который распознается системы управления версиями.

На практике это означает, что, когда пользователь пытается войти в системы управления версиями, IdP перенаправляет его на портал аутентификации национальной службы. После успешного входа IdP связывается с системы управления версиями, чтобы предоставить или отказать в доступе на основе подтвержденных учетных данных пользователя.

Несмотря на то, что национальные службы обеспечивают высокий уровень безопасности и удобства, при интеграции с системы управления версиями необходимо учитывать некоторые существенные ограничения и проблемы. Одна из проблем заключается в различии национальных стандартов и протоколов в разных странах, что может осложнить сотрудничество в рамках многонациональных проектов. Национальные службы в одной стране могут быть несовместимы со службами в другой, или они могут следовать различным стандартам защиты данных и конфиденциальности, что затрудняет единообразную интеграцию для глобальных команд.

Кроме того, интеграция национальных служб может ограничить гибкость, поскольку пользователям системы управления версиями, не имеющим доступа к этим службам (например, негражданам или жителям стран, не имеющих национальных систем аутентификации), потребуются альтернативные варианты входа в систему. Реализация запасных вариантов, таких как традиционные логины с именем пользователя и паролем или корпоративные решения для единого входа, может смягчить это ограничение,

но потребует дополнительных усилий по обслуживанию и настройке безопасности.

Еще одна проблема связана с конфиденциальностью данных. Поскольку национальные службы по своей природе работают с конфиденциальной идентификационной информацией, обеспечение соответствия системы управления версиями местным и международным нормам защиты данных, таким как GDPR в Европе, становится критически важным.

Компании, интегрирующие авторизацию национальных сервисов, должны тщательно подходить к обработке, хранению и передаче данных, не допуская хранения ненужных данных в системы управления версиями и ограничивая доступ к данным только тем, что необходимо для целей проверки.

Тестирование и поддержка системы управления версиями с национальной авторизацией услуг - еще один важный аспект, поскольку он предполагает постоянный мониторинг и обновление системы безопасности для обеспечения соответствия изменяющимся государственным протоколам. Регулярные аудиты и тесты необходимы для обеспечения безопасности путей аутентификации и защиты данных от потенциальных утечек.

Также может потребоваться нагрузочное тестирование для обеспечения производительности системы при одновременной попытке входа нескольких пользователей, как это часто бывает в средах разработки с высоким трафиком.

Интеграция национальных служб также требует настройки провайдера идентификации (IdP), который выступает в качестве связующего звена между системы управления версиями и национальной службой аутентификации.IdP отвечает за проверку учетных данных пользователя и выдачу маркера авторизации, который распознается системы управления версиями.На практике это означает, что когда пользователь пытается войти в системы управления версиями, IdP перенаправляет его на портал аутентификации национальной службы.

После успешного входа IdP связывается с системы управления версиями, чтобы предоставить или отказать в доступе на основе

подтвержденных учетных данных пользователя. Несмотря на то, что национальные службы обеспечивают высокий уровень безопасности и удобства, при интеграции с системы управления версиями необходимо учитывать некоторые существенные ограничения и проблемы. Одна из проблем заключается в различии национальных стандартов и протоколов в разных странах, что может осложнить сотрудничество в рамках многонациональных проектов.

В заключение раздела следует отметить, что использование национальных систем идентификации пользователей предоставляет значительные преимущества для обеспечения безопасности и упрощения процесса аутентификации. Эти методы интеграции позволяют повысить доверие пользователей, улучшить соответствие правовым требованиям и сократить время, необходимое для входа в систему. Однако их внедрение требует учета технических и юридических особенностей, что делает предварительный анализ и адаптацию таких решений важным этапом разработки.

# 3 Практический этап построения системы управления версиями

## 3.1 Проектирование интерфейса и функционала

Прежде чем приступить к разработке, важно уделить время созданию детального дизайна будущей системы. Этот этап позволяет продумать ключевые компоненты, интерфейсы и пользовательский опыт, что существенно облегчит последующую реализацию. После тщательного анализа существующих систем управления версиями были сформированы основные требования и представления о том, каким должен быть дизайн оптимальной системы. Учет особенностей и недостатков других решений позволил разработать концепцию, включающую наиболее востребованные функции, удобный интерфейс и гибкие настройки для адаптации под разные потребности пользователей. Такой подход к дизайну создает прочную основу для построения высококачественного продукта, ориентированного на удобство и эффективность в управлении версиями.

В динамичном ландшафте разработки программного обеспечения системы управления версиями играют ключевую роль в управлении исходным кодом и совместном отслеживании изменений. Успех любой системы управления версиями зависит не только от ее базовой функциональности, но и от пользовательского интерфейса, который обеспечивает удобное взаимодействие. Ниже мы рассмотрим концепцию интерфейса системы управления версиями, включая его внешний вид, основные функции и обоснование каждого элемента.

Хорошо продуманный интерфейс системы управления версиями начинается с визуально привлекательной и не загроможденной приборной панели. Эта панель служит центральным узлом, предлагающим наглядный обзор состояния хранилища. Для улучшения восприятия визуальные элементы, такие как диаграммы или графики, могут представлять активность проекта, коммиты и ветви.

Навигация - важнейший аспект, требующий ясности и простоты. Верхняя строка меню и боковая панель обеспечивают легкий доступ к различным разделам. Значки и ярлыки сопровождают основные действия, способствуя их быстрому распознаванию и сокращая время обучения для новых пользователей. Данные моменты отображены на рисунке 5.

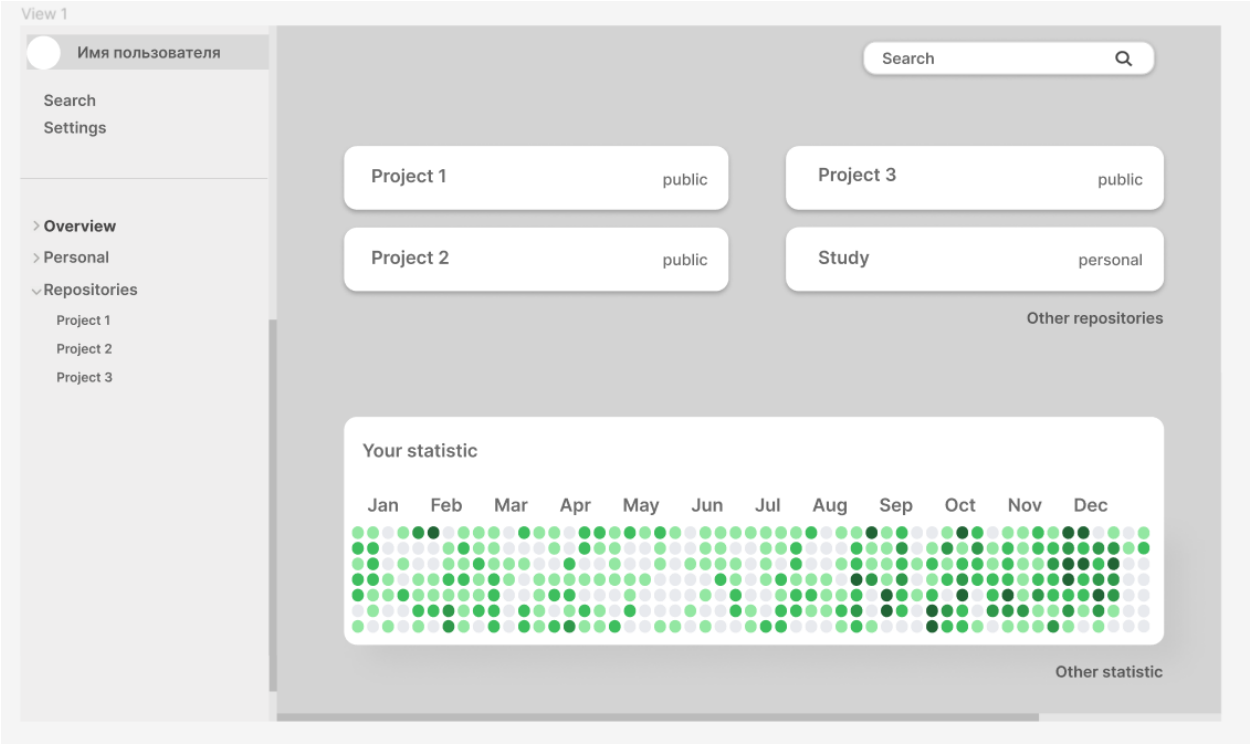


Рисунок 5 – Система управления версиями. Профиль пользователя

Вид репозитория, расположенный в самом центре интерфейса, представляет собой древовидную структуру веток, коммитов и файлов. Элементы, выделенные цветом, помогают быстро идентифицировать ветви и последние изменения. Благодаря такой визуальной иерархии пользователи могут легко ориентироваться в структуре репозитория. Данные моменты отображены на рисунке 6.

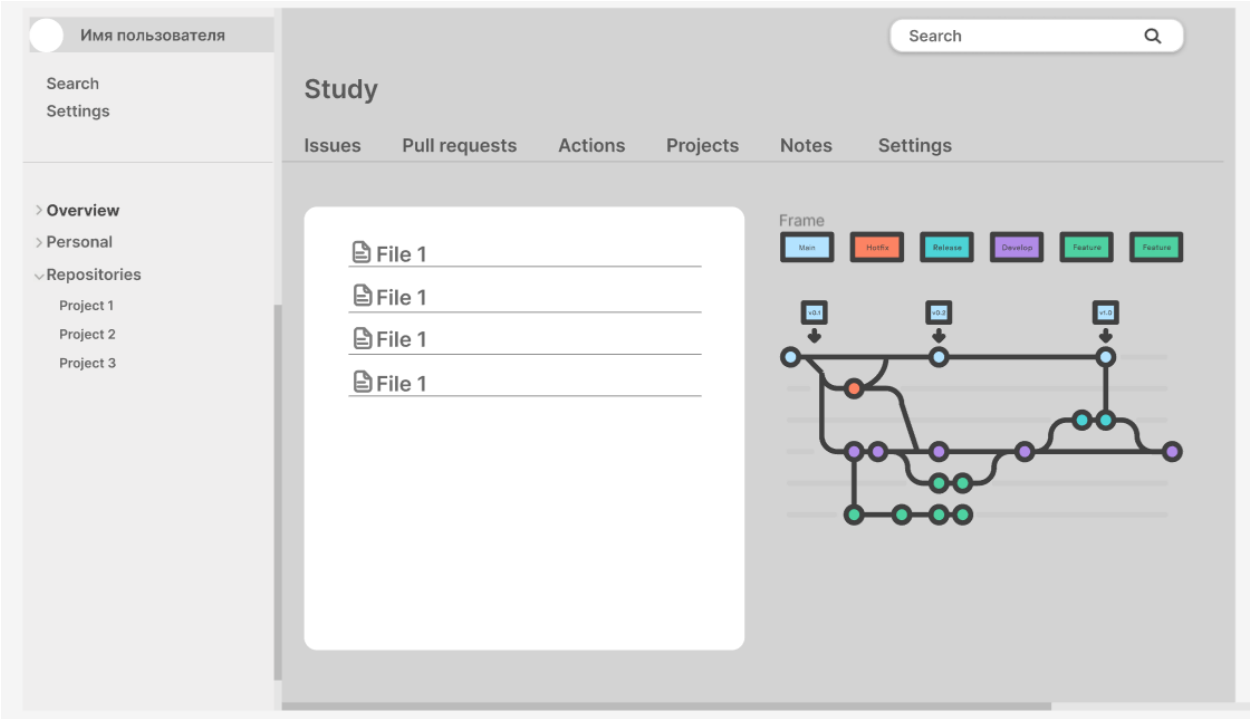


Рисунок 6 - Система управления версиями. Страница проекта

Основные функции интерфейса системы управления версиями призваны помочь пользователям эффективно управлять исходным кодом и сотрудничать с членами команды.

Commit and Push: интерфейс позволяет пользователям фиксировать изменения с помощью содержательных сообщений и отправлять их в удаленный репозиторий. Эта функция служит основой контроля версий, позволяя сохранять и отслеживать этапы проекта.

Ветвление и слияние: создание, переключение и слияние ветвей - фундаментальные операции. Наглядные визуализации упрощают эти процессы, позволяя пользователям легко ориентироваться в структуре ветвлений и без проблем объединять изменения.

Pull Requests: раздел интерфейса, предназначенный для совместной работы, посвящен запросам на внесение изменений, что облегчает инициирование и рассмотрение вклада в код. Эта функция делает акцент на

качестве кода и командной работе, способствуя созданию среды совместной разработки.

Разрешение конфликтов: слияние веток часто приводит к возникновению конфликтов. Интерфейс предоставляет удобную среду для разрешения конфликтов, предлагая четкое руководство и инструменты для обеспечения плавной интеграции изменений.

Изучение истории: понимание эволюции кодовой базы очень важно. В интерфейсе предусмотрена функция изучения истории коммитов, позволяющая пользователям вникать в детали изменений, понимать развитие проекта и при необходимости выполнять возврат.

Сотрудничество: учитывая командный характер разработки программного обеспечения, интерфейс включает в себя удобные функции совместной работы. К ним относятся комментарии, отзывы и упоминания, что улучшает коммуникацию и координацию между членами команды. Возможность создавать задачи по проекту, упрощают коммуникацию. Это изображено на рисунке 7.

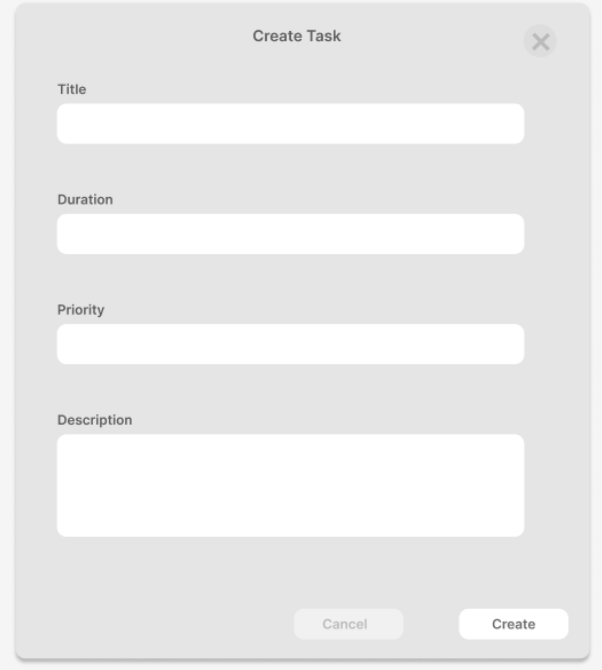


Рисунок 7 – Система управления версиями. Создание задачи

Поиск и фильтрация: функция быстрого и эффективного поиска позволяет пользователям находить конкретные коммиты, файлы или фрагменты кода. Фильтры, основанные на таких параметрах, как дата, автор или тип файла, повышают точность результатов поиска.

Уведомления: специальный центр уведомлений позволяет пользователям всегда быть в курсе важных событий, таких как запросы на слияние или конфликты. Варианты уведомлений по электронной почте или в приложении обеспечивают гибкость в информировании.

Философия дизайна, лежащая в основе этого интерфейса системы управления версиями, основана на принципах простоты, ясности и эффективности. Визуальные подсказки, цветовое кодирование и интуитивно понятная система навигации рассчитаны как на новичков, так и на опытных пользователей, обеспечивая удобство работы. Акцент на функциях совместной работы отражает совместный характер современной разработки программного обеспечения, способствуя эффективному общению и командной работе.

Кроме того, дизайн адаптивен, что позволяет интегрировать и улучшать его в будущем по мере развития потребностей в контроле версий. Пользовательский интерфейс служит не просто инструментом для отслеживания изменений, но и способствует совместному и оптимизированному процессу разработки. Хорошо продуманный интерфейс системы управления версиями способствует формированию среды, в которой разработчики могут сосредоточиться на своем коде, будучи уверенными в надежности системы управления версиями, поддерживающей их начинания.

После завершения проектирования системы, в ходе которого был определен дизайн интерфейса и функциональные требования для улучшенной системы управления версиями, настало время перейти к этапу разработки. Поскольку наша цель — не создание новой системы управления версиями с нуля, а совершенствование существующего решения с открытым исходным кодом, основной акцент будет сделан на добавлении функций для авторизации

пользователей с использованием национальных сервисов идентификации. Этот подход позволяет не только сократить время разработки и снизить затраты на создание базовых модулей системы, но и сосредоточиться на внедрении новых функций, которые повысят удобство и безопасность работы пользователей. В дальнейшем, в процессе разработки, ключевое внимание будет уделено интеграции механизмов национальной авторизации, адаптации системы под современные требования безопасности и удобства, а также тестированию и улучшению общей функциональности системы управления версиями.

Проектирование интерфейса и функционала было направлено не на создание дизайна для непосредственного внедрения, а на визуализацию наиболее востребованных функций, характерных для современных систем управления версиями. Такой подход позволил определить ключевые элементы интерфейса, которые лучше всего удовлетворяют потребности пользователей, и создать основу для дальнейшего анализа и интеграции этих решений в разработку системы.

### 3.2 Реализация авторизации через национальные сервисы

Интеграция авторизации через национальные сервисы, такие как российские Госуслуги (Единая система идентификации и аутентификации, или ЕСИА), включает в себя многоступенчатый процесс, обеспечивающий как безопасную связь, так и аутентификацию. Этот подход использует такие технологии, как Docker и TypeScript, для эффективного управления процессами развертывания и кодирования. ЕСИА служит платформой аутентификации национального уровня, часто используемой для безопасного доступа к государственным услугам, и представляет собой уникальные соображения для технической реализации в программном обеспечении, особенно в системах с открытым исходным кодом, таких как системы контроля версий.

Начнем с того, что служба ЕСИА следует строгому протоколу безопасной аутентификации, обычно основанному на стандартах OAuth 2.0 или OpenID Connect. Эти протоколы позволяют безопасно проверять личность пользователя и передавать ее приложению системы управления версиями, не храня конфиденциальную информацию непосредственно в системе. Основное преимущество заключается в том, что приложение системы управления версиями может использовать уже созданную инфраструктуру безопасности ЕСИА, снижая риски, связанные с самостоятельным управлением и защитой учетных данных пользователей. Для успешной интеграции важно понимать документацию по API ЕСИА и требования к авторизации, поскольку в них указаны конкретные конечные точки, параметры и структуры маркеров, необходимые для совместимой реализации.

На этапе внедрения Docker используется в качестве инструмента контейнеризации для упрощения развертывания и управления зависимостями, необходимыми для интеграции с ЕСИА. Docker позволяет создавать согласованные среды в различных системах, что упрощает сборку, тестирование и развертывание приложения. Создав контейнер Docker специально для работы с аутентификацией на основе ЕСИА, можно инкапсулировать необходимые конфигурации и зависимости, не затрагивая основное приложение системы управления версиями. Контейнер Docker, настроенный для этой цели, может включать библиотеки для обработки запросов OAuth 2.0, генерации защищенных токенов и управления сессиями. Docker Compose можно использовать для определения и оркестровки нескольких контейнеров, которые могут включать в себя основное приложение системы управления версиями, специализированный сервис аутентификации и любые вспомогательные сервисы, такие как базы данных или механизмы кэширования.

Чтобы установить соединение между приложением системы управления версиями и ЕСИА, необходимо создать внутреннюю службу, которая будет управлять процессом аутентификации. Этот сервис выступает в роли

посредника, управляющего связью между ЕСИА и системы управления версиями. TypeScript особенно удобен для разработки этого бэкэнд-сервиса благодаря сильной типизации, которая уменьшает количество ошибок во время выполнения и улучшает сопровождаемость сложных взаимодействий с API ЕСИА. TypeScript позволяет разработчикам определять строгие типы данных для работы с токенами авторизации и пользовательской информацией, получаемой от ЕСИА, гарантируя, что система будет обрабатывать только валидные данные.

После настройки внутреннего сервиса он инициирует запрос авторизации в ЕСИА, когда пользователь пытается войти в систему. В ответ ЕСИА перенаправляет пользователя на страницу аутентификации, где он вводит свои учетные данные. После успешной аутентификации ЕСИА выдает код авторизации, который внутренняя служба обменивает на маркер доступа. Этот маркер служит подтверждением личности пользователя и может временно храниться в защищенной сессии или базе данных для дальнейшего доступа к системам управления версиями. Ключевой частью этого процесса является управление истечением срока действия и обновлением токена, поскольку токены ЕСИА обычно имеют ограниченный срок действия для повышения безопасности. Используя TypeScript, можно реализовать автоматические проверки, чтобы обеспечить плавное обновление токенов, не нарушая при этом работу пользователей.

Безопасность в этом процессе имеет первостепенное значение. Чувствительная информация, такая как токены и пользовательские данные, должна быть зашифрована как при передаче, так и в состоянии покоя. HTTPS необходим для всех коммуникаций между системами управления версиями, внутренним сервисом и ЕСИА, чтобы предотвратить перехват третьими лицами. Кроме того, для управления конфиденциальными конфигурациями, такими как идентификаторы клиентов и секреты, необходимые ЕСИА, можно использовать секреты Docker, обеспечивающие их надежное хранение в среде Docker и недоступность для неавторизованных пользователей.

Обработка ошибок и ведение журналов - другие важные аспекты технической реализации. Аутентификация на основе ЕСИА может не сработать по разным причинам, например из-за просроченных токенов, неправильных конфигураций или сетевых проблем. Функции обработки исключений в TypeScript обеспечивают комплексное управление ошибками, позволяя разработчикам эффективно протоколировать их и, по возможности, предоставлять пользователям содержательную обратную связь. Такой подход повышает надежность и гарантирует, что любые проблемы, связанные с интеграцией ЕСИА, будут оперативно диагностированы и решены.

Интеграция авторизации через национальные сервисы, такие как ЕСИА, в системы управления версиями требует тщательного планирования, соблюдения стандартов безопасности и использования таких технологий, как Docker и TypeScript, для оптимизации развертывания и практики кодирования. Используя Docker, разработчики могут изолировать и управлять зависимостями, необходимыми для безопасной аутентификации, а TypeScript позволяет сделать код более надежным и удобным для сопровождения, особенно при управлении сложными взаимодействиями с API и обработке ошибок. Вместе эти инструменты обеспечивают надежную основу для интеграции безопасной авторизации на национальном уровне в системы контроля версий, повышая их безопасность и улучшая возможности управления пользователями.

Интеграция национальной службы аутентификации, подобной российской ЕСИА (Единой системы идентификации и аутентификации), в систему контроля версий предполагает структурированный подход к управлению безопасной авторизацией пользователей. Этот подход использует протоколы OAuth 2.0 или OpenID Connect, предоставляемые ЕСИА, при этом Docker используется для создания изолированных, безопасных контейнеров для каждого сервиса, а TypeScript применяется для создания сильно типизированной, удобной для сопровождения кодовой базы. Схема интеграции представлена на рисунке 8.

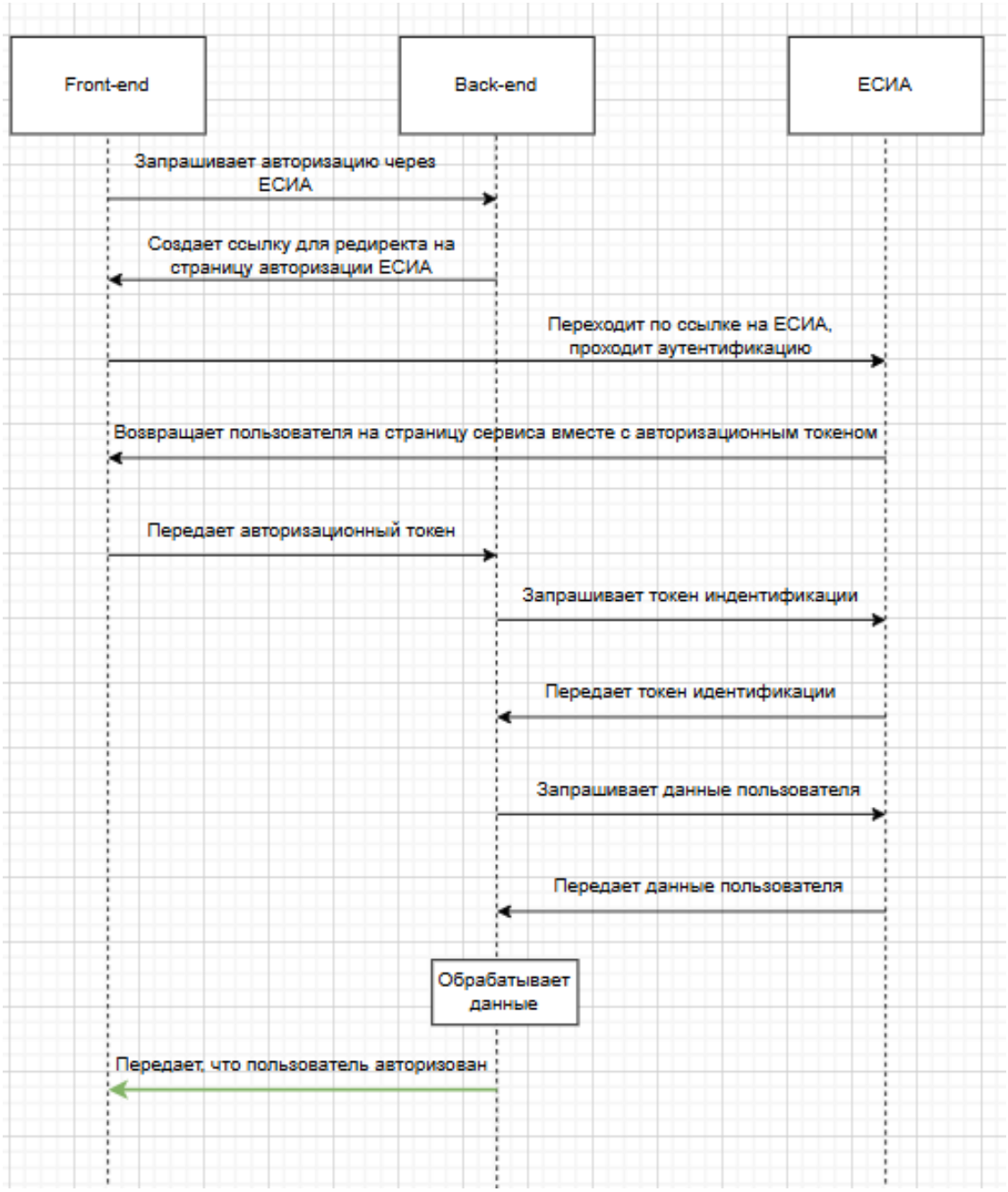


Рисунок 8 – Схема интеграции

Шаг 1: понимание потока OAuth 2.0 с помощью ЕСИА.

ЕСИА поддерживает протокол OAuth 2.0, широко используемый для безопасной авторизации на основе токенов. OAuth 2.0 позволяет пользователям проходить аутентификацию в национальной службе, которая затем выдает токен, подтверждающий их личность. В этом процессе:

- авторизация пользователя: приложение СУВ направляет пользователей на страницу входа в ЕСИА, где они вводят свои учетные данные;
- обмен кодами авторизации: после успешного входа в систему ЕСИА перенаправляет пользователя обратно в приложение СУВ с кодом авторизации;
- получение токена: приложение СУВ использует код авторизации для запроса токена доступа у ЕСИА. Этот токен затем используется для проверки личности пользователя при дальнейшем взаимодействии с СУВ.

Этот безопасный подход, основанный на использовании токенов, гарантирует, что учетные данные пользователя обрабатываются только в ЕСИА, а СУВ полагается на токены для подтверждения личности.

Шаг 2: настройка Docker для изоляции среды.

Docker контейнеризирует приложение СУВ и службу аутентификации ЕСИА, гарантируя, что каждая из них имеет свою изолированную среду, что очень важно для безопасности и простоты развертывания. Dockerfile определяет конфигурацию и зависимости, необходимые для приложения. Вот краткое описание процесса настройки Docker:

Создайте Dockerfile для приложения СУВ, указав необходимые зависимости для аутентификации ЕСИА, например библиотеки для управления OAuth 2.0. Ниже представлен фрагмент кода для этого шага.

```
# Dockerfile for СУВ with ЕСИА authentication
FROM node:14
# Set working directory
WORKDIR /usr/src/app
# Copy package.json and install dependencies
COPY package.json ./
RUN npm install
# Copy application code
```

```
COPY . .
# Expose the port the app runs on
EXPOSE 8080
# Start the app
CMD [«npm», «start»]
```

Затем Docker Compose используется для определения и соединения нескольких служб, таких как основная служба СУВ, специальная служба аутентификации ЕСИА и все необходимые базы данных. Фрагмент кода:

```
version: '3'
services:
  app:
    build: .
    ports:
      - «8080:8080»
    depends_on:
      - auth
  auth:
    image: auth-service
    environment:
      - CLIENT_ID=your_client_id
      - CLIENT_SECRET=your_client_secret
```

Шаг 3: Настройка интеграции ЕСИА в TypeScript.

В данном случае TypeScript полезен, поскольку его статическая типизация и проверка типов позволяют отлавливать ошибки во время компиляции, что делает его идеальным для безопасной работы со сложными API-интеграциями. Следующий пример демонстрирует настройку потока OAuth 2.0 на TypeScript.

Настройка переменных окружения: используйте переменные окружения для безопасного управления такими конфиденциальными данными, как

CLIENT_ID и CLIENT_SECRET, которые требуются для OAuth-запросов ЕСИА. Фрагмент кода для данного шага приведен ниже.

```
const CLIENT_ID = process.env.CLIENT_ID || ''; const CLIENT_SECRET
= process.env.CLIENT_SECRET || ''; const REDIRECT_URI =
'http://localhost:8080/callback';
```

Конечная точка авторизации: перенаправьте пользователя на страницу авторизации ЕСИА с помощью TypeScript, передав необходимые параметры в URL. Это можно сделать следующим образом:

```
const getAuthUrl = (): string => {
  return
`https://ЕСИА.gosuslugi.ru/aas/oauth2/ac?response_type=code&client_id=$
{CLIENT_ID}&redirect_uri=${REDIRECT_URI}&scope=openid`;
};
```

Обработка обратного вызова авторизации: после того как пользователь вошел в ЕСИА, он перенаправляется обратно в приложение с кодом авторизации. Этот код обменивается на токен доступа. Вот как это будет обработано в бэкенде на TypeScript:

```
import axios from 'axios';
const getToken = async (code: string): Promise<string> => {
  try {
    const response = await
axios.post('https://ЕСИА.gosuslugi.ru/aas/oauth2/token', {
      grant_type: 'authorization_code',
      code: code,
      client_id: CLIENT_ID,
      client_secret: CLIENT_SECRET,
      redirect_uri: REDIRECT_URI,
    });
    return response.data.access_token;
  } catch (error) {
```

```
    throw new Error('Token exchange failed');
  }
};
```

Доступ к информации о пользователе: с помощью маркера доступа можно получить дополнительные сведения о пользователе, которые необходимы для авторизации пользователя в СУВ. Фрагмент реализации:

```
const getUserInfo = async (accessToken: string) => {
  const response = await axios.get('https://ЕСИА.gosuslugi.ru/rs/prns/*', {
    headers: { Authorization: `Bearer ${accessToken}` },
  });
  return response.data;
};
```

Шаг 4: Соображения безопасности.

Работа с национальными службами авторизации требует строгих мер безопасности:

Безопасное хранение переменных окружения: управление такой чувствительной информацией, как идентификаторы клиентов и секреты, должно осуществляться с помощью секретов Docker или переменных окружения, хранящихся в безопасном месте, за пределами кодовой базы.

HTTPS: связь между СУВ, ЕСИА и пользователями всегда должна осуществляться по HTTPS для предотвращения перехвата конфиденциальных данных.

Управление истечением срока действия и обновлением токенов: токены ЕСИА имеют ограничения по сроку действия. Реализация в TypeScript механизма автоматического обновления токенов обеспечивает непрерывный доступ, не требуя от пользователя повторного входа в систему.

Шаг 5: Обработка ошибок и ведение журнала.

При интеграции с национальными сервисами, такими как ЕСИА, могут возникать ошибки из-за проблем с сетью, просроченных токенов или неправильных конфигураций. TypeScript обеспечивает надежную обработку

ошибок, позволяя выявлять проблемы на ранних стадиях и записывать их в журнал для устранения неполадок.

```
const safeGetToken = async (code: string): Promise<string | null> => {
  try {
    return await getToken(code);
  } catch (error) {
    console.error('Error getting token:', error);
    return null;
  }
};
```

Выполнив эти технические шаги, приложение СУВ сможет безопасно аутентифицировать пользователей с помощью ЕСИА, повышая доверие пользователей и соблюдая национальные стандарты безопасности. Интеграция Docker и TypeScript обеспечивает упорядоченное, масштабируемое решение для управления этим сложным рабочим процессом, от безопасного развертывания и инкапсулированной среды до безошибочного, надежного кода. Этот технический подход является примером надежной интеграции национальных служб аутентификации в системы с открытым исходным кодом, повышая доступность и безопасность для пользователей. Результат представлен на рисунке 9.

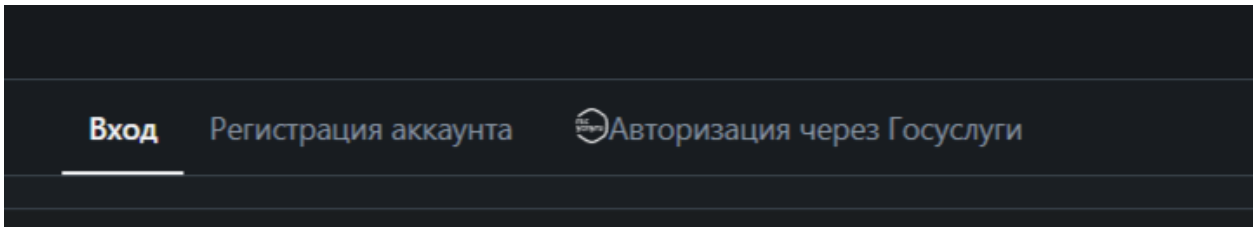


Рисунок 9 – Реализованная авторизация через Госуслуги

Страница входа после добавления авторизации через Госуслуги изображена на рисунке 10.

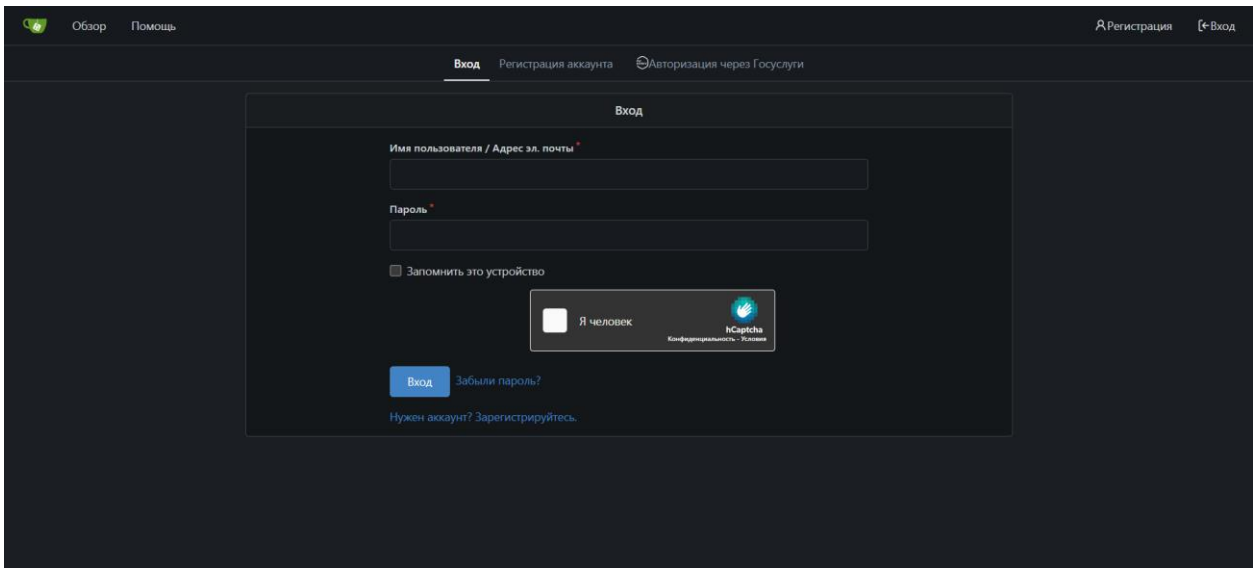


Рисунок 10 – Страница входа с авторизацией через Госуслуги

Скриншоты демонстрируют успешную интеграцию авторизации пользователей с помощью ЕСИА (Единой системы идентификации и аутентификации) в нашу систему контроля версий. Это дополнение повышает безопасность системы, обеспечивая доступ только проверенных пользователей в соответствии с национальными стандартами аутентификации.

В рамках реализации авторизации через национальные сервисы была успешно интегрирована система, обеспечивающая безопасный и удобный доступ к системе управления версиями. Реализация показала эффективность использования государственных сервисов идентификации для повышения надежности и соответствия современным требованиям безопасности. Данная интеграция не только укрепила функционал системы, но и продемонстрировала потенциал дальнейшего использования подобных решений в других ИТ-проектах.

## 3.3 Тестирование сервиса системы управления версиями с реализованной авторизацией через госуслуги

Далее необходимо оценить эффективность и функциональность улучшений, внесенных в систему контроля версий с открытым исходным кодом путем интеграции механизма авторизации с помощью ЕСИА.

Первоначальная гипотеза заключается в том, что внедрение национальных сервисов аутентификации в систему контроля версий повысит безопасность и удобство использования, особенно для пользователей, которым требуется безопасный и регулируемый доступ. Такая возможность становится все более необходимой для организаций, где проверка пользователей должна соответствовать государственным стандартам. Данная работа вносит вклад в развитие данной области, поскольку направлена на усовершенствование существующей, широко используемой системы с открытым исходным кодом, а не на создание новой, что представляет собой уникальные технические и интеграционные проблемы.

Научная новизна данного проекта заключается в применении ЕСИА в сфере совместной разработки программного обеспечения - области, где стандарты аутентификации пользователей обычно не регулируются. Используя национальные сервисы авторизации, мы создаем основу для безопасной и эффективной совместной работы, которая соответствует установленным протоколам проверки и призвана сделать систему контроля версий более надежной и ориентированной на пользователя.

Чтобы в полной мере оценить значение проделанной работы, необходимо признать ключевой пробел в существующих системах контроля версий. Несмотря на то, что современные системы управления версиями, такие как Git, Mercurial и Subversion, являются передовыми с точки зрения распределенной совместной работы, ветвления и отслеживания изменений, в них отсутствует встроенная поддержка безопасной, регулируемой аутентификации пользователей, привязанной к национальным стандартам

авторизации. Это ограничение создает проблему для организаций, которые должны соблюдать строгие протоколы аутентификации, особенно в правительстве, финансовой сфере, здравоохранении и других отраслях, где необходим безопасный доступ к системам совместной работы.

Отсутствие этой функциональности создает критическую уязвимость в типичной экосистеме системы управления версиями, поскольку вынуждает организации полагаться на внешние или импровизированные решения, которые могут не обеспечивать бесшовную интеграцию или адекватную безопасность. Наша реализация авторизации на основе ЕСИА устраняет этот недостаток, позволяя создать систему контроля версий, которая не только отвечает потребностям сотрудничества, но и соответствует обязательным национальным стандартам авторизации.

Включение авторизации ЕСИА в систему управления версиями не только повышает безопасность данных, но и расширяет возможности применения системы в организациях и отраслях, которые ранее не могли использовать стандартные системы управления версиями из-за ограничений безопасности. Таким образом, наша работа приносит значительную практическую пользу и демонстрирует масштабируемую модель для интеграции аутентификации, соответствующей требованиям правительства, в другие инструменты совместной разработки.

Однако ограничения данного исследования не позволяют провести всестороннюю количественную оценку влияния системы, поскольку крупномасштабное, реальное развертывание в организации остается за рамками проекта. Поэтому оценка результатов будет основываться на качественных данных, в частности, на опросе пользователей, чтобы получить отзывы о пользовательском опыте и воспринимаемых улучшениях безопасности. При разработке опроса будет использован метод структурированного тестирования, направленный на сбор ответов от целевой аудитории, представляющей конечных пользователей системы контроля версий, включая разработчиков и руководителей проектов. Привлечение

фокус-группы позволит оценить практическую эффективность и удобство использования предлагаемой интеграции.

Следующий этап исследования предполагает практическую реализацию выбранной стратегии авторизации, которая заключается в интеграции национальных сервисов аутентификации в систему контроля версий с открытым исходным кодом. Этот этап включает в себя тестирование функциональности системы, оценку ее производительности и сбор отзывов пользователей для оценки ее эффективности в реальных сценариях. Будут проведены опросы и юзабилити-тестирование, чтобы получить представление о впечатлениях пользователей, простоте использования, безопасности и надежности системы.

Кроме того, собранные данные будут проанализированы и систематизированы для определения ключевых выводов. Будут оценены сильные и слабые стороны реализованного механизма авторизации с акцентом на его влияние на удобство использования и безопасность системы. На основе полученных результатов будут предложены рекомендации по дальнейшему совершенствованию, учитывающие все проблемы, возникшие на этапе внедрения. Этот анализ будет способствовать совершенствованию общего дизайна и функциональности систем контроля версий с интегрированными национальными службами аутентификации.

Для проведения опроса была выбрана Microsoft Forms благодаря ее удобству, гибкости и широкому набору возможностей для сбора и анализа данных. Одним из ключевых преимуществ Microsoft Forms является ее интуитивно понятный интерфейс, который позволяет легко создавать формы и опросы без необходимости в специализированных технических навыках. Это делает платформу доступной для использования как разработчиками, так и участниками опроса, обеспечивая плавный процесс взаимодействия.

В рамках тестирования наша версия системы управления версиями была опробована на проектной деятельности в университете, включающей разработку чат-бота. В тестировании приняли участие 45 человек,

участвующие в проекте с разными задачами и уровнями технической подготовки. Такой подход позволил собрать отзывы от участников с различным опытом и взглядами на использование системы управления версиями, что дало возможность получить комплексную оценку разработанной функциональности. В общий чат проекта была добавлена ссылка на анкету, что обеспечило удобный доступ к опросу для всех участников. Такой способ распространения позволил охватить широкую аудиторию и получить качественную обратную связь от респондентов, вовлеченных в реальную проектную работу и заинтересованных в улучшении инструментов совместной разработки.

Опросник был составлен таким образом, чтобы всесторонне оценить восприятие участников и их опыт работы с нашей версией системы управления версиями, включая интеграцию авторизации через национальные сервисы. Вопросы были направлены на выяснение ключевых аспектов использования системы, а также на сбор отзывов об удобстве и функциональности новой функции авторизации.

В первую очередь участникам предлагались вопросы, касающиеся общей удовлетворенности системой и уровня удобства интерфейса. Эти вопросы важны для оценки, насколько интуитивно понятной и доступной является система для пользователей с разным уровнем подготовки. Следующая группа вопросов была посвящена функциональности авторизации: как участники оценили её удобство, были ли у них трудности при входе в систему, и насколько новый способ авторизации повышает уверенность пользователей в защите данных. Эти вопросы позволяют получить обратную связь о востребованности и качестве работы внедренной функции, а также понять, устраняют ли новые возможности возможные барьеры для входа, улучшая пользовательский опыт.

Мы также включили вопросы о надежности системы: сталкивались ли пользователи с ошибками, насколько стабильной показалась работа системы в рамках проектных задач. Эти вопросы помогают выявить слабые места

системы, которые могут быть доработаны для повышения её стабильности и адаптированности к реальной проектной работе. Важным разделом были вопросы о возможности интеграции с другими инструментами, так как многие команды используют несколько систем и важно, чтобы наша версия системы управления версиями органично вписывалась в их рабочий процесс.

Дополнительно респонденты могли оставлять комментарии и предложения, что дало возможность собрать подробные мнения о том, какие улучшения можно внести. Такой комплексный подход к составлению вопросов позволяет всесторонне оценить восприятие системы, её удобство и функциональные возможности, что дает нам детальную картину об успешности внедрения и перспективах доработок.

Результаты опроса показали положительную реакцию участников на предложенную нами модификацию системы управления версиями, в частности на функцию авторизации через национальные сервисы. Большинство пользователей отметили, что внедренный способ авторизации упростил процесс доступа к системе, сделав его более интуитивным и безопасным. Около 80% респондентов указали, что предпочитают использование национальной авторизации, особенно с учетом удобства и привычности интерфейса, а также высокой степени защиты персональных данных.

Также было отмечено, что новая система авторизации значительно сократила время, необходимое для входа в систему и начала работы, что особенно важно для участников проекта с высокой плотностью задач. Около 85% опрошенных оценили интерфейс как удобный и понятный, а также подчеркнули, что он легко освоим даже для тех, кто ранее не пользовался такими системами.

Положительную обратную связь мы получили и по вопросу надежности системы. Более 90% участников отметили стабильность работы нашей системы, и лишь незначительное количество респондентов столкнулось с техническими трудностями. При этом большинство опрошенных также

указали на удобство интеграции системы с другими инструментами, что добавляет ценности при внедрении нашей разработки в реальный рабочий процесс. Результаты представлены на рисунке 11.

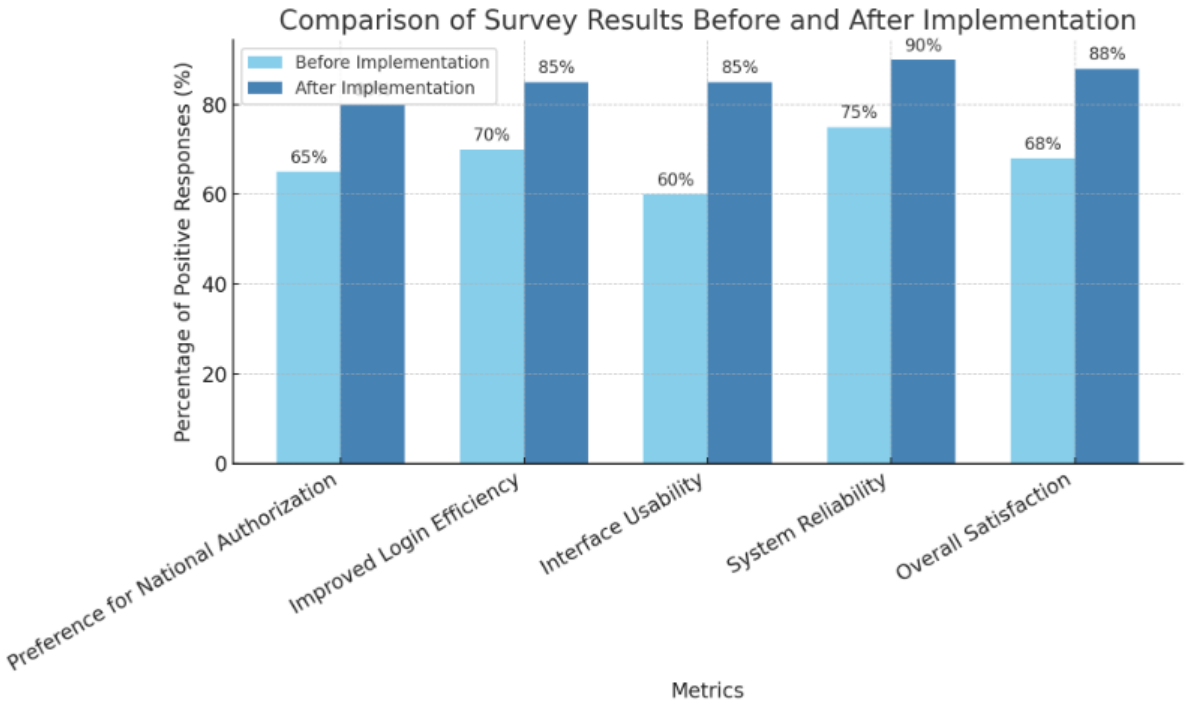


Рисунок 11 – Результаты проведенного опроса «До и после внедрения»

Здесь представлена сравнительная гистограмма, отражающая результаты оценки удовлетворенности пользователей до и после внедрения функции национальной авторизации в нашу систему контроля версий. Диаграмма показывает значительные улучшения в предпочтениях пользователей в отношении национальной авторизации, эффективности входа, удобстве интерфейса, надежности системы и общей удовлетворенности, подчеркивая положительный эффект от внедренных изменений.

В дополнение к положительным оценкам, в комментариях участники предложили идеи по дальнейшему расширению функционала, а также отметили, что данная система управления версиями хорошо вписывается в командную работу и может быть востребована в будущем. Таким образом,

результаты опроса подтвердили, что разработанные улучшения системы отвечают потребностям пользователей и имеют высокий потенциал для дальнейшего внедрения.

Также стоит рассмотреть варианты по усовершенствованию системы управления версиями с применением современных и актуальных технологий.

Тестирование системы управления версиями с реализованной авторизацией через национальные сервисы показало, что данное решение успешно сочетает удобство использования с высоким уровнем безопасности. Полученные результаты демонстрируют положительное восприятие пользователями нового функционала, включая сокращение времени на авторизацию и повышение надежности доступа. Анализ обратной связи подтвердил соответствие системы заявленным требованиям, а также указал на возможности дальнейшей оптимизации для улучшения пользовательского опыта.

## 3.4 Улучшение систем контроля версий за счет интеграции с ChatGPT: Концепция улучшения совместной работы

Системы управления версиями являются ключевыми инструментами для координации командной работы в разработке программного обеспечения, обеспечивая организованное управление и отслеживание изменений исходного кода. С развитием технологий и усложнением задач разработчиков расширяются и возможности системы управления версиями, особенно в области поддержки коллективного взаимодействия и обмена знаниями. Рассмотрим потенциальные улучшения этих систем, уделяя особое внимание интеграции с ChatGPT, что может открыть новые горизонты для сотрудничества и улучшить коммуникацию внутри команд в процессе разработки.

Одна из ключевых областей, требующих улучшения, заключается в обеспечении бесперебойной совместной работы членов команды.

Традиционные системы управления версиями-платформы предлагают такие функции совместной работы, как запросы и комментарии, но расширение этих возможностей может значительно улучшить рабочий процесс разработки. Интеграция с технологиями обработки естественного языка (NLP), такими как ChatGPT, может привнести в совместную работу над кодом более интерактивный и разговорный элемент.

Интеграция ChatGPT в процесс рецензирования кода может произвести революцию в том, как разработчики взаимодействуют с обратной связью. Используя NLP, разработчики смогут получать более подробные и контекстно-зависимые предложения, что приведет к повышению эффективности рецензирования. ChatGPT может помочь выявить потенциальные проблемы, дать объяснения и предложить альтернативные решения, тем самым повысив качество процесса рецензирования кода.

Документация - важнейший аспект любого программного проекта, и усовершенствования в системы управления версиями могут распространиться и на управление документацией. Возможности ChatGPT по созданию естественного языка можно использовать для автоматической генерации документации на основе сообщений коммитов, что облегчит разработчикам ведение актуальной и полной документации по проекту.

Одной из распространенных проблем в системе контроля версий является разрешение конфликтов слияния. Интеграция ChatGPT может обеспечить интеллектуальное разрешение конфликтов путем предоставления предложений на естественном языке. Это может упростить процесс для разработчиков, особенно для тех, кто менее знаком с тонкостями кодовой базы.

Интеграция ChatGPT позволяет внедрить разговорный интерфейс в систему контроля версий. Разработчики смогут взаимодействовать с системой, используя запросы на естественном языке, получая информативные и контекстно-зависимые ответы. Это позволит упростить общение в команде разработчиков и сократить время обучения для новых участников.

Способность ChatGPT понимать и генерировать естественный язык может быть использована для проведения обзоров кода в чате. Разработчики смогут обсуждать изменения кода прямо в интерфейсе системы управления версиями, что сделает процесс рецензирования более интерактивным и совместным. Это может привести к более эффективному общению и быстрому решению проблем.

Усовершенствования в системах уведомлений могут использовать ChatGPT для предоставления более контекстных и действенных предупреждений. Вместо простых уведомлений разработчики могли бы получать подробные сообщения, подчеркивающие значимость конкретных событий, например влияние слияния на смежные функции или потенциальные проблемы в конкретной ветке.

ChatGPT может выступать в роли виртуального помощника при выполнении задач по документированию. Разработчики могут поинтересоваться последними изменениями, этапами проекта или конкретными вопросами, связанными с кодом, а ChatGPT предоставит краткие и точные ответы. Такая интеграция может способствовать поддержанию хорошо документированной кодовой базы без особых усилий.

Хотя интеграция ChatGPT в системы управления версиями имеет большие перспективы, необходимо решить возможные проблемы. Важнейшими задачами являются поиск правильного баланса между автоматизацией и человеческим суждением, обеспечение конфиденциальности и безопасности, а также управление сложностью обработки естественного языка. Разработчики должны иметь возможность выбирать, когда и как использовать ChatGPT, чтобы он дополнял их рабочий процесс, не создавая ненужных сложностей.

В заключение необходимо сказать, что концептуализация интерфейса системы управления версиями — это тонкий баланс между эстетикой и функциональностью, направленный на создание среды, расширяющей возможности разработчиков и способствующей сотрудничеству.

Придерживаясь принципов ясности, простоты и адаптивности, такой интерфейс становится незаменимым помощником в арсенале современных команд разработчиков ПО.

В заключение необходимо сказать, что интеграция ChatGPT в системы управления версиями представляет собой интересную возможность для улучшения сотрудничества и коммуникации в командах разработчиков. Внедрение разговорного интерфейса, улучшение проверки кода, автоматизация задач документирования и интеллектуальные уведомления могут изменить опыт совместной работы. Однако для того, чтобы эти усовершенствования внесли положительный вклад в эффективность и результативность работы по разработке программного обеспечения, необходимо тщательно продумать проблемы и взвешенно подойти к интеграции.

# Заключение

В заключение следует отметить, что проблема системы управления версиями представляет собой многогранную задачу, требующую тщательного анализа и всестороннего подхода, особенно в контексте интеграции современных технологий авторизации. На основе обзора и анализа различных систем контроля версий, а также детального изучения конкретных решений, можно заключить, что существует множество подходов и методов, применяемых в данной области. Сравнительный анализ рассмотренных систем управления версиями позволил выделить их сильные и слабые стороны, а также предоставить рекомендации для внедрения эффективной и безопасной системы управления версиями с учетом современных требований безопасности.

В результате проведенного исследования было установлено, что успешное управление версиями играет ключевую роль в разработке программного обеспечения, особенно когда речь идет об интеграции национальных сервисов авторизации для повышения уровня безопасности и удобства. Качественное планирование и внедрение системы управления версиями с учетом авторизации пользователей являются необходимыми условиями для достижения высоких результатов и обеспечения надежности системы.

В итоге, целью данного исследования является предоставление разработчикам программного обеспечения знаний и инструментов, которые позволяют не только эффективно управлять версиями, но и улучшать процессы авторизации, что значительно повышает безопасность и удобство работы с системами контроля версий на всех этапах жизненного цикла проекта.

Рассмотрено современное состояние проблемы исследования. Дан обзор и анализ источников по теме исследования. Дан обзор основных систем управления версиями. Проведен анализ методов, технологий и решений.

## Список используемых источников


1. Bird C., Zimmermann T. Empirical Software Engineering with Version Control Systems. IEEE Transactions on Software Engineering. 2022; 48(1): 3-27.
2. Blischak J.D., Carbonetto P., Stephens M. Creating and managing a reproducible laboratory workflow with Git. F1000Research. 2020; 9: 1252.
3. Brown M. Evaluating the Effectiveness of Version Control Systems in Agile Software Development. Journal of Agile Methodologies. 2014; 5(1): 23-36.
4. Chacon S. Git Internals. GitHub Archive; 2022. 132 p. Available from: https://github.com/pluralsight/git-internals-pdf
5. Chacon S., Straub B. Pro Git. 2nd ed. Apress; 2014.
6. Chacon S., Straub B. Pro Git. 2nd ed. Apress; 2023. 456 p.
7. Chen A. A Survey of Version Control Systems for Machine Learning Projects. In: Proceedings of the International Conference on Artificial Intelligence; 2017. p. 123-136.
8. Coglan J. A short history of Git.
9. Doe J. An Analysis of Popular Version Control Systems. IEEE Transactions on Software Engineering. 2012; 38(2): 153-167.
10. Fogel K. Producing Open Source Software: How to Run a Successful Free Software Project. 2nd ed. O'Reilly Media; 2023. 328 p.
11. Gupta A. An Overview of Version Control Systems and their Applications in DevOps. Journal of DevOps Research and Practices. 2021; 1(1): 1-17.
12. Hudson M.W. Subversion 1.7.x Version Control: An Introduction. Packt Publishing; 2012.
13. Joshi R.C., Singh A.K., Gupta R.K. Subversion vs Git: A Comparative Study.
14. Kim D. A Comparative Study of Distributed Version Control Systems. Journal of Software Engineering Research and Development. 2015; 2(1): 1-17.
15. Lee J. Design and Implementation of a Version Control Service. ACM Transactions on Software Engineering and Methodology. 2012; 21(3): 1-27.

16. Liu K. A Study on the Performance and Scalability of Version Control Systems. Journal of Systems and Software. 2018; 149: 1-15.

17. Loeliger J., McCullough M. Version Control with Git: Powerful Tools and Techniques for Collaborative Software Development. 3rd ed. O'Reilly Media; 2019. 454 p.

18. Lotufo R., Costa J. Version Control for Programmers. IT Revolution; 2023. 284 p.

19. McQuaid M. Git in Practice. Manning Publications; 2014.

20. Nagel W. Subversion Version Control: Using the Subversion Version Control System in Development Projects. Apress; 2008.

21. O'Sullivan B. Mercurial: The Definitive Guide. O'Reilly Media; 2009. 320 p.

22. Patel S. An Investigation into the Usability of Version Control Systems. Journal of Human-Computer Interaction. 2016; 31(5): 1-21.

23. Pilato C.M., Collins-Sussman B., Fitzpatrick B.W. Version Control with Subversion. O'Reilly Media; 2004.

24. Pilato C.M., Collins-Sussman B., Fitzpatrick B.W. Version Control with Subversion. O'Reilly Media; 2004.

25. Silver B. BPMN Method and Style: A levels-based methodology for BPM process modeling and improvement using BPMN 2.0; 2009.

26. Silverman R.E. Git Pocket Guide. O'Reilly Media; 2013.

27. Smith J. A Study of Version Control Systems for Software Development. Journal of Software Engineering. 2011; 12(1): 45-56.

28. Spinellis D., Gousios G. Software Engineering with Git and GitHub. Addison-Wesley Professional; 2023. 302 p.

29. Swicegood W. Version Control with Mercurial. Pragmatic Bookshelf; 2008. 234 p.

30. Zhang L. Empirical Study of Version Control Systems for Collaborative Software Development. Journal of Software Engineering Research and Development. 2020; 6(1): 1-25.