ДЕПАРТАМЕНТ ОБРАЗОВАНИЯ И НАУКИ ГОРОДА МОСКВЫ
ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ПРОФЕССИОНАЛЬНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ Г. МОСКВЫ
«КОЛЛЕДЖ ПРЕДПРИНИМАТЕЛЬСТВА №11»
ЦЕНТР ИНФОРМАЦИОННО–КОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Допустить к защите

Заведующий отделением Центра ИКТ

ГАПОУ КП № 11

_____ Мамихин И.В.

« » 2025 г.

ДИПЛОМНАЯ РАБОТА

Разработка веб-приложения для сервиса «Друг-спортсмен» компании ООО «ШЭРИКС»

по специальности: **09.02.07 Информационные системы и программирование**

Выполнил:

студент группы ИСиП-41

Маркусь Евгений Олегович

Научный руководитель:

преподаватель Центра ИКТ

Панюкова Александра Анатольевна

_____ подпись

_____ подпись

Москва, 2025 г.

СОДЕРЖАНИЕ

АННОТАЦИЯ

Дипломная работа студента группы ИСиП-41 Маркусь Евгения Олеговича на тему: Разработка веб-приложения для сервиса «Друг-спортсмен» компании ООО «ШЭРИКС».

Объем дипломной работы – 70 страниц, на которых размещены 9 рисунков и 5 таблиц. При написании диплома использовано 30 источников.

Объектом дипломной работы является сервис «Друг-спортсмен», который предназначен для помощи пользователям в поиске партнеров для занятий спортом.

Предметом дипломной работы выступает функциональность и особенности реализации данного сервиса, включая его интерфейс, архитектуру и технологии, используемые для разработки.

Результат, который будет представлен к защите дипломной работы: веб-приложение сервиса, через которое можно найти партнёра для занятия спортом.

В состав дипломной работы входит:

– титульный лист;

– аннотация;

– содержание;

– введение;

– четыре основных главы;

– заключение;

– список использованных источников.

Во введении раскрывается актуальность дипломной работы, ставится проблема, цель и задачи дипломной работы, определяются объект, предмет.

В первой главе описаны предметная область, аналоги реализуемого решения, инструменты для реализации информационной системы, разработка дизайна информационной системы, база данных информационной системы,

проектирование информационной системы и описаны требования к разрабатываемому решению.

Во второй главе осуществляется практическая работа, в осуществлении которой реализуется информационная система, которая помогает автоматизировать рабочие процессы сервиса «Друг-спортсмен» компании ООО «ШЭРИКС».

В третьей главе приводится технико-экономическое обоснование. Проводится анализ рынка, определяется цена информационной системы и ее окупаемость.

В четвертой главе описывается охрана труда, жизнедеятельности и безопасность.

В заключении приведены основные выводы по результатам проведённого исследования и реализации информационной системы, а также даны рекомендации по её дальнейшему совершенствованию и внедрению. (“Разработанное решение принято компанией для дальнейшего использования и внедрения”)

В списках используемых источников представлены источники, которые использовались при реализации информационной системы.

ГЛОССАРИЙ

ООО (Общество с ограниченной ответственностью) — Общество с ограниченной ответственностью, это организационно-правовая форма предприятия, в которой ответственность его участников ограничена до размера их вкладов в уставный капитал.

API (RESTful (application programming interface) — это набор готовых функций, протоколов и инструментов, предоставляемых разработчиками для взаимодействия с определенным программным обеспечением, приложением или сервисом. Он определяет правила и способы коммуникации между различными компонентами программного обеспечения, позволяя им обмениваться данными и выполнять определенные операции.

XMPP (eXtensible Messaging and Presence Protocol) — является простым, легковесным и распространенным форматом данных для обмена информацией в web-приложениях и сервисах.

Frontend — презентационная часть информационной или программной системы, её пользовательский интерфейс и связанные с ним компоненты.

Backend — это внутренняя часть продукта, которая находится на сервере и скрыта от пользователей.

БД (база данных) — это организованный список данных, который хранится и управляется в компьютерной системе. БД предназначена для эффективного хранения, доступа и обработки информации.

СУБД (Система Управления Базами Данных) — это программное обеспечение, которое позволяет организовать хранение, управление и обработку больших объемов данных.

DRF (Django Rest Framework) — это расширение фреймворка Django, которое обеспечивает удобные инструменты для разработки web-приложений с использованием RESTful API.

ВВЕДЕНИЕ

ShariX – платформа для создания проектов для реализации sharing-сервисов, предоставляющих информационную услугу для совместного использования потребителями специально выделенных материальных и человеческих ресурсов.

Люди хотят заниматься спортом совместно по разным причинам. Может быть актуальная ситуация, в которой выбранный вид спорта является коллективным, в который одному человеку играть невозможно. Некоторым нравится заниматься с другими людьми для поддержания мотивации. Для них занятие спортом – совместный процесс, но из-за ограничения круга знакомств человек переходит в социальные сети, потому что они созданы для поиска и коммуникации людей, и там возможно найти партнёров для различных видов деятельности. Однако, существующие социальные сети были созданы со специфическим направлением, решая проблемы в своей сфере. Все они подразумевают общение между людьми, но с разными целями. В настоящее время занятие спортом не представлено в отдельной социальной сети с функционалом, необходимым для удовлетворения нужд спортсменов и любителей спорта. Если они захотят найти людей, им придётся использовать одно из существующих решений, если нужно записать место и время встречи – это другое приложение. Таким образом, процесс усложняется взаимодействием с несколькими инструментами. Это не удобно и ограничено. Создание платформы для поиска партнёров и организации совместных занятий спортом – актуальная задача цифрового общества.

Поиск напарника для совместного занятия спортом занимает огромное количество времени. На рынке мобильных приложений отсутствует такое приложение, которое бы позволяла искать напарников по виду спорта, уровню подготовки и другим параметрам. В случае разработки приложения «Друг-спортсмен» закрывается вопрос об автоматизации поиска напарника

для занятия спортом, ведь приложение автоматически покажет вам всех людей, которые подходят по вашим фильтрам и позволит организовать встречи с ними.

Проблема, решаемая в дипломной работе, заключается в невозможности удобно планировать совместные тренировки при желании расширить социальные контакты.

Цель дипломной работы заключается в увеличении клиентской базы и лояльности клиентов к платформе «ShariX» посредством создания сервиса «Друг-спортсмен» в целом, обеспечение работоспособности мобильного приложения друг спортсмен и возможности его администрирования посредством веб приложения.

Для достижения данной цели необходимо решить следующие задачи.

1. Провести исследование предметной области.

2. Исследовать потребности компании.

3. Проанализировать аналогичные информационные системы других компаний.

4. Спроектировать функционал веб-сервиса.

5. Спроектировать базу данных.

6. Спроектировать страницы для административной панели.

7. Определить инструменты для разработки информационной системы.

8. Разработать сервис на основе технического задания.

9. Сделать тестирование решения.

10. Сделать сопроводительную документацию.

11. Обосновать экономическую эффективность информационной системы.

12. Внедрить сервис в эксплуатацию.

Объектом дипломной работы является сервис «Друг-спортсмен», который предназначен для помощи пользователям в поиске партнеров для занятий спортом.

Предметом дипломной работы выступает функциональность и особенности реализации данного сервиса, включая его интерфейс, архитектуру и технологии, используемые для разработки.

Результат, который будет представлен к защите дипломной работы: веб-приложение сервиса, через которое можно найти партнёра для занятия спортом.

ГЛАВА 1 ПРОЕКТИРОВАНИЕ СЕРВИСА ДРУГ-СПОРТСМЕН

1.1 Анализ предметной области

ShariX – это стартап в области набирающих популярность шеринговых сервисов, реализуемый ООО «ШЭРИКС».

ShariX проектируется как платформа, которая включает в себя базу пользователей и инструменты для построения sharing-сервисов. На ее основе можно создавать собственные сервисы – как с помощью ShariX, так и самостоятельно. Одни из таких сервисов: Assist, Drive, Guide.

Шеринговые сервисы в ShariX – сервисы, позволяющие более эффективно управлять человеческими и материальными ресурсами за счет возможности использования данных ресурсов различными заказчиками с помощью специальных программных продуктов.

Базовая ценность ShariX – сохранение ценности человеческого труда в общем объеме оказываемой услуги, при автоматизации процессов, не затрагивающих смысл ожиданий клиентов. Управление доступностью нематериальных ресурсов, составление расписаний, принятие решений о возможности составления цепочек услуг, рейтинг и взаиморасчеты автоматизируется платформой.

Основополагающий принцип Sharix – “исполнитель, который хорош для конкретного клиента”. Мы считаем, что каждый исполнитель – личность, каждый клиент – личность, каждая личность имеет право на свободу выбора, предпочтений и собственный вкус. А истинное равенство заключается не во внешних проявлениях, а в равенстве отношений ко всем участникам процесса, а также их равенстве в отношениях между собой [1].

Предметной областью диплома является онлайн знакомства с целью занятия спортом. Люди хотят заниматься спортом совместно по разным причинам. Может быть такое, что сам спорт является коллективным, в

который одному играть невозможно. Некоторые занимаются с кем-то для поддержания мотивации.

Исходя из опроса населения, проведенного в 2006-2023 годах ВЦИОМ, результаты которого представлены на рисунке 1, отсутствие мотивации и силы воли является препятствием на пути к занятиям спортом у 22% россиян [2].

Рисунок 1 – Результаты опроса населения

Одной из основных причин, почему люди хотят заниматься спортом совместно – соперничество за лучший прогресс. Обычно партнеры соревнуются между собой и стараются выполнять упражнения качественнее. Поэтому почти всегда тренировки в паре получаются гораздо результативнее индивидуальных».

Спорт – это физическая активность, направленная на укрепление здоровья, развитие выносливости и улучшение общего самочувствия. Он помогает поддерживать тело в тонусе и способствует формированию дисциплины и силы воли.

Регулярные занятия спортом положительно влияют не только на физическое, но и на психическое здоровье. Спорт помогает справляться со стрессом, улучшает настроение и повышает качество жизни в целом.

Для многих спорт – это не просто индивидуальное занятие, а совместный процесс. Совместные тренировки и соревнования создают чувство команды, мотивируют и способствуют развитию социальных связей.

Чтобы объединять людей, занимающихся спортом, необходим специализированный сервис. Такой сервис – это не просто программный продукт, а система договорных отношений и взаимодействия с клиентами. Для эффективной работы сервиса нужны разные программные продукты, которые обеспечивают его функционирование.

Автоматизация процессов в сервисе значительно повышает эффективность. Например, если между пользователями заключается договор, то без автоматизации потребуется штат юристов и много времени на ручную обработку, что делает работу неэффективной и затратной.

Веб-приложение необходимо в первую очередь для организации и управления функционированием сервиса. Оно позволяет сервису работать не по принципу «повесили объявление и надеемся, что люди найдутся», а как полноценная единица, связывающая независимых людей, которые могут находиться на значительном расстоянии друг от друга, но в рамках одного города.

Социальные сети играют важную роль в объединении людей с общими интересами, в том числе и в спорте. Для спортсменов нужна специализированная социальная сеть, которая учитывает их особенности и задачи. Если такой соцсети нет, пользователям приходится пользоваться разрозненными решениями, что неудобно и снижает эффективность.

Связка мобильного приложения и веб-приложения – оптимальное решение для современного сервиса. Мобильное приложение удобно для

пользователей, позволяя быстро получить доступ к сервису в любое время и в любом месте. Однако мобильное приложение имеет ограничения по функционалу и удобству работы с большими объемами информации.

Веб-приложение дополняет мобильное, предоставляя расширенный функционал для администрирования. Оно необходимо для выполнения сложных задач и управления сервисом.

Таким образом, мобильное приложение недостаточно для полноценного функционирования сервиса, а веб-приложение необходимо для администрирования. Комбинация веб и мобильного приложений обеспечивает максимальную гибкость, удобство и эффективность, повышая лояльность пользователей и качество сервиса.

Приложение «Друг-спортсмен» решит перечисленные проблемы, реализуя сложные информационные процессы, основанные на тщательном анализе потребностей целевой аудитории и важных требований к практической разработке. Это позволит создать удобный, функциональный и эффективный сервис для объединения спортсменов.

1.2 Требования к разрабатываемому решению

Сервис создаётся для компании на основе программного обеспечения ShariX Open [3]. В рамках проекта планируется разработка backend-части и веб-приложения сервиса, а также доработка frontend-части в составе sharix-admin.

При разработке приложения Друг-спортсмен необходимо придерживаться следующих выявленных требований:

Автоматизация: практическая разработка должна быть нацелена на создание максимально возможно полного автоматизированного сервиса для минимизации затрат со стороны компании. Автоматизация значительно снижает вероятность ошибок, минимизирует ручной труд, повышает скорость выполнения многократно повторяющихся задач, качество работы, количество

данных, которыми возможно оперировать для расчета и поддержки процессов, точность управления. Благодаря этим преимуществам возрастает уровень конкурентоспособности на рынке, идет мощное использование ресурсной базы.

Модульность: не нарушать имеющуюся структуру и соблюдать имеющиеся подходы к разработке.

Лицензионность: соблюдать лицензии, т.к. в основе лежит продукт с открытым исходным кодом.

Локализация: приложение должно поддерживать возможность переводить на несколько языков, обеспечивая гибкость в адаптации интерфейса под потребности пользователей из разных стран и культур.

Безопасность: распространенным приемом на этом этапе является создание модели угроз. Обращаться к стандартам и правилам безопасности.

Возможность коммерциализировать: возможность добавления платных услуг в дальнейшем.

Простота в использовании: пользователь без опыта в подобных приложениях должен интуитивно разобраться в работе приложения.

Документация: программный код должен быть понятным, легко читаемым и задокументированным для дальнейшего использования и развития.

Функциональные требования к разработке backend части веб-сервиса предполагают шаги:

1) разработка моделей;

2) разработка API.

Функциональные требования к доработка backend части ShariX Open:

1) доработка моделей;

2) доработка обработчиков.

Функциональные требования к доработке веб-интерфейса модуля sharix-admin описаны ниже.

1. Регистрация ресурса предполагает процесс создания и внесения в систему нового ресурса, который будет использоваться для предоставления услуг или взаимодействия с пользователями. Включает ввод необходимых данных и настройку параметров ресурса для корректной работы в рамках платформы.

2. Регистрация партнера требует оформления учетной записи контрагента – юридического или физического лица, сотрудничающего с компанией и участвующего в оказании услуг через платформу. Партнер получает доступ к функционалу для взаимодействия с исполнителями и управления своими операциями.

3. Регистрация пользователя как исполнителя предусматривает создание профиля пользователя, который будет выполнять услуги на платформе. Включает подтверждение личности, предоставление необходимых документов и настройку учетной записи для возможности заключения договоров и участия в сервисе.

4. Управление свойствами (ресурса, партнера, исполнителя) требует администрирования и настройки характеристик и параметров ресурсов, партнеров и исполнителей. Позволяет обновлять информацию, изменять права доступа, а также контролировать состояние и настройки каждого элемента системы.

5. Генерация отчетов всем пользователем системы предполагает Формирование и предоставление аналитических и статистических отчетов для всех категорий пользователей и администраторов сервиса. Отчеты помогают контролировать эффективность работы, отслеживать ключевые показатели и принимать управленческие решения.

6. Управление правами для сотрудников сервиса требует настройки и распределения уровней доступа и полномочий среди сотрудников, обеспечивающих работу сервиса. Позволяет разграничивать функции и ответственность администраторов, модераторов, технической поддержки и других ролей для безопасного и эффективного управления системой.

1.3 Анализ аналогов панелей администрирования

В данном разделе проведен сравнительный анализ аналогичных информационных систем, позволяющих автоматизировать процессы управления задачами. Для анализа были выбраны системы: Laravel Nova, DirectAdmin.

Проанализируем приложение, автоматизирующее планирование и управление задачами, на примере Laravel Nova [4] и DirectAdmin.

Laravel Nova – официальная админ-панель для Laravel, предназначенная для быстрого создания административных интерфейсов и управления данными проектов.

Отличается от «Друг-спортсмен» тем, что ориентирована на разработчиков и интегрируется только с Laravel-приложениями, не предоставляя специализированных функций для спортивных сервисов или социальных взаимодействий.

Приложение хорошо тем, что обладает интуитивно понятным интерфейсом, мощным CRUD, гибкой системой прав доступа, поддержкой кастомных полей и расширяемостью за счёт собственных инструментов и интеграций.

Приложению недостаёт готовых решений для специфических задач спортивных сервисов, не подходит по стеку технологий, является платным и не является open source, что ограничивает гибкость и доступность для проекта.

DirectAdmin – панель управления хостингом, предназначенная для простого и быстрого администрирования серверов и сайтов.

Отличается от «Друг-спортсмен» тем, что ориентирована на управление веб-хостингом и серверными ресурсами, а не на социальные или спортивные сервисы.

Таблица 1 сравнивает Laravel Nova и DirectAdmin — две платные административные панели с разной специализацией. Laravel Nova предназначена для разработчиков Laravel и предлагает гибкую настройку CRUD интерфейсов с удобным интерфейсом, но требует знаний PHP и не является open source. DirectAdmin — панель для администрирования серверов и сайтов, отличается низкими системными требованиями и доступной ценой, но имеет ограниченную русификацию и сложности с обновлениями. Обе панели не подходят для проектов со специфическим стеком или спортивной направленностью.

Преимущества приложения в том, что оно обладает низкими системными требованиями, высокой скоростью работы, доступной ценой и удобным интерфейсом для разных уровней пользователей.

Приложению недостаёт специализированных функций для спортивных сервисов, сложностей с обновлениями, ограниченной русификации, а также не подходит по стеку технологий и не является open source.

Кроме того, DirectAdmin не бесплатен, что может увеличить затраты на проект по сравнению с бесплатными решениями.

Таблица 1 показывает, что ни Laravel Nova, ни DirectAdmin не поддерживают функции размещения заказа «здесь и сейчас» и выбора исполнителя. Обе платформы ориентированы на управление данными и администрирование, но не предназначены для оперативного распределения задач или заказов.

Таблица 1 – Сравнительный анализ функциональных возможностей

Функция	**Laravel Nova**	**DirectAdmin**
Возможность разместить заказ здесь и сейчас	Нет	Нет
Возможно выбрать человека исполнителя	Нет	Нет

Обе рассмотренные системы имеют существенные ограничения для использования в проекте «Друг-спортсмен». Laravel Nova, несмотря на мощные возможности и удобство, не подходит по технологическому стеку и специфике, а также является платной и закрытой. Это ограничивает гибкость внедрения и адаптации панели под уникальные задачи проекта.

Таблица 2 – Сравнительный анализ административных панелей

Приложение	**Laravel Nova**	**DirectAdmin**
Описание	Официальная административная панель для Laravel, для быстрого создания административных интерфейсов и CRUD.	Панель управления хостингом для администрирования серверов и сайтов.
Целевая аудитория	Разработчики, работающие с Laravel проектами.	Администраторы серверов и веб-хостинга.
Преимущества	Интуитивный интерфейс, гибкая настройка, поддержка кастомных полей, расширяемость.	Низкие системные требования, высокая скорость, доступная цена, удобный интерфейс.
Недостатки	Платная, не open source, требует знания Laravel и PHP, не подходит по стеку технологий, нет готовых решений для спорта.	Платная, не open source, ограниченная русификация, сложности с обновлениями, не подходит по стеку технологий, нет функций для спорта.

DirectAdmin ориентирована на управление серверными ресурсами и не содержит функций, необходимых для спортивного или социального сервиса. Кроме того, она также является платной и не имеет открытого исходного

кода, что усложняет кастомизацию и интеграцию с другими компонентами проекта.

Таким образом, закрытый исходный код, платность и отсутствие специализированного функционала делают обе панели неприемлемыми для данного проекта. Для успешной реализации требуется поиск или разработка решения с открытым кодом, специально адаптированного под нужды спортивного сервиса.

Существуют аналоги рассматриваемого сервиса, однако их административные панели являются внутренними разработками компаний и не предоставляются в открытом доступе. В связи с этим провести прямое и детальное сравнение с ними затруднительно, так как отсутствует возможность ознакомиться с их функционалом и интерфейсом.

1.4 Проектирование функционала веб-сервиса

Первым шагом в разработке веб-приложения сервиса "Друг спортсмен" является этап проектирования. На этом этапе формируются основные концепции и структура будущего решения, что позволяет четко определить требования, функциональность и взаимодействие компонентов системы. Качественное проектирование обеспечивает основу для успешной реализации и последующей поддержки проекта.

Существует несколько подходов к проектированию, одним из которых является использование различных видов диаграмм, которые визуализируют архитектуру, процессы и данные системы. Такой подход помогает разработчикам, аналитикам и заказчикам лучше понимать структуру и логику работы системы, а также выявлять потенциальные проблемы на ранних стадиях.

В рамках данного проекта будут представлены следующие диаграммы [5]: диаграмма прецедентов (Use Case Diagram), которая описывает взаимодействие пользователей с системой и основные функции; диаграмма

классов (Class Diagram), показывающая структуру данных и взаимосвязи между объектами; диаграмма последовательностей (Sequence Diagram), иллюстрирующая порядок взаимодействия компонентов во времени; а также диаграмма деятельности (Activity Diagram), отражающая бизнес-процессы и логику выполнения операций.

Каждая из этих диаграмм будет подробно рассмотрена и прокомментирована в тексте. Для удобства восприятия диаграммы будут представлены с увеличенным шрифтом, что обеспечит их читаемость и позволит лучше понять ключевые элементы и связи. При необходимости, все диаграммы могут быть вынесены в приложение, чтобы не перегружать основной текст, сохраняя при этом доступность и наглядность проектной документации.

Применение различных видов диаграмм на этапе проектирования значительно упрощает процесс анализа и уточнения требований к системе. Визуализация архитектуры и процессов помогает выявить возможные несоответствия и узкие места еще до начала разработки, что экономит время и ресурсы команды. Кроме того, диаграммы служат удобным инструментом для согласования видения проекта между всеми заинтересованными сторонами, обеспечивая единое понимание целей и задач.

Использование таких моделей также способствует более эффективному распределению ролей и обязанностей внутри команды разработчиков. Четкое представление о структуре данных и последовательности взаимодействий позволяет создавать более устойчивую и масштабируемую архитектуру, а также упрощает последующую поддержку и развитие сервиса. В итоге, системный подход к проектированию повышает качество конечного продукта и снижает риски, связанные с ошибками на ранних этапах разработки.

Рисунок 2 – диаграмма прецедентов

На рисунке 2 представлена Use Case диаграмма, которая представляет ключевые сценарии взаимодействия пользователей с системой «Друг-спортсмен». На диаграмме показаны основные участники – Пользователь и Администратор – и их основные действия: регистрация, авторизация, создание и просмотр заказов, назначение исполнителей, управление пользователями и формирование отчетов. Эта визуализация помогает понять, какие функции доступны разным ролям и как они взаимодействуют с системой, что важно для правильного проектирования и дальнейшей реализации функционала.

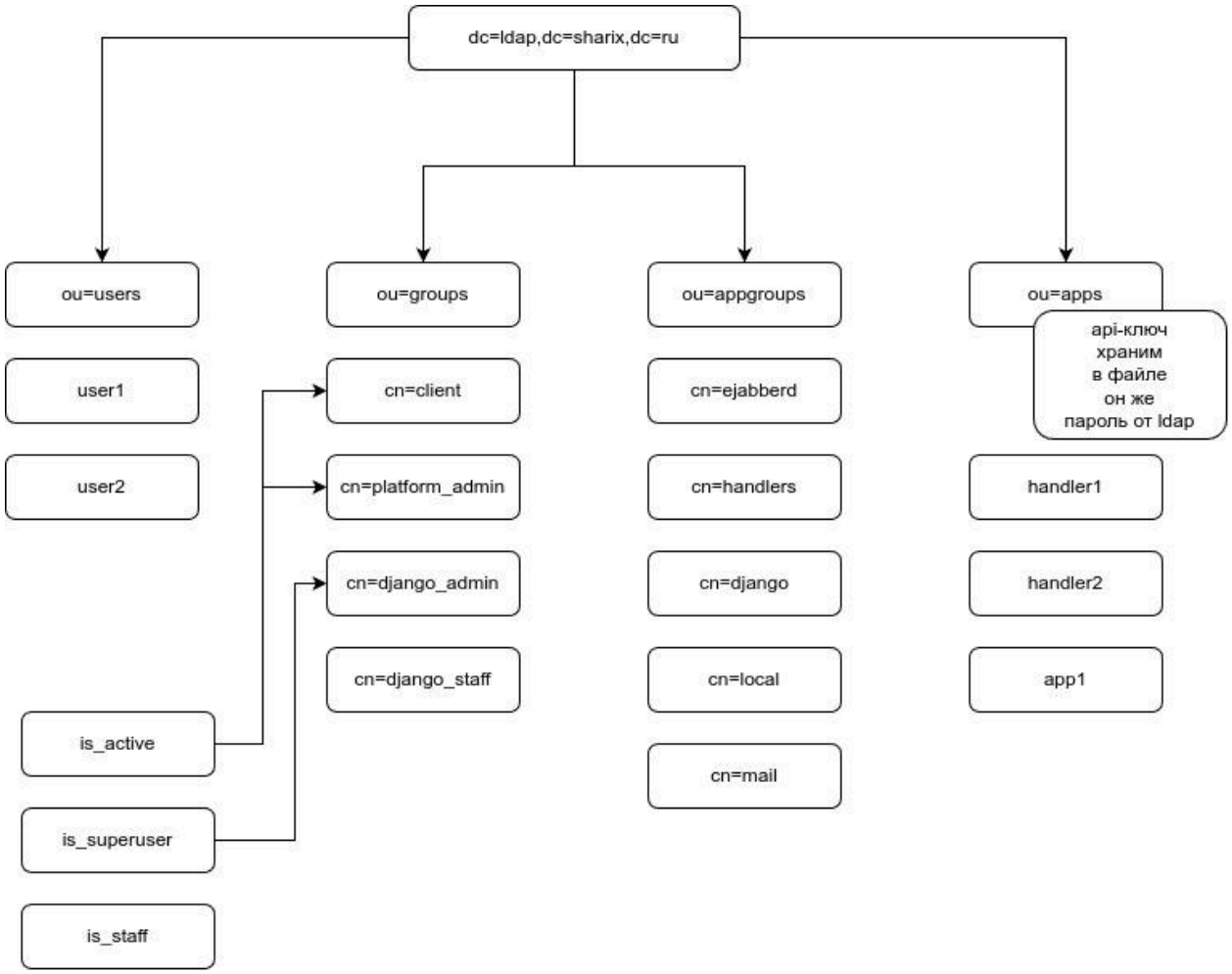


Рисунок 3 – Схема ldap

Схема LDAP, представленная на рисунке 3, отражает иерархическую структуру каталога, основанную на доменных компонентах (dc=ldap, dc=sharix, dc=ru), которые образуют корневую точку дерева. Внутри этого дерева выделены организационные единицы (organizational units, ou), такие как ou=users, ou=groups, ou=appgroups и ou=apps, которые группируют объекты по функциональному назначению.

В рамках ou=users находятся записи пользователей, например, user1 и user2, обладающие различными ролями и атрибутами, такими как cn=client, cn=platform_admin, cn=django_admin, cn=django_staff, что указывает на их принадлежность к разным группам и уровням доступа. Атрибуты is_active, is_staff и is_superuser характеризуют состояние и права пользователей.

В разделе ou=groups и ou=appgroups хранятся группы и приложения, например, cn=ejabberd, cn=handlers, cn=django, а также конкретные обработчики handler1 и handler2. Это позволяет централизованно управлять доступом и правами в рамках различных сервисов и приложений.

Особое место занимает хранение конфиденциальных данных, таких как api-ключ и пароль от LDAP, которые, согласно схеме, хранятся в файле, обеспечивая безопасность и удобство управления.

Таким образом, данная LDAP-схема обеспечивает структурированное хранение информации о пользователях, группах и приложениях в иерархическом каталоге, позволяя эффективно управлять доступом и аутентификацией в корпоративной среде. Она соответствует стандартам LDAP, включая определение объектных классов, атрибутов и правил их использования, что гарантирует целостность и согласованность данных в каталоге

Такая визуализация помогает понять логику обмена данными и ключевые шаги, обеспечивающие безопасность и удобство доступа к системе.

1.5 Проектирование базы данных

Для начала проектирования БД необходимо рассмотреть каким образом будет происходить интеграция БД сервиса с текущей БД организации [6].

Перед началом разработки моделей сервиса «Друга спортсмена» потребовалось внести корректировки в модели ShariX Open (приложения 1, 2, 3, 4) [7].

Были согласованы таблицы с учётом связей между ними, используя вторичные ключи (FK). Избыточные поля удалены, недостающие – добавлены, что позволило значительно улучшить структуру базы данных. В результате схема приобрела аккуратный и понятный вид: она стала легко читаемой, а взаимосвязи между таблицами – наглядными и очевидными.

В обновлённая схеме модуля METASERVICE_SYNCED исправлены отношения между таблицами, поля отсортированы, а также добавлены, перенесены и удалены необходимые элементы (приложение 5). Таблицы размещены таким образом, чтобы максимально удобно прослеживалась их взаимосвязь. Аналогичные исправления были применены и к остальным таблицам, перечисленным в приложениях 1-4.

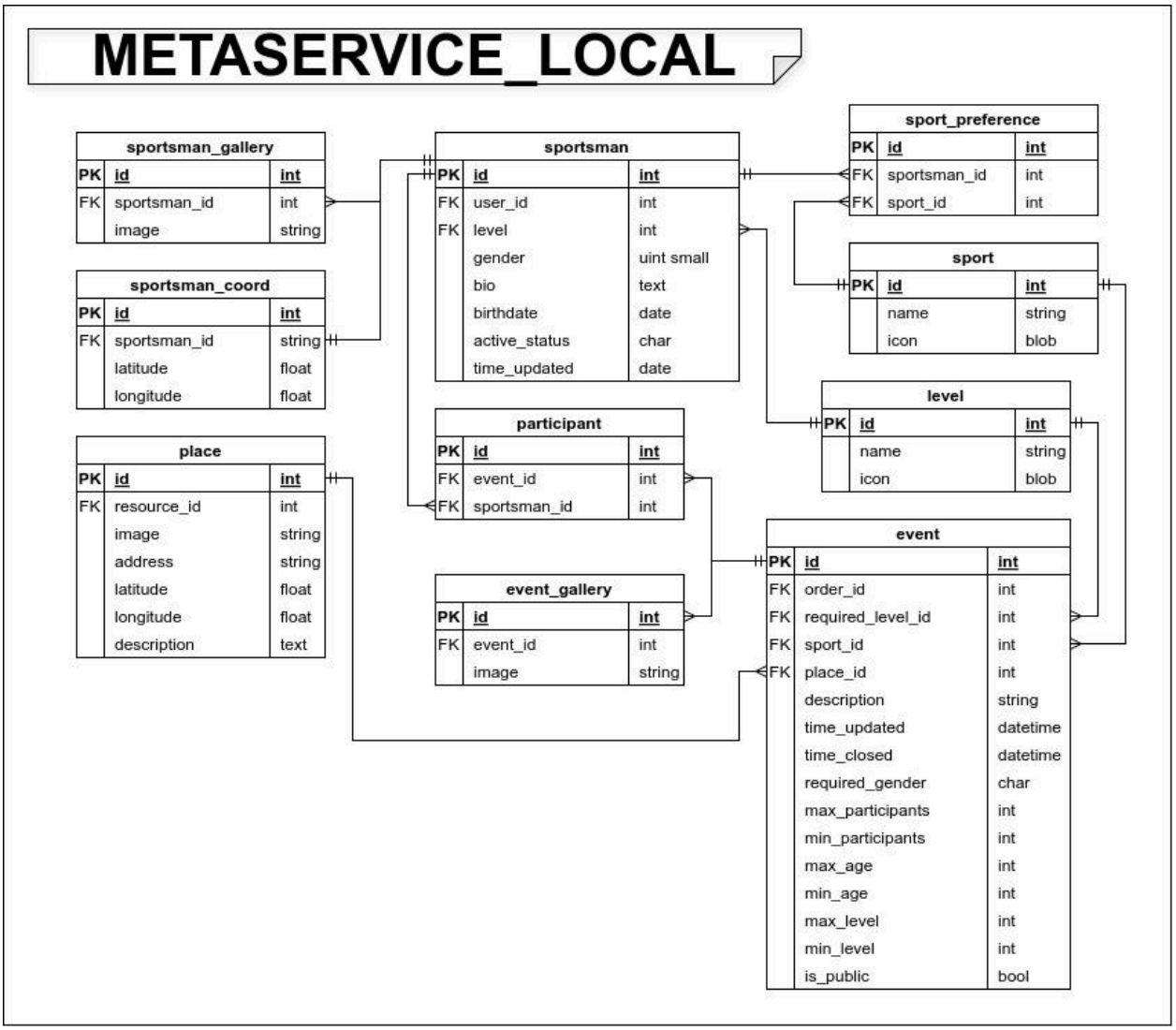


Рисунок 4 – Схема БД Sharix Friend

На основе собранной информации спроектирована база данных для веб-сервиса «Друг-спортсмен», обеспечивающая хранение и обработку данных о пользователях, их достижениях и мероприятиях. Определены основные сущности, атрибуты и связи, что позволило создать логичную и

масштабируемую структуру. Для наглядности создана диаграмма в draw.io с ключевыми таблицами и их взаимосвязями.

Учтены требования к производительности и расширяемости, предусмотрена интеграция с LDAP для централизованной аутентификации. Реализованы воспроизводимые сборки и автоматизация процессов сборки, тестирования и развертывания, что повышает стабильность и надёжность системы.

На рисунке 4 изображена расширенная модель для сервиса Друг Спортсмен. БД спроектирована с учётом гибкости и минимизации избыточности данных, что позволяет эффективно работать с большими объёмами информации и поддерживать развитие приложения «Друг спортсмен».

Основной сущностью системы выступает таблица sportsman, в которой хранится информация о каждом спортсмене: уникальный идентификатор (id), внешний ключ на пользователя (user_id), уровень подготовки (level), пол (gender), краткая биография (bio), дата рождения (birthdate), статус активности (active_status), а также временные метки обновления данных (time_updated). Такая структура позволяет гибко управлять профилями спортсменов и расширять их атрибуты по мере необходимости. Важной особенностью является наличие внешних ключей, обеспечивающих целостность данных и связь с другими таблицами, например, с таблицей пользователей.

Диаграмма базы данных, представленная в drawio, также включает таблицу event, предназначенную для хранения информации о спортивных мероприятиях. В ней фиксируются такие параметры, как идентификатор события, связанные заказы (order_id), необходимый уровень и пол участников, описание мероприятия, место проведения (place_id), а также временные метки. Такая модель данных обеспечивает возможность быстрого

поиска, фильтрации и анализа информации о спортсменах и событиях, что способствует эффективной поддержке пользовательских сценариев и развитию функциональности системы «Друг спортсмен».

1.6 Проектирование страниц для административной панели

Заказчиком уже предоставлены макеты страниц в программе Figma, поэтому дальнейшее проектирование макета не требуется. В рамках реализации проекта предусмотрены следующие основные функциональные возможности.

1. Регистрация ресурса.
2. Регистрация партнера.
3. Регистрация пользователя в роли исполнителя.
4. Управление свойствами ресурса, партнера и исполнителя.
5. Генерация отчетов для всех категорий пользователей.
6. Управление правами доступа сотрудников сервиса, включая администраторов, модераторов, техническую поддержку и другие роли.

Данная структура обеспечивает гибкое и удобное взаимодействие с системой, позволяя эффективно контролировать и администрировать все ключевые компоненты проекта.

1.7 Стек разработки для реализации веб-сервиса

Для реализации веб-сервиса используются инструменты, которые используются заказчиком.

В процессе разработки фронтенда использовались современные веб-технологии: HTML, CSS и JavaScript, а также популярный фреймворк Bootstrap 5, обеспечивающий адаптивный и эстетичный дизайн интерфейса. Для серверной части применялись языки программирования Python и ECMAScript, а также скрипты на Bash для автоматизации задач. Взаимодействие между клиентом и сервером организовано через RESTful API, что обеспечивает гибкость и масштабируемость системы.

В качестве backend-фреймворка выбран Django, который предоставляет мощный и удобный инструментарий для построения веб-приложений. Для хранения данных используется надежная и производительная база данных PostgreSQL. В работе над проектом применялись редакторы кода VSCode и Vim, а для визуализации структуры данных – утилита Graphviz. Тестирование API осуществлялось с помощью Postman, а для обмена сообщениями и коммуникации использовались программы Jabber и Converse. Такой набор технологий и инструментов позволил создать стабильное и функциональное приложение.

ГЛАВА 2 РЕАЛИЗАЦИЯ ВЕБ-СЕРВИСА

В рамках дипломного проекта была проведена доработка и рефакторинг существующего программного продукта ShariX Open. Особое внимание уделялось расширению функциональности сервиса «Друг спортсмен», интегрированного в ShariX Open, что позволило значительно повысить его возможности и удобство использования. Были усовершенствованы обработчики данных, обеспечившие более эффективную и корректную работу системы. В работе широко использовались источники [10–30].

Кроме того, проведено комплексное тестирование новых и доработанных компонентов, что гарантировало стабильность и надежность работы сервиса. Особое значение имело документирование всех изменений и функций, а также локализация интерфейса, что обеспечило удобство использования приложения для пользователей с различными языковыми предпочтениями. Все эти мероприятия способствовали созданию качественного и функционального программного продукта.

2.1 Разработка сервиса на основе технического задания

В ходе начала работы над реализацией веб-приложения сервиса «Друг спортсмен» было выявлено, что для успешного выполнения задачи требуется доработка ядра проекта ShariX Open. В связи с этим ShariX Open подвергся значительному рефакторингу и усовершенствованиям, направленным на улучшение архитектуры, исправление ошибок, повышение стабильности и удобства сопровождения кода. Ниже приведены ключевые изменения, сгруппированные по основным направлениям.

2.1.1 Рефакторинг моделей и структуры базы данных

Одной из ключевых задач в ходе разработки стала переработка моделей данных, направленная на оптимизацию структуры базы данных и улучшение логики взаимодействия между сущностями.

В первую очередь была проведена очистка моделей от избыточных полей и связей. Так, удаление ForeignKey-поля, например, поля company из модели Provider, позволило упростить структуру базы данных и устранить циклические зависимости, которые затрудняли дальнейшее развитие и сопровождение системы. Аналогично, удаление поля id_provider из модели Service способствовало сокращению дублирования данных и повышению целостности информации.

Для повышения читаемости и понятности кода были переименованы некоторые поля и классы. В частности, поле price_alg было заменено на price_type, а service_type – на service_category. Такие изменения улучшили семантику и сделали структуру моделей более интуитивной для разработчиков.

С Интеграция LDAP (Lightweight Directory Access Protocol) с различными компонентами информационной системы, такими как ejabberd и Django, является важным этапом при построении корпоративных решений, обеспечивающих централизованное управление пользователями и безопасный доступ к ресурсам.

LDAP представляет собой протокол для доступа и управления распределёнными каталогами, который широко применяется для хранения информации об учетных записях пользователей, группах, а также других атрибутах, необходимых для аутентификации и авторизации. Его использование позволяет централизовать управление пользователями, упростить администрирование и повысить безопасность системы.

В контексте интеграции с XMPP-сервером ejabberd, LDAP используется для организации аутентификации пользователей и управления их профилями. Конфигурация ejabberd предусматривает возможность подключения к LDAP-серверу, например, Active Directory или OpenLDAP, для проверки учетных данных пользователей при входе в систему. В файле конфигурации

ejabberd (ejabberd.yml) задаются параметры подключения к LDAP-серверу: адрес сервера, порт, учетная запись с правами для поиска в каталоге, база поиска, а также атрибуты, которые используются в качестве идентификаторов пользователей (например, sAMAccountName в Active Directory). Такой подход позволяет пользователям входить в систему под своими доменными учетными записями, обеспечивая единый вход (Single Sign-On) и упрощая управление доступом.

Помимо аутентификации, ejabberd может использовать LDAP для загрузки и синхронизации данных пользователей, таких как контактная информация, отображаемая в vCard. Это позволяет автоматически заполнять профили пользователей на основе данных из корпоративного каталога, что улучшает качество и актуальность информации в мессенджере. Также возможна настройка общих списков контактов (shared rosters), которые формируются на основе групп пользователей в LDAP, что облегчает организацию коммуникаций внутри компании.

В свою очередь, интеграция LDAP с Django реализуется через использование специализированных библиотек и модулей, обеспечивающих аутентификацию пользователей на основе LDAP-запросов. Это позволяет приложению Django использовать корпоративный каталог для проверки учетных данных, что обеспечивает единообразие политики безопасности и упрощает администрирование пользователей. В Django конфигурация подключения к LDAP обычно задается в настройках проекта, где указываются параметры сервера, база поиска, фильтры и атрибуты пользователя. Кроме того, возможно расширение функционала для синхронизации дополнительных данных пользователей из LDAP в модель пользователя Django.

Для упрощения развертывания и настройки LDAP-сервера была создана генерация начальной конфигурации файла init.ldif с помощью

скрипта, написанного на Python (приложение 6). Этот скрипт автоматизирует создание базовой структуры каталога, включая необходимые организационные единицы и учетные записи, что значительно ускоряет процесс подготовки LDAP-сервера к работе и снижает вероятность ошибок при ручном вводе данных.

Объединение LDAP с такими компонентами, как ejabberd и Django, обеспечивает создание единой инфраструктуры аутентификации и управления пользователями, что особенно важно для корпоративных систем с большим числом пользователей и сложной структурой прав доступа. Это повышает безопасность, снижает издержки на поддержку и улучшает пользовательский опыт за счет единого входа и актуальных данных профиля.

В итоге, интеграция LDAP с ejabberd и Django представляет собой эффективное решение для централизованного управления учетными записями, обеспечивая надежную и удобную систему аутентификации, а также поддерживая синхронизацию пользовательских данных и организацию коммуникаций внутри корпоративной среды.

Наконец, была скорректирована логика связей между сущностями. Например, поле service_status было перемещено из модели Service в модель Provider, что более точно отражает принадлежность статуса к поставщику услуг и улучшает семантическую структуру данных.

В результате проведённой работы модели данных стали более лаконичными, логичными и удобными для дальнейшего развития и поддержки системы.

2.1.2 Совершенствование скрипта установки и инфраструктуры

Скрипт install.sh и связанная с ним инфраструктура были значительно доработаны с целью автоматизации процесса развертывания и повышения его надежности.

В процессе модернизации была выполнена интеграция systemd-сервиса, включающая создание system-юнита непосредственно во время установки. Это позволило упростить управление фоновыми процессами и обеспечить их стабильную работу [8].

Особое внимание было уделено автоматизации генерации и заполнения тестовыми данными, что значительно ускорило подготовку среды для тестирования и разработки.

Кроме того, была проведена актуализация используемых в проекте библиотек, что повысило безопасность и производительность системы.

Для повышения устойчивости и универсальности скрипта были внесены многочисленные правки в install.sh, направленные на корректную обработку ошибок и улучшение кросс-платформенной совместимости. В результате скрипт стал более стабильным и удобным в использовании на различных системах.

Параллельно с этим была реализована современная и гибкая система конфигурации приложения, основанная на использовании переменных окружения. Изначально в проекте применялся шаблонный механизм разделения настроек через файлы core/_settings_vars.py и core/settings_vars.py, что является классическим подходом в Django для разграничения конфигураций разработки и продакшена, а также для обеспечения безопасности за счет исключения хранения чувствительных данных непосредственно в коде. Такой подход позволял централизованно управлять параметрами окружения и упрощал процесс развертывания приложения в разных средах.

Однако с целью повышения удобства и надежности работы с конфигурацией был внедрен механизм использования файла .env, в котором хранятся все необходимые переменные среды. Этот метод стал стандартом в современных веб-приложениях благодаря своей простоте и эффективности.

Использование .env файла обеспечивает более надежное и безопасное хранение конфиденциальных данных, таких как ключи доступа, пароли и параметры подключения к базе данных, а также облегчает переключение между различными конфигурациями без необходимости изменения исходного кода.

Особое внимание было уделено разделению конфигураций для различных сред — разработки (dev) и продакшена (prod). Для этого были созданы отдельные файлы .env.dev и .env.prod, каждый из которых содержит специфичные для соответствующей среды параметры. Такой подход позволяет изолировать настройки, минимизировать риски случайного использования данных промышленной эксплуатации в процессе разработки и наоборот, а также упростить процесс автоматического развертывания и тестирования.

При запуске приложения или скриптов происходит загрузка соответствующего файла окружения в зависимости от текущей среды, что обеспечивает гибкость и безопасность конфигурации.

В итоге внедрение разделения конфигураций через отдельные .env файлы для dev и prod сред значительно повысило удобство сопровождения проекта, улучшило управляемость и безопасность, а также способствовало более прозрачному и контролируемому процессу развертывания и эксплуатации приложения.

2.1.3 Рефакторинг кода и документации

Рефакторинг затронул не только функциональную часть проекта, но и стиль написания кода, структуру файлов, а также документацию.

В рамках работы было проведено форматирование кода в соответствии с единым стилем, что обеспечивалось использованием среды разработки PyCharm и специализированных линтеров. Это позволило повысить читаемость и поддерживаемость кода.

Особое внимание уделялось удалению устаревших и неиспользуемых элементов. Были устранены неактуальные файлы и произведён откат ошибочных изменений, что способствовало упрощению структуры проекта и снижению технического долга.

Также была проведена работа с документацией: внесены необходимые правки и дополнения, добавлены подробные описания, что улучшило понимание функционала и облегчило дальнейшее сопровождение системы.

В рамках выполнения дипломного проекта была проведена комплексная работа по улучшению документации кода, что значительно повысило качество сопровождения и понимания функционала системы. Особое внимание уделялось описаниям полей моделей Django с использованием параметра help_text. В процессе разработки было принято решение вынести эти описания в отдельные файлы и структуры, что позволило существенно разгрузить сами модели и сделать код более структурированным и удобным для восприятия.

Вынос описаний полей в отдельные файлы является важным шагом в организации кода, особенно в крупных и сложных проектах. При большом количестве полей и сложной бизнес-логике длинные текстовые описания, размещённые непосредственно в моделях, приводят к загромождению кода, усложняют его чтение и затрудняют внесение изменений. Централизация описаний в отдельных файлах позволяет сократить объем кода моделей, сделать их более компактными и понятными, а также упростить процесс обновления документации. Это особенно важно при командной разработке, где единообразие и доступность информации играют ключевую роль.

Кроме того, такая организация способствует улучшению структуры проекта в целом. Разделение кода и описаний помогает разработчикам быстрее ориентироваться в функционале, облегчает процесс тестирования и сопровождения системы. В результате повышается качество программного

продукта, снижается вероятность ошибок и ускоряется процесс внедрения новых функций.

Таким образом, вынос help_text в отдельные файлы и структуры стал эффективным решением, которое положительно сказалось на удобстве работы с кодом и документацией. Это позволило не только улучшить понимание функционала системы, но и значительно облегчить дальнейшее сопровождение проекта, что является важным аспектом при разработке сложных программных решений.

2.1.4 Создание единой системы авторизации пользователей сервиса через ldap

В рамках работы была реализована система единой авторизации пользователей сервиса с использованием протокола LDAP. Для автоматизации процесса создания базовых учетных записей пользователей был разработан механизм генерации файла инициализации init.ldif, который обеспечивает корректное и удобное добавление пользователей в LDAP-репозиторий.

Кроме того, была выполнена интеграция различных компонентов системы — сервера обмена сообщениями ejabberd, веб-приложения на базе Django и сопутствующих обработчиков — с LDAP. Это позволило обеспечить централизованное управление учетными данными и унифицированный процесс аутентификации пользователей во всех подсистемах сервиса, повысив безопасность и удобство эксплуатации.

2.1.5 Улучшение административной панели и разработка страниц

Административная панель была существенно доработана для повышения удобства и эффективности управления данными.

В частности, была реализована логика генерации системных тикетов, включающая функции create_main_ticket, create_partner_role_activation_ticket

и дополнительные вспомогательные методы, что автоматизировало и упростило процесс создания заявок.

Кроме того, проведена работа по исправлению ошибок: внесены корректировки в процесс генерации администраторов и обеспечено правильное отображение полей, что повысило стабильность и удобство использования панели.

Первым этапом работы с платформой является регистрация партнёра — уникального объекта, обладающего набором характеристик и параметров. Процесс регистрации реализован через интуитивно понятную форму, которая позволяет ввести все необходимые данные, включая название компании, контактную информацию, тип партнёра и другие свойства. Система обеспечивает валидацию введённой информации и сохраняет данные в базе, что позволяет использовать партнёров в различных бизнес-процессах.

Разработаны удобные страницы для управления партнёрами сервиса, ресурсами и другими связанными объектами. Эти страницы предоставляют полный набор инструментов для просмотра, редактирования и удаления записей, а также для мониторинга статусов и активности. Такое централизованное управление значительно упрощает администрирование и повышает эффективность работы с платформой.

Для упрощения процесса оформления сотрудника сервиса создана отдельная форма подачи заявки. Через неё пользователи могут заполнить необходимые данные и отправить заявку на получение соответствующего статуса. Система автоматически проверяет корректность введённой информации и передаёт заявку на рассмотрение, что ускоряет процесс аккредитации и интеграции новых сотрудников.

Для расширения экосистемы сервиса предусмотрена возможность регистрации партнёров – организаций или физических лиц, которые взаимодействуют с ресурсами и пользователями. Регистрация партнёра

включает ввод контактных данных, информации о компании и условиях сотрудничества. Этот модуль обеспечивает централизованное хранение и управление партнёрской информацией, что способствует прозрачности и эффективности взаимодействия.

Особое внимание уделено регистрации пользователей, выступающих в роли исполнителей. Такой пользователь получает доступ к специализированному функционалу, позволяющему выполнять задачи, связанные с ресурсами и партнёрами. Процесс регистрации исполнителя включает подтверждение квалификации и настройку персональных параметров, что гарантирует высокий уровень доверия и качества предоставляемых услуг.

Система предоставляет мощный интерфейс для управления свойствами всех ключевых сущностей – ресурсов, партнёров и исполнителей. Администраторы и уполномоченные сотрудники могут изменять, добавлять или удалять характеристики, адаптируя систему под текущие требования бизнеса. Такой подход обеспечивает гибкость и масштабируемость платформы, позволяя быстро реагировать на изменения внешней среды.

Важной составляющей платформы является модуль генерации отчётов, доступный всем категориям пользователей в зависимости от их прав. Отчёты формируются на основе актуальных данных и могут включать статистику по ресурсам, активности партнёров и исполнителей, а также показатели эффективности работы сервиса. Автоматизация создания отчётов значительно упрощает процесс анализа и принятия решений.

Для обеспечения безопасности и корректного распределения обязанностей реализована система управления правами доступа. В зависимости от роли – администратор, модератор, техническая поддержка и другие – пользователи получают соответствующие разрешения на просмотр, редактирование и администрирование данных. Такой механизм гарантирует

защиту информации и поддерживает высокий уровень организационной дисциплины.

2.1.6 Модернизация скрипта обработчика и замена библиотеки

Возникла проблема с запуском скрипта обработчика, которая была связана с тем, что используемая библиотека давно не обновлялась и перестала корректно работать с новой версией Python. В результате было принято решение заменить устаревшую библиотеку xmpppy на более современную и поддерживаемую slixmpp.

В связи с этим потребовалась полная переработка всех обработчиков, чтобы обеспечить их корректную работу с новой библиотекой.

В рамках проекта был разработан универсальный шаблон для создания и корректного функционирования обработчиков тикетов. Этот шаблон обеспечивает стандартизированный подход к обработке различных статусов и позволяет легко масштабировать систему под новые требования.

Особое внимание уделялось обработчику open_access_request_pending, который реализован на основе данного шаблона. Его основная задача — периодически проверять тикеты со статусом 1002 в заданные промежутки времени и выполнять необходимые действия для их дальнейшей обработки.

Периодически проверять тикеты со статусом 1002 в заданные промежутки времени.

В случае отсутствия изменений по тикету в течение определённого времени отправлять уведомление ответственному лицу.

Если по тикету не происходит никаких действий после ограниченного количества попыток, автоматически переводить его в статус access_request_closed.

По аналогичной логике были доработаны и остальные обработчики, что позволило повысить надежность и автоматизацию процесса обработки тикетов.

Создание обработчиков в системе управления заявками является ключевым элементом автоматизации процессов обработки обращений пользователей. В рамках данного проекта разработан бот на основе протокола XMPP (Jabber), который предназначен для работы с заявками в статусе «ожидание решения». Бот принимает входящие сообщения от других компонентов системы, взаимодействует с API для получения и обновления информации о заявках, а также управляет их жизненным циклом. Для повышения производительности и надежности обработки используется многопоточный режим, позволяющий одновременно обрабатывать несколько заявок и оперативно реагировать на изменения.

Архитектура бота включает несколько основных компонентов: базовый класс для работы с XMPP, класс для взаимодействия с API системы заявок, а также служебные модули для корректного завершения работы и управления потоками. Основной цикл обработки предусматривает регулярное получение актуального списка заявок, фильтрацию и параллельную обработку с использованием пула потоков. Такой подход обеспечивает стабильность и масштабируемость системы, позволяя эффективно интегрировать бота в существующую инфраструктуру и улучшать качество обслуживания пользователей за счет своевременного реагирования на поступающие заявки.

2.1.7 Реализация дополнительных моделей

Реализация дополнительных моделей в системе образования представляет собой важный этап модернизации учебного процесса, направленный на расширение возможностей индивидуализации и повышения качества образовательных услуг. В современных условиях особое внимание уделяется сетевым формам реализации дополнительных программ, которые обеспечивают гибкость, вариативность и доступность обучения за счет использования дистанционных и электронных технологий. Такие модели позволяют интегрировать усилия различных образовательных организаций,

формируя модульные и разноуровневые программы, что способствует созданию персонализированных образовательных маршрутов и удовлетворению разнообразных потребностей обучающихся. Внедрение дополнительных моделей способствует не только развитию компетенций студентов, но и повышению эффективности управления образовательным процессом, что актуально для современного высшего образования.

2.1.8 Разработка API для корректного функционирования мобильного приложения

Разработка API для корректного функционирования мобильного приложения является ключевым этапом создания современной клиент-серверной архитектуры. API (Application Programming Interface) обеспечивает надежное и стандартизированное взаимодействие между мобильным приложением и серверной частью, реализованной, например, на Django с использованием Django REST Framework. Такой подход позволяет мобильному приложению получать необходимые данные, отправлять запросы на изменение информации и получать ответы в формате JSON, что обеспечивает высокую скорость и удобство обмена данными.

В процессе разработки API важно продумать структуру данных, модели и маршруты, которые будут обрабатывать запросы клиентов. Использование REST архитектуры позволяет применять стандартные HTTP-методы (GET, POST, PUT, DELETE), что упрощает интеграцию и делает API универсальным для различных платформ. Для удобства и безопасности реализуются механизмы аутентификации и авторизации, а также функции фильтрации, пагинации и сортировки данных.

Кроме того, создание API предусматривает тщательное тестирование и документирование интерфейсов, что облегчает дальнейшую поддержку и развитие мобильного приложения. В итоге, правильно спроектированный и реализованный API становится надежным фундаментом для стабильной

работы мобильного клиента, обеспечивая плавный обмен данными и высокую отзывчивость системы.

Таким образом, разработка API на базе Django и Django REST Framework является эффективным решением для обеспечения корректного функционирования мобильного приложения, позволяя создавать масштабируемые и легко поддерживаемые сервисы

2.1.9 Развертывание виртуальной машины с решением

Развертывание виртуальной машины с решением является важным этапом внедрения программного продукта, обеспечивающим его стабильную и эффективную работу в заданной инфраструктуре. В процессе развертывания создается виртуальная среда, максимально приближенная к реальным условиям эксплуатации, что позволяет проводить тестирование и последующую эксплуатацию решения без риска для основной системы.

Виртуальная машина конфигурируется с учетом технических требований приложения, включая параметры процессора, объем оперативной памяти, дисковое пространство и сетевые настройки. На нее устанавливается операционная система, а затем – необходимое программное обеспечение и зависимости, обеспечивающие корректную работу разработанного решения. Такой подход позволяет быстро масштабировать систему, легко восстанавливать работоспособность и обеспечивать изоляцию от других сервисов.

Кроме того, развертывание виртуальной машины автоматизируется с помощью скриптов и инструментов управления конфигурациями, что снижает вероятность ошибок и ускоряет процесс внедрения. Это особенно важно при обновлениях и переносе решения на новые серверы или облачные платформы.

Таким образом, развертывание виртуальной машины с решением обеспечивает надежную, гибкую и управляемую среду для работы

программного продукта, что является ключевым фактором успешной реализации проекта и дальнейшего его сопровождения.

2.2 Тестирование

В рамках разработки нового функционала для сервиса "Друг-спортсмен" в ShariX Open было проведено тестирование, направленное на проверку удобства использования и стабильности работы. Тестирование включало в себя как автоматические тесты, так и ручное тестирование с участием реальных пользователей. В процессе были выявлены небольшие ошибки в интерфейсе, такие как некорректное отображение кнопок на мобильных устройствах и задержки при загрузке профилей спортсменов.

Особое внимание в работе было уделено воспроизводимой инсталляции как ключевому элементу современного художественного пространства. Такая инсталляция представляет собой многослойный синтез визуальных и аудиовизуальных компонентов, способных не только передавать замысел автора, но и активно взаимодействовать с восприятием зрителя. Воспроизводимость инсталляции обеспечивает её многократное воспроизведение в различных контекстах, сохраняя при этом целостность и уникальность художественного высказывания, что открывает новые возможности для исследования и интерпретации искусства в эпоху цифровых технологий.

Postman, благодаря которому было проведено тестирование API показан в приложении 7.

На основе полученных данных команда разработчиков внесла необходимые правки. Были оптимизированы запросы к серверу, что значительно ускорило загрузку данных, а также исправлены визуальные недочеты, улучшив адаптацию интерфейса под разные устройства. Дополнительно были доработаны функции поиска и фильтрации, чтобы пользователи могли быстрее находить подходящих спортсменов.

После внесения изменений тестирование было повторено, и результаты показали значительное улучшение производительности и удобства использования. Пользователи отметили, что взаимодействие с сервисом стало более интуитивным и быстрым. Таким образом, проведенное тестирование позволило не только устранить ошибки, но и повысить общее качество сервиса, что важно для дальнейшего развития платформы.

2.3 Документирование и локализация

В процессе разработки нового функционала для сервиса "Друг-спортсмен" в ShariX Open было уделено внимание документированию кода и локализации интерфейса. Каждый модуль и функция сопровождались подробными комментариями, что упрощает дальнейшую поддержку и развитие проекта. Документация также включает описание архитектуры, API и инструкции для разработчиков, что позволяет новым членам команды быстрее вникать в проект.

Локализация интерфейса была выполнена для поддержки нескольких языков, что делает сервис доступным для международной аудитории. Все текстовые элементы были вынесены в отдельные файлы ресурсов, что упрощает добавление новых языков и редактирование существующих. Это особенно важно для пользователей, которые предпочитают взаимодействовать с платформой на родном языке.

2.4 Вывод

В результате проведенной работы по разработке, тестированию и документированию нового функционала для сервиса "Друг-спортсмен" в ShariX Open, компания высоко оценила проделанные усилия. Решение было признано инновационным и соответствующим современным требованиям пользователей, что подтвердило его актуальность и востребованность. Руководство компании отметило, что новый функционал не только расширяет возможности платформы, но и укрепляет её позиции на рынке.

Особое внимание было уделено удобству использования и стабильности работы сервиса. После внесения правок, выявленных в ходе тестирования, платформа стала более отзывчивой и интуитивно понятной. Это положительно сказалось на обратной связи от пользователей, которые уже начали активно использовать новый функционал для поиска профессиональных спортсменов и получения персональных тренировок.

Компания также оценила качество документирования и локализации, что значительно упрощает дальнейшую поддержку и масштабирование сервиса. Наличие подробной документации и поддержка нескольких языков делают платформу привлекательной для международной аудитории, что открывает новые перспективы для роста.

Учитывая все преимущества и положительные отзывы, руководство компании приняло решение внедрить новый функционал в основную версию ShariX Open. Это решение подчеркивает уверенность в качестве разработки и её потенциале для привлечения новых пользователей и улучшения взаимодействия с существующими.

Компания отмечает, что исходная поставленная задача была объемной, и, несмотря на то, что запланированный функционал реализован не полностью или требует доработки перед выпуском системы для пользователей, компания признает проделанные доработки ShariX Open важными и существенными, влияющими на работоспособность шаблонного решения значительным образом, и надежной основой для построения новых сервисов.

Разработка сервиса Друг-спортсмен будет продолжаться, все внесенные изменения значительны и принимаются компанией для дальнейшей разработки решения.

ГЛАВА 3 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-СЕРВИСА

Технико-экономическое обоснование (ТЭО) – это раздел, в котором анализируется целесообразность разработки программного обеспечения (ПО). В рамках ТЭО оцениваются затраты, прогнозируются результаты и определяется экономический эффект от использования приложения и связанных с ним технологий [9].

3.1 Расчет затрат функционирования без использования ИС

Для оценки затрат компании на функционирование при обслуживании 1000 клиентов без использования ИС необходимо учитывать затраты на персонал, аренду помещений, техническое обслуживание оборудования, маркетинг, юридические услуги и другие расходы.

Приведем примерную оценку затрат на персонал, приведенную в таблице 3, которые могут потребоваться при обслуживании 1000 клиентов.

Таблица 3 – Расчет затрат функционирования сервиса без информационной системы

Сотрудник	**Количество**	**Оклад**	**Стоимость**
Сотрудник техподдержки	40	39 000,00 ₽	1 560 000,00 ₽
Линейный менеджер	5	50 000,00 ₽	250 000,00 ₽
Главный бухгалтер	1	95 000,00 ₽	95 000,00 ₽
Бухгалтер	1	60 000,00 ₽	60 000,00 ₽
Офис-менеджер	1	75 000,00 ₽	75 000,00 ₽
Директор	1	100 000,00 ₽	100 000,00 ₽
Юрист	1	55 000,00 ₽	55 000,00 ₽
Площадь офиса	250	1 000,00 ₽	250 000,00 ₽
Итого	300		2 445 000,00 ₽

Таким образом, затраты компании на функционирования сервиса «Друг-спортсмен» без ИС составляют 2 455 000 рублей в месяц.

3.2 Расчет затрат на разработку

Для оценки затрат компании на разработку информационной системы, нужно учитывать количество специалистов, их оклады и другие факторы.

Исходя из вышеперечисленного, примерный расчет затрат на разработку сервиса представлен в таблице 4.

Таблица 4 – Расчет затрат на разработку сервиса

Сотрудник	**Кол-во**	**Оклад**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Время**	**Стоимость**
Сисадмин	2	50 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	1 000 000,00 ₽
Middle разработчик	1	90 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	900 000,00 ₽
Junior разработчик	2	35 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	700 000,00 ₽
Технический директор	1	95 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	950 000,00 ₽
Архитектор	1	110 000,00 ₽	1	1	1	0	0	0	0	0	0	0	3	330 000,00 ₽
Проджект менеджер	1	85 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	850 000,00 ₽
Дизайнер	1	80 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	800 000,00 ₽
Прочие расходы	1	60 000,00 ₽	1	1	1	1	1	1	1	1	1	1	10	600 000,00 ₽
Итого														6 130 000,00 ₽

Таким образом, затраты компании на разработку сервиса «Друг-спортсмен» без ИС составляют 6 930 000 рублей.

Стоит отметить, что это только приблизительный расчет и конечная сумма может быть скорректирована в зависимости от конкретных условий проекта и рыночных факторов.

3.3 Расчет затрат функционирования с использованием ИС

При анализе затрат на функционирование с использованием информационной системы (ИС), необходимо учитывать расходы на разработку, внедрение и поддержку системы. Однако применение ИС может существенно сократить затраты на персонал, повысить качество обслуживания клиентов и улучшить общую эффективность работы компании. Таким образом, использование ИС способствует оптимизации бизнес-процессов и приводит к достижению значительных экономических выгод.

Приведем примерную оценку затрат на персонал, представленную в таблице 5, которые могут потребоваться при обслуживании 1000 клиентов.

Таблица 5 – Расчет затрат функционирования сервиса с информационной системой

Сотрудник	**Количество**	**Оклад**	**Стоимость**
Сотрудник техподдержки	3	39 000,00 ₽	117 000,00 ₽
Системный администратор	2	50 000,00 ₽	100 000,00 ₽
Линейный менеджер	2	50 000,00 ₽	100 000,00 ₽
Бухгалтер	1	60 000,00 ₽	60 000,00 ₽
Офис менеджер	1	75 000,00 ₽	75 000,00 ₽
Директор	1	100 000,00 ₽	100 000,00 ₽
Юрист	1	55 000,00 ₽	55 000,00 ₽
Площадь офиса	75	1 000,00 ₽	75 000,00 ₽
Технический директор	1	95 000,00 ₽	95 000,00 ₽
Middle разработчик	1	90 000,00 ₽	90 000,00 ₽
Junior разработчик	2	30 000,00 ₽	60 000,00 ₽
Итого	90		927 000,00 ₽

3.4 Срок окупаемости ИС

Ежемесячная экономия денег: 2 445 000 ₽ - 927 000 ₽ = 1 518 000 ₽

На графике 1 показана окупаемость ИС, которая составляет 10 месяцев. Это очень короткий срок, что указывает на высокую эффективность инвестиций.

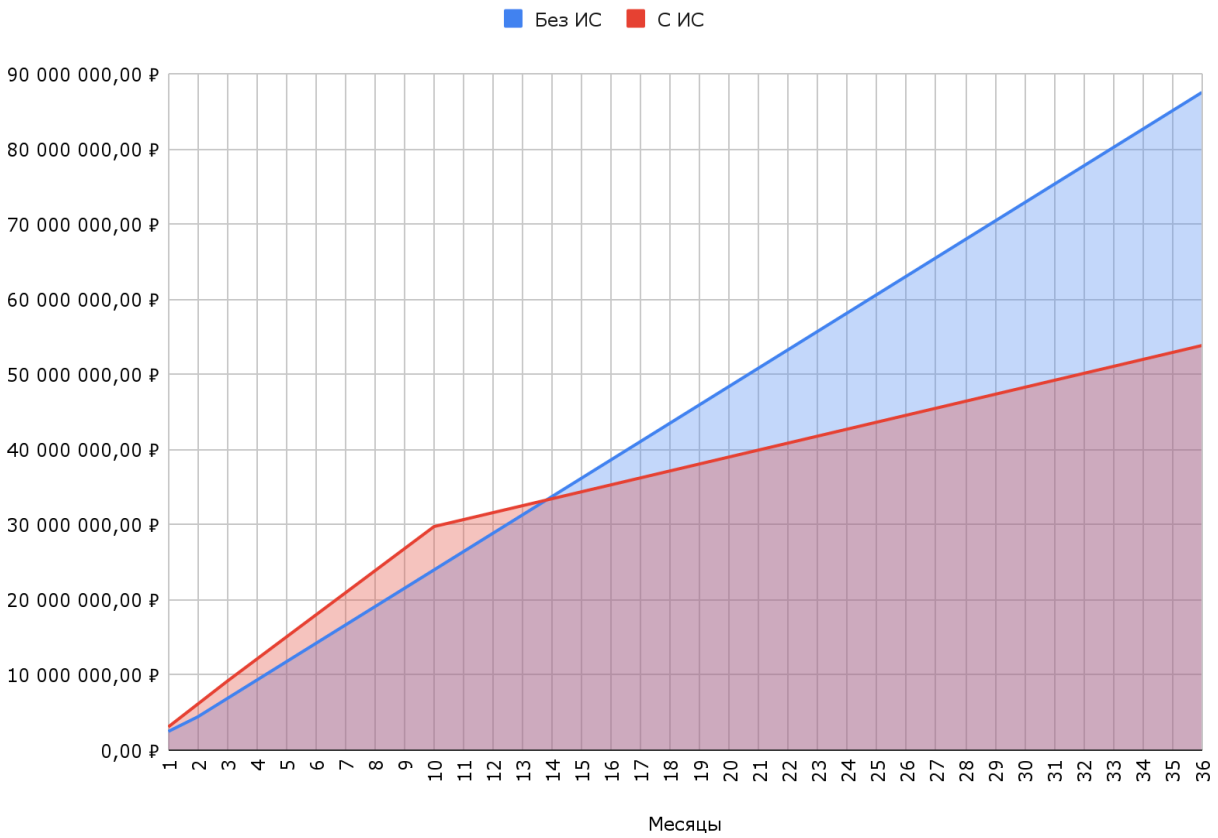


График 1 – Расчет затрат функционирования сервиса с информационной системой

3.5 Экономические выгоды компании

Использование сервиса «Друг-спортсмен» предприятиятием ООО «ШЭРИКС» принесет следующие выгоды:

– Увеличение клиентской базы: Сервис «Друг-спортсмен» будет привлекать новых пользователе. Это позволит компании увеличить свою клиентскую базу и привлечь больше потенциальных клиентов для использования других коммерческих сервисов предприятия.

– Репутационные выгоды и лояльность аудитории: Сервис «Друг-спортсмен» будет способствовать активному продвижению спорта среди широкой аудитории, предоставляя удобное и доступное приложение для спортсменов и любителей спорта. Это поможет вовлечь больше людей в активный образ жизни и укрепить их приверженность к занятиям спортом. Благодаря этому компания ООО «ШЭРИКС» станет активным участником

социальной инициативы, направленной на улучшение общественного здоровья и повышение интереса к спорту.

Такой подход будет способствовать формированию лояльной аудитории, которая станет регулярно пользоваться сервисом, что приведет к увеличению постоянной пользовательской базы. В дальнейшем можно будет отслеживать динамику вовлеченности аудитории с помощью аналитических инструментов, например, путем построения графиков, которые наглядно покажут рост интереса и активности пользователей. Это не только укрепит репутацию компании как социально ответственного предприятия, но и создаст прочную основу для долгосрочного взаимодействия с аудиторией.

– Генерация прибыли: Ожидается, что около 30% клиентов сервиса «Друг-спортсмен» также будут использовать другие коммерческие сервисы предприятия. Это позволит компании увеличить свой доход за счет дополнительных услуг, предоставляемых этими клиентами.

– Экономическая эффективность: Расчеты показывают, что затраты на функционирование сервиса с использованием ИС (927 000) значительно ниже, чем затраты на функционирование без ИС (2 445 000). Использование ИС позволит оптимизировать бизнес-процессы, снизить расходы на персонал и повысить эффективность работы предприятия.

Таким образом, внедрение информационной системы в сервис «Друг-спортсмен» принесет предприятию ООО «ШЭРИКС» репутационные выгоды, увеличит клиентскую базу, привлечет пользователей к другим коммерческим сервисам и обеспечит экономическую эффективность, превышающую затраты на функционирование без ИС.

ГЛАВА 4 ТЕХНИКА БЕЗОПАСНОСТИ

При разработке приложения необходимо соблюдать общие правила техники безопасности.

Проектирование, разработка и использование представленной информационной системы подразумевает работу за компьютером. Для безопасной работы нужно следовать правилам техники безопасности при работе с электрооборудованием (компьютерами, принтерами, интерактивными устройствами). Необходимо регулярно проверять исправность электропроводки, заземление оборудования и избегать работы с неисправной техникой, чтобы обеспечить безопасность при эксплуатации компьютеров и периферийных устройств.

4.1 Общие требования охраны труда

К работе на персональном компьютере (далее – ПК) допускаются лица, прошедшие медицинское освидетельствование, вводный инструктаж, первичный инструктаж, обучение и стажировку на рабочем месте, проверку знаний требований охраны труда, имеющие группу I по электробезопасности. В качестве ПК может использоваться ноутбук.

Требования охраны труда распространяются на работников, в том числе, в случае удаленной (дистанционной) работы.

При работе на ПК работник обязан выполнять технику безопасности.

Выполнять только ту работу, которая определена его должностной (рабочей) инструкцией.

Выполнять правила внутреннего трудового распорядка.

Соблюдать режим труда и отдыха в зависимости от продолжительности, вида и категории трудовой деятельности.

Не употреблять в рабочее время спиртные напитки, не находиться под воздействием запрещенных веществ и препаратов.

Не употреблять пищу и напитки непосредственно на рабочем месте, в помещениях с ПК и офисной техникой, а также, в производственных помещениях, не предназначенных для приема пищи.

Правильно применять средства индивидуальной и коллективной защиты.

Соблюдать требования охраны труда.

Немедленно извещать своего непосредственного или вышестоящего руководителя о любой ситуации, угрожающей жизни и здоровью людей, о каждом несчастном случае, происшедшем на производстве, или об ухудшении состояния своего здоровья, в том числе о проявлении признаков острого профессионального заболевания (отравления).

– Проходить обучение безопасным методам и приемам выполнения работ и оказанию первой помощи пострадавшим на производстве, инструктаж по охране труда, проверку знаний требований охраны труда.

– Проходить обязательные периодические (в течение трудовой деятельности) медицинские осмотры (обследования), а также проходить внеочередные медицинские осмотры (обследования) по направлению работодателя в случаях, предусмотренных Трудовым кодексом и иными федеральными законами.

– Уметь оказывать первую помощь пострадавшим от электрического тока и при других несчастных случаях.

– Уметь применять первичные средства пожаротушения.

При эксплуатации ПК на работника могут оказывать действие следующие опасные и вредные производственные факторы:

– повышенный уровень электромагнитных излучений и статического электричества;

– пониженная ионизация воздуха;

– недостаточная освещенность рабочего места;

– пыль;

– вредные вещества от принтеров и копировальных аппаратов;

– монотонная физическая активность;

– статические физические перегрузки;

– перенапряжение зрительных анализаторов;

– интеллектуальные и эмоциональные нагрузки;

– повышенный уровень шума от большого числа ПК и офисной техники.

Конструкция монитора должна обеспечивать возможность поворота экрана в горизонтальной и вертикальной плоскости с фиксацией в заданном положении для обеспечения фронтального наблюдения изображения. Для ноутбука фронтальное наблюдение экрана обеспечивается соответствующим положением его корпуса и крышки дисплея. Дизайн ПК или ноутбука должен предусматривать окраску корпуса в спокойные мягкие тона с диффузным рассеиванием света. Корпус ПК, клавиатура и другие блоки, и устройства ПК должны иметь матовую поверхность с коэффициентом отражения 0,4 – 0,6 и не иметь блестящих деталей, способных создавать блики.

Конструкция монитора должна предусматривать регулирование яркости и контрастности экрана.

Площадь на одно рабочее место пользователей ноутбуков, а также, ПК с мониторами на базе плоских дискретных экранов (жидкокристаллические, плазменные) – 4,5 $м^2$.

Помещения, где размещаются рабочие места с ПК, должны быть оборудованы защитным заземлением (занулением) в соответствии с техническими требованиями по эксплуатации.

Рабочие места с ПК должны размещаться таким образом, чтобы расстояние от экрана одного монитора до тыла другого было не менее 2 м, а расстояние между боковыми поверхностями мониторов – не менее 1,2 м.

Рабочие столы следует размещать таким образом, чтобы экраны мониторов были ориентированы боковой стороной к световым проемам, чтобы естественный свет падал преимущественно слева.

Оконные проемы в помещениях, где используются ПК, должны быть оборудованы регулируемыми устройствами типа: жалюзи, занавесей, внешних козырьков и др.

Искусственное освещение в помещениях для эксплуатации ПК должно осуществляться системой общего равномерного освещения. В производственных и административно-общественных помещениях, в случаях преимущественной работы с документами, следует применять системы комбинированного освещения (к общему освещению дополнительно устанавливаются светильники местного освещения, предназначенные для освещения зоны расположения документов).

Экран монитора должен находиться от глаз пользователя на расстоянии 600-700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов.

Рабочая мебель для пользователей ПК и компьютерной техникой должна отвечать следующим требованиям:

– высота рабочей поверхности стола должна регулироваться в пределах 680-800 мм;

– при отсутствии такой возможности высота рабочей поверхности стола должна составлять 725 мм;

– рабочий стол должен иметь пространство для ног высотой не менее 600 мм, шириной не менее 500 мм, глубиной на уровне колен – не менее 450 мм и на уровне вытянутых ног – не менее 650 мм;

– рабочий стул (кресло) должен быть подъемно-поворотным, регулируемым по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья, при этом регулировка каждого

параметра должна быть независимой, легко осуществляемой и иметь надежную фиксацию;

– рабочее место должно быть оборудовано подставкой для ног, имеющей ширину не менее 300 мм, глубину не менее 400мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20 градусов; поверхность подставки должна быть рифленой и иметь по переднему краю бортик высотой 10 мм;

– клавиатуру следует располагать на поверхности стола на расстоянии 100-300 мм от края, обращенного к пользователю, или на специальной, регулируемой по высоте рабочей поверхности, отделенной от основной столешницы.

В помещениях, оборудованных ПК, проводится ежедневная влажная уборка и систематическое проветривание после каждого часа работы на ПК.

Женщины со времени установления беременности переводятся на работы, не связанные с использованием ПК, или для них ограничивается время работы с ПК (не более 3-х часов за рабочую смену).

В случаях травмирования или недомогания необходимо прекратить работу, известить об этом руководителя работ и обратиться в медицинское учреждение.

За невыполнение данной инструкции виновные привлекаются к ответственности согласно законодательства Российской Федерации.

4.2 Требования охраны труда перед началом работы

Подготовить рабочее место.

Отрегулировать освещение на рабочем месте, убедиться в отсутствии бликов на экране.

Проверить правильность подключения ПК и оборудования к электросети.

Проверить исправность проводов питания и отсутствие поврежденных и оголенных участков проводов.

Убедиться в наличии заземления системного блока ПК и монитора.

Протереть антистатической салфеткой поверхность экрана монитора.

Проверить правильность установки стола, стула, подставки для ног, пюпитра, угла наклона экрана, положение клавиатуры, положение «мыши» на специальном коврике, при необходимости произвести регулировку рабочего стола и кресла, а также расположение элементов ПК в соответствии с требованиями эргономики и в целях исключения неудобных поз и длительных напряжений тела.

4.3 Требования охраны труда во время работы

Работнику при работе на ПК запрещается:

– прикасаться к задней панели системного блока (процессора) при включенном питании;

– переключать разъемы интерфейсных кабелей периферийных устройств при включенном питании;

– допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и других устройств;

– производить самостоятельное вскрытие и ремонт оборудования;

– работать на компьютере при снятых кожухах;

– отключать оборудование от электросети и выдергивать электрическую «вилку» из розетки, держась за шнур.

Для работы на ПК необходимо подбирать правильную одежду и обувь:

– для снижения электрических и электромагнитных рисков нежелательно использовать в одежде металлические и магнитные предметы и аксессуары;

– для исключения электростатических помех – исключить ношение одежды с повышенным образованием статического электричества;

– для предотвращения механических рисков – запрещено перемещение по производственным помещениям на чрезмерно высоких каблуках, не рекомендуется к ношению одежда с длинными элементами, которые могут попасть в движущиеся механизмы офисной техники.

Во время регламентированных перерывов с целью снижения нервно-эмоционального напряжения, утомления зрительного анализатора, устранения влияния гиподинамии и гипокинезии, предотвращения развития посттетанического утомления выполнять комплексы упражнений.

4.4 Требования охраны труда в аварийных и чрезвычайных ситуациях

Во всех случаях обрыва проводов питания, неисправности заземления и других повреждений, появления гари, немедленно отключить питание и сообщить об аварийной ситуации руководителю.

Не приступать к работе до устранения неисправностей.

При возникновении пожара, задымления, иной чрезвычайной ситуации (далее – ЧС):

– немедленно сообщить по телефону «101» или «112» в МЧС, оповестить работающих, поставить в известность руководителя подразделения, сообщить о возгорании или ЧС на пост охраны;

– открыть запасные выходы из здания, обесточить электропитание, закрыть окна и прикрыть двери;

– в случае возникновения пожара приступить к его тушению первичными средствами пожаротушения, если это не сопряжено с риском для жизни. Организовать встречу пожарной команды;

– покинуть здание и находиться в зоне эвакуации.

При несчастном случае:

– немедленно организовать первую помощь пострадавшему и при необходимости доставку его в медицинскую организацию;

– принять неотложные меры по предотвращению развития аварийной ситуации или иной ЧС и воздействия травмирующих факторов на других лиц;

– сохранить до начала расследования несчастного случая обстановку, какой она была на момент происшествия, если это не угрожает жизни и здоровью других лиц и не ведет к катастрофе, аварии или возникновению иных чрезвычайных обстоятельств, а в случае невозможности ее сохранения;

– зафиксировать сложившуюся обстановку (составить схемы, провести другие мероприятия).

4.5 Требования охраны труда по окончании работы

Отключить питание компьютера.

Привести в порядок рабочее место.

Выполнить упражнения для глаз и пальцев рук на расслабление.

ЗАКЛЮЧЕНИЕ

В ходе выполнения дипломной работы была проведена всесторонняя аналитическая и практическая работа, направленная на решение поставленных задач и достижение целей исследования. Полученные результаты подтвердили актуальность выбранной темы и позволили выявить ключевые закономерности и особенности исследуемого объекта.

В процессе разработки веб-приложения была реализована поддержка воспроизводимых сборок, что обеспечивает стабильность и повторяемость процесса сборки программного продукта на различных этапах разработки и внедрения. Для интеграции с корпоративной инфраструктурой предусмотрена возможность подключения к LDAP-серверу, что упрощает управление пользователями и обеспечивает централизованную аутентификацию и авторизацию. Также была внедрена автоматизация процессов развертывания и тестирования, что позволяет ускорить выпуск обновлений и повысить качество программного обеспечения за счёт минимизации ручных операций и ошибок.

Реализация предложенных рекомендаций и моделей способствует повышению эффективности рассматриваемой системы и открывает перспективы для дальнейших научных и прикладных разработок. Компания ООО «ШЭРИКС» положительно оценила результаты выполненной работы, отметив её соответствие поставленным задачам и требованиям. Разработанное веб-приложение «Друг-спортсмен» будет использоваться в качестве основы для построения действующего решения, что позволит существенно улучшить качество и удобство предоставляемых сервисов. Таким образом, проделанная работа вносит значимый вклад в развитие выбранной области и может быть использована как основа для последующих исследований и практических внедрений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Шеринговые сервисы (sharing-сервисы). — Текст : электронный // ShariX Open Docs: [сайт]. — URL: https://wiki.sharix-app.org/doku.php/sheringovie_servisi (дата обращения: 09.10.2024).

2. ВЦИОМ. Новости: Спорт для всех. — Текст : электронный // ВЦИОМ: [сайт]. — URL: https://wciom.ru/analytical-reviews/analiticheskii-obzor/sport-dlja-vsekh (дата обращения: 10.10.2024).

3. ShariX Open - ShariX. – Текст : электронный // Sharix: [сайт]. – URL: https://git.sharix-app.org/ShariX_Open/ (дата обращения: 10.10.2024).

4. Административная панель для Laravel – какую лучше выбрать в 2025 году?. — Текст : электронный // Рабочие решения: [сайт]. — URL: https://worksolutions.ru/useful/administrativnaya-panel-dlya-laravel-kakuyu-luchshe-vybrat/ (дата обращения: 10.03.2025).

5. Леоненков, А. В. Самоучитель UML 2 / А. В. Леоненков. — Санкт-Петербург: БХВ-Петербург, 2007. — 576 с. (дата обращения: 10.03.2025)

6. Шитов, В. Н. Основы проектирования баз данных: учебное пособие / В. Н. Шитов. — Москва: ИНФРА-М, 2023. — 237 с. (дата обращения: 11.03.2025)

7. Структура БД. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/db/structure (дата обращения: 11.03.2025).

8. ShariX Open Webapp Base. — Текст : электронный // © 2025 ShariX : [сайт]. — URL: https://git.sharix-app.org/ShariX_Open/sharix-open-webapp-base/src/master/README.md (дата обращения: 14.03.2025).

9. Технико-экономическое обоснование. — Текст : электронный // Рувики: Интернет-энциклопедия : [сайт]. — URL: https://ru.ruwiki.ru/wiki/Технико-экономическое_обоснование (дата обращения: 14.03.2025).

10. О ShariX. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/start (дата обращения: 15.03.2025).

11. Инструкция по запуску синхронизируемой части. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/starting_synced_db_instruction (дата обращения: 16.03.2025).

12. Как развернуть сервис ShariX Open. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/admin/instrukcija_po_zapusku_sharix_open (дата обращения: 16.03.2025).

13. Инструкция по настройке сети для функционирования сервиса ShariX_Open. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/admin/instrukcija_po_nastrojke_seti_dlja_sharix_open (дата обращения: 16.03.2025).

14. Как от Open перейти к собственному сервису?. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/instrukcija_po_adaptacii_sharix_open (дата обращения: 17.03.2025).

15. Словари. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/db/slovari_bd (дата обращения: 17.03.2025).

16. Определения. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/db/structure (дата обращения: 17.03.2025).

17. Общая идея обработки заявок. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/ticket_lifetime_schemas (дата обращения: 18.03.2025).

18. Типы пользователей ИС. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/common_user_roles (дата обращения: 18.03.2025).

19. Описание работы с картами. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/map/opisanie_raboty_s_kartami (дата обращения: 19.03.2025).

20. Примеры JSON для обмена данными между системами. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/sharix_open_-_primery_json (дата обращения: 20.03.2025).

21. Рекомендуемая структура репозиториев сервисов на основе ShariX Open. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/repository_structure (дата обращения: 20.03.2025).

22. Общая информация по настройке сервера. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/ejabberd-settings (дата обращения: 20.03.2025).

23. Пояснение про рейтинг. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/tech/dev/rating (дата обращения: 20.03.2025).

24. Подключение к платформе. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/site/join (дата обращения: 20.03.2025).

25. Политика конфиденциальности платформы Sharix. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/doc/policy_conf (дата обращения: 20.03.2025).

26. ГЕНЕРАЛЬНОЕ ПОЛЬЗОВАТЕЛЬСКОЕ СОГЛАШЕНИЕ ShariX. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/doc/usage (дата обращения: 20.03.2025).

27. Контакты ООО "ШЭРИКС". — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/doc/contacts (дата обращения: 21.03.2025).

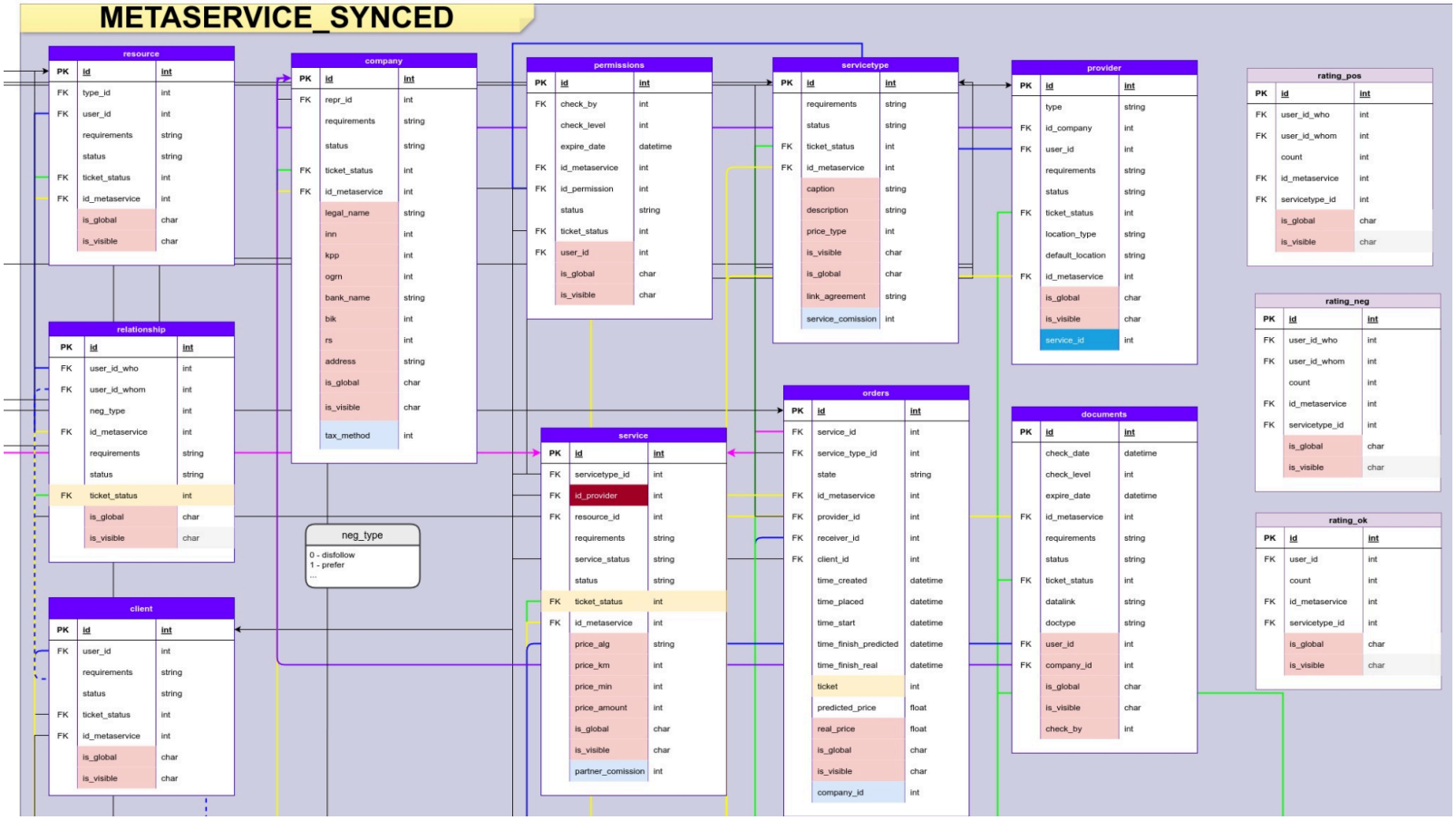
28. Соглашение на присоединение сервиса к платформе ShariX. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/doc/new_service (дата обращения: 21.03.2025).

29. Идеи сервисов для стартапов. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/business/idei_servisov (дата обращения: 22.03.2025).

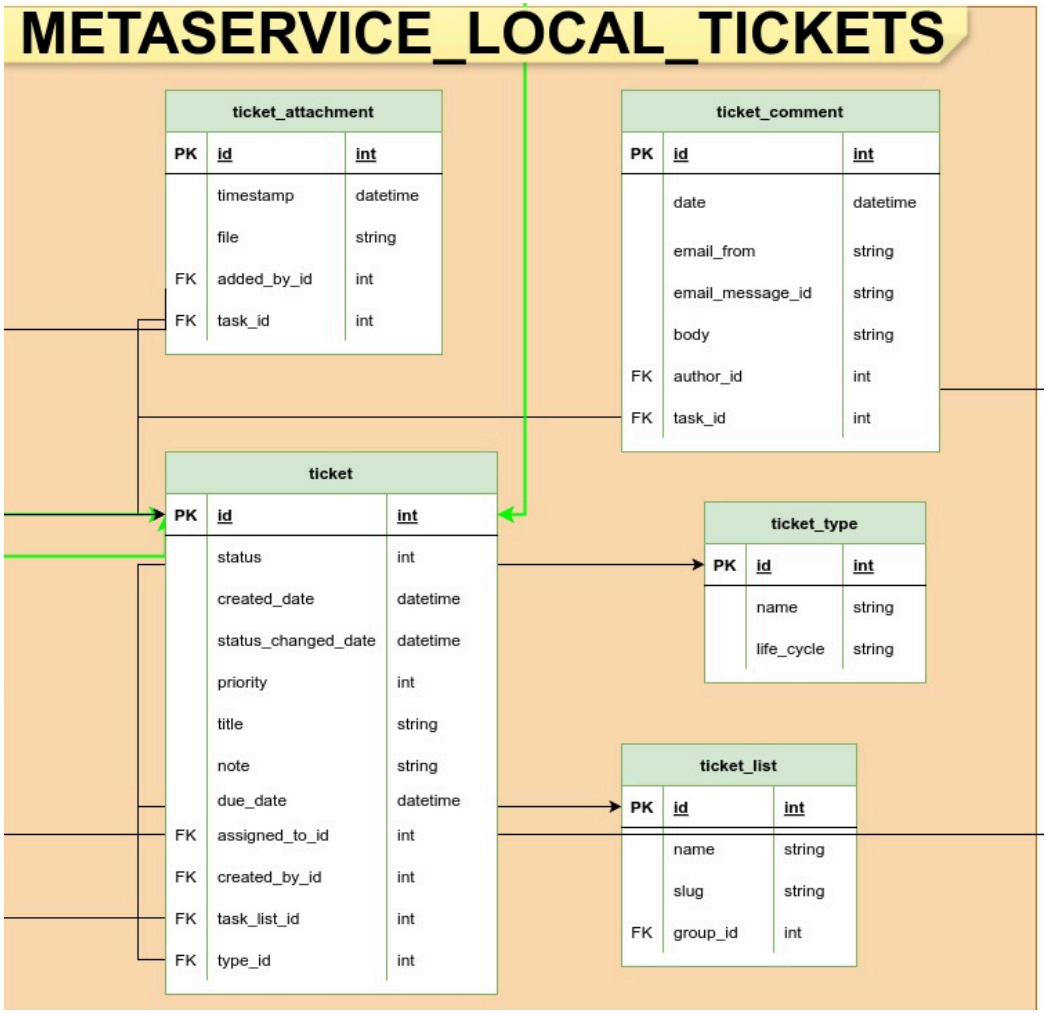
30. Термины и определения. — Текст : электронный // ShariX Open Docs : [сайт]. — URL: https://wiki.sharix-app.org/doku.php/open/doc/terms (дата обращения: 22.03.2025).

ПРИЛОЖЕНИЕ

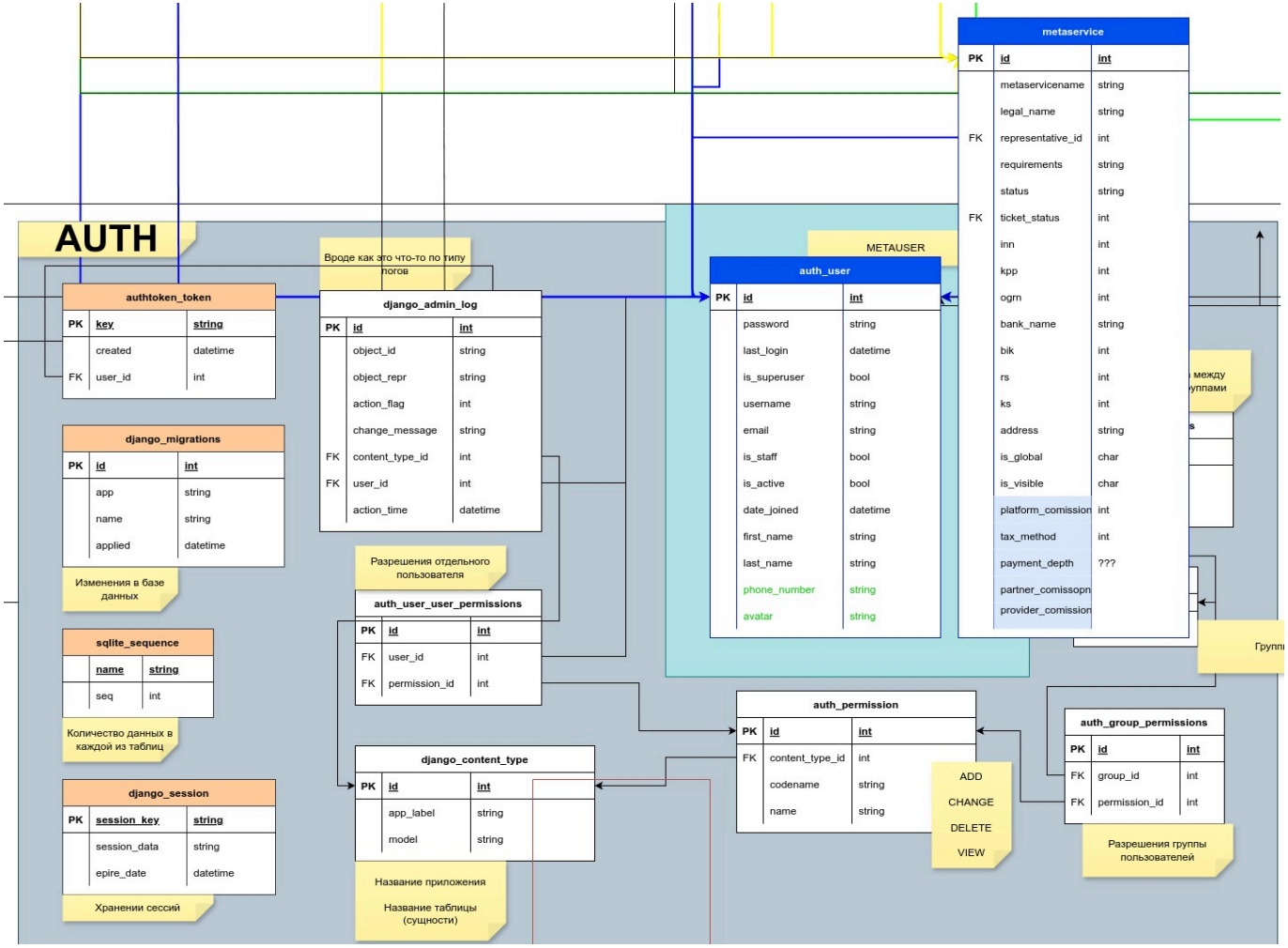
Приложение 1 Схема БД METASERVICE_SYNCED



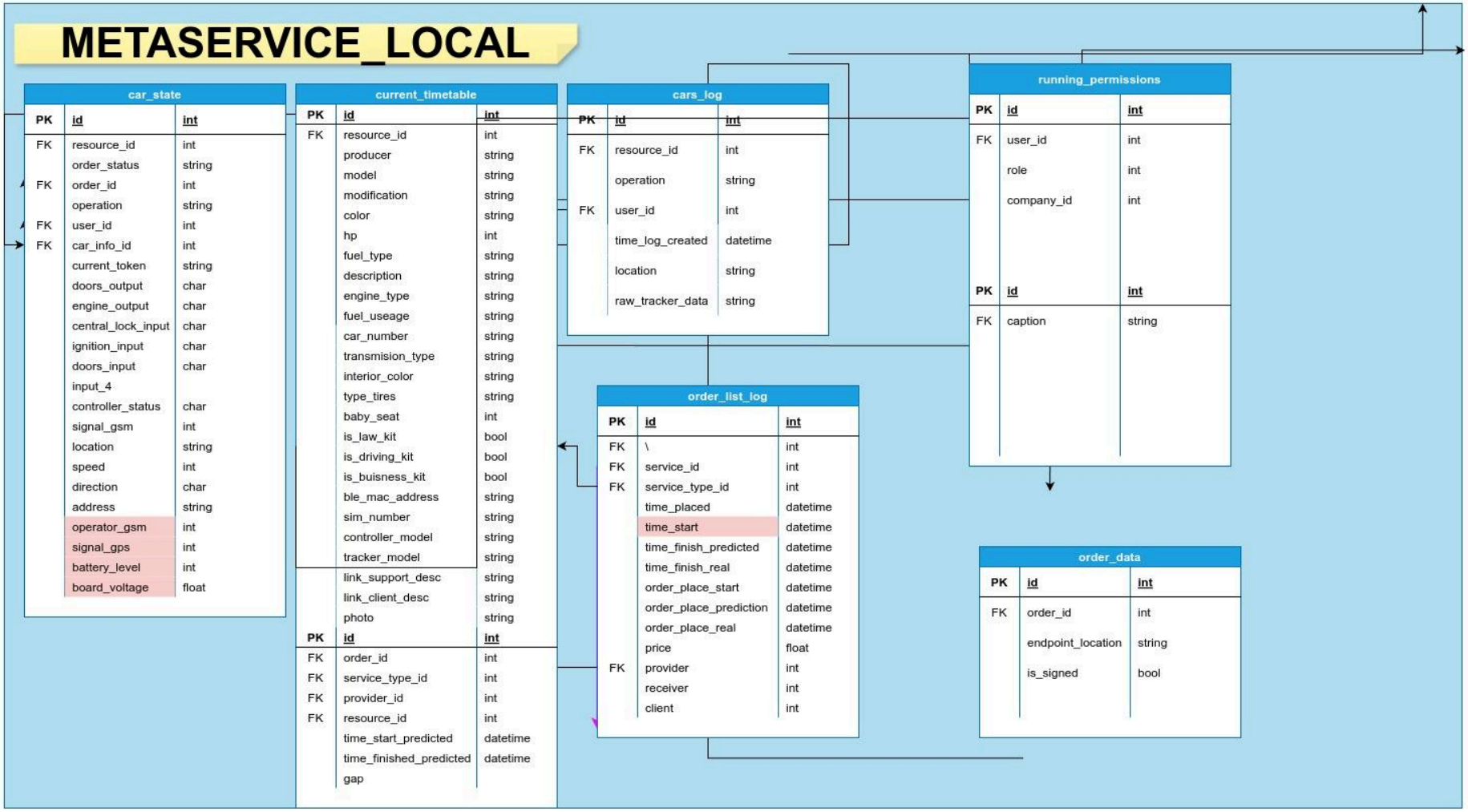
Приложение 2 Схема БД

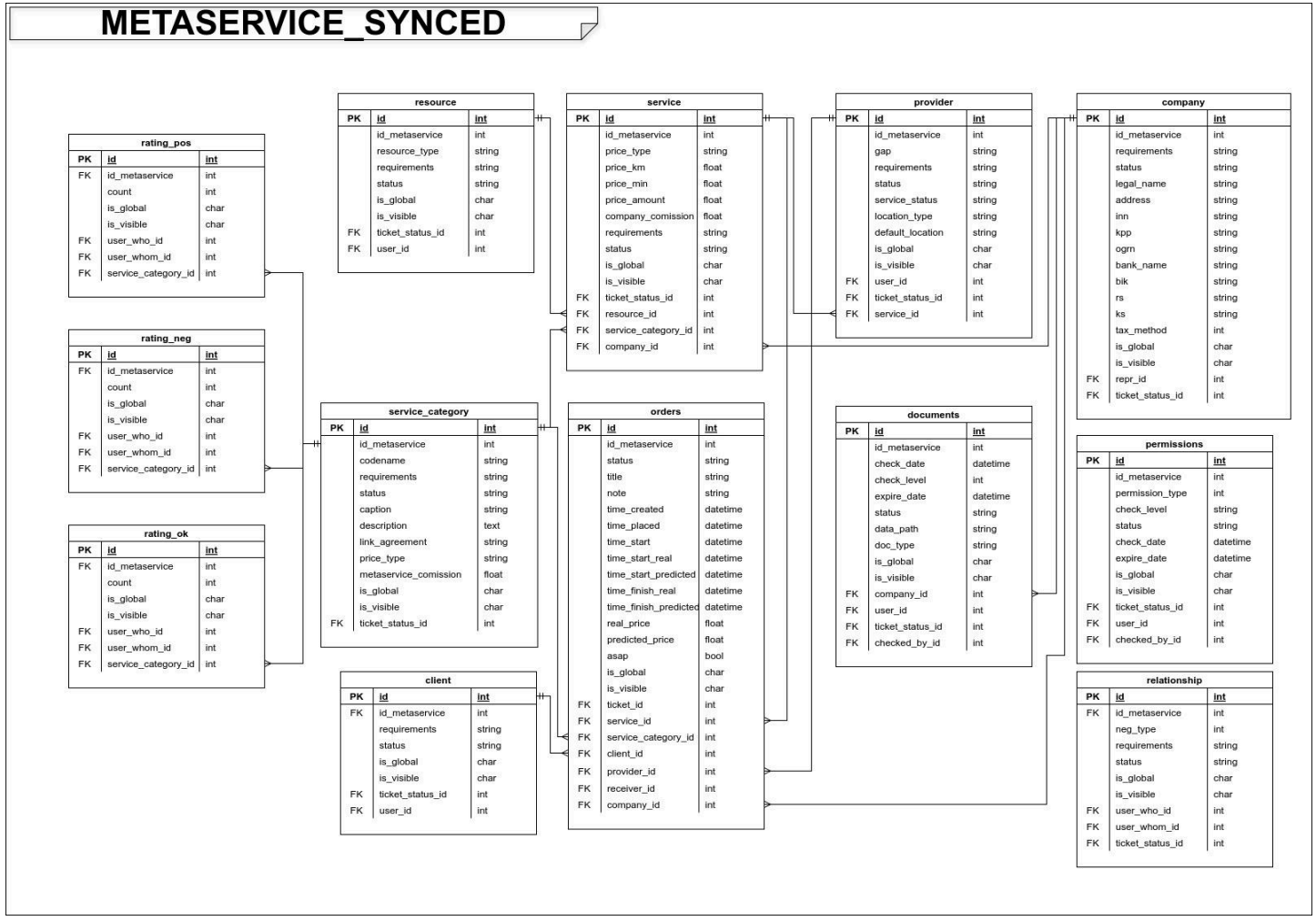


Приложение 3 Схема БД AUTH



Приложение 4 Схема БД METASERVICE_LOCAL



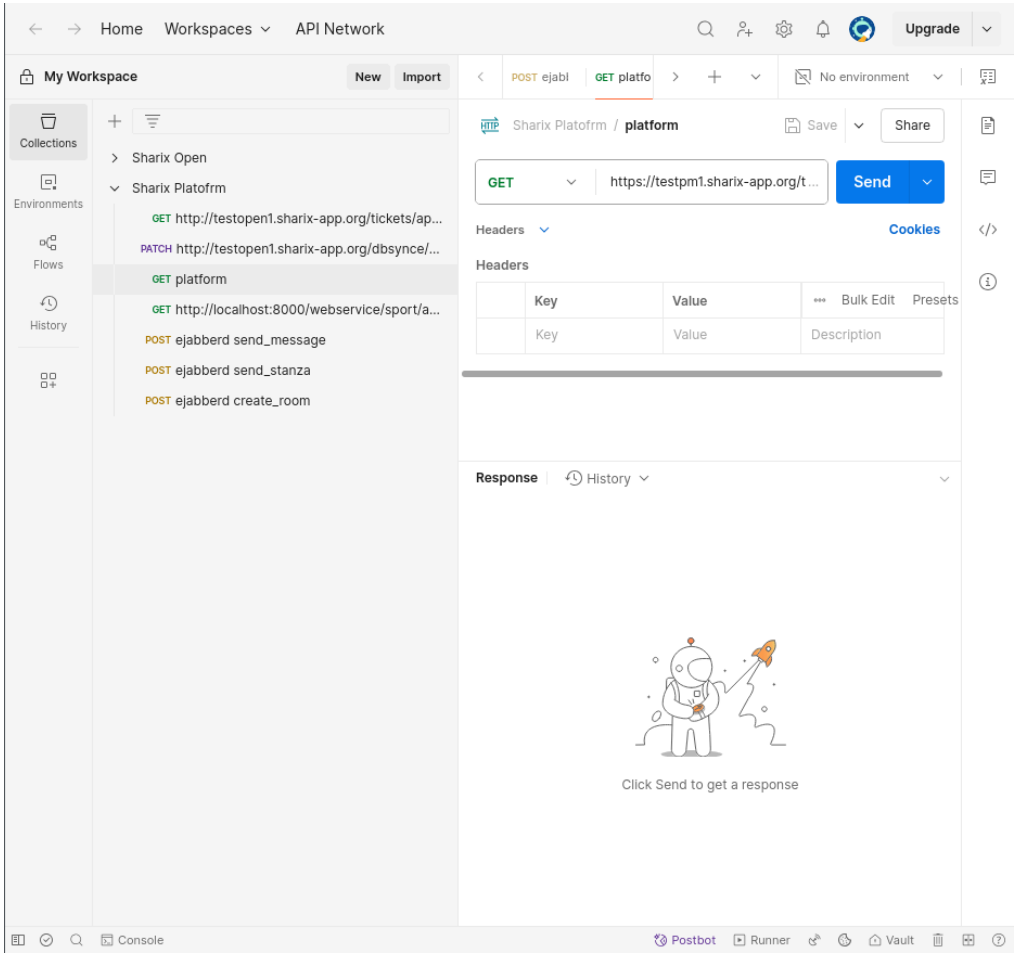

METASERVICE_SYNCED
rating_pos
PK id int
FK id_metaservice int
count int
is_global char
is_visible char
FK user_who_id int
FK user_whom_id int
FK service_category_id int
rating_neg
PK id int
FK id_metaservice int
count int
is_global char
is_visible char
FK user_who_id int
FK user_whom_id int
FK service_category_id int
rating_ok
PK id int
FK id_metaservice int
count int
is_global char
is_visible char
FK user_who_id int
FK user_whom_id int
FK service_category_id int
resource
PK id int
id_metaservice int
resource_type string
requirements string
status string
is_global char
is_visible char
FK ticket_status_id int
FK user_id int
service_category
PK id int
id_metaservice int
codename string
requirements string
status string
caption string
description text
link_agreement string
price_type string
metaservice_comission float
is_global char
is_visible char
FK ticket_status_id int
client
PK id int
FK id_metaservice int
requirements string
status string
is_global char
is_visible char
FK ticket_status_id int
FK user_id int
service
PK id int
id_metaservice int
price_type string
price_km float
price_min float
price_amount float
company_comission float
requirements string
status string
is_global char
is_visible char
FK ticket_status_id int
FK resource_id int
FK service_category_id int
FK company_id int
orders
PK id int
id_metaservice int
status string
title string
note string
time_created datetime
time_placed datetime
time_start datetime
time_start_real datetime
time_start_predicted datetime
time_finish_real datetime
time_finish_predicted datetime
real_price float
predicted_price float
asap bool
is_global char
is_visible char
FK ticket_id int
FK service_id int
FK service_category_id int
FK client_id int
FK provider_id int
FK receiver_id int
FK company_id int
provider
PK id int
id_metaservice int
gap string
requirements string
status string
service_status string
location_type string
default_location string
is_global char
is_visible char
FK user_id int
FK ticket_status_id int
FK service_id int
documents
PK id int
id_metaservice int
check_date datetime
check_level int
expire_date datetime
status string
data_path string
doc_type string
is_global char
is_visible char
FK company_id int
FK user_id int
FK ticket_status_id int
FK checked_by_id int
company
PK id int
id_metaservice int
requirements string
status string
legal_name string
address string
inn string
kpp string
ogrn string
bank_name string
bik string
rs string
ks string
tax_method int
is_global char
is_visible char
FK repr_id int
FK ticket_status_id int
permissions
PK id int
id_metaservice int
permission_type int
check_level string
status string
check_date datetime
expire_date datetime
is_global char
is_visible char
FK ticket_status_id int
FK user_id int
FK checked_by_id int
relationship
PK id int
FK id_metaservice int
neg_type int
requirements string
status string
is_global char
is_visible char
FK user_who_id int
FK user_whom_id int
FK ticket_status_id int

Приложение 6 Скрипт создание init.ldif

```
#DATA
main_init = """# init.ldif
# ROOT-DB
dn: dc=ldap,dc=sharix,dc=ru
objectClass: dcObject
objectClass: organization
dc: ldap
o: Sharix LDAP Server
"""
extra_test_users = """..."""
uid_django_user="""..."""
extra_groups = """..."""
extra_apps = """..."""
handlers = [
    ("platform_access_request_pending",  "ACCREQ pending"),
…
    ("platform_st_request_done",         "STREQ done"),]
groups = [
    ('PLATFORM-ADMIN',       11),
…
    ('PLATFORM-CLIENT',      51),]
groups_dict = dict(groups)
test_users = [
    ('PLATFORM-ADMIN', 'Platform', 'Admin', 1),
    …
    ('PLATFORM-CLIENT', 'Platform', 'Client', 3),]
test_users_dict = {user[0]: user[3] for user in test_users}
ous = [
    ("users" , "Подразделение для пользователей"),
    …]
def get_handlers_txt():
    txt = ""
    for name, display_name in handlers:
        HANDLERS_PASSWORD =
"{ARGON2}$argon2id$v=19$m=65536,t=3,p=1$8kx90bsuQRoLoQ3F4Uh+Pu
$PboqW5EPEfzQlFh3uDeWoXP8rXs7v510fwQgtoA2Lew"
        txt +=f"""dn:
uid={name},ou=apps,dc=ldap,dc=sharix,dc=ru
objectClass: handlerAccount
uid: {name}
cn: {name}
sn: handler
displayName: {display_name}
userPassword: {HANDLERS_PASSWORD}
"""
    return txt
def get_users_txt():
    USERS_PASSWORD =
"{ARGON2}$argon2id$v=19$m=65536,t=3,p=1$8kx90bsuQRoLoQ3F4Uh+Pu
$PboqW5EPEfzQlFh3uDeWoXP8rXs7v510fwQgtoA2Lew"
    txt = ""
    for group_name, place, role, count in test_users:
        for i in range(1, count + 1):
            phone_number=f"{groups_dict[group_name]}0{i}"
            uid = phone_number
            cn = place
            sn = f"{int_to_roman(i)}"
            givenName = role
            displayName = f"{place} {role} {int_to_roman(i)}"
 f"test-{group_name.lower()}-{i}@domain.org"
            mail = "test@sharix-app.org"
            telephoneNumber = phone_number
            user_entry = f"""
dn: uid={uid},ou=users,dc=ldap,dc=sharix,dc=ru
objectClass: sharixAccount
uid: {uid}
cn: {cn}
sn: {sn}
userPassword: {USERS_PASSWORD}
givenName: {givenName}
displayName: {displayName}
mail: {mail}
jpegPhoto: 0
telephoneNumber: {telephoneNumber}
"""
            txt += user_entry
    return txt
def get_groups_txt():
    txt = ""
    for group, uid in groups:
        txt += f"""
dn: cn={group},ou=groups,dc=ldap,dc=sharix,dc=ru
objectClass: groupOfNames
cn: {group}
description: Group for {group.replace("-", " ").title()}
"""
        count = test_users_dict[group]
        for i in range(1, count + 1):
            user_id=f"{uid}0{i}"
            txt += f"member:
uid={user_id},ou=users,dc=ldap,dc=sharix,dc=ru\n"

    return txt
def get_ou_txt():
    return "\n".join(
        f"""# {desc}
dn: ou={name},dc=ldap,dc=sharix,dc=ru
objectClass: organizationalUnit
ou: {name}
""" for name, desc in ous
    )
def get_group_handlers():
    return """
dn: cn=handlers,ou=apps,dc=ldap,dc=sharix,dc=ru
objectClass: groupOfNames
cn: handlers
description: Group for Handlers
""" + "\n".join([
        f"member:
cn={name},ou=handlers,dc=ldap,dc=sharix,dc=ru"
        for name, _ in handlers
    ])
print(main_init)
print(get_ou_txt())
print(uid_django_user)
print(get_handlers_txt())
print(get_users_txt())
print(extra_test_users)
print(get_groups_txt())
print(extra_groups)
print(get_group_handlers())
print(extra_apps)
```

Приложение 7 Тестирования API в Postman



Дипломная работа выполнена мной самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют соответствующие ссылки на них.

«30» мая 2025 г _____ (_____)
(подпись студента)