Введение

В современном мире компании сталкиваются с необходимостью безопасного хранения и управления конфиденциальными данными, такими как пароли, токены доступа и криптографические ключи. Утечка такой информации может привести к серьезным последствиям, включая финансовые потери, репутационный ущерб и юридические санкции. Ручное управление секретами или использование устаревших методов, таких как хранение данных в текстовых файлах, увеличивает риски утечек и затрудняет контроль доступа.

Даже актуальные решения, такие как HashiCorp Vault [1] или AWS Secrets Manager [2], часто оказываются сложными в настройке и интеграции, требуют значительных ресурсов и не всегда предоставляют достаточную гибкость для адаптации под конкретные бизнес-задачи. Это создает потребность в более простых, но при этом надежных и масштабируемых решениях, которые могли бы обеспечить безопасное хранение данных, гибкое управление доступом и удобство использования для сотрудников.

Цели и задачи:

Цель работы – разработка веб-приложения, предоставляющего централизованное решение для безопасного хранения и управления корпоративными секретами. Оно будет использовать ролевую модель доступа (RBAC [3]), интеграцию с Keycloak [4] для аутентификации и многофакторную аутентификацию (MFA [5]). Уникальной особенностью является использование изолированных рабочих пространств, где пользователи могут гибко управлять правами доступа к секретам.

Основные задачи разработки включают реализацию следующей функциональности:

- Управление секретами: Создание, редактирование и удаление конфиденциальных данных (пароли, токены, ключи) с настройкой срока действия.
- Ролевая модель доступа (RBAC): Гибкое управление правами пользователей в рамках рабочих пространств.
- Рабочие пространства: Изолированные области для хранения и управления секретами с возможностью добавления пользователей и назначения ролей.
- Интеграция с Keycloak: Авторизация и аутентификация пользователей через единую систему.
- Журнал аудита: Логирование всех действий пользователей для обеспечения прозрачности и безопасности.
- Уведомления через Telegram-бота [6]: Оперативное информирование о событиях, таких как изменения в секретах или истечение их срока действия.

- Шифрование данных: Защита конфиденциальной информации с использованием AES-256.

1. Глава 1. Предметная область и существующие решения

1.1. Описание предметной области

В современном мире, где информационные технологии играют ключевую роль в бизнесе и повседневной жизни, безопасность данных становится одной из самых актуальных проблем. Корпоративные секреты, такие как пароли, SSH-ключи, токены доступа и сертификаты, являются критически важными элементами, обеспечивающими функционирование ИТ-инфраструктуры. Утечка таких данных может привести к серьезным последствиям, включая финансовые потери, репутационный ущерб и даже юридические санкции. В связи с этим, управление корпоративными секретами становится важной задачей для любой организации, стремящейся минимизировать риски и обеспечить безопасность своих данных.

Однако управление секретами вручную или с использованием устаревших методов (например, хранение паролей в текстовых файлах) сопряжено с множеством вызовов. Это и сложность контроля доступа, и риск утечек, и трудности с аудитом и мониторингом. В условиях растущих киберугроз и ужесточения требований к защите данных, компании нуждаются в современных инструментах, которые помогут им эффективно управлять секретами и обеспечивать их безопасность.

Для пользователей, таких как ИТ-администраторы и сотрудники компаний, приложение предоставляет удобный и безопасный способ управления корпоративными секретами. Оно упрощает процесс хранения и доступа к конфиденциальной информации, снижая риск ошибок и утечек. Ролевая модель доступа позволяет администраторам гибко настраивать права для различных пользователей и команд, что обеспечивает соблюдение принципа минимальных привилегий. Кроме того, интеграция с системами аутентификации, такими как Keycloak, и поддержка многофакторной аутентификации (MFA) повышают уровень безопасности.

Для компаний, которые будут использовать это приложение, оно предоставляет не только удобство, но и уверенность в безопасности своих данных. Приложение позволяет минимизировать риски утечек, упростить управление учетными данными и обеспечить соответствие требованиям внутреннего и внешнего аудита. Ведение журналов аудита и возможность экспорта логов позволяют компаниям оперативно реагировать на инциденты и анализировать действия пользователей.

Кроме того, приложение обеспечивает баланс между простотой использования и индивидуальностью. Оно предлагает гибкие настройки, такие как создание рабочих пространств, настройка прав доступа и интеграция с Telegram-ботом для уведомлений, что делает его удобным для различных сценариев использования. При этом оно сохраняет высокий

уровень безопасности, что особенно важно для компаний, работающих с конфиденциальной информацией.

В современном контексте, где информационная безопасность становится критически важной, такие инструменты играют ключевую роль в обеспечении устойчивости бизнеса. Они помогают компаниям минимизировать риски, связанные с утечками данных, и обеспечивают соответствие требованиям регуляторов. Таким образом, данное приложение не только решает актуальные проблемы, но и способствует формированию культуры безопасности в организациях, что делает его ценным инструментом в современной ИТ-инфраструктуре.

1.2. Описание существующих решений

На рынке уже существует множество решений, таких как HashiCorp Vault, AWS Secrets Manager, 1Password [7] и другие, которые предлагают различные подходы к хранению и защите секретов. Однако каждое из этих решений имеет свои ограничения, будь то сложность настройки, высокая стоимость или недостаточная гибкость интеграции с корпоративными системами. В данной части я проведу сравнительный анализ моего приложения с ключевыми конкурентами, чтобы выделить его преимущества и определить нишу, в которой оно может быть наиболее востребованным.

1.2.1. Описание приложения HashiCorp Vault

HashiCorp Vault — это комплексное, но гибкое приложение для хранения и управления секретами. Взаимодействие с ним осуществляется через веб-интерфейс, API или командную строку. HashiCorp Vault предназначено для безопасной работы с конфиденциальными данными, такими как учетные данные, ключи API, сертификаты и токены доступа. Приложение широко используется в корпоративной среде, обеспечивая централизованный контроль и защиту секретов.

Для работы с HashiCorp Vault пользователи могут зарегистрироваться и войти в систему через различные методы аутентификации. Поддерживаются интеграции с LDAP, OAuth (например, Keycloak), JWT [8], GitHub и другими провайдерами, что позволяет использовать существующую инфраструктуру безопасности. Доступ к хранилищу строго регулируется с помощью ролей и политик (RBAC), что позволяет гибко управлять правами пользователей и сервисов.

Внутри HashiCorp Vault пользователи могут создавать изолированные пространства (namespaces) для хранения секретов. Добавление и управление секретами происходит через веб-интерфейс, API или CLI, где можно задать параметры, такие как имя, значение, время жизни (TTL) и метаданные. HashiCorp Vault поддерживает автоматическую ротацию ключей, обеспечивая дополнительный уровень безопасности.

Доступ к секретам можно получить через API-запросы или интерфейс приложения. Однако, чтобы предотвратить несанкционированное использование, HashiCorp Vault позволяет выдавать временные токены, которые автоматически истекают через заданный промежуток времени. Политики доступа гибко настраиваются, позволяя ограничивать права пользователей на чтение, изменение или удаление данных.

Помимо базового хранения секретов, HashiCorp Vault предоставляет дополнительные функции, такие как динамическая выдача учетных данных. Это означает, что приложение может создавать временные пароли для баз данных или облачных сервисов, которые автоматически удаляются после использования. Также поддерживается встроенное шифрование и дешифрование данных, что позволяет безопасно передавать информацию между сервисами без хранения открытых ключей.

Для бизнеса HashiCorp Vault предлагает расширенные возможности, включая интеграцию с облачными платформами AWS, Azure и GCP. Организации могут развертывать приложение в режиме высокой доступности, распределяя нагрузку между несколькими узлами. Для дополнительной защиты можно использовать аппаратные модули безопасности (HSM) для хранения криптографических ключей.

Таким образом, HashiCorp Vault — это надежное решение для защиты секретов, обеспечивающее централизованное управление доступом и безопасность конфиденциальных данных. Оно подходит как для небольших команд, так и для крупных организаций, которым необходим высокий уровень контроля и защиты информации.

1.2.2. Описание приложения AWS Secrets Manager

AWS Secrets Manager — это облачный сервис для безопасного хранения и управления конфиденциальными данными, такими как пароли, ключи API, учетные данные для баз данных и другие секреты. Сервис интегрирован с экосистемой AWS и предоставляет удобный способ автоматизации ротации секретов, аутентификации и доступа к защищенным данным.

AWS Secrets Manager поддерживает несколько методов аутентификации, включая AWS Identity and Access Management [9] (IAM), что позволяет точно настраивать права доступа для пользователей и сервисов. С помощью IAM-политик администраторы могут контролировать, кто может создавать, просматривать, изменять и удалять секреты. Для дополнительной безопасности можно настроить многофакторную аутентификацию (MFA).

Пользователи могут создавать и управлять секретами через AWS Management Console, API или командную строку (AWS CLI). Каждый секрет включает такие параметры, как имя, значение, версии, а также опциональные теги для организации ресурсов. AWS Secrets Manager

автоматически шифрует данные с использованием AWS Key Management Service (KMS), обеспечивая защиту на уровне облачной инфраструктуры.

Получить доступ к секрету можно с помощью API-запросов или SDK, что позволяет легко интегрировать его с приложениями и сервисами. AWS Secrets Manager поддерживает автоматическую ротацию учетных данных для баз данных AWS, включая Amazon RDS, Aurora и Redshift. Это означает, что сервис может периодически изменять пароли без вмешательства пользователя, обновляя их в базах данных и в самом хранилище.

Дополнительно AWS Secrets Manager поддерживает возможность использования кросс-аккаунтного доступа, что полезно для организаций с несколькими AWS-аккаунтами. Это позволяет централизованно управлять секретами и безопасно предоставлять доступ разным командам или сервисам.

Таким образом, AWS Secrets Manager — это надежное облачное решение для управления конфиденциальными данными, обеспечивающее безопасность, автоматизацию и удобную интеграцию с сервисами AWS. Оно идеально подходит для организаций, которые хотят централизованно управлять секретами и минимизировать риски, связанные с утечками данных.

1.2.3. Описание приложения 1Password

1Password — это кроссплатформенное приложение для хранения и управления паролями, конфиденциальными данными и другими секретами. Оно предназначено как для индивидуального, так и для корпоративного использования, обеспечивая надежную защиту учетных данных с помощью сквозного шифрования. Приложение доступно на мобильных устройствах, компьютерах и в виде веб-версии, а также поддерживает интеграцию с браузерами.

Для использования 1Password пользователи создают учетную запись и проходят двухфакторную аутентификацию (2FA) для дополнительной защиты. Вход в систему возможен с помощью мастер-пароля и специального секретного ключа, который генерируется при регистрации и хранится только у пользователя. Организации могут управлять доступом сотрудников через админ-панель.

Внутри 1Password можно создавать безопасные хранилища (vaults), в которых хранятся пароли, кредитные карты, документы, лицензии и другие важные данные. Каждая запись содержит название, логин, пароль, URL-адрес, заметки и дополнительные поля. Все данные зашифрованы с помощью AES-256 и хранятся на серверах 1Password или локально, в зависимости от настроек пользователя.

Доступ к сохраненным данным осуществляется через приложение, браузерные расширения или API. 1Password автоматически заполняет пароли на сайтах и в приложениях, устраняя

необходимость запоминания сложных комбинаций. Также есть функция генерации надежных паролей, которая помогает создавать уникальные учетные данные для каждого сервиса.

Приложение поддерживает функцию Watchtower, которая уведомляет пользователей о скомпрометированных паролях, утечках данных и устаревших паролях. Также предусмотрена возможность безопасного обмена учетными данными внутри команды или семьи через общие хранилища.

Для бизнеса 1Password предлагает централизованное управление учетными записями сотрудников, контроль доступа и интеграцию с корпоративными системами, такими как Azure AD, Okta и OneLogin. Администраторы могут настраивать политики безопасности, двухфакторную аутентификацию и мониторить активность пользователей.

Таким образом, 1Password — это простое и удобное решение для защиты конфиденциальной информации, обеспечивающее безопасность личных и корпоративных данных, удобный доступ и простоту управления секретами.

1.2.4. Описание приложения LastPass Enterprise

LastPass Enterprise [10] — это корпоративное решение для хранения, управления и безопасного обмена паролями внутри организаций. Оно позволяет сотрудникам удобно работать с учетными данными, минимизируя риски, связанные с утечками или слабыми паролями. LastPass Enterprise доступен как веб-приложение, desktop приложение, мобильное приложение и расширение для браузеров, что делает его удобным для использования на любых устройствах.

Система аутентификации LastPass Enterprise поддерживает вход через единый мастер-пароль, многофакторную аутентификацию (MFA) и интеграцию с провайдерами единого входа (SSO), такими как Azure AD, Okta и Google Workspace. Администраторы могут централизованно управлять доступом сотрудников, настраивать политики безопасности и контролировать действия в системе через панель управления.

Каждый пользователь получает личное зашифрованное хранилище (vault) для хранения паролей, секретных заметок и других конфиденциальных данных. Пароли можно организовывать в папки, а также делиться ими с коллегами, задавая уровни доступа (только просмотр или редактирование). Все данные зашифрованы с использованием алгоритма AES-256, а расшифровка происходит только на стороне пользователя, обеспечивая нулевое разглашение (Zero-Knowledge Security).

Доступ к паролям осуществляется через веб-интерфейс, мобильные приложения и расширения для браузеров, которые автоматически подставляют учетные данные на сайтах и в приложениях. Также есть встроенный генератор надежных паролей, который помогает создавать уникальные комбинации для каждого ресурса.

LastPass Enterprise предоставляет функции мониторинга безопасности, включая аудит паролей, выявление слабых и повторяющихся паролей, а также уведомления о скомпрометированных учетных данных. Инструмент Dark Web Monitoring анализирует утечки данных и предупреждает пользователей, если их пароли обнаружены в базах взломанных аккаунтов.

Для крупных организаций LastPass Enterprise предлагает детализированные политики доступа, отчеты об использовании и интеграцию с SIEM-системами. Также доступна возможность автоматического предоставления и отзыва доступа при изменении ролей сотрудников.

Таким образом, LastPass Enterprise — это надежное корпоративное решение для управления паролями, которое повышает безопасность организаций, снижает нагрузку на IT-отдел и обеспечивает удобный доступ сотрудников к их учетным данным.

1.2.5. Описание разрабатываемого решения

Веб-приложение для хранения корпоративной конфиденциальной информации — это приложение, предназначенное для безопасного управления конфиденциальными данными компаний, такими как SSH-ключи, токены доступа, сертификаты и пароли. Приложение ориентировано на корпоративное использование, обеспечивая строгий контроль доступа к секретам и защиту данных через ролевую модель управления и шифрование.

Приложение поддерживает систему учетных записей с авторизацией через Keycloak, что позволяет интегрировать его в корпоративную инфраструктуру безопасности. Гибкая настройка прав доступа позволяет разграничивать доступ к секретам в зависимости от роли пользователя и рабочего пространства.

Секреты хранятся в изолированных рабочих пространствах, к которым можно добавлять пользователей с различными уровнями прав. Создание, редактирование и удаление секретов происходит через веб-интерфейс, а также с помощью API. Встроенные механизмы логирования фиксируют все действия пользователей, обеспечивая полную прозрачность изменений.

Доступ к секретам осуществляется через веб-приложение и API, а уведомления о действиях с секретами могут отправляться через Telegram-бот. Бот также позволяет просматривать выбранные пользователем уведомления о событиях, что делает систему удобной для оперативного отслеживания изменений.

Приложение поддерживает автоматическое удаление секретов по истечении их срока действия. Для безопасности все данные хранятся в зашифрованном виде.

Для удобства использования веб-интерфейс разработан на React [11] с адаптивной верстской, а серверная часть реализована на Node.js и TypeScript, интеграция с Keycloak упрощает управление авторизацией.

Таким образом, веб-приложение для хранения корпоративных секретов предоставляет безопасную и гибкую платформу для управления конфиденциальной информацией в корпоративной среде, обеспечивая контроль, аудит и удобство работы с секретами.

1.3. Анализ существующих решений

В Таблице 1 представлено сравнение существующих решений и разрабатываемого сервиса.

Таблица 1 – Сравнение существующих и разрабатываемого решений

Параметр	**HashiCorp Vault**	**AWS Secrets Manager**	**1Password**	**LastPass Enterprise**	**Предложенное решение**
Способы аутентификации	LDAP, OAuth, GitHub, Keycloak, MFA	IAM (AWS), MFA	Мастер-пароль, 2FA	Мастер-пароль, MFA	Keycloak, MFA
Модель управления доступом	+	+	+	+	+
Где хранятся данные	Локально, в облаке	В облаке AWS	В облаке 1Password	В облаке LastPass	Локально
Шифрование	AES-256, TLS	AES-256, TLS	AES-256, сквозное шифрование	AES-256, Zero-Knowledge Security	AES-256
Интеграция с внешними сервисами	Kubernetes, AWS, GCP, Azure	AWS Lambda, RDS, DynamoDB	Браузеры	Браузеры	Telegram-бот
Мобильное приложение	-	-	+	+	-
Автоматическая ротация секретов	Да, для баз данных и облачных сервисов	Да, для AWS-ресурсов	-	-	Да, настройка сроков действия секретов
Журналирование и аудит	Подробное ведение логов	CloudTrail	Watchtower (мониторинг утечек паролей)	Dark Web Monitoring, аудит	Логирование всех действий пользователей
Поддержка MFA	+	+	+	+	+
Доступ через API	Да, REST API [12]	Да, AWS SDK	-	-	Да, REST API

Поддержка мобильных приложений	-	-	+	+	-
Стоимость	Open-source, платные корпоративные версии	Платная, зависит от количества секретов	Подписка для частных лиц и бизнеса	Подписка для бизнеса	Бесплатное, Open-source
Поддержка SSO	Да, с провайдерами OAuth и LDAP	Да, через IAM и AWS SSO	Да, с корпоративными аккаунтами	Да, с корпоративными аккаунтами	Да, через Keycloak
Возможность развёртывания on-premise	+	-	-	-	+
Поддержка командной работы	Да, с granular RBAC	Да, через IAM	Да, через общие хранилища	Да, через группы пользователей	Да, через рабочие пространства
Наличие техподдержки	Да, в платной версии	+	+	+	-

Выводы по главе

В данной главе рассматривается предметная область и описываются проблемы, которые решает приложение для хранения корпоративной конфиденциальной информации. Дается обзор проблемы с точки зрения бизнеса и его сотрудников.

Также производится обзор существующих решений и их функционала, а затем – обзор разрабатываемого решения в виде функциональных требований и сравнительный анализ между ними.

2. Глава 2. Проектирование хранилища

2.1. Пользовательские сценарии

В данном разделе содержатся пользовательские сценарии, которые описывают основные взаимодействия пользователей с системой, позволяя понять, как реализованы функции управления учетными записями, секретами и правами доступа. Основная цель веб-приложения – обеспечить надежное хранение и безопасный обмен конфиденциальными данными внутри корпоративной среды. Для удобства восприятия в раздел добавлена визуализация сценариев в виде диаграмм прецедентов, иллюстрирующих процессы аутентификации, работы с секретами и настройки доступов к ним.

Основой системы является учетная запись пользователя, которая обеспечивает доступ к функционалу приложения. Пользователи регистрируются в системе через внешнюю систему аутентификации Keycloak. Авторизация возможна как по логину и паролю, так и через SSO (Single Sign-On). После входа в систему пользователи могут управлять своими профилями, изменять настройки безопасности и настраивать многофакторную аутентификацию (MFA) для повышения уровня защиты данных.

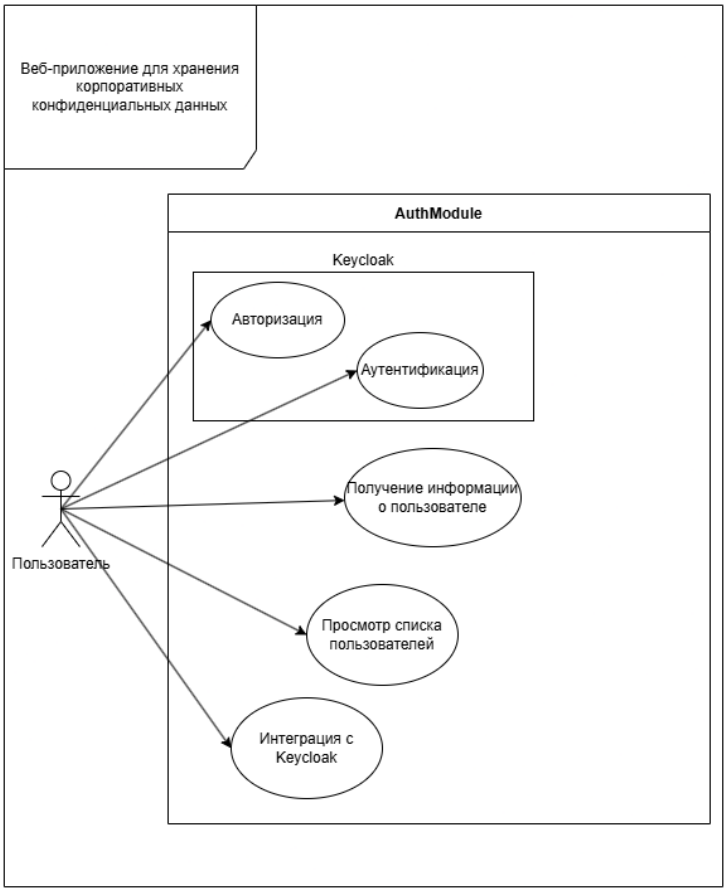

Рисунок 1 – Диаграмма прецедентов AuthModule

Приложение организует взаимодействие между пользователями через рабочие пространства. Его администратор может добавлять участников в пространство, назначая им определенные

роли и права. Например, одни пользователи могут только просматривать секреты, в то время как другие получают возможность их редактировать или удалять. При необходимости участники могут быть удалены из рабочего пространства, что автоматически закрывает им доступ ко всем связанным секретам.

Веб-интерфейс системы позволяет пользователям просматривать список участников рабочих пространств и управлять их правами. Такая гибкая модель доступа гарантирует, что конфиденциальные данные остаются в пределах установленных границ безопасности.

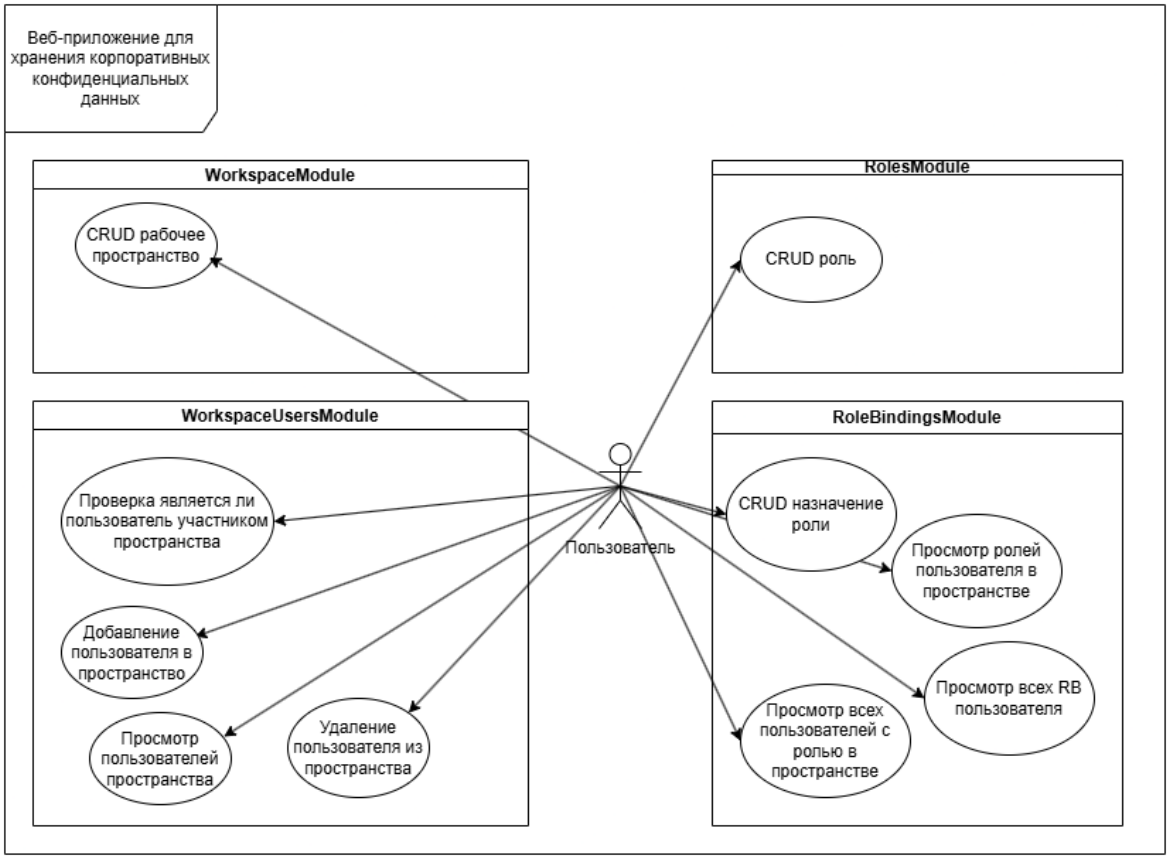

Рисунок 2 – Диаграмма прецедентов WorkspaceModule, WorkspaceUserModule, RolesModule, RoleBindingsModule

Ключевая функциональность системы сосредоточена вокруг управления секретами. Пользователи могут создавать записи с конфиденциальными данными, такими как SSH-ключи, API-токены, сертификаты и пароли. Каждому секрету можно задать уникальные параметры, включая срок действия и уровень доступа.

Редактирование и удаление секретов доступны только тем пользователям, у которых есть соответствующие права. Для удобства администраторы могут настраивать видимость секретов, определяя, кто может их просматривать и изменять. Это позволяет разделить доступ между различными командами внутри компании, предотвращая несанкционированный доступ к критически важным данным.

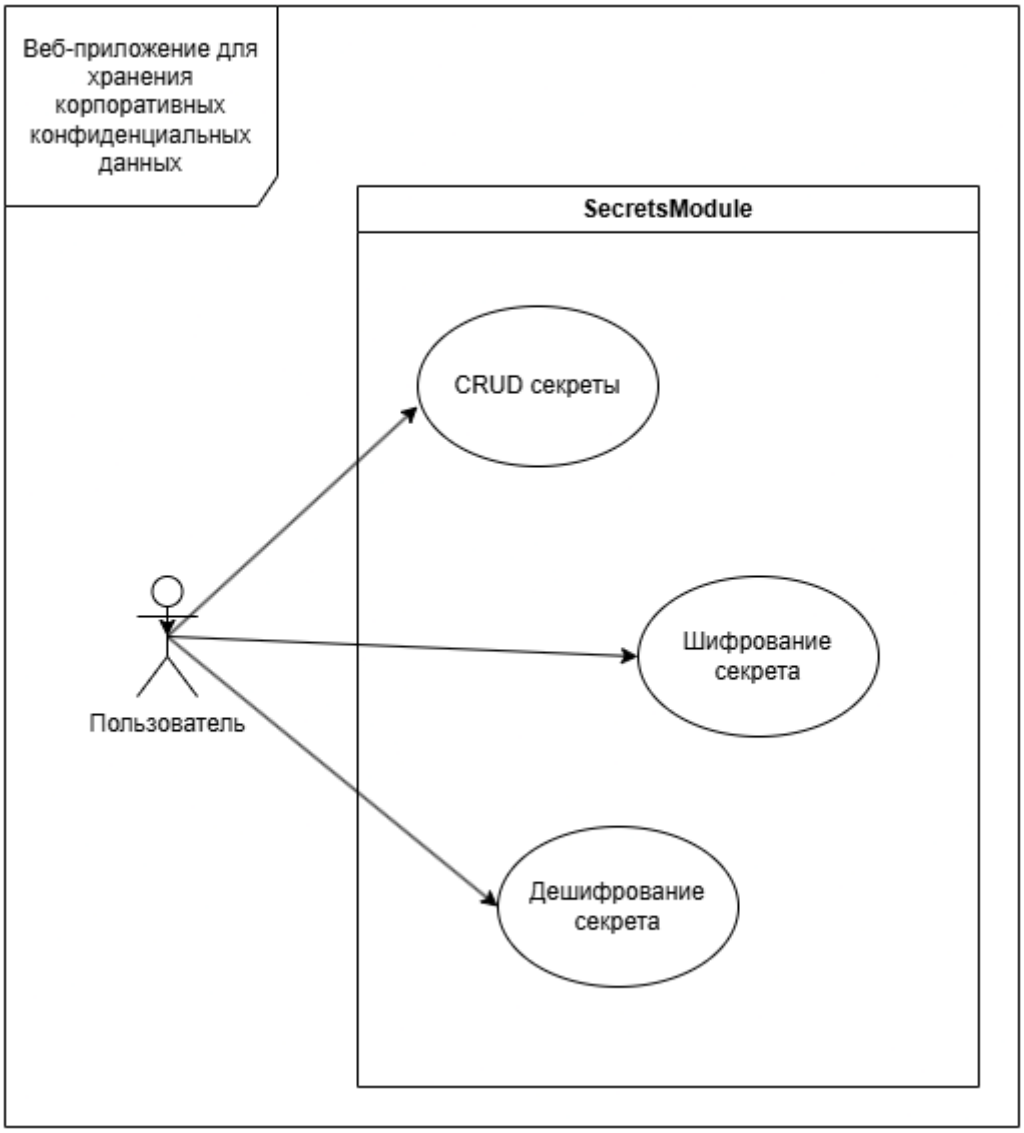

Рисунок 3 – Диаграмма прецедентов SecretsModule

Система также поддерживает интеграцию с внешними ресурсами. Пользователи могут экспортировать логи активности в текстовом формате для анализа, а уведомления о ключевых изменениях в секретах отправляются через Telegram-бот, что делает процесс мониторинга более удобным.

В отличие от платформ для хранения пользовательского контента, приложение не предусматривает публичного доступа к секретам. Однако внутри системы администраторы могут управлять шаблонами политик безопасности и настраивать рекомендации по конфигурациям доступа. Эти рекомендации доступны всем пользователям внутри организации и помогают выстраивать безопасную модель хранения секретов.

Приложение предоставляет автоматизированные инструменты для управления сроками действия секретов. Администраторы могут задавать политики автоматического обновления или удаления устаревших данных, что снижает риски хранения неактуальных учетных данных.

Система также ведет историю всех изменений, что позволяет пользователям отслеживать, кто и когда вносил правки в конфиденциальную информацию. Для неавторизованных пользователей работа с секретами недоступна, так как приложение требует входа для доступа к любым данным.

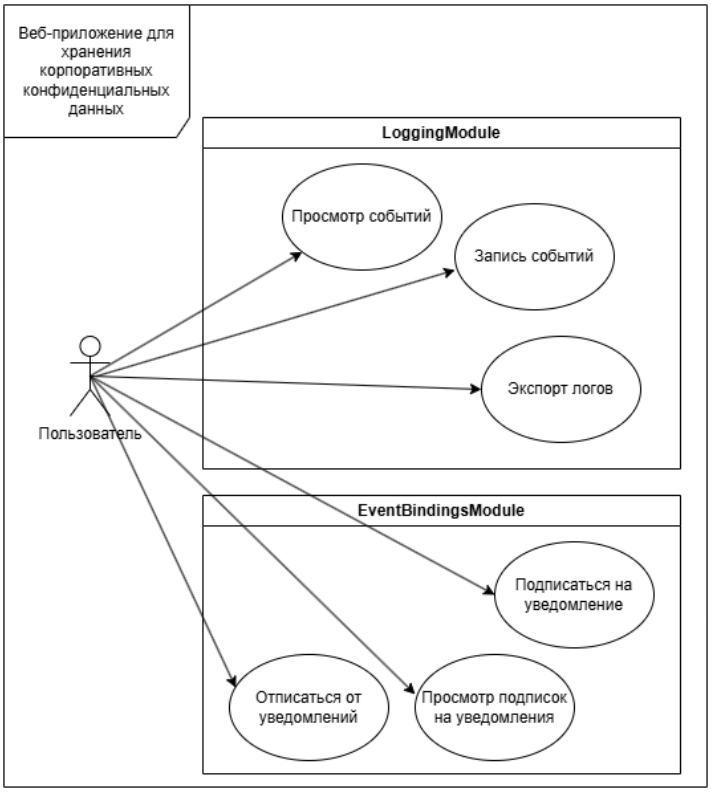

Рисунок 4 – Диаграмма прецедентов LoggingModule и EventBindingsModule

Для информирования пользователей о важных событиях реализована система уведомлений. Она позволяет получать сообщения о запросах на доступ, изменениях в секретах и системных предупреждениях, связанных с безопасностью.

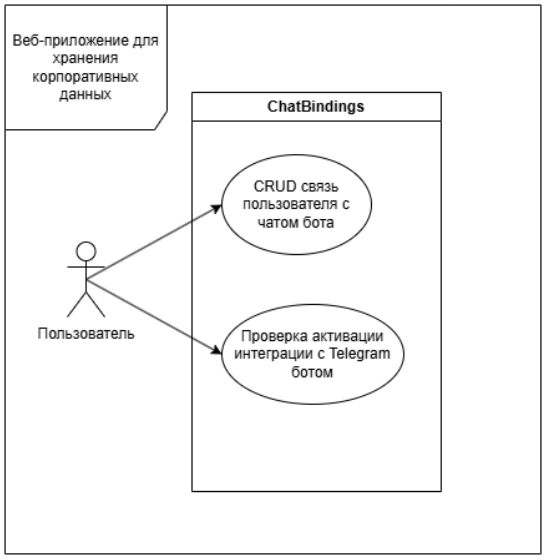

Рисунок 5 – Диаграмма прецедентов ChatBindingsModule

Уведомления отправляются в режиме реального времени через Telegram-бота. Пользователь может настроить типы событий, о которых он хочет получать оповещения, что делает систему более удобной и гибкой.

2.2. Архитектура приложения

Архитектура веб-приложения для хранения корпоративных конфиденциальных данных основана на современных принципах разработки, обеспечивающих безопасность, масштабируемость и гибкость системы. Для описания архитектурных решений используются визуальные инструменты, такие как UML [13] и C4-модели [14], которые помогают представить структуру компонентов, их взаимодействие и потоки данных. Основная цель этого раздела – объяснить устройство системы, ее логику и ключевые механизмы работы.

Приложение построено по трехслойной клиент-серверной архитектуре, включающей клиентскую часть, серверный слой с бизнес-логикой и базу данных. Клиентская часть представляет собой веб-приложение, разработанное на React с использованием TypeScript. Оно обеспечивает удобный интерфейс для управления секретами, рабочими пространствами и правами доступа. Серверная часть реализована на Node.js с использованием TypeScript и представляет собой REST API, отвечающее за обработку запросов клиентов, аутентификацию пользователей и управление данными. Вся информация хранится в реляционной базе данных PostgreSQL [15], где секреты, пользователи и рабочие пространства организованы в виде отдельных таблиц с четко определенными связями.

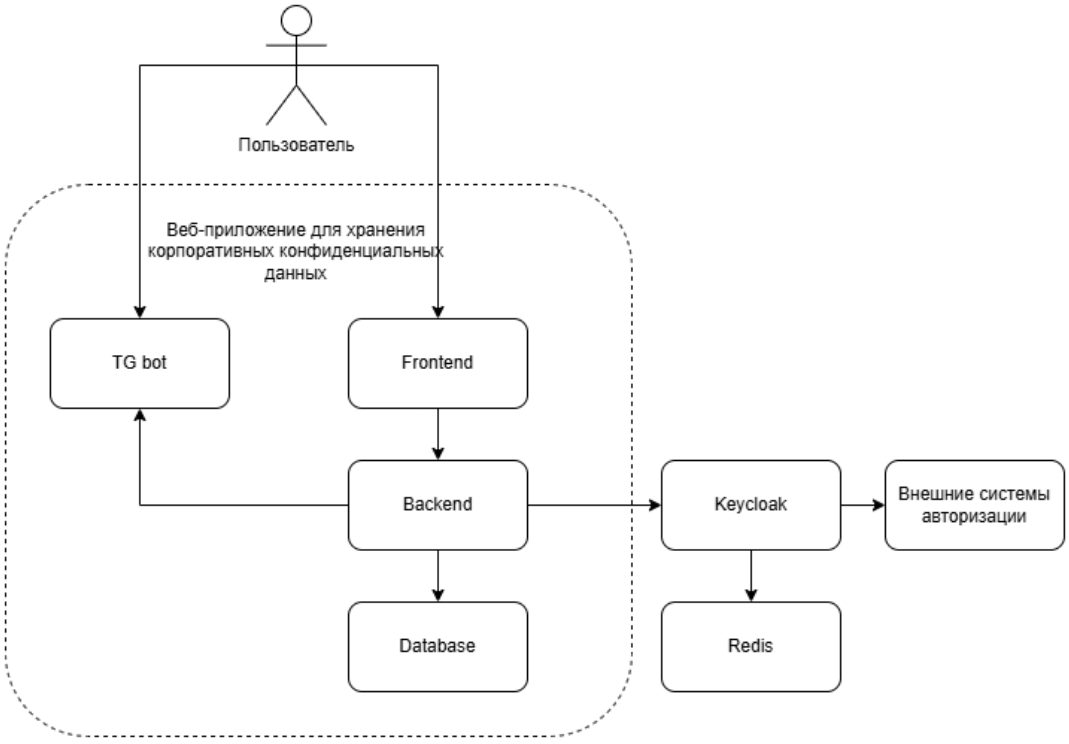

Рисунок 6 – Диаграмма C4 архитектуры приложения

Выбранная архитектура обладает рядом преимуществ. Гибкость системы позволяет легко адаптировать клиентскую часть под различные устройства и интегрировать ее с другими сервисами. Масштабируемость достигается за счет модульного подхода, который позволяет добавлять новые компоненты или изменять существующие без значительного влияния на систему. Безопасность обеспечивается за счет строгого разграничения прав доступа, шифрования данных и механизма аутентификации на основе Keycloak.

Структура базы данных представлена в виде диаграммы сущностей и связей, которая наглядно демонстрирует связи между пользователями, рабочими пространствами и секретами. Ключевые сущности включают пользователей, секреты, рабочие пространства и журналы аудита. Таблица пользователей хранит информацию об учетных записях, включая идентификаторы, имена и методы аутентификации. Рабочие пространства позволяют группировать секреты и предоставлять доступ к ним разным пользователям. Каждое рабочее пространство связано с несколькими пользователями и секретами, что позволяет гибко управлять правами доступа.

Взаимодействие между сущностями строится по принципу строгого контроля доступа. Пользователь может принадлежать к нескольким рабочим пространствам, но его уровень доступа зависит от назначенной роли. Секреты имеют поля с зашифрованными значениями, а их метаданные указывают срок действия и возможность автоматической ротации. Журналы аудита фиксируют все изменения, связанные с секретами и пользователями, что позволяет анализировать действия и обеспечивать прозрачность работы системы.

Таким образом, архитектура приложения обеспечивает баланс между удобством использования, безопасностью и масштабируемостью. Взаимосвязь всех компонентов системы позволяет гибко управлять доступами, обеспечивать централизованное хранение корпоративных секретов и гарантировать их защиту от несанкционированного доступа.

2.2.1. Схема базы данных

Архитектура базы данных веб-приложения для хранения корпоративных конфиденциальных данных разработана с учетом безопасности, гибкости управления доступами и удобства масштабирования. В ее основе лежит семь взаимосвязанных таблиц, каждая из которых выполняет четко определенную роль в управлении пользователями, рабочими пространствами, секретами и правами доступа.

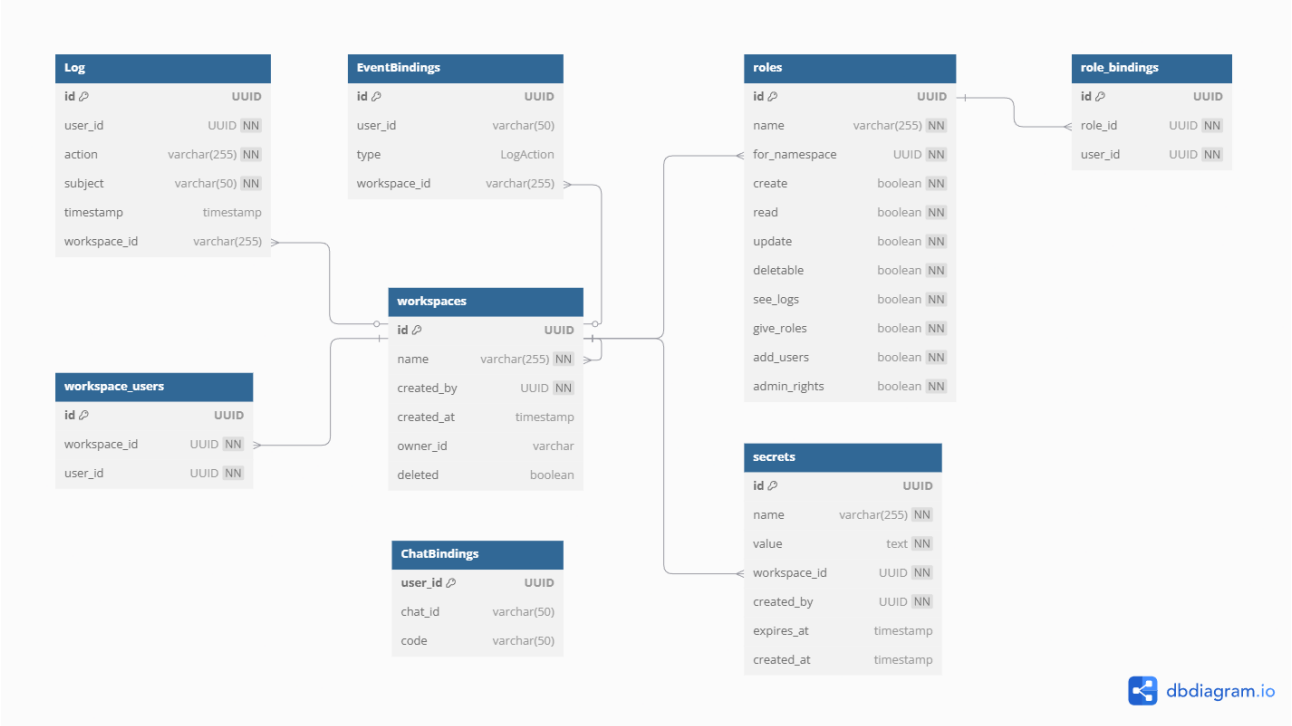

Рисунок 7 – Диаграмма отношения сущностей в базе данных

Рабочие пространства (workspaces) формируют изолированные области для хранения секретов, обеспечивая организационную структуру системы. Каждое рабочее пространство имеет уникальный id, название (name) и связано с пользователем, его создавшим (created_by), что позволяет контролировать владельца пространства.

Модель управления доступом строится на основе ролевой модели. Таблица roles описывает роли внутри каждого рабочего пространства (for_namespace), определяя, какие права (rights) принадлежат той или иной роли. Например, роль может позволять пользователю создавать, изменять или только просматривать секреты.

Назначение ролей пользователям реализовано через таблицу role_bindings, где связываются role и user. Это позволяет динамически управлять правами, добавлять и удалять доступы без изменения самой структуры данных. Дополнительно связь пользователей с рабочими пространствами отражена в таблице workspace_users, где фиксируются workspace_id, user_id и конкретная роль (role). Это создает гибкую систему управления, позволяющую одним пользователям администрировать пространство, а другим только просматривать секреты.

Ключевой сущностью системы являются секреты, хранящиеся в таблице secrets. Каждый секрет имеет name и value, где значение представляет собой зашифрованные данные, такие как пароли, ключи доступа или токены. Поле workspace_id связывает секрет с конкретным рабочим

пространством, a created_by указывает на автора. Дополнительно предусмотрено поле expires_at, позволяющее автоматически удалять или обновлять устаревшие данные.

Для обеспечения прозрачности и безопасности работы система ведет аудит всех действий в таблице Log. В ней фиксируются операции (action), выполненные пользователями (user_id), объекты, к которым они применялись (subject), а также временная метка (timestamp). Это позволяет анализировать изменения и отслеживать потенциально нежелательные операции.

Для подписки на уведомления существует таблица EventBindings, в которой содержатся идентификатор пользователя (user_id), тип записываемого действия (create, read, update, delete, access, export) и идентификатор рабочего пространства (workspace_id). Это позволяет пользователю подписаться на выбранные типы уведомлений в определенном пространстве.

Для связи системы логирования с Telegram ботом для отправки уведомлений есть таблица ChatBindings, которая для каждого пользователя (user_id) хранит связанный с ним Telegram чат (chat_id) и уникальный код для аутентификации в Telegram боте при первичной привязке учетной записи.

Взаимосвязь между всеми таблицами обеспечивает надежную модель хранения корпоративных секретов, поддерживает строгую политику доступа и интеграцию с внешними сервисами. Такое устройство базы данных делает систему масштабируемой, безопасной и удобной для корпоративного использования.

2.2.2. Схема backend.

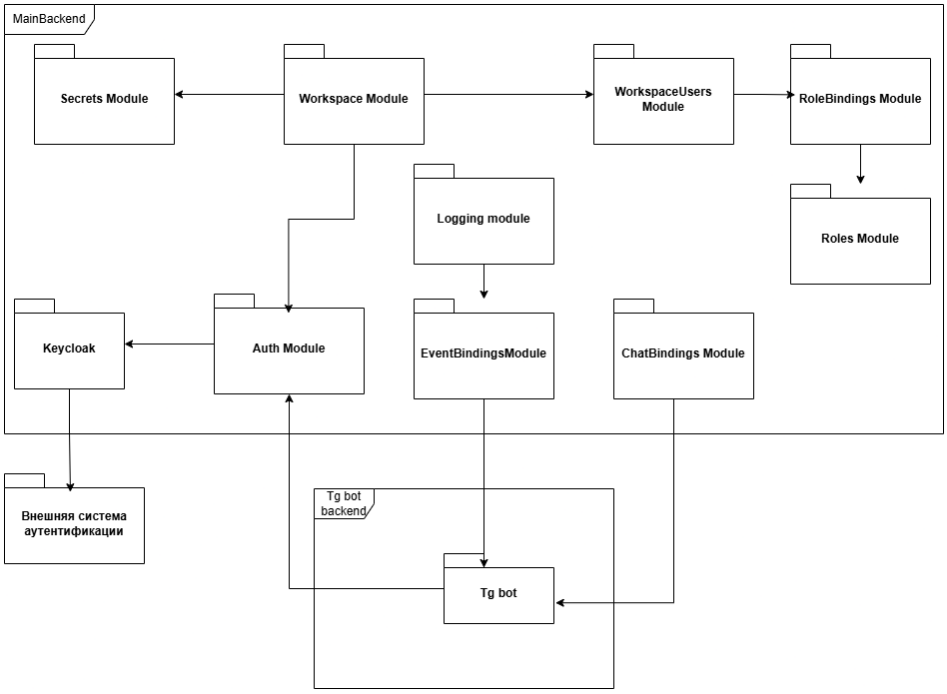

Рисунок 8 – Диаграмма связей между модулями backend

Архитектура backend-слоя приложения, представленная на диаграмме, отражает модульную структуру, обеспечивающую гибкость, масштабируемость и интеграцию с внешними сервисами. Каждый компонент выполняет определенную роль в функционировании системы, что обеспечивает надежность и удобство использования.

Основной частью системы является MainBackend, в который включены несколько ключевых модулей. Auth Module играет центральную роль, отвечая за аутентификацию и авторизацию пользователей. Он интегрируется с Keycloak — популярным инструментом для управления учетными записями и правами доступа. Это решение обеспечивает надежную защиту данных пользователей и упрощает управление учетными записями, предоставляя возможность масштабируемого подключения внешних систем аутентификации.

Модуль Workspace Module отвечает за управление рабочими пространствами, которые создают изолированные области для совместной работы команд. Этот модуль тесно связан с Secrets Module, где осуществляется управление корпоративными секретами. Secrets Module обеспечивает создание, хранение и шифрование данных, таких как ключи доступа, токены и пароли. Связь между этими модулями позволяет организовать безопасное распределение секретов внутри определенных рабочих пространств, исключая доступ для неавторизованных пользователей.

Event Module играет важную роль в отслеживании действий внутри системы. Этот модуль обрабатывает события, генерируемые пользователями и системой, такие как создание новых секретов, изменение прав доступа или регистрация новых пользователей. Это позволяет не только вести журнал аудита для повышения прозрачности и безопасности, но и отправлять уведомления, интегрированные в пользовательский интерфейс или внешние каналы, такие как Telegram-бот.

Telegram-бот, расположенный на отдельном сервере Tg bot backend, расширяет возможности приложения, предоставляя пользователям удобный способ для взаимодействия с системой событий. Это включает в себя получение уведомлений о событиях, выбранных пользователем. Бот использует Auth Module для первичной аутентификации пользователя, что гарантирует защиту пользовательских данных.

Все модули backend тесно связаны между собой, что обеспечивает целостность системы. Например, при создании нового рабочего пространства Workspace Module инициирует действия через Auth Module для проверки прав пользователя, а затем регистрирует событие в Event Module. Эта координация повышает эффективность системы и упрощает разработку новых функций.

Такая архитектура поддерживает требования масштабируемости, гибкости и безопасности, указанные в техническом задании. Разделение функциональности на независимые модули

позволяет легко адаптировать систему под новые задачи, добавлять интеграции с внешними сервисами и обеспечивать устойчивую работу приложения даже при увеличении числа пользователей или объема данных.

2.2.3. Описание frontend.

Архитектура фронтенда данного приложения, разработанного на React, построена с использованием модульного подхода, что обеспечивает высокую гибкость, масштабируемость и простоту сопровождения. В основе приложения лежит основной компонент App, который задает общую структуру и отвечает за маршрутизацию между различными разделами. Для реализации маршрутов используется библиотека React Router, позволяющая плавно переключаться между страницами, такими как авторизация, управление рабочими пространствами, работа с секретами и просмотр уведомлений.

Фронтенд состоит из нескольких модулей, каждый из которых выполняет строго определенные задачи. Модуль аутентификации отвечает за идентификацию пользователей через систему Keycloak. Он работает через библиотеку @react-keycloak/web, которая управляет состоянием текущего пользователя и токенами доступа. Это позволяет эффективно поддерживать сессии пользователей и обеспечивать безопасность операций.

Модуль управления рабочими пространствами предоставляет инструменты для создания, редактирования и просмотра рабочих пространств. Основные компоненты этого модуля включают список рабочих пространств, подробную информацию о конкретном пространстве и форму для создания новых. Эти компоненты активно взаимодействуют с backend, отправляя запросы через API и обновляя данные в реальном времени.

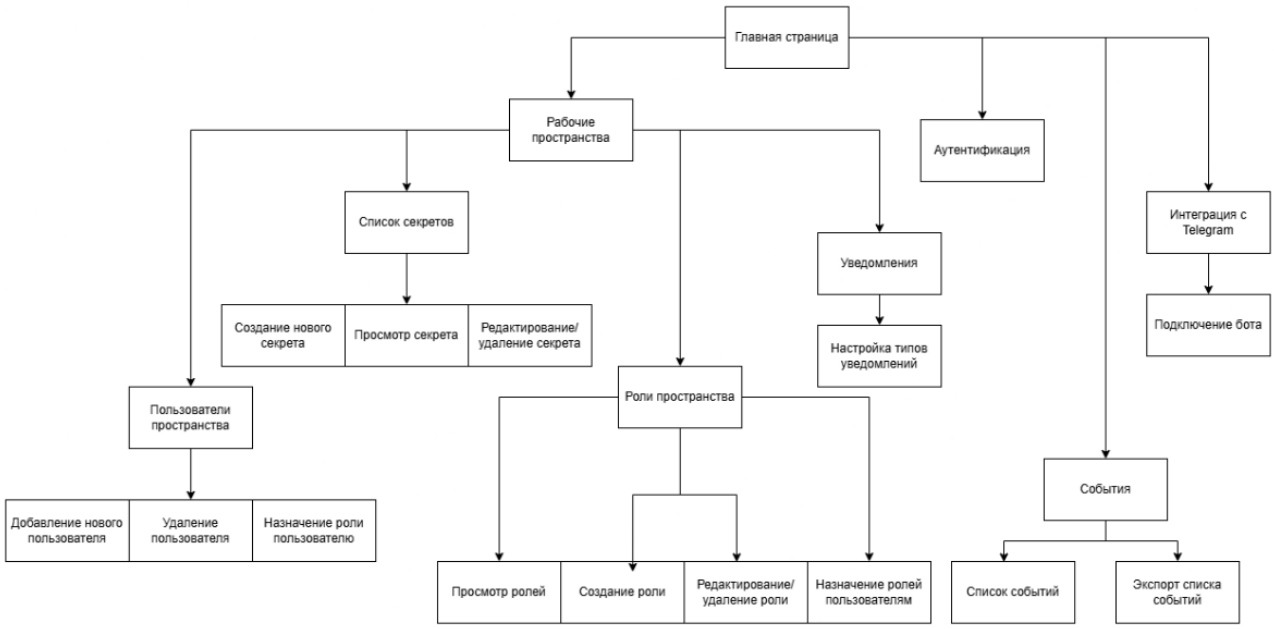

Рисунок 9 – Диаграмма иерархии страниц frontend

Модуль управления секретами играет ключевую роль в системе, обеспечивая безопасное хранение и управление конфиденциальной информацией, такой как токены и ключи. Он состоит из компонентов для отображения списка секретов, детального просмотра и создания новых записей. Все данные шифруются, что гарантирует их защиту как на стороне клиента, так и на сервере.

Отдельно выделен модуль уведомлений, который предоставляет пользователям актуальную информацию о событиях в системе, таких как изменения в рабочих пространствах или истечение срока действия секретов. Уведомления отображаются через специальный центр уведомлений, который использует периодические API-запросы для синхронизации данных в реальном времени.

Модуль интеграции с Telegram-ботом позволяет пользователям настраивать взаимодействие с ботом, включая получение уведомлений и выполнение базовых операций через Telegram. Для этого реализованы компоненты настройки интеграции и отображения статуса подключения.

Безопасность данных поддерживается через защищенные API-запросы, использующие токены доступа. Критически важные операции, такие как создание и удаление секретов, дополнительно защищены механизмами авторизации.

Таким образом, архитектура React-приложения сочетает в себе модульность, безопасность, удобство разработки и высокую производительность, создавая надежный и функциональный интерфейс для пользователей.

2.3. Выбор методов и средств реализации

2.3.1. База данных

Реляционная база данных является оптимальным выбором для данного приложения по нескольким ключевым причинам. Во-первых, приложение имеет строгую структуру данных с четко определенными связями между сущностями, такими как пользователи, рабочие пространства, роли, секреты и журналы действий. Реляционные базы данных, такие как PostgreSQL, идеально подходят для таких задач благодаря своей способности моделировать сложные отношения через механизмы внешних ключей, индексов и ссылочной целостности.

Структура приложения включает таблицы, в которых хранятся взаимосвязанные данные. Например, таблица пользователей связана с таблицей рабочих пространств, где каждый пользователь может быть создателем или участником одного или нескольких рабочих пространств. Кроме того, роли пользователей зависят от рабочей области, а права доступа детализированы через таблицу ролей. Такая взаимосвязанность требует системы, которая обеспечивает надежное управление отношениями между данными, что и является основной характеристикой реляционных баз данных.

Выбор PostgreSQL как системы управления реляционными базами данных обоснован наличием у него важных преимуществ. PostgreSQL предоставляет большой функционал, включая поддержку сложных запросов, транзакций и расширений, что делает её идеальным выбором для приложения с требованиями к безопасности и производительности. Для приложения, предполагающего работу с большим количеством пользователей, рабочих пространств и секретов, это критично.

Отдельное внимание заслуживает аспект безопасности. PostgreSQL предоставляет гибкие механизмы управления доступом, включая ролей и политик безопасности на уровне строк, что идеально сочетается с потребностью приложения в детализированном разграничении прав пользователей. Учитывая, что приложение работает с конфиденциальными данными, такими как секреты и их журналы изменений, использование PostgreSQL помогает обеспечить надежную защиту этих данных.

В итоге, реляционная модель данных в сочетании с мощным функционалом PostgreSQL обеспечивает надежность, масштабируемость и безопасность, необходимые для реализации данного приложения. Это делает PostgreSQL идеальным выбором для хранения данных, связанных с пользователями, рабочими пространствами, ролями и секретами, а также для обеспечения высокоэффективного управления этими данными.

2.3.2. Backend

Выбор стека технологий для backend-приложения, включающего TypeScript, Node.js, Express [16], TypeORM [17], а также node-telegram-bot-api [18] для Telegram-бота, обусловлен сочетанием гибкости, производительности, надежности и удобства разработки.

TypeScript был выбран в качестве основного языка программирования в первую очередь из-за наличия в нем статической типизации, в отличие от JavaScript. При разработке крупного приложения, которое включает в себя сложные сущности, такие как пользователи, роли, рабочие пространства и секреты, статическая типизация помогает минимизировать количество ошибок на этапе разработки. TypeScript обеспечивает более высокий уровень уверенности в коде за счет строгой проверки типов, улучшает читаемость и поддерживаемость проекта, а также предоставляет средства для работы с объектами и интерфейсами, что критически важно для интеграции с реляционной базой данных.

При этом TypeScript сохраняет простоту написания нового кода и поддержки существующего, присущую JavaScript, что делает его идеальным выбором для существенно ограниченного времени и ресурсов на этот проект.

Node.js является отличным выбором для backend-приложения благодаря своей высокой производительности и возможности обработки большого количества асинхронных операций. В

данном проекте необходимо обрабатывать запросы к базе данных, обеспечивать авторизацию через Keycloak и управлять различными сущностями, такими как секреты и рабочие пространства. Асинхронная модель ввода-вывода Node.js позволяет эффективно обрабатывать множество запросов одновременно, что делает её оптимальной для приложений, где высокая нагрузка на сервер — это норма.

Express был выбран как backend-фреймворк из-за его минималистичного, но структурированого подхода. Он предоставляет удобные инструменты для создания REST API, необходимых для взаимодействия с клиентом и Telegram-ботом. Express обеспечивает гибкость в настройке middleware, обработке маршрутов и интеграции сторонних библиотек.

TypeORM используется для работы с реляционной базой данных PostgreSQL, обеспечивая высокоуровневую абстракцию над SQL-запросами. Это позволяет мне сосредоточиться на бизнес-логике приложения, а не на ручной реализации SQL. TypeORM поддерживает такие функции, как миграции базы данных, создание отношений между таблицами и удобную работу с типами данных. В контексте проекта, где данные пользователей, роли, секреты и журналов событий требуют сложных взаимосвязей, TypeORM помогает эффективно управлять этой сетью. Более того, его интеграция с TypeScript позволяет максимально использовать преимущества типизации и автодополнения, повышая скорость разработки.

Для реализации Telegram-бота был выбран node-telegram-api. Эта библиотека обеспечивает удобные инструменты для взаимодействия с Telegram Bot API. Она поддерживает обработку сообщений, клавиатур, команд и других функций, которые необходимы для реализации удобного пользовательского интерфейса в Telegram. В рамках приложения Telegram-бот выступает как интерфейс для получения уведомлений и других действий. Использование node-telegram-api позволяет быстро интегрировать бота в экосистему приложения, минимизируя время на разработку.

Такой стек технологий был выбран из-за их отличной совместимости и соответствия требованиям проекта. В совокупности они обеспечивают надежную основу для создания производительного, масштабируемого и легко поддерживаемого backend-приложения.

2.3.3. Frontend

Для разработки frontend данного приложения был выбран стек технологий, включающий TypeScript, React и библиотеки для интеграции с Keycloak, такие как @react-keycloak/web и keycloak-js. Этот выбор обусловлен преимуществами, которые идеально подходят для реализации интерфейса сложного, интерактивного и надежного веб-приложения.

TypeScript был выбран по тем же причинам, что и для backend. Его статическая типизация помогает минимизировать количество ошибок на этапе разработки, обеспечивает высокую

читаемость кода и упрощает работу с данными, которые приходят от backend. В рамках frontend приложения это особенно важно для работы с REST API, сложными типами данных (такими как секреты, роли, пользователи и рабочие пространства), а также для поддержки типизации компонентов React.

React стал основным инструментом для разработки пользовательского интерфейса благодаря своей гибкости, популярности и богатой экосистеме. React позволяет создавать компоненты, которые легко переиспользовать и масштабировать, что критически важно для приложений с модульной структурой. React также предоставляет комплексные возможности для работы с состоянием и рендерингом, обеспечивая высокую производительность даже при высокой нагрузке.

Для реализации стилистической составляющей пользовательского интерфейса была выбрана библиотека material-ui, которая предоставляет огромный готовый набор элементов пользовательского интерфейса, которые адаптируются под различную ширину экрана и обладают значительной кастомизацией, позволяя внести небольшую уникальность в дизайн приложения, но сохраняя простоту и скорость разработки.

3. Глава 3. Программная реализация веб-приложения

В данной главе описывается программная реализация пользовательского интерфейса веб-приложения и публичного API.

3.1. Система авторизации

Система авторизации в приложении реализована с использованием Keycloak, который выступает в роли системы управления идентификацией и доступом (IAM). Keycloak обеспечивает аутентификацию пользователей через современные протоколы OAuth 2.0 [19] и OpenID Connect [20], что позволяет гибко интегрировать приложение с различными внешними провайдерами аутентификации, такими как Google, Facebook или корпоративные системы единого входа (SSO). После успешной аутентификации на сервере Keycloak пользователь получает токен доступа (JWT), который содержит информацию о его профиле и правах. Этот токен затем используется приложением для авторизации запросов. Для серверной части, построенной на Express.js, применяется специальный middleware, например, keycloak-connect, который интегрирует Keycloak с приложением. Middleware выполняет проверку валидности токенов, извлекает данные о пользователе (например, идентификатор, роли) и добавляет их в объект запроса (req), что позволяет другим частям приложения принимать решения об авторизации. Например, если пользователь запрашивает доступ к защищённому ресурсу, middleware проверяет наличие у него соответствующей роли, такой как admin или editor, и либо разрешает доступ, либо возвращает ошибку 403 Forbidden.

Для повышения безопасности и производительности в системе реализована обработка refresh-токенов. Когда срок действия основного токена доступа истекает, приложение автоматически использует refresh-токен для получения нового токена без необходимости повторного ввода учетных данных пользователем. Этот процесс управляется на стороне Keycloak и поддерживается клиентской логикой, что обеспечивает бесперебойную работу сессии. Кроме того, Keycloak позволяет централизованно управлять политиками доступа: администраторы могут задавать роли, рамки их действия и правила авторизации через административную консоль, что упрощает настройку и поддержку системы.

Для хранения сессий пользователей используется Redis [21] — высокопроизводительное in-memory хранилище. Redis сохраняет данные о сессиях, включая идентификаторы пользователей и связанные с ними токены, что позволяет приложению быстро восстанавливать состояние между запросами. Это особенно важно для масштабируемости: при развертывании приложения на нескольких серверах Redis обеспечивает синхронизацию сессий через кластер, гарантируя, что пользователь остаётся аутентифицированным даже при переключении между нодами. Для управления временем жизни сессий применяется механизм TTL (time-to-live), который автоматически удаляет устаревшие записи, минимизируя утечки памяти и поддерживая безопасность. Интеграция Redis с Express.js осуществляется через библиотеку, такую как express-session с адаптером connect-redis, что делает настройку прозрачной и эффективной.

С точки зрения пользовательского опыта, система авторизации проста и удобна. Пользователь видит страницу входа с полями для имени пользователя и пароля, а после успешной аутентификации сразу получает доступ к приложению.

Рисунок 10 – Страница авторизации через Keycloak

3.2. Система управления рабочими пространствами

Пользовательский интерфейс страницы управления рабочими пространствами простой и интуитивно понятный. Рабочие пространства представлены в виде сетки карточек, каждая из которых содержит название, дату создания и две кнопки: иконку карандаша для редактирования и корзины для удаления. Вверху страницы также представлена кнопка для создания нового пространства. Нажатие на карточку переводит пользователя на страницу с секретами этого пространства.

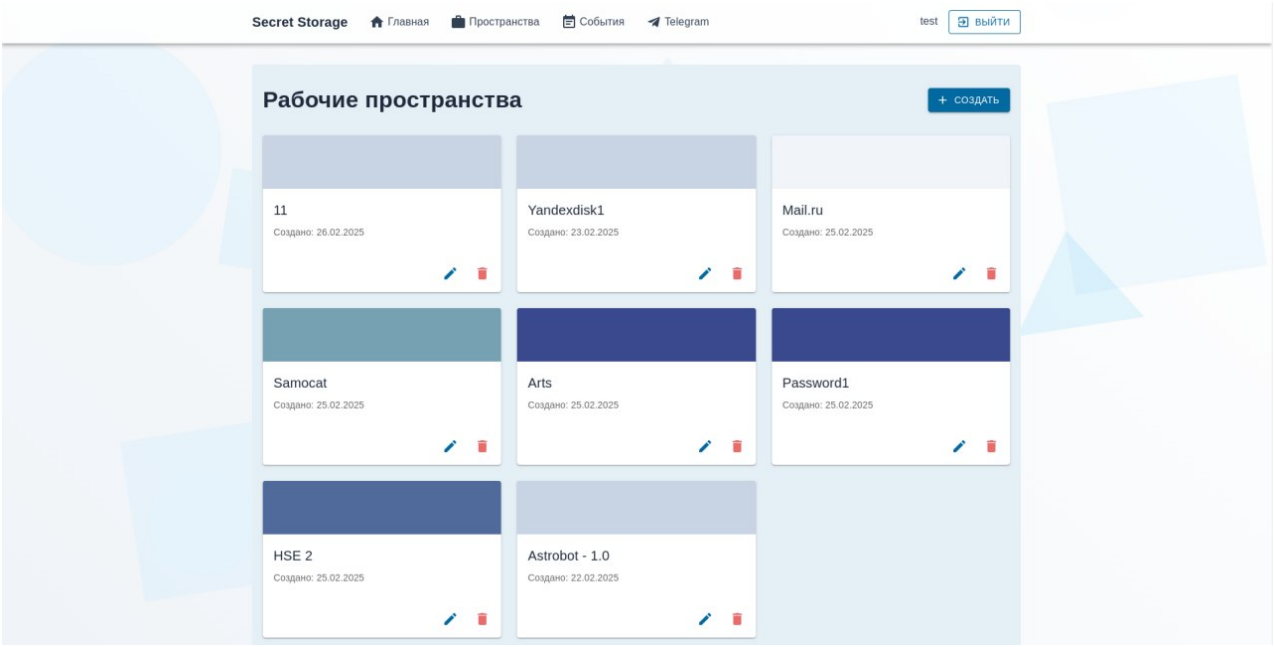

Рисунок 11 – Страница управления рабочими пространствами

На техническом уровне страница реализована в React-компоненте WorkspacesPage. Система ролевого доступа гарантирует, что только пользователи, имеющие соответствующие разрешения, могут редактировать или удалять пространства.

WorkspacesPage взаимодействует с бэкендом через следующие API endpoints:

- **GET /workspaces**: используется для получения списка рабочих пространств пользователя и отображения их на странице.
- **POST /workspaces**: отвечает за создание нового пространства при нажатии кнопки "Создать".
- **PUT /workspaces/:id**: позволяет редактировать данные пространства, например, изменять название через кнопку "Редактировать".
- **DELETE /workspaces/:id**: управляет удалением пространства и всех связанных данных при выборе опции "Удалить".

3.3. Система управления секретами

Страница управления секретами предоставляет лаконичный интерфейс для управления секретами в рабочем пространстве. Навигация между различными составляющими рабочего пространства осуществляется через кнопки "Секреты", "Пользователи", "Роли" и "Уведомления" в верхней части страницы. Создание нового секрета доступно через кнопку "Создать секрет", которая открывает диалоговое окно для ввода данных секрета. Список секретов отображает их названия и указывает, что срок действия не установлен, а для каждого элемента предусмотрены кнопки редактирования и удаления в виде иконок. При клике на секрет открывается окно с подробной информацией, включая значение секрета, срок действия и создателя, а также кнопку для копирования значения в буфер обмена. Поиск секретов реализован через строку поиска "Поиск секрета". Интерфейс ориентирован на удобство и ясность, обеспечивая выполнение задач без лишних сложностей.

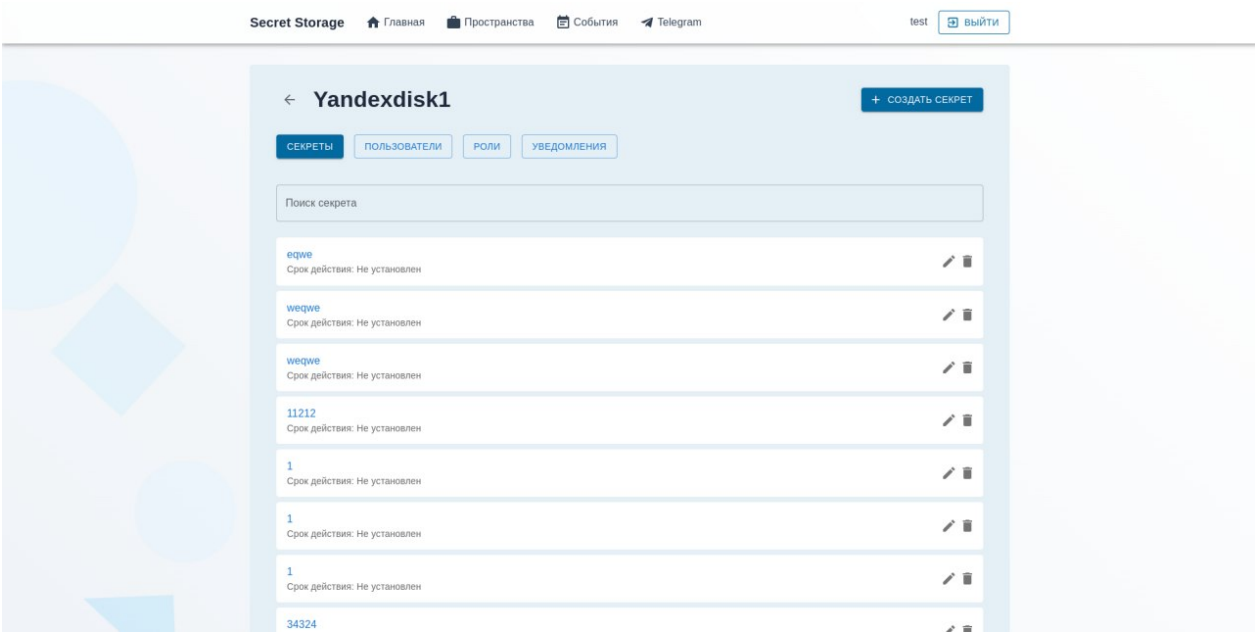

Рисунок 12 – Страница управления секретами пространства

Страница реализована в React-компоненте SecretsPage. Она взаимодействует с бэкендом через RESTful API для выполнения операций с секретами.

- **GET /secrets/workspace/:workspaceId:** запрашивает список секретов.
- **POST /secrets:** создает секрет.
- **PUT /secrets/:id**: редактирует секрет.
- **DELETE /secrets/:id**: удаляет секрет.

Все секреты шифруются с использованием алгоритма AES-256 с помощью библиотеки crypto в Node.js, а ключ шифрования хранится в переменных окружения для обеспечения безопасности. Доступ к секретам регулируется системой ролей, если у пользователя нет прав

для чтения секретов, то их список будет пустым. Поддерживается установка срока действия секретов.

3.4. Система управления ролями и правами доступа

Пользовательский попадает на страницу управления ролями через вкладку "Роли" в навигационной панели. Роли представлены в пользовательском интерфейсе в виде карточек. Каждая карточка содержит название роли и список прав, таких как "Создавать секреты" или "Управление пользователями", что позволяет быстро оценить область действия роли. Кнопка "Создать роль" открывает диалог для добавления новой роли, а кнопки на карточках — с иконками карандаша, корзины и пользователя — обеспечивают редактирование, удаление, просмотр обладателей роли и назначение ролей. Проверка прав выполняется перед каждым действием, что исключает несанкционированные операции.

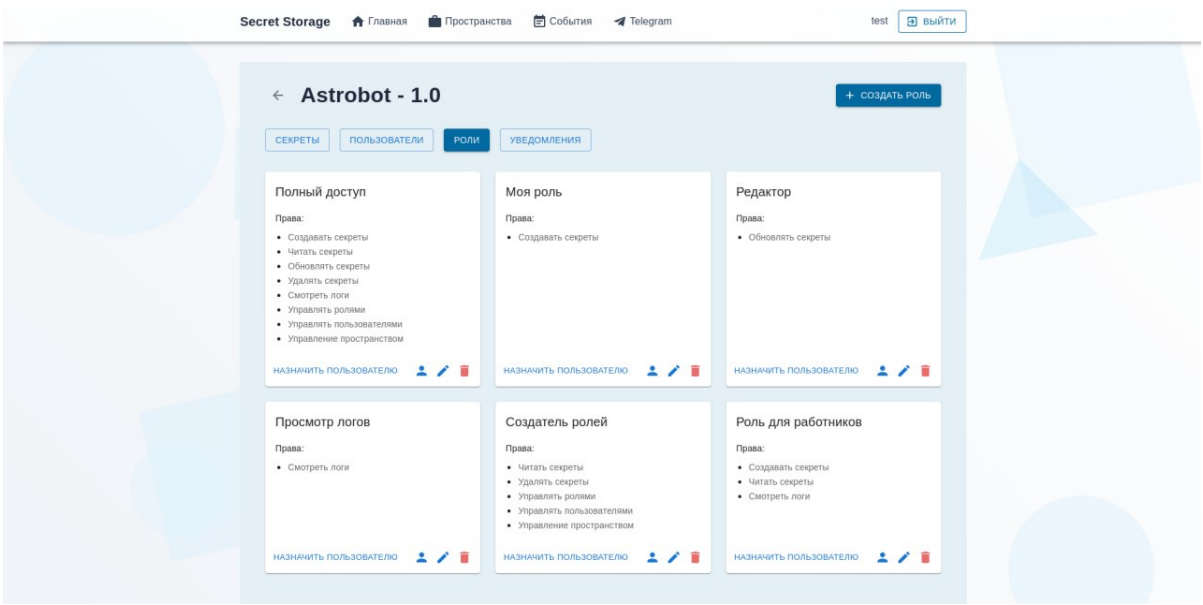

Рисунок 13 – Страница управления ролями пользователей

API-эндпоинты, используемые системой, включают:

- **GET /roles?workspace_id=:workspaceId**: Получает список ролей для рабочего пространства, отображая их на странице.
- **POST /roles**: Создает новую роль при нажатии кнопки "Создать роль".
- **PUT /roles/:id**: Обновляет существующую роль через функцию редактирования.
- **DELETE /roles/:id**: Удаляет роль и связанные привязки при выборе действия удаления.
- **GET /role-bindings/role/:roleId/workspace/:workspaceId**: Возвращает список пользователей с заданной ролью для отображения назначений.
- **POST /role-bindings**: Назначает роль пользователю через соответствующий диалог.
- **DELETE /role-bindings/:id**: Удаляет привязку роли к пользователю при отмене назначения.

3.5. Система управления пользователями рабочего пространства

Пользовательский интерфейс данной страницы предоставляет средства для управления пользователями рабочего пространства. Основная область содержит название рабочего пространства и вкладки навигации. Под вкладками находится поле "Поиск пользователей" для фильтрации списка и кнопка "Добавить пользователя" для добавления новых участников. Список пользователей отображает имена, роли и кнопки действий: "Назначить роль", "Отозвать роль" и "Исключить".

Данные о пользователях запрашиваются динамически с учетом активного рабочего пространства через GET-запрос. Функция поиска фильтрует список на стороне клиента в реальном времени.

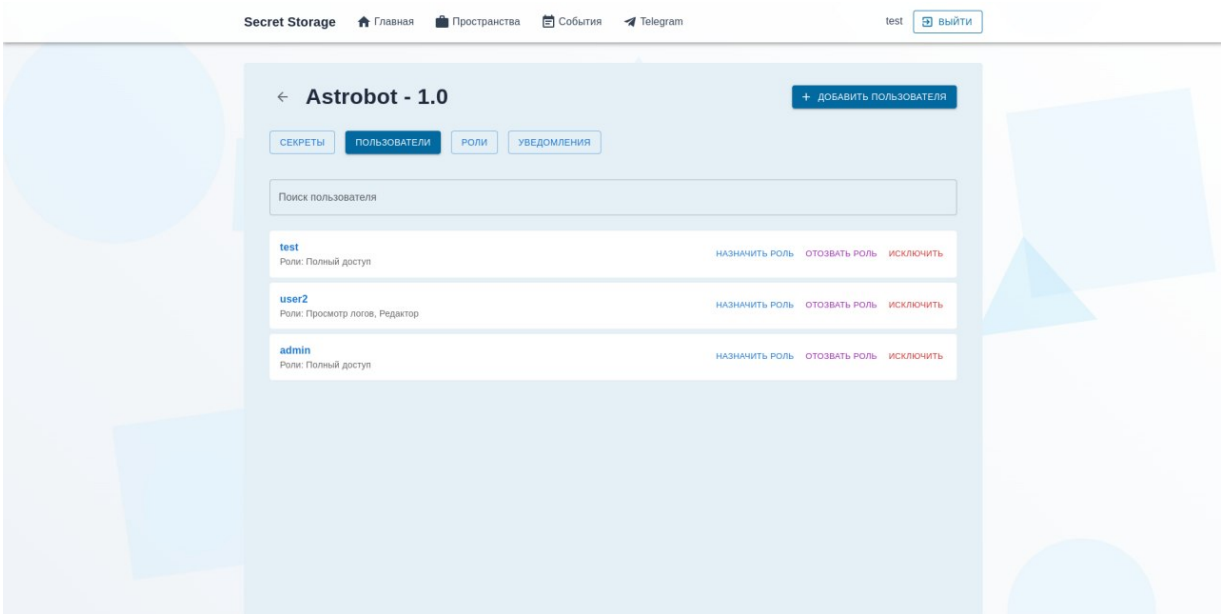

Рисунок 14 – Страница управления пользователями

API-эндпоинты, используемые на странице:

- **GET /workspace-users?workspace_id=:workspaceId**: возвращает список пользователей рабочего пространства, используется для заполнения списка.
- **POST /workspace-users**: добавляет нового пользователя, вызывается при нажатии на кнопку "Добавить пользователя".
- **DELETE /workspace-users/:id**: удаляет пользователя из рабочего пространства, активируется кнопкой "Исключить".

3.6. Система уведомлений о событиях в рабочем пространстве

Система уведомлений и событий позволяет пользователям получать информацию о ключевых действиях в рабочем пространстве (например, создание или удаление секретов), а

также управлять своими подписками на уведомления об этих событиях. Страница уведомлений реализована в React-компоненте NotificationsPage.

На странице представлен список событий, таких как "Создание", "Чтение", "Регистрация", "Удаление" и "Экспорт логов", каждое из которых сопровождается кнопкой для управления подпиской. Например, для события "Создание" доступна кнопка "Подписаться", а для "Удаление" — кнопка "Удалить", что соответствует текущему статусу подписки. При нажатии на кнопку "Подписаться" или "Удалить" появляется диалоговое окно с подтверждением действия, где пользователь может либо подтвердить, либо отменить операцию. После подписки уведомления отправляются через Telegram, что делает процесс информирования быстрым и доступным.

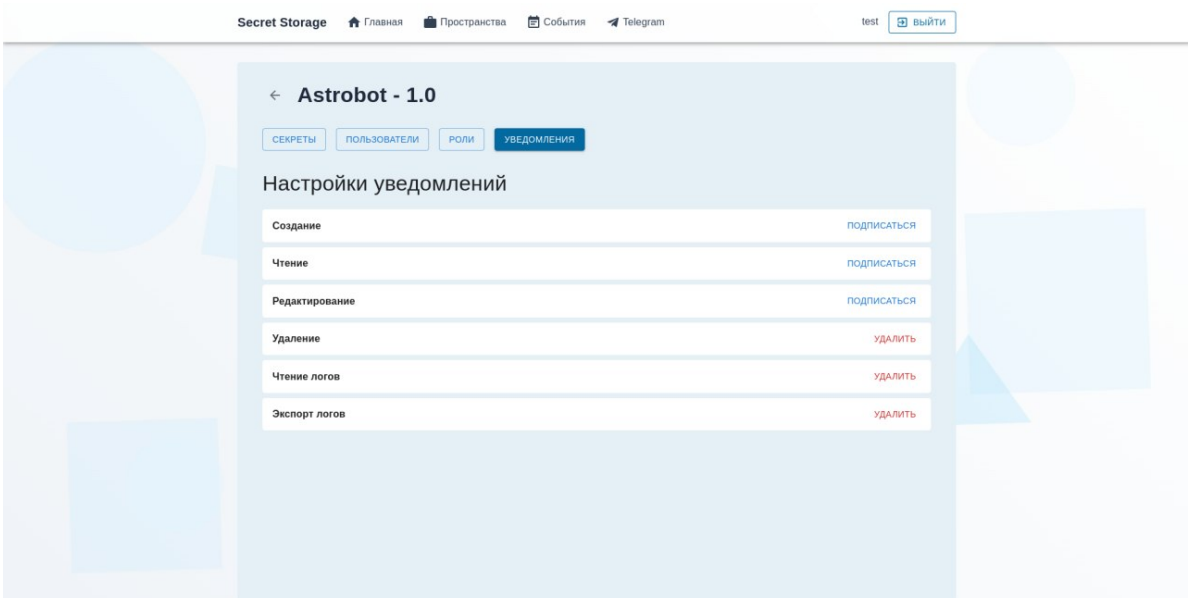

Рисунок 15 – Страница управления уведомлениями

Техническая реализация системы опирается на Telegram Bot для отправки уведомлений. Пользователи связывают свой Telegram-аккаунт с приложением, чтобы получать сообщения о событиях. Подписки хранятся в базе данных как Event Bindings, где каждая запись связывает пользователя, рабочее пространство и конкретное событие. Это позволяет системе точно определять, кому и о каких событиях нужно сообщить. При наступлении события приложение формирует уведомление и отправляет его через Telegram, обеспечивая оперативное информирование. Управление подписками происходит через запросы к API, которые обрабатывают создание, получение и удаление подписок.

API эндпоинты, используемые для взаимодействия с подсистемой уведомлений:

- **GET /event-bindings/bindings?workspace_id=:workspaceId**: Возвращает список подписок на события для указанного рабочего пространства, используется для отображения текущих подписок на странице уведомлений.

- **POST /event-bindings**: Создает новую подписку на событие, вызывается при нажатии кнопки "Подписаться".
- **DELETE /event-bindings/:id**: Удаляет подписку на событие, используется при нажатии кнопки "Удалить".

3.7. Система интеграции с Telegram

Для взаимодействия с системой интеграции с Telegram представлена соответствующая страница, где отображается текущий статус подключения, например, "Телеграм бот подключен к вашему аккаунту". Процесс начинается с генерации кода подтверждения, который пользователь копирует с помощью кнопки "Копировать код" и отправляет боту в Telegram для связывания аккаунта. После успешной привязки бот начинает отправлять уведомления о важных событиях, таких как создание секрета. Пользователь также может управлять интеграцией, отключив бота через кнопку "Отключить бота", что обеспечивает гибкость в использовании системы. Интерфейс минималистичен, с акцентом на простоту и понятность шагов.

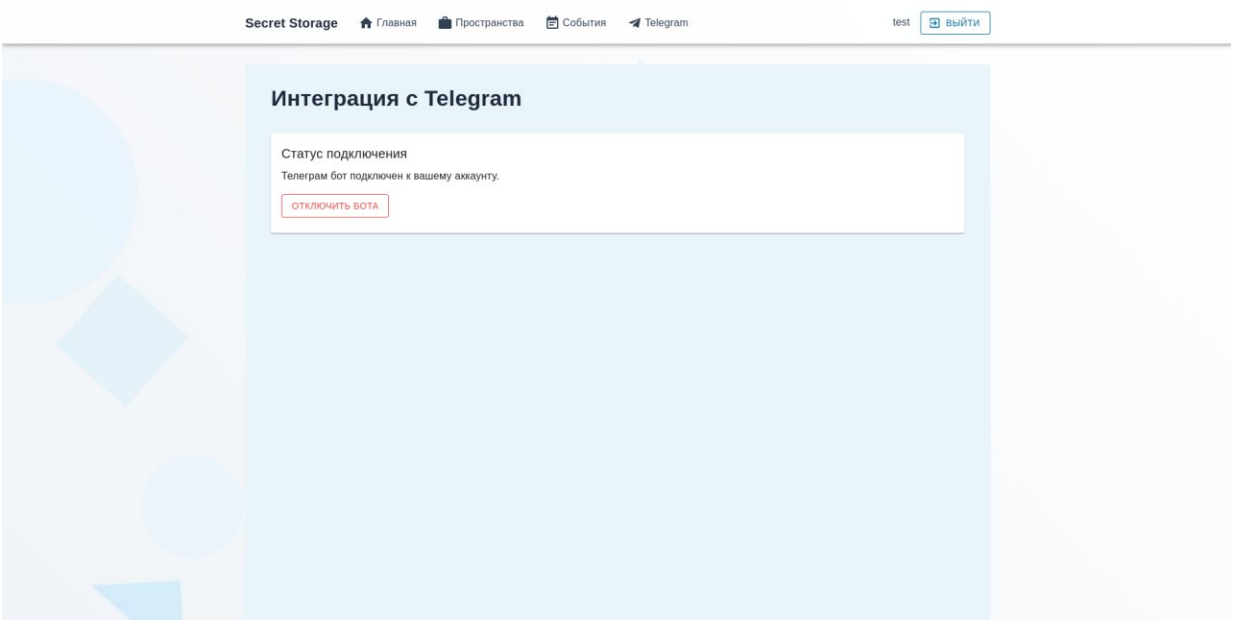

Рисунок 16 – Страница интеграции с Telegram

Система интеграции с Telegram построена на использовании Telegram Bot API, которое обеспечивает взаимодействие между приложением и ботом для обработки сообщений и отправки уведомлений. Для связывания аккаунта применяется система кодов подтверждения, генерируемых приложением и отправляемых пользователю, что гарантирует безопасную авторизацию. Среди функционала системы есть отправка уведомлений о ключевых событиях и управление подключением, позволяя пользователям отключать бота или изменять настройки уведомлений. Страница, реализуемая React-компонентом "TelegramPage" служит центральным

элементом интерфейса, интегрируя эти возможности и обеспечивая взаимодействие с серверными эндпоинтами.

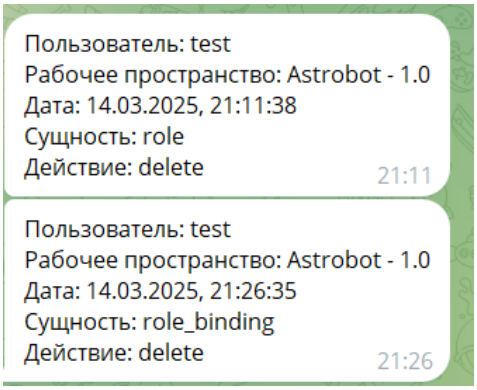

Рисунок 17 – Пример получаемых уведомлений

- **GET /chat-bindings**: Возвращает информацию о текущей привязке Telegram-аккаунта, используется для отображения статуса на странице интеграции.
- **POST /chat-bindings**: Инициирует создание новой привязки Telegram-аккаунта, вызывается при активации функции подключения.
- **PUT /chat-bindings**: Обновляет данные привязки, например, после ввода кода подтверждения пользователем.
- **DELETE /chat-bindings**: Удаляет существующую привязку Telegram-аккаунта, активируется при отключении бота.

3.8. Система аудита

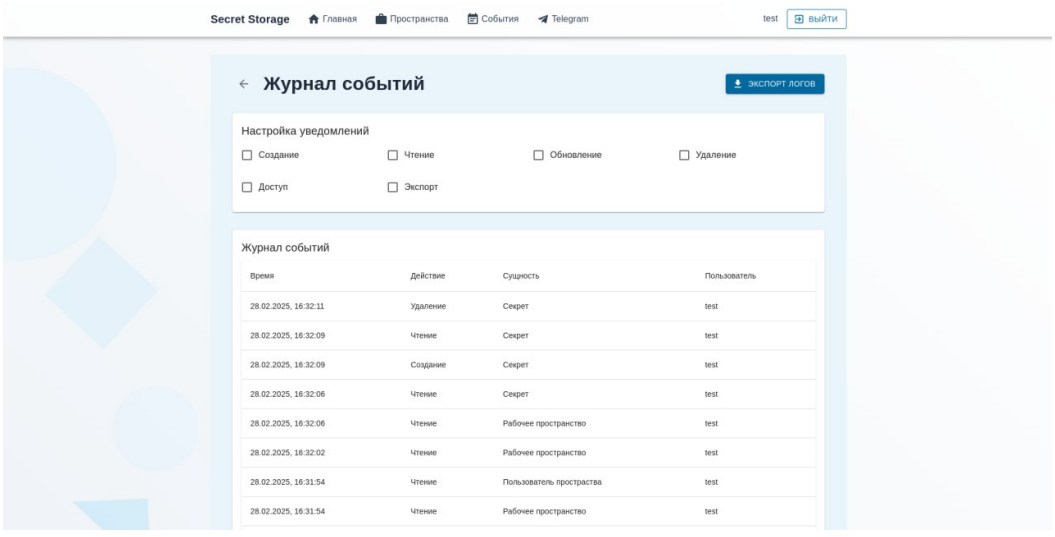

Рисунок 18 – Страница системы аудита

Пользовательский интерфейс страницы предлагает простой и интуитивно понятный способ взаимодействия с системой аудита. Пользователи могут выбирать типы действий для фильтрации (создание, редактирование, удаление, экспорт), что позволяет быстро находить

нужные записи. События отображаются в таблице с колонками для времени, действия, субъекта и дополнительных комментариев, обеспечивая удобный просмотр. Пользователи с разрешениями на экспорт логов имеют возможность экспортировать данные в CSV-файл с помощью кнопки, что упрощает дальнейший анализ вне системы.

Для экспорта логов применяется библиотека fs из Node.js. Для записи событий к каждой функции реализующей эндпоинт применяется специальный middleware. Система фиксирует ключевые события — создание, редактирование и удаление — с указанием времени и типа сущности, что поддерживает требования безопасности и аудита. Подход к реализации сосредоточен на простоте и масштабируемости, позволяя легко интегрировать новые типы действий в будущем.

- **GET /logs**: Получение списка всех логов с возможностью фильтрации по типу действия и временному диапазону.
- **POST /logs/export**: Экспорт логов в CSV-файл на основе заданных параметров фильтрации.
- **GET /logs/:id**: получение детальной информации о конкретной записи лога по его идентификатору.

Выводы по главе

Данная глава описывает программную реализацию веб-приложения для хранения корпоративной конфиденциальной информации. Приводятся подробности технической реализации и сценарии взаимодействия с приложением с точки зрения пользователя.

Заключение

Данное приложение было разработано с целью предоставить пользователям удобный и безопасный инструмент для управления конфиденциальной информацией в рамках командной работы. В современном мире, где данные играют ключевую роль, важно обеспечить надёжное хранение и контроль доступа к секретам, таким как пароли, API-ключи, токены и другие важные данные. Приложение решает эту задачу, предлагая централизованную платформу для создания, хранения и управления секретами с гибкой системой ролей и прав доступа. Это особенно полезно для команд разработчиков, DevOps-инженеров и других специалистов, которые работают с чувствительной информацией и нуждаются в эффективном инструменте для её защиты.

Приложение предлагает широкий набор функций, которые делают его универсальным решением для управления секретами. Пользователи могут создавать изолированные пространства для хранения секретов, что позволяет разделять данные между проектами или командами. Можно создавать, редактировать, удалять и просматривать секреты с поддержкой шифрования данных для обеспечения безопасности. Гибкая система ролей позволяет назначать пользователям различные права доступа к секретам и другим ресурсам. Интеграция с Telegram позволяет пользователям получать уведомления о важных событиях, таких как создание или удаление секретов. Все действия пользователей записываются системой аудита, что позволяет отслеживать изменения и обеспечивать прозрачность работы с данными.

Приложение можно внедрять в небольших компаниях, с числом сотрудников до 100 человек. В таком случае для его работы будет необходим сервер с 4 ядрами Cpu и 8 Gb Ram, также рекомендуется использовать сторонние системы резервного копирования для поддержания сохранности данных, на случай выхода из строя оборудования или хакерских атак.

Приложение имеет большой потенциал для дальнейшего развития. В будущем можно расширить количество интеграций и добавить поддержку других мессенджеров (например, Slack [22]) или систем мониторинга (например, Prometheus [23]) для отправки уведомлений. Также для пользователей было бы удобно наличие мобильного приложения для быстрого доступа к секретам и управления ими с мобильных устройств. Для DevOps-инженеров было бы полезным добавление функций автоматического обновления секретов (например, ротация API-ключей) и интеграция с CI/CD-системами [24] для автоматического использования секретов в процессах разработки.

Список используемой литературы

1. HashiCorp Vault [Электронный ресурс]. URL: https://www.vaultproject.io/ (дата обращения: 14.03.2025).
2. AWS Secrets Manager [Электронный ресурс]. URL: https://aws.amazon.com/secrets-manager/ (дата обращения: 14.03.2025).
3. Role-Based Access Control (RBAC) [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/Role-based_access_control (дата обращения: 14.03.2025).
4. Keycloak: Open Source Identity and Access Management [Электронный ресурс]. URL: https://www.keycloak.org/ (дата обращения: 14.03.2025).
5. Multi-Factor Authentication (MFA) [Электронный ресурс]. URL: https://www.nist.gov/itl/applied-cybersecurity/tig/back-basics-multi-factor-authentication (дата обращения: 14.03.2025).
6. Telegram Bots: An Introduction for Developers [Электронный ресурс]. URL: https://core.telegram.org/bots (дата обращения: 14.03.2025).
7. 1Password: Password Manager for Families, Businesses, Teams [Электронный ресурс]. URL: https://1password.com/ (дата обращения: 14.03.2025).
8. JSON Web Tokens (JWT) [Электронный ресурс]. URL: https://jwt.io/ (дата обращения: 14.03.2025).
9. What is IAM? [Электронный ресурс]. URL: https://aws.amazon.com/iam/ (дата обращения: 14.03.2025).
10. LastPass: Password Manager for Business [Электронный ресурс]. URL: https://www.lastpass.com/business (дата обращения: 14.03.2025).
11. React – A JavaScript library for building user interfaces [Электронный ресурс]. URL: https://reactjs.org/ (дата обращения: 14.03.2025).
12. REST API Tutorial [Электронный ресурс]. URL: https://restfulapi.net/ (дата обращения: 14.03.2025).
13. Unified Modeling Language (UML) [Электронный ресурс]. URL: https://www.uml.org/ (дата обращения: 14.03.2025).
14. The C4 model for visualising software architecture [Электронный ресурс]. URL: https://c4model.com/ (дата обращения: 14.03.2025).
15. PostgreSQL: The World's Most Advanced Open Source Relational Database [Электронный ресурс]. URL: https://www.postgresql.org/ (дата обращения: 14.03.2025).
16. Express - Node.js web application framework [Электронный ресурс]. URL: https://expressjs.com/ (дата обращения: 14.03.2025).
17. TypeORM: ORM for TypeScript and JavaScript [Электронный ресурс]. URL: https://typeorm.io/ (дата обращения: 14.03.2025).
18. node-telegram-bot-api [Электронный ресурс]. URL: https://github.com/yagop/node-telegram-bot-api (дата обращения: 14.03.2025).
19. OAuth 2.0 Authorization Framework [Электронный ресурс]. URL: https://oauth.net/2/ (дата обращения: 14.03.2025).
20. OpenID Connect [Электронный ресурс]. URL: https://openid.net/connect/ (дата обращения: 14.03.2025).
21. Redis: The open source, in-memory data store [Электронный ресурс]. URL: https://redis.io/ (дата обращения: 14.03.2025).
22. Slack: Where work happens [Электронный ресурс]. URL: https://slack.com/ (дата обращения: 14.03.2025).

23. Prometheus - Monitoring system & time series database [Электронный ресурс]. URL: https://prometheus.io/ (дата обращения: 14.03.2025).
24. What is CI/CD? [Электронный ресурс]. URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd (дата обращения: 14.03.2025).