

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

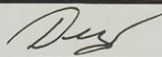
Направление подготовки: 09.03.01 «Информатика и вычислительная техника»  
Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ**  
**«ТЕЛЕФОННЫЙ СПРАВОЧНИК»**

Студент



28.05.2026

Демин Максим Сергеевич  
(ФИО полностью)

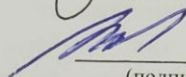
Руководитель



28.05.2026

Р. Н. Федюнин  
(фамилия, инициалы)

Нормоконтролёр

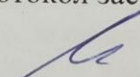


28.05.2026

А. С. Бычков  
(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 22.05.2026 г. № 14)

Заведующий кафедрой



(подпись)

М.А. Митрохин  
(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от

15 июня 2026

№

33

)

Секретарь ГЭК



Е.И. Гурин  
(подпись)

(фамилия, инициалы)

Пенза 2026

Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет  
вычислительной техники

Кафедра  
«Вычислительная техника»

«Утверждаю»  
Заведующий кафедрой  
М.А. Митрохин  
«07» 04 2026 г.

### ЗАДАНИЕ НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Демин Максим Сергеевич гр. 22ВВП1 факультета вычислительной техники

2. Тема работы: Программное обеспечение «Телефонный справочник»

Тема утверждена приказом ПГУ № 347/о от "07" апреля 2026 г.

3. Руководитель бакалаврской работы к. т. н., доцент кафедры ВТ ПГУ Федюнин Р. Н.

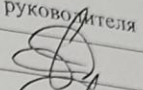
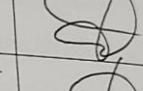
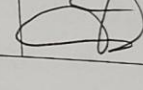
4. Задание на работу (назначение разработки, исходные данные и т.п.)

В рамках выпускной квалификационной работы необходимо разработать программное обеспечение «Телефонный справочник». Программа должна содержать следующий функционал для пользователя: регистрация и авторизация пользователей, поиск сотрудников, возможность просмотра карточки сотрудника, организационную структуру, отображение организаций на карте. Функционал администратора должен предоставлять возможность: создавать, редактировать и удалять сведения пользователей, поддерживать разграничение доступа между ролями, предоставлять возможность импорта и экспорта данных пользователей, ведения журнала аудита действий, управления учетными записями, ролями и интеграциями.

5. Перечень подлежащих разработке вопросов

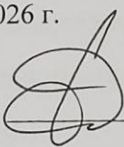
Требуется произвести анализ предметной области и изучить существующие аналоги телефонных справочников, сформулировать требования к разрабатываемой программе. Необходимо выбрать способы проектирования и разработки приложений. Требуется спроектировать структуру базы данных, определить основные сущности и связи БД, разработать серверную часть приложения, механизмы аутентификации, авторизации и разграничения доступа данных. Необходимо разработать клиентскую часть приложения и реализовать страницы авторизации, поиска, карточки сотрудника, организационной структуры, интерактивной карты, профиля пользователя, возможность импорта и экспорта данных, ведение журнала аудита и провести тестирование функций.

6. Календарный график выполнения работы

| № п/п | Наименование этапов работы                                    | Объем работы | Срок выполнения         | Подпись руководителя                                                                |
|-------|---------------------------------------------------------------|--------------|-------------------------|-------------------------------------------------------------------------------------|
| 1     | Изучение предметной области                                   | 4            | 07.04.2026              |  |
| 2     | Анализ требований                                             | 20           | 07.04.2026 – 10.04.2026 |  |
| 3     | Выбор технологий, применяемых при проектировании и разработке | 16           | 11.04.2026 – 14.04.2026 |  |
| 4     | Описание алгоритмов реализации                                | 24           | 15.04.2026 – 19.04.2026 |  |
| 5     | Разработка программного обеспечения «Телефонный справочник»   | 75           | 20.04.2026 – 04.05.2026 |  |
| 6     | Оформление пояснительной записки                              | 40           | 05.05.2026 – 20.05.2026 |  |

Дата выдачи задания "07" апреля 2026 г.

Руководитель бакалаврской работы



Федюнин Р. Н.

(подпись, дата)

Задание к исполнению принял студент



Демин М. С.

(подпись, дата)

Работу к защите допустить  
Декан факультета



09.06.2026

Фионова Л.Р.

(подпись, дата)

### Реферат

Работа содержит 162 л., 88 рисунков, 24 таблицы, 32 источника, 3 приложения.

JAVA, SPRING BOOT, REACT, TYPESCRIPT, POSTGRESQL, REST API, WEB-ПРИЛОЖЕНИЕ, SPRING DATA JPA, JWT, LEAFLET

Цель работы – разработать корпоративную информационную систему «Единый справочник контактов».

Система должна обеспечивать централизованное хранение сведений об организациях, структурных подразделениях, приемных и сотрудниках, позволять пользователям выполнять поиск сотрудников по контактными и организационным данным, просматривать карточки сотрудников, работать с организационной структурой и отображать организации на интерактивной карте. Также система должна предоставлять разграничение доступа по ролям администратора, руководителя и сотрудника, поддерживать импорт и экспорт данных, ведение журнала аудита, управление справочниками, пользователями и API-ключами для интеграции с внешними системами.

Результатом проделанной работы является информационная система «Единый справочник контактов», удовлетворяющая поставленной задаче. Система спроектирована, разработана и протестирована для централизованного хранения, поиска и сопровождения контактной информации внутри организации. При разработке были использованы язык программирования Java 21, фреймворк Spring Boot, Spring Web MVC, Spring Security, Spring Data JPA, СУБД PostgreSQL, а также React, TypeScript, Vite, Ant Design, React Router, TanStack Query, Axios, Zustand, Leaflet, Apache POI и OpenPDF.

|          |      |                |                                                                                     |          |
|----------|------|----------------|-------------------------------------------------------------------------------------|----------|
| Изм.     | Лист | № докум.       | Подпись                                                                             | Дата     |
| Разраб.  |      | Демин М. С.    |  | 28.05.24 |
| Провер.  |      | Федюнин Р. Н.  |  | 28.05.24 |
| Н. Контр |      | Бычков А. С.   |  | 28.05.24 |
| Утверд.  |      | Митрохин М. А. |  | 28.05.24 |

ПГУ1.090301.08.001 ПЗ

Программное обеспечение  
«Телефонный справочник»

Пояснительная записка

|                            |      |        |
|----------------------------|------|--------|
| Лит.                       | Лист | Листов |
|                            | 2    | 126    |
| ФВТ, каф. ВТ<br>гр. 22ВВП1 |      |        |

## Содержание

|                                                                 |    |
|-----------------------------------------------------------------|----|
| Перечень сокращений и обозначений .....                         | 6  |
| Введение.....                                                   | 8  |
| 1 Программное обеспечение «Телефонный справочник» .....         | 9  |
| 1.1 Анализ предметной области.....                              | 9  |
| 1.1.1 Анализ существующих аналогов.....                         | 11 |
| 1.1.2 Постановка задачи .....                                   | 18 |
| 1.1.3 Функционал приложения .....                               | 19 |
| 1.2 Выбор технологий и средств разработки приложения.....       | 24 |
| 1.2.1 Выбор платформы исполнения.....                           | 24 |
| 1.2.2 Выбор языка программирования для серверной части .....    | 24 |
| 1.2.3 Выбор фреймворка для серверной части.....                 | 26 |
| 1.2.4 Выбор языка и фреймворка для клиентской части.....        | 27 |
| 1.2.5 Выбор системы управления базами данных (СУБД) .....       | 28 |
| 1.2.6 Выбор среды разработки .....                              | 29 |
| 1.2.7 Выбор архитектурного паттерна проектирования.....         | 31 |
| 1.2.8 Итоговый стек технологий .....                            | 32 |
| 1.3 Реализация приложения.....                                  | 32 |
| 1.3.1 Описание работы приложения.....                           | 33 |
| 1.3.2 Структура базы данных .....                               | 37 |
| 1.3.3 Описание компонентов.....                                 | 44 |
| 1.3.4 Основные алгоритмы взаимодействия клиента и сервера ..... | 58 |
| 1.4 Тестирование .....                                          | 72 |
| 1.4.1 Выбор функций для составления тест-кейсов .....           | 72 |
| 1.4.2 Авторизация пользователя .....                            | 74 |

|                                                              |     |
|--------------------------------------------------------------|-----|
| 1.4.3 Поиск сотрудников и просмотр карточки сотрудника.....  | 75  |
| 1.4.4 Создание и редактирование сотрудников.....             | 77  |
| 1.4.5 Профиль пользователя и организационная структура ..... | 79  |
| 1.4.6 Импорт и экспорт данных .....                          | 81  |
| 1.4.7 Тестирование функционала администратора.....           | 83  |
| 1.5 Руководство пользователя.....                            | 87  |
| 1.5.1 Результат работы приложения.....                       | 88  |
| 1.5.2 Роли пользователей и разграничение доступа .....       | 90  |
| 1.5.3 Авторизация и завершение работы .....                  | 90  |
| 1.5.4 Поиск сотрудников .....                                | 92  |
| 1.5.5 Карточка сотрудника .....                              | 94  |
| 1.5.6 Создание, редактирование и удаление сотрудников.....   | 96  |
| 1.5.7 Организационная структура.....                         | 99  |
| 1.5.8 Карта организаций .....                                | 101 |
| 1.5.9 Профиль пользователя.....                              | 102 |
| 1.5.10 Импорт и экспорт данных .....                         | 105 |
| 1.5.11 Управление пользователями .....                       | 107 |
| 1.5.12 Управление справочниками .....                        | 109 |
| 1.5.13 Журнал аудита.....                                    | 112 |
| 1.5.14 Интеграции и API-ключи .....                          | 115 |
| 2 Безопасность жизнедеятельности.....                        | 118 |
| Заключение .....                                             | 122 |
| Список использованных источников .....                       | 123 |
| Приложение А .....                                           | 127 |
| Приложение А.1 – Диаграмма прецедентов.....                  | 128 |

|                                                      |     |
|------------------------------------------------------|-----|
| Приложение А.2 – Диаграмма последовательностей ..... | 129 |
| Приложение А.3 – Диаграмма развертывания .....       | 130 |
| Приложение А.4 – Диаграмма компонентов .....         | 131 |
| Приложение Б .....                                   | 132 |
| Приложение В.....                                    | 157 |

## Перечень сокращений и обозначений

В этой работе используются следующие сокращения и обозначения

БД – база данных

ИС – информационная система

ИС «ЕСК» – информационная система «Единый справочник контактов»

ПК – персональный компьютер

ПО – программное обеспечение

РФ – Российская Федерация

СУБД – система управления базой данных

ФИО – фамилия, имя, отчество

API (Application Programming Interface) – программный интерфейс приложения

AOP (Aspect-Oriented Programming) – аспектно-ориентированное программирование

CSS (Cascading Style Sheets) – каскадные таблицы стилей

DTO (Data Transfer Object) – объект передачи данных

HTML (HyperText Markup Language) – язык гипертекстовой разметки

HTTP (HyperText Transfer Protocol) – протокол передачи гипертекста

IP (Internet Protocol) – протокол межсетевого взаимодействия

JPA (Java Persistence API) – спецификация Java для работы с данными в реляционных базах данных

JSON (JavaScript Object Notation) – текстовый формат обмена структурированными данными

JWT (JSON Web Token) – открытый стандарт для передачи данных авторизации в виде токена

MVC (Model-View-Controller) – архитектурный шаблон разделения приложения на модель, представление и контроллер

ORM (Object-Relational Mapping) – технология объектно-реляционного отображения

PDF (Portable Document Format) – формат электронных документов

REST (Representational State Transfer) – архитектурный стиль взаимодействия клиентских и серверных приложений

SPA (Single Page Application) – одностраничное приложение

SQL (Structured Query Language) – язык структурированных запросов

UI (User Interface) – пользовательский интерфейс

UML (Unified Modeling Language) – унифицированный язык моделирования

URL (Uniform Resource Locator) – унифицированный указатель ресурса

XML (Extensible Markup Language) – расширяемый язык разметки

XLSX – формат файлов электронных таблиц Microsoft Excel

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <b>ПГУ1.090301.08.001 ПЗ</b> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                              | 7    |

## Введение

В современном мире процессы государственного управления и внутреннего документооборота всё активнее переходят в цифровую среду, особое значение приобретает оперативный доступ к актуальной и достоверной информации. Для организаций с разветвленной структурой важную роль играет не только хранение официальных данных, но и возможность быстро находить контактную информацию сотрудников и подразделений.

При отсутствии единого централизованного справочника контактные данные часто хранятся разрозненно, в отдельных файлах, локальных таблицах, устаревших списках или внутренних документах. Это приводит к дублированию информации, снижению её актуальности, увеличению времени поиска, а также к ошибкам.

Внедрение информационной системы «Единый справочник контактов» является логичным шагом для повышения эффективности работы организации. Информационная система «Единый справочник контактов» дополняет традиционные способы ведения телефонных справочников, предоставляя пользователям возможность быстро находить сотрудников, просматривать их контактные карточки, работать с организационной структурой и отображать организации на интерактивной карте. Кроме того, система поддерживает разграничение доступа по ролям, импорт и экспорт данных, ведение журнала аудита, управление справочниками и интеграцию с внешними системами через API-ключи.

Таким образом, внедрение электронного справочника контактов позволяет повысить актуальность контактной информации, сократить время поиска необходимых сведений. Централизация данных и автоматизация процессов сопровождения справочника способствуют более эффективной организации работы сотрудников и снижению количества ошибок.

В связи с вышеизложенным, разработка информационной системы «Единый справочник контактов» является актуальной темой для дипломного проекта.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
|      |      |          |         |      |                       | 8    |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

# 1 Программное обеспечение «Телефонный справочник»

## 1.1 Анализ предметной области

Перед разработкой информационной системы «Единый справочник контактов» (далее – ИС «ЕСК») необходимо провести всесторонний анализ предметной области, чтобы определить ключевые потребности сотрудников организации, а также выявить особенности и проблемы существующей модели внутренних организационных коммуникаций в органах государственной власти.

Рассмотрим современное состояние информатизации кадровых и административных процессов в государственных органах, а также существующие ограничения в оперативном доступе к актуальным контактным данным коллег внутри одного ведомства.

Курс на цифровую трансформацию государственного управления закреплён распоряжением Правительства РФ от 16 марта 2024 г. № 637-р. Утверждено актуализированное стратегическое направление в области цифровой трансформации государственного управления. Документ определяет в качестве приоритета развитие технологий, обеспечивающих выполнение основных функций государственных органов, ускорение обработки внутренних запросов и повышение общей эффективности работы [1].

В рамках данной стратегии Правительство РФ одобрило программы цифровой трансформации для десятков федеральных министерств и ведомств. Ключевыми задачами этих программ являются гармонизация информационных ресурсов ведомств, повышение скорости обработки внутренних, в том числе межведомственных, запросов и перевод систем на отечественное программное обеспечение [2].

Создание внутреннего справочника контактов полностью вписывается в логику ведомственных программ цифровой трансформации, направленных на упорядочение внутриорганизационных данных и ускорение операционной деятельности.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 9    |

Для государственных органов проблема разрозненности контактных данных стоит особенно остро. В отличие от бизнес-сектора, где корпоративные порталы и HRM-системы активно внедряются, в государственном секторе уровень цифровизации кадровых и административных процессов остаётся недостаточно качественным [3]. В некоторых ведомствах информация о сотрудниках, их должностях, принадлежности к отделам и контактных данных ведётся бессистемно. Как показывает практика, личные данные сотрудников хранятся в файлах Excel, Word или бумажных папках. Такой подход не только создаёт риски нарушения требований Федерального закона № 152-ФЗ «О персональных данных», но и приводит к тому, что значительная часть рабочего времени уходит на поиск нужного контакта в разрозненных источниках – телефонных книгах, личных записках, сообщениях в мессенджерах или через устный запрос коллег [4].

Отсутствие единого, централизованного и защищённого источника контактных данных внутри организации приводит к:

- дублированию информации и отсутствие гарантий её актуальности;
- повышению временных затрат на поиск нужного сотрудника или отдела;
- невозможности оперативной коммуникации в ситуациях, требующих срочного взаимодействия;
- риску использования устаревших данных, что ведёт к ошибкам и задержкам в работе.

На региональном уровне, в частности в Пензенской области, также взят курс на цифровую зрелость государственного управления. Согласно постановлению Правительства Пензенской области от 12.11.2024 № 870-пП «Об утверждении Положения о Министерстве цифрового развития и связи Пензенской области», в функции министерства входит координация внедрения цифровых технологий в деятельность органов власти региона [5]. В области активно развиваются электронные услуги, внедряются чат-боты. Однако внутренняя цифровизация самих ведомств, включая упорядочение кадровой и контактной информации, остаётся задачей, требующей дополнительного внимания и практических решений.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 10   |

Создание ИС «ЕСК» для одного ведомства призвано решить эти проблемы. Система позволит:

- исключить дублирование информации и снизить временные затраты на поиск коллег и их контактов;
- обеспечить актуальность данных за счёт разграничения зон ответственности;
- повысить оперативность внутриведомственных коммуникаций, предоставив сотрудникам единый достоверный источник контактной информации;
- улучшить общую эффективность работы организации за счёт сокращения непроизводительных потерь рабочего времени.

Актуальность разработки ИС «ЕСК» обусловлена современными тенденциями цифровизации государственного управления, потребностям Пензенской области в повышении эффективности работы государственных служащих и необходимости решения системных проблем разрозненности и неактуальности контактных данных внутри ведомства.

#### 1.1.1 Анализ существующих аналогов

Проблема ведения централизованного и достоверного реестра контактов для оперативного межведомственного взаимодействия является актуальной для государственных структур. В настоящее время на рынке присутствуют программные продукты, призванные решить задачу консолидации контактной информации. Для разработки информационной системы «Единый справочник контактов» (ИС «ЕСК»), учитывающей наработки решений и лишённой их недостатков, необходимо рассмотреть аналоги.

Среди аналогов можно выделить:

- Программный продукт «Корпоративный телефонный справочник» (СЗТК, nowecom.ru) [6].
- Приложение «Phonebook App» (phonebook-app.com) [7].
- Программный продукт «Корпоративный телефонный справочник» (KDMSoft) [8].

### 1.1.1.1 Программный продукт «Корпоративный телефонный справочник» (СЗТК)

На рисунке 1 представлен внешний вид главной страницы веб-интерфейса системы СЗТК.

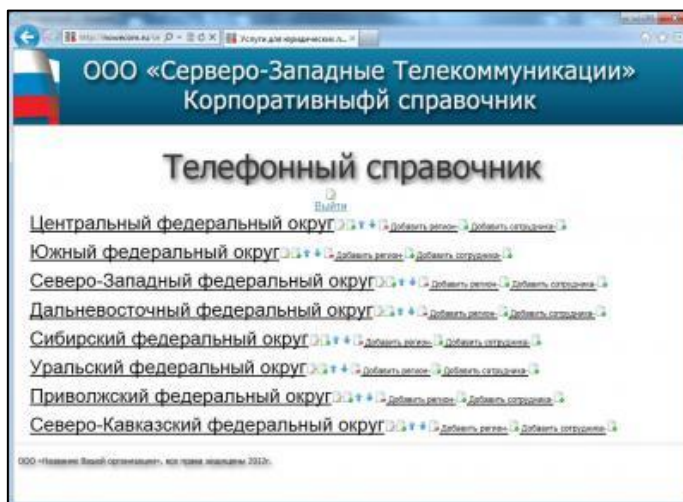


Рисунок 1 – Главная страница системы СЗТК

На рисунке 2 отображена карточка добавления записи.

Рисунок 2 – Карточка сотрудника в системе СЗТК

На рисунке 3 представлен интерфейс административной панели для управления пользователями.

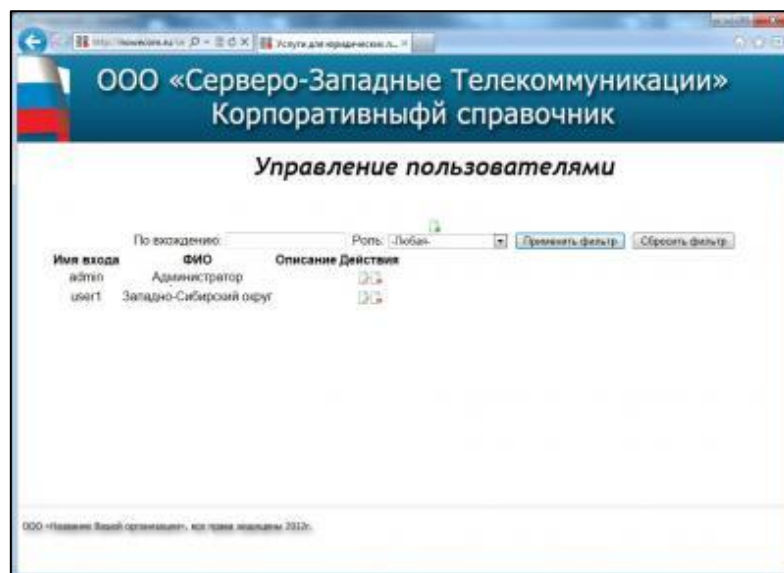


Рисунок 3 – Панель управления пользователями

Достоинства сервиса «Корпоративный телефонный справочник» (СЗТК):

- поддержка сложной иерархической структуры организаций и подразделений, что является критически важным для государственных ведомств;
- наличие развитой ролевой модели доступа, позволяющей назначать права администраторам и региональным представителям;
- доступ к системе возможен с любого устройства через браузер;
- наличие административной панели для централизованного управления пользователями и их полномочиями.

Недостатки сервиса «Корпоративный телефонный справочник» (СЗТК):

- отсутствие модуля геопозиционирования и отображения местоположения организаций или сотрудников на интерактивной карте;
- отсутствие информации о наличии инструментов для массового импорта данных из внешних источников (Excel, XML) и экспорта результатов поиска в PDF.

#### 1.1.1.2 Приложение «Phonebook App»

На рисунке 4 отображен адаптивный веб-интерфейс приложения «Phonebook App» на различных устройствах.

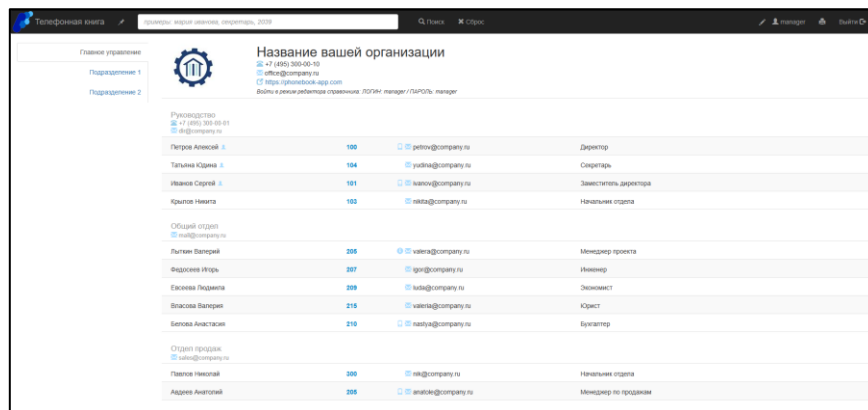


Рисунок 4 – Главная страница сервиса «Phonebook App»

На рисунке 5 представлен интерфейс редактирования записи справочника в режиме реального времени.

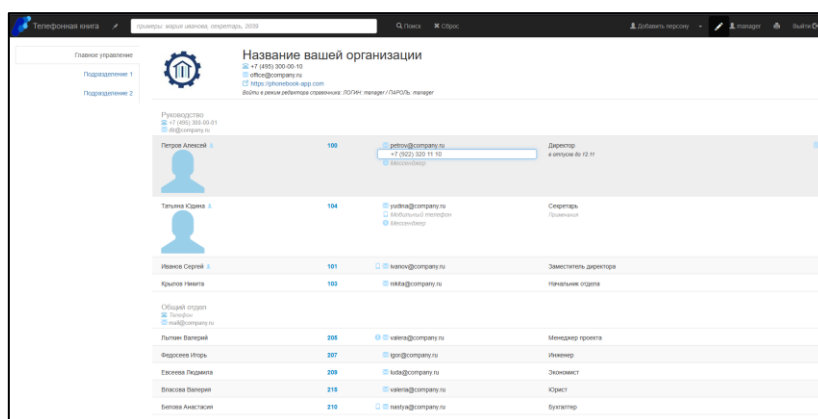


Рисунок 5 – Прямое редактирование контактных данных в «Phonebook App»

На рисунке 6 показана панель настройки прав доступа для пользователей и групп.

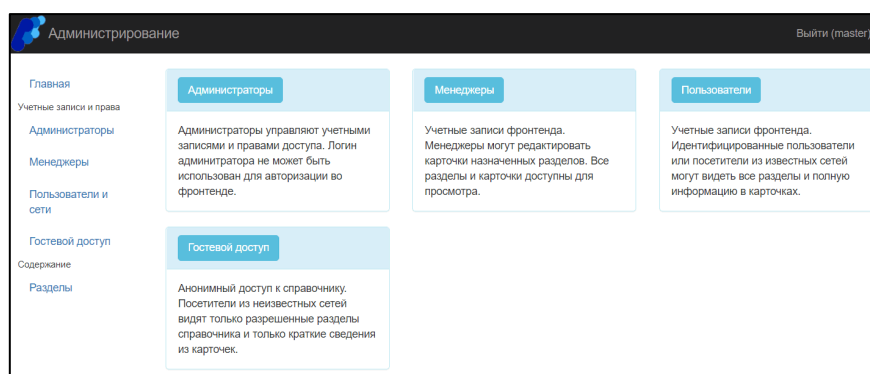


Рисунок 6 – Настройка прав доступа в «Phonebook App»

Достоинства приложения «Phonebook App»:

- интуитивно понятный адаптивный интерфейс, не требующий длительного обучения сотрудников;
- гибкая система разграничения доступа, позволяющая задавать права на уровне отдельных пользователей или IP-подсетей;
- возможность прямого редактирования данных в клиентской части без перезагрузки страницы, что ускоряет работу с системой.

Недостатки приложения «Phonebook App»:

- отсутствие специализации на нуждах государственного сектора, система не поддерживает сложную ролевую модель и не предоставляет журнала аудита действий;
- отсутствие модуля геопозиционирования и отображения данных на карте;
- ограниченная применимость для задач ведомственного справочника, нет функции интерактивной карты, аудита, отчетов по подразделениям и централизованного сопровождения организационной структуры.

#### 1.1.1.3 Программный продукт «Корпоративный телефонный справочник» (KDMSoft)

На рисунке 7 представлен внешний вид главного окна настольного приложения «Корпоративный телефонный справочник» от KDMSoft.

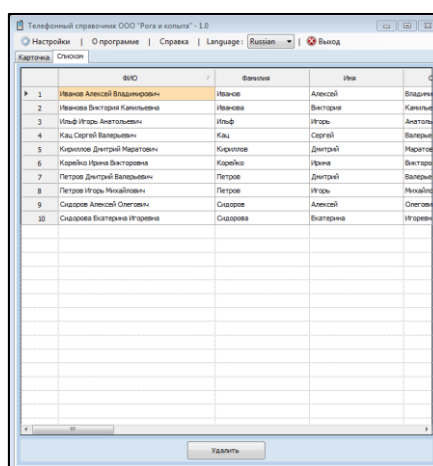


Рисунок 7 – Главное окно программы «Корпоративный телефонный справочник» (KDMSoft)

На рисунке 8 отображен интерфейс добавления и редактирования контактной записи с возможностью заполнения пользовательских полей.

The screenshot shows the 'Телефонный справочник: ООО "Рога и копыта" - 1.0' window. It has a menu bar with 'Настройки', 'О программе', 'Справка', 'Language: Russian', and 'Выход'. Below the menu is a 'Карточка' tab. On the left is a placeholder for a photo with a 'Выбор' button. The main area contains a form for contact details:

|                    |                             |
|--------------------|-----------------------------|
| ФИО                | Кириллов Дмитрий Маратович  |
| Фамилия            | Кириллов                    |
| Имя                | Дмитрий                     |
| Отчество           | Маратович                   |
| Подразделение      | Конструкторский отдел (КО)  |
| Сектор             | -                           |
| Должность          | Ведущий инженер-конструктор |
| Кабинет            | 305                         |
| Внутренний Телефон | 311                         |
| Городской Телефон  | -                           |
| Мобильный Телефон  | +7 903 300 57 51            |
| Мобильный Телефон  | -                           |
| Электронная почта  | support@kdmsoft.ru          |
| Электронная почта  | -                           |
| Автомобиль         | -                           |
| Автомобиль         | -                           |

At the bottom are buttons: 'Обновить', 'Сохранить изменения', and 'Добавить'. A small text block on the left mentions 'КДМСофт' and 'www.kdmsoft.ru'.

Рисунок 8 – Окно редактирования контакта с пользовательскими полями

На рисунке 9 показано окно поиска записей в справочнике.

The screenshot shows the search window of the 'Телефонный справочник'. It has a search bar with 'ФИО' and 'Внутренний Телефон' labels. The search results show 'Кириллов Дмитрий Маратович' and '311'. A 'Карточка' button is visible. The taskbar at the bottom shows the system clock as 20:38 on 03.06.2023.

Рисунок 9 – Окно поиска контактов в программе KDMSoft

Достоинства программного продукта «Корпоративный телефонный справочник»:

- наличие пользовательских полей, позволяющих гибко настраивать структуру хранимых данных под специфику конкретной организации;
- бесплатная основа программы и наличие пробного периода;

– низкие системные требования и совместимость с широким спектром версий операционной системы Windows.

Недостатки программного продукта «Корпоративный телефонный справочник» (KDMSoft):

– реализация в виде настольного приложения для операционной системы Windows, что делает невозможным централизованный многопользовательский доступ через веб-интерфейс;

– отсутствие развитой ролевой модели разграничения прав, журнала аудита действий, интерактивной карты и API для интеграции с внешними системами;

– ограниченная пригодность для реализованных в проекте сценариев массовой загрузки сотрудников из XLSX и формирования PDF-отчетов;

– настольное решение ограничивает его применимость в качестве единого веб-сервиса для большого числа пользователей.

Сравнительная характеристика рассмотренных решений представлена в таблице 1.

Таблица 1 – Сравнение аналогов с разрабатываемым приложением

| Критерий                                       | СЗТК          | PhonebookApp                  | KDMSoft                                   | «Единый Справочник<br>Контактов»                        |
|------------------------------------------------|---------------|-------------------------------|-------------------------------------------|---------------------------------------------------------|
| Веб-доступ                                     | есть          | есть                          | нет, настольное приложение                | есть                                                    |
| Иерархия организаций и подразделений           | есть частично | не является ключевой функцией | не является ключевой функцией             | есть                                                    |
| Ролевая модель                                 | есть частично | ограничена                    | пароль администратора, без развитых ролей | ADMIN, MANAGER, EMPLOYEE                                |
| Ограничение руководителя рамками подразделения | не выделено   | не выделено                   | нет                                       | есть                                                    |
| Поиск по сотрудникам и телефонам               | есть          | есть                          | есть                                      | есть, включая транслитерацию и поиск по цифрам телефона |
| Фильтры по структуре и размещению              | ограничены    | ограничены                    | ограничены                                | есть                                                    |

| Критерий            | СЗТК       | PhonebookApp                     | KDMSoft                       | «Единый Справочник<br>Контактов»               |
|---------------------|------------|----------------------------------|-------------------------------|------------------------------------------------|
| Интерактивная карта | нет        | нет                              | нет                           | есть                                           |
| Импорт данных       | не выделен | есть как вспомогательная функция | не является ключевой функцией | Excel-файл .xlsx с шаблоном и проверкой данных |

### 1.1.2 Постановка задачи

Необходимо разработать корпоративную информационную систему «Единый справочник контактов» (ИС «ЕСК») для использования сотрудниками организации. Система предназначена для централизованного хранения, оперативного поиска и предоставления актуальной контактной информации о структурных подразделениях и работниках. Приложение должно обладать следующими возможностями:

Ведение реестра сотрудников и структуры организации: возможность администраторам загружать и актуализировать в базе данных информацию об организационной структуре (департаменты, отделы, управления) и карточках сотрудников. Карточка должна содержать: ФИО, должность, подразделение, служебный и внутренний номера телефонов, городской телефон (при наличии), адрес расположения рабочего места, адрес электронной почты, дату рождения и иные открытые служебные данные. Поддержка иерархии подчиненности подразделений.

Обработка поисковых запросов пользователей: обеспечение поиска по полям контактной карточки. Наличие многоуровневых фильтров для уточнения результатов поиска. Поддержка ввода запросов с использованием транслитерации.

Наличие личного кабинета сотрудника и ролевая модель доступа. Разграничение прав на основе ролей: Администратор системы, Руководитель подразделения, Сотрудник. Реализация ограничения доступа, при котором сотрудники по умолчанию видят контакты всей организации, а изменение данных разрешено только для своего профиля. Обеспечение процедур аутентификации и авторизации.

Визуализация расположения на карте: Отображение корпусов и офисов организации на интерактивной карте. Возможность просмотра списка сотрудников, размещенных в конкретном здании.

Обработка сервером действий пользователей и ведение аудита. Внесение изменений в базу данных при редактировании карточек сотрудников или структуры организации. Автоматическое ведение журнала событий, фиксация входов в систему, просмотров карточек, внесенных правок. Обеспечение возможности формирования отчета по действиям пользователей для службы безопасности или администратора системы.

Обмен данными с внешними файлами: реализация функционала для первоначального массового наполнения справочника данными из кадровых систем посредством импорта файлов формата XLSX или XML. Обеспечение возможности экспорта результатов поисковой выдачи в форматы XLSX и PDF для формирования печатных версий внутренних телефонных справочников.

Разработка API для потенциальной интеграции с другими информационными системами организации.

Минимальные системные требования к клиентскому рабочему месту:

- браузер: Google Chrome, Mozilla Firefox, Opera (с поддержкой HTML5, CSS3 и JavaScript);
- минимальная скорость подключения к внутренней сети или интернету: 2 Мбит/сек;
- минимальный объем оперативной памяти: 1 ГБ (рекомендуется 2 ГБ и выше для работы с интерактивной картой).

### 1.1.3 Функционал приложения

Функциональные возможности клиент-серверного приложения информационной системы «Единый справочник контактов» (ИС «ЕСК») можно разделить на четыре группы в соответствии с ролями пользователей:

- для неавторизованного пользователя;
- для сотрудника организации;

- для руководителя подразделения;
- для администратора системы.

ИС «ЕСК» предназначена исключительно для внутреннего использования сотрудниками организации и содержит персональные данные, неавторизованный пользователь обладает минимальным набором возможностей. Ему доступна только страница входа в систему, на которой можно пройти процедуру аутентификации (ввести логин и пароль). Любые попытки доступа к внутренним разделам справочника перенаправляют пользователя на форму авторизации.

Диаграмма вариантов использования для неавторизованного пользователя приведена на рисунке 10.



Рисунок 10 – Диаграмма прецедентов неавторизованного пользователя

После успешной аутентификации сотрудник получает доступ к основным функциям системы, предназначенным для оперативного поиска и просмотра актуальной контактной информации. Сотрудник после входа в систему получает доступ к главной странице справочника, поиску сотрудников, фильтрации результатов, просмотру карточек сотрудников, просмотру организационной структуры и интерактивной карты. В личном кабинете сотрудник может просматривать свои данные, редактировать доступные контактные поля, например рабочий телефон, внутренний номер, кабинет и ссылку на фотографию, а также менять пароль. Также сотруднику доступен экспорт данных справочника в XLSX и PDF.

Диаграмма вариантов использования для авторизованного пользователя (сотрудника) приведена на рисунке 11.



Рисунок 11 – Диаграмма прецедентов авторизованного пользователя

Руководитель имеет все возможности сотрудника, а также может редактировать данные сотрудников в пределах своего подразделения и дочерних подразделений. Для руководителя доступны операции сопровождения структуры подразделений в рамках разрешенной области, работа со справочными данными, скачивание шаблона импорта, импорт сотрудников из XLSX и формирование выгрузок по подразделению. Доступ к данным других подразделений остаётся в режиме «только чтение».

Диаграмма вариантов использования для руководителя подразделения приведена на рисунке 12.

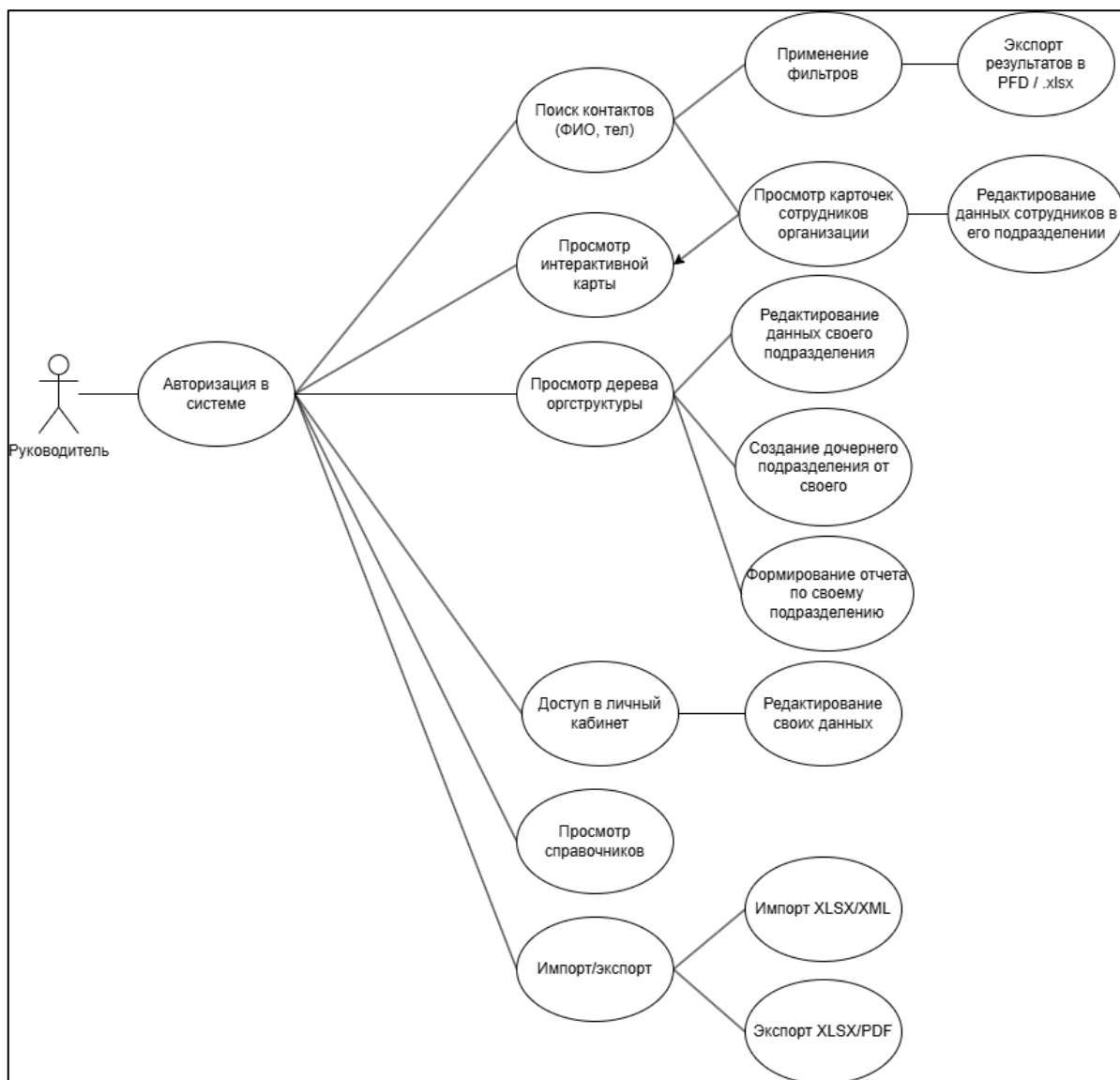


Рисунок 12 – Диаграмма прецедентов руководителя подразделения

Администратор системы имеет наиболее широкие полномочия, необходимые для поддержания актуальности данных, управления доступом и контроля безопасности. Он может создавать, редактировать и удалять сотрудников, управлять учетными записями пользователей, назначать роли, менять пароли, включать и отключать учетные записи, управлять организациями, подразделениями, приемными и городскими кодами. Администратору доступны импорт сотрудников из XLSX, экспорт данных в XLSX и PDF, просмотр и выгрузка журнала аудита, а также управление API-ключами для интеграций. Редактирование

координат организаций, используемых на интерактивной карте, через справочник организаций.

Диаграмма вариантов использования для администратора системы приведена на рисунке 13.

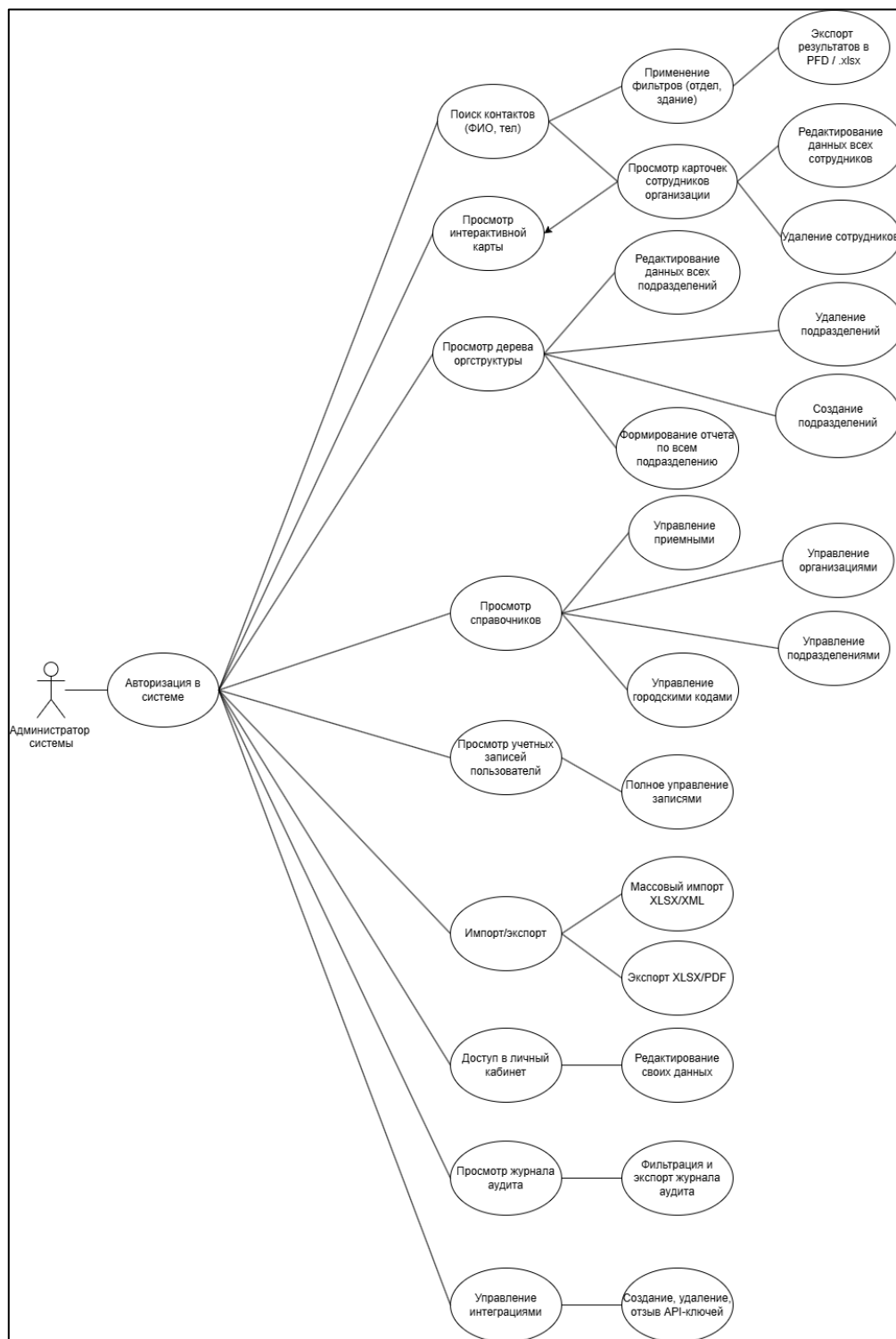


Рисунок 13 – Диаграмма прецедентов администратора системы

## 1.2 Выбор технологий и средств разработки приложения

В данном разделе проведён анализ языков программирования, фреймворков, систем управления базами данных (СУБД) и сред разработки, применимых для создания информационной системы «Единый справочник контактов» (ИС «ЕСК»). На основе сравнительного анализа выбраны технологии, обеспечивающие оптимальное соотношение производительности, безопасности, удобства разработки и последующей поддержки приложения в условиях государственной организации.

### 1.2.1 Выбор платформы исполнения

Выбор платформы определяет доступность приложения для конечных пользователей, требования к инфраструктуре и технологический стек. ИС «ЕСК» должна иметь доступность с любых устройств без установки дополнительного ПО, иметь возможность масштабироваться при росте числа пользователей, иметь централизованное управление данными и политиками безопасности.

Указанным требованиям в полной мере удовлетворяет веб-приложение с клиент-серверной архитектурой. Такой подход позволяет централизованно хранить данные, обновлять бизнес-логику на сервере и предоставлять пользователям доступ к системе без установки настольного клиента.

Альтернативные варианты, такие как настольное или мобильное приложение, были отклонены, поскольку они усложняют централизованное обновление, требуют установки клиентского ПО и хуже подходят для единого внутреннего справочника организации.

### 1.2.2 Выбор языка программирования для серверной части

Для реализации серверной логики ИС «ЕСК» рассматривались языки, обладающие развитыми экосистемами для создания корпоративных веб-приложений.

#### 1.2.2.1 Python

Python широко применяется в веб-разработке благодаря фреймворкам Django и Flask. Его преимущества – скорость написания кода, большое количество

библиотек. Однако Python уступает Java по строгой типизации и распространенности в корпоративных серверных системах.

В связи с этим Python не был выбран в качестве основного серверного языка.

#### 1.2.2.2 C# (.NET)

Платформа .NET предлагает высокую производительность, кроссплатформенность и мощный фреймворк [ASP.NET](#) Core. C# является строго типизированным языком с развитой поддержкой корпоративной разработки. Основные препятствия для использования в проекте:

C# и платформа .NET технически подходят для веб-разработки, но в рамках проекта предпочтение отдано Java из-за зрелой экосистемы Spring, переносимости и широкого набора инструментов для корпоративных приложений. К тому же, учитывая лицензионную политику и историческую связь с Microsoft, предпочтение часто отдаётся решениям на базе открытого ПО и свободных технологий.

#### 1.2.2.3 Java

Java – объектно-ориентированный язык с многолетней историей использования в корпоративном секторе и государственных информационных системах. Его ключевые преимущества для ИС «ЕСК»:

- кроссплатформенность;
- строгая статическая типизация, снижающая количество ошибок на этапе выполнения;
- развитая экосистема для построения безопасных веб-приложений (Spring Framework, Spring Security, Hibernate);
- высокая производительность и масштабируемость, включая поддержку многопоточности;
- инструменты для работы с базами данных, транзакциями и интеграции с внешними системами;
- соответствие требованиям регуляторов, включение Java и Spring в реестры отечественного ПО через сертифицированные сборки OpenJDK [9], [10].

В качестве основного языка серверной разработки выбран Java 21. Java подходит для создания серверных приложений с долгим жизненным циклом, строгой типизацией, развитой экосистемой библиотек и хорошей поддержкой инструментов разработки [11], [12].

### 1.2.3 Выбор фреймворка для серверной части

В экосистеме Java доминирующим решением для создания веб-приложений является Spring Framework и его надстройка Spring Boot. Рассмотрены альтернативы.

#### 1.2.3.1 Spring Framework / Spring Boot

Spring Boot – это проект экосистемы Spring, предназначенный для быстрого создания автономных, готовых к промышленной эксплуатации приложений на Java. Его преимущества:

- встроенный веб-сервер (Tomcat), упрощающий развёртывание;
- Spring Security – готовый модуль для реализации аутентификации, авторизации и защиты от распространённых уязвимостей;
- Spring Data JPA / Hibernate – упрощение работы с базой данных через ORM, поддержка транзакций и миграций схемы;
- Spring AOP и событийная модель – удобная реализация аудита действий пользователей;
- поддержка REST API – создание конечных точек для взаимодействия с фронтендом и будущих интеграций.

В проекте используется Spring Boot 4.0.3, который упрощает настройку приложения, подключение веб-слоя, безопасности, доступа к данным и тестирования [13], [14]. Для обработки HTTP-запросов и реализации REST API используется Spring Web MVC, для разграничения доступа, аутентификации, авторизации – Spring Security, для работы с базой данных – Spring Data JPA и Hibernate, для аудита действий – Spring AOP и AspectJ [15], [16].

### 1.2.3.2 Альтернативные фреймворки

– Jakarta EE (ранее Java EE) – полноценная платформа, но более тяжеловесная и требующая сервера приложений (WildFly, Payara).

– Micronaut, Quarkus – современные фреймворки для микросервисов и serverless.

Альтернативные фреймворки Jakarta EE, Micronaut и Quarkus также подходят для серверной разработки, однако Spring Boot выбран как более удобный вариант для монолитного модульного приложения с REST API, безопасностью, JPA и развитой документацией.

### 1.2.4 Выбор языка и фреймворка для клиентской части

Клиентская часть должна обеспечивать интерактивный интерфейс, работу с картой, динамическую фильтрацию и адаптивность. В качестве основных вариантов рассматривались React и Vue.js.

#### 1.2.4.1 React

React – декларативная JavaScript-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React выбран благодаря компонентному подходу, развитой экосистеме, удобной интеграции с TypeScript и большим количеством готовых решений для административных интерфейсов [17].

Клиентская часть реализована на React 19 и TypeScript. TypeScript добавляет статическую типизацию и снижает вероятность ошибок при работе с моделями данных и API [18]. Для сборки и запуска клиентского приложения выбран Vite 6, обеспечивающий быстрый dev-сервер и сборку проекта [19].

В проекте применяются следующие клиентские библиотеки:

- React Router 7 – маршрутизация между страницами приложения [20];
- Ant Design 5 – таблицы, формы, меню, вкладки, модальные окна и другие компоненты административного интерфейса [21];
- Axios – централизованный HTTP-клиент и обработка токена через interceptors [22];

– TanStack Query 5 – загрузка, кэширование и обновление серверных данных [23];

– Zustand 5 – локальное состояние авторизации и пользовательских настроек [24].

#### 1.2.4.2 Vue.js

Vue – прогрессивный фреймворк, отличающийся плавной кривой обучения и реактивностью. Несмотря на привлекательность, React был выбран исходя из:

– более широкая распространённость в корпоративной среде и большее количество готовых решений;

– лучшая интеграция с TypeScript;

– большая гибкость в работе с картографическими библиотеками.

#### 1.2.4.3 Leaflet

Для отображения организаций на интерактивной карте выбрана библиотека Leaflet. Она является open-source, легковесной, поддерживает OpenStreetMap и другие тайловые серверы, имеет простой API и множество плагинов. Альтернативы: OpenLayers, Яндекс.Карты API, Google Maps API. Они либо тяжеловесны, либо требуют внешних зависимостей и платных лицензий для коммерческого/государственного использования.

Клиентская часть реализуется на JavaScript (с TypeScript) с использованием библиотеки React и картографической библиотеки Leaflet [25].

#### 1.2.5 Выбор системы управления базами данных (СУБД)

База данных должна обеспечивать надежное хранение структурированных данных об организациях, сотрудниках, подразделениях, учетных записях, ролях, приемных, городских кодах, API-ключах и событиях аудита. В качестве основных вариантов рассматривались MySQL и PostgreSQL.

##### 1.2.5.1 MySQL

MySQL – одна из самых распространенных реляционных СУБД с высокой производительностью и подходит для многих веб-приложений. Однако мной выявлены недостатки:

- ограниченная поддержка сложных ограничений целостности и наследования;
- отсутствие встроенной поддержки Row-Level Security, необходимой для разграничения доступа на уровне строк;
- менее развитые средства аудита по сравнению с PostgreSQL.

#### 1.2.5.2 PostgreSQL

PostgreSQL – объектно-реляционная СУБД с открытым исходным кодом. PostgreSQL предоставляет более развитые механизмы ограничений целостности, транзакционности и работы со связанной структурированной информацией. Она поддерживает связи между таблицами, ограничения целостности, индексы и транзакционный режим работы [26].

Разграничение доступа к данным реализовано на уровне серверной бизнес-логики и Spring Security, а не средствами PostgreSQL Row-Level Security. Интерактивная карта также не требует PostGIS: координаты организаций хранятся в обычных полях базы данных, а отображение выполняется на клиенте средствами Leaflet и React Leaflet.

Поэтому PostgreSQL выбрана в качестве СУБД для ИС «ЕСК».

#### 1.2.6 Выбор среды разработки

Среда разработки должна обеспечивать продуктивную работу с Java (Spring Boot) и JavaScript (React), поддерживать инструменты сборки, системы контроля версий и отладку.

##### 1.2.6.1 IntelliJ IDEA

IntelliJ IDEA – интегрированная среда разработки от JetBrains, являющаяся одной из самых популярных IDE для Java-разработки. Преимущества:

- глубокая интеграция со Spring Boot (автодополнение конфигураций, визуализация бинов, поддержка эндпоинтов);
- встроенная поддержка JavaScript/TypeScript и React (подсветка JSX, рефакторинг, отладка клиентского кода);

- инструменты для работы с базами данных (встроенный клиент БД, редактор SQL);
- интеграция с системами сборки (Maven, Gradle) и контроля версий (Git) [27].

Недостатком является коммерческая лицензия, и несоответствие законодательству РФ, недоступность и ограниченность ресурса для скачивания.

#### 1.2.6.2 OpenIDE

OpenIDE – российская открытая интегрированная среда разработки (IDE) на базе IntelliJ Platform. Разработана как локальная альтернатива зарубежным средам программирования с учётом требований технологического суверенитета и импортозамещения. Включает в себя все преимущества IntelliJ IDEA, но не имеет коммерческих лицензий, соответствует законодательству РФ, интегрирована с Axiom JDK [10], [28].

Поэтому OpenIDE была выбрана в качестве основной среды для разработки серверной части.

#### 1.2.6.3 Visual Studio Code

VS Code – бесплатный редактор кода с широкими возможностями расширения. Для фронтенд-разработки (React, TypeScript) он является одним из лучших инструментов благодаря лёгкости и богатому набору плагинов.

#### 1.2.6.4 DBeaver

DBeaver – это бесплатная, кроссплатформенная универсальная программа для работы с базами данных, поддерживающая более 80 различных СУБД, включая PostgreSQL. Основные преимущества для проекта ИС «ЕСК»:

- удобный визуальный редактор таблиц;
- поддержка ER-диаграмм, возможность автоматического построения схемы базы данных для наглядного представления структуры и связей;
- мощный SQL-редактор с подсветкой синтаксиса, автодополнением и выполнением скриптов;

– наличие бесплатной Community Edition, функционала которой достаточно для решения всех задач проекта.

Альтернативные инструменты (pgAdmin, DataGrip) также являются качественными решениями, однако DBeaver выбран благодаря оптимальному сочетанию бесплатности, кроссплатформенности и широты функциональных возможностей [29].

### 1.2.7 Выбор архитектурного паттерна проектирования

Для обеспечения модульности, тестируемости и сопровождаемости кода необходимо следовать определённому архитектурному стилю.

#### 1.2.7.1 Архитектура серверной части

Многослойная архитектура Controller-Service-Repository (CSR). Серверная часть Spring Boot организуется в виде слоёв:

- контроллеры (Controllers) – принимают HTTP-запросы, валидируют входные данные, возвращают ответы (REST API);
- сервисы (Services) – содержат бизнес-логику, правила доступа;
- репозитории (Repositories) – инкапсулируют работу с БД через Spring Data JPA [9].

Такая архитектура является развитием паттерна MVC (Model-View-Controller), где View – это JSON/HTML ответы, Controller – Spring MVC контроллеры, Model – сервисы и сущности JPA.

#### 1.2.7.2 Архитектура клиентской части

Клиентская часть организована по функциональным слоям «app», «pages», «widgets», «features», «entities», «shared», что разделяет маршрутизацию, страницы, виджеты, бизнес-функции, доменные сущности и общий инфраструктурный код. Для серверных данных используется TanStack Query, а для локального состояния – Zustand [23], [24].

Таким образом, архитектура приложения представляет собой клиент-серверную схему SPA + REST API с многослойной серверной частью и компонентной клиентской частью.

### 1.2.8 Итоговый стек технологий

Сводка выбранных технологических решений представлена в таблице 2.

Таблица 2 – Итоговый технологический стек ИС «ЕСК»

| Компонент                  | Выбранная технология                                             |
|----------------------------|------------------------------------------------------------------|
| Архитектура                | клиент-серверное веб-приложение, SPA + REST API                  |
| Серверный язык             | Java 21                                                          |
| Серверный фреймворк        | Spring Boot 4.0.3                                                |
| Веб-слой                   | Spring Web MVC                                                   |
| Безопасность               | Spring Security, JWT                                             |
| Доступ к данным            | Spring Data JPA, Hibernate                                       |
| Аудит                      | Spring AOP, AspectJ, собственный журнал аудита                   |
| База данных                | PostgreSQL                                                       |
| Клиентский язык            | TypeScript 5.7                                                   |
| Клиентский фреймворк       | React 19                                                         |
| Сборка клиентской части    | Vite 6                                                           |
| Маршрутизация              | React Router 7                                                   |
| UI-библиотека              | Ant Design 5                                                     |
| HTTP-клиент                | Axios                                                            |
| Состояние серверных данных | TanStack Query 5                                                 |
| Локальное состояние        | Zustand 5                                                        |
| Карта                      | Leaflet, React Leaflet                                           |
| Импорт и экспорт XLSX      | Apache POI                                                       |
| Экспорт PDF                | OpenPDF                                                          |
| Сборка серверной части     | Maven                                                            |
| Тестирование               | JUnit, Spring Test, Testcontainers, Vitest, Testing Library, MSW |
| Инструменты разработки     | OpenIDE, Visual Studio Code, DBeaver                             |

Выбранные технологии соответствуют требованиям постановки задачи и фактической реализации проекта: они обеспечивают централизованное хранение данных, разграничение доступа, поиск, импорт и экспорт сведений, интерактивную карту и возможность дальнейшего развития системы.

### 1.3 Реализация приложения

В данном разделе описан процесс реализации информационной системы «Единый справочник контактов». Здесь предоставлена полная информация о разработке приложения, рассмотрены основные компоненты, структура базы данных и алгоритмы взаимодействия.

### 1.3.1 Описание работы приложения

В ходе выполнения работы разработано веб-приложение «Единый справочник контактов», предназначенное для централизованного хранения, поиска и сопровождения контактной информации об организациях, структурных подразделениях, приемных и сотрудниках. Приложение позволяет быстро находить сотрудника по ФИО, должности, подразделению, номеру телефона, адресу электронной почты или месту размещения, просматривать карточку сотрудника, работать с организационной структурой, отображать организации на карте, а также выполнять административные операции: управление пользователями, справочниками, импортом и экспортом данных, API-ключами и журналом аудита.

Топология системы и распределение ее компонентов по физическим узлам представлены на рисунке 14.

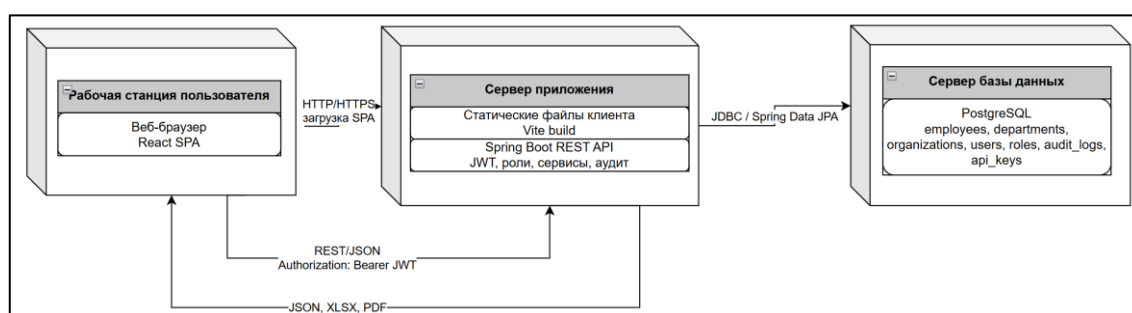


Рисунок 14 – UML-диаграмма развертывания системы

Приложение построено по клиент-серверной архитектуре. Пользователь взаимодействует с клиентской частью через веб-браузер. Клиентская часть реализована на React, TypeScript и Vite, использует библиотеку Ant Design для построения интерфейса, React Router для маршрутизации, TanStack Query для загрузки и кеширования серверных данных, Axios для выполнения HTTP-запросов [17], [22], [23]. Серверная часть реализована на Java с использованием Spring Boot и предоставляет REST API для клиентского приложения. В качестве системы управления базами данных используется PostgreSQL, а доступ к данным на сервере организован через Spring Data JPA [11], [13], [14], [16], [26].

Клиентская часть выполняет роль пользовательского интерфейса: отображает страницы входа, поиска сотрудников, карточки сотрудника, организационной структуры, карты организаций, профиля пользователя и административных разделов. Серверная часть принимает запросы от клиента, проверяет права доступа, выполняет бизнес-логику, обращается к базе данных и возвращает результат в формате JSON либо в виде файла при экспорте данных. База данных хранит сведения о сотрудниках, организациях, подразделениях, приемных, кодах городов, пользователях, ролях, API-ключях и записях аудита.

Работа приложения начинается с авторизации пользователя. На странице входа пользователь вводит имя пользователя и пароль. Клиентская часть отправляет запрос на серверный маршрут `/auth/login`. Сервер проверяет учетные данные, формирует JWT-токен и возвращает клиенту токен, имя пользователя и его роль [31]. Полученный токен сохраняется на клиенте и далее автоматически добавляется в заголовок `Authorization` каждого защищенного запроса. Если токен отсутствует или становится недействительным, пользователь перенаправляется на страницу входа.

После успешной авторизации пользователь попадает в основную часть приложения. Доступные разделы зависят от роли пользователя. В системе предусмотрены роли администратора, руководителя и сотрудника. На уровне клиентской части маршруты защищаются проверкой наличия токена и роли пользователя, а на уровне серверной части права дополнительно контролируются средствами `Spring Security` и аннотациями доступа [15]. Такой подход позволяет скрывать недоступные действия в интерфейсе и одновременно не допускать их выполнения через прямой запрос к API.

Основным пользовательским сценарием является поиск сотрудников. Пользователь вводит поисковый запрос и при необходимости задает дополнительные фильтры: организацию, подразделение, должность или здание. Клиентское приложение формирует GET-запрос к маршруту `/api/search` и передает параметры поиска на сервер. Сервер нормализует введенное значение, учитывает

возможную транслитерацию, поиск по цифрам телефонного номера и фильтры, после чего обращается к базе данных. Результаты сортируются по релевантности и возвращаются клиенту постранично. Клиент отображает найденных сотрудников в таблице, из которой можно перейти к подробной карточке сотрудника.

В карточке сотрудника отображаются основные контактные и организационные сведения: ФИО, должность, рабочие и мобильные телефоны, внутренний номер, адрес электронной почты, подразделение, организация, кабинет, этаж, место размещения и статус активности. В зависимости от роли пользователь может редактировать данные сотрудника, удалить запись или скопировать ссылку на карточку. При создании и изменении данных клиент отправляет на сервер DTO с введенными значениями, сервер выполняет валидацию, сохраняет изменения в базе данных и возвращает обновленную запись.

Для работы с организационной структурой предусмотрен отдельный раздел. В нем пользователь может просматривать дерево подразделений и список сотрудников выбранного подразделения. Организации также могут отображаться на карте, если для них заданы координаты. Административные и управленческие роли получают доступ к справочникам организаций, подразделений, приемных и городских кодов, где можно создавать, редактировать и удалять записи.

Отдельный блок приложения предназначен для обмена данными. Пользователь с соответствующими правами может скачать шаблон Excel, загрузить файл со списком сотрудников для импорта, а также выгрузить справочник сотрудников в форматах Excel и PDF. При импорте сервер разбирает загруженный файл, проверяет строки, сохраняет корректные данные и возвращает отчет о количестве успешно импортированных записей и найденных ошибках. При работе с Excel-файлами используется подход, поддерживаемый библиотекой Apache POI [32]. При экспорте сервер формирует файл и возвращает его клиенту как вложение для скачивания.

Для администратора реализованы дополнительные функции управления системой. В разделе пользователей можно создавать учетные записи, назначать

роли, менять пароль, блокировать и удалять пользователей. В разделе интеграций администратор может создавать и отзываться API-ключи для внешних систем. В разделе аудита отображаются действия пользователей: вход в систему, поиск, просмотр карточек, создание, изменение и удаление записей, импорт и экспорт данных. Записи аудита позволяют контролировать работу с контактным справочником и анализировать историю изменений.

Алгоритм работы приложения заключается в том, что пользователь выполняет действие в веб-интерфейсе, после чего клиентская часть формирует HTTP-запрос к серверу. Сервер проверяет JWT-токен и права пользователя, если проверка не пройдена, клиент получает сообщение об ошибке авторизации или запрете доступа [31]. Если доступ разрешен, сервер передает запрос в соответствующий контроллер, затем в сервисный слой, где выполняется основная бизнес-логика. При необходимости сервис обращается к репозиторию и базе данных для чтения, создания, обновления или удаления сведений. После выполнения операции сервер возвращает клиенту данные, статус выполнения или сформированный файл. Клиент обновляет состояние страницы и отображает результат пользователю. Последовательность выполняемых действий представлена на рисунке 15.

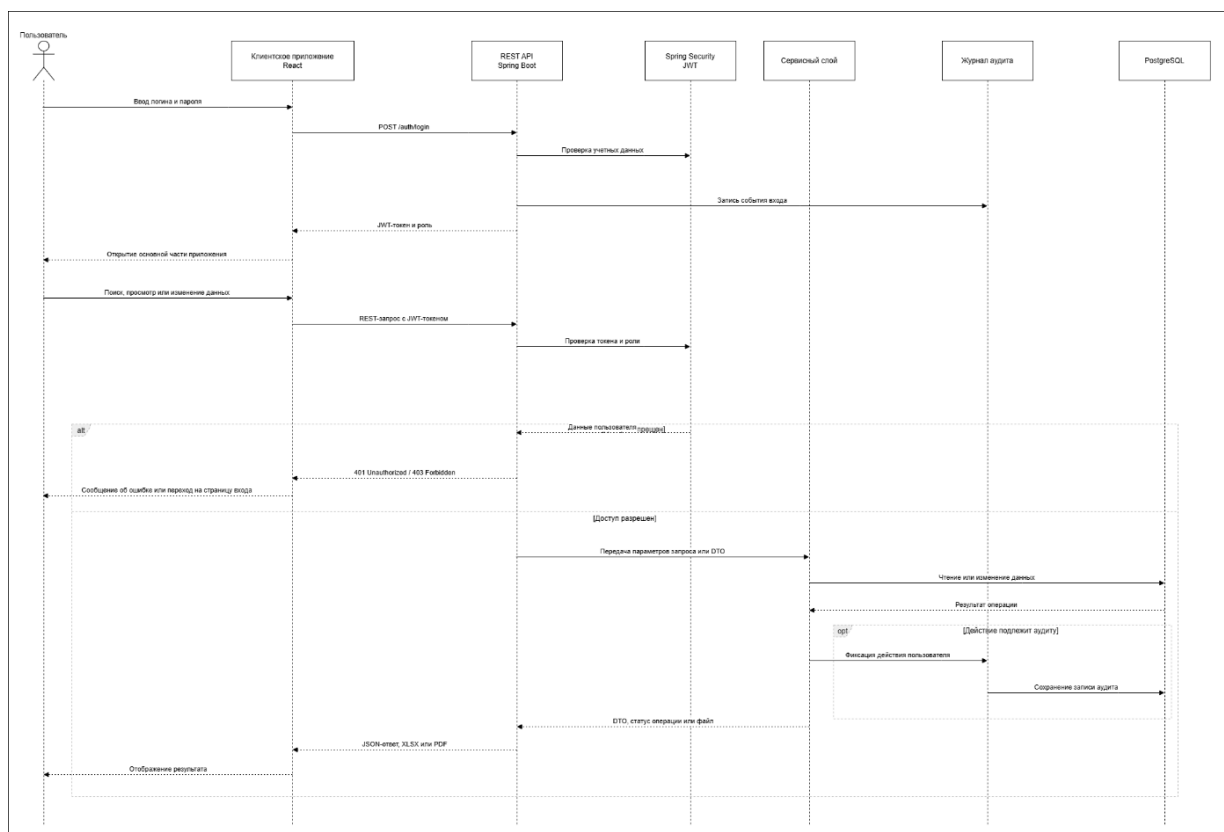


Рисунок 15 – UML-диаграмма последовательности функционирования приложения

Таким образом, приложение обеспечивает единый интерфейс для поиска и сопровождения контактной информации, разграничивает доступ пользователей по ролям, поддерживает импорт и экспорт данных, фиксирует значимые действия в журнале аудита и хранит сведения в централизованной базе данных PostgreSQL.

### 1.3.2 Структура базы данных

Для реализации информационной системы «Единый справочник контактов» необходимо определить сущности, которые будут храниться в базе данных, и связи между ними. Структура базы данных является одним из ключевых элементов приложения, поскольку от нее зависит корректность хранения контактной информации, возможность быстрого поиска сотрудников, разграничение доступа пользователей и ведение журнала действий.

В качестве системы управления базами данных в проекте используется PostgreSQL [26]. Серверная часть приложения обращается к базе данных через

Spring Data JPA, а структура таблиц создается и изменяется с помощью SQL-миграций [16]. В базе данных выделены таблицы для хранения организаций, подразделений, сотрудников, приемных, кодов городов, пользователей, ролей, API-ключей и журнала аудита. Проведя анализ, было решено, что в разрабатываемом приложении будут использоваться следующие таблицы:

В таблице organizations хранится информация об организациях, которые входят в единый справочник. Данные организации используются при построении организационной структуры, фильтрации сотрудников, отображении объектов на карте и ведении справочника организаций. Это представлено в таблице 3.

Таблица 3 – organizations

| Поле       | Тип данных    | Описание                                        | Ключ |
|------------|---------------|-------------------------------------------------|------|
| id         | BIGSERIAL     | Идентификатор организации                       | РК   |
| name       | VARCHAR(500)  | Полное наименование организации                 |      |
| short_name | VARCHAR(150)  | Краткое наименование организации                |      |
| address    | VARCHAR(500)  | Адрес организации                               |      |
| phone      | VARCHAR(30)   | Контактный телефон организации                  |      |
| email      | VARCHAR(150)  | Адрес электронной почты организации             |      |
| website    | VARCHAR(255)  | Сайт организации                                |      |
| latitude   | NUMERIC(10,7) | Географическая широта для отображения на карте  |      |
| longitude  | NUMERIC(10,7) | Географическая долгота для отображения на карте |      |

В таблице departments содержатся сведения о структурных подразделениях организаций. Подразделения могут образовывать иерархию за счет ссылки на родительское подразделение, что реализуется через внешний ключ [26]. Эти данные используются для построения дерева организационной структуры и привязки сотрудников к подразделениям. Это показано в таблице 4.

Таблица 4 – departments

| Поле | Тип данных | Описание                    | Ключ |
|------|------------|-----------------------------|------|
| id   | BIGSERIAL  | Идентификатор подразделения | РК   |

| Поле            | Тип данных   | Описание                                       | Ключ                   |
|-----------------|--------------|------------------------------------------------|------------------------|
| name            | VARCHAR(150) | Наименование подразделения                     |                        |
| organization_id | BIGINT       | Организация, к которой относится подразделение | FK -> organizations.id |
| parent_id       | BIGINT       | Родительское подразделение                     | FK -> departments.id   |

Таблица employees является основной в системе и хранит контактные данные сотрудников. Записи этой таблицы используются в поиске, карточке сотрудника, организационной структуре, профиле пользователя, импорте и экспорте справочника. Это отражено в таблице 5.

Таблица 5 – employees

| Поле           | Тип данных   | Описание                             | Ключ                 |
|----------------|--------------|--------------------------------------|----------------------|
| id             | BIGSERIAL    | Идентификатор сотрудника             | PK                   |
| last_name      | VARCHAR(100) | Фамилия сотрудника                   |                      |
| first_name     | VARCHAR(100) | Имя сотрудника                       |                      |
| middle_name    | VARCHAR(100) | Отчество сотрудника                  |                      |
| work_phone     | VARCHAR(30)  | Рабочий телефон                      |                      |
| internal_phone | VARCHAR(30)  | Внутренний телефон                   |                      |
| mobile_phone   | VARCHAR(30)  | Мобильный телефон                    |                      |
| email          | VARCHAR(150) | Адрес электронной почты              |                      |
| position       | VARCHAR(150) | Должность сотрудника                 |                      |
| department_id  | BIGINT       | Подразделение сотрудника             | FK -> departments.id |
| location       | VARCHAR(255) | Адрес или здание размещения          |                      |
| office         | VARCHAR(50)  | Кабинет                              |                      |
| floor          | VARCHAR(20)  | Этаж                                 |                      |
| birthday       | DATE         | Дата рождения                        |                      |
| photo_url      | VARCHAR(500) | Ссылка на фотографию сотрудника      |                      |
| is_active      | BOOLEAN      | Признак активности записи сотрудника |                      |

| Поле       | Тип данных | Описание                                    | Ключ |
|------------|------------|---------------------------------------------|------|
| updated_at | TIMESTAMP  | Дата и время последнего обновления карточки |      |

В таблице receptions хранится информация о приемных организациях. Данные используются в административном разделе справочников и позволяют хранить основные и дополнительные контактные номера приемных. Это показано в таблице 6.

Таблица 6 – receptions

| Поле             | Тип данных  | Описание                                  | Ключ                   |
|------------------|-------------|-------------------------------------------|------------------------|
| id               | BIGSERIAL   | Идентификатор приемной                    | PK                     |
| name             | VARCHAR(50) | Наименование приемной                     |                        |
| organization     | BIGINT      | Организация, к которой относится приемная | FK -> organizations.id |
| phone_main       | VARCHAR(20) | Основной телефон приемной                 |                        |
| phone_additional | VARCHAR(20) | Дополнительный телефон приемной           |                        |
| description      | TEXT        | Описание приемной                         |                        |

В таблице city\_codes хранятся телефонные коды городов. Таблица используется как редактируемый справочник в административном разделе приложения. Это описано в таблице 7.

Таблица 7 – city\_codes

| Поле      | Тип данных  | Описание             | Ключ   |
|-----------|-------------|----------------------|--------|
| id        | BIGSERIAL   | Идентификатор записи | PK     |
| region    | VARCHAR(50) | Регион               |        |
| city      | VARCHAR(40) | Город                |        |
| city_code | VARCHAR(10) | Код города           | UNIQUE |

Таблица roles содержит роли пользователей системы. Роль определяет доступ пользователя к функциональным разделам приложения. Это показано в таблице 8.

Таблица 8 – roles

| Поле | Тип данных  | Описание           | Ключ   |
|------|-------------|--------------------|--------|
| id   | BIGSERIAL   | Идентификатор роли | PK     |
| name | VARCHAR(50) | Наименование роли  | UNIQUE |

В приложении используются следующие роли: ADMIN, MANAGER и EMPLOYEE. Администратор получает доступ к управлению пользователями, справочниками, интеграциями и аудитом. Руководитель может работать со справочниками, импортом и экспортом. Сотрудник получает доступ к поиску, просмотру контактных данных и ограниченному экспорту.

В таблице users хранятся учетные записи пользователей. Каждая учетная запись связана с ролью, а также может быть связана с конкретным сотрудником справочника. Таблица используется при авторизации, управлении пользователями и формировании профиля текущего пользователя. Это представлено в таблице 9.

Таблица 9 – users

| Поле        | Тип данных   | Описание                             | Ключ                       |
|-------------|--------------|--------------------------------------|----------------------------|
| id          | BIGSERIAL    | Идентификатор пользователя           | PK                         |
| username    | VARCHAR(100) | Имя пользователя для входа в систему | UNIQUE                     |
| password    | VARCHAR(255) | Хеш пароля пользователя              |                            |
| enabled     | BOOLEAN      | Признак активности учетной записи    |                            |
| role_id     | BIGINT       | Роль пользователя                    | FK -> roles.id             |
| employee_id | BIGINT       | Связанный сотрудник                  | FK -> employees.id, UNIQUE |
| created_at  | TIMESTAMP    | Дата создания учетной записи         |                            |

Таблица audit\_logs предназначена для хранения журнала действий пользователей. В нее записываются события авторизации, поиска, просмотра карточек, создания, изменения и удаления записей, импорта, экспорта и операций с API-ключами. Это описано в таблице 10.

Таблица 10 – audit\_logs

| Поле        | Тип данных   | Описание                                          | Ключ |
|-------------|--------------|---------------------------------------------------|------|
| id          | BIGSERIAL    | Идентификатор записи аудита                       | РК   |
| user_id     | BIGINT       | Идентификатор пользователя, выполнившего действие |      |
| username    | VARCHAR(255) | Имя пользователя                                  |      |
| action      | VARCHAR(100) | Тип выполненного действия                         |      |
| entity_type | VARCHAR(100) | Тип сущности, с которой связано действие          |      |
| entity_id   | BIGINT       | Идентификатор сущности                            |      |
| details     | TEXT         | Детальное описание действия                       |      |
| ip_address  | VARCHAR(100) | IP-адрес пользователя                             |      |
| created_at  | TIMESTAMP    | Дата и время выполнения действия                  |      |

Таблица api\_keys хранит API-ключи, предназначенные для интеграции справочника с внешними системами. Полный ключ показывается пользователю только при создании, а в базе данных хранится его хеш и отображаемый префикс. Это отражено в таблице 11.

Таблица 11 – api\_keys

| Поле       | Тип данных   | Описание                       | Ключ   |
|------------|--------------|--------------------------------|--------|
| id         | BIGSERIAL    | Идентификатор API-ключа        | РК     |
| name       | VARCHAR(150) | Название API-ключа             |        |
| key_prefix | VARCHAR(32)  | Отображаемый префикс ключа     |        |
| key_hash   | VARCHAR(64)  | Хеш API-ключа                  | UNIQUE |
| scopes     | VARCHAR(500) | Описание области доступа ключа |        |
| active     | BOOLEAN      | Признак активности ключа       |        |
| created_at | TIMESTAMP    | Дата создания ключа            |        |
| revoked_at | TIMESTAMP    | Дата отзыва ключа              |        |

| Поле         | Тип данных | Описание                            | Ключ |
|--------------|------------|-------------------------------------|------|
| last_used_at | TIMESTAMP  | Дата последнего использования ключа |      |

Основные связи между таблицами строятся следующим образом. Одна организация может содержать несколько подразделений и несколько приемных. Подразделение относится к одной организации и может иметь родительское подразделение. Один сотрудник относится к одному подразделению. Пользователь имеет одну роль и может быть связан с одним сотрудником. Такие связи реализуются с использованием первичных и внешних ключей [26]. Журнал аудита логически связан с пользователем через поле user\_id, однако запись аудита хранит также имя пользователя, чтобы сохранить информацию о действии даже при изменении или удалении учетной записи.

Графически структура базы данных показана на рисунке 16.

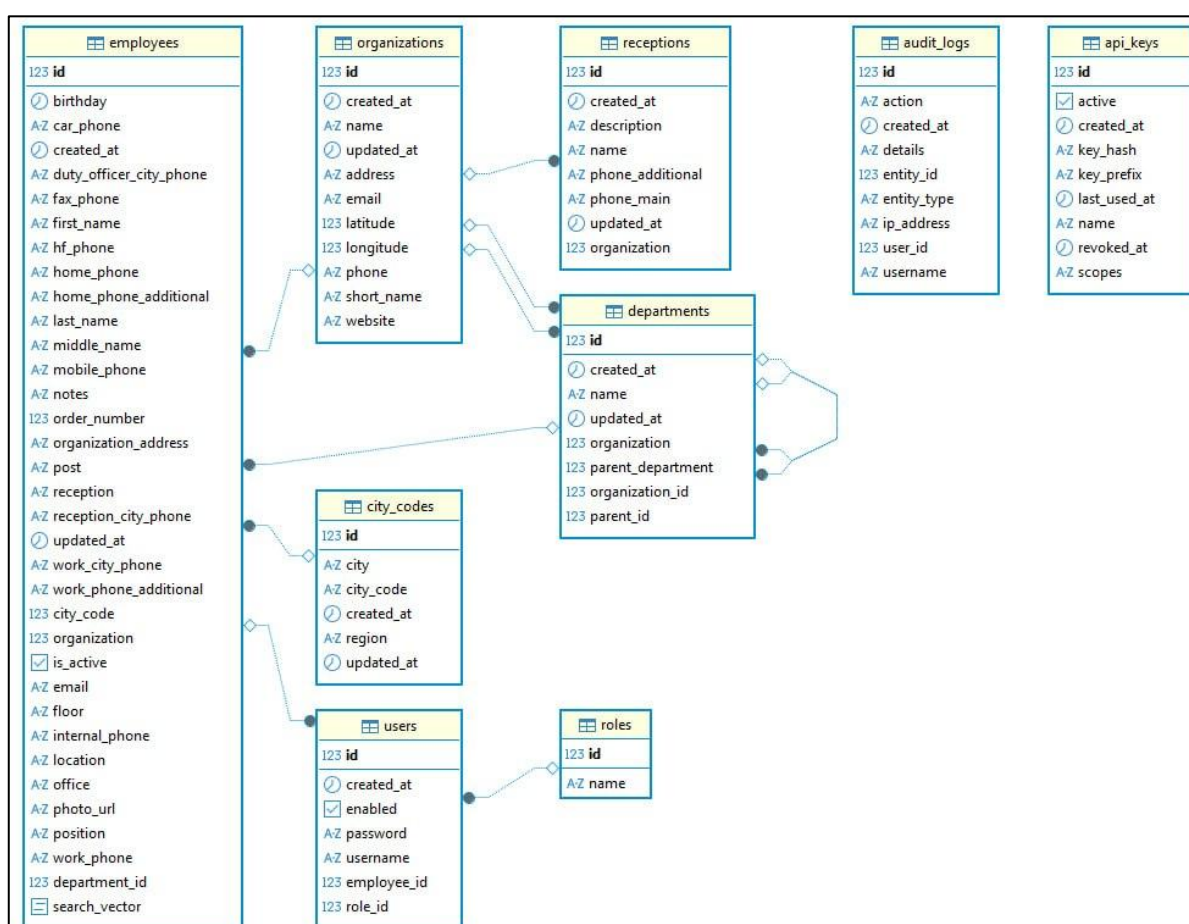


Рисунок 16 – Графическая структура базы данных

### 1.3.3 Описание компонентов

Разработанное приложение имеет клиент-серверную архитектуру и состоит из двух основных частей: клиентской части, реализующей пользовательский интерфейс, и серверной части, реализующей API, бизнес-логику, безопасность и работу с базой данных. Клиентское приложение реализовано на React и TypeScript, серверное приложение написано на Java с использованием Spring Boot [11], [13], [14], [17]. На UML-диаграмме компонентов серверная часть, включая API, сервисы, репозитории и базу данных, представлена как Model, а клиентские слои React-приложения, отвечающие за состояние интерфейса и пользовательские сценарии, представлены как ViewModel. Общая UML-диаграмма компонентов приложения представлена на рисунке 17.

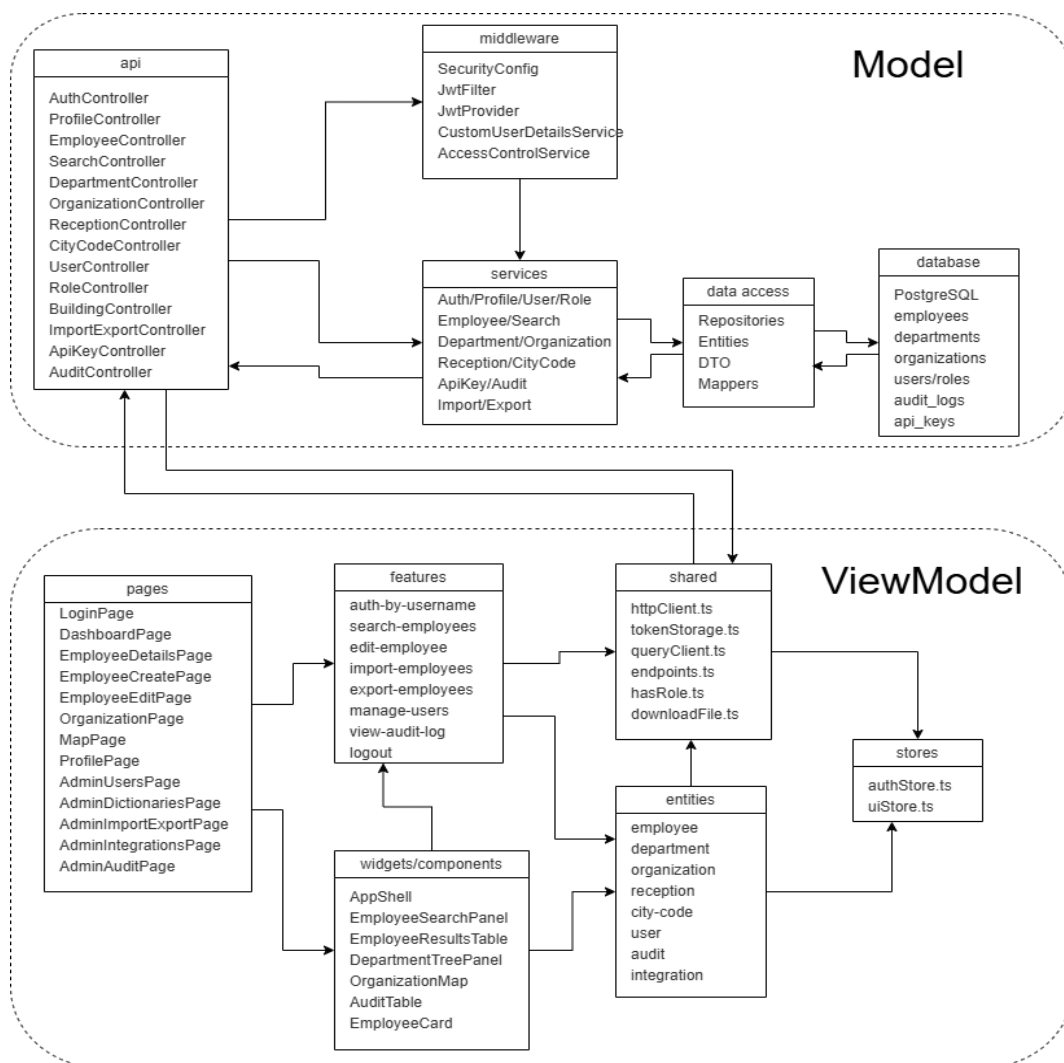


Рисунок 17 – UML-диаграмма компонентов приложения

Клиентская часть выполняет функции представления и клиентской модели состояния. Она отвечает за отображение страниц, ввод данных пользователем, локальное хранение JWT-токена, маршрутизацию, выполнение HTTP-запросов к серверу и отображение полученных данных [31]. Серверная часть принимает запросы клиента, проверяет авторизацию и права доступа, выполняет бизнес-логику, обращается к базе данных через репозитории, формирует ответы в формате JSON или файлы Excel/PDF.

Серверная часть построена по слоистой архитектуре. В ней можно выделить следующие основные группы компонентов:

- контроллеры REST API;
- сервисы бизнес-логики;
- репозитории доступа к данным;
- сущности базы данных;
- DTO и маркер-компоненты;
- компоненты безопасности;
- компоненты аудита;
- компоненты импорта и экспорта;
- компоненты обработки ошибок и конфигурации.

Схема серверных компонентов показана на рисунке 18.

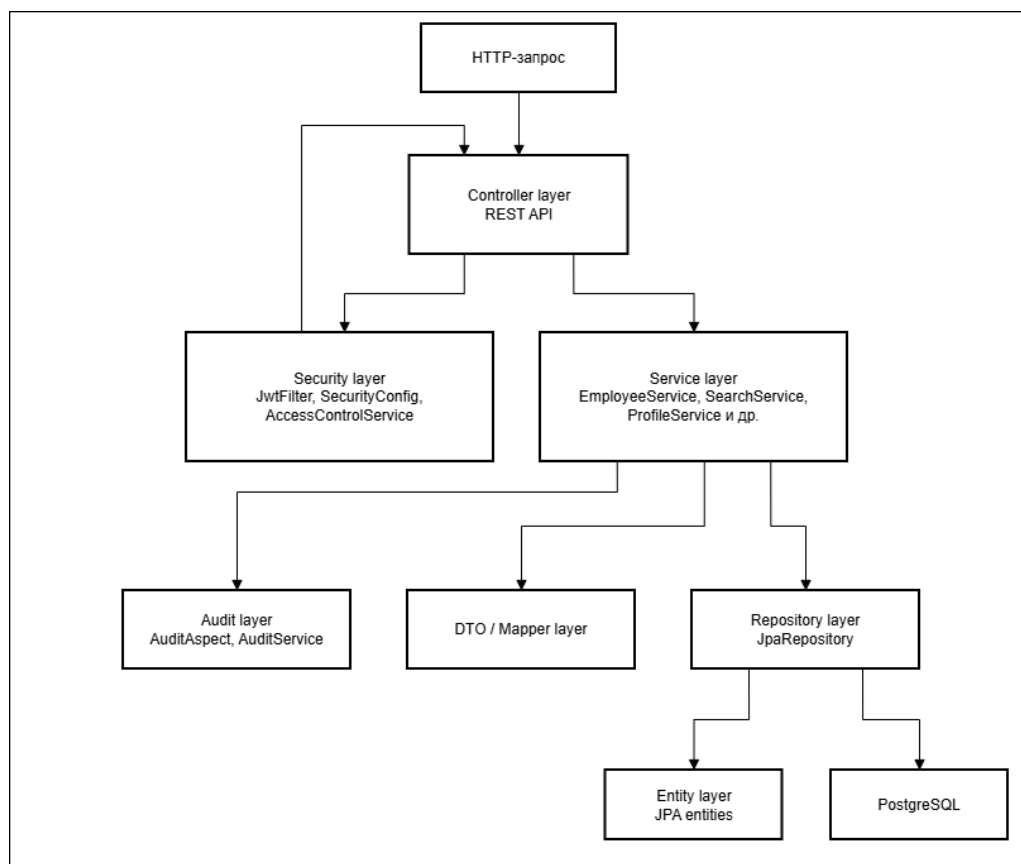


Рисунок 18 – Структура серверных компонентов

Контроллеры REST API принимают HTTP-запросы от клиентского приложения, извлекают параметры запроса или тело JSON, вызывают соответствующие сервисы и возвращают результат клиенту. Контроллеры не содержат основную бизнес-логику, а выступают входными точками приложения. В проекте используются следующие контроллеры:

- AuthController – выполняет аутентификацию пользователя и возвращает JWT-токен;
- ProfileController – предоставляет данные профиля текущего пользователя, позволяет обновлять собственные контактные данные и менять пароль;
- EmployeeController – реализует получение списка сотрудников, просмотр карточки, создание, редактирование и удаление сотрудников;
- SearchController – выполняет поиск сотрудников по строке запроса и фильтрам;

- DepartmentController – управляет подразделениями и их иерархией;
- OrganizationController – управляет организациями;
- ReceptionController – управляет справочником приемных;
- CityCodeController – управляет справочником кодов городов;
- BuildingController – формирует точки организаций для отображения на карте;
- UserController – предоставляет администратору функции управления учетными записями;
- RoleController – возвращает справочник ролей;
- ImportExportController – выполняет импорт сотрудников из Excel, выдачу шаблона импорта, экспорт справочника в Excel и PDF;
- ApiKeyController – управляет API-ключами для интеграций;
- AuditController – предоставляет журнал действий пользователей и экспорт отчета аудита;
- SpaController – обслуживает маршруты одностраничного React-приложения при открытии страниц напрямую.

Сервисный слой содержит основную бизнес-логику приложения. Сервисы проверяют входные данные, контролируют ограничения доступа, вызывают репозитории и преобразуют результат в DTO. Такой подход отделяет обработку HTTP-запросов от правил предметной области и соответствует распространенной организации Spring-приложений [14]. Основные сервисы приложения:

- UserService – создание, изменение, блокировка и удаление пользователей, смена пароля, назначение ролей;
- RoleService – создание и получение ролей ADMIN, MANAGER и EMPLOYEE;
- ProfileService – получение профиля текущего пользователя, обновление собственных контактных данных, смена пароля;

- EmployeeService – работа с карточками сотрудников, фильтрация списка сотрудников и проверка права руководителя на изменение сотрудника;
- DepartmentService – работа с подразделениями, поддержка иерархии подразделений и проверка прав на управление ими;
- OrganizationService – работа со справочником организаций;
- ReceptionService – работа со справочником приемных;
- CityCodeService – работа со справочником кодов городов;
- SearchService – нормализация поискового запроса, обработка транслитерации и подготовка параметров поиска;
- ApiKeyService – создание, отзыв и удаление API-ключей;
- AuditService – запись и выборка событий аудита;
- ExcellImportService, ExcelExportService, PdfExportService – импорт и экспорт данных сотрудников.

Репозитории отвечают за доступ к данным. Они наследуются от JpaRepository и используют Spring Data JPA для выполнения стандартных операций создания, чтения, обновления и удаления [16]. Для отдельных сценариев определены дополнительные методы: поиск пользователя по username, поиск сотрудников по подразделению или организации, подсчет сотрудников организации, проверка уникальности email, фильтрация записей аудита. Для сложного поиска сотрудников используется SearchRepository, в котором определен SQL-запрос с расчетом релевантности результата.

Сущности базы данных описывают таблицы PostgreSQL на уровне Java-классов [26]. К основным сущностям относятся Organization, Department, Employee, Reception, CityCode, User, Role, ApiKey и AuditLog. Между сущностями заданы связи: организация содержит подразделения и приемные, подразделение содержит сотрудников и может иметь родительское подразделение, пользователь связан с ролью и может быть связан с сотрудником.

DTO используются для обмена данными между сервером и клиентом. Они отделяют внутренние JPA-сущности от внешнего API и позволяют возвращать

клиенту только необходимые данные. Например, `EmployeeResponseDto` содержит не только поля сотрудника, но и наименование подразделения, организации, а также данные связанной учетной записи. Mapper-компоненты преобразуют сущности в DTO и применяют данные из request DTO к существующим сущностям.

Компоненты безопасности отвечают за аутентификацию, авторизацию и разграничение доступа. `SecurityConfig` настраивает Spring Security, отключает серверные сессии, подключает JWT-фильтр и определяет публичные маршруты [15]. `JwtProvider` создает и проверяет JWT-токены [31]. `JwtFilter` извлекает токен из заголовка `Authorization`, проверяет его и помещает пользователя в `SecurityContext`. `CustomUserDetailsService` загружает пользователя из базы данных по имени. `AccessControlService` проверяет, может ли пользователь с ролью `MANAGER` изменять сотрудника или подразделение только в рамках своего подразделения и вложенных подразделений.

Фрагмент метода аутентификации представлен на листинге 1. Пользователь передает логин и пароль, после чего сервер проверяет учетные данные, формирует JWT-токен, записывает событие входа в журнал аудита и возвращает данные клиенту.

#### Листинг 1 – Метод авторизации в `AuthController`

```
@PostMapping("/login")
public JwtResponse login(@RequestBody LoginRequest request,
    HttpServletRequest servletRequest) {
    Authentication authentication =
        authenticationManager.authenticate(
            new
                UsernamePasswordAuthenticationToken(request.username(),
                    request.password())
        );
    UserPrincipal principal = (UserPrincipal)
        authentication.getPrincipal();
    String token = jwtProvider.generateToken(principal);
    auditService.log(
        principal,
        AuditActionType.LOGIN,
        "User",
        principal.getId(),
        "Successful login",
    );
}
```

```

        servletRequest
    );
    return new JwtResponse(token, "Bearer",
principal.getUsername(), principal.getRole());
}

```

Компонент JwtFilter работает как middleware серверной части. Он выполняется перед стандартной обработкой запроса Spring Security, извлекает JWT-токен из заголовка Authorization, проверяет подпись и срок действия токена, после чего загружает пользователя и устанавливает объект аутентификации [15], [31].

Компонент аудита реализован с помощью AuditService, AuditRepository, AuditLog и AuditAspect. AuditAspect перехватывает методы, помеченные аннотацией @Auditable, получает состояние объекта до изменения, выполняет основной метод, формирует описание действия и передает запись в AuditService. В журнал попадают события поиска, просмотра карточек, создания, изменения и удаления сотрудников, операций с подразделениями, импорта, экспорта, входа в систему и управления API-ключами.

Компоненты импорта и экспорта вынесены в отдельный пакет importexport. ImportExportController принимает файл Excel или возвращает сформированный файл. ExcelImportService и EmployeeExcelImporter разбирают строки загруженного файла и формируют результат импорта. ExcelExportService и EmployeeExcelExporter формируют таблицу сотрудников в формате XLSX. Для чтения и формирования Excel-файлов используется библиотека Apache POI [32]. PdfExportService и EmployeePdfExporter формируют PDF-отчет. Благодаря этому логика работы с файлами отделена от основной логики справочника.

Клиентская часть приложения организована по слоям, близким к Feature-Sliced Design. Такая структура позволяет разделять маршруты, крупные виджеты, пользовательские действия, сущности предметной области и общие вспомогательные компоненты. Схема клиентских компонентов показана на рисунке 19.

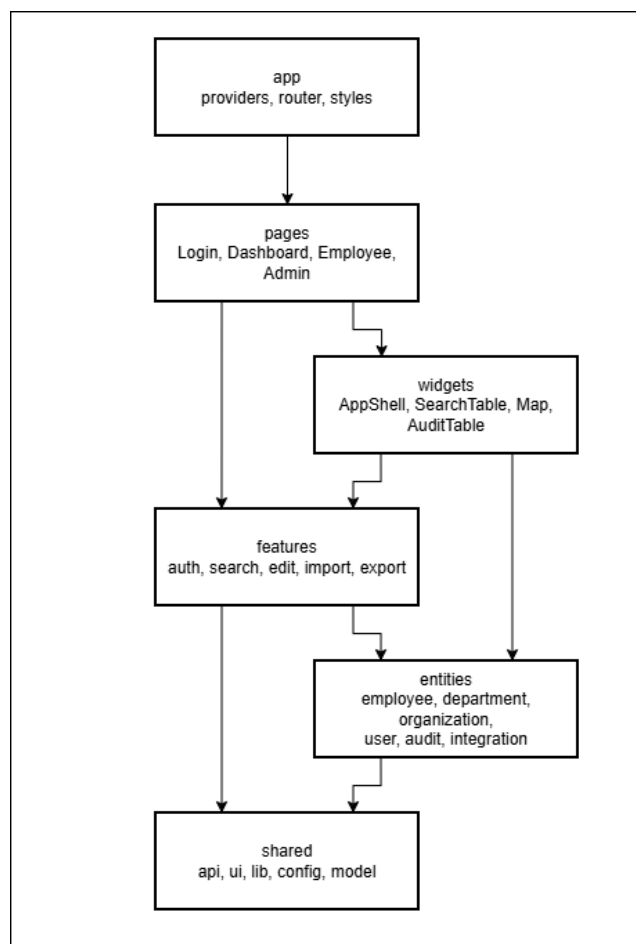


Рисунок 19 – Структура клиентских компонентов

Слой app содержит точку входа приложения, провайдеры, маршрутизацию и глобальные стили. AppProviders подключает Ant Design ConfigProvider, QueryProvider, AuthSessionProvider и RouterProvider. QueryProvider предоставляет клиент React Query для кэширования серверных данных [23]. AuthSessionProvider при наличии токена выполняет запрос профиля пользователя и восстанавливает состояние после обновления страницы. RouterProvider подключает маршруты приложения. ProtectedRoute защищает страницы от неавторизованных пользователей, RoleRoute ограничивает доступ по ролям.

Слой shared содержит переиспользуемую инфраструктуру и базовые UI-компоненты. В shared/api находятся httpClient, tokenStorage, queryClient, endpoints и общие типы API. В shared/ui расположены PageHeader, LoadingScreen, ErrorState, EmptyState, CopyButton, DataToolbar и ConfirmDeleteButton. В shared/lib находятся

утилиты форматирования ФИО и телефона, скачивания файлов, проверки ролей и debounce-функция.

Файл `httpClient.ts` является важным компонентом клиентской инфраструктуры. Он создает общий экземпляр `Axios`, автоматически добавляет JWT-токен в каждый запрос и выполняет перенаправление на страницу входа при получении ответа 401 [22]. Фрагмент компонента представлен на листинге 2.

#### Листинг 2 – Добавление JWT-токена к HTTP-запросам

```
httpClient.interceptors.request.use((config) => {
  const token = getStoredToken();

  if (token) {
    config.headers.Authorization = `Bearer ${token}`;
  }

  return config;
});

httpClient.interceptors.response.use(
  (response) => response,
  (error) => {
    if (axios.isAxiosError(error) && error.response?.status ===
401) {
      clearStoredToken();
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);
```

Слой `entities` содержит описание предметных сущностей, типы данных, API-обертки и небольшие UI-компоненты, связанные с конкретными объектами. В проекте выделены следующие сущности:

- `employee` – типы `Employee` и `EmployeeRequest`, API для работы с сотрудниками, компоненты `EmployeeAvatar`, `EmployeeCard`, `EmployeeContactList`, `EmployeeStatusTag`;
- `department` – типы подразделений, API, компонент `DepartmentSelect` и функция `buildDepartmentTree` для построения дерева;
- `organization` – типы организаций, API, компонент `OrganizationSelect`;

- reception – типы и API приемных;
- city-code – типы и API кодов городов;
- user – типы пользователей, API и компонент UserRoleTag;
- audit – типы записей аудита и компонент AuditActionTag;
- integration – типы и API для API-ключей.

Слой features содержит пользовательские действия и сценарии. В нем находятся компоненты, которые объединяют UI, API и локальное состояние конкретного действия:

- auth-by-username – авторизация пользователя, хранение токена и получение профиля;
- search-employees – поисковая форма, фильтры и hook useEmployeeSearch;
- edit-employee – форма создания и редактирования сотрудника;
- export-employees – кнопки экспорта справочника в Excel и PDF;
- import-employees – загрузка Excel-файла и скачивание шаблона импорта;
- manage-users – форма создания и редактирования пользователя, выбор роли;
- view-audit-log – фильтры журнала аудита и API экспорта отчета;
- logout – выход пользователя из системы.

Функция useEmployeeSearch инкапсулирует повторяющуюся логику загрузки результатов поиска через React Query [23]. Она принимает параметры поиска, формирует queryKey и вызывает API-функцию searchEmployees. Компонент представлен на листинге 3.

### Листинг 3 – Hook поиска сотрудников

```
export function useEmployeeSearch(params: EmployeeSearchParams)
{
  return useQuery({
    queryKey: ['employees', 'search', params],
    queryFn: () => searchApi.searchEmployees(params)
  });
}
```

Слой widgets содержит крупные составные компоненты страниц. Они объединяют сущности и features в самостоятельные блоки интерфейса:

- MainLayout, AppSidebar, AppHeader, UserMenu – основной каркас приложения с боковым меню, шапкой и меню пользователя;
- EmployeeSearchPanel – панель ввода поисковой строки и фильтров;
- EmployeeResultsTable – таблица результатов поиска сотрудников с действиями просмотра, редактирования и удаления;
- DepartmentTreePanel – дерево подразделений с поиском по названию;
- OrganizationMap – карта организаций с фильтрами и списком сотрудников по кабинетам;
- AuditTable – таблица записей аудита и модальное окно с подробностями;
- EmployeeCard – виджет карточки сотрудника.

Слой pages содержит страницы, соответствующие маршрутам приложения. В проекте реализованы следующие страницы:

- LoginPage – страница входа;
- DashboardPage – главная страница поиска сотрудников;
- EmployeeDetailsPage – карточка сотрудника;
- EmployeeCreatePage – создание сотрудника;
- EmployeeEditPage – редактирование сотрудника;
- OrganizationPage – просмотр организационной структуры;
- MapPage – карта организаций;
- ProfilePage – профиль текущего пользователя;
- AdminUsersPage – управление пользователями;
- AdminDictionariesPage – управление справочниками организаций, подразделений, приемных и кодов городов;
- AdminImportExportPage – импорт и экспорт данных;
- AdminIntegrationsPage – управление API-ключами;
- AdminAuditPage – просмотр журнала аудита;

- ForbiddenPage – страница запрета доступа;
- NotFoundPage – страница отсутствующего маршрута.

Маршрутизация клиентской части описана вручную в routes.tsx. Страница /login доступна без токена. Остальные маршруты размещаются внутри ProtectedRoute и требуют авторизации. Для административных страниц дополнительно используется RoleRoute, который проверяет роль пользователя. Например, доступ к управлению пользователями, интеграциями и аудитом разрешен только администратору, а доступ к справочникам разрешен администратору и руководителю.

При авторизации пользователь вводит логин и пароль в компоненте LoginForm. Компонент вызывает authApi.login, сохраняет полученный JWT-токен в authStore и localStorage, затем выполняет запрос profileApi.getProfile. После успешной загрузки профиля пользователь перенаправляется на страницу, которую пытался открыть до авторизации, либо на главную страницу поиска. Состояние авторизации хранится в Zustand-хранилище authStore.

Клиентский компонент LoginForm связывает форму входа, API авторизации и глобальное состояние пользователя. Упрощенный фрагмент логики входа представлен на листинге 4.

#### Листинг 4 – Логика входа в LoginForm

```
const mutation = useMutation({
  mutationFn: async (payload: LoginRequest) => {
    const jwt = await authApi.login(payload);
    setToken(jwt.token);
    const profile = await profileApi.getProfile();
    setUser(profile);
    return profile;
  },
  onSuccess: () => {
    navigate(state?.from?.pathname ?? '/', { replace: true });
  }
});
```

Для разграничения доступа на сервере используется не только проверка роли, но и проверка области ответственности руководителя. Если пользователь имеет

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 55   |

роль MANAGER, он может управлять только сотрудниками и подразделениями своего подразделения и вложенных подразделений. Фрагмент проверки представлен на листинге 5.

#### Листинг 5 – Проверка области доступа руководителя

```
public void assertCanManageEmployee(Employee employee) {
    if (isAdmin()) {
        return;
    }
    assertManager();
    assertDepartmentInManagerScope(employee == null ? null :
employee.getDepartment());
}

private void assertDepartmentInManagerScope(Department
targetDepartment) {
    Department managerDepartment = currentManagerDepartment();
    Long managerDepartmentId = managerDepartment.getId();
    Department cursor = targetDepartment;
    while (cursor != null && cursor.getId() != null) {
        if (cursor.getId().equals(managerDepartmentId)) {
            return;
        }
        cursor = cursor.getParent();
    }
    throw new AccessDeniedException("Manager can edit only own
department and child departments");
}
```

Компонент поиска сотрудников состоит из формы ввода параметров, hook-запроса и таблицы результатов. DashboardPage хранит параметры поиска в URL, что позволяет обновлять страницу без потери текущего запроса. EmployeeSearchPanel собирает строку поиска и фильтры, useEmployeeSearch загружает данные с сервера, EmployeeResultsTable отображает результаты и предоставляет действия над сотрудниками. На сервере запрос принимает SearchController, а SearchService нормализует строку поиска, учитывает транслитерацию и поиск по цифрам телефона.

Компонент организационной структуры состоит из OrganizationPage, DepartmentTreePanel и таблицы сотрудников выбранного подразделения.

DepartmentTreePanel загружает список подразделений, фильтрует их по строке поиска и строит дерево с помощью функции buildDepartmentTree. При выборе подразделения OrganizationPage запрашивает сотрудников этого подразделения и отображает их в таблице.

Компонент карты организаций построен на React Leaflet [25]. OrganizationMap получает точки организаций через /api/buildings, отображает маркеры на карте, позволяет фильтровать организации и открывать боковую панель со списком сотрудников выбранной организации, сгруппированных по кабинетам. На сервере BuildingController формирует данные карты на основе таблицы организаций и подсчета сотрудников.

Компонент импорта и экспорта разделен между клиентом и сервером. На клиенте ImportEmployeesCard позволяет скачать шаблон и загрузить Excel-файл, ExportButtons выполняет скачивание Excel или PDF. На сервере ImportExportController принимает multipart-запрос, вызывает ExcelImportService и возвращает ImportResultDto с количеством успешных и ошибочных строк. При экспорте сервер формирует файл и возвращает его в ответе с заголовком Content-Disposition.

Компонент управления пользователями доступен администратору. AdminUsersPage загружает список пользователей и ролей, отображает таблицу с фильтрами по имени, роли и статусу, использует UserFormDrawer для создания и редактирования пользователей, а также предоставляет действия смены пароля, блокировки и удаления учетной записи.

Компонент справочников реализован в AdminDictionariesPage. На одной странице размещены вкладки организаций, подразделений, приемных и кодов городов. Для каждой вкладки используется таблица записей и Drawer-форма создания или редактирования. В зависимости от активной вкладки страница вызывает соответствующий API-модуль: organizationApi, departmentApi, receptionApi или cityCodeApi.

Компонент аудита состоит из AdminAuditPage, AuditFilters и AuditTable. AuditFilters формирует параметры фильтрации по пользователю, дате, периоду, действию и типу сущности. AuditTable отображает журнал действий и позволяет открыть подробности события. Серверный AuditController возвращает постраничный список записей и CSV-отчет для скачивания.

Компонент интеграций представлен страницей AdminIntegrationsPage и API-модулем apiKeysApi. Администратор может создать API-ключ, увидеть его значение только один раз после создания, отозвать ключ или удалить его. На сервере ApiKeyService генерирует случайный ключ, сохраняет только хеш, а пользователю возвращает исходное значение ключа один раз в ответе на создание.

### 1.3.4 Основные алгоритмы взаимодействия клиента и сервера

Взаимодействие клиентской и серверной частей приложения построено по клиент-серверной модели. Клиентская часть, реализованная на React и TypeScript, формирует HTTP-запросы к REST API серверного приложения [17]. Серверная часть, реализованная на Java с использованием Spring Boot, принимает запросы, проверяет JWT-токен и права доступа, выполняет бизнес-логику, обращается к базе данных PostgreSQL и возвращает клиенту результат в формате JSON либо в виде файлов XLSX, PDF или CSV [11], [13], [14], [26]. На диаграммах деятельности алгоритмы показаны в укрупненном виде, пользователь выполняет действие в интерфейсе, а система объединяет клиентскую и серверную обработку запроса.

Клиентское приложение использует общий HTTP-клиент на базе Axios [22]. Перед отправкой запроса он считывает JWT-токен из локального хранилища и добавляет его в заголовок Authorization в формате Bearer [31]. Если сервер возвращает код 401, клиент удаляет сохраненный токен и перенаправляет пользователя на страницу входа. За счет этого все функциональные сценарии используют единый механизм авторизации и обработки истекшей сессии.

На сервере обработка защищенных запросов начинается с JwtFilter. Фильтр извлекает токен из заголовка Authorization, проверяет его подпись и срок действия, загружает пользователя через CustomUserDetailsService и помещает данные

пользователя в SecurityContext [31]. Далее Spring Security проверяет правила доступа к методу контроллера. Для отдельных операций используются аннотации @PreAuthorize, а для более сложных ограничений применяется AccessControlService [15]. Например, пользователь с ролью MANAGER может изменять сотрудников и подразделения только в рамках своего подразделения и вложенных подразделений.

В приложении не предусмотрена самостоятельная регистрация пользователей. Учетные записи создаются администратором в разделе управления пользователями, после чего пользователь выполняет вход по логину и паролю. Алгоритм авторизации пользователя показан на рисунке 20.

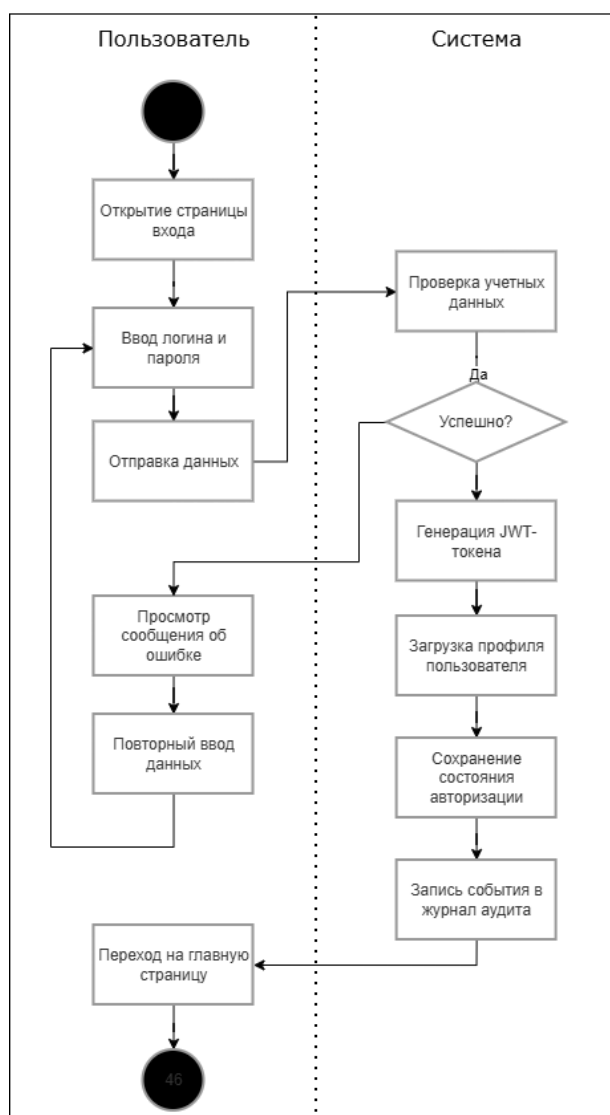


Рисунок 20 – UML-диаграмма деятельности авторизации пользователя

При успешном входе клиент отправляет запрос POST /auth/login, сервер проверяет учетные данные через AuthenticationManager, формирует JWT-токен, записывает событие LOGIN в журнал аудита и возвращает JwtResponse. Клиент сохраняет токен, запрашивает профиль текущего пользователя через /api/profile и открывает защищенную часть приложения. Если введенные данные некорректны, система выводит сообщение об ошибке и пользователь может повторить ввод.

Общий алгоритм открытия защищенного раздела представлен на рисунке 21.

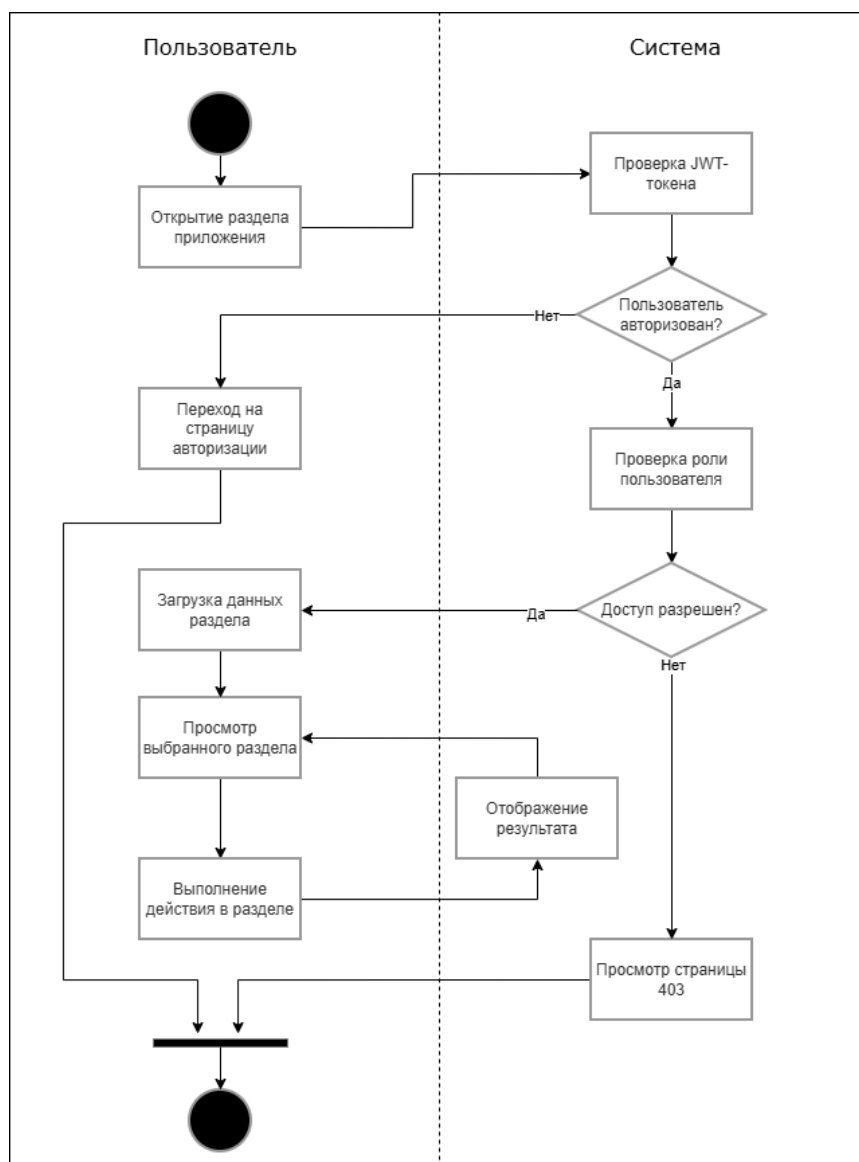


Рисунок 21 – UML-диаграмма деятельности открытия защищенного раздела

Данный алгоритм используется для всех операций, кроме публичного входа в систему и отдачи статических файлов. При переходе в раздел система проверяет

наличие и корректность JWT-токена, затем выполняет проверку роли пользователя. Если пользователь не авторизован, он перенаправляется на страницу входа. Если роль пользователя не соответствует требуемой операции, отображается страница 403 Forbidden. Если проверка пройдена, система загружает данные раздела и позволяет выполнить доступные действия. Для действий, помеченных аннотацией @Auditable, дополнительно формируется запись в таблице audit\_logs.

Одним из основных пользовательских сценариев является поиск сотрудников. Алгоритм поиска приведен на рисунке 22.

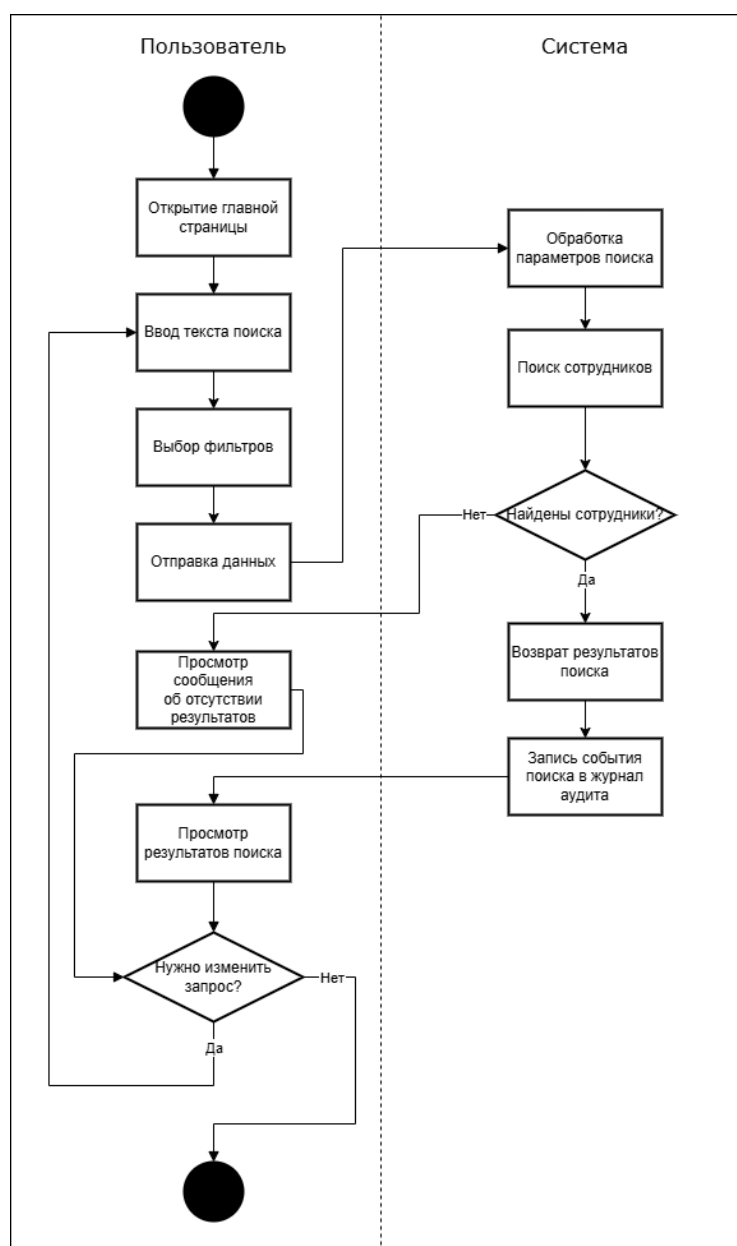


Рисунок 22 – UML-диаграмма деятельности поиска сотрудников

На главной странице клиент собирает строку поиска и фильтры по организации, подразделению, должности и зданию. Параметры сохраняются в URL, что позволяет не терять состояние при обновлении страницы. Далее клиент вызывает GET /api/search. На сервере SearchController передает параметры в SearchService, где строка запроса нормализуется: удаляются лишние пробелы, выполняется перевод к нижнему регистру, выделяются цифры телефона, а также строится альтернативный вариант запроса для поиска по латинице и кириллице. SearchRepository выполняет SQL-запрос к таблицам employees, departments и organizations, рассчитывает ранг совпадения, применяет фильтры и возвращает постраничный результат. После этого AuditAspect фиксирует событие SEARCH, а клиент отображает найденных сотрудников в таблице результатов.

Просмотр и изменение карточки сотрудника выполняются через отдельные endpoint-ы /api/employees. Алгоритм управления карточками сотрудников показан на рисунке 23.

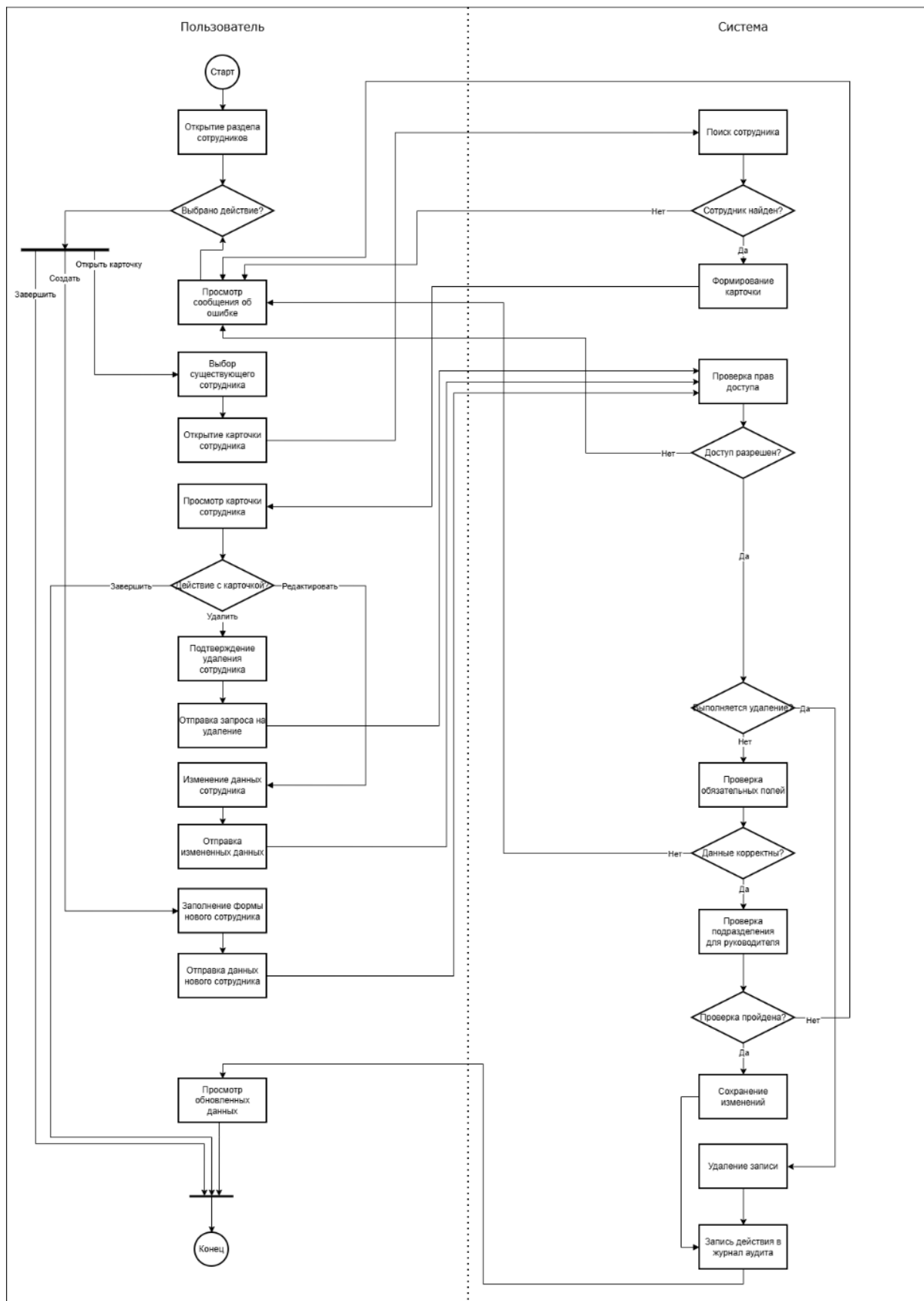


Рисунок 23 – UML-диаграмма деятельности управления карточками сотрудников

При открытии карточки клиент отправляет GET /api/employees/{id}; сервер ищет сотрудника в базе данных, преобразует сущность в EmployeeResponseDto и возвращает данные клиенту. Изменение карточки доступно пользователям с ролями ADMIN и MANAGER. Для операции обновления клиент отправляет PUT /api/employees/{id} с телом EmployeeRequestDto. Сервер проверяет обязательные поля lastName, firstName и departmentId, загружает сотрудника и целевое подразделение, затем выполняет проверку прав. Администратор может изменять любые записи, а руководитель может изменять только сотрудников своего подразделения и дочерних подразделений. После успешной проверки сервис применяет изменения и сохраняет сущность. Создание сотрудника выполняется через POST /api/employees и доступно только администратору, удаление выполняется через DELETE /api/employees/{id} и также доступно только администратору.

Справочники организаций, подразделений, приемных и кодов городов используются для поддержки основных данных телефонного справочника. Алгоритм управления справочниками приведен на рисунке 24.

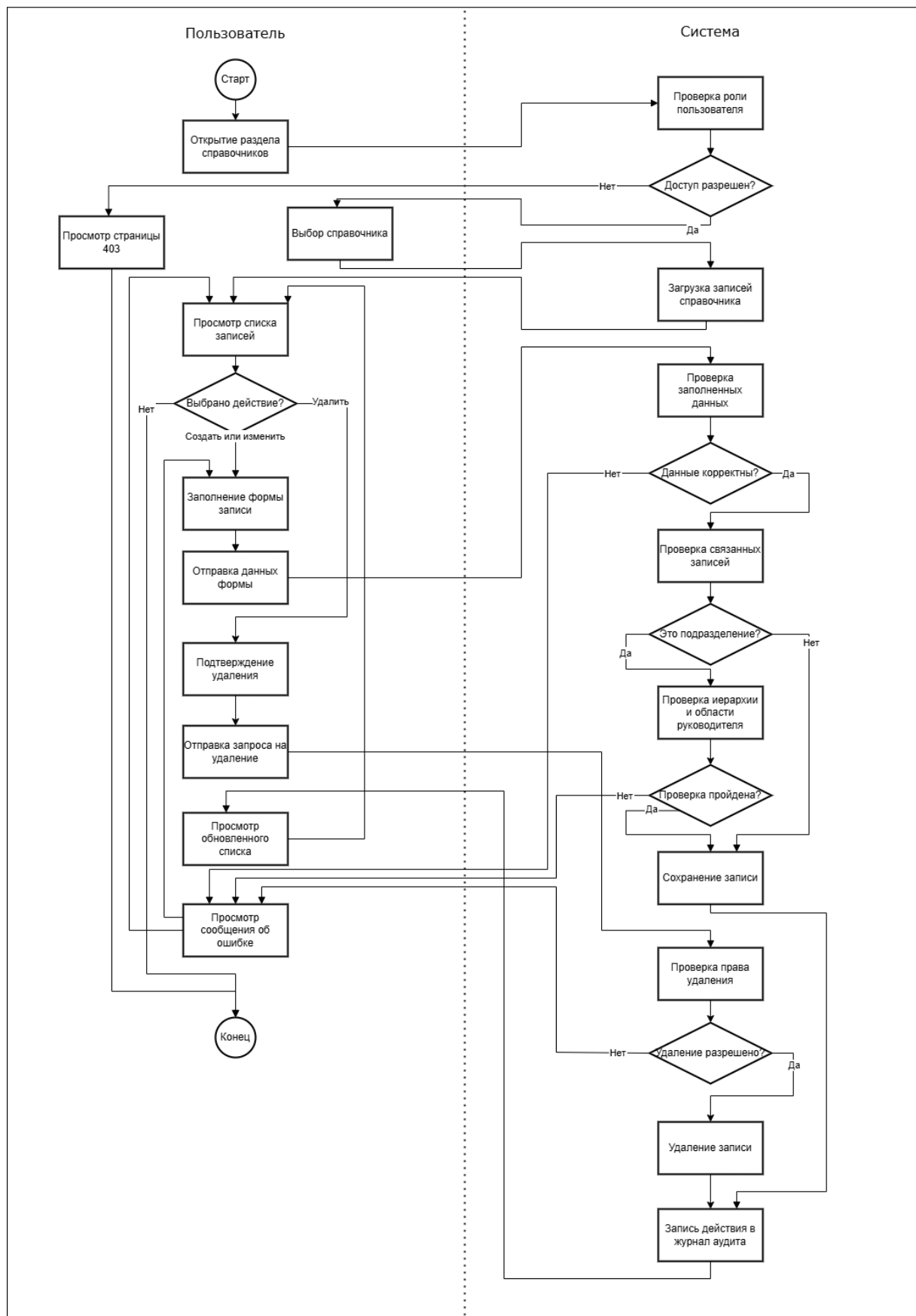


Рисунок 24 – UML-диаграмма деятельности управления справочниками

Клиентская страница AdminDictionariesPage открывает одну из вкладок справочников и загружает список записей соответствующим GET-запросом. При создании или редактировании запись отправляется на сервер в виде DTO. Сервер проверяет роль пользователя, валидирует обязательные поля, загружает связанные сущности и сохраняет изменения. Для подразделений дополнительно проверяется иерархия: подразделение не может быть своим собственным родителем, а пользователь с ролью MANAGER не может создавать корневые подразделения и работать вне зоны своего подразделения. Удаление справочников имеет более строгие ограничения: например, удаление подразделения доступно только администратору.

Для массового наполнения справочника реализован импорт сотрудников из файла Excel. Алгоритм импорта показан на рисунке 25.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 66   |

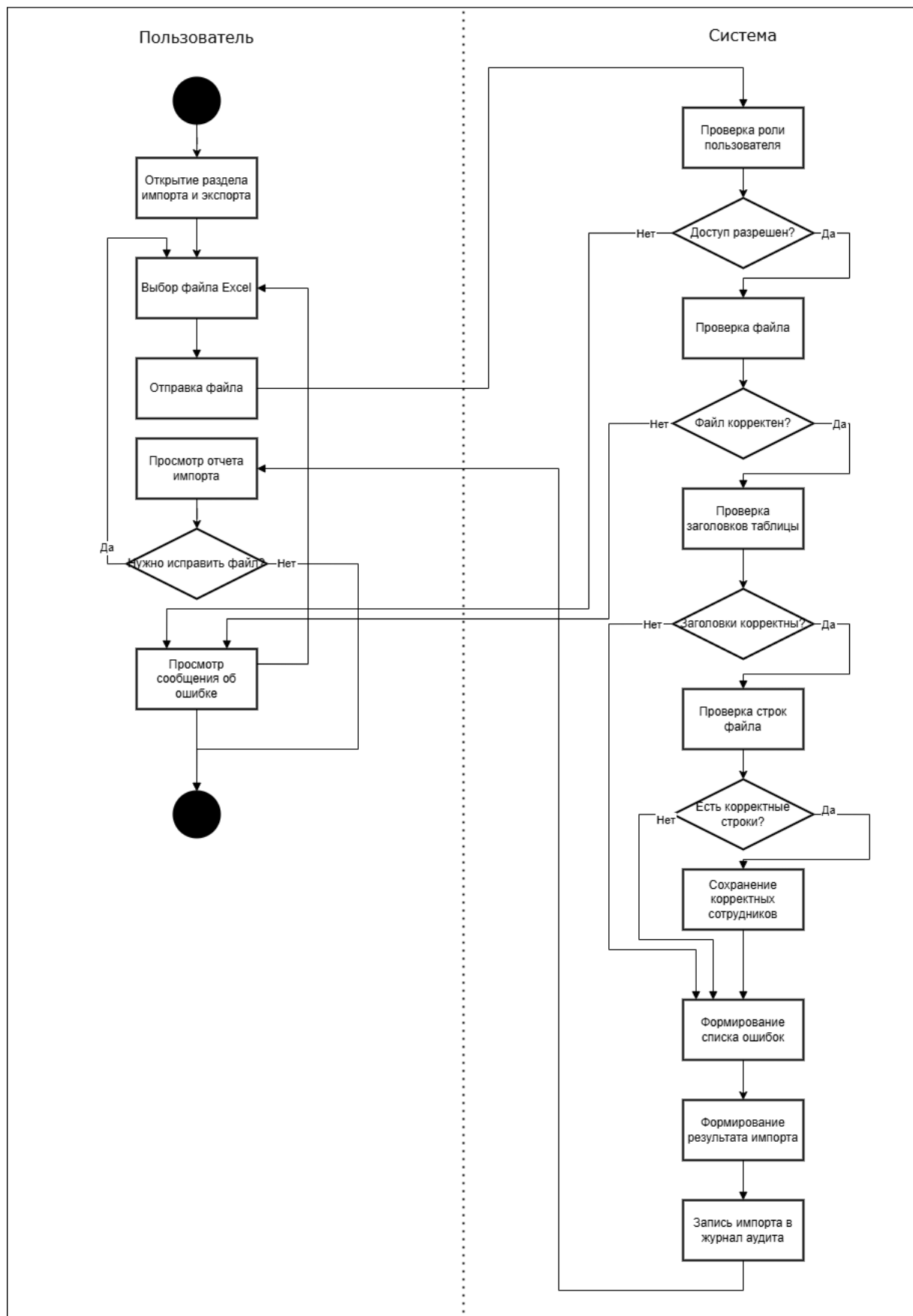


Рисунок 25 – UML-диаграмма деятельности импорта сотрудников из Excel

Клиент выбирает файл формата XLSX и отправляет multipart-запрос POST /api/import/employees. Сервер проверяет роль пользователя ADMIN или MANAGER, проверяет файл на пустоту, размер и расширение, затем открывает первую страницу Excel-файла. EmployeeExcelImporter считывает заголовок, определяет обязательные колонки, проходит по строкам и для каждой строки проверяет фамилию, имя, подразделение, формат электронной почты и телефонов. Работа с XLSX-файлами выполняется с использованием возможностей Apache POI [32]. Если подразделение найдено, дополнительно проверяется право текущего пользователя импортировать сотрудника в это подразделение. Корректные строки сохраняются в таблицу employees, а ошибочные строки попадают в список ошибок. Клиент получает ImportResultDto с количеством успешно загруженных записей и перечнем ошибок.

Выгрузка данных сотрудников выполняется в форматах XLSX и PDF. Алгоритм экспорта представлен на рисунке 26.

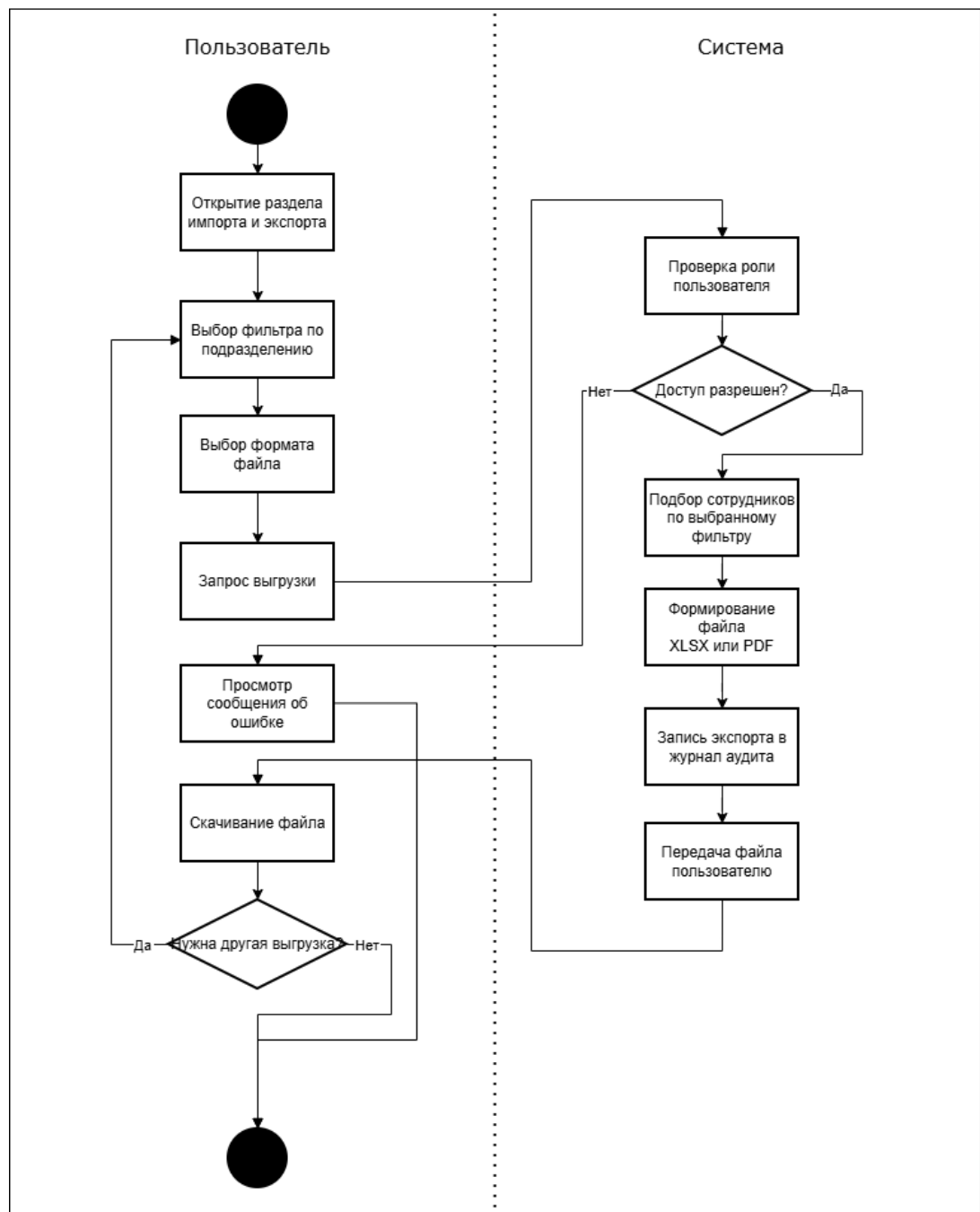


Рисунок 26 – UML-диаграмма деятельности экспорта сотрудников

Клиент выбирает формат и, при необходимости, фильтр по подразделению. Для Excel вызывается GET /api/export/employees, для PDF – GET /api/export/employees/pdf. Сервер проверяет роль пользователя; экспорт доступен пользователям с ролями ADMIN, MANAGER и EMPLOYEE. После этого сервис выбирает сотрудников из базы данных, формирует файл выбранного формата и возвращает его в теле HTTP-ответа с заголовком Content-Disposition. Клиент

получает Blob-объект и запускает скачивание файла. Каждая операция экспорта фиксируется в журнале аудита с указанием формата, количества выгруженных сотрудников и примененного фильтра.

Административные сценарии аудита и управления API-ключами показаны на рисунке 27.

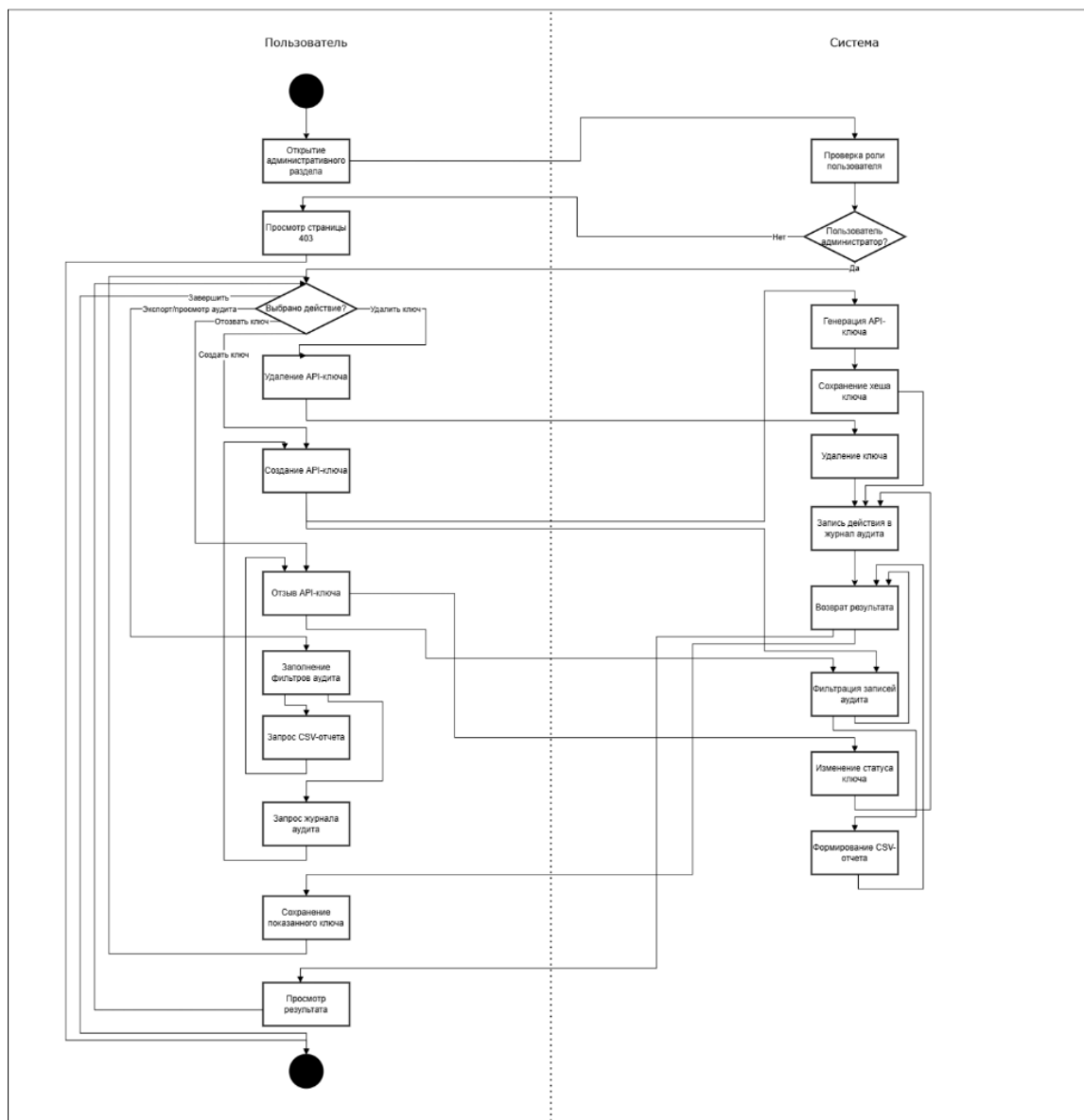


Рисунок 27 – UML-диаграмма деятельности просмотра аудита и управления API-ключами

Раздел аудита доступен только администратору. Клиент отправляет GET /api/audit с фильтрами по пользователю, действию, типу сущности и периоду.

Сервер преобразует параметры даты, применяет фильтры в AuditService и возвращает постраничный список записей. При необходимости администратор может скачать CSV-отчет через GET /api/audit/export. Управление API-ключами также доступно только администратору. При создании ключа клиент отправляет POST /api/integrations/api-keys, сервер генерирует случайное значение ключа, сохраняет в базе только его хеш и возвращает исходное значение ключа только один раз. При отзыве или удалении ключа сервер обновляет или удаляет запись и фиксирует действие в журнале аудита.

Отдельный сценарий связан с отображением организаций на карте. Алгоритм загрузки карты представлен на рисунке 28.

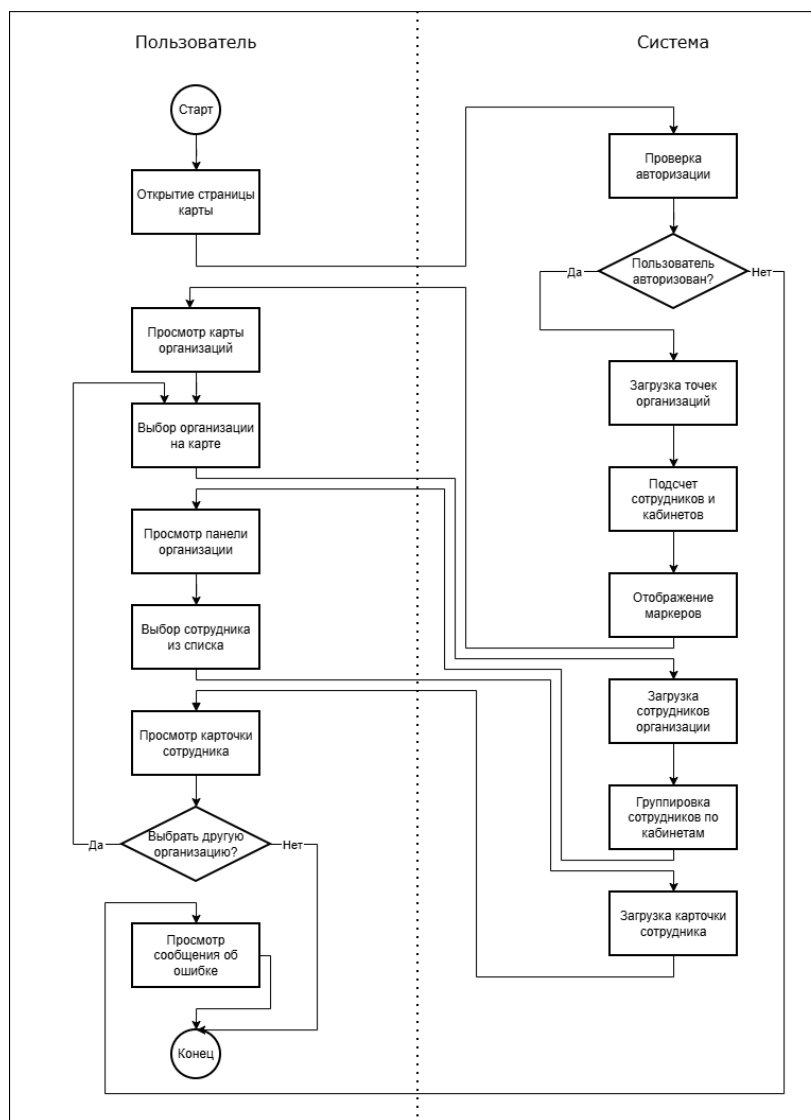


Рисунок 28 – UML-диаграмма деятельности просмотра карты организаций

Клиентская страница карты запрашивает список зданий через GET /api/buildings. Сервер получает организации из таблицы organizations, для каждой организации вычисляет количество сотрудников и количество кабинетов на основе таблицы employees, после чего возвращает список BuildingResponseDto. Клиент отображает точки на карте с помощью React Leaflet, позволяет выбрать организацию и открыть список сотрудников выбранной организации [25].

Таким образом, основные алгоритмы взаимодействия клиента и сервера строятся вокруг единых принципов: клиент формирует запрос через общий HTTP-клиент, сервер проверяет JWT-токен и права доступа, контроллер передает данные в сервисный слой, сервис выполняет бизнес-логику и обращается к репозиториям, а результат возвращается клиенту в подходящем формате [31]. Использование JWT, разграничения ролей, сервисного слоя и журнала аудита позволяет обеспечить контролируемый доступ к данным и прослеживаемость действий пользователей.

## 1.4 Тестирование

### 1.4.1 Выбор функций для составления тест-кейсов

Для проверки работоспособности информационной системы «Единый справочник контактов» необходимо провести функциональное тестирование основных пользовательских сценариев. Система предназначена для централизованного хранения, поиска и сопровождения контактной информации об организациях, подразделениях, приемных и сотрудниках, поэтому тестирование должно подтвердить корректность поиска данных, просмотра карточек, изменения справочников, разграничения доступа и выполнения административных операций.

Разрабатываемое приложение имеет клиент-серверную архитектуру. Пользователь выполняет действия через веб-интерфейс, клиентская часть отправляет запросы к серверному REST API, сервер проверяет авторизацию и права доступа, обрабатывает данные и обращается к базе данных PostgreSQL. При составлении тест-кейсов учитываются не только действия пользователя на

страницах приложения, но и ожидаемый результат обработки запроса: успешное выполнение операции, вывод сообщения об ошибке, ограничение доступа по роли или появление записи в журнале аудита.

Для тестирования выбраны сценарии, которые наиболее полно отражают назначение системы и используются в работе с электронным справочником контактов. Тест-кейсы составлены для следующих действий:

- авторизация пользователя в системе;
- отказ в авторизации при вводе неверных учетных данных;
- поиск сотрудника по ФИО, телефону, должности и дополнительным фильтрам;
- просмотр карточки найденного сотрудника;
- создание новой записи сотрудника администратором;
- редактирование контактных данных сотрудника;
- проверка ограничения прав руководителя при попытке изменить сотрудника вне доступного подразделения;
- обновление пользователем собственных контактных данных в профиле;
- просмотр организационной структуры и списка сотрудников выбранного подразделения;
- импорт списка сотрудников из Excel-файла;
- экспорт справочника сотрудников в форматы Excel и PDF;
- управление учетными записями пользователей администратором;
- просмотр и фильтрация журнала аудита.

Выбранные действия охватывают основные роли системы: администратора, руководителя и сотрудника. Такой набор тест-кейсов позволяет проверить как основные пользовательские функции, так и важные служебные механизмы приложения: авторизацию, разграничение доступа, валидацию данных, импорт, экспорт и аудит действий.

### 1.4.2 Авторизация пользователя

В данном разделе представлен тест-кейс авторизации пользователя в информационной системе «Единый справочник контактов». Проверка включает обработку пустых полей, неверных учетных данных, успешный вход в систему и ограничение доступа к защищенным разделам. Предварительные действия – пользователь перешел на страницу авторизации «/login». В базе данных существует активная учетная запись администратора «admin» с паролем «admin».

Таблица 12 – Тест «Авторизация пользователя»

| Шаг                                                                                                  | Данные                                                     | Ожидаемый результат                                                                                                                                                                     | Фактический результат             |
|------------------------------------------------------------------------------------------------------|------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать на кнопку «Войти», не заполняя поля формы                                                     | Все поля ввода пустые                                      | Под полями отображаются сообщения о необходимости ввода логина и пароля: «Введите логин», «Введите пароль». Запрос авторизации не выполняется.                                          | Совпадает с ожидаемым результатом |
| Ввести неверные учетные данные и нажать кнопку «Войти»                                               | Логин: «admin»<br>Пароль: «wrong-password»                 | На странице отображается сообщение «Неверный логин или пароль». Переход на главную страницу не выполняется, JWT-токен не сохраняется.                                                   | Совпадает с ожидаемым результатом |
| Ввести корректные учетные данные администратора и нажать кнопку «Войти»                              | Логин: «admin»<br>Пароль: «admin»                          | Отображается сообщение «Вход выполнен». Пользователь перенаправляется на главную страницу «Поиск сотрудников». JWT-токен сохраняется на клиенте и используется для защищенных запросов. | Совпадает с ожидаемым результатом |
| Открыть защищенный раздел без авторизации                                                            | Адрес: «/admin/users»<br>JWT-токен отсутствует             | Пользователь перенаправляется на страницу «/login». Содержимое административного раздела не отображается.                                                                               | Совпадает с ожидаемым результатом |
| Авторизоваться пользователем без роли администратора и открыть административный раздел пользователей | Учетная запись с ролью «EMPLOYEE»<br>Адрес: «/admin/users» | Доступ к разделу запрещен. Пользователь перенаправляется на страницу запрета доступа или видит сообщение об отсутствии прав.                                                            | Совпадает с ожидаемым результатом |
| Обновить страницу после успешной авторизации                                                         | Активный JWT-токен сохранен в браузере                     | Система восстанавливает сессию пользователя через запрос профиля. Пользователь остается в защищенной части приложения.                                                                  | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки входа пользователя в систему, обработки ошибок авторизации и защиты маршрутов, доступных только авторизованным пользователям с соответствующей ролью.

#### 1.4.3 Поиск сотрудников и просмотр карточки сотрудника

В данном разделе представлены тест-кейсы поиска сотрудников и просмотра карточки найденного сотрудника. Эти сценарии относятся к основным пользовательским функциям системы, так как справочник предназначен для быстрого поиска контактной информации. Предварительные действия – пользователь авторизован и перешел на главную страницу «Поиск сотрудников». В базе данных есть записи сотрудников, организаций и подразделений.

Таблица 13 – Тест «Поиск сотрудников»

| Шаг                                                       | Данные                             | Ожидаемый результат                                                                                                                                                                      | Фактический результат             |
|-----------------------------------------------------------|------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Оставить строку поиска пустой и нажать кнопку «Найти»     | Поле «Поиск»: пустое               | На экране отображается список сотрудников с пагинацией. В таблице присутствуют колонки «Сотрудник», «Должность», «Подразделение», «Организация», «Контакты», «Расположение», «Действия». | Совпадает с ожидаемым результатом |
| Ввести фамилию сотрудника и нажать кнопку «Найти»         | Поле «Поиск»: «Иванов»             | В таблице отображаются сотрудники, найденные по ФИО. Список результатов соответствует введенному значению.                                                                               | Совпадает с ожидаемым результатом |
| Ввести фамилию транслитом и дождаться выполнения поиска   | Поле «Поиск»: «Ivanov»             | Система учитывает транслитерацию. В результатах отображается сотрудник с фамилией «Иванов», если такая запись есть в базе данных.                                                        | Совпадает с ожидаемым результатом |
| Выполнить поиск по номеру телефона или внутреннему номеру | Поле «Поиск»: «12045»              | В результатах отображаются сотрудники, у которых рабочий, мобильный или внутренний телефон содержит указанные цифры.                                                                     | Совпадает с ожидаемым результатом |
| Задать дополнительные фильтры и нажать кнопку «Найти»     | Организация: выбранная организация | Список результатов ограничивается выбранной организацией, подразделением, должностью и местом размещения. В URL сохраняются параметры поиска.                                            | Совпадает с ожидаемым результатом |

| Шаг                                       | Данные                                                                                          | Ожидаемый результат                                                                                       | Фактический результат             |
|-------------------------------------------|-------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------|-----------------------------------|
|                                           | Подразделение:<br>выбранное подразделение<br>Должность:<br>«инженер»<br>Здание / кабинет: «301» |                                                                                                           |                                   |
| Нажать кнопку «Сбросить»                  | Заполненные поля поиска и фильтров                                                              | Поля формы очищаются, параметры поиска сбрасываются, таблица возвращается к исходному списку сотрудников. | Совпадает с ожидаемым результатом |
| Перейти на следующую страницу результатов | Таблица содержит больше одной страницы                                                          | Отображается следующая страница результатов, номер страницы изменяется в пагинации.                       | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки поиска сотрудников по строке запроса, телефонным данным, транслитерации и дополнительным фильтрам.

Также необходимо проверить открытие карточки пользователя непосредственно из таблицы результатов поиска. Предварительные действия – пользователь авторизован. На странице поиска отображается таблица найденных сотрудников.

Таблица 14 – Тест «Просмотр карточки сотрудников»

| Шаг                                                   | Данные                       | Ожидаемый результат                                                                                                                                                          | Фактический результат             |
|-------------------------------------------------------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать на ФИО сотрудника в таблице результатов        | Любой сотрудник из списка    | Открывается страница «Карточка сотрудника». В адресной строке отображается маршрут «/employees/{id}».                                                                        | Совпадает с ожидаемым результатом |
| Проверить состав данных в карточке                    | Данные выбранного сотрудника | В карточке отображаются ФИО, должность, подразделение, организация, рабочий телефон, внутренний телефон, мобильный телефон, email, кабинет, этаж, адрес и статус активности. | Совпадает с ожидаемым результатом |
| Нажать кнопку «Скопировать ссылку»                    | Открытая карточка сотрудника | Ссылка на карточку копируется в буфер обмена. На экране отображается сообщение «Ссылка скопирована».                                                                         | Совпадает с ожидаемым результатом |
| Нажать кнопку копирования рядом с телефоном или email | Телефон или email сотрудника | Значение копируется в буфер обмена. На экране отображается сообщение «Скопировано».                                                                                          | Совпадает с ожидаемым результатом |

| Шаг                                               | Данные                       | Ожидаемый результат                                                           | Фактический результат             |
|---------------------------------------------------|------------------------------|-------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку «Назад»                             | Открытая карточка сотрудника | Выполняется возврат на предыдущую страницу со списком сотрудников.            | Совпадает с ожидаемым результатом |
| Открыть карточку с несуществующим идентификатором | Адрес: «/employees/999999»   | Отображается сообщение «Сотрудник не найден». Ошибка приложения не возникает. | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки перехода из результатов поиска в карточку сотрудника, корректного отображения контактных данных и вспомогательных действий копирования.

#### 1.4.4 Создание и редактирование сотрудников

В данном разделе представлены тест-кейсы создания и редактирования карточек сотрудников. Проверяются обязательные поля, валидация email, сохранение данных, удаление записи и ограничение прав руководителя. Предварительные действия – пользователь авторизован с ролью «ADMIN» и перешел на страницу «Добавление сотрудника» через кнопку «Добавить сотрудника».

Таблица 15 – Тест Создание сотрудников

| Шаг                                                              | Данные                                                                                                | Ожидаемый результат                                                                                                                   | Фактический результат             |
|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку «Сохранить», не заполняя обязательные поля         | Все поля ввода пустые                                                                                 | Под обязательными полями «Фамилия», «Имя» и «Подразделение» отображаются сообщения валидации. Карточка сотрудника не создается.       | Совпадает с ожидаемым результатом |
| Ввести некорректный email и нажать кнопку «Сохранить»            | Email: «test-email»                                                                                   | Под полем email отображается сообщение «Некорректный email». Сохранение не выполняется.                                               | Совпадает с ожидаемым результатом |
| Заполнить основные данные сотрудника и нажать кнопку «Сохранить» | Фамилия: «Петров»<br>Имя: «Алексей»<br>Должность: «Инженер»<br>Подразделение: выбранное подразделение | На экране отображается сообщение «Карточка сотрудника создана». Пользователь перенаправляется на страницу карточки нового сотрудника. | Совпадает с ожидаемым результатом |

| Шаг                                            | Данные                                                                                                                           | Ожидаемый результат                                                                                             | Фактический результат             |
|------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|-----------------------------------|
|                                                | Рабочий телефон: «+7 (495) 120-45-99»<br>Внутренний телефон: «4599»<br>Email: «petrov.test@npo-alliance.local»<br>Кабинет: «301» |                                                                                                                 |                                   |
| Проверить наличие созданной записи через поиск | Поиск: «Петров»                                                                                                                  | Новый сотрудник отображается в таблице результатов поиска. Данные в строке совпадают с введенными при создании. | Совпадает с ожидаемым результатом |
| Нажать кнопку «Отмена» при создании сотрудника | Открыта форма добавления сотрудника                                                                                              | Пользователь возвращается на главную страницу поиска сотрудников. Новая запись не создается.                    | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки формы добавления сотрудника, обязательных полей, валидации контактных данных и появления новой записи в справочнике.

Надо также проверить редактирование сотрудника. Предварительные действия – пользователь авторизован с ролью «ADMIN». В системе существует карточка сотрудника, доступная для редактирования.

Таблица 16 – Тест Редактирование сотрудников

| Шаг                                                         | Данные                                                                                | Ожидаемый результат                                                                                                                                      | Фактический результат             |
|-------------------------------------------------------------|---------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Открыть карточку сотрудника и нажать кнопку «Редактировать» | Существующий сотрудник                                                                | Открывается страница «Редактирование сотрудника» с заполненными полями текущей карточки.                                                                 | Совпадает с ожидаемым результатом |
| Изменить контактные данные и нажать кнопку «Сохранить»      | Рабочий телефон: «+7 (495) 120-45-77»<br>Внутренний телефон: «4577»<br>Кабинет: «407» | На экране отображается сообщение «Карточка сотрудника сохранена». Пользователь возвращается на карточку сотрудника, где отображаются обновленные данные. | Совпадает с ожидаемым результатом |
| Нажать кнопку «Сбросить изменения»                          | Изменены значения формы,                                                              | Значения формы возвращаются к исходным данным сотрудника.                                                                                                | Совпадает с ожидаемым результатом |

| Шаг                                                                                    | Данные                                                                                          | Ожидаемый результат                                                                                                                                             | Фактический результат             |
|----------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| после изменения полей                                                                  | сохранение не выполнено                                                                         |                                                                                                                                                                 |                                   |
| Авторизоваться руководителем и открыть сотрудника из своего подразделения              | Роль: «MANAGER»<br>Сотрудник относится к подразделению руководителя или дочернему подразделению | Кнопка редактирования доступна. При сохранении изменений данные сотрудника обновляются.                                                                         | Совпадает с ожидаемым результатом |
| Авторизоваться руководителем и попытаться изменить сотрудника вне зоны ответственности | Роль: «MANAGER»<br>Сотрудник относится к другому подразделению                                  | Система запрещает сохранение изменений. На сервере возвращается ошибка доступа, данные сотрудника не изменяются.                                                | Совпадает с ожидаемым результатом |
| Нажать кнопку «Удалить» на карточке сотрудника и подтвердить удаление                  | Сотрудник, которого можно удалить                                                               | После подтверждения отображается сообщение «Сотрудник удален». Пользователь возвращается на страницу поиска, удаленный сотрудник не отображается в результатах. | Совпадает с ожидаемым результатом |
| Нажать кнопку «Удалить», затем отменить подтверждение                                  | Окно подтверждения «Удалить сотрудника?»                                                        | Удаление не выполняется. Карточка сотрудника остается доступной.                                                                                                | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки изменения карточки сотрудника, сброса формы, удаления записи и ограничения действий руководителя рамками своего подразделения.

#### 1.4.5 Профиль пользователя и организационная структура

В данном разделе представлены тест-кейсы работы с профилем пользователя и просмотра организационной структуры. Проверяется отображение данных профиля, изменение собственных контактов, смена пароля и просмотр сотрудников выбранного подразделения. Предварительные действия – пользователь авторизован и перешел в раздел «Профиль».

Таблица 17 – Тест Профиль пользователя

| Шаг                                                                                          | Данные                                                                                                                                                                                 | Ожидаемый результат                                                                                                                                                                                                                                              | Фактический результат             |
|----------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Открыть вкладку «Профиль»                                                                    | Авторизованный пользователь                                                                                                                                                            | Отображаются логин, роль, ФИО, должность, подразделение, организация, кабинет и email пользователя.                                                                                                                                                              | Совпадает с ожидаемым результатом |
| Открыть вкладку «Доступ»                                                                     | Авторизованный пользователь                                                                                                                                                            | Отображается роль пользователя, зона ответственности и список доступных разрешений. Для администратора указано «Все организации и подразделения»; для руководителя – подразделение или организация; для сотрудника – собственный профиль и просмотр справочника. | Совпадает с ожидаемым результатом |
| Открыть вкладку «Контакты», изменить контактные данные и нажать кнопку «Сохранить изменения» | Рабочий телефон: «+7 (495) 120-45-88»<br>Внутренний телефон: «4588»<br>Кабинет: «508»<br>Ссылка на фото: « <a href="https://example.com/photo.png">https://example.com/photo.png</a> » | На экране отображается сообщение «Профиль обновлен». Новые значения сохраняются и отображаются в профиле пользователя.                                                                                                                                           | Совпадает с ожидаемым результатом |
| Открыть вкладку «Безопасность» и нажать «Сменить пароль», не заполняя обязательные поля      | Поля «Текущий пароль» и «Новый пароль» пустые                                                                                                                                          | Под полями отображаются сообщения «Введите текущий пароль» и «Введите новый пароль». Пароль не изменяется.                                                                                                                                                       | Совпадает с ожидаемым результатом |
| Ввести неверный текущий пароль и нажать «Сменить пароль»                                     | Текущий пароль: неверный<br>Новый пароль: «newPassword123»                                                                                                                             | Система не изменяет пароль и отображает ошибку проверки текущего пароля.                                                                                                                                                                                         | Совпадает с ожидаемым результатом |
| Ввести корректный текущий пароль и новый пароль, нажать «Сменить пароль»                     | Текущий пароль: корректный<br>Новый пароль: «newPassword123»                                                                                                                           | На экране отображается сообщение «Пароль изменен». При следующей авторизации пользователь может войти с новым паролем.                                                                                                                                           | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки просмотра профиля, прав доступа, самостоятельного обновления контактных данных и смены пароля.

Также необходима проверка для организационной структуры. Предварительные действия – пользователь авторизован и перешел в раздел «Оргструктура». В системе есть организации, подразделения и сотрудники.

Таблица 18 – Тест Организационная структура

| Шаги                                            | Данные                         | Ожидаемый результат                                                                                                                                  | Фактический результат             |
|-------------------------------------------------|--------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Открыть раздел «Оргструктура»                   | Авторизованный пользователь    | На странице отображается дерево подразделений и таблица организаций с колонками «Организация», «Адрес», «Телефон».                                   | Совпадает с ожидаемым результатом |
| Выбрать подразделение в дереве подразделений    | Любое подразделение из списка  | Справа отображается таблица сотрудников выбранного подразделения с колонками «Сотрудник», «Должность», «Телефон», «Кабинет».                         | Совпадает с ожидаемым результатом |
| Выбрать другое подразделение                    | Другое подразделение из дерева | Таблица сотрудников обновляется и показывает записи, относящиеся к выбранному подразделению.                                                         | Совпадает с ожидаемым результатом |
| Открыть раздел пользователем с ролью «ADMIN»    | Роль: «ADMIN»                  | На странице доступны кнопки «Добавить организацию», «Добавить подразделение» и «Добавить сотрудника».                                                | Совпадает с ожидаемым результатом |
| Открыть раздел пользователем с ролью «MANAGER»  | Роль: «MANAGER»                | На странице доступна кнопка «Добавить подразделение». Действия ограничиваются зоной ответственности руководителя.                                    | Совпадает с ожидаемым результатом |
| Открыть раздел пользователем с ролью «EMPLOYEE» | Роль: «EMPLOYEE»               | Пользователь может просматривать организационную структуру, но не видит административные кнопки добавления организаций, подразделений и сотрудников. | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки просмотра организационной структуры и различий интерфейса в зависимости от роли пользователя.

#### 1.4.6 Импорт и экспорт данных

В данном разделе представлены тест-кейсы импорта и экспорта справочника сотрудников. Проверяется скачивание шаблона, загрузка Excel-файла, отображение ошибок импорта и выгрузка данных в форматы Excel и PDF. Предварительные действия – пользователь авторизован и имеет право импорта сотрудников. Пользователь перешел в раздел импорта и экспорта данных.

Таблица 19 – Тест Проверка импорта

| Шаги                                                      | Данные                                                           | Ожидаемый результат                                                                                                                                                                       | Фактический результат             |
|-----------------------------------------------------------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку «Скачать шаблон»                            | Нет дополнительных данных                                        | Браузер скачивает Excel-шаблон для импорта сотрудников. Ошибка загрузки не отображается.                                                                                                  | Совпадает с ожидаемым результатом |
| Нажать область загрузки и выбрать корректный «.xlsx» файл | Excel-файл с корректными строками сотрудников                    | Файл отправляется на сервер. После обработки отображается сообщение вида «Импортировано: N, ошибок: 0». Новые сотрудники доступны через поиск.                                            | Совпадает с ожидаемым результатом |
| Перетащить корректный «.xlsx» файл в область загрузки     | Excel-файл с корректными строками сотрудников                    | Импорт выполняется аналогично выбору файла через диалог. Отображается количество успешно импортированных записей.                                                                         | Совпадает с ожидаемым результатом |
| Загрузить Excel-файл с ошибками в строках                 | Файл содержит пустые обязательные поля или некорректные значения | Система импортирует только корректные строки. На странице отображается предупреждение «Ошибки импорта» и список найденных ошибок. В сообщении указано количество успешных строк и ошибок. | Совпадает с ожидаемым результатом |
| Загрузить файл неподдерживаемого формата                  | Файл не в формате «.xlsx»                                        | Файл не принимается как корректный файл импорта. Импорт сотрудников не выполняется.                                                                                                       | Совпадает с ожидаемым результатом |
| Повторно найти сотрудника из импортированного файла       | ФИО или email импортированного сотрудника                        | Импортированный сотрудник отображается в результатах поиска с данными из Excel-файла.                                                                                                     | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки загрузки сотрудников из Excel-файла, обработки корректных данных и отображения ошибок импорта.

Также необходимо проверить экспорт. Предварительные действия – пользователь авторизован и имеет право экспорта сотрудников. В справочнике есть записи сотрудников.

Таблица 20 – Тест Проверка экспорта

| Шаги                                                         | Данные                      | Ожидаемый результат                                                                                              | Фактический результат             |
|--------------------------------------------------------------|-----------------------------|------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку «Экспорт Excel» на странице поиска сотрудников | Нет дополнительных фильтров | Браузер скачивает файл справочника в формате Excel. Файл содержит записи сотрудников и основные контактные поля. | Совпадает с ожидаемым результатом |

| Шаги                                                             | Данные                                 | Ожидаемый результат                                                                                                                                          | Фактический результат             |
|------------------------------------------------------------------|----------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку «Экспорт PDF» на странице поиска сотрудников       | Нет дополнительных фильтров            | Браузер скачивает PDF-отчет со справочником сотрудников. Файл открывается без ошибок.                                                                        | Совпадает с ожидаемым результатом |
| Выбрать подразделение в фильтре поиска и выполнить экспорт Excel | Подразделение: выбранное подразделение | Скачанный Excel-файл содержит сотрудников выбранного подразделения.                                                                                          | Совпадает с ожидаемым результатом |
| Выбрать подразделение в фильтре поиска и выполнить экспорт PDF   | Подразделение: выбранное подразделение | Скачанный PDF-файл содержит сотрудников выбранного подразделения.                                                                                            | Совпадает с ожидаемым результатом |
| Выполнить экспорт пользователем с ролью «EMPLOYEE»               | Роль: «EMPLOYEE»                       | Пользователь может выполнить доступный ограниченный экспорт согласно своим правам. Административные операции импорта и управления пользователями недоступны. | Совпадает с ожидаемым результатом |
| Выполнить экспорт при отсутствии авторизации                     | JWT-токен отсутствует                  | Запрос экспорта не выполняется, пользователь перенаправляется на страницу авторизации или получает ошибку доступа.                                           | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки формирования файлов экспорта, учета фильтра подразделения и контроля доступа к операциям выгрузки.

#### 1.4.7 Тестирование функционала администратора

В данном разделе представлены тест-кейсы административных функций системы: управление пользователями, справочниками, журналом аудита и API-ключами. Данные функции доступны пользователю с ролью «ADMIN» и относятся к критичным сценариям сопровождения справочника. Предварительные действия – пользователь авторизован с ролью «ADMIN» и перешел в раздел «Пользователи».

Таблица 21 – Тест Администрирование пользователей

| Шаги                                 | Данные                    | Ожидаемый результат                                                                                    | Фактический результат             |
|--------------------------------------|---------------------------|--------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку «Создать пользователя» | Нет дополнительных данных | Открывается боковая форма «Создание пользователя» с полями «Логин», «Пароль», «Роль», «ID сотрудника». | Совпадает с ожидаемым результатом |

| Шаги                                                          | Данные                                                                                                               | Ожидаемый результат                                                                                               | Фактический результат             |
|---------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Нажать «Сохранить», не заполняя обязательные поля             | Все поля ввода пустые                                                                                                | Под обязательными полями отображаются сообщения валидации. Пользователь не создается.                             | Совпадает с ожидаемым результатом |
| Заполнить форму создания пользователя и нажать «Сохранить»    | Логин: «test.user»<br>Пароль: «test12345»<br>Роль: «Сотрудник»<br>ID сотрудника: существующий ID или пустое значение | На экране отображается сообщение «Пользователь создан». Новая учетная запись появляется в таблице пользователей.  | Совпадает с ожидаемым результатом |
| Выполнить поиск пользователя по username                      | Поле поиска: «test.user»                                                                                             | В таблице отображается созданная учетная запись. Остальные пользователи скрываются фильтром.                      | Совпадает с ожидаемым результатом |
| Отфильтровать пользователей по роли и статусу                 | Роль: «Сотрудник»<br>Статус: «Активен»                                                                               | Таблица отображает только пользователей, соответствующих выбранной роли и статусу.                                | Совпадает с ожидаемым результатом |
| Нажать кнопку смены пароля, ввести новый пароль и подтвердить | Новый пароль: «newUserPassword123»                                                                                   | Отображается сообщение «Пароль изменен». Пользователь может войти с новым паролем.                                | Совпадает с ожидаемым результатом |
| Нажать кнопку блокировки пользователя                         | Активный пользователь                                                                                                | Статус пользователя изменяется на «Заблокирован». Заблокированный пользователь не может выполнить вход в систему. | Совпадает с ожидаемым результатом |
| Нажать кнопку удаления пользователя и подтвердить удаление    | Пользователь «test.user»                                                                                             | После подтверждения пользователь удаляется из списка.                                                             | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки создания, поиска, фильтрации, смены пароля, блокировки и удаления учетных записей пользователей.

Также следует проверить администрирование раздела «Справочники». Предварительные действия – пользователь авторизован с ролью «ADMIN» и перешел в раздел «Справочники».

Таблица 22 – Тест Администрирование справочников

| Шаги                                                          | Данные                                                                                                                                                          | Ожидаемый результат                                                                                                                 | Фактический результат             |
|---------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Открыть вкладку «Организации»                                 | Нет дополнительных данных                                                                                                                                       | Отображается таблица организаций с колонками «Название», «Короткое», «Адрес», «Телефон», «Email», «Сайт», «Координаты», «Действия». | Совпадает с ожидаемым результатом |
| Нажать кнопку «Добавить запись» на вкладке организаций        | Нет дополнительных данных                                                                                                                                       | Открывается форма «Добавить организацию».                                                                                           | Совпадает с ожидаемым результатом |
| Нажать «Сохранить», не заполняя обязательное поле             | Название: пустое                                                                                                                                                | Под полем «Название» отображается сообщение валидации. Запись не сохраняется.                                                       | Совпадает с ожидаемым результатом |
| Заполнить данные организации и нажать «Сохранить»             | Название: «Тестовая организация»<br>Короткое название: «ТО»<br>Адрес: «г. Пенза, ул. Тестовая, 1»<br>Телефон: «+7 (8412) 00-00-00»<br>Email: «test@example.com» | На экране отображается сообщение «Запись сохранена». Организация появляется в таблице.                                              | Совпадает с ожидаемым результатом |
| Открыть созданную запись на редактирование и изменить телефон | Телефон: «+7 (8412) 11-11-11»                                                                                                                                   | После сохранения отображается сообщение «Запись сохранена». В таблице отображается обновленный телефон.                             | Совпадает с ожидаемым результатом |
| Перейти на вкладку «Подразделения» и добавить подразделение   | Название: «Тестовый отдел»<br>Организация: «Тестовая организация»                                                                                               | Подразделение сохраняется и отображается в таблице подразделений.                                                                   | Совпадает с ожидаемым результатом |
| Перейти на вкладку «Приемные» и добавить приемную             | Название: «Приемная директора»<br>Организация: «Тестовая организация»<br>Основной телефон: «+7 (8412) 22-22-22»                                                 | Приемная сохраняется и отображается в таблице приемных.                                                                             | Совпадает с ожидаемым результатом |
| Перейти на вкладку «Коды городов» и добавить код              | Регион: «Пензенская область»<br>Город: «Пенза»<br>Код города: «8412»                                                                                            | Код города сохраняется и отображается в таблице кодов городов.                                                                      | Совпадает с ожидаемым результатом |

| Шаги                                                 | Данные                            | Ожидаемый результат                                                                  | Фактический результат             |
|------------------------------------------------------|-----------------------------------|--------------------------------------------------------------------------------------|-----------------------------------|
| Нажать кнопку удаления записи и подтвердить удаление | Любая тестовая запись справочника | Отображается сообщение «Запись удалена». Запись исчезает из соответствующей таблицы. | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки создания, редактирования и удаления записей основных справочников системы.

Необходима также проверка раздела «Аудит». Предварительные действия – пользователь авторизован с ролью «ADMIN» и перешел в раздел «Аудит». В системе предварительно выполнены действия поиска, просмотра и изменения сотрудников.

Таблица 23 – Тест Администрирование аудита

| Шаг                                    | Данные                                  | Ожидаемый результат                                                                                                             | Фактический результат             |
|----------------------------------------|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Открыть раздел «Аудит»                 | Нет дополнительных данных               | Отображается таблица аудита с колонками «Дата», «Пользователь», «Действие», «Сущность», «ID», «IP адрес», «Детали», «Действия». | Совпадает с ожидаемым результатом |
| Применить фильтр по пользователю       | Пользователь: «admin»                   | В таблице отображаются записи аудита, относящиеся к указанному пользователю.                                                    | Совпадает с ожидаемым результатом |
| Применить фильтр по действию           | Действие: «Поиск»                       | В таблице отображаются только события поиска сотрудников.                                                                       | Совпадает с ожидаемым результатом |
| Применить фильтр по дате или периоду   | Дата: текущая дата или выбранный период | В таблице отображаются только события, попадающие в указанный период.                                                           | Совпадает с ожидаемым результатом |
| Нажать кнопку «Детали» у записи аудита | Любая запись аудита                     | Открывается модальное окно «Детали события» с датой, пользователем, действием, сущностью, ID сущности, IP-адресом и деталями.   | Совпадает с ожидаемым результатом |
| Нажать кнопку «Экспорт отчёта»         | Текущие фильтры журнала аудита          | Браузер скачивает отчет аудита с учетом примененных фильтров.                                                                   | Совпадает с ожидаемым результатом |
| Нажать кнопку «Сбросить» в фильтрах    | Активные фильтры                        | Поля фильтрации очищаются, таблица возвращается к общему списку событий.                                                        | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки просмотра журнала аудита, фильтрации событий, просмотра подробностей и экспорта отчета.

Проверка раздела «Интеграции». Предварительные действия – пользователь авторизован с ролью «ADMIN» и перешел в раздел «Интеграции».

Таблица 24 – Тест Администрирование интеграций

| Шаг                                         | Данные                                                       | Ожидаемый результат                                                                                                                     | Фактический результат             |
|---------------------------------------------|--------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Открыть вкладку «API-ключи»                 | Нет дополнительных данных                                    | Отображается таблица API-ключей с колонками «Название», «Префикс», «Доступ», «Статус», «Создан», «Последнее использование», «Действия». | Совпадает с ожидаемым результатом |
| Нажать кнопку «Создать ключ»                | Нет дополнительных данных                                    | Открывается модальное окно «Создание API-ключа» с полями «Название» и «Доступ».                                                         | Совпадает с ожидаемым результатом |
| Нажать «Создать», не заполняя поле названия | Название: пустое                                             | Под полем отображается сообщение «Введите название ключа». API-ключ не создается.                                                       | Совпадает с ожидаемым результатом |
| Заполнить данные ключа и нажать «Создать»   | Название: «Корпоративный портал»<br>Доступ: «read:employees» | Отображается сообщение «API-ключ создан». Открывается окно «API-ключ создан» с предупреждением о необходимости скопировать ключ.        | Совпадает с ожидаемым результатом |
| Скопировать созданный ключ и закрыть окно   | Сгенерированный API-ключ                                     | Ключ копируется в буфер обмена. После закрытия окна полный ключ больше не отображается, в таблице остается только префикс.              | Совпадает с ожидаемым результатом |
| Отозвать активный API-ключ                  | Существующий активный ключ                                   | После подтверждения отображается сообщение «API-ключ отозван». Статус ключа меняется на «Отозван».                                      | Совпадает с ожидаемым результатом |
| Удалить API-ключ                            | Существующий ключ                                            | После подтверждения отображается сообщение «API-ключ удален». Ключ исчезает из таблицы.                                                 | Совпадает с ожидаемым результатом |

Тест-кейс предназначен для проверки жизненного цикла API-ключа: создания, однократного отображения полного значения, отзыва и удаления.

## 1.5 Руководство пользователя

Информационная система «Единый справочник контактов» предназначена для ведения и использования корпоративного телефонного справочника. Система

позволяет искать сотрудников, просматривать контактные данные, работать с организационной структурой, отображать организации на карте, выгружать справочник в файлы, импортировать сотрудников из Excel и выполнять административное сопровождение данных.

Руководство описывает функции, доступные пользователям системы, и отдельно фиксирует ограничения: какие действия выполняются только при наличии соответствующей роли, а какие возможности в текущей версии не реализованы.

### 1.5.1 Результат работы приложения

После запуска пользователь работает с многостраничным веб-приложением. Для входа используется отдельная страница авторизации. Она отображена на рисунке 29.

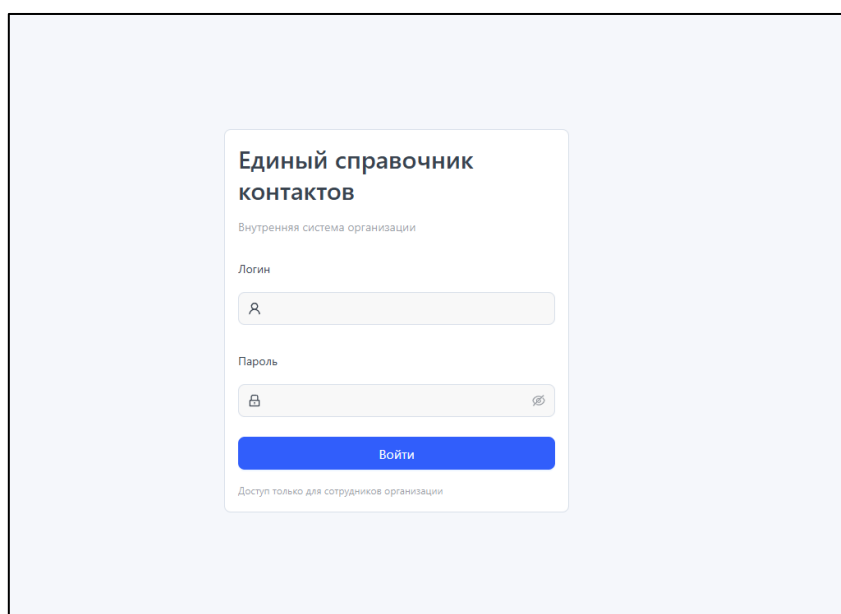


Рисунок 29 – Страница авторизации пользователя

После успешного входа открывается защищенная часть системы, главный экран, с боковым меню и верхней панелью. Верхняя панель содержит кнопку сворачивания бокового меню, строку быстрого поиска по ФИО, телефону или кабинету, подсказку горячей клавиши Ctrl+K и меню пользователя.

Основные разделы приложения:

- «Поиск» – поиск сотрудников и просмотр результатов в таблице.
- «Оргструктура» – просмотр организаций, подразделений и сотрудников выбранного подразделения.
- «Карта» – отображение организаций и зданий на интерактивной карте.
- «Профиль» – просмотр данных текущего пользователя, его прав, контактов и смена пароля.
- «Администрирование» – разделы для управления пользователями, справочниками, импортом/экспортом, интеграциями и журналом аудита.

Интерфейс главной страницы приложения показан на рисунке 30.

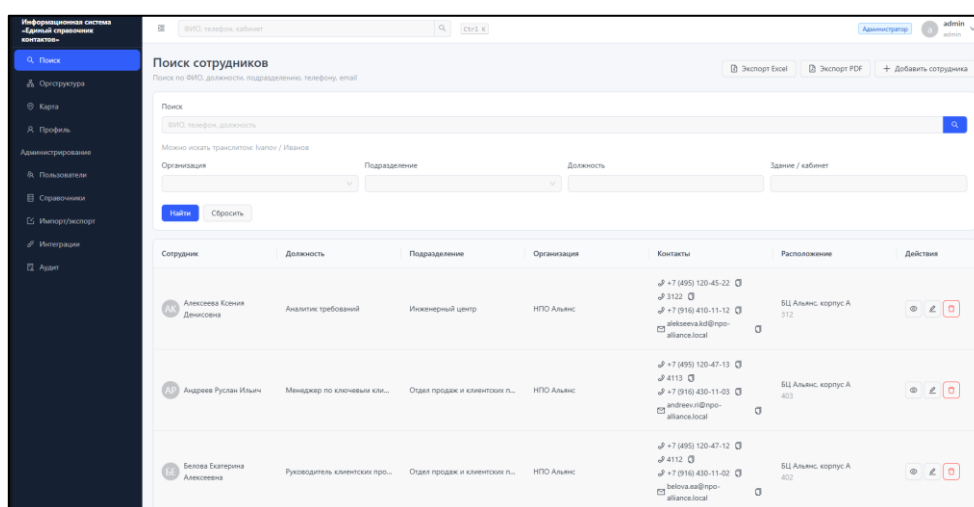


Рисунок 30 – Главный экран системы после входа

Через меню пользователя можно перейти в профиль, сменить пароль, переключить светлую или темную тему и выйти из системы. Меню пользователя представлено на рисунке 31.

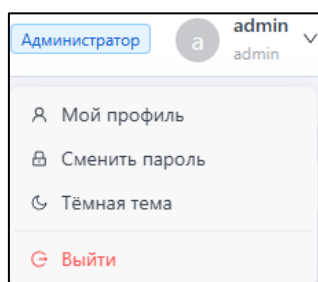


Рисунок 31 – Меню текущего пользователя

### 1.5.2 Роли пользователей и разграничение доступа

В системе используются три роли: ADMIN, MANAGER и EMPLOYEE. Набор доступных пунктов меню и действий зависит от роли пользователя.

Если пользователь пытается открыть раздел без авторизации, система перенаправляет его на страницу входа. Если пользователь авторизован, но его роль не подходит для выбранного раздела, отображается страница запрета доступа. Это описано на рисунке 32.

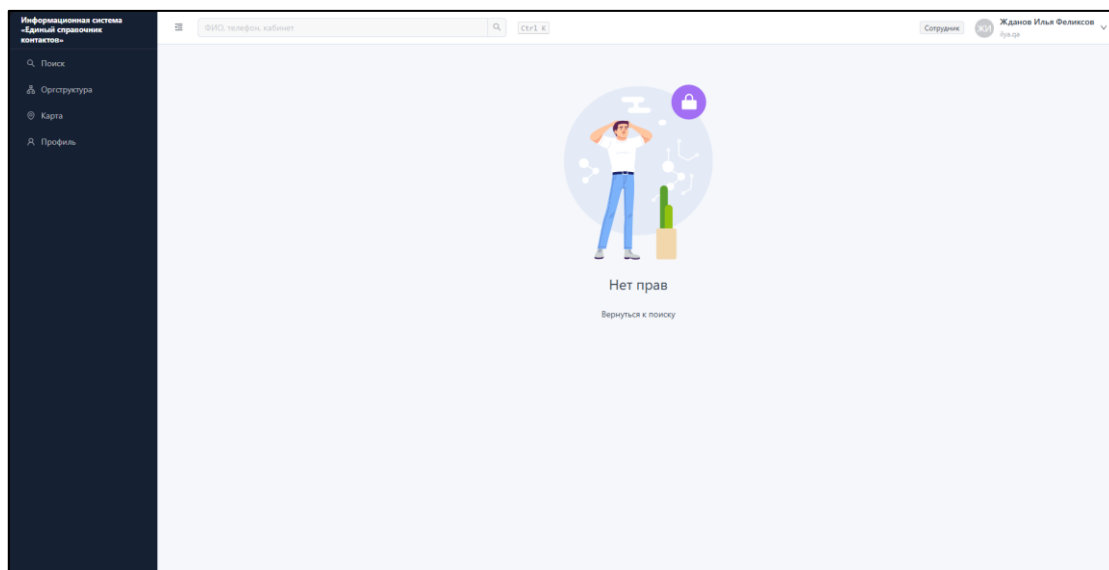


Рисунок 32 – Страница запрета доступа при недостаточных правах

Система не предусматривает самостоятельную регистрацию новых пользователей. Учетную запись создает администратор в разделе «Пользователи».

### 1.5.3 Авторизация и завершение работы

Для входа пользователь на странице авторизации вводит логин и пароль и нажимает кнопку «Войти». При пустых полях система показывает сообщения «Введите логин» и «Введите пароль». При неверных данных отображается сообщение «Неверный логин или пароль». Возможные сообщения при авторизации отражены на рисунках 33, 34.

**Единый справочник контактов**  
Внутренняя система организации

Логин

Введите логин

Пароль

Введите пароль

**Войти**

Доступ только для сотрудников организации

Рисунок 33 – Валидация формы входа при пустых полях

**Единый справочник контактов**  
Внутренняя система организации

Неверный логин или пароль

Логин

Пароль

**Войти**

Доступ только для сотрудников организации

Рисунок 34 – Сообщение об ошибке при неверном логине или пароле

При успешной авторизации система загружает профиль текущего пользователя и открывает главную страницу поиска сотрудников. После обновления страницы состояние пользователя восстанавливается автоматически через запрос профиля.

Для выхода из системы нужно открыть меню пользователя в правой части верхней панели и выбрать пункт «Выйти». После выхода пользователь возвращается на страницу авторизации. Выход из системы показан на рисунке 35.

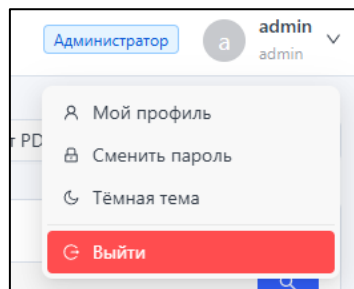


Рисунок 35 – Пункт выхода из системы в меню пользователя

#### 1.5.4 Поиск сотрудников

Раздел «Поиск» является основной рабочей страницей справочника. На странице отображается форма поиска, кнопки экспорта и таблица найденных сотрудников. Страница поиска сотрудников показана на рисунке 36.

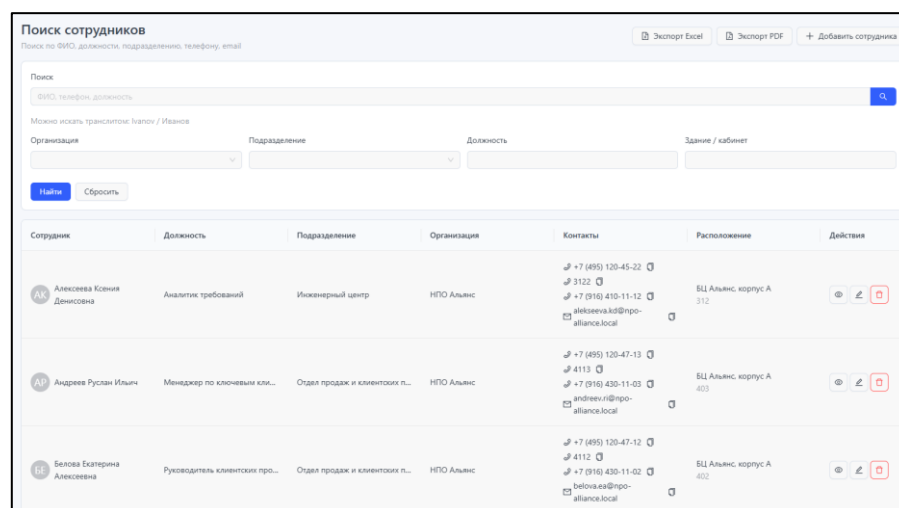


Рисунок 36 – Страница поиска сотрудников

Поиск может выполняться по следующим данным:

- ФИО сотрудника;
- должность;
- подразделение;
- организация;
- email;
- рабочий, внутренний или мобильный телефон;
- адрес, здание или кабинет.

Для уточнения результата могут использоваться фильтры:

- «Организация»;
- «Подразделение»;
- «Должность»;
- «Здание / кабинет».

Основная форма поиска с фильтрами показана на рисунке 37.

Рисунок 37 – Форма поиска с дополнительными фильтрами

После ввода запроса пользователь нажимает кнопку «Найти» или использует кнопку поиска в поле ввода. При вводе запроса длиной от двух символов поиск также может выполняться автоматически с небольшой задержкой. Результаты отображаются в таблице с колонками «Сотрудник», «Должность», «Подразделение», «Организация», «Контакты», «Расположение», «Действия». Кнопка «Сбросить» очищает строку поиска и фильтры, после чего таблица возвращается к общему списку сотрудников.

Пример работы поиска показан на рисунке 38.

Рисунок 38 – Таблица результатов поиска сотрудников

В таблице доступны действия:

- открыть карточку сотрудника;
- скопировать контактные данные;
- редактировать, если роль пользователя позволяет изменять запись;
- удалить сотрудника, если пользователь является администратором.

Если результатов больше, чем помещается на одну страницу, используется пагинация. Пользователь может перейти на следующую страницу и изменить количество строк на странице. Пример работы с таблицей результатов поиска показан на рисунке 39.

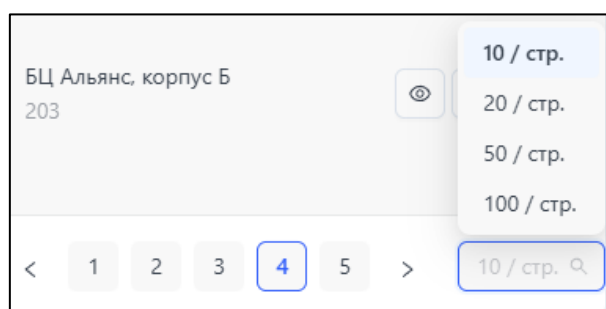


Рисунок 39 – Пагинация таблицы результатов поиска

#### 1.5.5 Карточка сотрудника

Чтобы открыть карточку сотрудника, нужно нажать на ФИО сотрудника в таблице результатов или на кнопку просмотра в колонке «Действия». В адресной строке открывается маршрут вида `/employees/{id}`. Это показано на рисунке 40.

| Сотрудник                                                                                                      | Должность           | Подразделение    | Организация | Контакты                                                                            | Расположение               | Действия                                                                                                                                                                                                                                                          |
|----------------------------------------------------------------------------------------------------------------|---------------------|------------------|-------------|-------------------------------------------------------------------------------------|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  Алексеева Ксения Денисовна | Аналитик требований | Инженерный центр | НПО Альянс  | +7 (495) 120-45-22<br>3122<br>+7 (916) 410-11-12<br>alekseeva.kd@npo-alliance.local | БЦ Альянс, корпус А<br>312 |    |

Рисунок 40 – Открытие карточки сотрудника из таблицы поиска

В карточке отображаются:

- ФИО и аватар сотрудника;
- должность;

- статус активности;
- организация и подразделение;
- кабинет, этаж и адрес;
- рабочий, внутренний и мобильный телефон;
- email;
- логин, если с сотрудником связана учетная запись;
- дата рождения;
- дата последнего обновления записи.

Пример карточки сотрудника представлен на рисунке 41.

**Карточка сотрудника**

НПО Альянс / Инженерный центр / Алексеева

**Алексеева Ксения Денисовна**  
Аналитик требований  
Активен

Позвонить Написать Скопировать контакты Показать на карте

|               |     |               |            |               |                     |
|---------------|-----|---------------|------------|---------------|---------------------|
| ID сотрудника | 12  | Организация   | НПО Альянс | Подразделение | Инженерный центр    |
| Кабинет       | 312 | Этаж          | 3          | Адрес         | БЦ Альянс, корпус А |
| Логин         | -   | Дата рождения | 1996-06-11 | Обновлено     | 15.05.2026 00:02    |

Рабочий +7 (495) 120-45-22  
 Внутренний 3122  
 Мобильный +7 (916) 410-11-12  
 Email alekseeva.kd@npo-alliance.local

Рисунок 41 – Карточка сотрудника с контактной информацией

На карточке доступны быстрые действия:

- «Позвонить» – открывает телефонную ссылку, если указан номер;
- «Написать» – открывает почтовую ссылку, если указан email;
- «Скопировать контакты» – копирует основные контакты в буфер обмена;
- «Редактировать» – открывает окно редактирования контактных данных сотрудника;
- «Показать на карте» – открывает раздел карты организаций;
- «Назад» – возвращает пользователя к предыдущей странице;
- «Скопировать ссылку» – копирует ссылку на карточку сотрудника;
- «Удалить» – удаляет сотрудника.

Кнопки быстрых действий показаны на рисунке 42.

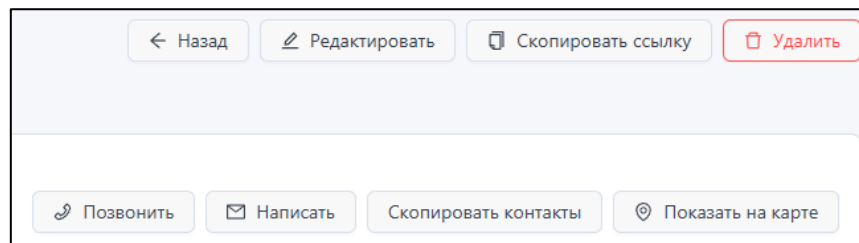


Рисунок 42 – Кнопки быстрых действий в карточке сотрудника

Кнопка «Редактировать» доступна пользователям, которым разрешено изменять запись. Администратор может редактировать сотрудников без ограничений. Руководитель может редактировать сотрудников только в пределах своего подразделения и вложенных подразделений. Удаление карточки доступно только администратору и выполняется после подтверждения.

Если сотрудник с указанным идентификатором не найден, система показывает сообщение «Сотрудник не найден», это представлено на рисунке 43.



Рисунок 43 – Сообщение при открытии несуществующей карточки сотрудника

### 1.5.6 Создание, редактирование и удаление сотрудников

Создание сотрудника выполняется через кнопку «Добавить сотрудника». В текущей версии эта операция доступна администратору. Форма создания содержит группы полей: ФИО, должность и структура, контакты, расположение и дополнительные данные. Форма добавления нового сотрудника показана на рисунке 44.

Рисунок 44 – Форма добавления сотрудника

Обязательные поля:

- «Фамилия»;
- «Имя»;
- «Подразделение».

Дополнительно можно указать отчество, должность, рабочий телефон, внутренний телефон, мобильный телефон, email, кабинет, этаж, адрес, дату рождения, ссылку на фото и статус активности.

Система проверяет обязательные поля и формат email. При попытке сохранить форму без обязательных данных запись не создается, а под полями отображаются сообщения валидации. При некорректном email выводится сообщение «Некорректный email», как показано на рисунке 45.

Рисунок 45 – Валидация формы сотрудника при незаполненных обязательных полях

После заполнения данных пользователь нажимает «Сохранить». При успешном создании появляется сообщение «Карточка сотрудника создана», а пользователь переходит к карточке нового сотрудника, что видно на рисунке 46.

Рисунок 46 – Успешно созданная карточка сотрудника

Редактирование существующей записи открывается из карточки сотрудника или таблицы поиска. В форме отображаются текущие значения. Пользователь изменяет нужные поля и нажимает «Сохранить». Кнопка «Сбросить изменения» возвращает значения формы к исходным, а «Отмена» закрывает форму без сохранения. Руководитель может сохранить изменения только для сотрудников

своего подразделения или дочерних подразделений. Попытка изменить сотрудника вне зоны ответственности блокируется серверной проверкой доступа. Администратор может изменять любые записи. Форма редактирования сотрудника изображена на рисунке 47.

Рисунок 47 – Форма редактирования сотрудника

Удаление сотрудника доступно только администратору и требует подтверждения в диалоговом окне, пример подтверждения показан на рисунке 48.

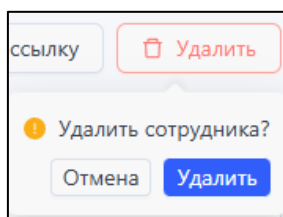


Рисунок 48 – Подтверждение удаления сотрудника

### 1.5.7 Организационная структура

Раздел «Оргструктура» предназначен для просмотра организаций, подразделений и сотрудников по подразделениям. Слева отображается дерево подразделений, справа – таблица организаций или сотрудников.

Если подразделение не выбрано, в правой части отображается таблица организаций с колонками «Организация», «Адрес», «Телефон». После выбора подразделения справа появляется таблица сотрудников выбранного подразделения с колонками «Сотрудник», «Должность», «Телефон», «Кабинет». Пример отображения показан на рисунке 49.

| Сотрудник         | Должность                               | Телефон            | Кабинет |
|-------------------|-----------------------------------------|--------------------|---------|
| Сафонова Лилия    | Маркетолог B2B-направления              | +7 (495) 120-47-24 | 414     |
| Ефимов Даниил     | Проектный менеджер                      | +7 (495) 120-47-19 | 409     |
| Тарасов Владислав | Специалист по коммерческим предложениям | +7 (495) 120-47-25 | 415     |
| Заваров Леонид    | Пресс-инженер                           | +7 (495) 120-47-17 | 407     |
| Белова Екатерина  | Руководитель клиентских проектов        | +7 (495) 120-47-12 | 402     |
| Петров Иван       | Менеджер по развитию бизнеса            | +7 (495) 120-47-15 | 405     |
| Орлов Сергей      | Директор по продажам                    | +7 (495) 120-47-11 | 401     |
| Коваленко Оксана  | Менеджер по тендерам                    | +7 (495) 120-47-14 | 404     |
| Дыкова Марина     | Специалист CRM                          | +7 (495) 120-47-22 | 412     |
| Андреев Руслан    | Менеджер по ключевым клиентам           | +7 (495) 120-47-13 | 403     |

Рисунок 49 – Просмотр сотрудников выбранного подразделения

На странице доступны кнопки экспорта Excel и PDF. Администратор также видит кнопку «Добавить сотрудника», а администратор и руководитель – кнопку «Добавить подразделение». Кнопка «Добавить организацию» доступна администратору. Сотрудник может просматривать организационную структуру, но не видит административные кнопки добавления и редактирования. Доступные действия представлены на рисунке 50.



Рисунок 50 – Различие доступных действий в оргструктуре для администратора

### 1.5.8 Карта организаций

Раздел «Карта» отображает организации и здания на интерактивной карте. Слева находится панель фильтров, справа – карта с маркерами организаций. Иллюстрация раздела показана на рисунке 51.

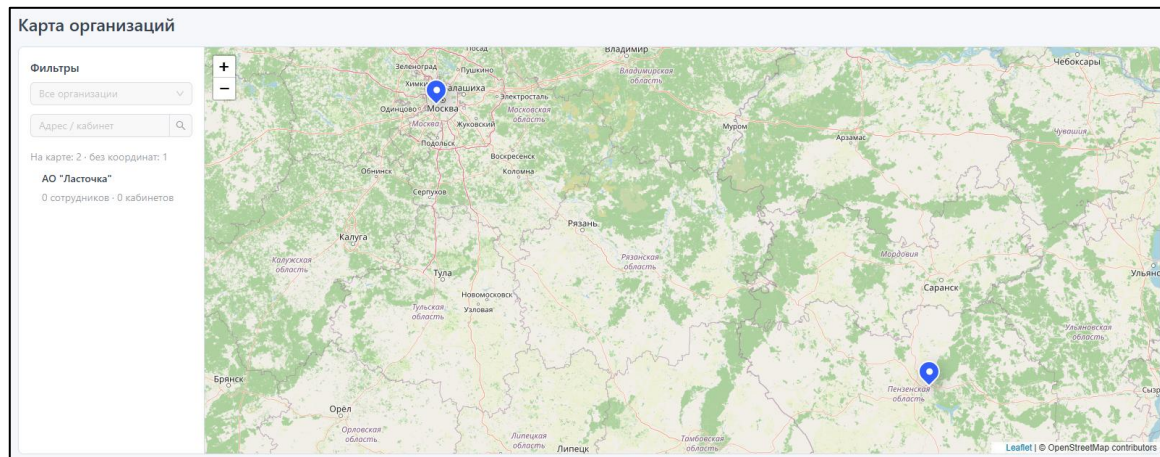


Рисунок 51 – Раздел «Карта организаций»

Пользователь может выбрать организацию в фильтре или ввести адрес, название здания либо кабинет в строку поиска. На карте отображаются только объекты, для которых в справочнике организаций указаны координаты. Если у организации или здания нет координат, запись попадает в список «Без координат» на левой панели. Окно фильтров представлено на рисунке 52.

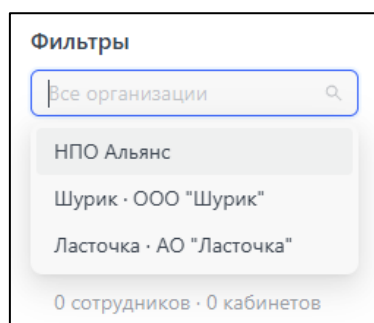


Рисунок 52 – Фильтры карты и список объектов без координат

При нажатии на маркер открывается всплывающее окно с названием, адресом, количеством сотрудников и количеством кабинетов. Кнопка «Сотрудники здесь» открывает боковую панель со списком сотрудников выбранной

организации, сгруппированным по кабинетам. Если координаты у организации не заполнены, система не может поставить маркер на карту.

На рисунке 53 представлены всплывающее окно и боковая панель со списком сотрудников.

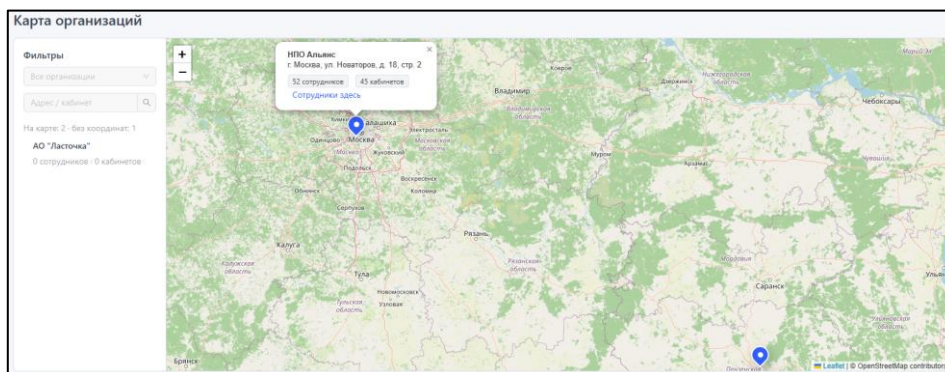


Рисунок 53 – Всплывающее окно маркера на карте

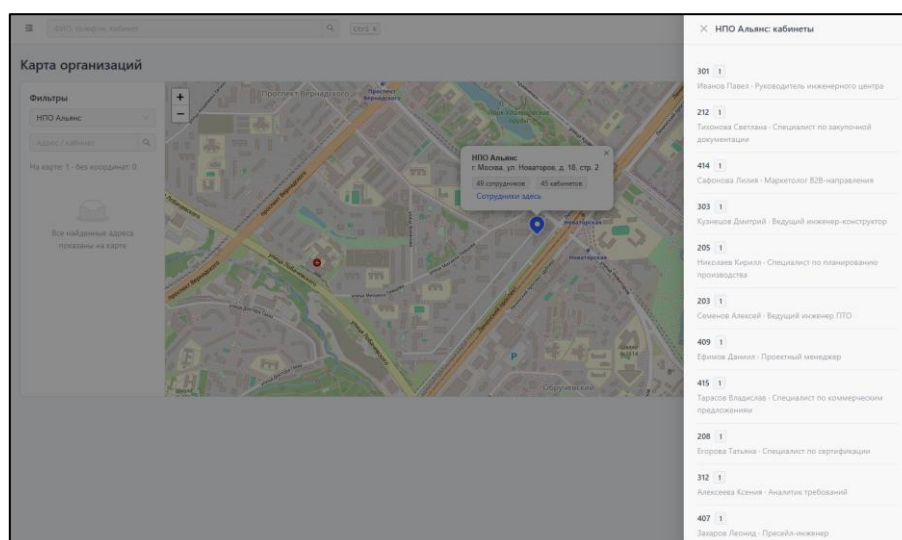


Рисунок 54 – Боковая панель со списком сотрудников организации по кабинетам

### 1.5.9 Профиль пользователя

Раздел «Профиль» показывает данные текущего пользователя и позволяет выполнять действия, связанные с собственной учетной записью. Основное окно раздела «Профиль» показано на рисунке 55.

Рисунок 55 – Раздел «Профиль»

Вкладка «Доступ» показывает роль пользователя, зону ответственности и список разрешений. Для администратора отображается доступ ко всем организациям и подразделениям. Для руководителя указывается подразделение или организация, к которой он привязан. Для сотрудника указывается собственный профиль и просмотр справочника. На рисунке 56 показана вкладка «Доступ».

Рисунок 56 – Вкладка «Доступ» со списком разрешений

Вкладка «Контакты» позволяет пользователю обновить собственный рабочий телефон, внутренний телефон, кабинет и ссылку на фото. После нажатия «Сохранить изменения» система сохраняет данные и показывает сообщение «Профиль обновлен». Успешное изменение данных на вкладке «Контакты» показано на рисунке 57.

Рисунок 57 – Вкладка «Контакты» для самостоятельного обновления контактных данных

Вкладка «Безопасность» используется для смены пароля. Пользователь вводит текущий пароль и новый пароль, затем нажимает «Сменить пароль». Если обязательные поля пустые, система показывает сообщения валидации. Если текущий пароль указан неверно, пароль не изменяется. После успешной смены появляется сообщение «Пароль изменен». Также можно выйти из профиля, нажав кнопку «Выйти». На рисунке 58 отображена вкладка «Безопасность».

Профиль

Профиль Доступ Контакты **Безопасность** Активность

Текущий пароль

Новый пароль

Выйти

Сменить пароль

Рисунок 58 – Вкладка «Безопасность» для смены пароля

Вкладка «Активность» отображает основные действия, которые совершал пользователь в системе. Пользователь видит только свои действия. Просмотр общего подробного журнала событий выполняется администратором в разделе «Аудит». Вкладка «Активность» показана на рисунке 59.

Профиль

Профиль Доступ Контакты Безопасность **Активность**

Администратор видит журнал действий всех пользователей.

| Дата             | Пользователь | Действие |
|------------------|--------------|----------|
| 22.05.2026 00:16 | admin        | Просмотр |
| 22.05.2026 00:16 | admin        | Поиск    |
| 21.05.2026 23:52 | admin        | Поиск    |
| 21.05.2026 23:52 | admin        | Вход     |

Рисунок 59 – Вкладка «Активность»

### 1.5.10 Импорт и экспорт данных

Экспорт сотрудников доступен в форматах Excel и PDF. Кнопки экспорта находятся на странице поиска и в разделе «Импорт/экспорт». При необходимости пользователь может выбрать подразделение, чтобы выгрузить только сотрудников этого подразделения. Возможные действия по экспорту и импорту видны на рисунках 60, 61.

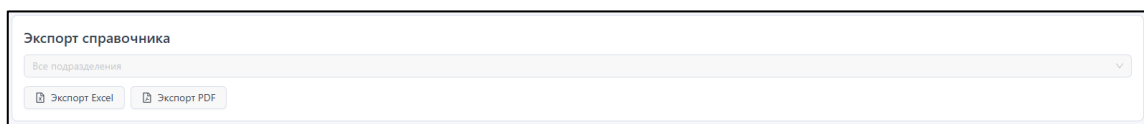


Рисунок 60 – Кнопки «Экспорт Excel» и «Экспорт PDF» на странице поиска

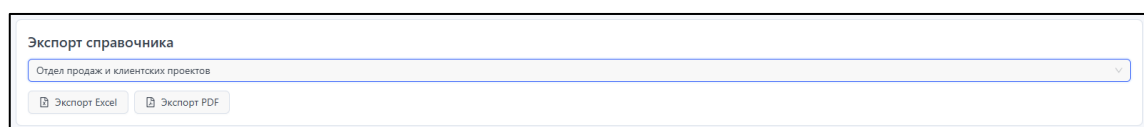


Рисунок 61 – Экспорт справочника с фильтром по подразделению

Excel-файл содержит основные поля сотрудника: Фамилия, Имя, Отчество, Рабочий телефон, Внутренний телефон, Мобильный телефон, Email, Должность, Отдел, Организация, Локация, Кабинет, Этаж, Дата рождения. PDF-отчет формируется как файл для просмотра или печати.

Импорт сотрудников из Excel доступен администраторам и руководителям. Для импорта нужно открыть раздел «Импорт и экспорт», вкладку «Excel», скачать шаблон, заполнить файл и загрузить его через область загрузки. Файл можно выбрать через диалог или перетащить мышью. Это видно из рисунка 62.

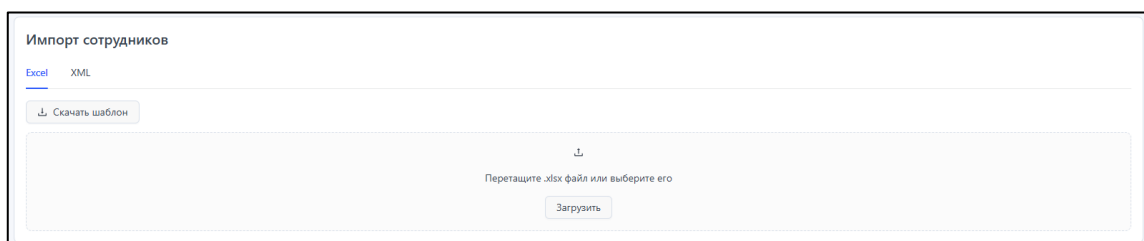


Рисунок 62 – Вкладка Excel-импорта сотрудников

Для успешного импорта в файле должны быть обязательные колонки:

- Фамилия;
- Имя;
- Отдел или Подразделение.

Дополнительно поддерживаются колонки: Отчество, Рабочий телефон, Внутренний телефон, Мобильный телефон, Email, Должность, Организация, Локация, Кабинет, Этаж, Дата рождения.

При обработке файла система проверяет обязательные поля, формат email, формат телефонов и существование подразделения. Если строка корректна, сотрудник создается. Если строка содержит ошибку, она пропускается, а ошибка показывается пользователю. После обработки выводится сообщение вида «Импортировано: N, ошибок: M». Пример ошибок при импорте показан на рисунках 63, 64.

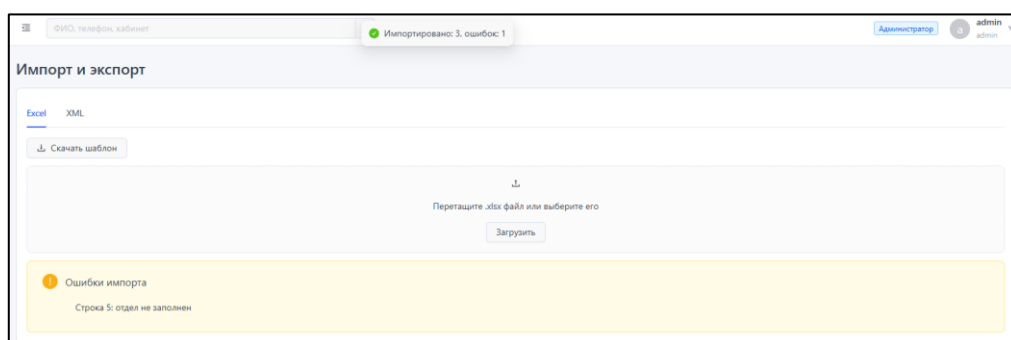


Рисунок 63 – Результат импорта Excel с количеством успешных строк и ошибок

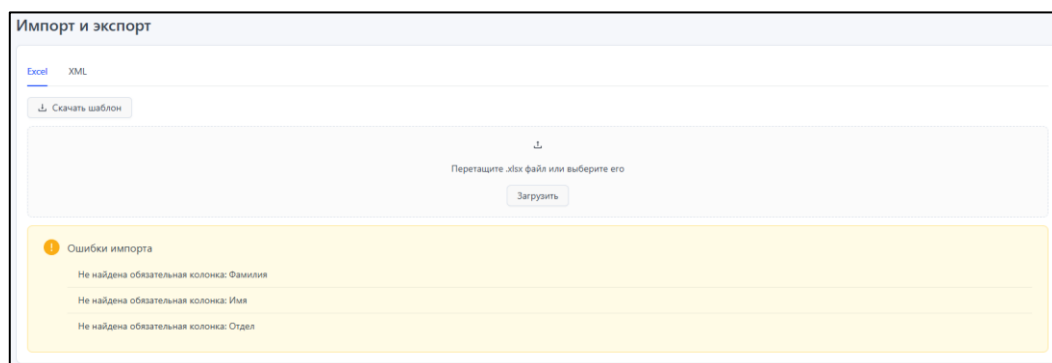


Рисунок 64 – Список ошибок импорта

Руководитель может импортировать сотрудников только в подразделения, которые входят в его зону ответственности. Попытка импортировать сотрудников в чужое подразделение добавляется в список ошибок.

Вкладка «XML» позволяет импортировать сотрудников в формате XML. Можно загрузить XML файл через эту вкладку, это показано на рисунке 65.

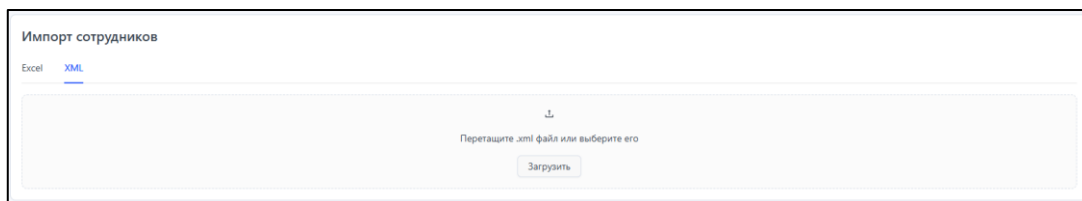


Рисунок 65 – Вкладка XML-импорта

#### 1.5.11 Управление пользователями

Раздел «Пользователи» доступен только администратору. В нем отображается таблица учетных записей с логином, связанным сотрудником, ролью, статусом, датой создания и действиями. На рисунке 66 показан раздел «Пользователи».

| Пользователи                                                                                                                                              |                                 |               |         |                            |                                                                                                                                      | <a href="#">+ Создать пользователя</a> |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------|---------------|---------|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|
| <input type="text" value="Поиск по username"/> <input type="button" value="Q"/> <input type="button" value="Роль"/> <input type="button" value="Статус"/> |                                 |               |         |                            |                                                                                                                                      |                                        |
| Логин                                                                                                                                                     | Сотрудник                       | Роль          | Статус  | Создан                     | Действия                                                                                                                             |                                        |
| manager.rto                                                                                                                                               | #39 - Морозов Денис Анатольевич | Руководитель  | Активен | 2026-05-15T00:02:41.989588 | <input type="button" value="✎"/> <input type="button" value="🔒"/> <input type="button" value="🔄"/> <input type="button" value="🗑️"/> |                                        |
| manager.sales                                                                                                                                             | #26 - Орлов Сергей Михайлович   | Руководитель  | Активен | 2026-05-15T00:02:41.989588 | <input type="button" value="✎"/> <input type="button" value="🔒"/> <input type="button" value="🔄"/> <input type="button" value="🗑️"/> |                                        |
| employee.demo                                                                                                                                             | #41 - Соколова Наталья Олеговна | Сотрудник     | Активен | 2026-05-15T00:02:41.989588 | <input type="button" value="✎"/> <input type="button" value="🔒"/> <input type="button" value="🔄"/> <input type="button" value="🗑️"/> |                                        |
| manager.engineering                                                                                                                                       | #3 - Иванов Павел Андреевич     | Руководитель  | Активен | 2026-05-15T00:02:41.989588 | <input type="button" value="✎"/> <input type="button" value="🔒"/> <input type="button" value="🔄"/> <input type="button" value="🗑️"/> |                                        |
| ilya.qa                                                                                                                                                   | #50 - Жданов Илья Феликсов      | Сотрудник     | Активен | 2026-05-16T19:34:25.427164 | <input type="button" value="✎"/> <input type="button" value="🔒"/> <input type="button" value="🔄"/> <input type="button" value="🗑️"/> |                                        |
| admin                                                                                                                                                     | -                               | Администратор | Активен | 2026-05-11T18:13:18.742527 | <input type="button" value="✎"/> <input type="button" value="🔒"/> <input type="button" value="🔄"/> <input type="button" value="🗑️"/> |                                        |

Рисунок 66 – Раздел управления пользователями

Администратор может:

- создать пользователя;
- отредактировать логин, роль и привязку к сотруднику;
- найти пользователя по username;
- отфильтровать список по роли и статусу;

- сменить пароль пользователя;
- заблокировать или разблокировать учетную запись;
- удалить учетную запись.

Фильтр по списку пользователей показан на рисунке 67.

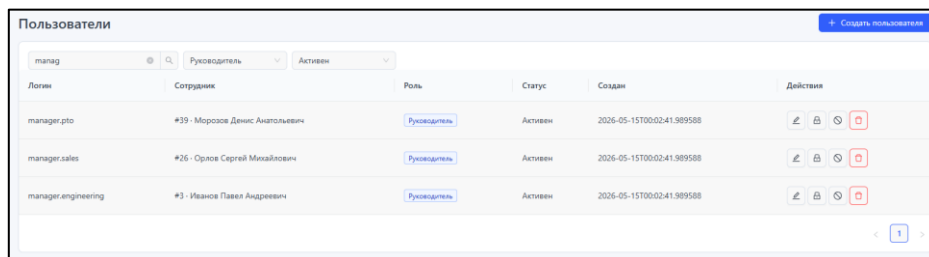


Рисунок 67 – Фильтры пользователей в административной панели

Для создания пользователя администратор нажимает «Создать пользователя». Открывается боковая форма с полями «Логин», «Пароль», «Роль» и «Сотрудник». Обязательные поля для заполнения обозначены символом «\*». На рисунке 68 отображена форма создания нового пользователя.

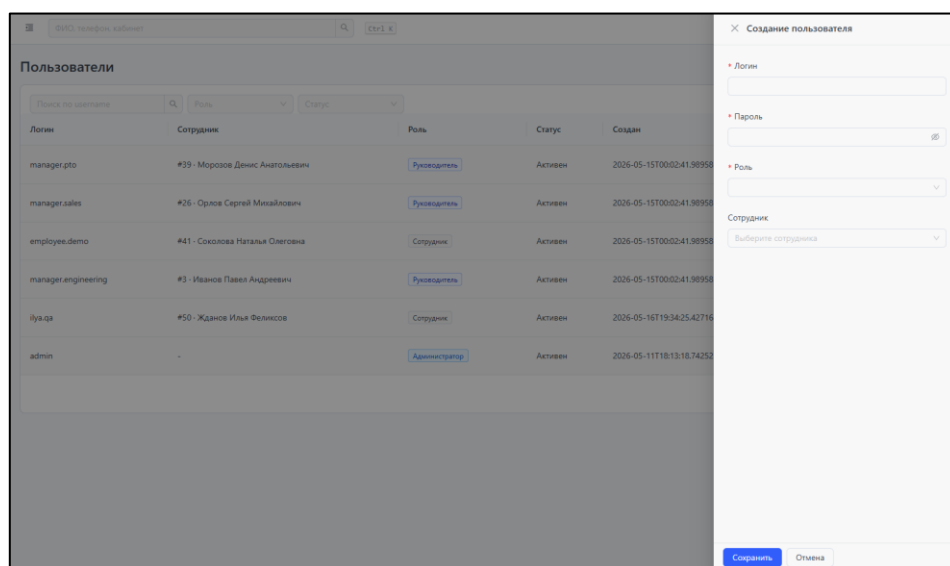


Рисунок 68 – Форма создания пользователя

После создания учетная запись появляется в таблице. Если пользователь заблокирован, он не может выполнить вход в систему. Удаление пользователя

выполняется после подтверждения. Окна смены пароля и блокировки пользователя показаны на рисунках 69, 70.

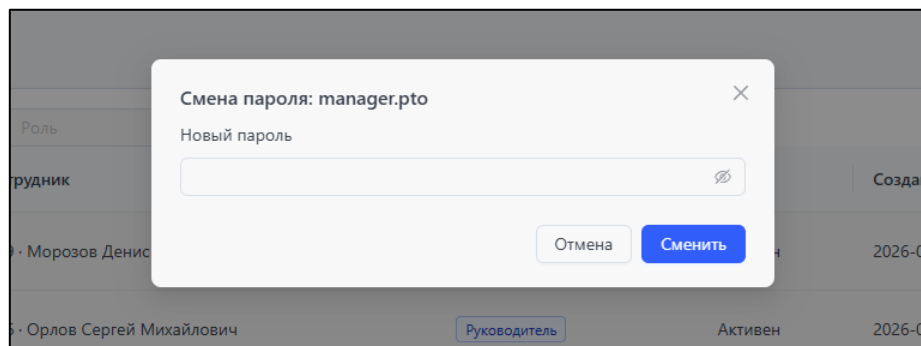


Рисунок 69 – Диалог смены пароля пользователя администратором

| Логин               | Сотрудник                       | Роль          | Статус       | Создан                     | Действия                                                                 |
|---------------------|---------------------------------|---------------|--------------|----------------------------|--------------------------------------------------------------------------|
| manager.sales       | #26 - Орлов Сергей Михайлович   | Руководитель  | Активен      | 2026-05-15T00:02:41.989588 | <a href="#">✎</a> <a href="#">🔒</a> <a href="#">🔓</a> <a href="#">🗑️</a> |
| employee.demo       | #41 - Соколова Наталья Олеговна | Сотрудник     | Активен      | 2026-05-15T00:02:41.989588 | <a href="#">✎</a> <a href="#">🔒</a> <a href="#">🔓</a> <a href="#">🗑️</a> |
| manager.engineering | #3 - Иванов Павел Андреевич     | Руководитель  | Активен      | 2026-05-15T00:02:41.989588 | <a href="#">✎</a> <a href="#">🔒</a> <a href="#">🔓</a> <a href="#">🗑️</a> |
| ilya.qa             | #50 - Жданов Илья Феликсов      | Сотрудник     | Активен      | 2026-05-16T19:34:25.427164 | <a href="#">✎</a> <a href="#">🔒</a> <a href="#">🔓</a> <a href="#">🗑️</a> |
| admin               | -                               | Администратор | Активен      | 2026-05-11T18:13:18.742527 | <a href="#">✎</a> <a href="#">🔒</a> <a href="#">🔓</a> <a href="#">🗑️</a> |
| manager.pto         | #39 - Морозов Денис Анатольевич | Руководитель  | Заблокирован | 2026-05-15T00:02:41.989588 | <a href="#">✎</a> <a href="#">🔒</a> <a href="#">🔓</a> <a href="#">🗑️</a> |

Рисунок 70 – Блокировка или разблокировка пользователя

### 1.5.12 Управление справочниками

Раздел «Справочники» доступен администраторам и руководителям. В нем ведутся основные справочные данные, от которых зависят поиск, оргструктура, карта и импорт сотрудников. На рисунке 71 показан раздел «Справочники».

| Организации   |            |                                         |                    |                         |                            |                      |                                      |
|---------------|------------|-----------------------------------------|--------------------|-------------------------|----------------------------|----------------------|--------------------------------------|
| Название      | Короткое   | Адрес                                   | Телефон            | Email                   | Сайт                       | Координаты           | Действия                             |
| НПО Альянс    | НПО Альянс | г. Москва, ул. Новаторов, д. 18, стр. 2 | +7 (495) 120-45-00 | info@npo-alliance.local | https://npo-alliance.local | 55.67054, 37.51786   | <a href="#">✎</a> <a href="#">🗑️</a> |
| ООО "Шурик"   | Шурик      | улица Захарова, 19, Пенза, 440008       |                    |                         |                            | 53.193952, 44.998975 | <a href="#">✎</a> <a href="#">🗑️</a> |
| АО "Ласточка" | Ласточка   | улица Мира, 35, Пенза, 440046           |                    |                         |                            | -                    | <a href="#">✎</a> <a href="#">🗑️</a> |

Рисунок 71 – Раздел «Справочники»

В разделе есть вкладки:

- «Организации»;
- «Подразделения»;
- «Приемные»;
- «Коды городов».

На вкладке «Организации» отображаются название, короткое название, адрес, телефон, email, сайт, координаты и действия. Для карты важны поля «Широта» и «Долгота»: без них организация будет отображаться в списке объектов без координат, но не появится как маркер на карте. Вкладка «Организации» и форма добавления организации изображены на рисунках 72, 73.

| Организации   |            |                                         |                    |                         |                            |                      |                                                                                                                                                                           | + Добавить запись |  |
|---------------|------------|-----------------------------------------|--------------------|-------------------------|----------------------------|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|--|
| Название      | Короткое   | Адрес                                   | Телефон            | Email                   | Сайт                       | Координаты           | Действия                                                                                                                                                                  |                   |  |
| НПО Альянс    | НПО Альянс | г. Москва, ул. Новаторов, д. 18, стр. 2 | +7 (495) 120-45-00 | info@npo-alliance.local | https://npo-alliance.local | 55.67054, 37.51786   |     |                   |  |
| ООО "Шурик"   | Шурик      | улица Захарова, 19, Пенза, 440008       |                    |                         |                            | 53.193952, 44.998975 |     |                   |  |
| АО "Ласточка" | Ласточка   | улица Мира, 35, Пенза, 440046           |                    |                         |                            | -                    |   |                   |  |
|               |            |                                         |                    |                         |                            |                      |                                                                                                                                                                           | < 1 >             |  |

Рисунок 72 – Вкладка «Организации»

Справочники

Организации
Подразделения
Приемные
Коды городов

| Название      | Короткое   | Адрес                                   | Телефон            | Email                   | Сайт                       |
|---------------|------------|-----------------------------------------|--------------------|-------------------------|----------------------------|
| НПО Альянс    | НПО Альянс | г. Москва, ул. Новаторов, д. 18, стр. 2 | +7 (495) 120-45-00 | info@npo-alliance.local | https://npo-alliance.local |
| ООО "Шурик"   | Шурик      | улица Захарова, 19, Пенза, 440008       |                    |                         |                            |
| АО "Ласточка" | Ласточка   | улица Мира, 35, Пенза, 440046           |                    |                         |                            |

Добавить организацию

Название

Короткое название

Адрес

Телефон

Email

Сайт

Широта

Долгота

Создать

Отмена

Рисунок 73 – Форма добавления или редактирования организации

На вкладке «Подразделения» отображаются название, организация, родительское подразделение, количество дочерних подразделений и сотрудников.

При создании подразделения указывается организация и при необходимости родительское подразделение. Подразделение не может быть собственным родителем. На рисунках 74, 75 изображены вкладка «Подразделения» и форма добавления подразделения.

| Название                               | Организация | Родитель                               | Детей | Сотрудников | Действия |
|----------------------------------------|-------------|----------------------------------------|-------|-------------|----------|
| Инженерный центр                       | НПО Альянс  |                                        | 0     | 15          |          |
| Производственно-технический отдел      | НПО Альянс  |                                        | 1     | 15          |          |
| Отдел продаж и клиентских проектов     | НПО Альянс  |                                        | 0     | 15          |          |
| Центр разработки программных продуктов | НПО Альянс  | Производственно-технический отдел      | 1     | 2           |          |
| Отдел тестирования                     | НПО Альянс  | Центр разработки программных продуктов | 0     | 5           |          |

Рисунок 74 – Вкладка «Подразделения»

Организации

Подразделения

Приемные

Коды городов

| Название                               | Организация | Родитель                               |
|----------------------------------------|-------------|----------------------------------------|
| Инженерный центр                       | НПО Альянс  |                                        |
| Производственно-технический отдел      | НПО Альянс  |                                        |
| Отдел продаж и клиентских проектов     | НПО Альянс  |                                        |
| Центр разработки программных продуктов | НПО Альянс  | Производственно-технический отдел      |
| Отдел тестирования                     | НПО Альянс  | Центр разработки программных продуктов |

×

Добавить подразделение

Название

Организация

Родительское подразделение

Сохранить

Отмена

Рисунок 75 – Форма добавления или редактирования подразделения

На вкладке «Приемные» хранятся приемные организаций: название, организация, основной телефон, дополнительный телефон и описание. Это отображено на рисунке 76.

| Организации | Подразделения | Приемные           | Коды городов   | + Добавить запись               |                                                                                                                                                                         |
|-------------|---------------|--------------------|----------------|---------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Название    | Организация   | Телефон            | Дополнительный | Описание                        | Действия                                                                                                                                                                |
| Приемная    | НПО Альянс    | +7 (495) 120-45-00 | 1000           | Центральная приемная НПО Альянс |   |
|             |               |                    |                |                                 | < 1 >                                                                                                                                                                   |

Рисунок 76 – Вкладка «Приемные»

На вкладке «Коды городов» хранятся регион, город и телефонный код города. Вкладка «коды городов» показана на рисунке 77.

| Организации     | Подразделения   | Приемные | Коды городов                                                                        | + Добавить запись                                                                   |                                                       |
|-----------------|-----------------|----------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------|
| Регион          | Город           | Код      | Действия                                                                            |                                                                                     |                                                       |
| Москва          | Москва          | 499      |  |  |                                                       |
| Санкт-Петербург | Санкт-Петербург | 812      |  |  |                                                       |
| Москва          | Москва          | 495      |  |  |                                                       |
|                 |                 |          |                                                                                     |                                                                                     | <div><div>&lt;</div><div>1</div><div>&gt;</div></div> |

Рисунок 77 – Вкладка «Коды городов»

Для добавления записи используется кнопка «Добавить запись». Для изменения записи используется кнопка редактирования в таблице. Для удаления используется кнопка удаления с подтверждением, это видно из рисунка 78.

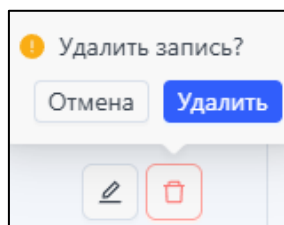


Рисунок 78 – Подтверждение удаления записи справочника

Администратор может удалять записи справочников. Руководитель может создавать и редактировать доступные ему записи, но операции ограничиваются его зоной ответственности, особенно для подразделений.

#### 1.5.13 Журнал аудита

Раздел «Аудит» доступен только администратору. Он предназначен для просмотра событий, которые происходят в системе: поиск, просмотр карточки

сотрудника, создание, изменение и удаление сотрудников, импорт, экспорт, скачивание шаблона, операции с API-ключами и другие действия. На рисунке 79 показан раздел «Аудит».

| Аудит            |              |                     |                |    |               |                                           |          |
|------------------|--------------|---------------------|----------------|----|---------------|-------------------------------------------|----------|
| Пользователь     |              | Дата                | Начальная дата | →  | Конечная дата | Действие                                  | Сущность |
| ↓ Экспорт отчёта |              |                     |                |    |               | Применить                                 | Сбросить |
| Дата             | Пользователь | Действие            | Сущность       | ID | IP адрес      | Детали                                    | Действия |
| 21.05.2026 19:40 | admin        | Импорт сотрудников  | Employee       |    | 127.0.0.1     | Imported employees: success=3, errors=1   | ⌵ Детали |
| 21.05.2026 19:39 | admin        | Импорт сотрудников  | Employee       |    | 127.0.0.1     | Imported employees: success=0, errors=4   | ⌵ Детали |
| 21.05.2026 19:39 | admin        | Поиск               | Search         |    | 127.0.0.1     | query=Алешин; results=0                   | ⌵ Детали |
| 21.05.2026 19:39 | admin        | Поиск               | Search         |    | 127.0.0.1     | results=49                                | ⌵ Детали |
| 21.05.2026 19:39 | admin        | Импорт сотрудников  | Employee       |    | 127.0.0.1     | Imported employees: success=0, errors=4   | ⌵ Детали |
| 21.05.2026 19:38 | admin        | Импорт сотрудников  | Employee       |    | 127.0.0.1     | Imported employees: success=0, errors=4   | ⌵ Детали |
| 21.05.2026 19:37 | admin        | Импорт сотрудников  | Employee       |    | 127.0.0.1     | Imported employees: success=0, errors=3   | ⌵ Детали |
| 21.05.2026 19:12 | admin        | Просмотр            | Employee       | 51 | 127.0.0.1     | Viewed employee card                      | ⌵ Детали |
| 21.05.2026 19:11 | admin        | Просмотр            | Employee       | 51 | 127.0.0.1     | Viewed employee card                      | ⌵ Детали |
| 21.05.2026 19:11 | admin        | Создание сотрудника | Employee       | 51 | 127.0.0.1     | Created employee Горев Дмитрий Михайлович | ⌵ Детали |

Рисунок 79 – Раздел «Аудит»

Таблица аудита содержит колонки:

- «Дата»;
- «Пользователь»;
- «Действие»;
- «Сущность»;
- «ID»;
- «IP адрес»;
- «Детали»;
- «Действия».

Для поиска событий используются фильтры:

- пользователь;
- дата;
- период;

- действие;
- сущность.

Фильтры раздела «Аудит» видно из рисунка 80.

Аудит

Пользователь

2026-05-15

Начальная дата

→

Конечная дата

Действие

Сущность

Применить

Сбросить

Экспорт отчёта

| Дата             | Пользователь        | Действие            | Сущность | ID | IP адрес  | Детали                                    | Действия |
|------------------|---------------------|---------------------|----------|----|-----------|-------------------------------------------|----------|
| 15.05.2026 00:08 | manager.pto         | LOGIN               | User     | 2  | 127.0.0.1 | Successful login                          | Детали   |
| 15.05.2026 00:07 | manager.engineering | Экспорт сотрудников | Employee |    | 127.0.0.1 | Exported employees: format=xlsx, count=45 | Детали   |
| 15.05.2026 00:06 | manager.engineering | LOGIN               | User     | 5  | 127.0.0.1 | Successful login                          | Детали   |
| 15.05.2026 00:05 | admin               | LOGIN               | User     | 1  | 127.0.0.1 | Successful login                          | Детали   |

<

1

>

10 / стр.

Рисунок 80 – Фильтры журнала аудита

Кнопка «Детали» открывает модальное окно с полным описанием события: датой, пользователем, действием, типом сущности, идентификатором сущности, IP-адресом и деталями. Просмотр деталей аудита показан на рисунке 81.

Аудит

Пользователь

Дата

Экспорт отчёта

| Дата             | Пользователь | Действие              | Сущность | ID | IP адрес      | Детали                       | Действия |
|------------------|--------------|-----------------------|----------|----|---------------|------------------------------|----------|
| 21.05.2026 18:55 | ilya.qa      | LOGIN                 | User     | 6  | 0.0.0.0:0.0.1 | Successful login             | Детали   |
| 21.05.2026 18:54 | admin        | Поиск                 | Search   |    | 127.0.0.1     | results=48                   | Детали   |
| 21.05.2026 18:54 | admin        | Поиск                 | Search   |    | 127.0.0.1     | Successful login             | Детали   |
| 20.05.2026 20:16 | admin        | Поиск                 | Search   |    | 127.0.0.1     | results=48                   | Детали   |
| 20.05.2026 20:14 | admin        | Просмотр              | Employee | 50 | 127.0.0.1     | Viewed employee card         | Детали   |
| 20.05.2026 20:14 | admin        | Обновление сотрудника | Employee | 50 | 127.0.0.1     | Updated employee             | Детали   |
| 20.05.2026 20:13 | admin        | Поиск                 | Search   |    | 127.0.0.1     | query=Жданов Илья; results=1 | Детали   |
| 20.05.2026 20:13 | admin        | Просмотр              | Employee | 50 | 127.0.0.1     | Viewed employee card         | Детали   |
| 20.05.2026 20:13 | admin        | Обновление сотрудника | Employee | 50 | 127.0.0.1     | Updated employee             | Детали   |
| 20.05.2026 20:12 | admin        | Просмотр              | Employee | 50 | 127.0.0.1     | Viewed employee card         | Детали   |

Детали события

Дата

21.05.2026 18:55:24

Пользователь

ilya.qa

Действие

LOGIN

Сущность

User

ID сущности

6

IP адрес

0.0.0.0:0.0.1

Детали

Successful login

Рисунок 81 – Модальное окно «Детали события»

Кнопка «Экспорт отчета» скачивает отчет аудита с учетом текущих фильтров. Кнопка «Сбросить» очищает фильтры и возвращает общий список событий. Экспорт отчета аудита показан на рисунке 82.

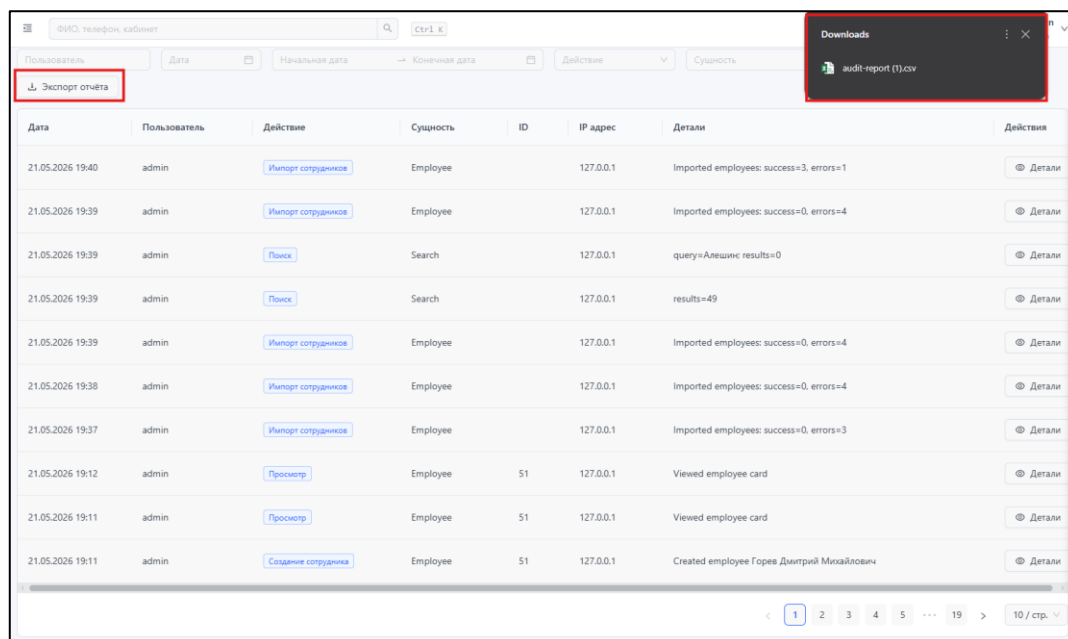


Рисунок 82 – Экспорт отчета аудита

Пользователи без роли администратора не видят раздел аудита и не могут просматривать или изменять журнал событий.

#### 1.5.14 Интеграции и API-ключи

Раздел «Интеграции» доступен только администратору. Основная функция раздела – управление API-ключами для внешних систем. Раздел «Интеграции» показан на рисунке 83.

| Интеграции                                                                                                                                                                         |                  |                      |         |                  |                         |          |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|----------------------|---------|------------------|-------------------------|----------|
| API-ключи и внешние подключения                                                                                                                                                    |                  |                      |         |                  |                         |          |
| + Создать ключ                                                                                                                                                                     |                  |                      |         |                  |                         |          |
| API-ключ нужен для внешних систем, которые будут читать или обновлять данные справочника без входа через веб-интерфейс. Область доступа фиксирует разрешенный сценарий интеграции. |                  |                      |         |                  |                         |          |
| Название                                                                                                                                                                           | Префикс          | Доступ               | Статус  | Создан           | Последнее использование | Действия |
| master_key                                                                                                                                                                         | pbk_OESrheH53o_K | Просмотр сотрудников | Активен | 21.05.2026 19:55 | -                       | ⚙️ 🗑️    |
| test_key                                                                                                                                                                           | pbk_WfmpY5ISXgTT | Просмотр сотрудников | Отозван | 21.05.2026 19:54 | -                       | ⚙️ 🗑️    |
| Тест портал                                                                                                                                                                        | pbk_pjAdoUhqniGr | Просмотр сотрудников | Активен | 17.05.2026 21:24 | -                       | ⚙️ 🗑️    |

Рисунок 83 – Раздел «Интеграции»

На вкладке «API-ключи» отображаются название ключа, префикс, область доступа, статус, дата создания, дата последнего использования и действия. Полное

значение ключа хранится безопасно и показывается только один раз сразу после создания. Таблица API-ключей показана на рисунке 84.

| Название    | Префикс          | Доступ                               | Статус  | Создан           | Последнее использование | Действия |
|-------------|------------------|--------------------------------------|---------|------------------|-------------------------|----------|
| master_key  | pbk_OESxheH53o_K | <a href="#">Просмотр сотрудников</a> | Активен | 21.05.2026 19:55 | -                       |          |
| test_key    | pbk_WfmpY5ISXg1T | <a href="#">Просмотр сотрудников</a> | Отозван | 21.05.2026 19:54 | -                       |          |
| Тест портал | pbk_pjAdoUhgniGr | <a href="#">Просмотр сотрудников</a> | Активен | 17.05.2026 21:24 | -                       |          |

Рисунок 84 – Таблица API-ключей

Для создания ключа администратор нажимает «Создать ключ», вводит название и при необходимости область доступа, например read:employees. После создания открывается окно с полным значением ключа. Пользователь должен скопировать его сразу: после закрытия окна полный ключ больше не отображается, остается только префикс. Процесс создания API-ключа показан на рисунках 85, 86.

Рисунок 85 – Форма создания API-ключа

Рисунок 86 – Окно с созданным API-ключом

API-ключ можно отозвать или удалить. Отозванный ключ остается в списке со статусом «Отозван», но не должен использоваться внешними системами. Удаленный ключ исчезает из таблицы. Отзыв API-ключа изображен на рисунке 87.

| Название   | Префикс          | Доступ         | Статус  | Создан           | Последнее использование | <div>Отозвать API-ключ?</div> <div>Отмена Отозвать</div> |  |
|------------|------------------|----------------|---------|------------------|-------------------------|----------------------------------------------------------|--|
| master_key | pbk_OESxheH53o_K | read-employees | Активен | 21.05.2026 19:55 | -                       |                                                          |  |
| test_key   | pbk_WfmpY5ISXg1T | read-employees | Отозван | 21.05.2026 19:54 | -                       |                                                          |  |

Рисунок 87 – Отзыв API-ключа

Вкладка «Подключения» показывает предполагаемые внешние подключения, например HR-систему и корпоративный портал. В текущей версии эта вкладка носит информационный характер. На рисунке 88 показана вкладка «Подключения».

Интеграции

API-ключи и внешние подключения

+

Создать ключ

API-ключи

Подключения

HR-система

Импорт сотрудников и подразделений

Корпоративный портал

Публикация справочника контактов

Рисунок 88 – Вкладка «Подключения»

## 2 Безопасность жизнедеятельности

При работе с персональным компьютером необходимо соблюдать требования безопасности, так как длительное взаимодействие с монитором, клавиатурой и мышью может отрицательно сказаться на состоянии здоровья пользователя. Неправильная организация рабочего места, недостаточное освещение, отсутствие перерывов и несоблюдение правил эксплуатации оборудования могут привести к утомлению, снижению работоспособности, ухудшению зрения и появлению болей в спине, шее и кистях рук.

Правильно организованное рабочее место является одним из основных условий безопасной и продуктивной работы. Помещение должно быть достаточно просторным, чистым, хорошо проветриваемым и освещенным. Естественный свет положительно влияет на самочувствие человека, однако рабочее место необходимо располагать так, чтобы солнечные лучи не создавали бликов на экране монитора. Блики заставляют пользователя напрягать зрение, быстрее вызывают усталость и могут стать причиной головной боли.

Искусственное освещение в помещениях, где выполняется работа с ПЭВМ, должно обеспечивать комфортные условия зрительного восприятия. Освещенность рабочего места должна составлять примерно 300–500 лк. Запрещается работать в темном или полутемном помещении, а также при чрезмерно ярком освещении. Свет должен распределяться равномерно, без резких теней и контрастов. Естественный свет должен падать преимущественно слева от рабочего стола. Не рекомендуется располагать монитор напротив окна или таким образом, чтобы окно находилось за спиной пользователя.

В качестве источников общего искусственного освещения рекомендуется использовать светильники с рассеивателями или экранирующими решетками. Они позволяют создать равномерное освещение и уменьшить вероятность появления отражений на экране монитора и поверхности клавиатуры. При необходимости на рабочем месте может использоваться дополнительный локальный источник света, например настольная лампа с регулируемым направлением и яркостью.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 118  |

Несмотря на то, что современные LCD и LED-мониторы значительно безопаснее устаревших мониторов с электронно-лучевой трубкой, полностью исключать влияние электромагнитного излучения нельзя. Поэтому пользователь должен соблюдать разумное расстояние от экрана и не находиться слишком близко к монитору. При длительной работе также рекомендуется использовать дисплеи с антибликовым покрытием и правильно настраивать яркость, контрастность и размер шрифта.

Особое внимание следует уделять эргономике рабочего места. Экран видеомонитора должен находиться от глаз пользователя на расстоянии 600–700 мм, но не ближе 500 мм. Верхняя часть экрана должна располагаться примерно на уровне глаз или немного ниже. Такое положение позволяет уменьшить напряжение глаз и мышц шеи. Клавиатуру и мышь необходимо размещать так, чтобы руки находились в естественном положении, а кисти и предплечья не испытывали постоянного напряжения.

Рабочий стол должен обеспечивать удобное размещение оборудования с учетом характера выполняемой работы. Его поверхность должна быть устойчивой, достаточно просторной и не создавать бликов. Рабочее кресло должно быть подъемно-поворотным, регулироваться по высоте, иметь удобную спинку и обеспечивать поддержку поясницы. Желательно, чтобы регулировка высоты сиденья и положения спинки выполнялась независимо и имела надежную фиксацию. Сиденье и спинка должны быть выполнены из удобных, не электризующихся материалов.

Для комфортной работы с персональным компьютером необходимо поддерживать благоприятный микроклимат. Температура воздуха в помещении должна находиться примерно в пределах 19–23 °С, а относительная влажность воздуха составлять около 40–60 %. Помещение, где установлены компьютеры, должно регулярно проветриваться. Также рекомендуется проводить влажную уборку, так как скопление пыли ухудшает санитарное состояние помещения и может отрицательно влиять на работу техники.

Важным условием безопасной работы является соблюдение режима труда и отдыха. Непрерывная работа за компьютером не должна превышать одного часа. После этого необходимо делать перерыв продолжительностью 10–15 минут. Во время перерыва рекомендуется сменить положение тела, пройтись, выполнить легкую разминку для спины, шеи, плеч и кистей рук. При работе в ночное время продолжительность перерывов следует увеличивать, так как нагрузка на организм становится выше.

Для снижения зрительного напряжения рекомендуется выполнять простые упражнения для глаз:

- часто поморгать в течение одной-двух минут;
- закрыть глаза на несколько секунд, затем открыть их;
- перевести взгляд с близко расположенного предмета на удаленный объект;
- выполнить несколько круговых движений глазами;
- слегка помассировать закрытые веки без сильного давления.

Отдельное внимание необходимо уделять электробезопасности. Все устройства должны подключаться к исправным розеткам и сетевым фильтрам. Важно контролировать исправность кабелей, вилок и корпусов оборудования. Не допускается работа с поврежденными проводами, самостоятельная разборка техники, подключение устройств мокрыми руками и перегрузка электросети. Правильно выполненное заземление снижает риск поражения электрическим током и повышает безопасность эксплуатации оборудования.

Также необходимо соблюдать правила пожарной безопасности. Рабочее место не должно быть загромождено бумагой, упаковочными материалами и другими легковоспламеняющимися предметами. Нельзя закрывать вентиляционные отверстия системного блока и монитора, так как это может привести к перегреву. При появлении запаха гари, искрения или нестабильной работы оборудования необходимо немедленно прекратить работу и сообщить об этом ответственному лицу.

Таким образом, соблюдение требований безопасности при работе с персональным компьютером позволяет снизить вредное воздействие на организм пользователя, сохранить работоспособность и предотвратить опасные ситуации. Правильная организация рабочего места, освещения, микроклимата, режима труда и отдыха, а также соблюдение правил электробезопасности являются обязательными условиями безопасной работы с компьютерной техникой.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
|      |      |          |         |      |                       | 121  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## Заключение

В ходе выполнения выпускной квалификационной работы была разработана корпоративная информационная система «Единый справочник контактов». В процессе ВКР проанализирована предметная область, рассмотрены аналоги, определены требования к разрабатываемой системе и выбраны технологии. Разработанное приложение построено на базе клиент-серверной архитектуры, клиентская часть реализована с использованием React и TypeScript, серверная часть – на языке Java с применением Spring Boot, для хранения данных используется СУБД PostgreSQL.

Результатом работы является веб-приложение, предоставляющее пользователям возможность проходить авторизацию, выполнять поиск сотрудников по контактными и организационным данным, просматривать карточки сотрудников, работать с организационной структурой и отображать организации на интерактивной карте, а также разграничение доступа по ролям сотрудника, руководителя и администратора.

Для сопровождения данных предусмотрены функции создания и редактирования сведений о сотрудниках, организациях и подразделениях, импорта информации, а также экспорта справочника. Для контроля действий пользователей реализован журнал аудита, а также предусмотрено управление API-ключами для возможной дальнейшей интеграции с внешними информационными системами.

В ходе тестирования проверены основные пользовательские и административные сценарии работы приложения. В результате тестирования подтверждено, что приложение соответствует заявленным требованиям. Задание на проектирование выполнено в полном объеме.

В дальнейшем система может быть расширена за счет интеграции с кадровыми информационными системами 1С, автоматического обновления сведений о сотрудниках, добавления внутренней связи между пользователями, формирования аналитических отчетов и добавления средств уведомления пользователей об изменениях контактных данных.

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 122  |

## Список использованных источников

1. Распоряжение Правительства Российской Федерации от 16.03.2024 г. № 637-р / [Электронный ресурс] // Правительство России : [сайт]. – URL: <http://government.ru/docs/all/152609/> (дата обращения: 08.04.2026).

2. Григоренко Д. Ю. Одобрены первые программы цифровой трансформации ведомств / Григоренко Д. Ю. [Электронный ресурс] // Правительство России : [сайт]. – URL: <http://government.ru/news/54175/> (дата обращения: 08.04.2026).

3. Мартынова С. Э., Богатырева Т. Г. Цифровые HR -платформы и экосистемы в государственном управлении / Мартынова С. Э., Богатырева Т. Г. [Электронный ресурс] // Среднерусский вестник общественных наук : [сайт]. – URL: <https://orelvestnik.ru/index.php/srvon/article/view/118> (дата обращения: 09.04.2026).

4. Федеральный закон от 27.07.2006 г. № 152-ФЗ О персональных данных / [Электронный ресурс] // Президент России : [сайт]. – URL: <http://www.kremlin.ru/acts/bank/24154> (дата обращения: 09.04.2026).

5. Постановление Правительства Пензенской области от 12.11.2024 № 870-пП "Об утверждении Положения о Министерстве цифрового развития и связи Пензенской области" / [Электронный ресурс] // Официальное опубликование правовых актов : [сайт]. – URL: <http://publication.pravo.gov.ru/document/5800202411140004> (дата обращения: 10.04.2026).

6. Корпоративный телефонный справочник / [Электронный ресурс] // СЗТК : [сайт]. – URL: <https://nowecom.ru/node/442> (дата обращения: 17.04.2026).

7. Web Phone Book Software / [Электронный ресурс] // Phonebook App : [сайт]. – URL: <https://phonebook-app.com/> (дата обращения: 17.04.2026).

8. KDMsoft. Телефонный справочник / [Электронный ресурс] // KDMsoft : [сайт]. – URL: <https://kdmsoft.ru/phonebook.html> (дата обращения: 17.04.2026).

9. Аxiom доверенный репозиторий Java библиотек Реестр российского ПО / [Электронный ресурс] // Реестр программного обеспечения : [сайт]. – URL: <https://reestr.digital.gov.ru/reestr/3467723/> (дата обращения: 18.04.2026).

10. Фреймворк «Аxiom Spring» Реестр российского ПО / [Электронный ресурс] // Реестр программного обеспечения : [сайт]. – URL: <https://reestr.digital.gov.ru/reestr/3523693/> (дата обращения: 18.04.2026).

11. Bates B., Sierra K. Head First Java [Текст] / Bates B., Sierra K. – 2nd ed. –: O'Reilly Media, 2005 – 690 с.

12. JDK 21 Documentation / [Электронный ресурс] // Java Documentation : [сайт]. – URL: <https://docs.oracle.com/javase/21/> (дата обращения: 14.04.2026).

13. Spring Boot. Reference Documentation / [Электронный ресурс] // Spring Documentation : [сайт]. – URL: <https://docs.spring.io/spring-boot/index.html> (дата обращения: 11.04.2026).

14. Cosmina I., Harrop R., Schaefer C., Ho C. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools [Текст] / Cosmina I., Harrop R., Schaefer C., Ho C.: Apress, 2017 – 849 с.

15. Spring Security. Method Security / [Электронный ресурс] // Spring Documentation : [сайт]. – URL: <https://docs.spring.io/spring-security/reference/servlet/authorization/method-security.html> (дата обращения: 11.04.2026).

16. Spring Data JPA. Repository core concepts / [Электронный ресурс] // Spring Documentation : [сайт]. – URL: <https://docs.spring.io/spring-data/jpa/reference/repositories/core-concepts.html> (дата обращения: 11.04.2026).

17. Quick Start – React / [Электронный ресурс] // The React documentation website : [сайт]. – URL: <https://react.dev/learn/describing-the-ui> (дата обращения: 11.04.2026)

18. TypeScript. The TypeScript Handbook / [Электронный ресурс] // TypeScript : [сайт]. – URL: <https://www.typescriptlang.org/docs/handbook/intro.html> (дата обращения: 14.04.2026).

19. Vite. Getting Started / [Электронный ресурс] // Vite : [сайт]. – URL: <https://vite.dev/guide/> (дата обращения: 15.04.2026).

20. React Router Home / [Электронный ресурс] // React Router : [сайт]. – URL: <https://reactrouter.com/home> (дата обращения: 15.04.2026).

21. Ant Design of React / [Электронный ресурс] // Ant Design : [сайт]. – URL: <https://ant.design/docs/react/introduce/> (дата обращения: 15.04.2026).

22. Axios. Interceptors / [Электронный ресурс] // Axios Docs : [сайт]. – URL: <https://axios-http.com/docs/interceptors> (дата обращения: 13.04.2026).

23. TanStack Query. Overview / [Электронный ресурс] // TanStack Query React Docs : [сайт]. – URL: <https://tanstack.com/query/latest/docs/framework/react/overview> (дата обращения: 13.04.2026)

24. Introduction / [Электронный ресурс] // Zustand .docs : [сайт]. – URL: <https://zustand.docs.pmnd.rs/learn/getting-started/introduction> (дата обращения: 16.04.2026).

25. React Leaflet. Introduction / [Электронный ресурс] // React Leaflet Docs : [сайт]. – URL: <https://react-leaflet.js.org/docs/start-introduction/> (дата обращения: 13.04.2026).

26. PostgreSQL. Data Definition / [Электронный ресурс] // PostgreSQL: Documentation : [сайт]. – URL: <https://www.postgresql.org/docs/current/ddl-constraints.html> (дата обращения: 12.04.2026).

27. Maven Getting Started Guide / [Электронный ресурс] // Apache Maven. : [сайт]. – URL: <https://maven.apache.org/guides/getting-started/index.html> (дата обращения: 16.04.2026).

28. OpenIDE – открытая среда разработки / [Электронный ресурс] // OpenIDE : [сайт]. – URL: <https://openide.ru/> (дата обращения: 11.04.2026).

29. DBeaver Documentation / [Электронный ресурс] // DBeaver : [сайт]. – URL: <https://dbeaver.com/docs/dbeaver> (дата обращения: 16.04.2026).

|      |      |          |         |      |                              |      |
|------|------|----------|---------|------|------------------------------|------|
|      |      |          |         |      | <b>ПГУ1.090301.08.001 ПЗ</b> | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                              | 125  |

30. OpenPDF repository / [Электронный ресурс] // LibrePDF : [сайт]. – URL: <https://github.com/LibrePDF/OpenPDF/wiki/Tutorial> (дата обращения: 16.04.2026).

31. N. Sakimura RFC 7519. JSON Web Token (JWT) / N. Sakimura [Электронный ресурс] // IETF (Internet Engineering Task Force) : [сайт]. – URL: <https://datatracker.ietf.org/doc/html/rfc7519> (дата обращения: 13.04.2026).

32. Java API To Access Microsoft Excel Format Files / [Электронный ресурс] // Apache POI. Spreadsheet components : [сайт]. – URL: <https://poi.apache.org/components/spreadsheet/> (дата обращения: 13.04.2026).

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 126  |

## Приложение А

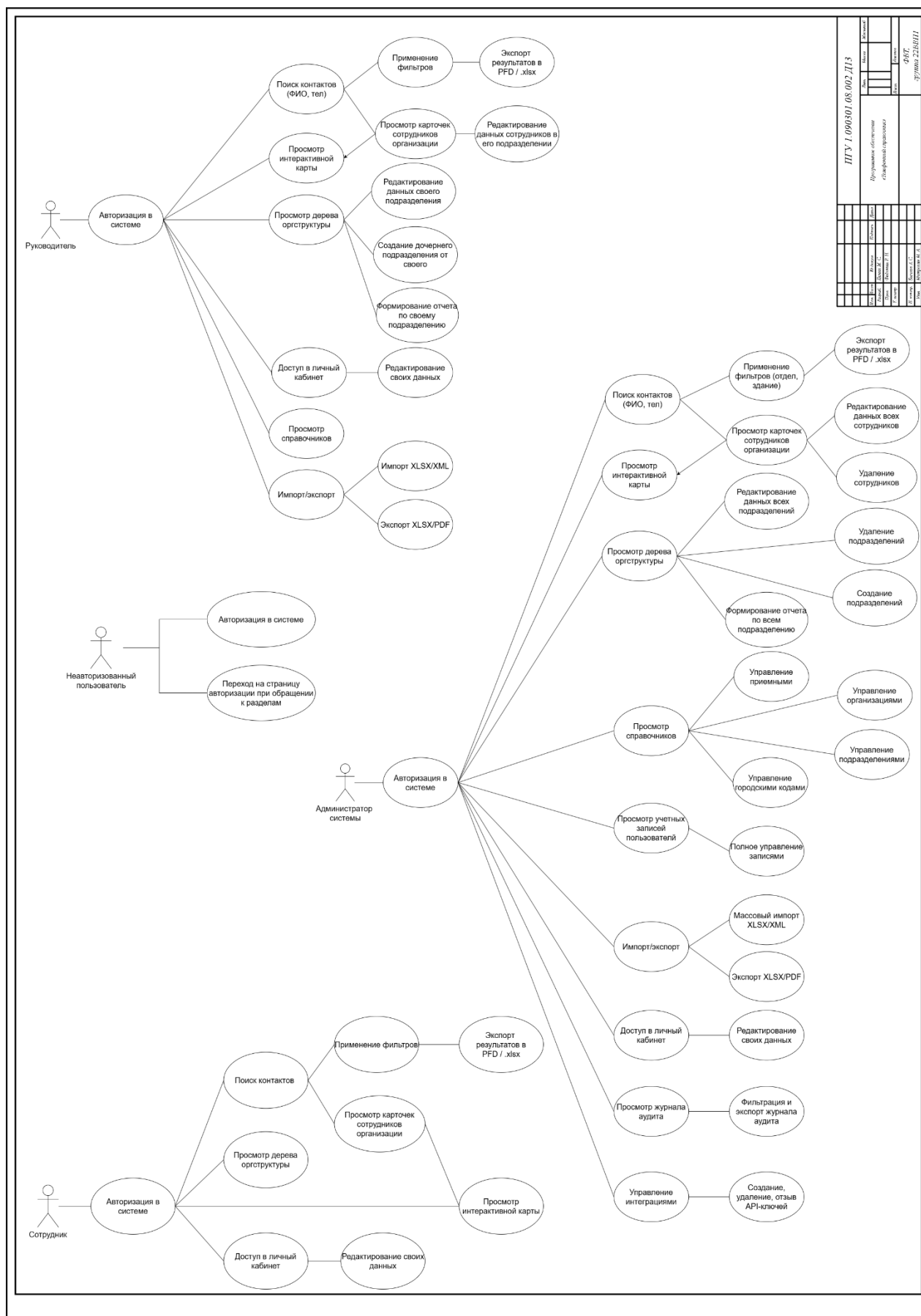
(обязательное)

Программное обеспечение «Телефонный справочник»

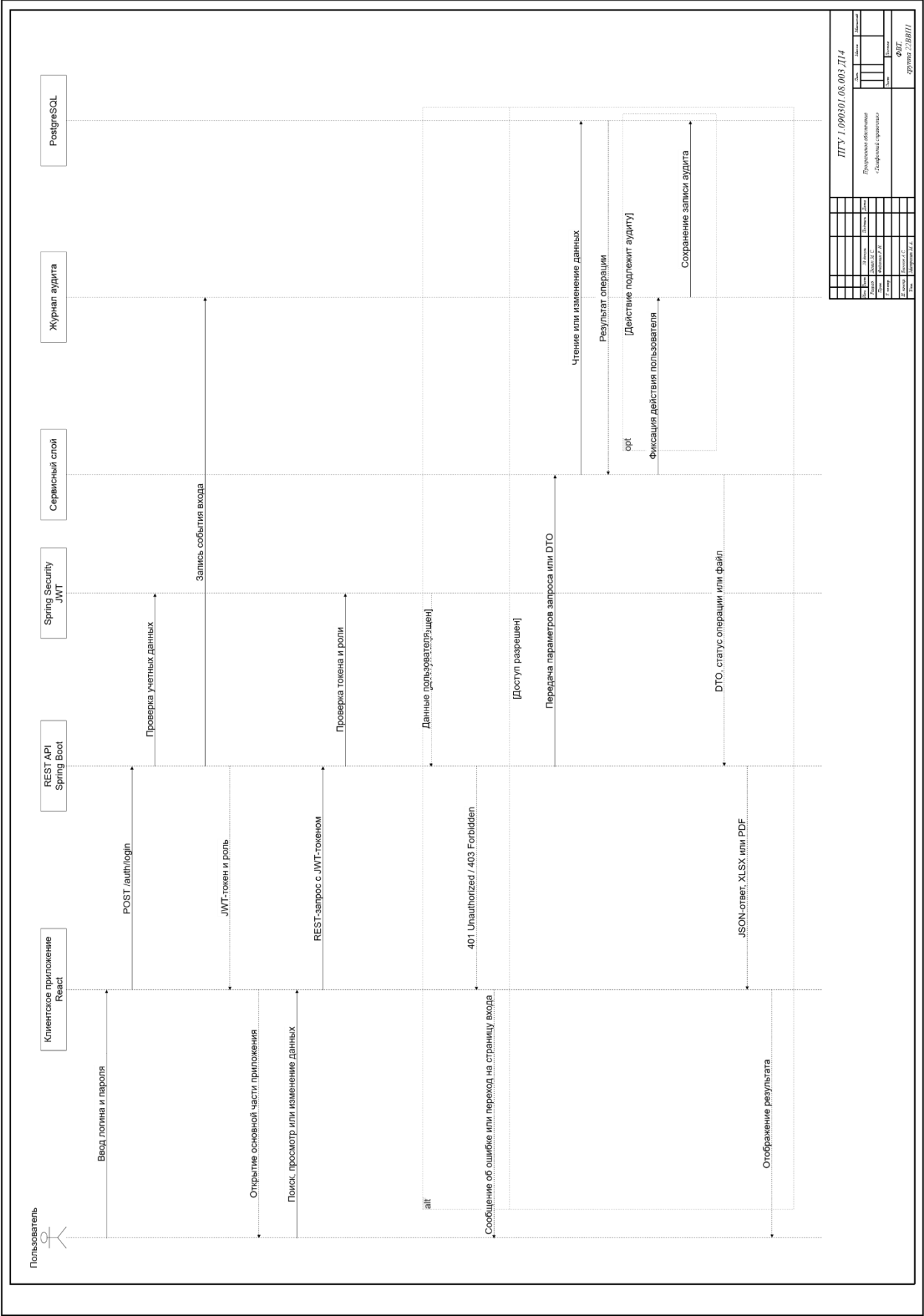
UML-диаграммы

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
|      |      |          |         |      |                       | 127  |
| Изм. | Лист | № докум. | Подпись | Дата |                       |      |

## Приложение А.1 – Диаграмма прецедентов

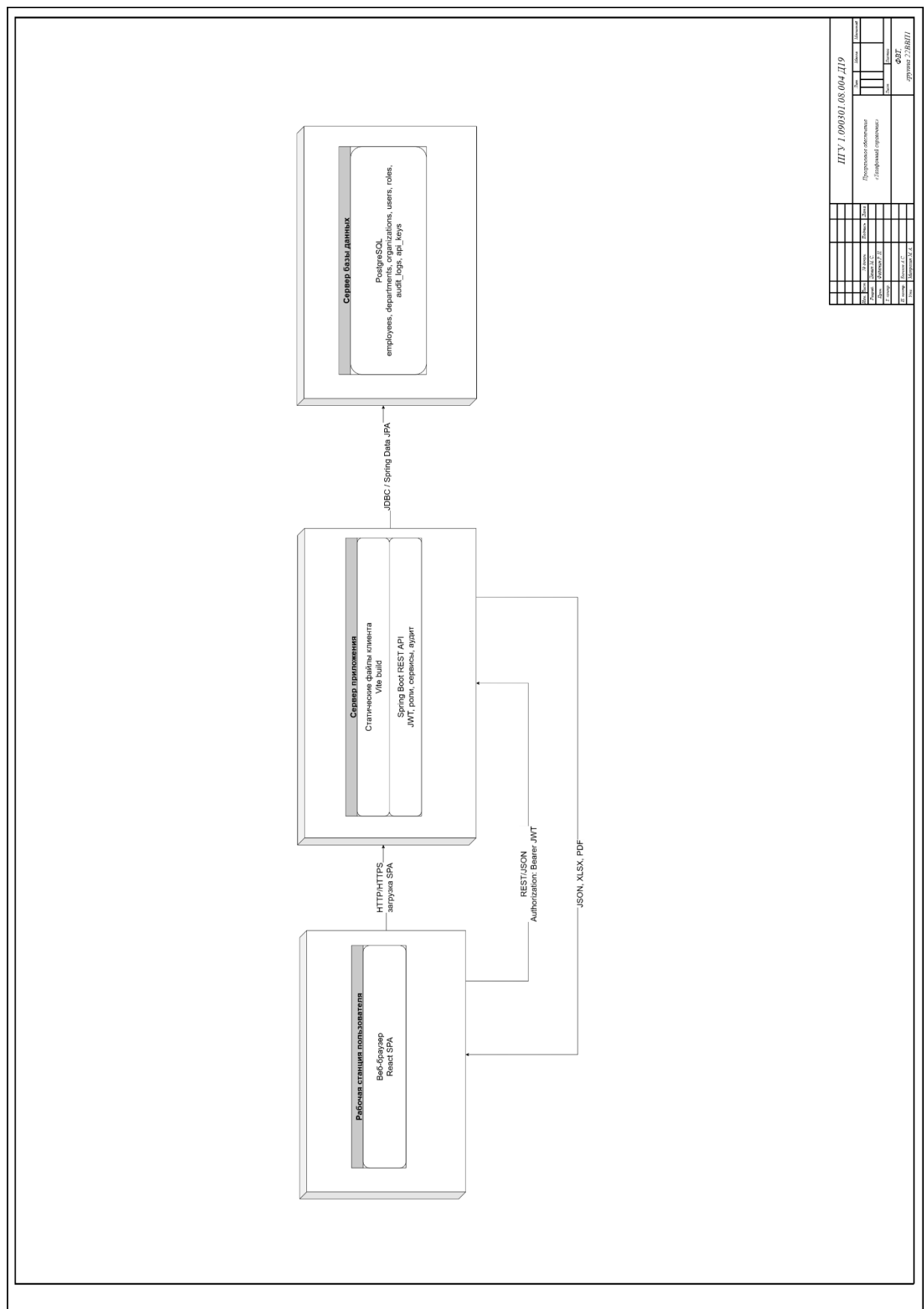


Приложение А.2 – Диаграмма последовательностей



|                        |      |          |         |      |      |      |          |         |      |      |      |
|------------------------|------|----------|---------|------|------|------|----------|---------|------|------|------|
| ПГУ 1.090301.08.001 ПЗ |      |          |         |      |      |      |          |         |      |      |      |
| Регистрация изменений  |      |          |         |      |      |      |          |         |      |      |      |
| Служебный документ     |      |          |         |      |      |      |          |         |      |      |      |
| Изм.                   | Лист | № докум. | Подпись | Дата | Изм. | Лист | № докум. | Подпись | Дата | Изм. | Лист |
|                        |      |          |         |      |      |      |          |         |      |      |      |
| ФОРМ                   |      |          |         |      |      |      |          |         |      |      |      |
| Формы 1/2011           |      |          |         |      |      |      |          |         |      |      |      |

### Приложение А.3 – Диаграмма развертывания



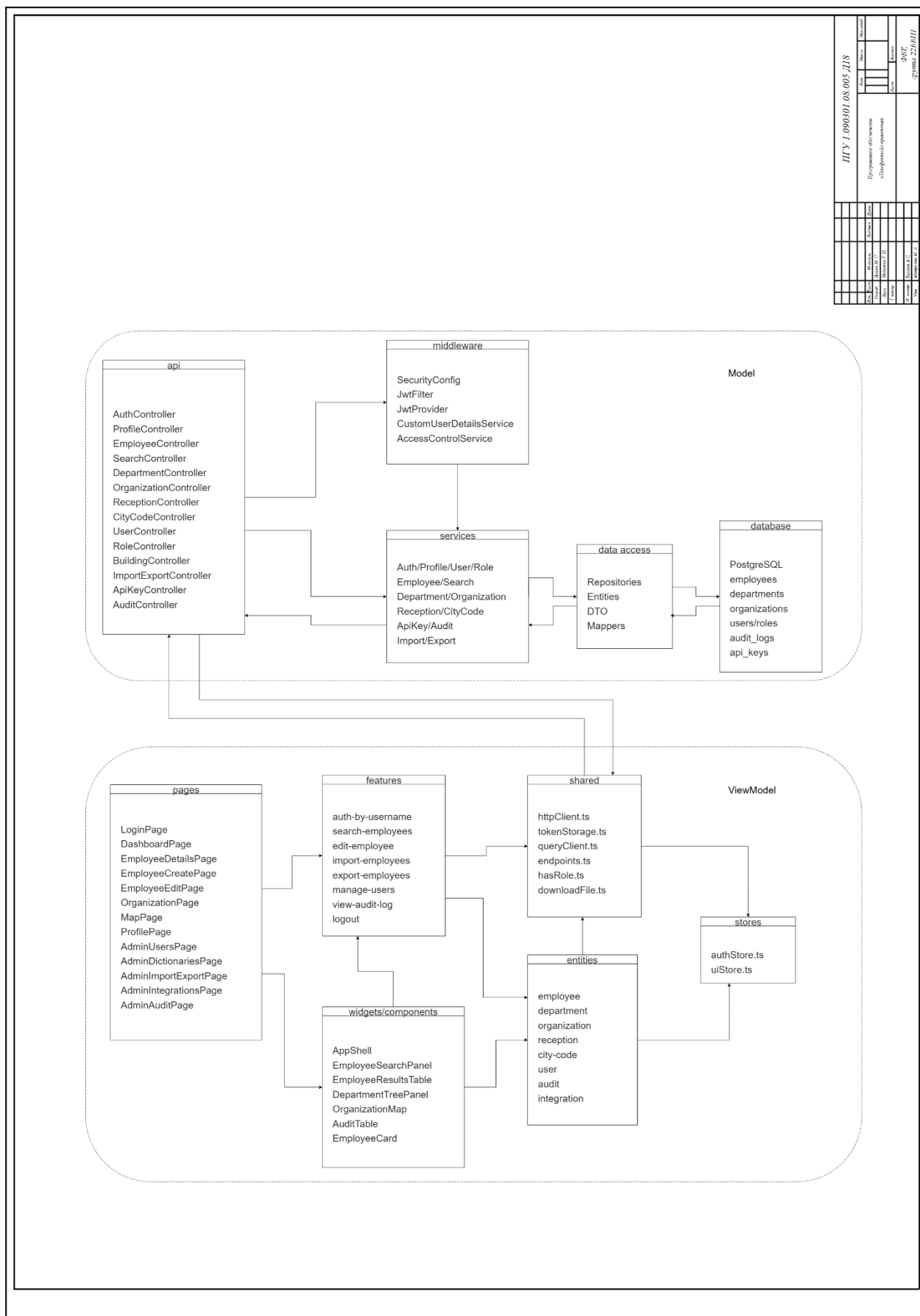
|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

*ПГУ1.090301.08.001 ПЗ*

Лист

130

## Приложение А.4 – Диаграмма компонентов



## Приложение Б

### (обязательное)

#### Программное обеспечение «Телефонный справочник»

#### Листинг программы

##### Файл AuthController.java

```
package com.yourcompany.phonebook.controller.auth;

import com.yourcompany.phonebook.audit.AuditActionType;
import com.yourcompany.phonebook.audit.AuditService;
import com.yourcompany.phonebook.dto.auth.JwtResponse;
import com.yourcompany.phonebook.dto.auth.LoginRequest;
import com.yourcompany.phonebook.security.JwtProvider;
import com.yourcompany.phonebook.security.UserPrincipal;
import jakarta.servlet.http.HttpServletRequest;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/auth")
public class AuthController {

    private final AuthenticationManager authenticationManager;
    private final JwtProvider jwtProvider;
    private final AuditService auditService;

    public AuthController(
        AuthenticationManager authenticationManager,
        JwtProvider jwtProvider,
        AuditService auditService
    ) {
        this.authenticationManager = authenticationManager;
        this.jwtProvider = jwtProvider;
        this.auditService = auditService;
    }

    @PostMapping("/login")
    public JwtResponse login(@RequestBody LoginRequest request,
        HttpServletRequest servletRequest) {
```

```

        Authentication authentication =
authenticationManager.authenticate(
            new
UsernamePasswordAuthenticationToken(request.username(),
request.password())
        );
        UserPrincipal principal = (UserPrincipal)
authentication.getPrincipal();
        String token = jwtProvider.generateToken(principal);
        auditService.log(
            principal,
            AuditActionType.LOGIN,
            "User",
            principal.getId(),
            "Successful login",
            servletRequest
        );
        return new JwtResponse(token, "Bearer",
principal.getUsername(), principal.getRole());
    }
}

```

### Файл SecurityConfig.java

```

package com.yourcompany.phonebook.security;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.Customizer;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.authentication.ProviderManager;
import
org.springframework.security.authentication.dao.DaoAuthenticationPro
vider;
import
org.springframework.security.config.annotation.method.configuration.
EnableMethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecu
rity;
import
org.springframework.security.config.annotation.web.configurers.Abstr
actHttpConfigurer;
import
org.springframework.security.config.http.SessionCreationPolicy;
import
org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;

```

```

import
org.springframework.security.web.authentication.UsernamePasswordAuth
enticationFilter;

@Configuration
@EnableMethodSecurity
public class SecurityConfig {

    private final JwtFilter jwtFilter;
    private final JwtAuthenticationEntryPoint
authenticationEntryPoint;

    public SecurityConfig(JwtFilter jwtFilter,
JwtAuthenticationEntryPoint authenticationEntryPoint) {
        this.jwtFilter = jwtFilter;
        this.authenticationEntryPoint =
authenticationEntryPoint;
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity
http) throws Exception {
        return http
            .cors(Customizer.withDefaults())
            .csrf(AbstractHttpConfigurer::disable)
            .sessionManagement(session ->
session.sessionCreationPolicy(SessionCreationPolicy.STATELESS))
            .exceptionHandling(exception -> exception

.authenticationEntryPoint(authenticationEntryPoint)
                .accessDeniedHandler((request,
response, accessDeniedException) ->
authenticationEntryPoint.writeError(response, 403, "Forbidden")))
                .authorizeHttpRequests(auth -> auth

.requestMatchers("/auth/login").permitAll()
                .requestMatchers("/error").permitAll()
                .requestMatchers(
                    "/",
                    "/home",
                    "/login",
                    "/employees/**",
                    "/organization",
                    "/map",
                    "/profile",
                    "/admin/**",
                    "/403",
                    "/status"
                ).permitAll()
    }
}

```

```

        .requestMatchers("/index.html",
"/assets/**", "/css/**", "/js/**", "/images/**",
"/favicon.ico").permitAll()
        .anyRequest().authenticated()
        .addFilterBefore(jwtFilter,
UsernamePasswordAuthenticationFilter.class)
        .build();
    }

    @Bean
    public AuthenticationManager authenticationManager(
        CustomUserDetailsService userDetailsService,
        PasswordEncoder passwordEncoder
    ) {
        DaoAuthenticationProvider authenticationProvider = new
        DaoAuthenticationProvider(userDetailsService);
        authenticationProvider.setPasswordEncoder(passwordEncoder);
        return new ProviderManager(authenticationProvider);
    }
}

```

### Файл EmployeeController.java

```

package com.yourcompany.phonebook.controller.employee;

import com.yourcompany.phonebook.audit.AuditActionType;
import com.yourcompany.phonebook.audit.annotation.Auditable;
import
com.yourcompany.phonebook.dto.employee.EmployeeRequestDto;
import
com.yourcompany.phonebook.dto.employee.EmployeeResponseDto;
import
com.yourcompany.phonebook.service.employee.EmployeeService;
import org.springframework.http.HttpStatus;
import
org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

```

```

@RestController
@RequestMapping("/api/employees")
public class EmployeeController {

    private final EmployeeService service;

    public EmployeeController(EmployeeService service) {
        this.service = service;
    }

    @GetMapping
    public List<EmployeeResponseDto> findAll(
        @RequestParam(required = false) Long
organizationId,
        @RequestParam(required = false) Long departmentId,
        @RequestParam(required = false) String lastName,
        @RequestParam(required = false) Boolean active
    ) {
        return service.findAll(organizationId, departmentId,
lastName, active);
    }

    @GetMapping("/{id}")
    @Auditable(action = AuditActionType.VIEW_EMPLOYEE)
    public EmployeeResponseDto findById(@PathVariable Long id)
{
        return service.findById(id);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    @PreAuthorize("hasRole('ADMIN')")
    @Auditable(action = AuditActionType.CREATE_EMPLOYEE)
    public EmployeeResponseDto create(@RequestBody
EmployeeRequestDto request) {
        return service.create(request);
    }

    @PutMapping("/{id}")
    @PreAuthorize("hasAnyRole('ADMIN', 'MANAGER')")
    @Auditable(action = AuditActionType.UPDATE_EMPLOYEE)
    public EmployeeResponseDto update(@PathVariable Long id,
@RequestBody EmployeeRequestDto request) {
        return service.update(id, request);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    @PreAuthorize("hasRole('ADMIN')")
    @Auditable(action = AuditActionType.DELETE_EMPLOYEE)
    public void delete(@PathVariable Long id) {
        service.delete(id);
    }
}

```

```

    }
}

```

## Файл SearchService.java

```

package com.yourcompany.phonebook.search;

import com.yourcompany.phonebook.dto.search.SearchResponseDto;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import java.util.ArrayList;
import java.util.List;
import java.util.Locale;

@Service
@Transactional(readOnly = true)
public class SearchService {

    private static final int DEFAULT_PAGE_SIZE = 20;
    private static final int MAX_PAGE_SIZE = 100;

    private final SearchRepository repository;

    public SearchService(SearchRepository repository) {
        this.repository = repository;
    }

    public Page<SearchResponseDto> search(
        String query,
        Long organizationId,
        String department,
        String position,
        String building,
        Pageable pageable
    ) {
        String normalizedQuery = normalize(query);
        boolean hasQuery = !normalizedQuery.isBlank();
        String alternateQuery =
buildAlternateQuery(normalizedQuery);
        String phoneDigits = onlyDigits(normalizedQuery);

        return repository.searchEmployees(
            hasQuery,
            normalizedQuery,
            toLike(normalizedQuery),

```

```

        toLike(alternateQuery),
        phoneDigits,
        toPhoneLike(phoneDigits),
        organizationId,
        toNullableLike(department),
        toNullableLike(position),
        toNullableLike(building),
        normalizePageable(pageable)
    )
    .map(this::toResponse);
}

private SearchResponseDto
toResponse(SearchRepository.SearchResultProjection projection) {
    return new SearchResponseDto(
        projection.getId(),
        projection.getFullName(),
        projection.getPosition(),
        projection.getDepartment(),
        projection.getOrganization(),
        collectPhones(projection),
        projection.getEmail(),
        projection.getLocation(),
        projection.getOffice(),
        projection.getRank()
    );
}

private List<String>
collectPhones(SearchRepository.SearchResultProjection projection) {
    List<String> phones = new ArrayList<>();
    addIfPresent(phones, projection.getWorkPhone());
    addIfPresent(phones, projection.getInternalPhone());
    addIfPresent(phones, projection.getMobilePhone());
    return phones;
}

private void addIfPresent(List<String> values, String
value) {
    if (value != null && !value.isBlank()) {
        values.add(value.trim());
    }
}

private Pageable normalizePageable(Pageable pageable) {
    if (pageable == null || pageable.isUnpaged()) {
        return PageRequest.of(0, DEFAULT_PAGE_SIZE);
    }
    int size = Math.min(pageable.getPageSize(),
MAX_PAGE_SIZE);
    return PageRequest.of(pageable.getPageNumber(), size);
}

```

```

private String normalize(String value) {
    if (value == null || value.isBlank()) {
        return "";
    }
    return value
        .replaceAll("\\s+", " ")
        .trim()
        .toLowerCase(Locale.ROOT);
}

private String toNullableLike(String value) {
    if (value == null || value.isBlank()) {
        return null;
    }
    return toLike(value);
}

private String toLike(String value) {
    return "%" + value.trim().toLowerCase(Locale.ROOT) +
"%";
}

private String toPhoneLike(String phoneDigits) {
    return phoneDigits == null || phoneDigits.isBlank() ?
"" : "%" + phoneDigits + "%";
}

private String onlyDigits(String value) {
    return value == null ? "" : value.replaceAll("\\D",
"");
}

private String buildAlternateQuery(String query) {
    if (containsLatin(query)) {
        return transliterateLatinToCyrillic(query);
    }
    if (containsCyrillic(query)) {
        return transliterateCyrillicToLatin(query);
    }
    return query;
}

private boolean containsLatin(String value) {
    return value.chars().anyMatch(character ->
        (character >= 'a' && character <= 'z') ||
(character >= 'A' && character <= 'Z'));
}

private boolean containsCyrillic(String value) {
    return value.chars().anyMatch(character ->

```

```

        (character >= '\u0430' && character <=
'\u044f') || (character >= '\u0410' && character <= '\u042f')
        || character == '\u0451' || character
== '\u0401');
    }

```

```

private String transliterateLatinToCyrillic(String value) {
    String lower = value.toLowerCase(Locale.ROOT);
    StringBuilder result = new StringBuilder();
    int index = 0;
    while (index < lower.length()) {
        String rest = lower.substring(index);
        if (rest.startsWith("shch")) {
            result.append('\u0449');
            index += 4;
        } else if (rest.startsWith("sch")) {
            result.append('\u0449');
            index += 3;
        } else if (rest.startsWith("yo") ||
rest.startsWith("jo")) {
            result.append('\u0451');
            index += 2;
        } else if (rest.startsWith("zh")) {
            result.append('\u0436');
            index += 2;
        } else if (rest.startsWith("kh")) {
            result.append('\u0445');
            index += 2;
        } else if (rest.startsWith("ts")) {
            result.append('\u0446');
            index += 2;
        } else if (rest.startsWith("ch")) {
            result.append('\u0447');
            index += 2;
        } else if (rest.startsWith("sh")) {
            result.append('\u0448');
            index += 2;
        } else if (rest.startsWith("yu") ||
rest.startsWith("ju")) {
            result.append('\u044e');
            index += 2;
        } else if (rest.startsWith("ya") ||
rest.startsWith("ja")) {
            result.append('\u044f');
            index += 2;
        } else if (rest.startsWith("ye") ||
rest.startsWith("je")) {
            result.append('\u0435');
            index += 2;
        } else {
            result.append(latinCharToCyrillic(lower.charAt(index)));
        }
    }
}

```

```

        index++;
    }
}
return result.toString();
}

private char latinCharToCyrillic(char value) {
    return switch (value) {
        case 'a' -> '\u0430';
        case 'b' -> '\u0431';
        case 'v', 'w' -> '\u0432';
        case 'g' -> '\u0433';
        case 'd' -> '\u0434';
        case 'e' -> '\u0435';
        case 'z' -> '\u0437';
        case 'i' -> '\u0438';
        case 'j' -> '\u0439';
        case 'k' -> '\u043a';
        case 'l' -> '\u043b';
        case 'm' -> '\u043c';
        case 'n' -> '\u043d';
        case 'o' -> '\u043e';
        case 'p' -> '\u043f';
        case 'r' -> '\u0440';
        case 's' -> '\u0441';
        case 't' -> '\u0442';
        case 'u' -> '\u0443';
        case 'f' -> '\u0444';
        case 'h' -> '\u0445';
        case 'c' -> '\u043a';
        case 'y' -> '\u044b';
        default -> value;
    };
}

private String transliterateCyrillicToLatin(String value) {
    StringBuilder result = new StringBuilder();
    for (int i = 0; i < value.length(); i++) {
        result.append(cyrillicCharToLatin(value.charAt(i)));
    }
    return result.toString();
}

private String cyrillicCharToLatin(char value) {
    return switch (Character.toLowerCase(value)) {
        case '\u0430' -> "a";
        case '\u0431' -> "b";
        case '\u0432' -> "v";
        case '\u0433' -> "g";
        case '\u0434' -> "d";
        case '\u0435', '\u044d' -> "e";
    };
}

```

```

        case '\u0451' -> "yo";
        case '\u0436' -> "zh";
        case '\u0437' -> "z";
        case '\u0438' -> "i";
        case '\u0439' -> "y";
        case '\u043a' -> "k";
        case '\u043b' -> "l";
        case '\u043c' -> "m";
        case '\u043d' -> "n";
        case '\u043e' -> "o";
        case '\u043f' -> "p";
        case '\u0440' -> "r";
        case '\u0441' -> "s";
        case '\u0442' -> "t";
        case '\u0443' -> "u";
        case '\u0444' -> "f";
        case '\u0445' -> "kh";
        case '\u0446' -> "ts";
        case '\u0447' -> "ch";
        case '\u0448' -> "sh";
        case '\u0449' -> "shch";
        case '\u044b' -> "y";
        case '\u044e' -> "yu";
        case '\u044f' -> "ya";
        case '\u044a', '\u044c' -> "";
        default -> String.valueOf(value);
    };
}
}

```

### Файл AccessControlService.java

```

package com.yourcompany.phonebook.security;

import com.yourcompany.phonebook.entity.Department;
import com.yourcompany.phonebook.entity.Employee;
import com.yourcompany.phonebook.entity.User;
import com.yourcompany.phonebook.repository.UserRepository;
import
org.springframework.security.access.AccessDeniedException;
import org.springframework.security.core.Authentication;
import
org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Service;
import
org.springframework.transaction.annotation.Transactional;

import java.util.HashSet;
import java.util.Set;

```

```

@Service
@Transactional(readonly = true)
public class AccessControlService {

    private final UserRepository userRepository;

    public AccessControlService(UserRepository userRepository)
    {
        this.userRepository = userRepository;
    }

    public void assertCanManageEmployee(Employee employee) {
        if (isAdmin()) {
            return;
        }
        assertManager();
        assertDepartmentInManagerScope(employee == null ? null
: employee.getDepartment());
    }

    public void assertCanManageDepartment(Department
department) {
        if (isAdmin()) {
            return;
        }
        assertManager();
        assertDepartmentInManagerScope(department);
    }

    public void assertCanCreateDepartment(Department parent) {
        if (isAdmin()) {
            return;
        }
        assertManager();
        if (parent == null) {
            throw new AccessDeniedException("Manager cannot
create root departments");
        }
        assertDepartmentInManagerScope(parent);
    }

    public boolean isAdmin() {
        UserPrincipal principal = currentPrincipal();
        return principal != null &&
RoleType.ADMIN.name().equals(principal.getRole());
    }

    public boolean isManager() {
        UserPrincipal principal = currentPrincipal();
        return principal != null &&
RoleType.MANAGER.name().equals(principal.getRole());
    }
}

```

```

        private void assertManager() {
            if (!isManager()) {
                throw new AccessDeniedException("Access denied");
            }
        }

        private void assertDepartmentInManagerScope(Department
targetDepartment) {
            Department managerDepartment =
currentManagerDepartment();
            if (managerDepartment == null || targetDepartment ==
null || targetDepartment.getId() == null) {
                throw new AccessDeniedException("Manager is not
linked to a department");
            }

            Long managerDepartmentId = managerDepartment.getId();
            Set<Long> visited = new HashSet<>();
            Department cursor = targetDepartment;
            while (cursor != null && cursor.getId() != null &&
visited.add(cursor.getId())) {
                if (cursor.getId().equals(managerDepartmentId)) {
                    return;
                }
                cursor = cursor.getParent();
            }

            throw new AccessDeniedException("Manager can edit only
own department and child departments");
        }

        private Department currentManagerDepartment() {
            UserPrincipal principal = currentPrincipal();
            if (principal == null) {
                throw new AccessDeniedException("Authentication
required");
            }
            User user = userRepository.findById(principal.getId())
                .orElseThrow(() -> new
AccessDeniedException("Authentication user was not found"));
            Employee employee = user.getEmployee();
            return employee == null ? null :
employee.getDepartment();
        }

        private UserPrincipal currentPrincipal() {
            Authentication authentication =
SecurityContextHolder.getContext().getAuthentication();
            if (authentication == null ||
!(authentication.getPrincipal() instanceof UserPrincipal principal))
{

```

```

        return null;
    }
    return principal;
}
}

```

## Файл ImportExportController.java

```

package com.yourcompany.phonebook.importexport.controller;

import com.yourcompany.phonebook.audit.AuditActionType;
import com.yourcompany.phonebook.audit.AuditService;
import
com.yourcompany.phonebook.importexport.dto.ImportResultDto;
import
com.yourcompany.phonebook.importexport.excel.ExcelExportService;
import
com.yourcompany.phonebook.importexport.excel.ExcelImportService;
import
com.yourcompany.phonebook.importexport.pdf.PdfExportService;
import
com.yourcompany.phonebook.importexport.xml.XmlImportService;
import com.yourcompany.phonebook.repository.EmployeeRepository;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

@RestController
public class ImportExportController {

    private static final MediaType XLSX_MEDIA_TYPE =
MediaType.parseMediaType(
        "application/vnd.openxmlformats-
officedocument.spreadsheetml.sheet"
    );

    private final ExcelImportService excelImportService;
    private final XmlImportService xmlImportService;
    private final ExcelExportService excelExportService;
    private final PdfExportService pdfExportService;
    private final EmployeeRepository employeeRepository;
    private final AuditService auditService;

```

```

public ImportExportController(
    ExcelImportService excelImportService,
    XmlImportService xmlImportService,
    ExcelExportService excelExportService,
    PdfExportService pdfExportService,
    EmployeeRepository employeeRepository,
    AuditService auditService
) {
    this.excelImportService = excelImportService;
    this.xmlImportService = xmlImportService;
    this.excelExportService = excelExportService;
    this.pdfExportService = pdfExportService;
    this.employeeRepository = employeeRepository;
    this.auditService = auditService;
}

@PostMapping(value = "/api/import/employees", consumes =
MediaType.MULTIPART_FORM_DATA_VALUE)
@PreAuthorize("hasAnyRole('ADMIN', 'MANAGER')")
public ImportResultDto
importEmployees(@RequestParam("file") MultipartFile file) {
    ImportResultDto result =
excelImportService.importEmployees(file);
    auditService.log(
        AuditActionType.IMPORT_EMPLOYEES,
        "Employee",
        null,
        "Imported employees: success=" +
result.successCount() + ", errors=" + result.errorCount()
    );
    return result;
}

@PostMapping(value = "/api/import/employees/xml", consumes
= MediaType.MULTIPART_FORM_DATA_VALUE)
@PreAuthorize("hasAnyRole('ADMIN', 'MANAGER')")
public ImportResultDto
importEmployeesXml(@RequestParam("file") MultipartFile file) {
    ImportResultDto result =
xmlImportService.importEmployees(file);
    auditService.log(
        AuditActionType.IMPORT_EMPLOYEES,
        "Employee",
        null,
        "Imported employees from XML: success=" +
result.successCount() + ", errors=" + result.errorCount()
    );
    return result;
}

@GetMapping("/api/import/template/employees")
@PreAuthorize("hasAnyRole('ADMIN', 'MANAGER')")

```

```

        public ResponseEntity<byte[]> employeeImportTemplate() {
            byte[] content =
excelExportService.createEmployeeTemplate();
            auditService.log(
AuditActionType.DOWNLOAD_EMPLOYEE_IMPORT_TEMPLATE,
                "Employee",
                null,
                "Downloaded employee import template"
            );
            return file(content, XLSX_MEDIA_TYPE, "employees-
template.xlsx");
        }

        @GetMapping("/api/export/employees")
        @PreAuthorize("hasAnyRole('ADMIN', 'MANAGER', 'EMPLOYEE')")
        public ResponseEntity<byte[]>
exportEmployees(@RequestParam(required = false) String department) {
            byte[] content =
excelExportService.exportEmployees(department);
            auditService.log(
                AuditActionType.EXPORT_EMPLOYEES,
                "Employee",
                null,
                exportDetails("xlsx", department)
            );
            return file(content, XLSX_MEDIA_TYPE,
"employees.xlsx");
        }

        @GetMapping("/api/export/employees/pdf")
        @PreAuthorize("hasAnyRole('ADMIN', 'MANAGER', 'EMPLOYEE')")
        public ResponseEntity<byte[]>
exportEmployeesPdf(@RequestParam(required = false) String
department) {
            byte[] content =
pdfExportService.exportEmployees(department);
            auditService.log(
                AuditActionType.EXPORT_EMPLOYEES_PDF,
                "Employee",
                null,
                exportDetails("pdf", department)
            );
            return file(content, MediaType.APPLICATION_PDF,
"employees.pdf");
        }

        private String exportDetails(String format, String
department) {
            long count = department == null || department.isBlank()
                ? employeeRepository.count()

```

```

        :
employeeRepository.countByDepartmentNameContainingIgnoreCase(departm
ent.trim());
        String details = "Exported employees: format=" + format
+ ", count=" + count;
        if (department != null && !department.isBlank()) {
            details += ", department=" + department.trim();
        }
        return details;
    }

    private ResponseEntity<byte[]> file(byte[] content,
MediaType mediaType, String filename) {
        return ResponseEntity.ok()
            .contentType(mediaType)
            .contentLength(content.length)
            .header(HttpHeaders.CONTENT_DISPOSITION,
"attachment; filename=\"" + filename + "\"")
            .body(content);
    }
}

EmployeeExcelImporter.java
@Transactional
public ImportResultDto importEmployees(MultipartFile file) {
    List<String> errors = new ArrayList<>();
    List<Employee> employees = new ArrayList<>();

    try (Workbook workbook = new
XSSFWorkbook(file.getInputStream())) {
        if (workbook.getNumberOfSheets() == 0) {
            throw new BadRequestException("Excel-файл не
содержит листов");
        }

        Sheet sheet = workbook.getSheetAt(0);
        Map<String, Integer> columns =
readHeader(sheet.getRow(0));
        validateHeader(columns, errors);
        if (!errors.isEmpty()) {
            return ImportResultDto.of(0, errors);
        }

        for (int rowIndex = 1; rowIndex <=
sheet.getLastRowNum(); rowIndex++) {
            Row row = sheet.getRow(rowIndex);
            if (isEmpty(row)) {
                continue;
            }
            parseEmployee(row, columns, rowIndex + 1,
errors).ifPresent(employees::add);
        }
    }
}

```

```

        employeeRepository.saveAll(employees);
        return ImportResultDto.of(employees.size(), errors);
    } catch (IOException | RuntimeException exception) {
        if (exception instanceof BadRequestException
badRequestException) {
            throw badRequestException;
        }
        throw new BadRequestException("Не удалось прочитать
Excel-файл: " + exception.getMessage());
    }
}

    private void validateHeader(Map<String, Integer> columns,
List<String> errors) {
        if (findColumn(columns, "фамилия", "lastname",
"lastName").isEmpty()) {
            errors.add("Не найдена обязательная колонка: Фамилия");
        }
        if (findColumn(columns, "имя", "firstname",
"firstName").isEmpty()) {
            errors.add("Не найдена обязательная колонка: Имя");
        }
        if (findColumn(columns, "отдел", "подразделение",
"department").isEmpty()) {
            errors.add("Не найдена обязательная колонка: Отдел");
        }
    }

    private Optional<Employee> parseEmployee(
        Row row,
        Map<String, Integer> columns,
        int excelRowNumber,
        List<String> errors
    ) {
        String lastName = value(row, columns, "фамилия",
"lastname", "lastName");
        String firstName = value(row, columns, "имя", "firstname",
"firstName");
        String workPhone = value(row, columns, "рабочийтелефон",
"телефон", "workphone", "phone");
        String email = value(row, columns, "email", "почта",
"электроннаяпочта");
        String position = value(row, columns, "должность",
"position");
        String departmentName = value(row, columns, "отдел",
"подразделение", "department");
        String location = value(row, columns, "локация", "здание",
"location", "building");
        String office = value(row, columns, "кабинет", "офис",
"office");
    }

```

```

List<String> rowErrors = new ArrayList<>();
if (lastName.isBlank()) {
    rowErrors.add("фамилия не заполнена");
}
if (firstName.isBlank()) {
    rowErrors.add("имя не заполнено");
}
if (departmentName.isBlank()) {
    rowErrors.add("отдел не заполнен");
}
if (!email.isBlank() && !ValidationUtils.isEmail(email)) {
    rowErrors.add("email неверный");
}

Department department = null;
if (!departmentName.isBlank()) {
    department =
departmentRepository.findFirstByNameIgnoreCase(departmentName)
        .orElse(null);
    if (department == null) {
        rowErrors.add("отдел \"" + departmentName + "\"" не
найден");
    }
}

if (department != null) {
    try {
accessControlService.assertCanManageDepartment(department);
    } catch (RuntimeException exception) {
        rowErrors.add("нет прав на импорт в отдел \"" +
departmentName + "\"");
    }
}

if (!rowErrors.isEmpty()) {
    rowErrors.forEach(error -> errors.add("Строка " +
excelRowNumber + ": " + error));
    return Optional.empty();
}

Employee employee = new Employee();
employee.setLastName(lastName);
employee.setFirstName(firstName);
employee.setWorkPhone(blankToNull(workPhone));
employee.setEmail(blankToNull(email));
employee.setPosition(blankToNull(position));
employee.setDepartment(department);
employee.setLocation(blankToNull(location));
employee.setOffice(blankToNull(office));
employee.setActive(true);
return Optional.of(employee);

```

```
}
```

## Файл OrganizationMap.tsx

```
function getBuildingPosition(building: Building): [number,
number] {
    return [Number(building.latitude),
Number(building.longitude)];
}

function MapViewport({ points }: { points: Building[] }) {
    const map = useMap();

    useEffect(() => {
        if (points.length === 1) {
            map.setView(getBuildingPosition(points[0]),
SELECTED_ORGANIZATION_ZOOM, { animate: true });
            return;
        }

        if (points.length > 1) {
            map.fitBounds(points.map(getBuildingPosition), {
                animate: true,
                maxZoom: SELECTED_ORGANIZATION_ZOOM,
                padding: [40, 40]
            });
            return;
        }

        map.setView(MOSCOW_CENTER, MOSCOW_ZOOM, { animate: true });
    }, [map, points]);

    return null;
}

const buildings = useQuery({
    queryKey: ['buildings'],
    queryFn: () => organizationApi.findBuildings()
});

const filteredBuildings = useMemo(
    () =>
        (buildings.data ?? []).filter(
            (building) =>
                (!selectedOrganizationId || building.organizationId ===
selectedOrganizationId) &&
                (!addressQuery ||
building.address?.toLowerCase().includes(addressQuery.toLowerCase())
||
```

|      |      |          |         |      |                       |      |
|------|------|----------|---------|------|-----------------------|------|
|      |      |          |         |      | ПГУ1.090301.08.001 ПЗ | Лист |
| Изм. | Лист | № докум. | Подпись | Дата |                       | 151  |

```

building.name.toLowerCase().includes(addressQuery.toLowerCase()))
    ),
    [addressQuery, buildings.data, selectedOrganizationId]
  );

  const points = useMemo(
    () => filteredBuildings.filter((building) =>
building.latitude !== null && building.longitude !== null),
    [filteredBuildings]
  );

  <MapContainer center={MOSCOW_CENTER} zoom={MOSCOW_ZOOM}
style={{ minHeight: 560, width: '100%' }}>
    <MapViewport points={points} />
    <TileLayer
      attribution="&copy; OpenStreetMap contributors"
      url="https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png"
    />
    {points.map((building) => (
      <Marker
        key={building.id}
        icon={organizationMarkerIcon}
        position={[Number(building.latitude),
Number(building.longitude)]}
      >
        <Popup>
          <strong>{building.name}</strong>
          <br />
          {building.address}
          <br />
          <Tag>{building.employeeCount} сотрудников</Tag>
          <Tag>{building.officeCount} кабинетов</Tag>
        </Popup>
      </Marker>
    ))}
  </MapContainer>

```

### Файл EmployeeSearchPanel.tsx

```

import { Button, Form, Space, Typography } from 'antd';
import { useEffect } from 'react';
import { SearchFilters, type SearchFiltersValue } from
'@features/search-employees/ui/SearchFilters';
import { SearchInput } from '@features/search-
employees/ui/SearchInput';

export type EmployeeSearchFormValue = SearchFiltersValue & {
  q: string;
};

```

```

type EmployeeSearchPanelProps = {
  initialValue?: EmployeeSearchFormValue;
  loading?: boolean;
  onSearch: (value: EmployeeSearchFormValue) => void;
};

export function EmployeeSearchPanel({ initialValue, loading,
onSearch }: EmployeeSearchPanelProps) {
  const [form] = Form.useForm<EmployeeSearchFormValue>();
  const values = Form.useWatch([], form);

  useEffect(() => {
    if (initialValue) {
      form.setFieldsValue(initialValue);
    }
  }, [form, initialValue]);

  useEffect(() => {
    const query = values?.q?.trim() ?? '';
    const canSearch = query.length >= 2 ||
/^\\d{1,2}$/.test(query);

    if (!canSearch) {
      return;
    }

    const timer = window.setTimeout(() => onSearch({ ...(values
?? { q: query }), q: query }), 300);
    return () => window.clearTimeout(timer);
  }, [onSearch, values]);

  return (
    <div className="surface">
      <div className="surface__body">
        <Form
          form={form}
          layout="vertical"
          initialValues={initialValue}
          onFinish={(value) => onSearch(value)}
        >
          <Form.Item name="q" label="Поиск">
            <SearchInput onSearch={() => form.submit()} />
          </Form.Item>
          <Typography.Text type="secondary" style={{ display:
'block', marginTop: -8, marginBottom: 12 }}>
            Можно искать транслитом: Ivanov / Иванов
          </Typography.Text>
          <SearchFilters />
          <Space>
            <Button type="primary" htmlType="submit"
loading={loading}>

```

```

        Найти
    </Button>
    <Button
        onClick={() => {
            form.resetFields();
            onSearch({ q: '' });
        }}
    >
        Сбросить
    </Button>
</Space>
</Form>
</div>
</div>
);
}

```

### Файл AuditController.java

```

package com.yourcompany.phonebook.audit;

import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.format.annotation.DateTimeFormat;
import org.springframework.http.HttpHeaders;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import
org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.nio.charset.StandardCharsets;
import java.util.List;

@RestController
@RequestMapping("/api/audit")
@PreAuthorize("hasRole('ADMIN')")
public class AuditController {

    private final AuditService service;

    public AuditController(AuditService service) {
        this.service = service;
    }

```

```

    @GetMapping
    public Page<AuditLog> findAll(
        @RequestParam(required = false) Long userId,
        @RequestParam(required = false) String username,
        @RequestParam(required = false) AuditActionType
action,
        @RequestParam(required = false) String entityType,
        @RequestParam(required = false) @DateTimeFormat(iso
= DateTimeFormat.ISO.DATE) LocalDate date,
        @RequestParam(required = false) @DateTimeFormat(iso
= DateTimeFormat.ISO.DATE_TIME) LocalDateTime from,
        @RequestParam(required = false) @DateTimeFormat(iso
= DateTimeFormat.ISO.DATE_TIME) LocalDateTime to,
        Pageable pageable
    ) {
        LocalDateTime effectiveFrom = date == null ? from :
date.atStartOfDay();
        LocalDateTime effectiveTo = date == null ? to :
date.plusDays(1).atStartOfDay();
        return service.findByFilters(userId, username, action,
entityType, effectiveFrom, effectiveTo, pageable);
    }

    @GetMapping("/export")
    public ResponseEntity<byte[]> export(
        @RequestParam(required = false) Long userId,
        @RequestParam(required = false) String username,
        @RequestParam(required = false) AuditActionType
action,
        @RequestParam(required = false) String entityType,
        @RequestParam(required = false) @DateTimeFormat(iso
= DateTimeFormat.ISO.DATE) LocalDate date,
        @RequestParam(required = false) @DateTimeFormat(iso
= DateTimeFormat.ISO.DATE_TIME) LocalDateTime from,
        @RequestParam(required = false) @DateTimeFormat(iso
= DateTimeFormat.ISO.DATE_TIME) LocalDateTime to
    ) {
        LocalDateTime effectiveFrom = date == null ? from :
date.atStartOfDay();
        LocalDateTime effectiveTo = date == null ? to :
date.plusDays(1).atStartOfDay();
        List<AuditLog> logs = service.findForExport(
            userId,
            username,
            action,
            entityType,
            effectiveFrom,
            effectiveTo
        );
        byte[] content =
toCsv(logs).getBytes(StandardCharsets.UTF_8);
        return ResponseEntity.ok()

```

```

.contentType(MediaType.parseMediaType("text/csv; charset=UTF-8"))
    .header(HttpHeaders.CONTENT_DISPOSITION,
"attachment; filename=\"audit-report.csv\"")
    .body(content);
}

private String toCsv(List<AuditLog> logs) {
    StringBuilder builder = new StringBuilder();
    builder.append('\uFEFF');

builder.append("id,createdAt,username,action,entityType,entityId,ipA
ddress,details\n");
    for (AuditLog log : logs) {
        builder.append(csv(log.getId())).append(',')

.append(csv(log.getCreatedAt())).append(',')
        .append(csv(log.getUsername())).append(',')
        .append(csv(log.getAction())).append(',')

.append(csv(log.getEntityType())).append(',')
        .append(csv(log.getEntityId())).append(',')

.append(csv(log.getIpAddress())).append(',')

.append(csv(log.getDetails())).append('\n');
    }
    return builder.toString();
}

private String csv(Object value) {
    if (value == null) {
        return "";
    }
    String text = value.toString().replace("\"", "\"\"");
    return "\"" + text + "\"";
}
}

```

# Приложение В

## (обязательное)

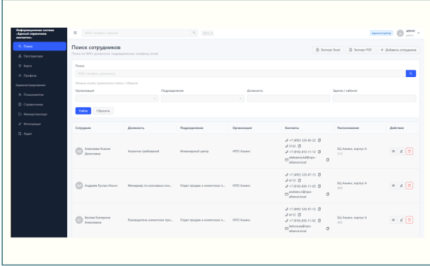
### Программное обеспечение «Телефонный справочник»

#### Презентация

Выпускная квалификационная работа

## Программное обеспечение «Телефонный справочник»

Корпоративная информационная система «Единый справочник контактов» для централизованного хранения, поиска и сопровождения контактной информации.



|                                |                                   |
|--------------------------------|-----------------------------------|
| Выполнил студент группы 22ВВП1 | Руководитель                      |
| Демин Максим Сергеевич         | К. Т. Н., доцент<br>Федонин Р. Н. |

Пенза 2026

### АКТУАЛЬНОСТЬ

## Единый справочник решает проблему разрозненных контактных данных

В организациях контактные сведения часто находятся в таблицах, устаревших списках и внутренних документах. Это замедляет коммуникацию и усложняет сопровождение данных.

**Дублирование**

одни и те же сведения ведутся в разных местах

**Неактуальность**

нет единого источника достоверных контактов

**Долгий поиск**

сотрудники тратят время на уточнение телефонов и отделов

**Слабый контроль**

сложно разграничить доступ и отследить изменения

**Цифровой справочник повышает актуальность данных и сокращает непроизводительные потери времени.**

02

### ЦЕЛЬ И ЗАДАЧИ

## Цель: разработать ИС «Единый справочник контактов»

**Цель работы**

Разработать корпоративную информационную систему для централизованного хранения, поиска и сопровождения контактной информации.

**Назначение**

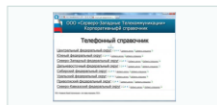
Пользователь быстро находит сотрудника, а администратор сопровождает структуру, роли, импорт/экспорт и аудит.

- 1 проанализировать предметную область и аналоги
- 2 сформулировать требования к системе
- 3 выбрать технологии и инструменты разработки
- 4 спроектировать архитектуру и структуру БД
- 5 реализовать клиентскую и серверную части
- 6 реализовать роли, поиск, карту, импорт/экспорт и аудит
- 7 провести функциональное тестирование

03

## Существующие справочники закрывают только часть сценариев

### СЗТК



#### Сильная сторона

Веб-интерфейс и иерархия организаций

#### Ограничение

есть ограничения по современным сценариям и интеграциям

### Phonebook App



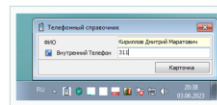
#### Сильная сторона

Быстрый поиск контактов

#### Ограничение

не является комплексной ведомственной ИС

### KDMSoft



#### Сильная сторона

Каталогизация и карточки

#### Ограничение

настольный формат и слабая ролевая модель

04

## Разработанная система объединяет функции, которых не хватает аналогам

| Критерий                           | СЗТК     | Phonebook App | KDMSoft | ЕСК  |
|------------------------------------|----------|---------------|---------|------|
| Веб-доступ                         | есть     | есть          | нет     | есть |
| Роли ADMIN / MANAGER / EMPLOYEE    | частично | ограничено    | нет     | есть |
| Поиск по ФИО, телефону и транслиту | частично | есть          | есть    | есть |
| Оргструктура и фильтры             | частично | нет           | нет     | есть |
| Интерактивная карта                | нет      | нет           | нет     | есть |
| Импорт, экспорт, аудит, API-ключи  | частично | нет           | нет     | есть |

05

## Функционал разделен по ролям пользователя

Система поддерживает повседневный поиск контактов и отдельный контур сопровождения данных.

### Сотрудник

- поиск сотрудников
- просмотр карточек
- оргструктура
- карта организаций

### Руководитель

- редактирование данных подразделения
- экспорт справочника
- контроль структуры

### Администратор

- пользователи и роли
- импорт/экспорт
- аудит
- API-ключи

06

## Неавторизованный пользователь получает только безопасный вход

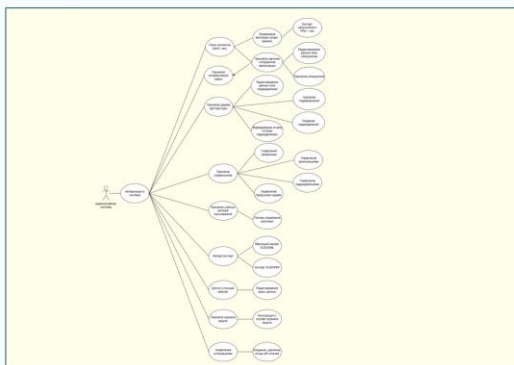


### Что показывает диаграмма

- доступна только авторизация
- закрытые разделы требуют входа
- попытка прямого перехода перенаправляется на страницу входа

07

## Администратор сопровождает данные, роли, аудит и интеграции



### Основные группы действий

- полное управление сотрудниками
- организации, подразделения, приемные
- импорт и экспорт XLSX/PDF
- журнал аудита и API-ключи
- разграничение доступа по ролям

08

## Стек подобран под SPA, REST API, безопасность и работу с данными

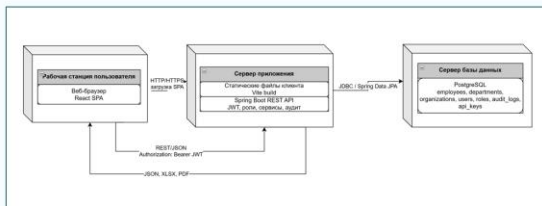


Такой набор технологий позволяет разделить интерфейс, бизнес-логику, хранение данных, авторизацию и обмен файлами.

09

## Клиент, сервер и БД разделены через REST/JWT

Пользователь работает через браузер, сервер Spring Boot обрабатывает бизнес-логику, PostgreSQL хранит справочные данные.

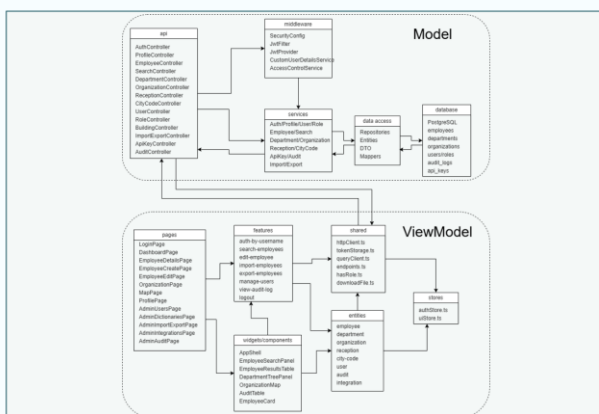


### Ключевые связи

- React SPA загружается через HTTP/HTTPS
- REST/JSON используется для API
- JWT передается в Authorization Bearer
- JDBC / Spring Data JPA связывает сервер с БД
- XLSX и PDF используются для обмена файлами

10

## Диаграмма компонентов связывает frontend, API, сервисы и БД

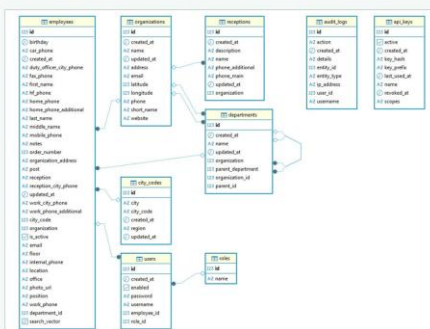


### Как читать схему

- api: REST-контроллеры
- middleware: Spring Security и JWT
- services: бизнес-операции
- data access: репозитории, DTO, mapper
- frontend: pages, features, shared, stores

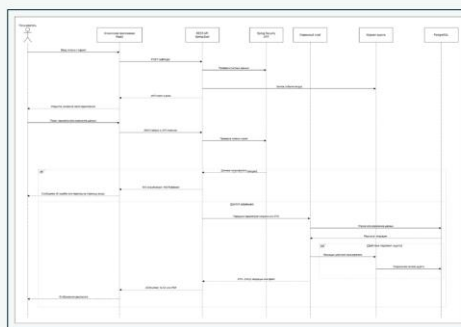
11

## Структура БД поддерживает контакты, роли, аудит и интеграции



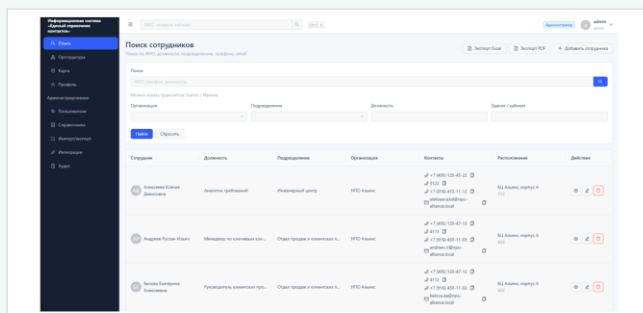
Графическая структура базы данных

Основные сущности employees, departments, organizations, users, roles, audit\_logs, api\_keys



12

## Главный сценарий - быстрый поиск сотрудников

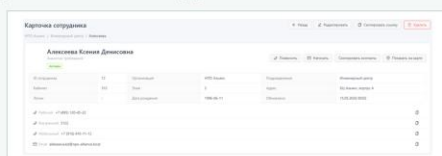


### Что доступно

- поиск по ФИО, должности, подразделению, телефону и email
- фильтры по организации, подразделению, должности и кабинету
- транслитерация запроса
- экспорт результатов в XLSX и PDF

13

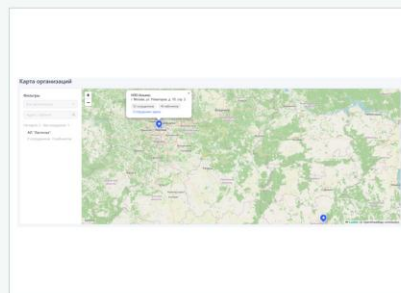
## Карточка, оргструктура и карта превращают справочник в рабочий инструмент



Карточка сотрудника: контакты и быстрые действия



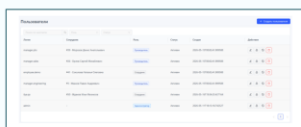
Организационная структура и сотрудники подразделения



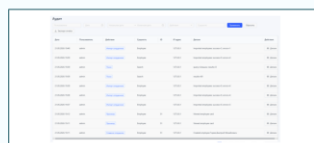
Карта организаций с фильтрами и метками

14

## Администратор сопровождает данные и контролирует изменения



Пользователи и роли



Журнал аудита



Импорт и экспорт



API-ключи и интеграции

### Зачем это нужно

- разграничение доступа
- проверка действий
- массовое обновление справочника
- подготовка выгрузок
- внешние интеграции

15

## Проверены основные пользовательские и административные сценарии

- |   |                                       |   |                                     |
|---|---------------------------------------|---|-------------------------------------|
| 1 | авторизация пользователя              | 4 | профиль и организационная структура |
| 2 | поиск и карточка сотрудника           | 5 | импорт и экспорт данных             |
| 3 | создание и редактирование сотрудников | 6 | администрирование и аудит           |

**Результат** Информационная система спроектирована, реализована и протестирована. Проверки подтвердили соответствие приложения заявленным требованиям.

16

### Заключение

## Спасибо за внимание

Готов ответить на ваши вопросы

### Единый справочник контактов

Демин Максим Сергеевич  
ПГУ, 2026

|      |      |          |         |      |
|------|------|----------|---------|------|
|      |      |          |         |      |
| Изм. | Лист | № докум. | Подпись | Дата |

ПГУ1.090301.08.001 ПЗ

Лист

162