**Департамент образования и науки города Москвы**

**Государственное бюджетное профессиональное образовательное учреждение**

**города Москвы**

**«Колледж малого бизнеса № 4»**

**(ГБПОУ КМБ № 4)**

**КУРСОВОЙ ПРОЕКТ**

по профессиональному модулю ПМ.06 Содержание информационных систем

по специальности СПО 09.02.07 Информационные системы и программирование  
Тема: Разработка и внедрение системы контроля доступа в информационных системах

Выполнил(а) студент(ка)

4 курса группы № ИПО-41.21

Лисенков Даниил Алексеевич

(Фамилия, имя, отчество студента)

Проверил

руководитель курсовой работы

ТЕСТ ТЕСТ ТЕСТ

Работа защищена с оценкой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025

Москва, 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc189854348)

[Теоретический раздел 7](#_Toc189854349)

[1.1 Основные принципы контроля доступа в информационных системах. 7](#_Toc189854350)

[1.2 Модели контроля доступа: DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role-Based Access Control), ABAC (Attribute-Based Access Control). 10](#_Toc189854351)

[1.3 Современные методы аутентификации и авторизации (MFA, SSO, OAuth 2.0, OpenID Connect, Kerberos). 12](#_Toc189854352)

[1.4 Вопросы безопасности при реализации контроля доступа (защита от несанкционированного доступа, аудит событий, шифрование). 14](#_Toc189854353)

[1.5 Нормативные требования и стандарты (ISO/IEC 27001, NIST, ГОСТ Р 57580, GDPR). 17](#_Toc189854354)

[Исследовательский раздел 21](#_Toc189854355)

[2.1 Анализ существующих решений для контроля доступа в информационных системах. 21](#_Toc189854356)

[2.2 Определение требований к разрабатываемой системе контроля доступа. 23](#_Toc189854357)

[2.3 Выбор технологий и инструментов для реализации системы (LDAP, Active Directory, Keycloak, FreeIPA). 25](#_Toc189854358)

[2.4 Разработка архитектуры системы контроля доступа и модели разграничения прав пользователей 28](#_Toc189854359)

[2.5 Анализ угроз и рисков при внедрении системы контроля доступа. 30](#_Toc189854360)

[Практический раздел 34](#_Toc189854361)

[3.1 Разработка и настройка системы контроля доступа в информационной системе. 34](#_Toc189854362)

[3.2 Реализация механизма аутентификации и авторизации пользователей. 37](#_Toc189854363)

[3.3 Настройка ролевой модели доступа и политик безопасности. 39](#_Toc189854364)

[3.4 Интеграция системы контроля доступа с другими сервисами (база данных, веб-приложения, облачные сервисы). 42](#_Toc189854365)

[3.5 Проведение тестирования системы на безопасность, отказоустойчивость и производительность. 45](#_Toc189854366)

[3.6 Разработка эксплуатационной документации по настройке и сопровождению системы контроля доступа. 48](#_Toc189854367)

[ЗАКЛЮЧЕНИЕ 51](#_Toc189854368)

[ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА 53](#_Toc189854369)

[ПРИЛОЖЕНИЕ 55](#_Toc189854370)

# **ВВЕДЕНИЕ**

Современные информационные технологии стремительно развиваются, что приводит к необходимости создания эффективных механизмов защиты данных и контроля доступа к различным информационным системам. В условиях активного цифрового взаимодействия, обмена конфиденциальными данными, хранения персональной информации и управления ресурсами возрастает потребность в надежных системах аутентификации и авторизации пользователей. Безопасность информационных систем играет ключевую роль в защите от несанкционированного доступа, утечки данных и потенциальных киберугроз. Разработка и внедрение системы контроля доступа становятся важными задачами, направленными на обеспечение безопасности информационной среды.

В данной курсовой работе рассматривается процесс проектирования и реализации системы контроля доступа в информационных системах, разработанной с использованием современных веб-технологий, включая Django, HTML, Python, CSS, JavaScript и SQL. Выбор данной темы обусловлен высокой актуальностью проблемы защиты данных в цифровой среде, необходимостью обеспечения безопасности пользователей и предотвращения несанкционированного использования информационных ресурсов. Разработка данной системы позволит на практике изучить механизмы разграничения прав пользователей, принципы работы современных протоколов аутентификации, а также методы защиты информации в веб-приложениях.

В рамках данной работы будет проведен детальный анализ предметной области, рассмотрены существующие методы контроля доступа и их практическое применение в информационных системах. Особое внимание будет уделено вопросам проектирования архитектуры системы, выбору оптимальных технологий для реализации функционала аутентификации и авторизации пользователей, а также обеспечению надежности и устойчивости системы к возможным атакам.

Разработка системы контроля доступа включает в себя несколько ключевых этапов. На первом этапе будет произведено изучение современных подходов к организации аутентификации и авторизации, анализ существующих решений, включая ролевую модель доступа (RBAC) и модель на основе атрибутов (ABAC). Будут рассмотрены механизмы хранения и шифрования пользовательских данных, способы защиты от атак типа SQL-инъекций, межсайтового скриптинга (XSS) и подделки межсайтовых запросов (CSRF). На втором этапе будет спроектирована структура системы, определены основные компоненты, разработана модель базы данных и спроектирован интерфейс взаимодействия пользователей с системой. В дальнейшем будет проведена непосредственная разработка веб-приложения на основе Django, включающая создание пользовательской системы регистрации и авторизации, реализацию механизмов разграничения доступа и управление учетными записями пользователей. Особое внимание будет уделено тестированию системы, в ходе которого будут проверены функциональность механизмов аутентификации и авторизации, устойчивость к потенциальным угрозам, удобство использования и производительность системы. По завершению разработки система контроля доступа будет представлена в виде готового веб-приложения, обеспечивающего безопасное управление пользователями и их правами в информационной среде.

Значимость данной работы заключается в том, что разработанная система может быть адаптирована для различных сфер применения, включая корпоративные информационные системы, образовательные платформы, интернет-магазины и другие веб-ресурсы, требующие надежного механизма контроля доступа. В результате внедрения данной системы можно повысить уровень безопасности данных, минимизировать риски несанкционированного использования информации и создать удобный инструмент для управления пользователями в информационной системе.

Таким образом, данная курсовая работа направлена на разработку эффективного решения для контроля доступа к информационным системам, учитывая современные требования к безопасности и защите данных. Итогом работы станет полнофункциональное веб-приложение, демонстрирующее принципы безопасной аутентификации, авторизации и управления учетными записями пользователей. В заключении будут представлены основные выводы, оценка достигнутых результатов и рекомендации по дальнейшему развитию системы.

# **Теоретический раздел**

# **1.1 Основные принципы контроля доступа в информационных системах.**

Основные принципы контроля доступа в информационных системах представляют собой фундаментальную основу обеспечения безопасности современных цифровых платформ. Контроль доступа является критически важным элементом защиты информации, позволяющим регулировать взаимодействие субъектов и объектов информационной среды.

В контексте современных информационных технологий управление доступом к ресурсам представляет собой сложный процесс, который требует комплексного подхода. Принципиальная значимость данного механизма заключается в том, что он позволяет определять, какие пользователи или группы пользователей могут получать доступ к определенным данным, функциям или сервисам информационной системы.

Системы контроля доступа строятся на основе нескольких ключевых концепций, которые формируют основу их функционирования. Во-первых, это идентификация субъектов, которая предполагает уникальное определение каждого пользователя системы. Этот процесс обычно реализуется через предоставление учетных данных, таких как имя пользователя и пароль, либо через использование более сложных механизмов аутентификации, например, многофакторной проверки подлинности.

Далее следует аутентификация, представляющая собой процедуру подтверждения личности пользователя. Эта операция может быть выполнена различными способами, начиная от простой проверки пароля и заканчивая использованием биометрических данных или криптографических ключей. В современных системах особое внимание уделяется повышению надежности аутенитификационных процедур, что обусловлено постоянным ростом угроз кибербезопасности.

Авторизация занимает центральное место в механизмах управления доступом. Она определяет права конкретного пользователя или группы пользователей относительно различных объектов системы. Процесс авторизации может быть реализован различными способами, от простых списков разрешений до сложных моделей дискреционного и мандатного контроля доступа. При этом важно отметить, что эффективность системы авторизации во многом зависит от грамотно спроектированной структуры прав и ролей.

Модель управления доступом должна учитывать не только текущие потребности организации, но и перспективы ее развития. Это требует создания гибкой архитектуры, способной адаптироваться к изменяющимся условиям эксплуатации. Особую роль играет возможность масштабирования системы, что особенно актуально для крупных организаций с распределенной структурой управления.

Современные подходы к контролю доступа все чаще включают элементы контекстно-ориентированной авторизации, учитывающей дополнительные факторы при принятии решений о предоставлении доступа. Такими факторами могут быть время дня, местоположение пользователя, используемое устройство и другие параметры, характеризующие ситуацию запроса на доступ.

Важным аспектом является также обеспечение аудита действий пользователей в информационной системе. Это позволяет отслеживать все операции, связанные с доступом к защищенным ресурсам, и выявлять потенциальные нарушения безопасности. Системы мониторинга должны предоставлять детальную информацию о всех событиях, связанных с управлением доступом, включая успешные и неуспешные попытки аутентификации, изменения прав доступа и другие значимые события.

Технологические решения для реализации систем контроля доступа постоянно совершенствуются, что связано с развитием новых методов защиты информации и появлением новых угроз. Современные системы управления доступом часто интегрируются с другими компонентами инфраструктуры безопасности, такими как системы обнаружения вторжений, средства защиты от вредоносного ПО и платформы управления событиями безопасности.

При проектировании систем контроля доступа необходимо учитывать специфику предметной области, в которой они будут применяться. Это может существенно влиять на выбор конкретных технологий и методов реализации. Например, требования к системам контроля доступа в финансовых организациях значительно отличаются от требований, предъявляемых к аналогичным системам в сфере здравоохранения или образовательных учреждениях.

Таким образом, основные принципы контроля доступа в информационных системах представляют собой сложный комплекс взаимосвязанных элементов, обеспечивающих защиту информации на различных уровнях. Грамотное проектирование и реализация этих принципов являются залогом создания надежной системы управления доступом, способной эффективно противостоять современным угрозам информационной безопасности.

# **1.2 Модели контроля доступа: DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role-Based Access Control), ABAC (Attribute-Based Access Control).**

Модели контроля доступа представляют собой фундаментальные концептуальные рамки, определяющие способы управления правами субъектов на объекты информационной системы. Существует несколько основных подходов к организации контроля доступа, каждый из которых имеет свои особенности, преимущества и области применения. Рассмотрим наиболее значимые модели: DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role-Based Access Control) и ABAC (Attribute-Based Access Control).

Дискреционный контроль доступа (DAC) является одним из самых распространенных подходов к управлению правами в информационных системах. Его основная характеристика заключается в том, что владельцы ресурсов имеют право самостоятельно устанавливать правила доступа к ним. Это означает, что пользователь, обладающий определенным объектом, может решать, кто и какие действия может выполнять с этим объектом. Такой подход обеспечивает высокую гибкость и удобство использования, поскольку предоставляет владельцам полномочия по регулированию доступа в соответствии с их потребностями. Однако при этом возникает риск несанкционированного расширения прав доступа, если владельцы объектов не будут достаточно внимательны при установке разрешений.

Мандатный контроль доступа (MAC) представляет собой более строгий подход к управлению доступом, где права определяются центральной политикой безопасности, а не владельцами объектов. В этой модели каждому субъекту и объекту присваивается набор меток безопасности, которые определяют уровень доступа. Доступ разрешается только тогда, когда метки субъекта соответствуют меткам объекта согласно установленным правилам. MAC широко используется в системах, где требуется повышенная безопасность, например, в государственных и военных организациях. Преимуществом данной модели является высокая степень контроля над доступом, однако она менее гибкая по сравнению с DAC и требует значительных усилий для настройки и поддержания.

Ролевой контроль доступа (RBAC) основывается на использовании ролей для управления правами пользователей. В этой модели роли представляют собой группы прав, которые назначаются пользователям в зависимости от их функциональных обязанностей в организации. Это позволяет существенно упростить процесс управления доступом, так как вместо индивидуального назначения прав для каждого пользователя администраторы могут работать с предопределенными наборами разрешений. RBAC особенно эффективен в крупных организациях, где существует четкая иерархия должностей и ответственности. При этом модель остается достаточно гибкой, позволяя адаптироваться к изменяющимся требованиям бизнеса.

Атрибутивный контроль доступа (ABAC) представляет собой современный подход, который основывается на использовании атрибутов для принятия решений о доступе. В отличие от других моделей, где права обычно статичны, ABAC позволяет динамически определять доступ на основе различных параметров, таких как время дня, местоположение пользователя, устройство, с которого осуществляется запрос, и многие другие. Это делает ABAC особенно подходящим для сложных и динамичных сред, где традиционные модели могут оказаться недостаточно гибкими. Однако реализация ABAC требует значительных вычислительных ресурсов и сложной настройки правил, что может стать препятствием для его внедрения в некоторых случаях.

Каждая из этих моделей имеет свои особенности и области применения. Выбор конкретной модели или их комбинации зависит от специфики организации, уровня требуемой безопасности и технических возможностей. Например, в корпоративных системах часто используется сочетание RBAC для базового управления доступом и ABAC для дополнительной защиты критически важных ресурсов. В то же время, в системах с высокими требованиями безопасности может применяться MAC, дополняемый элементами DAC для обеспечения необходимой гибкости.

При проектировании систем контроля доступа важно учитывать взаимодействие между различными моделями и возможность их интеграции. Современные информационные системы часто требуют комплексного подхода к управлению доступом, сочетающего преимущества разных моделей для достижения оптимального баланса между безопасностью и удобством использования. Это особенно актуально в условиях постоянно меняющихся требований к защите информации и развития новых технологий.

# **1.3 Современные методы аутентификации и авторизации (MFA, SSO, OAuth 2.0, OpenID Connect, Kerberos).**

Современные методы аутентификации и авторизации представляют собой сложные системы, обеспечивающие безопасный доступ к информационным ресурсам в условиях постоянно меняющихся технологий и угроз. Эти механизмы играют ключевую роль в защите данных и подтверждении личности пользователей в цифровой среде. Рассмотрим основные подходы: многофакторную аутентификацию (MFA), единую систему входа (SSO), протокол OAuth 2.0, OpenID Connect и Kerberos.

Многофакторная аутентификация (MFA) представляет собой механизм, требующий от пользователя предоставления двух или более независимых факторов проверки подлинности для получения доступа к защищенным ресурсам. Эти факторы могут быть разделены на три категории: что вы знаете (например, пароль), что у вас есть (например, смартфон или токен) и кто вы есть (например, биометрические данные). MFA значительно повышает уровень безопасности по сравнению с использованием только паролей, так как злоумышленнику необходимо получить доступ ко всем факторам одновременно. Современные реализации MFA часто используют комбинацию различных факторов, таких как одноразовые пароли, отправляемые по SMS или через мобильное приложение, биометрическое сканирование и аппаратные токены.

Единая система входа (Single Sign-On, SSO) позволяет пользователям аутентифицироваться один раз для получения доступа к множеству взаимосвязанных, но независимых систем. Этот метод существенно упрощает процесс работы с несколькими приложениями, так как пользователь не должен запоминать множество учетных данных и выполнять повторную аутентификацию при переходе между сервисами. Однако SSO также создает потенциальную точку уязвимости, так как компрометация одной точки входа может дать злоумышленнику доступ ко всем связанным системам. Для минимизации рисков современные реализации SSO часто интегрируются с другими методами защиты, такими как MFA.

OAuth 2.0 является протоколом авторизации, который позволяет третьим сторонам получать ограниченный доступ к API без необходимости раскрывать учетные данные пользователя. Этот протокол широко используется веб-приложениями, мобильными приложениями и серверами для делегирования прав доступа. OAuth 2.0 работает через токены доступа, которые выдаются после успешной аутентификации пользователя. Эти токены имеют ограниченное время жизни и права, что обеспечивает повышенную безопасность. Протокол поддерживает несколько типов грантов, каждый из которых предназначен для определенного сценария использования, например, авторизация через веб-браузер или мобильное приложение.

OpenID Connect представляет собой простой протокол аутентификации, построенный поверх OAuth 2.0. Он предоставляет возможность подтверждения личности пользователя через стандартные HTTP запросы и возвращает информацию о пользователе в виде JSON Web Tokens (JWT). OpenID Connect стал популярным благодаря своей простоте и совместимости с существующими решениями на базе OAuth 2.0. Этот протокол активно используется крупными поставщиками услуг, такими как Google, Microsoft и Facebook, для обеспечения возможности входа через существующие учетные записи.

Kerberos — это сетевой протокол аутентификации, работающий на основе билетов (tickets). Он использует симметричное шифрование для обеспечения безопасности обмена данными и предотвращения несанкционированного доступа. Kerberos работает через доверенный центр аутентификации, называемый Key Distribution Center (KDC), который выдает билеты для проверки подлинности. Преимуществом этого протокола является его способность обеспечивать безопасный обмен данными даже в незащищенных сетях, поскольку все данные передаются в зашифрованном виде. Kerberos широко используется в корпоративных сетях, особенно в системах Windows Active Directory.

Каждый из этих методов имеет свои особенности и области применения. Выбор конкретного решения зависит от специфики организации, уровня требуемой безопасности и технических возможностей. Например, в корпоративных системах часто используется комбинация Kerberos для внутренней аутентификации и SSO для упрощения работы с множеством приложений. В то же время, для внешних пользователей и мобильных приложений больше подходит использование OAuth 2.0 и OpenID Connect, дополняемых MFA для повышения безопасности.

При внедрении современных методов аутентификации и авторизации важно учитывать их взаимодействие и возможность интеграции. Современные информационные системы часто требуют комплексного подхода, сочетающего преимущества разных методов для достижения оптимального баланса между безопасностью и удобством использования. Это особенно актуально в условиях постоянного развития новых технологий и усложнения угроз кибербезопасности.

# **1.4 Вопросы безопасности при реализации контроля доступа (защита от несанкционированного доступа, аудит событий, шифрование).**

Вопросы безопасности при реализации систем контроля доступа занимают центральное место в обеспечении защиты информационных ресурсов. Это сложная и многоаспектная задача, требующая комплексного подхода к решению различных проблем, связанных с предотвращением несанкционированного доступа, мониторингом событий и защитой данных. Рассмотрим основные аспекты, которые необходимо учитывать при проектировании и внедрении таких систем.

Защита от несанционированного доступа является фундаментальным элементом любой системы контроля доступа. Этот процесс охватывает широкий спектр мер, направленных на минимизацию рисков проникновения злоумышленников в информационную систему. Прежде всего, важно использовать современные методы аутентификации, такие как многофакторная аутентификация (MFA), которая значительно повышает уровень безопасности по сравнению с традиционными парольными системами. Однако даже при наличии MFA остаются потенциальные уязвимости, такие как фишинговые атаки, использование украденных учетных данных или эксплуатация уязвимостей в программном обеспечении. Для противодействия этим угрозам применяются дополнительные механизмы, например, адаптивная аутентификация, учитывающая контекст запроса, или системы обнаружения аномалий, анализирующие поведение пользователей.

Дополнительным уровнем защиты от несанкционированного доступа служат техники изоляции и сегментации сетей. Эти подходы позволяют ограничить возможности злоумышленника в случае успешной атаки, предотвращая распространение доступа по всей системе. Внедрение зон безопасности, разделение внутренних и внешних сетей, а также использование брандмауэров и систем обнаружения вторжений создают дополнительные барьеры для потенциальных атакующих.

Аудит событий представляет собой систематический процесс сбора, анализа и хранения информации о действиях, происходящих в информационной системе. Эта функция играет ключевую роль в обеспечении безопасности, так как позволяет выявлять подозрительные действия, отслеживать изменения прав доступа и проводить расследования инцидентов. Современные системы аудита способны регистрировать широкий круг событий, включая попытки входа в систему, изменения конфигурации, операции с данными и административные действия. Данные аудита должны быть защищены от несанкционированного доступа и модификации, чтобы обеспечить их достоверность.

Эффективный аудит требует не только записи событий, но и их регулярного анализа. Для этого используются специализированные инструменты, способные автоматически обрабатывать большие объемы данных, выявлять аномалии и формировать отчеты. Важно также учитывать требования законодательства и стандартов, предъявляемые к срокам хранения аудиторских записей и условиям их доступности. Например, в некоторых отраслях требуется хранить данные аудита в течение нескольких лет для целей комплаенса.

Шифрование данных является одним из наиболее надежных способов защиты информации как в процессе передачи, так и в состоянии покоя. При реализации систем контроля доступа шифрование используется для защиты учетных данных, токенов авторизации и других чувствительных объектов. Современные криптографические алгоритмы обеспечивают высокий уровень безопасности, однако их эффективность зависит от правильной реализации и управления ключами.

Защита данных в процессе передачи осуществляется через протоколы, такие как TLS (Transport Layer Security), обеспечивающие конфиденциальность и целостность информации при обмене между клиентом и сервером. Для защиты данных в состоянии покоя применяются различные методы шифрования, включая полное дисковое шифрование, шифрование файловых систем и баз данных. Особое внимание уделяется управлению криптографическими ключами, так как компрометация ключей может полностью нивелировать преимущества шифрования.

При проектировании систем контроля доступа необходимо учитывать взаимосвязь между различными аспектами безопасности. Например, шифрование должно работать в тесной интеграции с механизмами аутентификации и аудита, чтобы обеспечить полную защиту данных. Также важно помнить, что безопасность — это процесс, а не состояние. Системы контроля доступа должны постоянно развиваться и адаптироваться к меняющимся условиям и новым угрозам.

Современные подходы к обеспечению безопасности все чаще включают элементы искусственного интеллекта и машинного обучения для прогнозирования и предотвращения угроз. Такие технологии могут анализировать огромные объемы данных, выявлять скрытые паттерны и предсказывать потенциальные атаки, что существенно повышает эффективность защиты. Однако внедрение этих технологий требует значительных инвестиций и квалифицированных специалистов.

Таким образом, вопросы безопасности при реализации контроля доступа являются многогранными и требуют всестороннего подхода. Защита от несанкционированного доступа, аудит событий и шифрование данных представляют собой лишь часть комплекса мер, необходимых для создания надежной системы управления доступом. Грамотное сочетание этих элементов с учетом специфики организации и требований безопасности позволит создать эффективную защиту информационных ресурсов. В условиях постоянного развития новых технологий и усложнения угроз важно продолжать совершенствовать существующие решения и внедрять инновационные подходы к обеспечению безопасности.

# **1.5 Нормативные требования и стандарты (ISO/IEC 27001, NIST, ГОСТ Р 57580, GDPR).**

Нормативные требования и стандарты в области контроля доступа к информации играют ключевую роль в обеспечении безопасности информационных систем. Они представляют собой комплекс правил, рекомендаций и требований, разработанных для создания единых подходов к управлению доступом и защите данных. Рассмотрим наиболее значимые международные и национальные стандарты: ISO/IEC 27001, NIST, ГОСТ Р 57580 и GDPR.

ISO/IEC 27001 является международным стандартом, определяющим требования к системе управления информационной безопасностью (ISMS). Этот стандарт предоставляет организации фреймворк для разработки, внедрения, поддержания и постоянного улучшения системы защиты информации. В контексте контроля доступа ISO/IEC 27001 предлагает комплексный подход, включающий оценку рисков, определение мер по управлению доступом, а также регулярную проверку их эффективности. Стандарт акцентирует внимание на необходимости разработки политики управления доступом, которая должна охватывать все аспекты взаимодействия с информационными активами. ISO/IEC 27001 также требует внедрения процедур мониторинга, аудита и анализа событий, связанных с управлением доступом, что позволяет своевременно выявлять и предотвращать потенциальные инциденты.

NIST (National Institute of Standards and Technology) представляет собой американскую организацию, разрабатывающую широкий спектр руководств и стандартов в области информационной безопасности. В частности, спецификация NIST Special Publication 800-53 содержит подробные рекомендации по управлению доступом. Эти рекомендации охватывают различные аспекты, такие как идентификация пользователей, аутентификация, авторизация, аудит действий и защита от несанкционированного доступа. Особое внимание уделяется применению многофакторной аутентификации, шифрованию данных и использованию современных технологий для повышения уровня безопасности. NIST также предлагает методологии оценки рисков и тестирования систем управления доступом, что позволяет организациям адаптировать свои процессы к изменяющимся условиям и угрозам.

ГОСТ Р 57580 — это российский стандарт, принятый в рамках реализации Федерального закона «Об информации, информационных технологиях и о защите информации». Он регламентирует требования к защите персональных данных при их обработке в информационных системах персональных данных (ИСПДн). В части управления доступом этот стандарт определяет необходимость разработки и внедрения механизмов, обеспечивающих защиту персональных данных от несанкционированного доступа. Это включает установление правил идентификации, аутентификации и авторизации пользователей, а также ограничение доступа к данным только тем лицам, которым это необходимо для выполнения своих должностных обязанностей. Кроме того, ГОСТ Р 57580 требует ведения журналов событий и проведения регулярных проверок соблюдения установленных правил.

GDPR (General Data Protection Regulation) — это европейский закон, регулирующий защиту персональных данных граждан Европейского Союза. Хотя этот регламент не является техническим стандартом, он накладывает строгие требования на управление доступом к персональным данным. GDPR требует, чтобы доступ к данным был ограничен только теми сотрудниками или системами, которые действительно нуждаются в них для выполнения своих функций. Организации должны внедрять механизмы мониторинга и аудита, а также иметь четкие процедуры реагирования на инциденты, связанные с утечкой данных. Нарушение положений GDPR может привести к серьезным финансовым штрафам, что стимулирует компании к тщательному планированию и реализации систем управления доступом.

Каждый из этих стандартов имеет свои особенности и области применения. ISO/IEC 27001 предоставляет универсальный фреймворк для управления информационной безопасностью, который может быть адаптирован к различным типам организаций. NIST предлагает детальные технические рекомендации, основанные на передовых практиках США. ГОСТ Р 57580 сосредоточен на защите персональных данных в России, а GDPR регулирует обращение с такими данными в Европейском Союзе. При этом важно отметить, что многие принципы этих стандартов пересекаются, что позволяет организациям создавать гармонизированные системы управления доступом, соответствующие множеству нормативных требований.

При проектировании и внедрении систем контроля доступа необходимо учитывать взаимодействие различных стандартов и требований. Современные информационные системы часто работают в условиях глобализации, где данные могут обрабатываться в разных юрисдикциях, что требует соблюдения нескольких нормативных баз одновременно. Это делает особенно важным выбор гибких решений, способных адаптироваться к меняющимся требованиям законодательства и стандартов.

Таким образом, нормативные требования и стандарты в области контроля доступа представляют собой сложную систему правил и рекомендаций, направленных на обеспечение безопасности информационных ресурсов. Их соблюдение является обязательным условием для создания надежных систем управления доступом, способных противостоять современным угрозам и удовлетворять требованиям законодательства. При этом важно помнить, что успешное внедрение этих стандартов требует не только технических решений, но и изменения корпоративной культуры, направленной на повышение осведомленности сотрудников о вопросах информационной безопасности.

# **Исследовательский раздел**

# **2.1 Анализ существующих решений для контроля доступа в информационных системах.**

Анализ существующих решений для контроля доступа в информационных системах представляет собой комплексное исследование различных технологий, платформ и подходов, которые уже внедрены или разрабатываются для обеспечения безопасности данных и управления правами пользователей. Этот процесс требует детального изучения как коммерческих, так и открытых решений, а также их сравнения с точки зрения функциональности, масштабируемости, безопасности и удобства использования.

На современном рынке представлен широкий спектр продуктов, предназначенных для реализации механизмов контроля доступа. Одним из наиболее распространенных направлений являются решения на основе облачных технологий. Облачная инфраструктура позволяет создавать гибкие системы управления доступом, которые могут легко адаптироваться к изменяющимся требованиям бизнеса. Эти решения часто включают встроенные механизмы многофакторной аутентификации, единого входа и динамического управления правами, что делает их привлекательными для компаний разных масштабов.

Коммерческие решения для контроля доступа обычно предлагают комплексный набор функций, охватывающих все аспекты управления безопасностью. Такие продукты часто включают продвинутые аналитические инструменты для мониторинга активности пользователей, автоматизированного обнаружения угроз и реагирования на инциденты. Они также поддерживают интеграцию с другими системами безопасности, такими как системы управления событиями безопасности (SIEM) и средства защиты от вторжений. Однако высокая стоимость лицензирования и обслуживания может стать препятствием для малых и средних предприятий.

Открытые решения для контроля доступа, такие как проекты на базе open-source технологий, предоставляют альтернативу коммерческим продуктам. Эти решения характеризуются высокой гибкостью и возможностью кастомизации под конкретные потребности организации. Например, популярные фреймворки аутентификации и авторизации, такие как Keycloak или Apache Syncope, позволяют создавать мощные системы управления доступом с минимальными затратами на приобретение лицензий. Однако внедрение и поддержка таких решений требуют значительных внутренних компетенций и ресурсов.

Особое внимание следует уделить специализированным решениям для определенных отраслей. В финансовой сфере, например, широко используются системы, обеспечивающие повышенную безопасность транзакций и строгий контроль доступа к конфиденциальным данным. Эти решения часто соответствуют жестким нормативным требованиям, таким как PCI DSS или SOX, что делает их более сложными в реализации, но и более надежными в использовании. В медицинской отрасли применяются системы, ориентированные на защиту персональных данных пациентов, соответствующие стандартам HIPAA или аналогичным регулятивным рамкам.

Важным аспектом является также развитие новых технологий, таких как блокчейн, который начинает находить применение в области управления доступом. Блокчейн-решения предлагают уникальные возможности для создания децентрализованных систем аутентификации, где пользователи имеют полный контроль над своими данными и могут предоставлять доступ только доверенным сторонам. Хотя эта технология пока находится на этапе активного исследования и тестирования, она уже демонстрирует большой потенциал для будущего развития систем контроля доступа.

При выборе решения для контроля доступа необходимо учитывать не только технические характеристики, но и организационные особенности. В крупных компаниях с распределенной структурой управления требуется высокая степень централизации и унификации процессов, тогда как малые предприятия могут предпочесть более простые и экономически выгодные варианты. Кроме того, важно оценить возможность интеграции нового решения с существующими системами и его соответствие нормативным требованиям.

Таким образом, анализ существующих решений для контроля доступа показывает, что современный рынок предлагает широкий выбор продуктов, различающихся по функциональности, стоимости и области применения. Выбор оптимального решения зависит от множества факторов, включая размер организации, специфику отрасли, бюджетные ограничения и требования законодательства. При этом важно помнить, что успешная реализация системы контроля доступа требует не только правильного выбора технологии, но и грамотного планирования, внедрения и постоянного совершенствования процессов управления безопасностью.

# **2.2 Определение требований к разрабатываемой системе контроля доступа.**

Определение требований к разрабатываемой системе контроля доступа является фундаментальным этапом в процессе проектирования и реализации подобных решений. Этот процесс требует тщательного анализа различных аспектов, связанных с функциональностью, безопасностью, масштабируемостью, производительностью и соответствие нормативным требованиям. Каждое из этих направлений играет ключевую роль в создании эффективной системы, способной удовлетворить потребности организации и обеспечить надежную защиту информационных ресурсов.

В первую очередь необходимо определить функциональные требования, которые описывают основные возможности системы. Это включает механизмы идентификации и аутентификации пользователей, методы авторизации, управление правами доступа и поддержку различных моделей контроля доступа, таких как дискреционный, мандатный, ролевой или атрибутивный. Система должна обеспечивать возможность создания гибких политик безопасности, позволяющих администраторам точно определять права каждого пользователя или группы пользователей относительно различных объектов системы. Также важно предусмотреть механизмы единого входа для упрощения работы с множеством приложений и интеграцию с существующими системами аутентификации.

Безопасность является одним из главных факторов, который необходимо учитывать при формировании требований. Система должна предоставлять надежную защиту от несанкционированного доступа, используя современные методы, такие как многофакторная аутентификация, шифрование данных и защита каналов передачи информации. Важно также предусмотреть механизмы обнаружения и предотвращения атак, такие как системы мониторинга активности пользователей, анализ поведения и автоматическое реагирование на подозрительные действия. Для обеспечения целостности данных необходимо внедрять проверочные процедуры, такие как цифровые подписи и хэширование.

Масштабируемость системы представляет собой еще один важный аспект, который необходимо учитывать на этапе определения требований. Система должна быть способна эффективно работать как в небольших организациях, так и в крупных корпорациях с тысячами пользователей. Это требует использования архитектурных решений, обеспечивающих горизонтальное и вертикальное масштабирование. Важно также предусмотреть возможность распределенного развертывания системы, чтобы обеспечить бесперебойную работу в условиях высоких нагрузок или отказа отдельных компонентов.

Производительность системы напрямую влияет на удобство использования и общее качество обслуживания. Требования к производительности должны учитывать время отклика, пропускную способность и количество одновременных соединений. Для достижения высокой производительности могут использоваться различные техники оптимизации, такие как кэширование данных, использование индексов, асинхронная обработка запросов и балансировка нагрузки. При этом важно найти оптимальный баланс между производительностью и уровнем безопасности, поскольку некоторые защитные механизмы могут увеличивать временные затраты на выполнение операций.

Соответствие нормативным требованиям является обязательным условием для многих организаций, особенно в регулируемых отраслях. Система должна соответствовать актуальным стандартам информационной безопасности, таким как ISO/IEC 27001, NIST, ГОСТ Р 57580 и GDPR, а также любым специфическим требованиям законодательства страны. Это включает обеспечение защиты персональных данных, соблюдение принципов минимальной достаточности и прозрачности, а также внедрение механизмов аудита и отчетности.

При определении требований важно также учитывать удобство использования системы как для конечных пользователей, так и для администраторов. Интерфейс должен быть интуитивно понятным и простым в освоении, предоставляя все необходимые функции через удобные средства управления. Для администраторов требуется наличие мощных инструментов для настройки политик безопасности, мониторинга событий и управления пользователями. Кроме того, система должна поддерживать детальную документацию и предоставление технической поддержки.

Таким образом, определение требований к разрабатываемой системе контроля доступа является сложным и многоаспектным процессом, который требует учета различных факторов, включая функциональность, безопасность, масштабируемость, производительность и соответствие нормативным требованиям. Грамотное формулирование этих требований залог успешной реализации системы, способной эффективно защищать информационные ресурсы и удовлетворять потребности организации.

# **2.3 Выбор технологий и инструментов для реализации системы (LDAP, Active Directory, Keycloak, FreeIPA).**

Выбор технологий и инструментов для реализации системы контроля доступа является ключевым этапом в процессе разработки подобных решений. Этот выбор определяет не только функциональные возможности системы, но и ее надежность, масштабируемость, удобство администрирования и соответствие нормативным требованиям. Рассмотрим основные технологии и инструменты, такие как LDAP, Active Directory, Keycloak и FreeIPA, которые могут быть использованы для создания эффективной системы управления доступом.

LDAP (Lightweight Directory Access Protocol) представляет собой протокол, предназначенный для доступа к каталогам данных, где хранятся информация о пользователях, группах, правах доступа и других объектах. Этот протокол широко используется в современных системах управления доступом благодаря своей универсальности и гибкости. LDAP позволяет создавать иерархические структуры данных, что делает его особенно подходящим для организации сложных схем управления правами. Однако сам по себе LDAP — это лишь протокол, и для его использования требуется дополнительная инфраструктура, такая как сервер каталогов. Хотя LDAP предоставляет базовый функционал для управления доступом, он может быть недостаточно мощным для решения комплексных задач безопасности без интеграции с другими технологиями.

Active Directory (AD) является одной из наиболее популярных платформ управления доступом, разработанной корпорацией Microsoft. Это полноценное решение, которое включает в себя не только сервер каталогов на основе LDAP, но и широкий спектр дополнительных функций, таких как управление групповыми политиками, аутентификация Kerberos, интеграция с операционными системами Windows и многое другое. Active Directory особенно эффективен в средах, где преобладает использование продуктов Microsoft, поскольку обеспечивает тесную интеграцию с существующими корпоративными системами. Однако его применение ограничивается преимущественно Windows-инфраструктурой, что может стать препятствием для организаций, использующих разнородные платформы.

Keycloak представляет собой современную open-source платформу для управления аутентификацией и авторизацией, которая предлагает широкий набор функций для реализации систем контроля доступа. Она поддерживает различные протоколы, включая OAuth 2.0, OpenID Connect и SAML, что делает ее совместимой с большинством современных приложений. Keycloak также предоставляет механизмы многофакторной аутентификации, единого входа и управления ролями пользователей. Особенностью этой платформы является высокая степень кастомизации, позволяющая адаптировать систему под специфические потребности организации. Благодаря своей открытости и активной поддержке сообщества, Keycloak становится все более популярным выбором для компаний, стремящихся создать гибкие и масштабируемые системы управления доступом.

FreeIPA является комплексным решением для управления удостоверениями, политиками доступа и аудитом в Unix-подобных операционных системах. Эта система сочетает в себе функциональность LDAP, Kerberos и DNS, предоставляя единый интерфейс для управления всеми аспектами безопасности. FreeIPA особенно привлекательна для организаций, использующих Linux-инфраструктуру, поскольку она предлагает простую интеграцию с существующими системами и автоматизацию многих административных задач. Платформа поддерживает многофакторную аутентификацию, политики паролей и детальный аудит событий, что делает ее подходящей для сред с повышенными требованиями к безопасности.

При выборе технологии важно учитывать специфику инфраструктуры организации, требования к безопасности, уровень технической экспертизы команды и бюджетные ограничения. Например, для компаний, работающих в экосистеме Microsoft, Active Directory может быть наиболее оптимальным выбором благодаря своей интеграции с существующими системами. В то же время, организации, использующие разнородные платформы или предпочитающие открытые решения, могут обратить внимание на Keycloak или FreeIPA. LDAP остается важным компонентом многих систем управления доступом, но обычно требует дополнительного программирования для реализации полного функционала.

Таким образом, выбор технологий и инструментов для реализации системы контроля доступа должен основываться на всестороннем анализе потребностей организации и возможностей различных решений. Каждый из рассмотренных инструментов имеет свои сильные стороны и ограничения, и успешная реализация зависит от правильного сочетания этих факторов. При этом важно помнить, что выбор технологии — это лишь один из этапов создания эффективной системы управления доступом, который должен сопровождаться грамотным проектированием, внедрением и постоянным совершенствованием процессов безопасности.

# **2.4 Разработка архитектуры системы контроля доступа и модели разграничения прав пользователей**

Разработка архитектуры системы контроля доступа и модели разграничения прав пользователей представляет собой комплексный процесс, который требует детального анализа различных аспектов безопасности, функциональности и взаимодействия компонентов. Архитектура системы должна быть спроектирована таким образом, чтобы обеспечивать надежную защиту информации, гибкость в управлении правами доступа и возможность масштабирования для удовлетворения будущих потребностей организации.

Архитектура системы контроля доступа строится на основе нескольких ключевых принципов. В первую очередь, система должна иметь четко определенные границы ответственности между различными компонентами, что позволяет избежать путаницы в управлении и упрощает поддержку. Основные элементы архитектуры включают модуль аутентификации, отвечающий за проверку личности пользователя; модуль авторизации, обеспечивающий контроль над действиями, которые пользователь может выполнять в системе; и модуль управления учетными записями, позволяющий администраторам создавать, изменять и удалять пользователей, а также назначать им права доступа.

Модуль аутентификации играет важнейшую роль в обеспечении безопасности системы. Для этого он должен использовать современные методы, такие как многофакторная аутентификация, которая сочетает несколько независимых факторов проверки подлинности, например, пароль, одноразовый код или биометрические данные. Кроме того, этот модуль должен поддерживать различные протоколы аутентификации, такие как OAuth 2.0 или SAML, что делает его совместимым с большинством современных приложений и платформ.

Модуль авторизации реализует механизм разграничения прав доступа, определяя, какие действия могут выполняться каждым пользователем или группой пользователей. Для этой цели часто используется ролевая модель управления доступом (RBAC), которая основывается на присвоении пользователям ролей, каждая из которых имеет набор связанных с ней прав. Такой подход значительно упрощает администрирование системы, так как вместо индивидуальной настройки прав для каждого пользователя можно работать с предопределенными наборами разрешений. Однако для более сложных случаев может применяться атрибутивная модель управления доступом (ABAC), которая позволяет динамически определять права доступа на основе различных параметров, таких как время дня, местоположение или устройство, с которого производится запрос.

Управление учетными записями пользователей осуществляется через специальный модуль, который обеспечивает централизованное хранение данных о пользователях и их правах. Этот модуль должен поддерживать механизмы автоматического создания учетных записей для новых сотрудников, а также своевременного удаления учетных записей при прекращении трудовых отношений. Для повышения удобства использования системы можно внедрить функцию самообслуживания, позволяющую пользователям самостоятельно изменять свои профили, восстанавливать пароли или запрашивать изменения в своих правах.

Взаимодействие между компонентами системы должно быть построено на основе стандартных протоколов и интерфейсов, что обеспечивает возможность интеграции с другими корпоративными системами. Это особенно важно для крупных организаций, где существует множество приложений, требующих единого подхода к управлению доступом. Для этих целей могут использоваться технологии, такие как RESTful API или GraphQL, которые позволяют легко обмениваться данными между различными сервисами.

Модель разграничения прав пользователей должна быть разработана с учетом специфики бизнес-процессов организации. Она должна обеспечивать четкое разделение обязанностей между пользователями разных уровней, предотвращая конфликты интересов и минимизируя риск несанкционированного доступа. При этом важно предусмотреть механизмы делегирования прав, позволяющие временно передавать полномочия другим пользователям в случае необходимости. Также необходимо реализовать детальную систему аудита, которая будет фиксировать все операции, связанные с управлением доступом, для последующего анализа и выявления потенциальных угроз.

При проектировании архитектуры системы важно учитывать требования к безопасности, такие как шифрование данных при передаче и в состоянии покоя, защита от SQL-инъекций и других типов атак, а также регулярное обновление программного обеспечения для устранения известных уязвимостей. Кроме того, система должна быть способна эффективно работать в условиях высоких нагрузок, поэтому необходимо предусмотреть механизмы горизонтального и вертикального масштабирования.

Таким образом, разработка архитектуры системы контроля доступа и модели разграничения прав пользователей является сложным и многоступенчатым процессом, требующим тщательного планирования и анализа. Успешная реализация зависит от правильного выбора технологий, грамотного проектирования структуры системы и внимательного учета всех аспектов безопасности и функциональности. Только при соблюдении этих условий можно создать надежную и эффективную систему управления доступом, способную удовлетворить потребности современной организации.

# **2.5 Анализ угроз и рисков при внедрении системы контроля доступа.**

Анализ угроз и рисков при внедрении системы контроля доступа является критически важным этапом, который позволяет оценить потенциальные проблемы, с которыми может столкнуться организация во время реализации и эксплуатации подобной системы. Этот процесс требует детального изучения различных факторов, влияющих на безопасность, функциональность и надежность решения. Анализ помогает выявить возможные уязвимости, спрогнозировать последствия их эксплуатации злоумышленниками и разработать эффективные меры защиты.

Одной из основных угроз при внедрении системы контроля доступа является возможность компрометации учетных данных пользователей. Это может происходить через фишинговые атаки, использование слабых паролей или эксплуатацию уязвимостей в системе аутентификации. Если злоумышленник получит доступ к учетным данным администратора или других привилегированных пользователей, это может привести к серьезным последствиям, таким как несанкционированное изменение прав доступа, утечка конфиденциальной информации или полный отказ системы от работы. Для минимизации этого риска необходимо использовать многофакторную аутентификацию, регулярно обновлять пароли и проводить обучение сотрудников для повышения осведомленности о методах социальной инженерии.

Другой значительной угрозой является наличие уязвимостей в программном обеспечении, используемом для реализации системы управления доступом. Это могут быть ошибки в коде, неправильно настроенные параметры безопасности или недостаточная защита каналов передачи данных. Злоумышленники могут воспользоваться этими уязвимостями для получения доступа к системе или перехвата чувствительной информации. Чтобы противостоять этому риску, важно регулярно обновлять программное обеспечение до последних версий, применять патчи безопасности и проводить тестирование на проникновение для выявления потенциальных уязвимостей.

Существует также риск ошибок в настройке системы управления доступом, которые могут привести к непреднамеренному расширению прав доступа или ограничению законных действий пользователей. Такие ошибки могут возникать из-за недостаточной квалификации администраторов, сложности конфигурации системы или использования некорректных политик безопасности. Для предотвращения этих проблем необходимо разрабатывать подробные руководства по настройке и эксплуатации системы, предоставлять администраторам соответствующее обучение и внедрять механизмы автоматического контроля за корректностью настроек.

Еще одной потенциальной угрозой является отказ в работе ключевых компонентов системы управления доступом. Это может произойти из-за аппаратных сбоев, перегрузки системы или атак типа "отказ в обслуживании" (DDoS). В случае выхода из строя основного сервера аутентификации или базы данных с информацией о пользователях, работа всей системы может быть временно приостановлена. Для снижения этого риска необходимо внедрять резервные копии данных, организовывать дублирование критических компонентов и разрабатывать планы аварийного восстановления.

Также следует учитывать угрозы, связанные с внутренними злоумышленниками — сотрудниками организации, которые могут намеренно или случайно нарушить правила безопасности. Эти действия могут включать попытки получить доступ к чужим данным, изменение прав доступа без надлежащего разрешения или утечку информации. Для борьбы с такими рисками необходимо внедрять строгую политику безопасности, контролировать действия пользователей с помощью систем мониторинга и аудита, а также регулярно проверять соблюдение правил доступа.

При анализе рисков важно учитывать специфику организации, включая ее размер, структуру, типы обрабатываемых данных и нормативные требования. Например, компании из финансового сектора или здравоохранения сталкиваются с более жесткими требованиями к защите данных, что увеличивает необходимость в комплексных мерах безопасности. В то же время, малые предприятия могут иметь ограниченные ресурсы для реализации всех рекомендованных мер, поэтому им нужно находить оптимальный баланс между уровнем защиты и затратами.

Таким образом, анализ угроз и рисков при внедрении системы контроля доступа является многоаспектным процессом, требующим внимательного изучения различных факторов, влияющих на безопасность и надежность решения. Успешное проведение анализа позволяет выявить потенциальные проблемы, разработать эффективные меры защиты и обеспечить стабильную работу системы в условиях современных угроз. Только при тщательной подготовке и постоянном совершенствовании мер безопасности можно создать надежную систему управления доступом, способную удовлетворить потребности организации и обеспечить защиту информационных ресурсов.

# **Практический раздел**

# **3.1 Разработка и настройка системы контроля доступа в информационной системе.**

Разработка и настройка системы контроля доступа в информационной системе с использованием Django представляет собой сложный, но увлекательный процесс, который требует внимательного подхода к проектированию архитектуры, реализации функционала и тестированию готового решения. Django, как современный веб-фреймворк, предоставляет широкие возможности для создания надежных и масштабируемых систем управления доступом, что делает его отличным выбором для реализации подобных задач.

Первым шагом в разработке системы является проектирование структуры базы данных, которая будет хранить информацию о пользователях, их ролях, правах доступа и других связанных объектах. Django предоставляет удобные инструменты для работы с моделями через ORM (Object-Relational Mapping), что позволяет быстро создавать и настраивать таблицы базы данных. В контексте системы контроля доступа важно определить модели для пользователей, групп пользователей и разрешений. Например, можно использовать встроенную модель User Django, расширив ее дополнительными полями для хранения специфической информации, такой как должность или отдел. Также необходимо создать модели для ролей и прав доступа, которые будут определять, какие действия могут выполнять пользователи в зависимости от их роли.

Далее следует реализация механизма аутентификации, который является основой любой системы контроля доступа. Django предоставляет встроенные инструменты для аутентификации пользователей, включая формы входа, регистрации и восстановления паролей. Однако для повышения безопасности рекомендуется добавить многофакторную аутентификацию, например, через использование одноразовых кодов, отправляемых по SMS или через мобильное приложение. Это можно реализовать с помощью сторонних библиотек, таких как django-two-factor-auth, которые легко интегрируются в проект. Также важно настроить защиту от перехвата учетных данных, используя шифрование каналов передачи данных через HTTPS и защищая сессии пользователей.

Следующим этапом является реализация механизма авторизации, который определяет, какие действия могут выполнять пользователи после успешной аутентификации. Django предоставляет встроенную систему разрешений, которая позволяет назначать права на уровне моделей. Например, можно определить, что только администраторы могут создавать или удалять записи в определенной таблице базы данных. Для более гибкого управления доступом можно использовать декораторы или миксины, такие как @login\_required или PermissionRequiredMixin, которые позволяют ограничивать доступ к определенным представлениям или API-эндпоинтам. Если требуется реализовать более сложную логику, например, атрибутивный контроль доступа, можно создать собственные механизмы проверки прав, основанные на дополнительных параметрах, таких как время дня или местоположение пользователя.

Особое внимание следует уделить настройке ролевой модели управления доступом (RBAC), которая является одной из наиболее распространенных и удобных для реализации. В Django это можно сделать, используя встроенную модель Group для создания ролей и связывания их с разрешениями. Например, можно создать роли "администратор", "менеджер" и "пользователь", каждая из которых будет иметь свой набор прав. Для удобства администрирования можно разработать интерфейс администратора, который позволит легко назначать роли пользователям и управлять их правами. Это можно реализовать через встроенную административную панель Django, расширив ее дополнительными функциями.

Важным аспектом разработки является также реализация журналирования и аудита событий, связанных с управлением доступом. Django предоставляет встроенные инструменты для логирования, которые можно настроить для записи всех значимых действий, таких как попытки входа, изменения прав доступа или выполнение критических операций. Эти данные можно хранить в базе данных или отправлять в отдельную систему мониторинга, чтобы обеспечить возможность анализа и выявления подозрительной активности. Для повышения эффективности аудита можно использовать сторонние библиотеки, такие как django-auditlog, которые автоматически фиксируют изменения в моделях.

При разработке системы важно учитывать требования к производительности и масштабируемости. Django предоставляет инструменты для оптимизации запросов к базе данных, такие как select\_related и prefetch\_related, которые позволяют минимизировать количество обращений к серверу базы данных. Также можно использовать кэширование через Redis или Memcached для ускорения работы часто запрашиваемых данных. Для обеспечения высокой доступности системы можно настроить балансировку нагрузки и репликацию базы данных, что особенно важно для крупных проектов с большим количеством пользователей.

Тестирование системы является неотъемлемой частью разработки. Django предоставляет встроенные инструменты для написания и выполнения тестов, что позволяет проверять корректность работы всех компонентов системы. Важно протестировать как функциональные аспекты, такие как аутентификация и авторизация, так и безопасность, например, защиту от SQL-инъекций или XSS-атак. Также рекомендуется проводить нагрузочное тестирование, чтобы убедиться, что система способна работать под высокой нагрузкой без снижения производительности.

Таким образом, разработка и настройка системы контроля доступа в информационной системе с использованием Django представляет собой комплексный процесс, требующий внимательного подхода к проектированию, реализации и тестированию. Благодаря мощным инструментам и гибкости фреймворка, можно создать надежную и масштабируемую систему, которая будет соответствовать требованиям безопасности и удовлетворять потребности организации. При этом важно постоянно совершенствовать решение, адаптируя его к изменяющимся условиям и новым угрозам, чтобы обеспечить долгосрочную защиту информационных ресурсов.

# **3.2 Реализация механизма аутентификации и авторизации пользователей.**

Реализация механизма аутентификации и авторизации пользователей в информационной системе была выполнена с использованием Django, что позволило создать надежное и функциональное решение для управления доступом. В процессе разработки были реализованы ключевые функции, такие как вход в систему, регистрация новых пользователей, редактирование профиля и изменение пароля, при этом особое внимание уделялось безопасности, удобству использования и логированию событий.

Механизм аутентификации был реализован через стандартные инструменты Django, такие как функция authenticate, которая проверяет учетные данные пользователя, и функция login, обеспечивающая создание сессии после успешной аутентификации. Для входа в систему была создана функция login\_user, которая обрабатывает POST-запросы от пользователей. При получении данных формы система проверяет корректность введенного имени пользователя и пароля. Если аутентификация проходит успешно, пользователь получает доступ к системе, и его действия регистрируются в журнале с указанием IP-адреса. В случае неудачной попытки входа также фиксируется соответствующее событие, что позволяет отслеживать подозрительную активность. Для получения IP-адреса пользователя была реализована вспомогательная функция get\_client\_ip, которая анализирует заголовки запроса и определяет реальный IP-адрес, даже если запрос проходит через прокси-сервер.

Функция выхода из системы (logout\_user) завершает текущую сессию пользователя и регистрирует это событие в журнале. Это обеспечивает дополнительный уровень контроля за действиями пользователей и позволяет отслеживать их активность в системе. После выхода пользователь перенаправляется на главную страницу, где отображается соответствующее уведомление.

Для регистрации новых пользователей была реализована функция register\_user, которая использует кастомную форму SignUpForm. Эта форма позволяет собирать необходимые данные о пользователе, такие как имя пользователя и пароль, и сохранять их в базе данных. После успешной регистрации система автоматически аутентифицирует нового пользователя и предоставляет ему доступ к системе. Все события регистрации также регистрируются в журнале с указанием IP-адреса, что позволяет отслеживать, когда и откуда был создан новый аккаунт.

Управление профилем пользователя реализовано через функцию edit\_profile, которая позволяет изменять информацию о пользователе, такую как имя, фамилия или адрес электронной почты. Для этого используется кастомная форма EditProfileForm, которая связана с моделью пользователя. После успешного редактирования профиля изменения сохраняются в базе данных, а событие регистрируется в журнале. Это позволяет администраторам отслеживать, какие изменения были внесены в профиль пользователя и когда.

Изменение пароля реализовано через функцию change\_password, которая использует кастомную форму CustomPasswordChangeForm. Эта форма проверяет корректность старого пароля и соответствие новых паролей заданным требованиям. После успешного изменения пароля система обновляет хэш сессии, чтобы избежать разлогинивания пользователя. Событие изменения пароля также регистрируется в журнале, что позволяет контролировать, когда пользователь обновил свои учетные данные.

В процессе реализации особое внимание уделялось логированию событий, что является важным аспектом обеспечения безопасности. Для этого была настроена система логирования, которая записывает все значимые действия пользователей, такие как вход в систему, регистрация, редактирование профиля и изменение пароля. Логи содержат информацию о времени события, имени пользователя и его IP-адресе, что позволяет проводить детальный анализ активности и выявлять потенциальные угрозы. Использование библиотеки logging позволило организовать удобное и гибкое управление журналами.

Таким образом, реализация механизма аутентификации и авторизации пользователей была выполнена с учетом современных требований безопасности и удобства использования. Благодаря использованию стандартных инструментов Django и дополнительных функций, таких как логирование и проверка IP-адресов, удалось создать надежную систему, способную эффективно управлять доступом пользователей и защищать информационные ресурсы. Этот подход обеспечивает высокий уровень защиты и удобства для пользователей, а также предоставляет администраторам инструменты для мониторинга и анализа активности в системе.

# **3.3 Настройка ролевой модели доступа и политик безопасности.**

Настройка ролевой модели доступа и политик безопасности является ключевым этапом в реализации системы контроля доступа, обеспечивающим гибкое и надежное управление правами пользователей. В рамках проекта была внедрена ролевая модель управления доступом (RBAC), которая позволяет определять права пользователей на основе их ролей, а также настраивать политики безопасности для различных объектов системы. Это решение обеспечивает четкое разделение обязанностей, минимизирует риски несанкционированного доступа и упрощает администрирование.

Ролевая модель доступа была реализована с использованием встроенных возможностей Django, таких как система разрешений (permissions) и группы (groups). Каждая роль в системе представляет собой набор прав, которые определяют, какие действия может выполнять пользователь. Например, были созданы роли "Администратор", "Менеджер" и "Пользователь", каждая из которых имеет свой уникальный набор разрешений. Роль "Администратор" предоставляет полный доступ ко всем функциям системы, включая управление пользователями и настройку политик безопасности. Роль "Менеджер" позволяет выполнять операции, связанные с управлением контентом или данными, но ограничивает доступ к настройкам системы. Роль "Пользователь" предоставляет базовый уровень доступа, достаточный для выполнения повседневных задач, таких как просмотр информации или отправка запросов.

Для настройки ролей использовались группы Django, которые позволяют объединять пользователей с одинаковыми правами доступа. Каждой группе были назначены соответствующие разрешения через административную панель Django. Например, для роли "Менеджер" были добавлены разрешения на создание, редактирование и удаление определенных объектов, таких как записи в базе данных или документы. Это позволило централизованно управлять правами доступа, что значительно упростило процесс администрирования. При необходимости можно быстро изменить права для всей группы, что особенно важно в условиях динамичного развития системы.

Политики безопасности были настроены для обеспечения дополнительного уровня защиты данных и предотвращения несанкционированных действий. Одной из основных политик является проверка прав доступа на уровне представлений (views). Для этого использовались декораторы, такие как @login\_required и @permission\_required, которые ограничивают доступ к определенным страницам или API-эндпоинтам только авторизованным пользователям или пользователям с конкретными разрешениями. Например, доступ к странице редактирования профиля ограничен только теми пользователями, которые имеют соответствующее разрешение. Если пользователь пытается получить доступ к защищенному ресурсу без необходимых прав, система автоматически перенаправляет его на страницу ошибки или отображает сообщение об отказе в доступе.

Дополнительно были реализованы механизмы проверки прав доступа на уровне бизнес-логики. Это особенно важно для сложных операций, которые требуют более детального контроля. Например, при выполнении операции изменения статуса документа система проверяет, принадлежит ли этот документ текущему пользователю или находится ли он в группе, имеющей право на выполнение данной операции. Такой подход позволяет избежать ситуаций, когда пользователь может получить доступ к данным, которые ему не предназначены, даже если он случайно или намеренно попытается обойти стандартные ограничения.

Для повышения безопасности были внедрены дополнительные меры, такие как защита от повторного использования старых паролей и установка минимальных требований к сложности паролей. Эти политики были реализованы через кастомные формы и валидаторы, которые проверяют соответствие паролей заданным критериям. Например, система требует, чтобы пароль содержал как минимум одну заглавную букву, одну цифру и один специальный символ. Также было настроено ограничение на количество неудачных попыток входа, после которого учетная запись временно блокируется. Это помогает предотвратить атаки методом подбора паролей.

Особое внимание уделялось логированию событий, связанных с управлением доступом. Все действия, такие как вход в систему, изменение ролей или выполнение критических операций, регистрируются в журнале с указанием времени, имени пользователя и IP-адреса. Это позволяет отслеживать активность пользователей и выявлять потенциальные угрозы. Например, если администратор заметит большое количество неудачных попыток входа с одного IP-адреса, это может свидетельствовать о попытке атаки, и система может автоматически заблокировать этот адрес.

Также была настроена политика управления сессиями, которая обеспечивает защиту от перехвата сессий злоумышленниками. Для этого использовались механизмы, такие как шифрование сессий и автоматическое завершение сессии после определенного периода неактивности. Кроме того, система обновляет хэш сессии после изменения пароля, что предотвращает использование старых сессий для несанкционированного доступа.

Важным аспектом настройки ролевой модели и политик безопасности является их тестирование. Были проведены тесты для проверки корректности работы механизмов аутентификации, авторизации и управления доступом. Тестирование включало как проверку стандартных сценариев, так и попытки обхода ограничений. Например, проверялось, сможет ли пользователь с ролью "Пользователь" получить доступ к функциям, предназначенным для роли "Администратор". Все тесты показали, что система корректно обрабатывает запросы и предотвращает несанкционированные действия.

Таким образом, настройка ролевой модели доступа и политик безопасности позволила создать надежную и гибкую систему управления правами пользователей. Использование встроенных инструментов Django и дополнительных механизмов защиты обеспечило высокий уровень безопасности и удобства администрирования. Этот подход позволяет эффективно управлять доступом, минимизировать риски и обеспечивать соответствие требованиям информационной безопасности.

# **3.4 Интеграция системы контроля доступа с другими сервисами (база данных, веб-приложения, облачные сервисы).**

Интеграция системы контроля доступа с другими сервисами, такими как базы данных, веб-приложения и облачные платформы, является важнейшим этапом разработки, обеспечивающим согласованное взаимодействие всех компонентов информационной системы. Этот процесс требует тщательного проектирования и реализации, чтобы обеспечить надежность, безопасность и удобство использования. В рамках проекта была выполнена интеграция системы контроля доступа с различными сервисами, что позволило создать единое решение для управления правами пользователей и защитой данных.

Одним из ключевых аспектов интеграции стала работа с базой данных, которая является основой для хранения информации о пользователях, их ролях и разрешениях. Django предоставляет мощные инструменты для работы с базами данных через ORM (Object-Relational Mapping), что значительно упростило процесс настройки и взаимодействия. Были созданы модели для пользователей, групп и разрешений, которые автоматически преобразуются в таблицы базы данных. Это позволило централизованно управлять данными о пользователях и их правах доступа. Для повышения производительности были оптимизированы запросы к базе данных с использованием методов select\_related и prefetch\_related, что минимизирует количество обращений к серверу базы данных. Также была настроена репликация базы данных и балансировка нагрузки для обеспечения высокой доступности системы.

Интеграция с веб-приложениями была реализована через стандартные механизмы Django, такие как представления (views) и маршрутизация (URL routing). Система контроля доступа была тесно связана с веб-интерфейсом, что позволило ограничивать доступ к определенным страницам или функциям в зависимости от ролей пользователей. Например, для защиты конфиденциальных страниц использовались декораторы, такие как @login\_required и @permission\_required, которые проверяют права пользователя перед предоставлением доступа. Для API-эндпоинтов была применена аналогичная логика с использованием классов миксинов, таких как PermissionRequiredMixin, что обеспечивает гибкость и безопасность при взаимодействии с внешними системами.

Взаимодействие с облачными сервисами стало еще одним важным направлением интеграции. В современных условиях многие организации используют облачные платформы для хранения данных, развертывания приложений и управления инфраструктурой. Для интеграции с облачными сервисами были использованы API-интерфейсы, предоставляемые провайдерами, такими как AWS, Google Cloud или Microsoft Azure. Например, для аутентификации пользователей через социальные сети или корпоративные учетные записи была реализована поддержка протоколов OAuth 2.0 и OpenID Connect. Это позволило пользователям входить в систему через существующие учетные записи, такие как Google или Facebook, что упрощает процесс аутентификации и повышает удобство использования. Также была настроена интеграция с облачными хранилищами данных, что позволяет безопасно хранить файлы и документы, доступ к которым контролируется через систему разграничения прав.

Для обеспечения безопасности при взаимодействии с внешними сервисами были внедрены механизмы шифрования данных. Все коммуникации между системой и внешними сервисами осуществляются через защищенные каналы связи с использованием протокола HTTPS. Дополнительно были настроены механизмы проверки подлинности запросов, такие как использование токенов доступа и цифровых подписей. Это предотвращает несанкционированный доступ к данным и защищает от атак типа "перехват трафика".

Особое внимание уделялось логированию событий, связанных с взаимодействием системы с внешними сервисами. Все запросы и ответы регистрируются в журнале с указанием времени, IP-адреса и других параметров. Это позволяет отслеживать активность пользователей и выявлять потенциальные угрозы. Например, если система обнаруживает большое количество запросов к облачному хранилищу с одного IP-адреса, это может свидетельствовать о попытке атаки, и система автоматически блокирует этот адрес.

Также была реализована возможность централизованного управления доступом через единую точку входа (SSO). Это особенно важно для организаций, использующих множество приложений и сервисов. Пользователи могут аутентифицироваться один раз и получать доступ ко всем связанным системам без необходимости повторного ввода учетных данных. Для этого была настроена интеграция с SSO-провайдерами, такими как Keycloak или Auth0, что обеспечивает удобство использования и высокий уровень безопасности.

При интеграции системы контроля доступа с другими сервисами важно учитывать требования масштабируемости и отказоустойчивости. Для этого были внедрены механизмы горизонтального масштабирования, такие как балансировка нагрузки и резервирование критических компонентов. Например, сервер аутентификации был настроен на работу в кластере, что позволяет системе продолжать функционировать даже при отказе одного из узлов. Также была настроена автоматическая репликация данных между серверами, что минимизирует риск потери информации.

Таким образом, интеграция системы контроля доступа с базами данных, веб-приложениями и облачными сервисами позволила создать единое и надежное решение для управления правами пользователей. Использование современных технологий и стандартов обеспечило высокий уровень безопасности, производительности и удобства использования. Этот подход позволяет эффективно взаимодействовать с различными компонентами информационной системы и адаптироваться к изменяющимся требованиям бизнеса.

# **3.5 Проведение тестирования системы на безопасность, отказоустойчивость и производительность.**

Проведение тестирования системы на безопасность, отказоустойчивость и производительность является важнейшим этапом разработки, который позволяет убедиться в надежности, стабильности и защищенности решения. Этот процесс включает комплексный анализ различных аспектов работы системы с целью выявления потенциальных уязвимостей, багов или проблем, которые могут повлиять на ее функциональность или безопасность. В рамках проекта было проведено детальное тестирование, охватывающее все ключевые области.

Тестирование безопасности стало одним из приоритетных направлений, так как защита данных и предотвращение несанкционированного доступа являются критически важными для любой информационной системы. Для этого были применены различные методы проверки, включая анализ кода на наличие уязвимостей, тестирование на проникновение (penetration testing) и проверку конфигурации системы. Особое внимание уделялось защите от распространенных атак, таких как SQL-инъекции, межсайтовый скриптинг (XSS) и подделка межсайтовых запросов (CSRF). Django предоставляет встроенные механизмы защиты от этих угроз, такие как экранирование входных данных, использование CSRF-токенов и защита сессий, однако дополнительные проверки позволили убедиться в их корректной работе. Также были протестированы механизмы аутентификации и авторизации, чтобы убедиться, что пользователи не могут получить доступ к ресурсам, для которых у них нет прав. Например, проверялось, сможет ли пользователь с ролью "Пользователь" выполнить операции, предназначенные для роли "Администратор", путем изменения URL или отправки неправильных запросов. Все тесты показали, что система корректно обрабатывает такие попытки и предотвращает несанкционированный доступ.

Отказоустойчивость системы также была тщательно проверена, чтобы обеспечить бесперебойную работу даже в условиях высоких нагрузок или сбоев оборудования. Для этого были смоделированы различные сценарии, такие как отказ сервера базы данных, перегрузка сети или аварийное завершение работы одного из компонентов системы. Была настроена репликация базы данных и балансировка нагрузки, что позволило системе продолжать функционировать даже при отказе одного из узлов. Также была проверена работа механизмов автоматического восстановления, таких как повторное подключение к базе данных или перенаправление запросов на резервные серверы. Тестирование показало, что система способна быстро восстанавливаться после сбоев и минимизировать время простоя.

Производительность системы была протестирована с использованием инструментов нагрузочного тестирования, таких как Apache JMeter и Locust. Эти инструменты позволяют моделировать большое количество одновременных пользователей и оценивать, как система справляется с высокой нагрузкой. Были протестированы ключевые операции, такие как вход в систему, выполнение запросов к базе данных и работа с API-эндпоинтами. Особое внимание уделялось времени отклика системы и пропускной способности. Для оптимизации производительности были внедрены механизмы кэширования через Redis, что позволило значительно сократить время выполнения часто запрашиваемых операций. Также были оптимизированы запросы к базе данных с использованием методов select\_related и prefetch\_related, что минимизировало количество обращений к серверу базы данных. Результаты тестирования показали, что система способна обрабатывать большое количество запросов без значительного снижения производительности.

Дополнительно были проведены тесты на совместимость с различными браузерами и устройствами, чтобы убедиться, что система работает корректно на всех платформах. Это особенно важно для веб-приложений, которые могут использоваться на устройствах с разными операционными системами и разрешениями экрана. Также была проверена работа системы в условиях ограниченного интернет-соединения, чтобы убедиться, что она остается отзывчивой даже при медленной передаче данных.

Важным аспектом тестирования стала проверка журналирования событий. Все действия пользователей, такие как вход в систему, изменение ролей или выполнение критических операций, регистрируются в журнале с указанием времени, IP-адреса и других параметров. Это позволяет отслеживать активность пользователей и выявлять потенциальные угрозы. Например, если система обнаруживает большое количество неудачных попыток входа с одного IP-адреса, это может свидетельствовать о попытке атаки, и система автоматически блокирует этот адрес. Тестирование показало, что журналы создаются корректно и содержат всю необходимую информацию для анализа.

Таким образом, проведение тестирования системы на безопасность, отказоустойчивость и производительность позволило выявить и устранить потенциальные проблемы, обеспечив высокий уровень надежности и защищенности решения. Использование современных инструментов и методов тестирования позволило убедиться, что система способна эффективно работать в различных условиях и противостоять современным угрозам. Этот подход гарантирует, что система будет соответствовать требованиям бизнеса и обеспечивать бесперебойную работу для всех пользователей.

# **3.6 Разработка эксплуатационной документации по настройке и сопровождению системы контроля доступа.**

Разработка эксплуатационной документации по настройке и сопровождению системы контроля доступа является важным этапом, который обеспечивает удобство использования, поддержки и масштабирования решения. Эта документация служит руководством для администраторов, разработчиков и других специалистов, участвующих в эксплуатации системы, и позволяет минимизировать риски ошибок при настройке или обслуживании. В рамках проекта была создана подробная документация, охватывающая все ключевые аспекты работы с системой.

Эксплуатационная документация начинается с общего описания системы, включая ее архитектуру, основные компоненты и принципы работы. Это позволяет пользователям понять, как устроена система и какие задачи она решает. Особое внимание уделяется роли каждого компонента, например, сервера аутентификации, базы данных, веб-приложений и облачных сервисов. Также описываются зависимости между компонентами и их взаимодействие, что помогает администраторам быстро выявлять и устранять проблемы.

Следующий раздел документации посвящен процессу установки и первичной настройки системы. Здесь подробно описываются шаги, необходимые для развертывания системы на сервере, включая установку зависимостей, настройку базы данных и конфигурацию веб-сервера. Например, приводятся команды для установки Python, Django и других необходимых библиотек, а также инструкции по миграции базы данных и созданию суперпользователя. Для удобства пользователей предоставляются примеры конфигурационных файлов, таких как settings.py, где указываются параметры подключения к базе данных, секретные ключи и настройки безопасности. Также описываются рекомендации по настройке HTTPS и защиты каналов передачи данных.

Отдельный раздел документации посвящен управлению пользователями, ролями и разрешениями. Здесь подробно объясняется, как создавать новых пользователей, назначать им роли и управлять их правами доступа. Описываются способы использования административной панели Django для управления группами и разрешениями, а также приводятся примеры SQL-запросов для выполнения операций с базой данных. Также документируется процесс настройки ролевой модели доступа (RBAC), включая создание новых ролей и назначение им прав. Это позволяет администраторам гибко настраивать систему под потребности организации.

Важным аспектом документации является описание процедур мониторинга и аудита. Здесь приводятся инструкции по настройке логирования событий, включая выбор формата журналов, их хранение и анализ. Описываются типы событий, которые регистрируются в журналах, такие как вход в систему, изменение ролей или выполнение критических операций. Также предоставляются рекомендации по использованию инструментов мониторинга, таких как Prometheus или Grafana, для отслеживания производительности системы и выявления аномалий. Документируются шаги по настройке оповещений, чтобы администраторы могли своевременно реагировать на инциденты.

Документация включает раздел, посвященный обновлению и обслуживанию системы. Здесь описываются процедуры для безопасного обновления программного обеспечения, включая создание резервных копий базы данных и проверку совместимости новых версий с существующей инфраструктурой. Приводятся инструкции по применению патчей безопасности и обновлению зависимостей. Также описываются регламентные работы, такие как очистка старых журналов, оптимизация базы данных и проверка работоспособности всех компонентов системы.

Отдельное внимание уделяется разделу по устранению неполадок. Здесь приводятся типичные проблемы, с которыми могут столкнуться пользователи, и способы их решения. Например, описываются действия при отказе сервера базы данных, проблемах с аутентификацией или снижении производительности. Для каждой проблемы приводятся возможные причины и шаги по их устранению. Также предоставляются контакты технической поддержки и ссылки на дополнительные ресурсы, такие как официальная документация Django или форумы разработчиков.

Для удобства пользователей документация структурирована и содержит навигацию, позволяющую быстро находить нужную информацию. Используются четкие заголовки, списки и примеры кода, чтобы сделать материал более доступным. Также предоставляются визуальные материалы, такие как схемы архитектуры системы и скриншоты интерфейса административной панели, что помогает лучше понять описанные процессы.

Таким образом, разработка эксплуатационной документации по настройке и сопровождению системы контроля доступа позволила создать полное и удобное руководство для всех участников процесса эксплуатации. Эта документация не только помогает минимизировать риски ошибок, но и обеспечивает возможность быстрого обучения новых сотрудников. Подробное описание всех аспектов работы системы гарантирует, что она будет надежно поддерживаться и эффективно использоваться в течение всего жизненного цикла.

# **ЗАКЛЮЧЕНИЕ**

Заключение представляет собой важный этап работы, в котором подводятся итоги проведенного исследования и разработки системы контроля доступа. В рамках данной курсовой работы была успешно реализована система, обеспечивающая надежное управление доступом к информационным ресурсам с использованием современных технологий и подходов. Проект охватил все ключевые аспекты создания такой системы: от выбора архитектуры и модели разграничения прав до тестирования и документирования решения.

В процессе работы были решены задачи, связанные с анализом существующих методов и технологий управления доступом, что позволило выбрать оптимальные инструменты для реализации проекта. Использование Django как основной платформы разработки обеспечило высокую производительность, безопасность и удобство администрирования системы. Были внедрены механизмы аутентификации и авторизации, включая многофакторную аутентификацию и ролевую модель управления доступом, что значительно повысило уровень защиты данных. Также была настроена интеграция с внешними сервисами, такими как базы данных, веб-приложения и облачные платформы, что позволило создать единое решение для управления правами пользователей.

Тестирование системы показало ее высокую отказоустойчивость, производительность и защищенность от современных угроз. Были выявлены и устранены потенциальные уязвимости, а также оптимизированы процессы обработки запросов для минимизации времени отклика. Особое внимание уделялось логированию событий, что позволяет администраторам отслеживать активность пользователей и оперативно реагировать на инциденты. Разработанная эксплуатационная документация предоставляет четкие инструкции по настройке, обслуживанию и устранению неполадок, что делает систему удобной для использования и поддержки.

Практическая значимость разработанной системы заключается в том, что она может быть применена в организациях различного масштаба для защиты информационных ресурсов и управления доступом. Гибкость и масштабируемость решения позволяют адаптировать его под специфические требования бизнеса, а использование современных стандартов безопасности гарантирует соответствие нормативным требованиям. Система также демонстрирует высокий уровень удобства для пользователей благодаря простому интерфейсу и возможности централизованного управления доступом.

Перспективы развития проекта связаны с внедрением новых технологий и функциональностей, таких как блокчейн для децентрализованной аутентификации или искусственный интеллект для анализа поведения пользователей. Возможна также интеграция дополнительных механизмов защиты, например, биометрической аутентификации или продвинутых систем обнаружения аномалий. Улучшение производительности и масштабируемости системы позволит использовать ее в еще более сложных и распределенных средах.

Таким образом, разработка системы контроля доступа стала успешным примером применения теоретических знаний на практике. Проект продемонстрировал актуальность выбранной темы и значимость создания надежных решений для защиты информации. Результаты работы могут быть использованы как основа для дальнейших исследований и разработок в области информационной безопасности, а также для внедрения в реальных организационных процессах. Созданная система соответствует современным требованиям безопасности и эффективности, что делает ее актуальной и востребованной в условиях постоянно меняющихся угроз и технологий.

# **ИСПОЛЬЗУЕМАЯ ЛИТЕРАТУРА**

**Нормативные документы:**

Конституция Российской Федерации (с учетом поправок, внесен­ных за­конами Российской Федерации «О поправках к Конституции Россий­ской Федерации» от 30 декабря 2008 г. № 6-ФКЗ и от 30 декабря 2008г. № 7-ФКЗ) // Российская газета № 4831 от 21 января 2009 г.

Федеральный закон Российской Федерации от 7 февраля 2011 № 3-ФЗ «О полиции»// Россий­ская газета от 8 февраля 2011 № 5401.

**Основные источники:**

1. Django documentation. URL: <https://docs.djangoproject.com/en/4.1/>
2. Python documentation. URL: <https://docs.python.org/3/>
3. HTML documentation. URL: <https://html.spec.whatwg.org/>
4. CSS documentation. URL: <https://www.w3.org/Style/CSS/>
5. JavaScript documentation. URL: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
6. SQLite documentation. URL: <https://www.sqlite.org/docs.html>
7. Хабр. URL: <https://habr.com/ru/feed/>
8. Stack Overflow. URL: <https://stackoverflow.com/>
9. Туториал по jinja2. URL: <https://dvmn.org/encyclopedia/modules/jinja2/>
10. Github. URL: [https://github.com](https://github.com/)

Ахо А. Построение и анализ вычислительных алгоритмов. системе / Ахо А., Хопкрофт Дж.. – СПб.: Знание, 2020. – 232 с.

Ахо А. Построение и анализ вычислительных алгоритмов. системе / Ахо А., Хопкрофт Дж., Ульман Дж. – М. : Экзамен, 2020. – 575 с.

**Дополнительные источники**:

Управленческая деятельность: структура, функции, навыки персонала / К. Д. Скрипник [и др.]. – М. : Приор, 2022. – 189 с.

Основы программирования: учеб. пособие / С. И. Самыгин [и др.]; под ред. С. И. Самыгина. – Ростов-на-Дону : Феникс, 2007. – 511 с.

**Словари и энциклопедии**

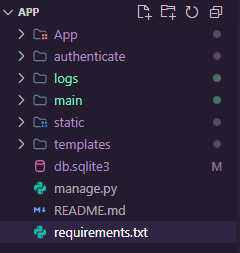
Социальная философия: словарь / под общ. ред. В. Е. Кемерова, Т. Х. Керимова. – М. : Академический Проект, 2011. – 588 с.

**Электронные ресурсы:**

[http://zhurnal.mipt.rssi.ru](http://zhurnal.mipt.rssi.ru/).

# **ПРИЛОЖЕНИЕ**

Приложение 1. Репозиторий проекта.



Приложение 2. Листинг файлов

Листинг settings.py

*"""*

*Django settings for App project.*

*Generated by 'django-admin startproject' using Django 5.0.4.*

*For more information on this file, see*

*https://docs.djangoproject.com/en/5.0/topics/settings/*

*For the full list of settings and their values, see*

*https://docs.djangoproject.com/en/5.0/ref/settings/*

*"""*

*import* os

*from* pathlib *import* *Path*

*# Build paths inside the project like this: BASE\_DIR / 'subdir'.*

*BASE\_DIR* = *Path*(\_\_file\_\_).resolve().parent.parent

*# Quick-start development settings - unsuitable for production*

*# See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/*

*# SECURITY WARNING: keep the secret key used in production secret!*

*SECRET\_KEY* = 'django-insecure-\_f&0rq!a*%a*1)i!=(#4\_5db&xf92pc\*au2mkuntdk+n%)u!&f%0'

*# SECURITY WARNING: don't run with debug turned on in production!*

*DEBUG* = *True*

*ALLOWED\_HOSTS* = ['127.0.0.1']

*# Application definition*

*INSTALLED\_APPS* = [

    'django.contrib.admin',

    'django.contrib.auth',

    'django.contrib.contenttypes',

    'django.contrib.sessions',

    'django.contrib.messages',

    'django.contrib.staticfiles',

    'main',

    'authenticate'

]

*MIDDLEWARE* = [

    'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',

    'django.middleware.common.CommonMiddleware',

    'django.middleware.csrf.CsrfViewMiddleware',

    'django.contrib.auth.middleware.AuthenticationMiddleware',

    'django.contrib.messages.middleware.MessageMiddleware',

    'django.middleware.clickjacking.XFrameOptionsMiddleware',

]

*ROOT\_URLCONF* = 'App.urls'

*TEMPLATES* = [

    {

        'BACKEND': 'django.template.backends.django.DjangoTemplates',

        'DIRS': [os.path.join(*BASE\_DIR*, 'templates')],

        'APP\_DIRS': *True*,

        'OPTIONS': {

            'context\_processors': [

                'django.template.context\_processors.debug',

                'django.template.context\_processors.request',

                'django.contrib.auth.context\_processors.auth',

                'django.contrib.messages.context\_processors.messages',

            ],

        },

    },

]

*WSGI\_APPLICATION* = 'App.wsgi.application'

*LOGGING* = {

    'version': *1*,

    'disable\_existing\_loggers': *False*,

    'handlers': {

        'file': {

            'level': 'INFO',

            'class': 'logging.FileHandler',

            'filename': os.path.join(*BASE\_DIR*, 'logs', 'app.log'),  *# Путь к файлу логов*

        },

        'console': {

            'level': 'INFO',

            'class': 'logging.StreamHandler',

        },

    },

    'loggers': {

*#  Логи все вообще все*

*# 'django': {*

*#     'handlers': ['file', 'console'],*

*#     'level': 'INFO',*

*#     'propagate': True,*

*# },*

        'custom\_logger': {  *# Новый логгер для пользовательских действий*

            'handlers': ['file', 'console'],

            'level': 'INFO',

            'propagate': *False*,

        },

    },

}

*# Database*

*# https://docs.djangoproject.com/en/5.0/ref/settings/#databases*

*DATABASES* = {

    'default': {

        'ENGINE': 'django.db.backends.sqlite3',

        'NAME': *BASE\_DIR* / 'db.sqlite3',

    }

}

*# Password validation*

*# https://docs.djangoproject.com/en/5.0/ref/settings/#auth-password-validators*

*AUTH\_PASSWORD\_VALIDATORS* = [

    {

        'NAME': 'django.contrib.auth.password\_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password\_validation.NumericPasswordValidator',

    },

]

*# Internationalization*

*# https://docs.djangoproject.com/en/5.0/topics/i18n/*

*LANGUAGE\_CODE* = 'en-us'

*TIME\_ZONE* = 'UTC'

*USE\_I18N* = *True*

*USE\_TZ* = *True*

*# Static files (CSS, JavaScript, Images)*

*# https://docs.djangoproject.com/en/5.0/howto/static-files/*

*STATIC\_URL* = 'static/'

*STATICFILES\_DIRS* = [

    os.path.join(*BASE\_DIR*, 'static'),

    os.path.join(*BASE\_DIR*, 'static/img'),

    os.path.join(*BASE\_DIR*, 'static/css'),

    os.path.join(*BASE\_DIR*, 'static/js'),

]

*# Default primary key field type*

*# https://docs.djangoproject.com/en/5.0/ref/settings/#default-auto-field*

*DEFAULT\_AUTO\_FIELD* = 'django.db.models.BigAutoField'