

СОДЕРЖАНИЕ

Введение.....	6
1 Описание кадровой политики предприятия.....	8
1.1 Организационная структура предприятия.....	8
1.2 Цели и задачи отдела кадров на предприятии	9
1.3 Автоматизация кадровых процессов	9
2 Постановка задачи и обзор методов ее решения	13
2.1 Постановка задачи	13
2.2 Обзор методов решения задачи.....	14
2.3 Объектно-ориентированное программирование	14
2.4 Язык программирования Java.....	15
2.5 Библиотека JavaFX.....	16
2.6 База данных MySQL	17
2.7 Архитектурный шаблон MVC	17
2.8 Паттерны проектирования	19
3 Функциональное моделирование на основе стандарта IDEF0.....	21
3.1 Функциональное моделирование	21
3.2 Процесс деятельности HR-службы организации.....	21
4 Информационная модель системы и ее описание	27
5 Описание алгоритмов, реализующих бизнес-логику серверной части проектируемой системы	32
6 Модели представления системы и их описание	34
6.1 Диаграмма классов.....	34
6.2 Диаграмма компонентов	36
6.3 Диаграмма развертывания	37
6.4 Диаграмма состояний	38
6.5 Диаграмма последовательности	39
6.6 Диаграмма вариантов использования	41
7 Руководство пользователя	44
8 Результаты тестирования разработанной системы	59
Заключение	65
Список использованных источников	66
Приложение А (обязательное)	67
Приложение Б (обязательное).....	69
Приложение В (обязательное)	72

ВВЕДЕНИЕ

В последние годы наблюдается быстрое развитие компьютерных технологий. Компьютер внедряется практически во все сферы нашей жизни, а во многих из них становится просто незаменимым. Что касается предприятий сложно представить эффективное управление любой отраслью предприятия без использования компьютерных технологий.

В современных условиях бизнеса управление – ценный ресурс организации, наряду со многими другими, такими как финансовые, материальные и человеческие ресурсы. Поэтому улучшение управленческой деятельности становится одним из основных направлений совершенствования деятельности предприятия в целом. Именно поэтому важно внедрять информационные технологии.

В управлении используются автоматизированные информационные технологии, которые реализуются с помощью технических и программных средств.

Автоматизированные системы управления – это, пожалуй, именно то, в чем особо остро нуждаются современные предприятия. Автоматизация процесса способна значительно повысить производительность и эффективность работы организации.

В наше время трудно переоценить роль использования информационных технологий в управлении предприятием, а также в других областях общественной жизни. Наблюдается тенденция к существенным переменам во всех областях, где присутствуют человеческие ресурсы. Руководители компаний стараются максимально оптимизировать все процессы и увеличить получаемую прибыль.

Стоит отметить, что не что иное, как люди являются основой абсолютно любого предприятия. Именно человеческий фактор является причиной успеха или, наоборот, неудачи компании, поэтому грамотное управление персоналом с учетом индивидуальных потребностей людей и выстроенной системой мотивации позволяет достигнуть максимальных результатов [3].

Автоматизация управления кадрами – важная часть автоматизации управления предприятием. Чтобы обеспечить продуктивную работу HR отдела, необходимо внедрять системы по сбору и обработке данных.

До появления таких систем такие документы, как личные дела сотрудников, их трудовые книжки, договоры и т.д., велись в бумажном варианте. Это, во-первых, отнимало много времени у сотрудников, на заполнение различных отчетов уходила масса времени, а большое количество хранимой печатно информации хранилось в сложно организованных архивах. Так, к примеру, в Европе уже давно нет понятия трудовая книжка. Все необходимые для нанимателя данные хранятся в электронном виде и на карточках социального страхования. Во-вторых, ручное ведение всех документов и составление отчетов нередко могло привести к ошибкам из-за человеческого фактора (невнимательность, усталость и т.д.).

Основные функции автоматизированных систем по управлению кадровой политикой предприятия:

- учет количества действующих сотрудников;
- фиксирование операций по оплате труда;
- менеджмент трудовых ресурсов.

Также существует ряд параметров, которым должно соответствовать разрабатываемая система, если ее цель – оптимизация деятельности HR отдела предприятия. Некоторые из них:

- высокая степень защиты от доступа посторонних лиц;
- понятный для пользователя интерфейс;
- ранжирование прав в зависимости от должности и полномочий сотрудников.

Несмотря на то, что на сегодняшний момент существует большое количество продуктов от разных производителей, многие компании до сих пор не используют автоматизированные системы, а ведут учет кадров вручную, в бумажном варианте.

Причиной этому может стать как высокая стоимость разработки новой или настройки существующей программы, так и сложность использования.

В курсовой работе поставлена цель: проектирование приложения по управлению кадровой политикой предприятия с целью повышения эффективности работы HR отдела и оптимизации всех его процессов.

Объект исследования – процессы кадровой политики предприятий, работа HR отдела.

Предмет исследования – автоматизация процессов управления персоналом.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1 Изучение предметной области кадровой политики предприятия, управления персоналом и деятельности HR-службы.
- 2 Выделение требований к разрабатываемому программному средству.
- 3 Выбор средств проектирования и разработки.
- 4 Разработка графического интерфейса приложения.
- 5 Создание базы данных для хранения информации.
- 6 Разработка приложения.
- 7 Описание руководства пользования.
- 8 Тестирование приложения.

1 ОПИСАНИЕ КАДРОВОЙ ПОЛИТИКИ ПРЕДПРИЯТИЯ

1.1 Организационная структура предприятия

На эффективность деятельности организации влияют такие факторы, как:

1 Реальные взаимосвязи между людьми и их работой, которые отражаются в схемах организационных структур и должностных обязанностей.

2 Политика руководства и методы, воздействующие на поведение персонала.

3 Полномочия и функции работников организации на различных уровнях управления.

Рациональная структура организации предполагает комбинацию этих трех факторов, обеспечивающую высокий уровень эффективности производства.

Любое предприятие или организация обладают организационной структурой, разработанной на этапе их создания, но в процессе деятельности она может быть откорректирована. Организационная структура формируется в зависимости от целей деятельности предприятия и необходимых для этого подразделений.

Организационная структура компании или предприятия определяет взаимоотношения между руководящим составом и рабочим персоналом, служит основой распределения обязанностей между работниками предприятия, дает представление, кто на предприятии отвечает за принятие управленческих решений.

Организационная структура может быть представлена в виде схемы, отдельными блоками на которой будут выступать директор или руководитель предприятия, его структурные подразделения, отдельные управленческие единицы и связи между ними. Пример организационной структуры предприятия представлен на рисунке 1.1.

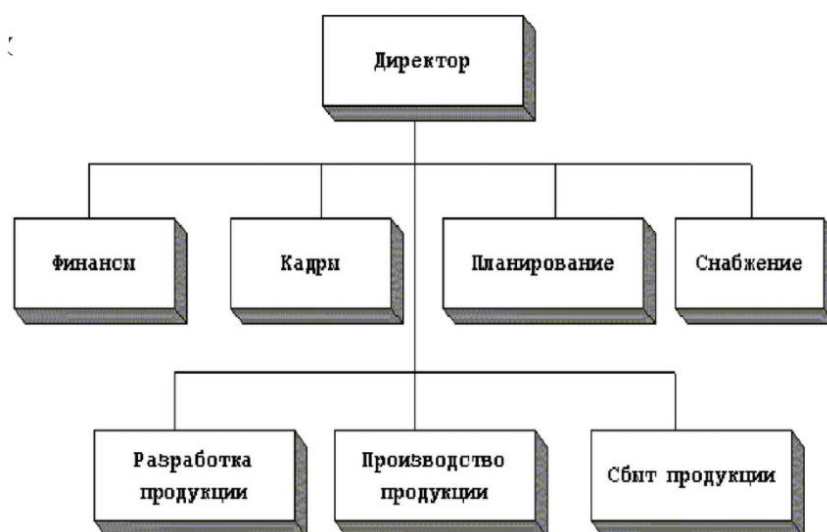


Рисунок 1.1 – Организационная структура предприятия

1.2 Цели и задачи отдела кадров на предприятии

Организационная структура абсолютно любого предприятия содержит отдел по управлению персоналом, или отдел кадров.

Отдел кадров – это структура в организации, которая занимается управлением персонала. Кроме того, отдел кадров – это не только функциональная единица, он является также и лицом компании, ведь именно с него любой соискатель начинает знакомиться с организацией.

Цель отдела кадров – это способствование достижению целей предприятия путем обеспечения предприятия необходимыми кадрами и эффективного использования потенциала работников.

Отдел кадров занимается учетом персонала, формированием отчетов по количеству уволенных/принятых сотрудников и т.д. Кроме того, именно отдел кадров должен подавать сведения о сотрудниках предприятия в различные государственные инстанции (пенсионный фонд, налоговую службу и др.)

Таким образом, задачами отдела кадров являются:

- отбор и найм персонала необходимой квалификации;
- создание эффективной системы штатных сотрудников;
- разработка карьерных планов сотрудников;
- разработка кадровых технологий.

К основным функциям отдела кадров на предприятии можно отнести:

- определение потребности организации в кадрах и подбор персонала;
- анализ текучести кадров;
- борьба с большой текучестью кадров;
- разработка и внедрение систем мотивации сотрудников;
- разработка штатного расписания предприятия;
- учет, оформление и хранение личных дел сотрудников.
- заполнение и хранение трудовых книжек сотрудников;
- выдача необходимых документов на руки сотрудникам;
- составление и учет графика отпусков сотрудников;
- организация аттестации сотрудников;
- составление планов повышения квалификации сотрудников.

Стоит отметить, что в зависимости от типа предприятия и его целей функции и задачи отдела кадров могут отличаться. Однако вышеперечисленные цели, задачи и функции отдела кадров являются основными и существуют вне зависимости от типа предприятия.

1.3 Автоматизация кадровых процессов

Для наиболее эффективной работы отдела кадров руководству предприятия требуется автоматизировать процессы. Можно выделить две основные причины, по которым автоматизация кадрового отдела обязательно должна проводиться на любом предприятии.

Первой причиной является повышение скорости работы отдела. Задержки при оформлении трудовых книжек, обработке заявлений или выдаче

справок сильно затягивают решение проблем, мешают продуктивной работе предприятия и ухудшают репутацию отдела кадров. Кроме того, иногда скорость работы отдела кадров не соответствует нагрузкам ввиду нехватки персонала или небольшого опыта, в связи с этим нерешенные проблемы накапливаются.

Второй причиной является уменьшение количества ошибок. При работе по старым принципам проверка всех документов и данных происходит вручную, поэтому ошибки, вызванные человеческим фактором, неизбежны.

Стоит отметить, что такие ошибки, как ошибки при начислении заработной платы или налоговых отчислений, могут привести к потере денежных средств предприятия или к штрафам со стороны государственных органов.

Преимущества автоматизации отдела кадров:

- ускорение рутинных процессов;
- упрощение процессов поиска сотрудников;
- учет личных дел всех сотрудников;
- быстрый учет необходимых для работы документов;
- упрощение документооборота отдела кадров;
- значительное уменьшение затрат времени;
- уменьшение количества ошибок.

Среди основных этапов автоматизации отдела кадров можно выделить следующие:

- подготовка персонала (обучение);
- выбор системы автоматизации;
- внедрение системы;
- тестирование системы;
- доработка по необходимости;
- начало использования системы;
- оценка эффективности автоматизации с последующим внесением доработок.

Существуют несколько типов автоматизации отдела кадров (см. рис 1.2). Выбор определенного типа происходит в зависимости от того, насколько крупным является предприятие, какие задачи планируются решить и степени интегрированности системы.

Первый тип – кастомные разработки. Такие продукты способны решать какие-то определенные задачи, например, такие как учет рабочего времени.

Второй тип – коробочные программные продукты. В отличие от предыдущих, они способны решать более широкий спектр задач.

Третий тип – HRM-системы (англ. Human Resource Management). Данные системы способны эффективно управлять бизнес-процессами, расходами и соблюдением правовых форм как одного отдела, так и всего предприятия в целом. Однако, внедрение таких систем в производство является дорогой услугой, а также требуют, во-первых, адаптации под конкретное предприятие, а во-вторых, специального обучения сотрудников. Существуют также

HRM-модули, которые используются для уже внедренных на предприятиях ERP-систем.

Четвертый тип – BPM-системы (англ. business process management). Они подразумевают комплексную автоматизацию всего предприятия и в частности отдела кадров.



Рисунок 1.2 – Инструменты автоматизации HR

Для всех этих систем характерны такие недостатки для небольших предприятий, как высока стоимость внедрения, долгий процесс внедрения и внесения изменений, сложность использования. Именно поэтому многие предприятия делают выбор в пользу low-code систем.

Такие системы не требуют серьезной подготовки к внедрению. Кроме того, есть возможность последовательного расширения и добавления функций с минимальным бюджетом. Внедрив такую систему, есть возможность охватить лишь часть ее процессов, протестировав качество ее работы, а затем, по мере потребностей отдела кадров и возможностей предприятия, система легко дорабатывается и расширяется.

Примером low-code системы может стать Comindware Business Application Platform. В основе платформы лежат управление процессами (BPMS), кейсами (ACM) и задачами. На платформе можно автоматизировать любые бизнес-задачи, включая все HR-процессы. Такое решение автоматизирует абсолютно все задачи отдела кадров.

Важным достоинством является работа в браузере, при которой нет необходимости устанавливать ПО на компьютер, а также доступна работа с помощью мобильных устройств.

Отдельно можно отметить, что предприятие может также использовать ЭДО, или электронный документооборот. Он также позволяет повысить эффективность организации. Электронный документооборот – это совокуп-

ность автоматизированных процессов по работе с документами, представленными в электронном виде без использования бумажных носителей. Но чтобы цифровой документ обладал той же юридической силой, что и бумажный, он должен быть подписан электронной подписью. Применяя систему ЭДО по отношению к отделу кадров, появится возможность вести архив всех необходимых документов (расписание, график отпусков, личные дела сотрудников и т.д.) в электронном виде.

KPI (Key Performance Indicator) – метод управления персоналом, при котором, опираясь на цели руководителя отдела и департамента выстраиваются задачи для выполнения. С помощью данного метода можно управлять производительностью труда, снижать потери фонда рабочего времени сотрудников компании, своевременно заполнять вакансии, снижать текучесть, выполнять бюджет затрат на персонал. К отрицательным сторонам метода можно отнести требуемую высокую управленческую культуру, а также хорошо продуманную систему показателей, чтобы не было улучшения одних показателей за счет других.

Таким образом, под каждое отдельное предприятие подбирается система, которая подходит данному предприятию исходя из целей и задач, выполнение которых необходимо для получения результата.

Однако, стоит отметить, что грамотно организованная, а также наиболее эффективная система по управлению персоналом является неотъемлемой частью работы HR-отдела предприятия.

2 ПОСТАНОВКА ЗАДАЧИ И ОБЗОР МЕТОДОВ ЕЕ РЕШЕНИЯ

2.1 Постановка задачи

В курсовой работе поставлена цель: проектирование приложения по управлению кадровой политикой предприятия с целью повышения эффективности работы HR отдела и оптимизации всех его процессов.

Для достижения поставленной цели необходимо разработать клиент-серверное приложение с доступом к базе данных для хранения информации. Приложение должно иметь понятный и легкий в освоении графический пользовательский интерфейс и должно содержать все необходимые функции для эффективного управления персоналом.

Благодаря клиент-серверной технологии разработки разрабатываемое приложение имеет возможность поддерживать подключение и работу нескольких клиентов, что необходимо для беспрепятственного доступа к приложению для всех сотрудников.

Предполагается, что программой могут пользоваться как сотрудники HR-отдела предприятия, так и управляющие отделом. При этом сотрудники HR-отдела будут авторизовываться в системе в качестве пользователей, а управляющие – в качестве администраторов. Соответственно, управляющие имеют больше прав, чем сотрудники.

Серверная часть представляет из себя консольное приложение, в котором помимо просмотра подключенных клиентов, осуществлена возможность загрузки базы данных.

Клиентское приложение, в отличие от серверного, представляет из себя оконное приложение с эргономичным и простым в использовании графическим интерфейсом. Как отмечалось ранее, в приложение доступно для пользователей с разными ролями, поэтому доступный функционал для каждого будет разный.

Первое отличие в доступном функционале для различных ролей заключается на этапе авторизации или регистрации. Так, регистрироваться в приложении может только администратору, а пользователю доступен лишь вход, так как его в качестве сотрудника регистрирует администратор.

После успешного входа в систему в качестве пользователя предоставляются две раздела: учет кадров и создание отчетов. При входе в качестве администратора также доступен третий раздел – настройки приложения.

При переходе к разделу «Учет кадров» функционал пользователя и администратора одинаков, а именно они могут:

- просматривать всех сотрудников предприятия и их личных дел;
- добавление новых сотрудников;
- обновление данных о сотрудниках;
- удаление сотрудников.

Кроме этого, для более удобного просмотра всех сотрудников осуществлена сортировка по фамилии или дате рождения сотрудников. Для

удобного поиска необходимого сотрудника также осуществлен поиск по фамилии или личному номеру сотрудника.

Второй раздел «Создание отчетов» позволяет пользователю разработанного программного средства создать визуальный отчет по количеству уволенных или принятых сотрудников за выбранный промежуток времени. Данный раздел помогает отделу кадров следить за текучестью кадров на предприятии.

Специальная функция администратора, а именно настройки приложения, позволяет сменить данные компании в текстовом документе, которые применяются при создании документов различного рода (трудовых договоров, трудовых книжек и т.д.).

2.2 Обзор методов решения задачи

Для достижения всех поставленных целей и решения задач разработка программного средства осуществляется на объектно-ориентированном языке программирования Java.

Как было отмечено ранее, клиентская часть приложения реализована в виде оконного приложения, графический пользовательский интерфейс которого реализован с помощью библиотеки JavaFX.

Для хранения всей необходимой информации была использована реляционная база данных с помощью MySQL. Для связи с базой данных использовалась технология JDBC.

2.3 Объектно-ориентированное программирование

Объектно-ориентированное программирование – это подход, при котором вся программа рассматривается как совокупность взаимодействующих друг с другом объектов.

Основные задачи объектно-ориентированного программирования:

- структурировать код и повысить его читабельность;
- ускорить понимание логики программы.

Косвенно выполняются и другие задачи, например, повышается безопасность кода и сокращается его дублирование.

Преимущества объектно-ориентированного программирования:

- код читается: не надо долго искать в коде функции и выяснять, за что они отвечают;
- меньше повторений в коде: нет необходимости писать однотипные функции для разных сущностей;
- сложные программы значительно упрощаются: большую программу можно логически разбить на несколько блоков, и постепенно наполнять их все больше и больше;
- увеличение скорости написания: сначала программист может создать прототип программы, то есть минимально наполнить ее компонентами, а после – постепенно дополнять для правильной работы программы.

Объектно-ориентированное программирование позволяет упростить сложные объекты, составляя их из более маленьких и простых, поэтому над одной программой могут работать десятки разработчиков. Каждый из них будет занят своим блоком.

Недостатки объектно-ориентированного программирования:

- объектно-ориентированное программирование значительно сложнее процедурного программирования, для начала требуется хорошо изучить теорию и только после этого начинать писать программы;

- объектно-ориентированное программирование требует много памяти, так как объекты состоят из методов, данных и т.д., все это занимает больше памяти, чем обычные переменные в процедурном программировании;

- производительность кода может быть ниже. Особенность объектно-ориентированного программирования такова, что некоторые вещи могут быть реализованы сложнее, чем в процедурном программировании, но стоит отметить, что с современными процессорами это практически не заметно.

Поэтому в небольших простых программах использование объектно-ориентированного программирования не будет оправдано. Однако в больших и сложных проектах использование объектно-ориентированное программирование необходимо, так как неразделенный на отдельные сущности код быстро перестанет читаться.

2.4 Язык программирования Java

Java—это язык программирования общего назначения, который следует парадигме объектно-ориентированного программирования и подходу «Написать один раз и использовать везде». Это означает, что написанное на Java приложение можно запустить на любой платформе, если на ней установлена среда исполнения Java, или JRE (англ. Java Runtime Environment).

Java Runtime Environment (JRE)—исполняющая система Java. Механизм распространения программного обеспечения, состоит из автономной виртуальной машины Java, стандартной библиотеки Java (Java Class Library) и инструментов настройки.

На самом деле, Java – это не только язык программирования, это также целая система различных инструментов, которая охватывает все, что необходимо при программировании на Java. Помимо отмеченного ранее JRE сюда также входят JDK, JVM и IDE.

Понятие JDK является одним из трех основных технологий, используемых в программировании на языке Java. К ним также относятся JVM и JRE.

Важно понимать, чем они отличаются и как связаны. Так, JVM (Java Virtual Machine) отвечает за исполнение Java-программ, JRE – создает и запускает JVM, а JDK позволяет разработчикам создавать программы, которые могут выполняться и запускаться посредством JVM и JRE. Таким образом, JDK представляет собой пакет инструментов для разработки программного обеспечения, тогда как JRE представляет собой пакет инструментов для запуска Java-кода.

IDE (Integrated Development Environment) – это интегрированная среда разработки. Инструменты, которые помогают запускать, редактировать и компилировать код. Самые популярные из них—IntelliJ IDEA, Eclipse и NetBeans. Для выполнения курсового проекта в качестве среды разработки была выбрана IntelliJ IDEA. (почему дописать)

Java — мультифункциональный объектно-ориентированный язык со строгой типизацией.

Строгая типизация не позволяет смешивать в выражениях разные типы и не выполняет автоматически неявные преобразования. Несмотря на неудобства для программиста (необходимость вручную прописывать преобразования), благодаря этому достигается высокая надежность.

Многофункциональность означает, что на языке программирования Java можно разрабатывать абсолютно разные виды приложений: десктопные, под Android, игры и т.д.

Итак, на языке программирования Java можно написать следующие приложения:

- приложения под Android;
- десктопные приложения;
- веб-приложения;
- программы для работы с Big Data;
- корпоративный софт;
- банковские программы и др.

Чаще всего Java используется в веб-разработке и в разработке приложений для Android.

Плюсы Java:

- независимость – код будет работать на любой платформе, поддерживающей Java;
- надежность достигается благодаря строгой типизации;
- мультифункциональность.

2.5 Библиотека JavaFX

Графический пользовательский интерфейс, или GUI (англ. Graphical user interface) – это разновидность пользовательского интерфейса, в котором все элементы, такие как кнопки, меню, поля для ввода и др., представлены пользователю на дисплее в виде графических элементов или картинок.

Библиотека JavaFX используется для создания игр и настольных приложений на Java. JavaFX предоставляет разработчикам большой набор частей графического интерфейса, таких как кнопки, текстовые поля, меню, диаграммы и многое другое.

JavaFX также предоставляет FXML – декларативный язык разметки XML, который значительно упрощает работу с графическими компонентами в приложении.

Для создания графического интерфейса для разрабатываемого программного обеспечения используется Scene Builder – автономное приложе-

ние, которое генерирует разметку FXML. Scene Builder значительно упрощает создание сложных интерфейсов и работу с их дизайном. Кроме того, стилизовать приложение, создаваемое с помощью JavaFX можно с помощью CSS.

2.6 База данных MySQL

MySQL – это система управления реляционными базами данных (СУРБД). Реляционная база данных означает, что данные в ней хранятся в виде таблиц.

Система имеет открытый исходный код, что позволяет свободно использовать его и изменять или дополнять его под свои потребности. Кроме того, система совместима со многими платформами, такими как MacOS, Windows, Linux. Система MySQL хорошо масштабируется и считается достаточно гибкой и простой в использовании.

Для работы с базой данных необходимо знать SQL — это язык структурированных запросов, который позволяет взаимодействовать между вами и базой данных.

2.7 Архитектурный шаблон MVC

Шаблон проектирования Модель – Представление – Контроллер (MVC) – это шаблон программной архитектуры, построенный на основе сохранения представления данных отдельно от методов, которые взаимодействуют с данными [5].

Несмотря на то, что схема MVC была первоначально разработана для персональных компьютеров, она была адаптирована и широко используется веб-разработчиками из-за точного разграничения задач и возможности повторного использования кода. Схема стимулирует развитие модульных систем, что позволяет разработчикам быстро обновлять, добавлять или удалять функционал.

Название шаблона проектирования определяется тремя его основными составляющими частями: Модель, Представление и Контроллер. Визуальное представление шаблона MVC выглядит, как показано на приведенной ниже диаграмме (см. рис. 2.1).

На рисунке показана структура одностороннего потока данных и пути его следования между различными компонентами, а также их взаимодействие.

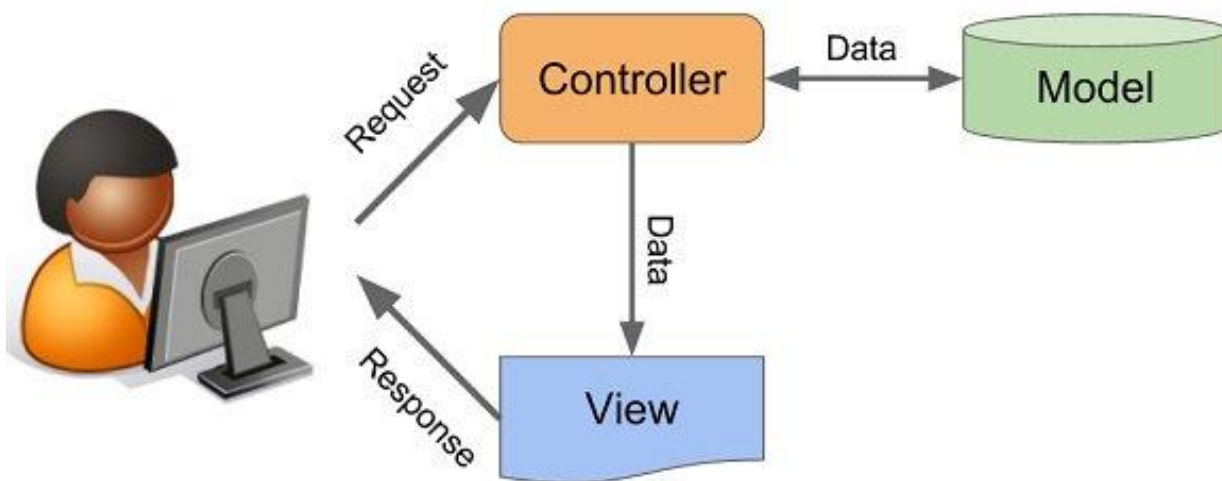


Рисунок 2.1 – Схема шаблона MVC

Моделью называют постоянное хранилище данных, используемых во всей структуре. Она должна обеспечивать доступ к данным для их просмотра, отбора или записи. В общей структуре Модель является мостом между компонентами Представление и Контроллер.

При этом Модель не имеет никакой связи или информации о том, что происходит с данными, когда они передаются компонентам Представление или Контроллер. Единственная задача Модели – обработка данных в постоянном хранилище, поиск и подготовка данных, передаваемых другим составляющим MVC.

Модель должна выступать в качестве «привратника», стоящего возле хранилища данных и не задающего вопросов, но принимающего все поступающие запросы. Зачастую это наиболее сложная часть системы MVC. Компонент Модель – это вершина всей структуры, так как без нее невозможна связь между Контроллером и Представлением.

Представление – это часть системы, в которой данным, запрашиваемым у Модели, задается окончательный вид их вывода. В веб-приложениях, созданных на основе MVC, Представление – это компонент, в котором генерируется и отображается HTML-код.

Представление также перехватывает действие пользователя, которое затем передается Контроллеру. Характерным примером этого является кнопка, генерируемая Представлением. Когда пользователь нажимает ее, запускается действие в Контроллере.

Существует несколько распространенных заблуждений относительно компонента Представление. Например, многие ошибочно полагают, что Представление не имеет никакой связи с Моделью, а все отображаемые данные передаются от Контроллера. В действительности такая схема потока данных не учитывает теорию, лежащую в основе MVC архитектуры.

Кроме этого определение Представления как файла шаблона также является неточным. Но это не вина одного человека, а результат множества ошибок различных разработчиков, которые приводят общему заблуждению.

После чего они неправильно объясняют это другим. На самом деле Представление – это намного больше, чем просто шаблон. Но современные MVC-ориентированные фреймворки до такой степени впитали этот подход, что никто уже не заботится о том, поддерживается ли верная структура MVC или нет.

Компоненту Представление никогда не передаются данные непосредственно Контроллером. Между Представлением и Контроллером нет прямой связи – они соединяются с помощью Модели.

Его задача заключается в обработке данных, которые пользователь вводит и обновлении Модели. Это единственная часть схемы, для которой необходимо взаимодействие пользователя.

Контроллер можно определить, как сборщик информации, которая затем передается в Модель с последующей организацией для хранения. Он не содержит никакой другой логики, кроме необходимости собрать входящие данные. Контроллер также подключается только к одному Представлению и одной Модели. Это создает систему с односторонним потоком данных с одним входом и одним выходом в точках обмена данными.

Контроллер получает задачи на выполнение только когда пользователь взаимодействует с Представлением, и каждая функция зависит от взаимодействия пользователя с Представлением. Наиболее распространенная ошибка разработчиков заключается в том, что они путают Контроллер со шлюзом, поэтому присваивают ему функции и задачи, которые относятся к Представлению.

Также распространенной ошибкой является наделение Контроллера функциями, которые отвечают только за обработку и передачу данных из Модели в Представление. Но согласно структуре MVC паттерна, это взаимодействие должно осуществляться между Моделью и Представлением.

2.8 Паттерны проектирования

В разработке программного обеспечения паттерн проектирования программного обеспечения - это общее, многократно используемое решение часто встречающейся проблемы в данном контексте при разработке программного обеспечения. Это не законченный проект, который может быть преобразован непосредственно в исходный или машинный код. Скорее, это описание или шаблон для решения проблемы, которые можно использовать в самых разных ситуациях. Паттерны проектирования - это формализованные передовые практики, которые программист может использовать для решения общих проблем при разработке приложения или системы.

Паттерны объектно-ориентированного проектирования обычно показывают отношения и взаимодействия между классами или объектами без указания конечных классов приложений или объектов, которые в них участвуют. Паттерны, которые подразумевают изменяемое состояние, могут не подходить для функциональных языков программирования, некоторые паттерны могут быть лишними в языках, которые имеют встроенную поддержку

для решения проблемы, которую они пытаются решить, и объектно-ориентированные паттерны не обязательно подходят для не объектно-ориентированных языков.

Паттерны могут ускорить процесс разработки, предоставляя проверенные, одобренные парадигмы разработки.

Паттерны проектирования изначально были сгруппированы по следующим категориям:

- порождающие паттерны (creational patterns);
- структурные паттерны (structural patterns);
- поведенческие паттерны (behavioral patterns).

Порождающие паттерны предназначены для создания экземпляров классов или объектов.

Структурные паттерны – для организации различных классов и объектов для формирования более крупных структур и обеспечения новой функциональности.

Поведенческие паттерны – для эффективной коммуникации между объектами.

Строитель (англ. Builder) – порождающий паттерн проектирования. Он отделяет конструкцию сложного объекта от его представления, что позволяет одному и тому же процессу конструирования создавать различные представления. Шаблон позволяет создать различные виды объекта, избежав засорения конструктора. Он может быть полезен, когда может быть несколько видов объекта или когда необходимо множество шагов для его создания.

Одиночка (англ. Singleton) – порождающий паттерн проектирования. Убеждается, что у класса есть только один экземпляр, и предоставляет глобальную точку доступа к нему.

Стратегия (англ. Strategy) – поведенческий паттерн проектирования. Он определяет набор схожих алгоритмов, инкапсулирует каждый из них в собственный класс и обеспечивает их взаимозаменяемость.

3 ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ НА ОСНОВЕ СТАНДАРТА IDEF0

3.1 Функциональное моделирование

Под функциональным моделированием понимается процесс построения функциональных моделей объекта автоматизации, либо отдельных его процессов.

Функциональное моделирование рассматривает бизнес как функцию или, другими словами, «черный ящик». Стоит отметить, что в функциональной модели функция не имеет так называемой временной последовательности, а только точку входа и точку выхода. Функциональное моделирование помогает рассматривать бизнес-модель с точки зрения результативности, поэтому при моделировании необходимо исходить из того, что мы имеем на входе, и того, что хотим получить на выходе.

Таким образом, в функциональной модели изначально известны точка входа и желаемый результат, а последовательность действий и является объектом разработки. При этом использование функциональных моделей как «черных ящиков» позволяет детализировать каждый этап по мере необходимости. А вся работа при моделировании направлена на поиск оптимального решения для достижения цели.

IDEF0 – Function Modeling – это методология функционального моделирования, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является её акцент на соподчинённость объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность. Нотация IDEF0 легка в использовании и построении.

Функциональная модель IDEF0 представляет собой набор блоков, каждый из которых представляет собой «черный ящик» со входами и выходами, управлением и механизмами, которые детализируются (декомпозируются) до необходимого уровня. Наиболее важная функция расположена в верхнем левом углу. А соединяются функции между собой при помощи стрелок и описаний функциональных блоков.

3.2 Процесс деятельности HR-службы организации

Кадровая политика предприятия и деятельность HR-службы организации направлены на управление персоналом предприятия. Рассмотрим процесс деятельности HR-службы организации (см. рис 3.1).

При этом входными данными будут служить соискатели, а выходными – сотрудники предприятия, их данные и отчеты. Механизмами служат HR-менеджеры, выполняющие свои обязанности, система управления персоналом и база данных сотрудников и соискателей. Управление служат как законы РБ, так и внутренние стандарты и правила предприятия.

Далее декомпозируем контекстную модель, представленную на рисунке 3.1.



Рисунок 3.1 – Контекстная диаграмма

Деятельность HR-службы включает в себя следующие этапы (см. рис. 3.2):

- обновление кадров;
- рассмотрение резюме соискателей;
- интервьюирование кандидата;
- прием на работу;
- ведение статистики и составление отчетов.

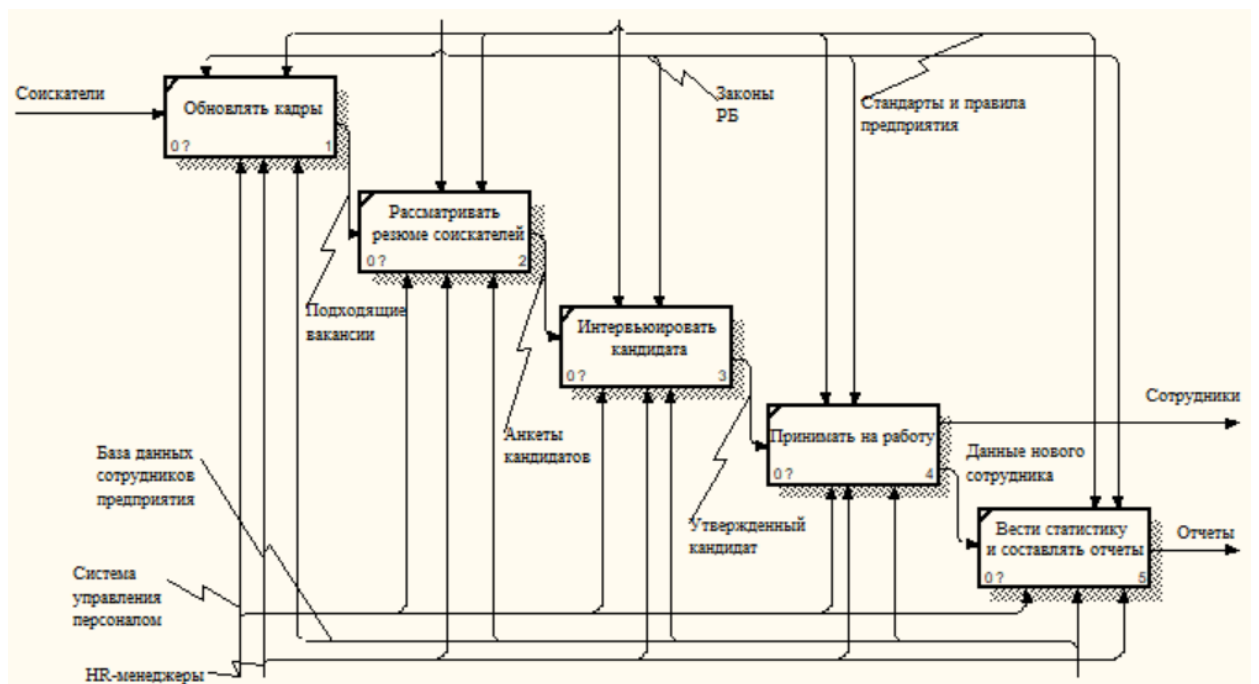


Рисунок 3.2 – Декомпозиция контекстной модели

Далее декомпозируем процесс обновления кадров:

- обработка актуальных данных о вакансиях на предприятии;
- изменение данных о вакансиях на предприятии;
- выбор подходящей для соискателя вакансии.

В результате данного этапа на выходе мы получаем список подходящих вакансий (см. рис. 3.3).

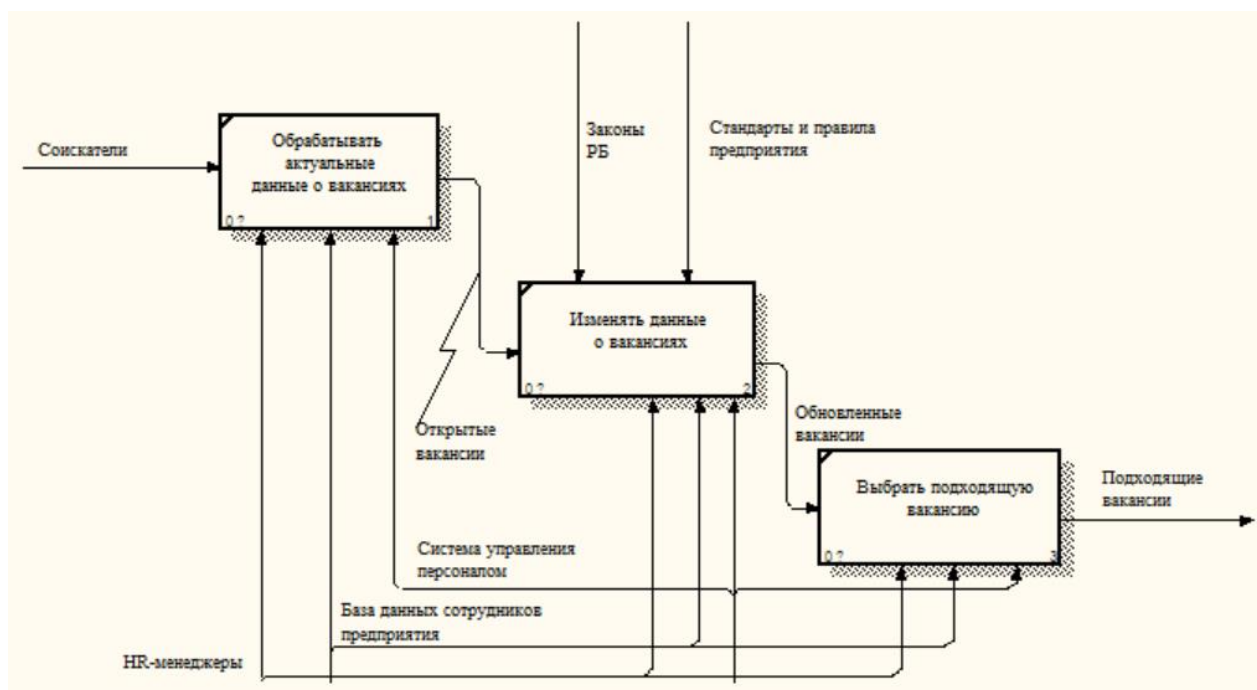


Рисунок 3.3 – Декомпозиция блока «Обновлять кадры»

Декомпозиция процесса рассмотрения резюме соискателей:

- осуществляется поиск соискателей, соответствующих вакансиям;
- производится рассмотрение резюме подходящих кандидатов;
- необходимая информация из резюме переносится в систему.

В результате данного этапа на выходе мы получаем анкеты кандидатов, зарегистрированных в системе и находящихся в базе данных кандидатов (см. рис. 3.4).

Далее декомпозируем процесс интервьюирования кандидата (см. рис. 3.5):

- произведение тестирования кандидата;
- собеседование с кандидатом;
- утверждение кандидата.

Декомпозиция процесса тестирования кандидата (см. рис. 3.6):

- анализ заявленных качеств кандидата;
- составление теста на проверку качеств;
- выдача теста кандидату на заполнение.

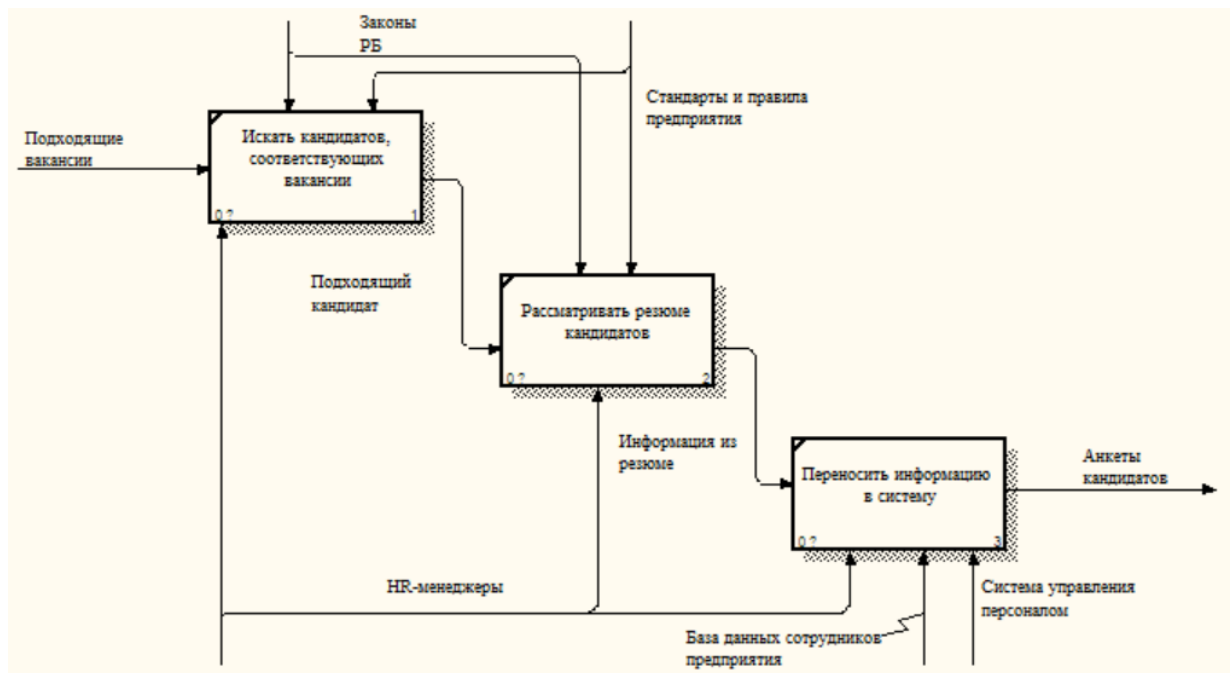


Рисунок 3.4 – Декомпозиция блока «Рассматривать резюме соискателей»

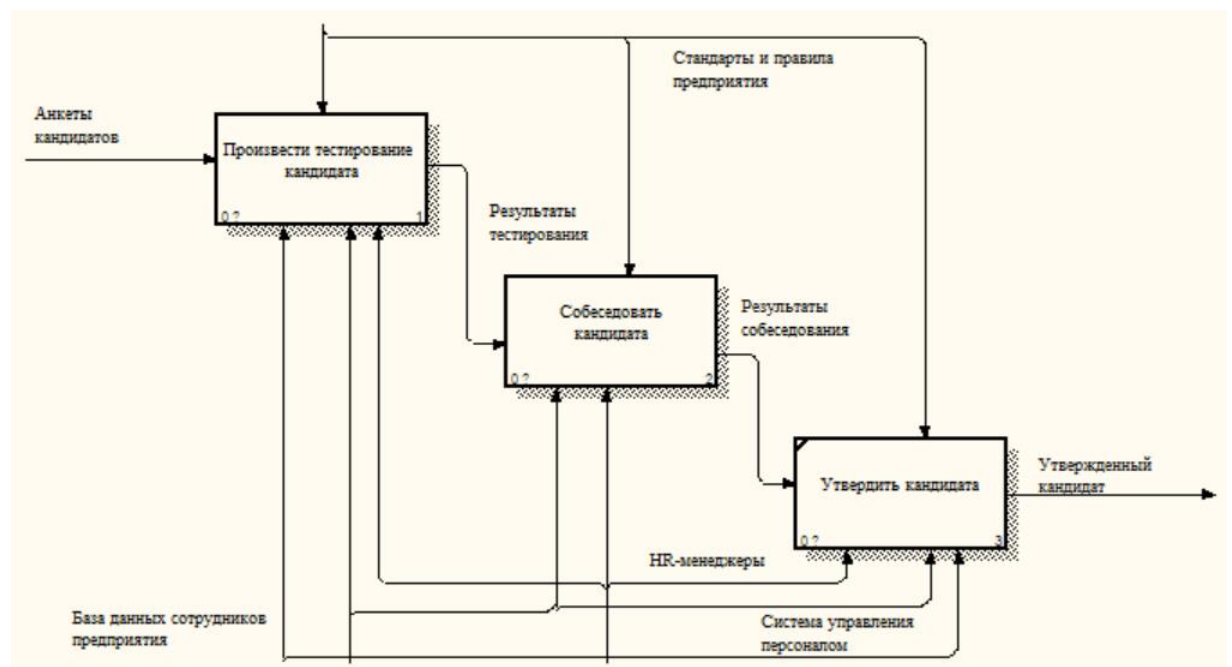


Рисунок 3.5 – Декомпозиция блока «Интервьюировать кандидата»

С полученными результатами тестирования переходим к следующему процессу, а именно, собеседованию с кандидатом (см. рис. 3.7).

Декомпозиция собеседования с кандидатом:

- составление вопросов для собеседования;
- опрос кандидата по составленным вопросам;
- психологический тест.

После успешного завершения собеседования кандидат утверждается. Этот этап является финальным.

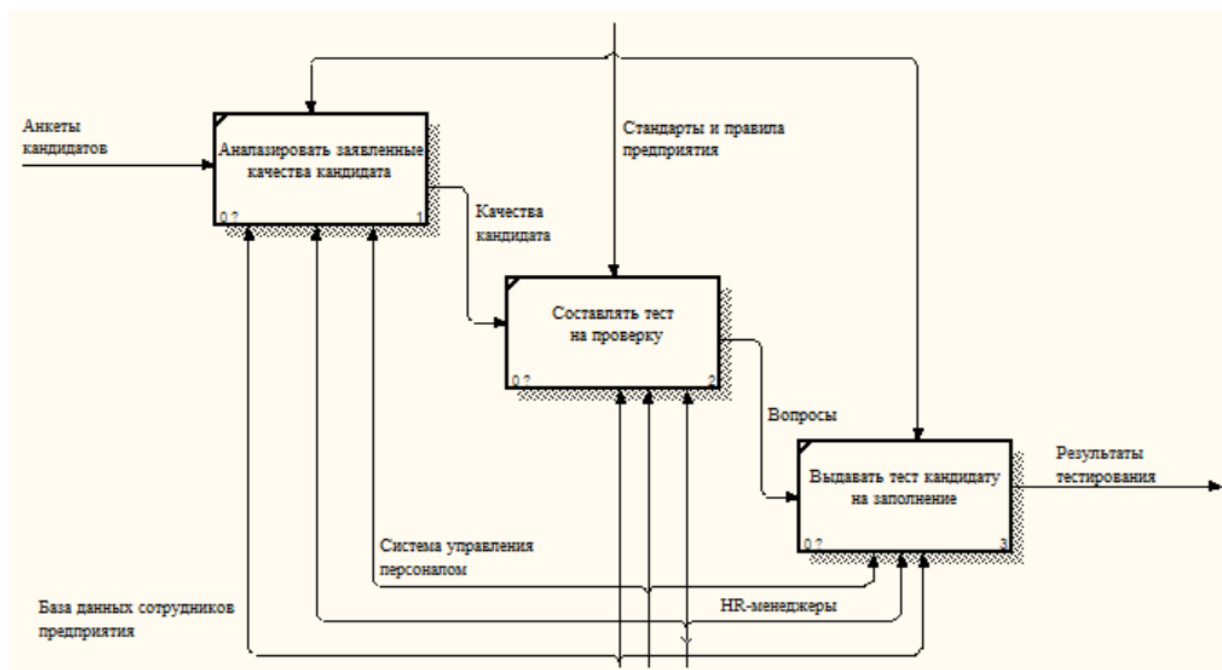


Рисунок 3.6 – Декомпозиция блока «Провести тестирование кандидата»

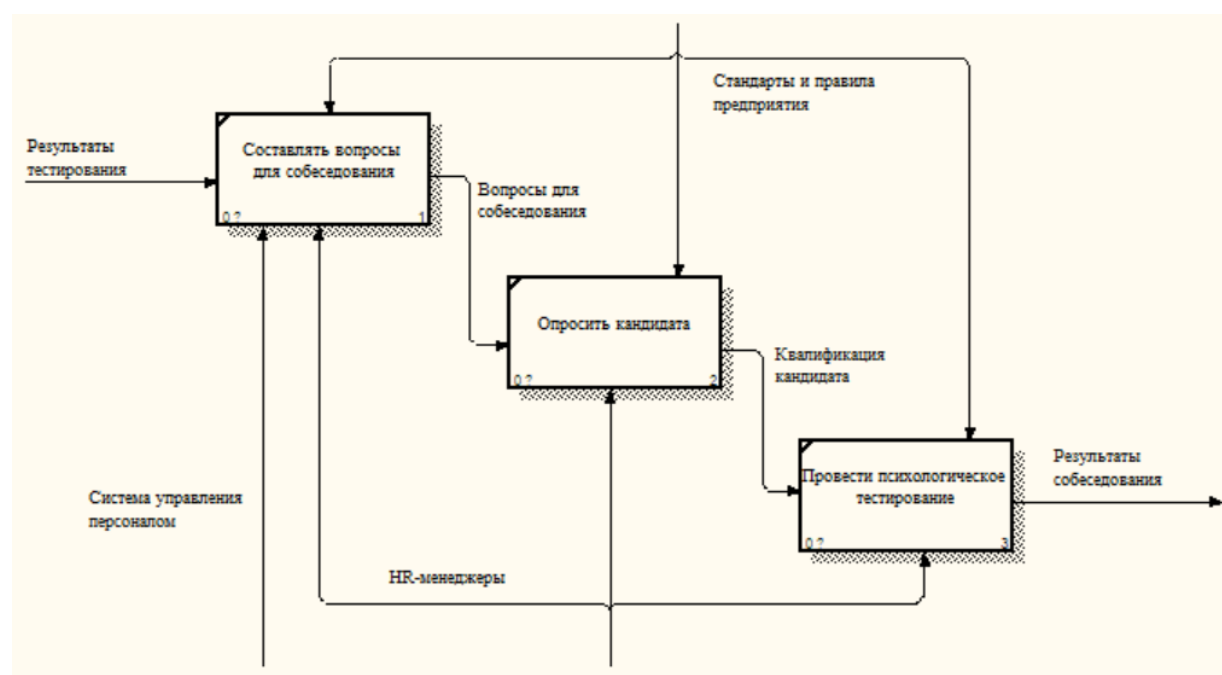


Рисунок 3.7 – Декомпозиция блока «Собеседовать кандидата»

Следующий процесс – прием на работу (см. рис. 3.8).

Декомпозиция приема на работу:

- занесение в базу данных анкеты нового сотрудника;
- заключение трудового договора с сотрудником.

В результате данного этапа на выходе мы получаем данные нового сотрудника.

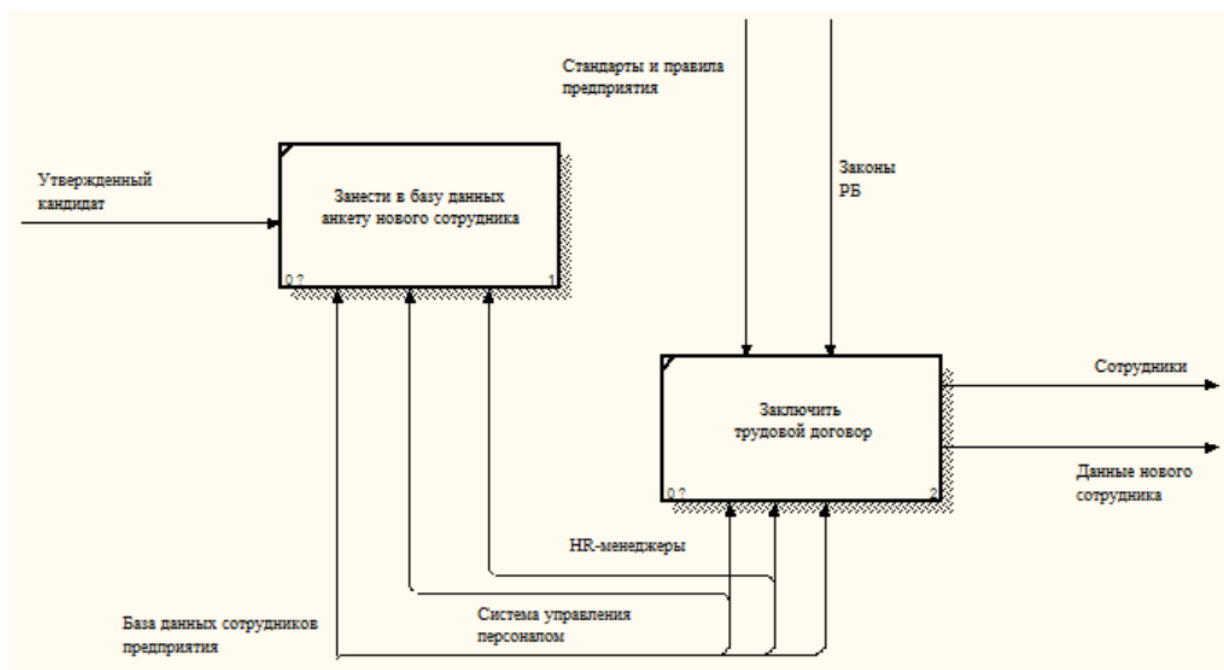


Рисунок 3.8 – Декомпозиция блока «Принимать на работу»

Последний рассматриваемый процесс – ведение статистики и составление отчетов (см. рис. 3.9) включает в себя:

- поиск и сортировка необходимых данных;
- построение отчетов.

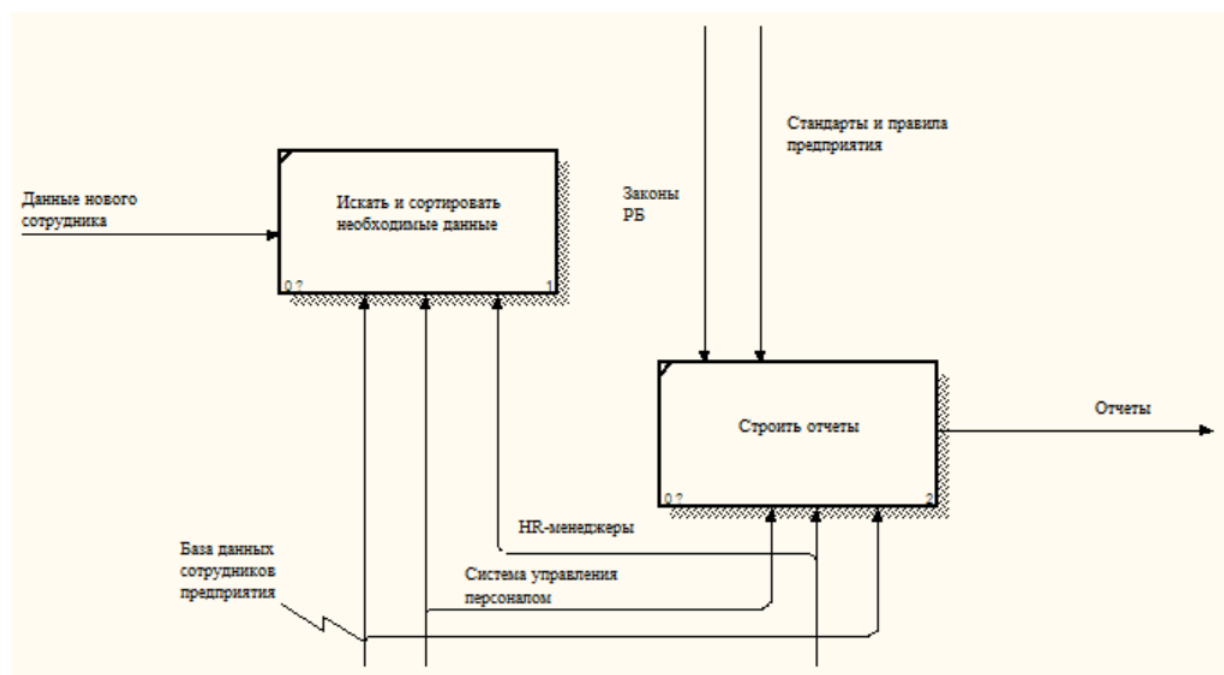


Рисунок 3.9 – Декомпозиция блока «Вести статистику и составлять отчеты»

4 ИНФОРМАЦИОННАЯ МОДЕЛЬ СИСТЕМЫ И ЕЕ ОПИСАНИЕ

Одним из ключевых моментов создания ИС с целью автоматизации информационных процессов организации является всестороннее изучение объектов автоматизации, их свойств, взаимоотношений между этими объектами и представление полученной информации в виде информационной модели данных.

Информационная модель объекта – модель, описывающая объект, применяя индивидуальную информацию, состоящую из различных взаимосвязанных параметров и переменных. Информационные модели – это неосознаваемые и невидимые процессы, потому что состоят из информации. Информация в модели, описывающая объект, затрагивает:

- состояние объекта;
- свойства объекта;
- процессы, связанные с объектом;
- связь с другими объектами и внешней средой и др.

В широком смысле, информационная модель объекта представляет собой совокупность информации из разных сфер, но объединенной описанием объекта. Примером информационной модели может служить карта местности, математическая формула или чертеж здания.

Информационные модели делятся на описательные и формальные. Описательные информационные модели – это модели, созданные на естественном языке, в устной или письменной форме. Формальные информационные модели – это модели, созданные на формальном языке. Примером формальной информационной модели могут служить формулы, таблицы, графы и т.д.

Основные компоненты информационной модели:

- сущности;
- связи;
- атрибуты.

Сущность (entity) – это объект, идентифицированный определенным способом, который его отличает от других объектов. Примером сущности может быть предприятие, событие или человек.

Множество сущностей, обладающих одинаковыми свойствами, называются набором сущностей (англ. entity set). Примером набора сущностей могут служить все предприятия, праздники или люди.

Несколько сущностей могут быть связаны друг с другом. Связь – это ассоциация, установленная между несколькими сущностями.

Сущность представляет из себя множество атрибутов, описывающих свойства всех членов данного набора сущностей. Существуют ключевые и не ключевые атрибуты.

Ключ сущности – один или группа атрибутов, однозначно определяющих экземпляр сущности.

Диаграмма, иллюстрирующая сущности и связи между ними согласно простым, интуитивно понятным правилам (аннотациям), называется ER-диаграммой (от англ. Entity-Relationship, сущность-связь). Это семантическая модель данных, предназначенная для упрощения процесса проектирования баз данных [7].

Для создания ER-модели необходимо выделить сущности данной предметной области, которые образуют структуру проектируемой информационной системы:

- Работник;
- HR-менеджер;
- Отдел;
- Должность;
- Гражданство;
- Семейное положение.

Данная модель данных, состоящая из 6 таблиц, представлена на рисунке 4.1.

В таблице «Работник» хранится информация о всех сотрудниках предприятия. Атрибутами данной сущности являются:

- уникальный номер сотрудника;
- фамилия сотрудника;
- имя сотрудника;
- отчество сотрудника;
- код отдела;
- код должности;
- код гражданства;
- код семейного положения;
- дата приема на работу;
- дата увольнения;
- номер страхования.

Атрибуты код должности, код гражданства, код семейного положения, код подразделения используются для связи по ключу с соответствующими им таблицами.

В таблице «HR-менеджер» хранится информация о всех сотрудниках отдела кадров на предприятии. Атрибутами данной сущности являются:

- уникальный номер сотрудника;
- роль;
- имя пользователя;
- пароль.

Атрибут уникальный номер используется для связи с таблицей «Пользователь». Атрибуты роль, имя пользователя и пароль используются для авторизации пользователя в приложении. Роль определяет функционал пользователя системы. Как было отмечено ранее, функционал различается в зависимости от роли: управляющий отделом или HR-менеджер.

В таблице «Отдел» хранится информация о всех отделах на предприятии:

- код отдела;
- название отдела.

В таблице «Должность» хранится информация о должностях предприятия:

- код должности;
- описание должности.

В таблице «Гражданство» хранится информация о гражданстве сотрудника предприятия:

- код гражданства;
- гражданство.

В таблице «Семейное положение» хранится информация о семейном положении сотрудников предприятия:

- код семейного положения;
- семейное положение;
- дети.

После того, как были определены таблицы базы данных, необходима нормализация таблиц. Здесь осуществляется приведение каждой таблицы к 3НФ. В результате нормализации получается очень гибкий проект базы данных, позволяющий легко вносить в нее нужные расширения.

Для начала стоит привести определение некоторых понятий.

Нормальная форма (НФ) представляет собой ограничение на схему базы данных, вводимое с целью устранения определенных нежелательных свойств при выполнении реляционных операций [7].

Для приведения таблиц в базе данных к первой нормальной форме (1НФ) необходимо, чтобы значение в каждом поле было неделимым (атомарным) [7].

Для приведения таблиц в базе данных ко второй нормальной форме (2НФ) в таблицах, приведенных к 1НФ, необходимо устранить зависимость неключевых полей от части составного ключа [7].

Для приведения таблиц в базе данных к третьей нормальной форме (3НФ) в таблицах, приведенных ко 2НФ, необходимо устранить зависимость неключевых полей от других неключевых полей. Если упростить данное правило, то можно сформулировать его так: необходимо выносить все неключевые поля, содержимое которых может относиться к нескольким записям таблицы, в отдельные таблицы [7].

Разработанная база данных приведена к третьей нормальной форме, так как у каждой таблицы есть всего один первичный ключ, и каждое неключевое поле не транзитивно зависит от первичного ключа, что означает, если мы изменим значение в одном столбце, то изменение в другом столбце не потребуется.

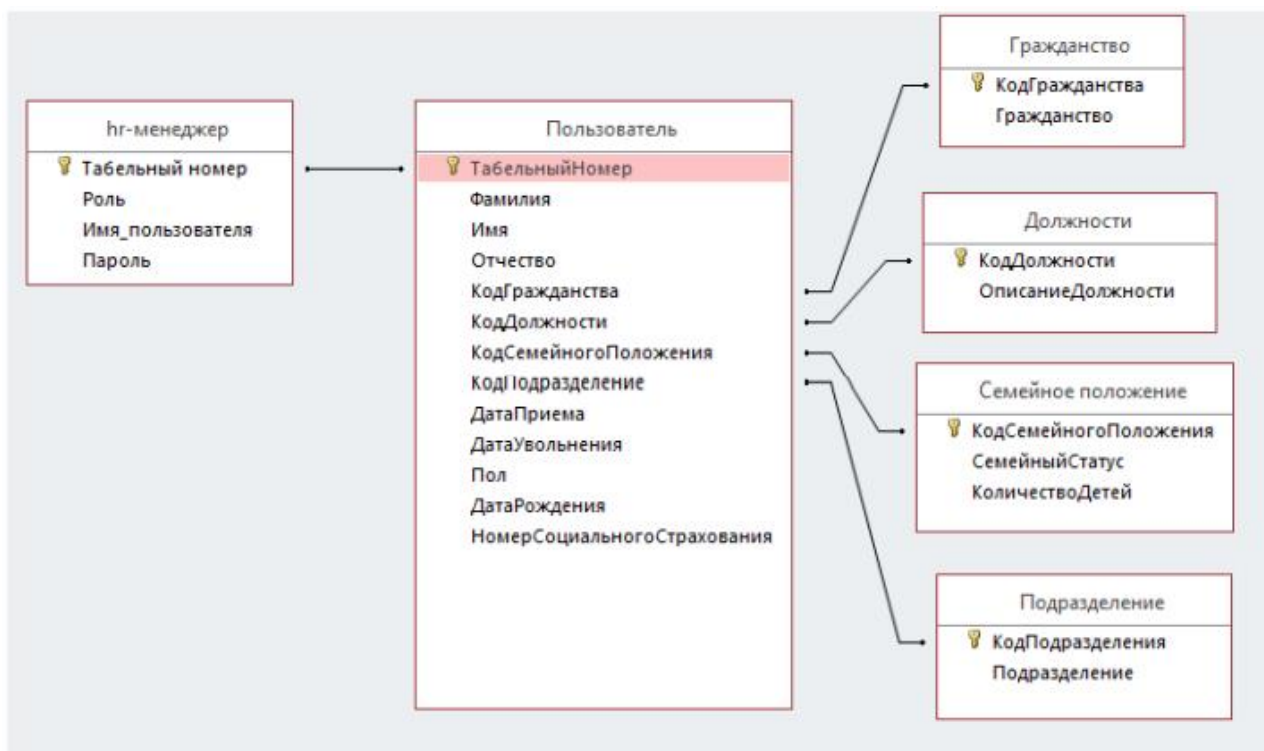


Рисунок 4.1 – База данных системы управления персоналом

Описание всех таблиц базы данных приведено ниже в таблицах 4.1, 4.2, 4.3, 4.4, 4.5, 4.6.

Таблица 4.1 – Описание сущности «HR-менеджер»

Название атрибута	Заполнение	Описание
Табельный номер	INT	Уникальный идентификатор отдела
Роль	VARCHAR(45)	Используется для определения прав пользователя.
Имя пользователя	VARCHAR(45)	Используется для входа в систему
Пароль	VARCHAR(45)	Используется для входа в систему

Таблица 4.2 – Описание сущности «Семейное положение»

Название атрибута	Заполнение	Описание
Код семейного положения	INT	Уникальный идентификатор семейного положения
Семейное положение	VARCHAR(45)	Описание семейного положения
Дети	INT	Количество детей

Таблица 4.3 – Описание сущности «Отдел»

Название атрибута	Заполнение	Описание
Код отдела	INT	Уникальный идентификатор отдела
Подразделение	VARCHAR(45)	Описание подразделения

Таблица 4.4 – Описание сущности «Должность»

Название атрибута	Заполнение	Описание
Код должности	INT	Уникальный идентификатор должности
Должность	VARCHAR(45)	Описание должности

Таблица 4.5 – Описание сущности «Гражданство»

Название атрибута	Заполнение	Описание
Код гражданства	INT	Уникальный идентификатор гражданства
Гражданство	VARCHAR(45)	Описание гражданства

Таблица 4.6 – Описание сущности «Работник»

Название атрибута	Тип данных	Описание
Уникальный номер сотрудника	INT	Уникальный идентификатор сотрудника
Фамилия	VARCHAR(45)	Фамилия сотрудника
Имя	VARCHAR(45)	Имя сотрудника
Отчество	VARCHAR(45)	Отчество сотрудника
Код отдела	INT	Используется для определения отдела (внешний ключ)
Код должности	INT	Используется для определения должности (внешний ключ)
Код гражданства	INT	Используется для определения гражданства (внешний ключ)
Код семейного положения	INT	Используется для определения семейного положения (внешний ключ)
Дата приема на работу	DATETIME	Дата приема сотрудника на работу
Дата увольнения	DATETIME	Дата увольнения или окончания контракта
Номер страхования	INT	Личный номер страхования сотрудника

5 ОПИСАНИЕ АЛГОРИТМОВ, РЕАЛИЗУЮЩИХ БИЗНЕС-ЛОГИКУ СЕРВЕРНОЙ ЧАСТИ ПРОЕКТИРУЕМОЙ СИСТЕМЫ

При проектировании и реализации программных систем разработчики часто сталкиваются с такими понятиями, как бизнес-логика, бизнес-правила, бизнес-ограничения и т.д. Для того, чтобы раскрыть данные понятия термин «бизнес» можно заменить на понятие предметная область.

Предметная область (англ. domain) – это часть реального мира занимающаяся деятельностью, которая служит объектом автоматизации.

Бизнес-правило – это положение, определяющее или ограничивающее какие-либо стороны бизнеса (предметной области).

Бизнес-логика приложения – это описание схем, по которым приложение взаимодействует с пользователем. Например, когда пользователь авторизуется в системе, эти действия обрабатываются на сервере. Бизнес-логика в данном случае отвечает на вопросы: Что должен ввести пользователь? Соответствуют ли введенные данные заданному формату? Что произойдет после того, как пользователь нажмет на кнопку? Есть ли у пользователя права на совершение какого-либо действия.

С точки зрения разрабатываемой системой управления персоналом на предприятии основным алгоритмом, реализующим бизнес-логику, является работа с персоналом. Работа с персоналом включает в себя:

- добавление нового сотрудника на предприятие;
- редактирование данных сотрудников предприятия;
- удаление сотрудника;
- построение отчета за выбранный промежуток времени.

Алгоритм редактирования данных сотрудника представлен на рисунке 5.1.

После успешного входа в систему и перехода пользователя на вкладку «Управление кадрами предприятия» пользователю представляются все сотрудники, находящиеся в базе данных. Далее необходимо выбрать нужного сотрудника, нажав на кнопку «Выбрать». После выполнения данного действия все данные выбранного сотрудника запишутся в специально выделенный список `selected_us`, в котором соответственно будут храниться поля из таблицы.

Следующий шаг – запрос к базе данных по уникальному номеру сотрудника. Полученный результат – объект класса `ResultSet`, данные из которого записываются в объект `ObservableList` для редактирования. Также данные из объекта класса `ResultSet` отображаются в пользовательском интерфейсе.

Для совершения действия редактирования пользователя необходимо нажать на кнопку «Редактировать». На этом этапе все поля из объекта `ObservableList` переносятся в виджеты пользовательского интерфейса для их редактирования.

После редактирования всех необходимых данных выбранного сотрудника необходимо нажать на кнопку «Сохранить». После этого осуществляется запрос к базе данных на обновление информации о сотруднике. Об успешном выполнении операции редактирования или же, наоборот, об ошибке пользователя информирует диалоговое окно.

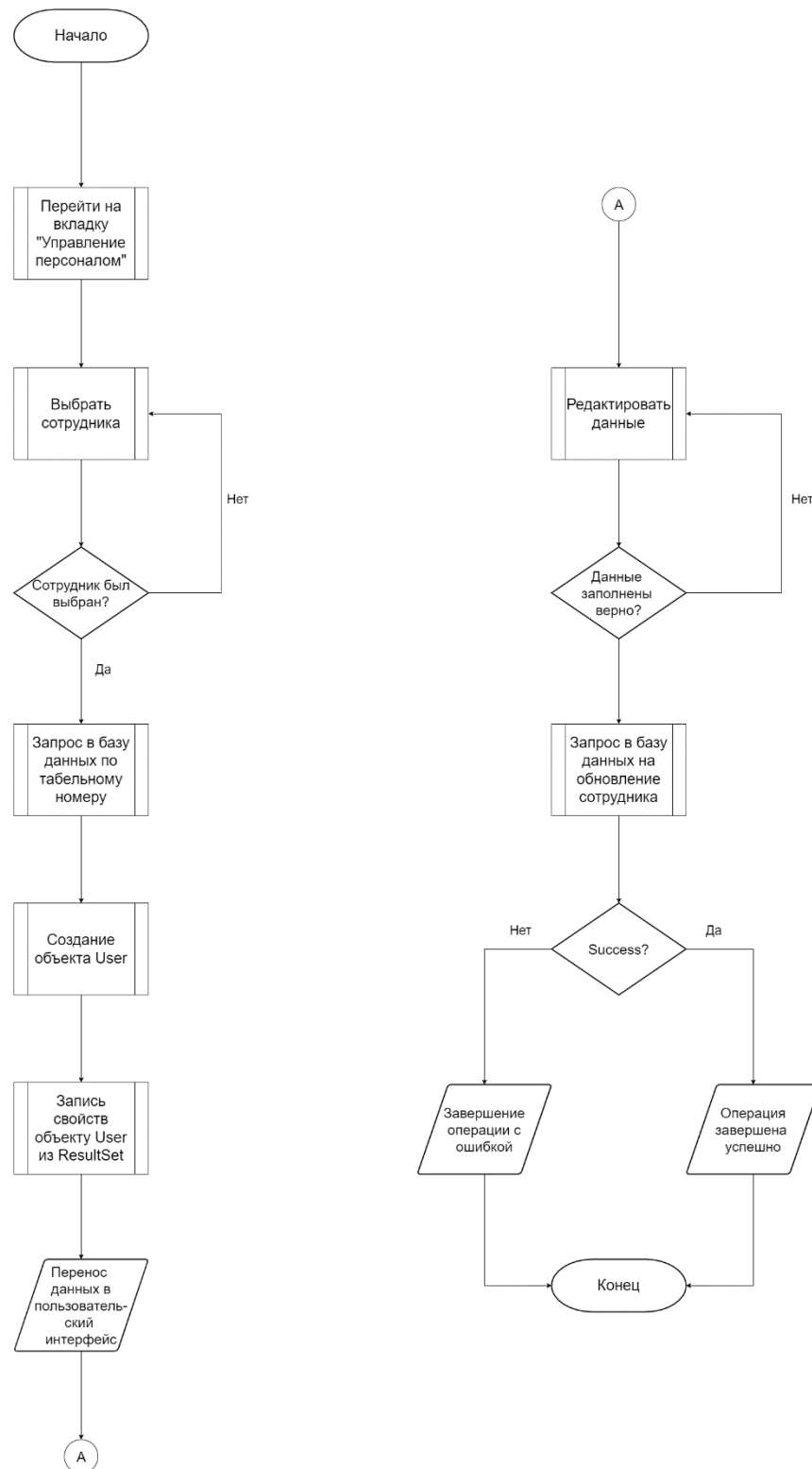


Рисунок 5.1 – Алгоритм редактирования данных сотрудника

6 МОДЕЛИ ПРЕДСТАВЛЕНИЯ СИСТЕМЫ И ИХ ОПИСАНИЕ

UML (англ. Unified Modeling Language) – унифицированный язык моделирования. Он состоит из интегрированного набора диаграмм, разработанных, чтобы помочь разработчикам систем и программного обеспечения в определении, визуализации, конструировании и документировании артефактов программных систем, а также, к примеру, для бизнес-моделирования.

Одна из задач UML – служить средством коммуникации внутри команды и при общении с заказчиком.

Как и любой другой язык, UML имеет собственные правила оформления моделей и синтаксис. С помощью графической нотации UML можно визуализировать систему, объединить все компоненты в единую структуру, уточнять и улучшать модель в процессе работы. На общем уровне графическая нотация UML содержит 4 основных типа элементов:

- фигуры;
- линии;
- значки;
- надписи.

Для создания диаграмм можно использовать различные сервисы, такие как diagrams.net, google drawings, microsoft visio и т.д.

Диаграммы UML подразделяют на два типа — это структурные диаграммы и диаграммы поведения. Структурные диаграммы показывают статическую структуру системы и ее частей на разных уровнях абстракции и реализации, а также их взаимосвязь [14]. Существует семь типов структурных диаграмм. Среди них можно выделить следующие диаграммы:

- диаграмма классов;
- диаграмма развертывания;
- диаграмма компонентов.

Диаграммы поведения показывают динамическое поведение объектов в системе, которое можно описать, как серию изменений в системе с течением времени. К диаграммам поведения относятся:

- диаграмма прецедентов;
- диаграмма состояний;
- диаграмма последовательности.

Перечисленные выше диаграммы были построены для разработанной системы.

6.1 Диаграмма классов

Диаграмма классов — это центральная методика моделирования, которая используется практически во всех объектно-ориентированных методах. Эта диаграмма описывает типы объектов в системе и различные виды статических отношений, которые существуют между ними [13].

Диаграмма классов дает наиболее полное и развернутое представление о связях в программном коде, функциональности и информации об отдельных классах.

Выделяют 3 наиболее важных типа связей на диаграмме классов: ассоциация, агрегация и наследование. Ассоциация показывает отношения между экземплярами классов. Агрегация показывает связь целого с частью (например, когда один класс является частью другого). Наследование – это отношение, непосредственно соответствующее наследованию в объектно-ориентированном программировании.

Диаграмма классов проектируемой системы представлена на рисунке 6.1. на диаграмме отображены связи наследования между родительскими и дочерними классами. Кроме того, на ней отображен созданный интерфейс Navigatable, который реализуют классы-контроллеры пользовательского интерфейса. ReactStatus – класс, который реализуют для отображения итога выполнения операции.

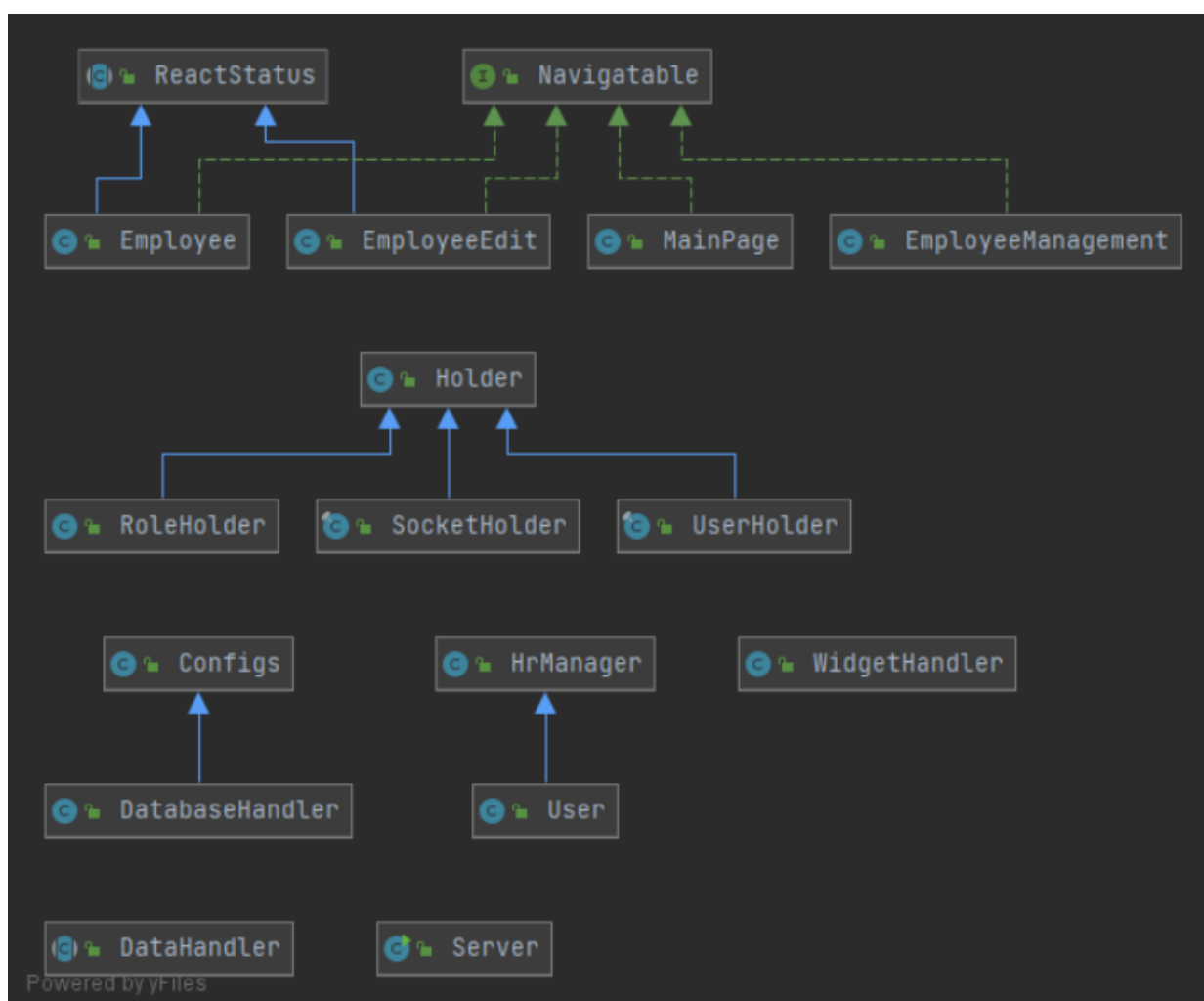


Рисунок 6.1 – Диаграмма классов

При проектировании и реализации программных систем разработчики часто сталкиваются с такими понятиями, как бизнес-логика, бизнес-правила,

бизнес-ограничения и т.д. Для того, чтобы раскрыть данные понятия термин «бизнес» можно заменить на понятие предметная область.

6.2 Диаграмма компонентов

На языке унифицированного моделирования диаграмма компонентов показывает, как компоненты соединяются вместе для формирования более крупных компонентов или программных систем.

Она иллюстрирует архитектуры компонентов программного обеспечения и зависимости между ними.

Другими словами, диаграмма компонентов дает упрощенное представление о сложной системе, разбивая ее на более мелкие компоненты. Каждый из элементов показан в прямоугольной рамке с названием, написанным внутри. Соединители определяют отношения и зависимости между различными компонентами [13].

Среди основных целей диаграммы компонентов можно выделить:

- визуализация общей структуры исходного кода программы;
- обеспечение многократного использования фрагментов кода;
- представление концептуальной и физической базы данных.

Диаграмма компонентов представлена на рисунке 5.2. Она позволяет определить состав компонентов программы, а также установить зависимость между ними.

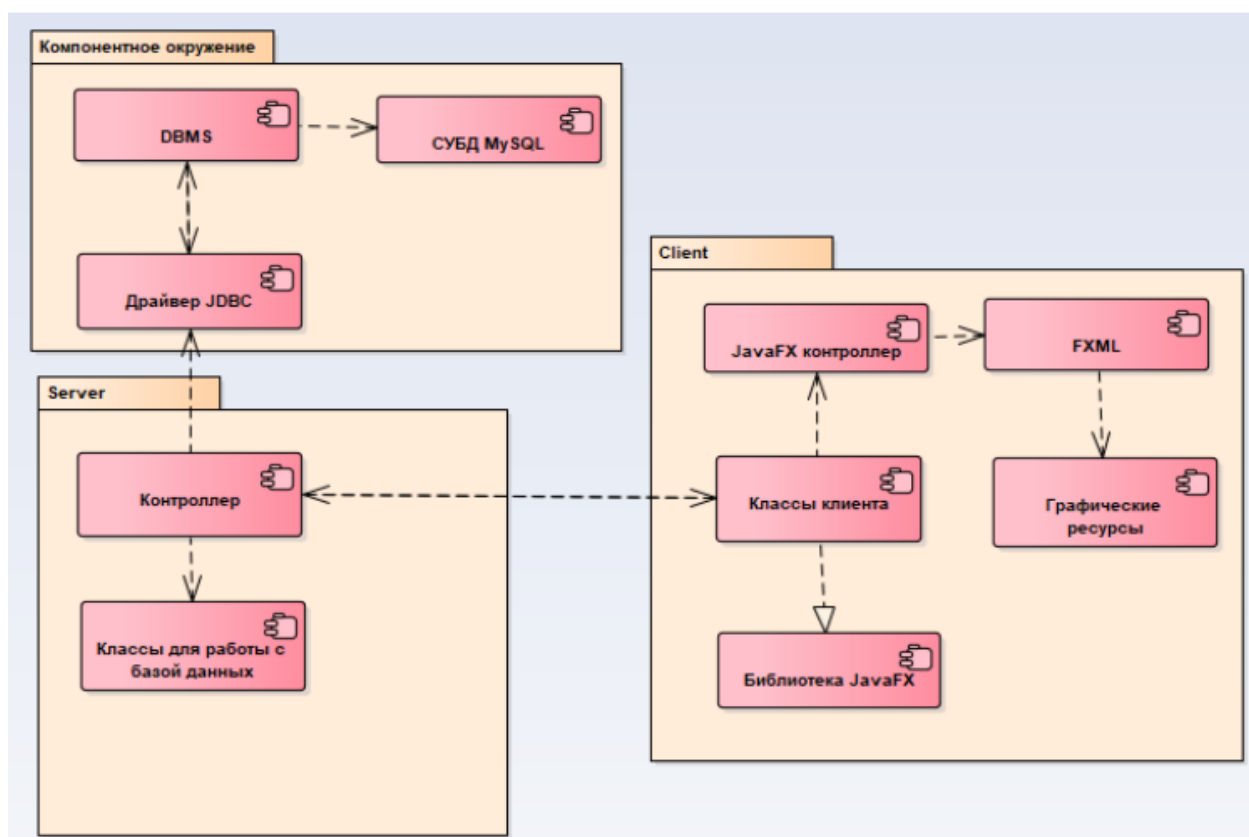


Рисунок 6.2 – Диаграмма компонентов

6.3 Диаграмма развертывания

Диаграмма развертывания помогает моделировать физический аспект объектно-ориентированной программной системы. Это структурная схема, которая показывает архитектуру системы, как развертывание (дистрибуции) программных артефактов. То есть на этой диаграмме показаны аппаратные (узлы) и программные (артефакты) компоненты и их взаимосвязи. Она предлагает наглядное представление о том, где именно развернут каждый программный компонент [13].

Артефакты представляют собой конкретные элементы в физическом мире, которые являются результатом процесса разработки.

Диаграмма моделирует конфигурацию времени выполнения в статическом представлении и визуализирует распределение артефактов в приложении.

В большинстве случаев это включает в себя моделирование конфигураций оборудования вместе с компонентами программного обеспечения, на которых они размещены.

Диаграмма развертывания содержит графические изображения процессоров, устройств, процессов и связей между ними. В отличие от диаграмм логического представления, диаграмма развертывания является единственной для системы в целом, поскольку должна отражать все особенности ее реализации. Разработка диаграммы развертывания, как правило, является последним этапом спецификации модели. Диаграмма развертывания разрабатывается совместно системными аналитиками, сетевыми инженерами и системотехниками.

Помимо демонстрации топологии системы и распределения ее компонентов по узлам, диаграмма развертывания также позволяет отследить маршруты передачи между аппаратными узлами.

От правильности построения данной диаграммы зависит и производительность системы, т.к. позволяет рационально распределить компоненты по узлам системы.

Кроме того, диаграмма развертывания позволяет решить такую задачу, как обеспечение безопасности системы.

Разработанная диаграмма развертывания представлена на рисунке 6.3. Цели разработки данной диаграммы можно сформулировать так:

- распределение компонентов системы по ее физическим узлам;
- отображение физических связей между узлами системы на этапе исполнения;
- выявление узких мест системы и реконфигурация ее топологии с целью достижения требуемой производительности.

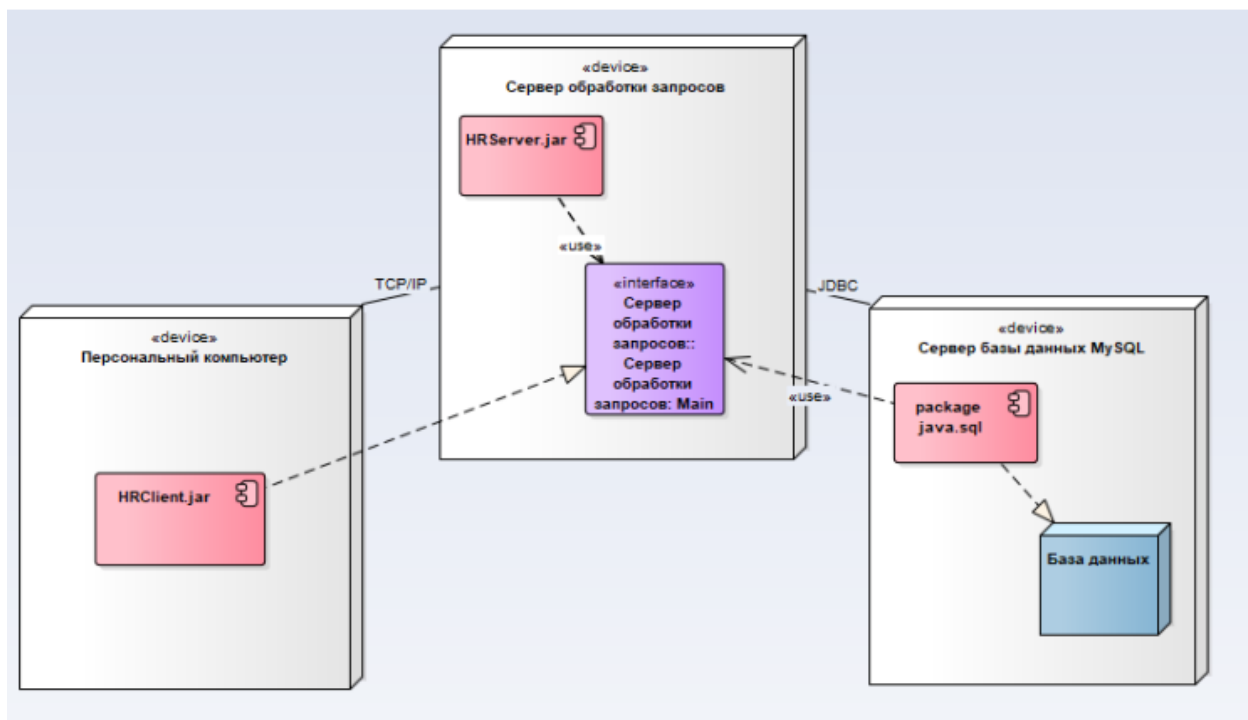


Рисунок 6.3 – Диаграмма развертывания

6.4 Диаграмма состояний

Диаграмма состояний относится к поведенческому виду. Она показывает, как объект переходит из одного состояния в другое.

Все объекты в системе характеризуются поведением и состоянием, в котором они находятся.

Состояние (state) - ситуация в жизненном цикле объекта, во время которой он удовлетворяет некоторому условию, выполняет определенную деятельность или ожидает какого-то события. Состояние объекта определяется значениями некоторых его атрибутов и присутствием или отсутствием связей с другими объектами.

Диаграммы состояний отображают разрешенные состояния и переходы, а также события, которые влияют на эти переходы. Она помогает визуализировать весь жизненный цикл объектов и, таким образом, помогает лучше понять системы, основанные на состояниях.

Очевидно, что диаграммы состояний служат для моделирования динамических аспектов системы (как и диаграммы последовательностей, кооперации, прецедентов и, как мы увидим далее, диаграммы деятельности).

Диаграмма состояний полезна при моделировании жизненного цикла объекта.

От других диаграмм диаграмма состояний отличается тем, что описывает процесс изменения состояний только одного экземпляра определенного класса - одного объекта, причем объекта реактивного, то есть объекта, поведение которого характеризуется его реакцией на внешние события.

На рисунке 6.4 представлена диаграмма состояний сотрудника при его добавлении.

На диаграмме видно, что, когда пользователь переходит на страницу управления персоналом, новый сотрудник не добавлен. Далее, после нажатия на кнопку «Добавить», открывается окно добавления сотрудника, заполняется необходимая информация, после чего передается на проверку. Если все заполнено правильно, запись о новом сотруднике добавляется в базу данных. Если же поля не заполнены либо заполнены неверно – отправляется запрос на повторный ввод.

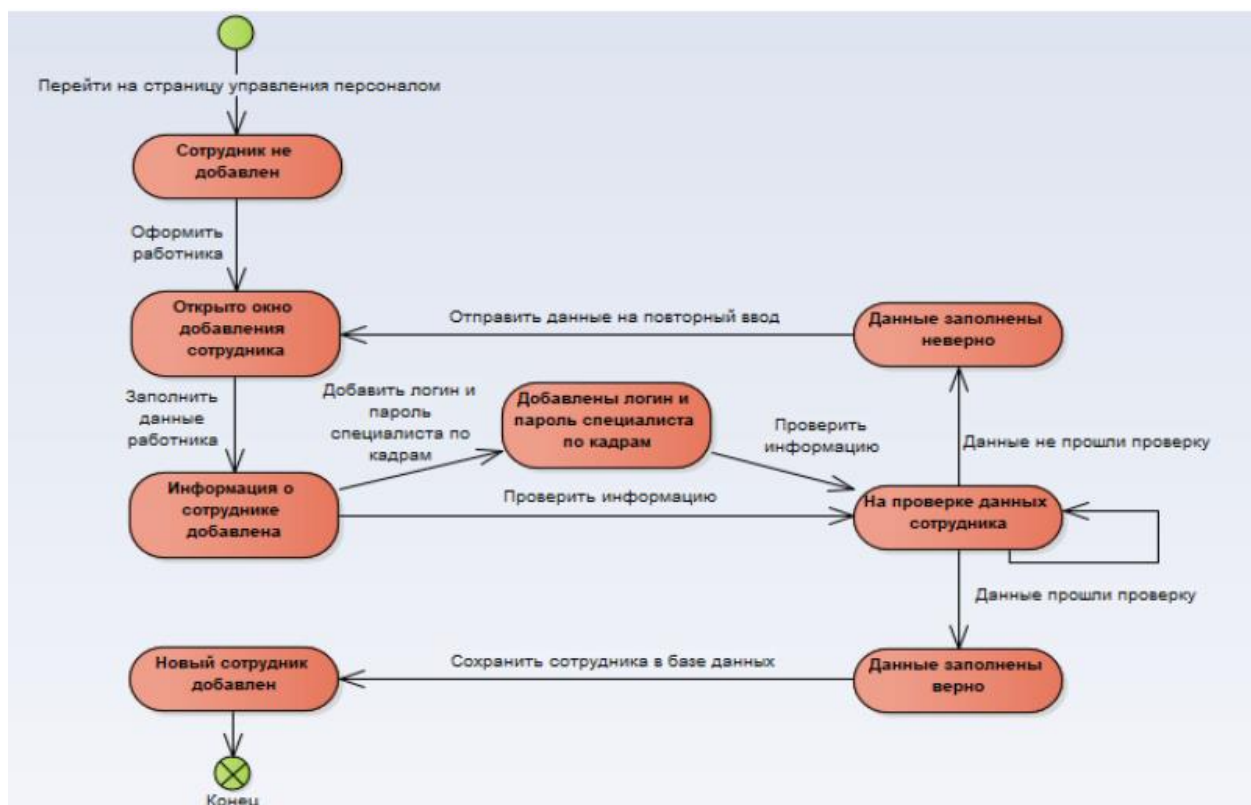


Рисунок 6.4 – Диаграмма состояний сотрудника при добавлении

Скругленные прямоугольники представляют собой состояния, через которые проходит объект в течение своего жизненного цикла. Стрелками обозначаются переходы между состояниями, которые вызваны выполнением некоторых действий.

6.5 Диаграмма последовательности

Диаграмма последовательности – это поведенческий тип UML диаграмм, использующийся для описания логики сценариев использования.

Диаграмма последовательности моделирует взаимодействие объектов на основе временной последовательности. Она показывает, как одни объекты взаимодействуют с другими в конкретном прецеденте.

Данный вид диаграммы содержит объекты, которые взаимодействуют в рамках сценария, сообщения, передаваемые другим объектам, а также полу-

чаемые ответы. У каждого объекта есть определенная линия жизни, в течение которой он способен функционировать и общаться с другими объектами [16].

На рисунке 6.5 представлена диаграмма последовательности, которая демонстрирует работу администратора по выбору сотрудника предприятия и получении данных о нем.

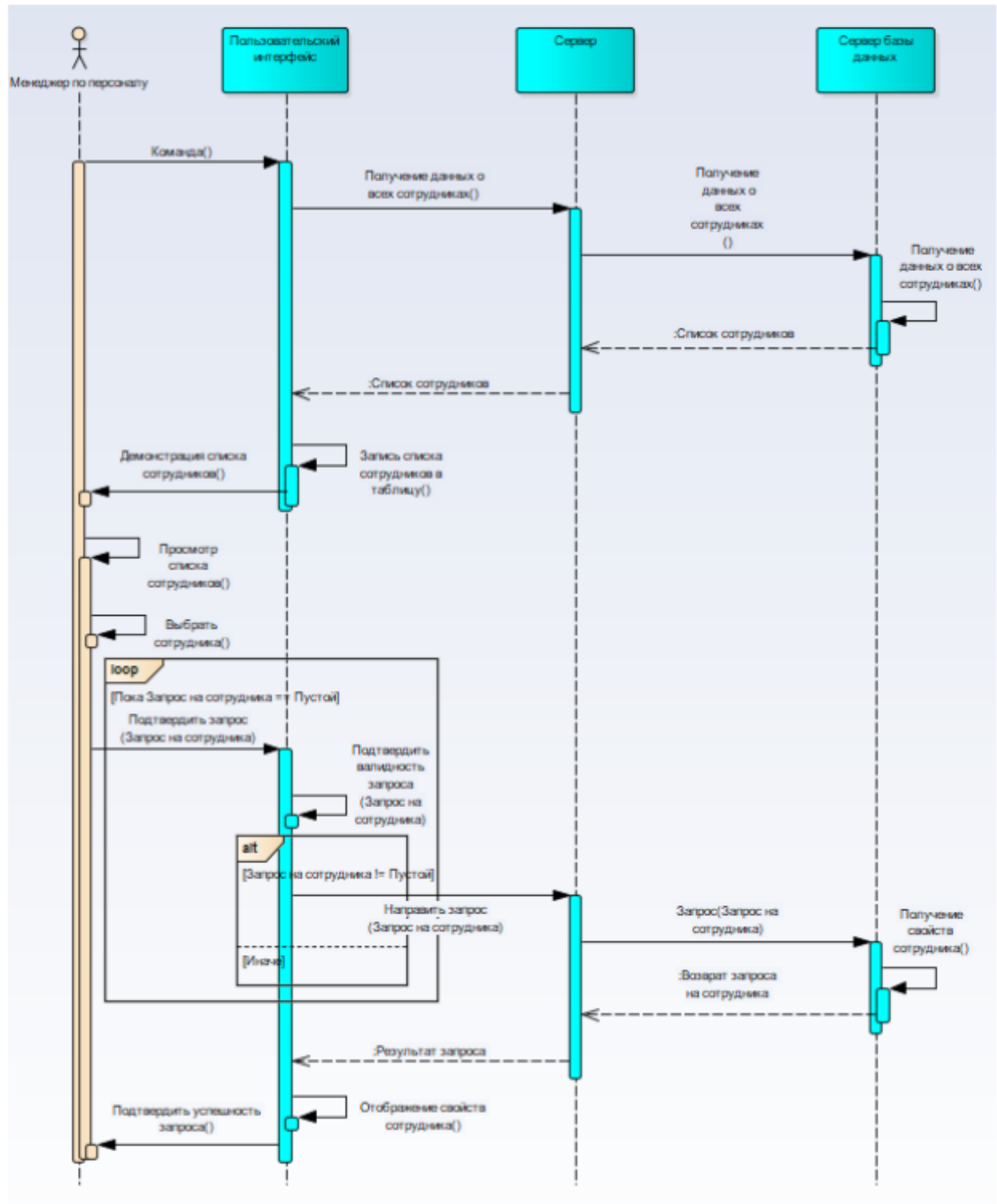


Рисунок 6.5 – Диаграмма последовательности просмотра данных сотрудника

Итак, сначала администратору необходимо отправить команду в пользовательский интерфейс о том, что он хочет посмотреть всех сотрудников. Далее интерфейс передает сообщение на сервер, через который на базе данных выполняется запрос и передается возвращаемое значение в виде списка всех сотрудников, которые работают на предприятии и находятся в базе данных. Список выводится на экран в виде таблицы в пользовательском интерфейсе для демонстрации администратору.

Следующий шаг работы пользователя – это выбор нужного сотрудника, подробную информацию о котором он желает посмотреть. Если пользователь не сделал выбор, то интерфейс будет ожидать выбора в цикле.

После выбора нужного сотрудника, запрос передается на сервер, а из него – в базу данных для получения информации. На обратном пути в пользовательский интерфейс возвращается результат запроса, а именно вся подробная информация о сотруднике, находящаяся в базе данных.

По завершении пользовательский интерфейс передает пользователю системы уведомление об успешности выполнения запроса.

6.6 Диаграмма вариантов использования

Диаграмма последовательности – это поведенческий тип UML диаграмм, использующийся для описания логики сценариев использования.

Диаграмма вариантов использования - это исходное концептуальное представление или концептуальная модель системы в процессе ее проектирования и разработки. Создание диаграммы вариантов использования имеет следующие цели:

- определить общие границы и контекст моделируемой предметной области на начальных этапах проектирования системы;
- сформулировать общие требования к функциональному поведению проектируемой системы;
- разработать исходную концептуальную модель системы для ее последующей детализации в форме логических и физических моделей;
- подготовить исходную документацию для взаимодействия разработчиков системы с ее заказчиками и пользователями.

Диаграмма вариантов использования использует 2 основных элемента.

Участник, или действующее лицо (англ. actor)— множество логически связанных ролей, исполняемых при взаимодействии с прецедентами или сущностями (система, подсистема или класс). Участником может быть человек, роль человека в системе или другая система, подсистема или класс, которые представляют нечто вне сущности.

Прецедент, или вариант использования (англ. use case) — описание отдельного аспекта поведения системы с точки зрения пользователя. Прецедент не показывает, "как" достигается некоторый результат, а только "что" именно выполняется.

Между вариантами использования и действующими лицами на диаграмме вариантов использования устанавливаются следующие 4 типа отношений:

- отношение ассоциации;
- отношение обобщения;
- отношение включения;
- отношение расширения.

Ассоциация – это связь между вариантом использования и действующим лицом.

Обобщение означает, что один вариант использования или актер может быть обобщен от другого. Тогда первый вариант использования будет дочерним, а второй – родительским. Следует отметить, что дочерний вариант использования наследует от родительского все свойства, но при этом может иметь и свои свойства.

Отношение включения (англ. include) означает, что поведение одного варианта использования является обязательным компонентом поведения другого.

Отношение расширения (англ. extend) применяется тогда, когда свойства одного варианта использования могут быть дополнены свойствами другого.

На рисунке 6.6 представлена диаграмма вариантов использования для администратора и пользователя. Они изображены на ней в виде актеров. Стоит отметить, что актер на диаграмме вариантов использования – это собирательный образ и не означает конкретно одного человека. Например, актер «Администратор» не обозначает одного конкретного администратора, а всех пользователей системы, имеющих роль «Администратор». Таких пользователей может быть сколько угодно много.

В виде овалов на диаграмме вариантов использования приведены варианты использования системы, которые связаны с актерами отношением ассоциаций. При этом некоторые варианты использования связаны друг с другом отношениями включения или обобщения.

На диаграмме администратор обобщен от пользователя. То есть в данном случае администратор является дочерним актером, а пользователь – родительским. Это было сделано, потому что администратор имеет все те же варианты использования, что и пользователь. Однако, он также имеет дополнительные свойства, не присущие пользователю, такие как регистрация в системе, настройки и т.д.

Отношение включения означает, что информация о статусе пользователя в системе обязательно включает в себя как текущую дату, так и роль пользователя в системе.

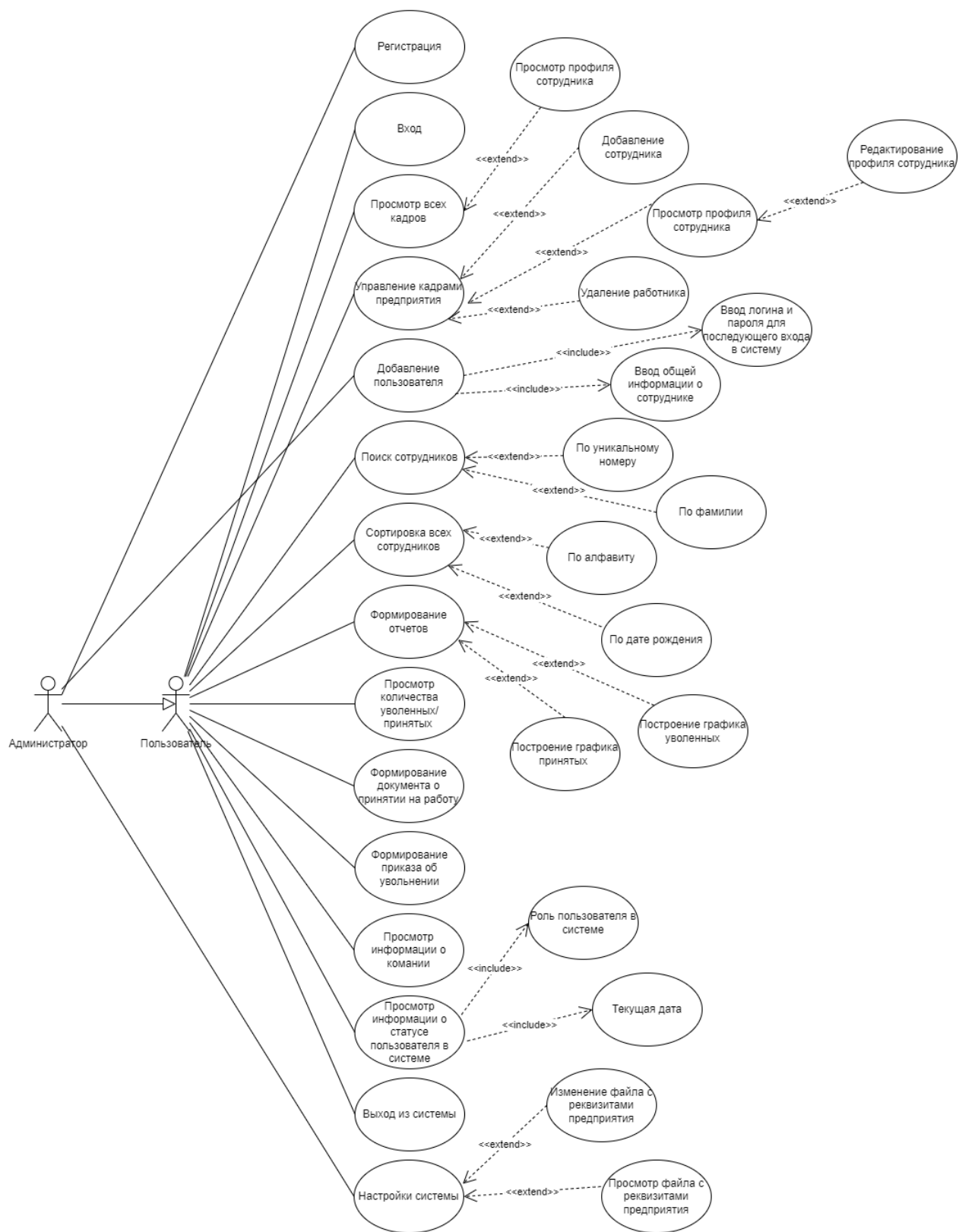


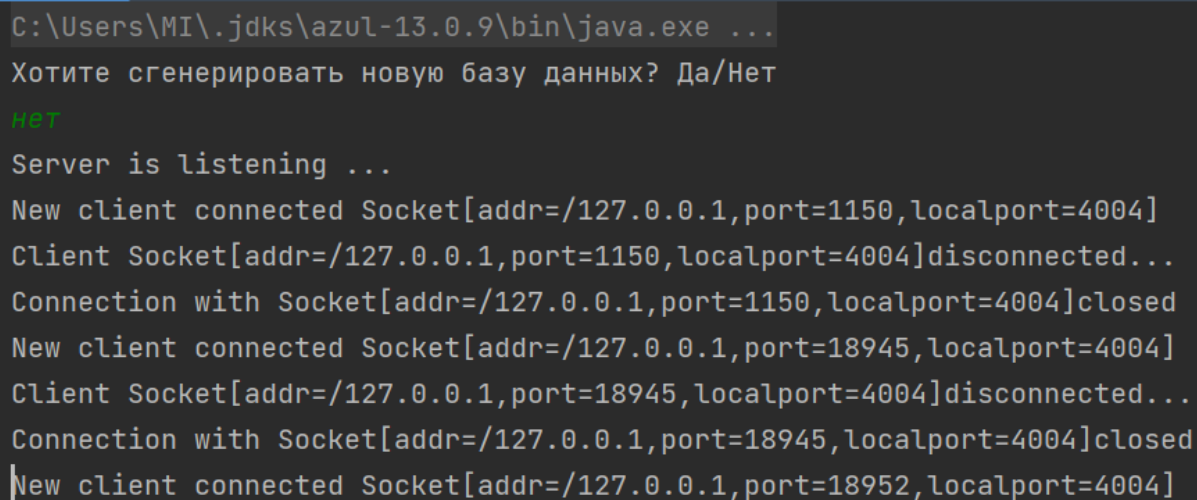
Рисунок 6.6 – Диаграмма вариантов использования

7 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Как было отмечено ранее, разработанная система программной поддержки деятельности службы HR-службы организации представляет собой клиент-серверное приложение, позволяющее одновременно работать нескольким пользователям.

Пользователю дана возможность авторизации (с разными ролями), добавления, редактирования, удаления данных сотрудников, сортировки и поиска необходимого сотрудника по нескольким параметрам, а также построения визуальных отчетов, изменения настроек предприятия и автоматического создания основных документов.

Как и во всех клиент-серверных приложениях, сначала необходимо запустить сервер, а потом – клиента. При запуске сервера приложение задает вопрос о необходимости генерировать новую базу данных. После ответа на вопрос сервер переходит в режим ожидания запросов от клиентов (см. рис. 7.1).



```
C:\Users\MI\.jdk\azul-13.0.9\bin\java.exe ...
Хотите сгенерировать новую базу данных? Да/Нет
нет
Server is listening ...
New client connected Socket[addr=/127.0.0.1,port=1150,localport=4004]
Client Socket[addr=/127.0.0.1,port=1150,localport=4004]disconnected...
Connection with Socket[addr=/127.0.0.1,port=1150,localport=4004]closed
New client connected Socket[addr=/127.0.0.1,port=18945,localport=4004]
Client Socket[addr=/127.0.0.1,port=18945,localport=4004]disconnected...
Connection with Socket[addr=/127.0.0.1,port=18945,localport=4004]closed
New client connected Socket[addr=/127.0.0.1,port=18952,localport=4004]
```


Рисунок 7.1 – Результат запуска сервера

В процессе работы, сервер также выводит информацию о подключившихся и отключившихся клиентах.

После запуска программного средства пользователям по умолчанию предлагается войти в систему, либо же выбрать регистрацию. Как отмечалось ранее, зарегистрироваться может только администратор, он же – управляющий отделом кадров на предприятии.

Первый экран приложения продемонстрирован на рисунке 7.2.

Приложение по управлению персоналом на предприятии



Вход

Введите пароль

Войти

Зарегистрироваться

Рисунок 7.2 – Экран входа в приложение

Сначала рассмотрим вход обычного пользователя в систему (сотрудника отдела кадров) с ролью «Пользователь».

Для входа в приложение необходимо ввести имя пользователя и пароль (рисунок 7.2). После ввода необходимых данных и нажатия на кнопку «Войти», при наличии введенных пользователем данных для авторизации в базе данных, происходит успешный вход в систему и открывается главная страница для пользователя.

Если данных пользователя нет в базе данных, он может зарегистрироваться в системе, нажав на кнопку «Зарегистрироваться». После этого он перейдет на экран регистрации. Экран регистрации представлен на рисунке 7.3.

Для регистрации также необходимо ввести имя пользователя и пароль. После регистрации пользователю по умолчанию присваивается роль «Администратор».

Как только пользователь регистрируется в системе он переходит на главную страницу для администратора, с отличающимся функционалом.

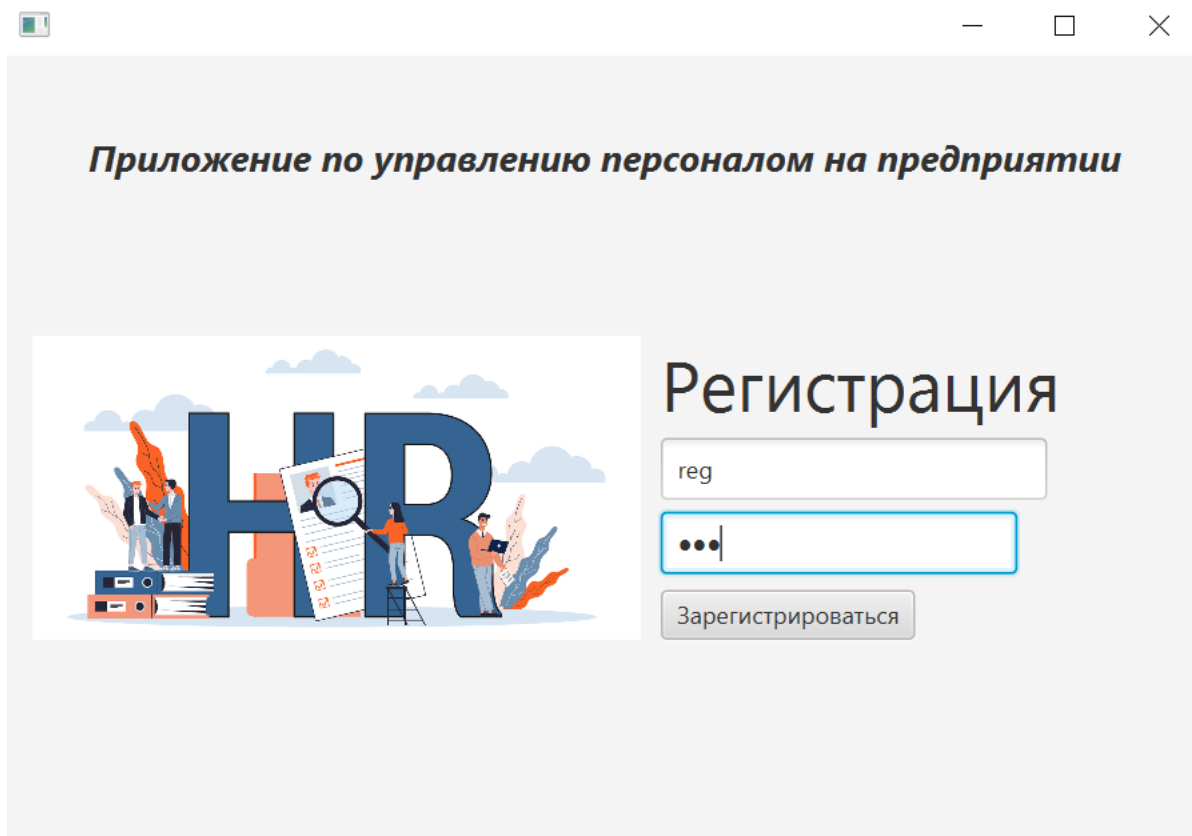


Рисунок 7.3 – Экран регистрации

Итак, главная страница администратора представлена ниже на рисунке 7.4. Она разделена на две части: меню и основную информацию.

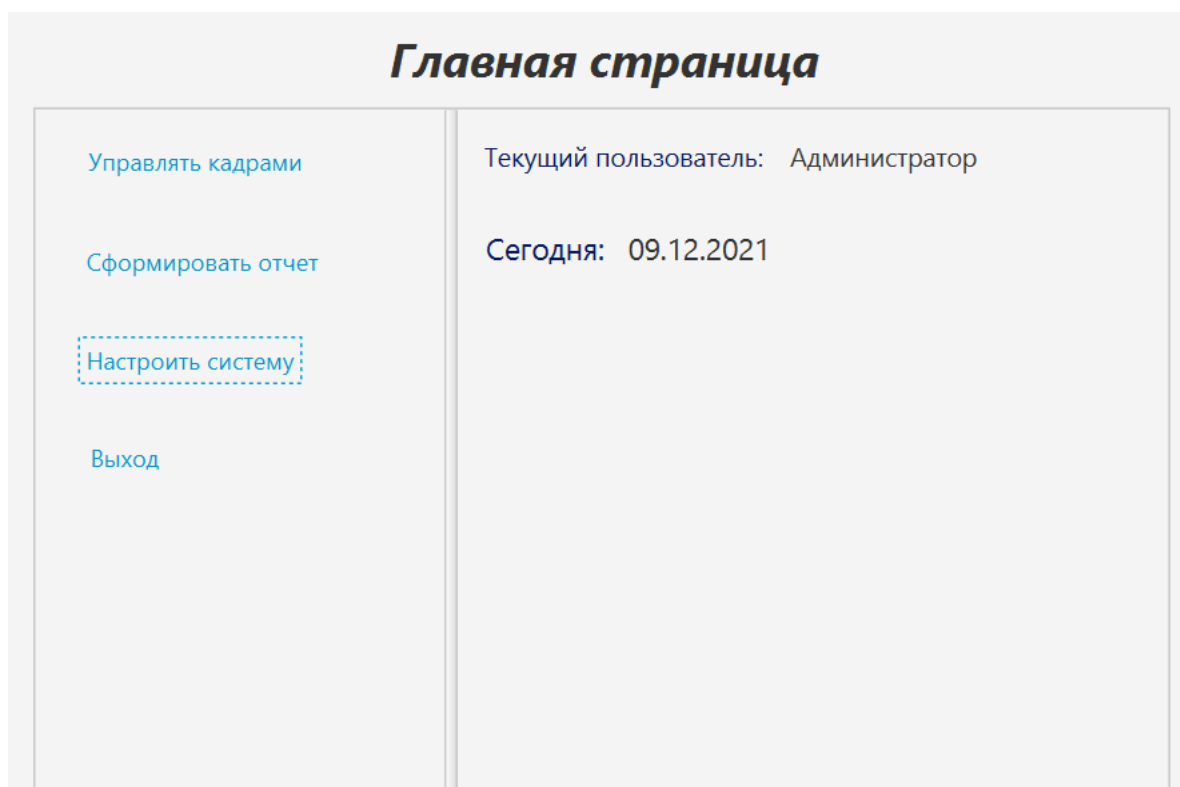


Рисунок 7.4 – Главная страница администратора

Итак, пользователю даются возможности:

- управления кадрами;
- формирования визуальных отчетов;
- выхода из системы.

Справа отображается роль текущего пользователя, а также актуальная дата.

Рассмотрим вкладку «Управление кадрами предприятия». При переходе по ней пользователь попадает на следующую страницу, которая представлена на рисунке 7.5.

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
28	yu	yu	yu	2021-12-02T0...
29	uliana	ulianova	al	2021-12-01T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...
32	dfgh	sdfg	sdfg	2021-12-03T0...
34	jhg	jhgf	jhgf	2021-12-01T0...

Рисунок 7.5 – Управление кадрами

Главное место на данном экране занимает таблица с сотрудниками предприятия. В ней отображаются:

- уникальный ID сотрудника;
- фамилия сотрудника;
- имя сотрудника;
- отчество сотрудника;
- дата рождения сотрудника.

Над таблицей располагаются три кнопки: «Добавить», «Редактировать», «Удалить». Каждая из них позволяет, соответственно, добавлять нового сотрудника, редактировать анкеты уже существующих сотрудников и удалять сотрудника.

В левом верхнем углу располагается кнопка «Назад», ведущая пользователя на главную страницу.

Слева реализована возможность выбора поиска или сортировки. Для того, чтобы найти сотрудника необходимо ввести известные данные в поле для ввода, затем выбрать критерий, по которому будет осуществляться по-

иск, и нажать на кнопку «Ок». Поиск можно проводить на ФИО сотрудника или по уникальному номеру сотрудника. Результаты поиска отобразятся слева, путем исключения из таблицы неподходящих полей.

Ниже, на рисунке 7.6 демонстрируется поиск по ФИО сотрудника.

Управление кадрами

Назад

Во

☒ по ФИО
☐ по табельному номеру

Сортировать сотрудников

☐ по алфавиту
☐ по дате рождения

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...

Рисунок 7.6 – Поиск по фамилии сотрудника

На рисунке 7.7 демонстрируется поиск сотрудника по его уникальному номеру.

Управление кадрами

Назад

3

☐ по ФИО
☒ по табельному номеру

Сортировать сотрудников

☐ по алфавиту
☐ по дате рождения

ID	Фамилия	Имя	Отчество	Дата рождения
31	Воронцов	Александр	Анатольевич	2021-12-02T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...

Рисунок 7.7 – Поиск по номеру сотрудника

Для сортировки необходимо выбрать критерий, по которому будет произведена сортировка, а затем нажать на кнопку «Ок».

Сортировка может быть осуществлена по алфавиту, а также по дате рождения сотрудников. Результаты сортировки также отобразятся слева, путем расположения полей в необходимом порядке. Сортировка по алфавиту представлена на рисунке 7.8.

The screenshot shows a web application titled "Управление кадрами". On the left, there is a sidebar with a "Назад" button at the top. Below it is a search section with a "Поиск" input field and an "Ок" button. Further down are two radio buttons: "по ФИО" (selected) and "по табельному номеру". Below these is a section "Сортировать сотрудников" with an "Ок" button and two radio buttons: "по алфавиту" (selected) and "по дате рождения". The main area on the right contains three buttons: "Добавить", "Выбрать", and "Удалить". Below these is a table with the following data:

ID	Фамилия	Имя	Отчество	Дата рождения
37	Васильев	Денис	Сергеевич	2002-03-15T0...
5	Воронова	Юлиана	Александровна	1998-08-08T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...

Рисунок 7.8 – Сортировка по алфавиту

Сортировка по дате рождения сотрудников представлена на рисунке 7.9.

The screenshot shows the same web application "Управление кадрами". In the sidebar, the "по дате рождения" radio button is now selected. The table in the main area is sorted by date of birth, showing the following data:

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...

Рисунок 7.9 – Сортировка по дате рождения

При нажатии на кнопку «Добавить» пользователь переходит на страницу добавления нового сотрудника (рисунок 7.10)

Назад

Добавить сотрудника

Васильев Денис Сергеевич

Дата рождения: 15.03.2002

Гражданство: Украина

Семейное положение: Есть супруг(а) 1 детей

Номер социального страхования: 1234567

Пол: ☐ Женский ☒ Мужской

Подразделение: Финансовый отдел

Должность: Менеджер по проектам

Дата приема: 05.12.2015

Дата окончания контракта: 29.12.2023

Добавить

Рисунок 7.10 – Добавление сотрудника

Здесь ему предлагается заполнить следующие поля:

- фамилия сотрудника;
- имя сотрудника;
- отчество сотрудника;
- гражданство;
- семейное положение;
- номер страховки;
- пол сотрудника;
- отдел;
- должность;

- дата приема на работу;
- дата увольнения (окончания контракта).

Следует отметить, что гражданство, семейное положение, отдел и должность сотрудника выбираются из выпадающего списка, значения которого хранятся в базе данных в соответствующих им таблицах (см. рис. 7.11).

Дата рождения

Гражданство

Семейное положение

Номер социального страхования

Пол

Подразделение

Должность

Дата приема

Дата окончания контракта

Добавить

Рисунок 7.11 – Выпадающий список для выбора нужного варианта

После заполнения всех полей необходимо нажать на кнопку «Добавить», расположенную внизу экрана. На данном экране также предусмотрена возможность возврата на шаг назад, а именно на страницу управления кадрами предприятия. Кнопка «Назад» для удобства пользователя также расположена в левом верхнем углу экрана.

После успешного добавления данных нового сотрудника в таблицу специальное диалоговое окно информирует об этом (см. рис. 7.12).

В результате, после успешного добавления сотрудника в базе данных, на странице «Управление кадрами» появляется новая строка в таблице с данными нового сотрудника.

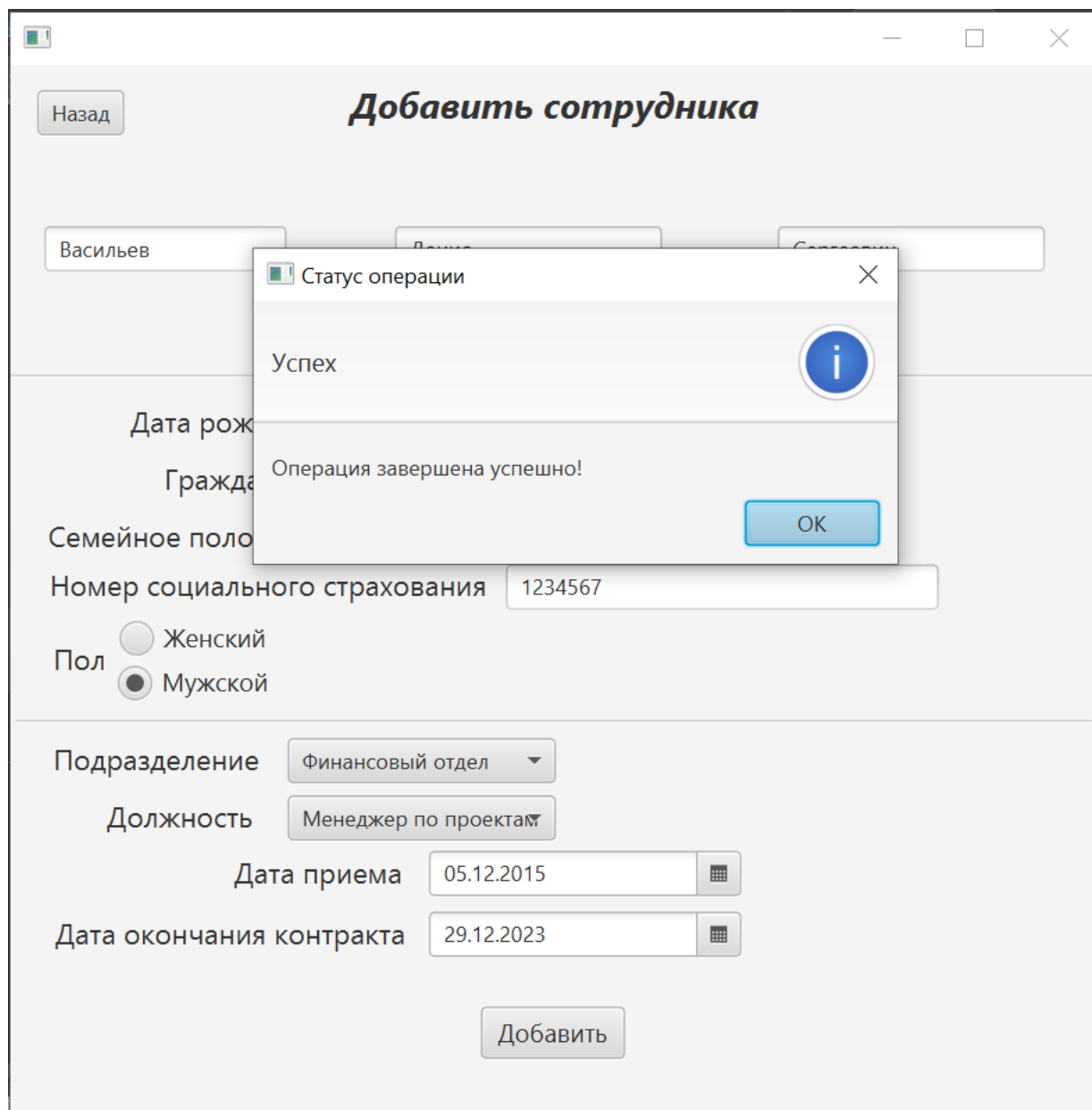


Рисунок 7.12 – Успешное добавление сотрудника

Результат добавления нового сотрудника в базу данных представлен на рисунке 7.13.

Приложение предназначено для автоматизации работы отдела кадров на предприятии, а неотъемлемая часть работы отдела кадров – это составлять различного рода документы, такие как договор о найме сотрудника или приказ об увольнении.

Поэтому в системе предусмотрена функция автоматического создания документов при принятии сотрудника на работу, а также при его увольнении.

При этом создаются документы, в которые вносятся личные данные выбранного сотрудника.

Для просмотра полной информации о сотруднике необходимо сначала выделить интересующего сотрудника, а затем нажать на кнопку «Выбрать».

Назад

Управление кадрами

Поиск

Ок

☐ по ФИО
 ☐ по табельному номеру

Сортировать сотрудников

Ок

☐ по алфавиту
 ☐ по дате рождения

Добавить

Выбрать

Удалить

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
28	yu	yu	yu	2021-12-02T0...
29	uliana	ulianova	al	2021-12-01T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...
32	dfgh	sdfg	sdfg	2021-12-03T0...
34	jhg	jhgf	jhgf	2021-12-01T0...
36	jh	jh	jh	2021-12-02T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...

Рисунок 7.13 – Результат добавления нового сотрудника

Далее осуществляется переход на страницу со всеми данными о выбранном сотруднике. Данная страница представлена на рисунке 7.14.

Назад

Просмотр профиля сотрудника

Табельный номер 37

Васильев Денис Сергеевич

Редактировать профиль

Дата рождения

2002-03-15 00:00:00

Гражданство

Украина

Семейное положение

Есть супруг(а) 1 детей

Номер социального страхования

1234567

Пол

Мужской

Подразделение

Финансовый отдел

Должность

Менеджер по проектам

Дата приема на работу

2015-12-05 00:00:00

Дата окончания контракта

2023-12-29 00:00:00

Рисунок 7.14 – Просмотр профиля сотрудника

При необходимости редактирования данных необходимо нажать на кнопку «Редактировать профиль», находящуюся над основной информацией о сотруднике.

После нажатия на кнопку происходит переход на страницу редактирования, на которой по умолчанию будут находиться изначальные данные сотрудника (см. рисунок 7.15). Следует отметить, что уникальный номер сотрудника нельзя редактировать.

На данном этапе также возможен выход к предыдущей странице, а именно к «Управлению кадрами».

The screenshot shows a web application window titled "Редактировать сотрудника" (Edit Employee). The window has standard OS controls (minimize, maximize, close) in the top right corner. Below the title bar, there is a header section with the title "Редактировать сотрудника" in bold, italicized font, and a text input field containing "Табельный номер 37". To the right of this field is a "Назад" (Back) button. Below the header, the form is organized into several sections. The first section contains three text input fields: "Васильев" (highlighted with a blue border), "Денис", and "Сергеевич". The second section contains four fields: "Дата рождения" (Date of birth) with value "2002-03-15", "Гражданство" (Citizenship) with a dropdown menu showing "Украина", "Семейное положение" (Marital status) with a dropdown menu showing "Есть супруг(а) 1 детей", and "Номер социального страхования" (Social security number) with value "1234567". Below these are two radio buttons for "Пол" (Gender): "Женский" (Female) and "Мужской" (Male). The third section contains two dropdown menus: "Подразделение" (Department) showing "Финансовый отдел" and "Должность" (Position) showing "Менеджер по проектам". The fourth section contains two text input fields: "Дата приема" (Date of admission) with value "2015-12-05" and "Дата окончания контракта" (Contract expiration date) with value "2023-12-29". At the bottom center of the form is a "Сохранить" (Save) button.

Рисунок 7.15 – Редактирование сотрудника

После редактирования необходимой информации необходимо нажать на кнопку «Сохранить». После успешного сохранения измененных данных система также информирует пользователя с помощью диалогового окна (рисунок 7.16)

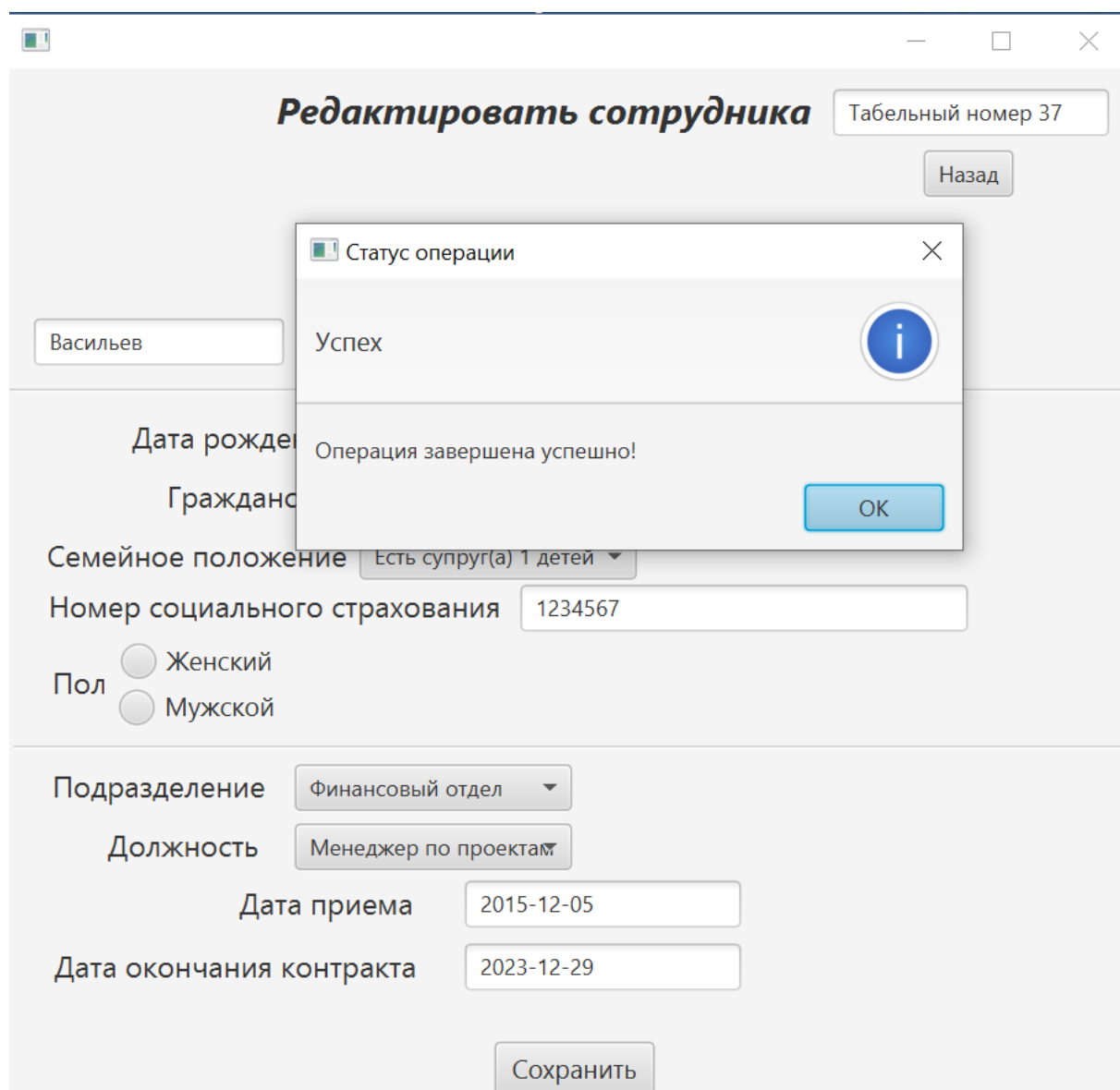


Рисунок 7.16 – Успешное редактирование сотрудника

Для удаления необходимо выбрать нужного сотрудника, а затем нажать на кнопку «Удалить». После удаления сотрудника также появляется диалоговое окно, информирующее об успешном удалении.

При добавлении или удалении сотрудников автоматически формируются договор о найме сотрудника и приказ об увольнении соответственно. Документы создаются в формате документа Microsoft Word и сохраняются на компьютер.

Следующая вкладка в меню пользователя – формирование отчетов. При переходе на нее открывается окно, в котором предлагается выбор даты, по которой будет построен отчет. Данная функция демонстрируется на рисунке 7.17.

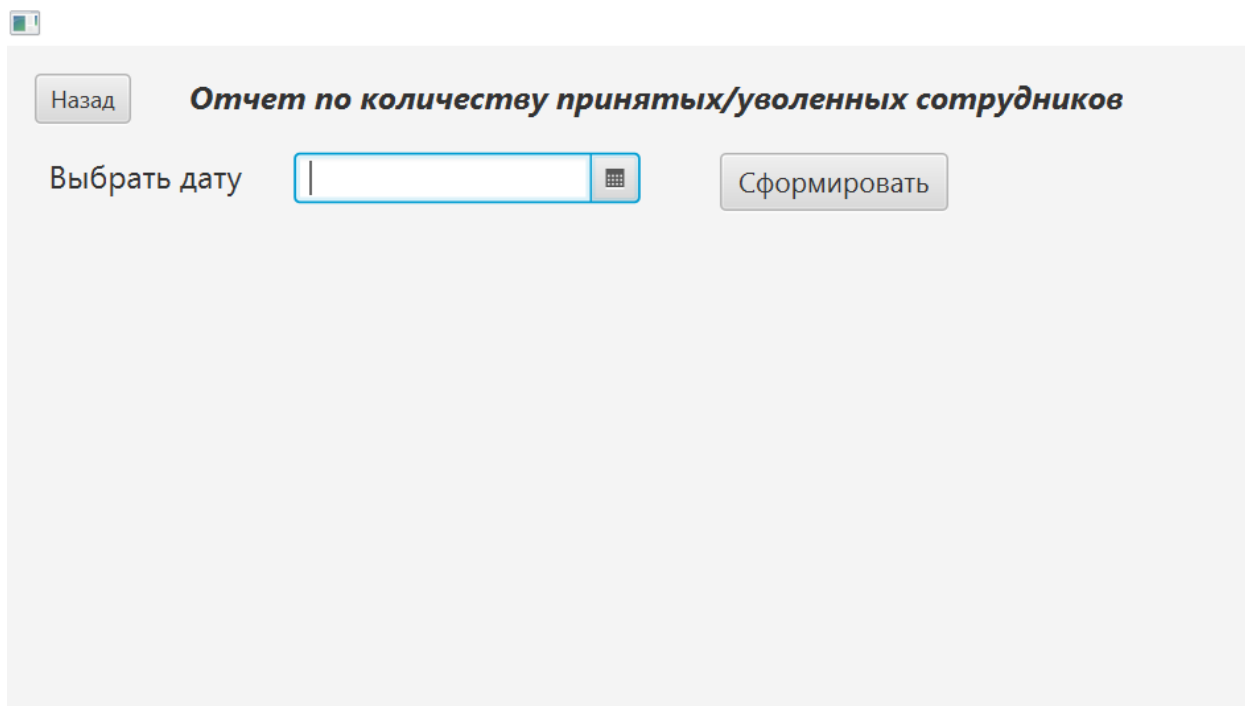


Рисунок 7.17 – Формирование отчетов

После выбора даты необходимо нажать на кнопку «Сформировать отчет». После этого будет построен график, состоящий из двух кривых, одна из которых отображает количество уволенных, а вторая – количество принятых сотрудников в выбранный пользователем год (см. рис. 7.18).

Следует отметить, что все перечисленные пункты меню («Управление сотрудниками», «Формирование отчетов») имеют одинаковый функционал как для пользователя, так и для администратора.

Отличие функционала администратора заключается в дополнительной функции – «Настройка системы». Она позволяет изменять реквизиты компании, которые в дальнейшем могут быть использованы для построения отчетов, заключения договоров и т.д.

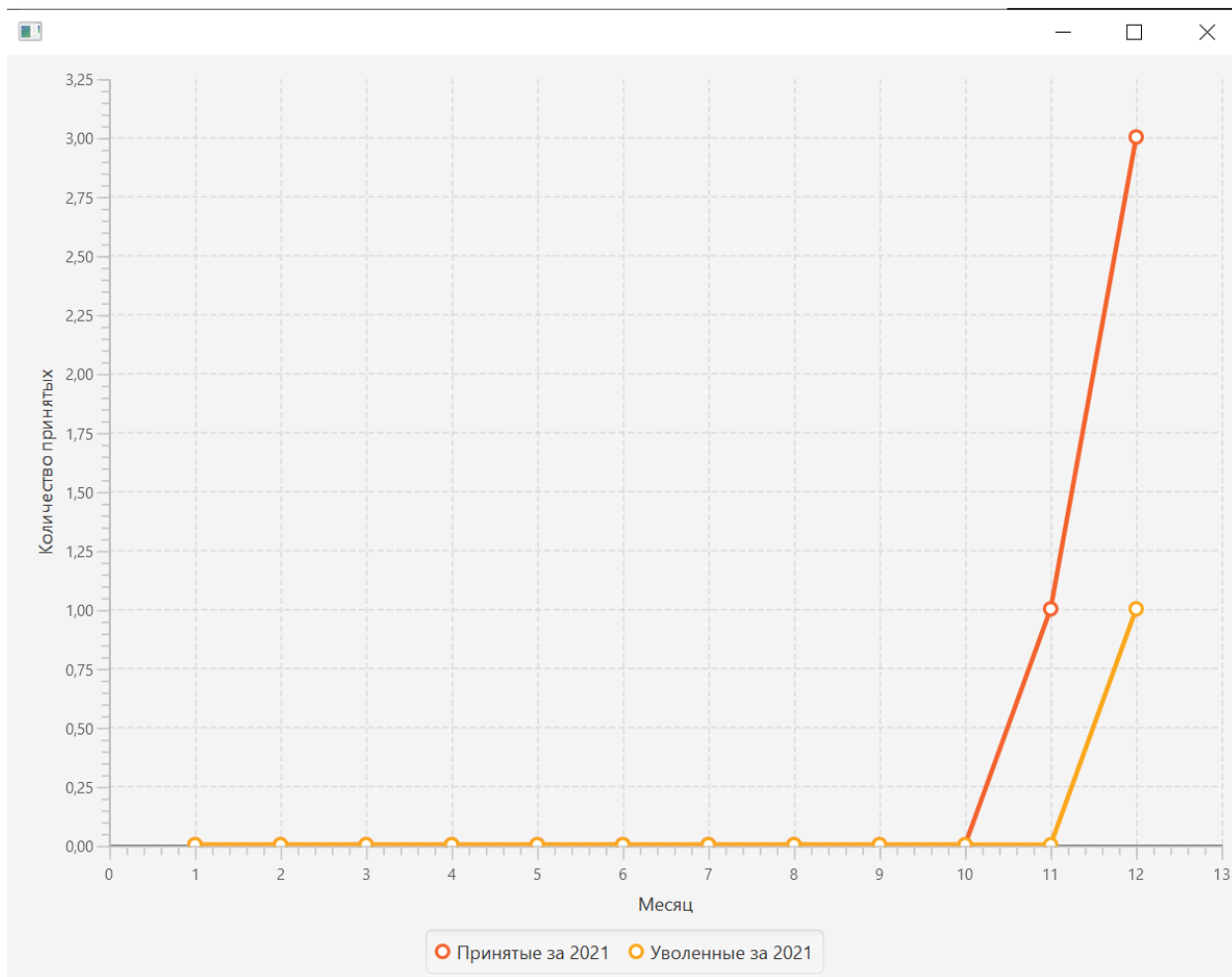


Рисунок 7.18 – Построенные графики

После перехода по вкладке «Настроить систему» пользователь перемещается на специальную страницу (см. рис. 7.19).

Назад

Настройка реквизитов организации

Введите название предприятия

Введите номер государственной регистрации

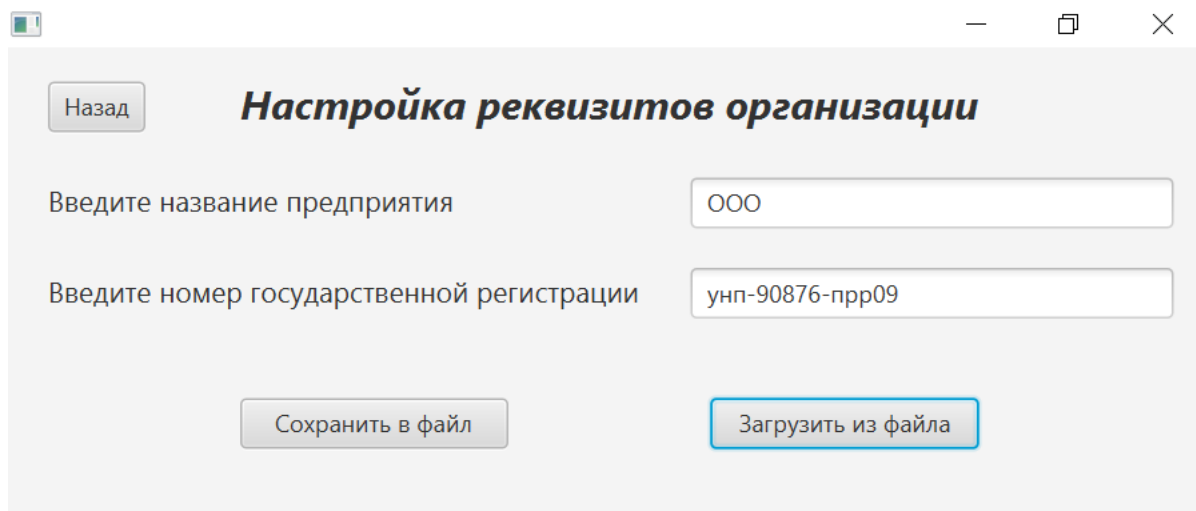
Сохранить в файл

Загрузить из файла

Рисунок 7.19 – Настройка системы

Здесь он может задать реквизиты организации путем ввода данных в поля название предприятия и номер государственной регистрации.

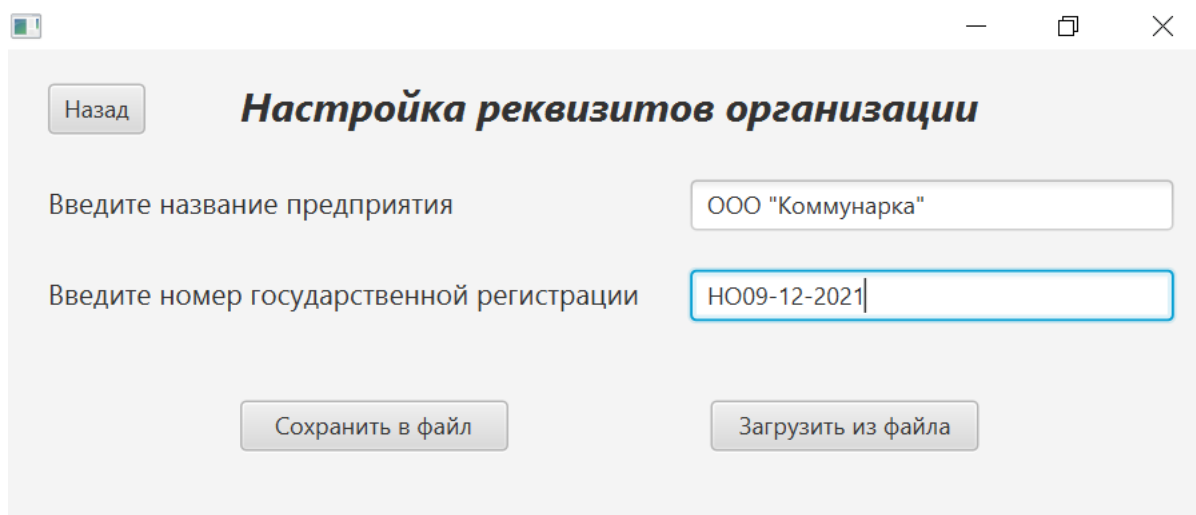
Далее пользователю дается возможность загрузить из файла существующие реквизиты организации, а именно название предприятия и номер государственной регистрации. Это создано для удобства пользователя при редактировании данных. Для этого необходимо нажать на кнопку «Загрузить из файла» (см. рис. 7.20)



The screenshot shows a window titled "Настройка реквизитов организации" (Organization Details Configuration). It has a "Назад" (Back) button in the top left. There are two input fields: "Введите название предприятия" (Enter company name) with the value "ООО" and "Введите номер государственной регистрации" (Enter state registration number) with the value "унп-90876-прр09". At the bottom, there are two buttons: "Сохранить в файл" (Save to file) and "Загрузить из файла" (Load from file), with the latter being highlighted with a blue border.

Рисунок 7.20 – Настройка реквизитов организации

Введенные пользователем новые данные можно сохранить в файл путем нажатия на кнопку «Сохранить в файл» (см. рис. 7.21).



The screenshot shows the same window as Figure 7.20, but with updated data. The "Введите название предприятия" field now contains "ООО 'Коммунарка'" and the "Введите номер государственной регистрации" field contains "НО09-12-2021". The "Сохранить в файл" button is now highlighted with a blue border, while the "Загрузить из файла" button is no longer highlighted.

Рисунок 7.21 – Изменение данных в файле

При этом на каждом шаге система информирует пользователя об успешно завершенной операции. Так, окно уведомлений появляется и на этапе загрузки данных из файла, и на этапе редактирования данных в файле.

8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ

Тестирование является неотъемлемой частью жизненного цикла программного обеспечения. Само по себе тестирование – длительный процесс проверок на соответствие ожидаемого результата. Нельзя выделить какой-то один этап как важный, каждый из них имеет одинаковый вес.

Важно понимать: невозможно найти все ошибки в продукте. Но и не найти ошибки при тестировании можно считать провалом. Главная цель – не сделать идеальный продукт без ошибок, а найти максимальное количество дефектов, которые потенциально могут сломать систему.

Существующие на сегодняшний день методы тестирования программного средства не позволяют однозначно и полностью выявить все дефекты и установить корректность функционирования анализируемой программы, поэтому все существующие методы тестирования действуют в рамках формального процесса проверки исследуемого или разрабатываемого программного средства.

Такой процесс формальной проверки или верификации может доказать, что дефекты отсутствуют с точки зрения используемого метода. То есть нет никакой возможности точно установить или гарантировать отсутствие дефектов в программном продукте с учетом человеческого фактора, присутствующего на всех этапах жизненного цикла программного средства.

Программное средство, разработанное в ходе выполнения данного курсового проекта, является настольным приложением. Для выявления некорректной работы тех либо иных элементов приложения было необходимо провести функциональное тестирование и тестирование пользовательского интерфейса на наличие визуальных ошибок. Учитывая небольшие масштабы выполняемого проекта, создание дополнительных программ для проведения автоматизированного тестирования нецелесообразно, поэтому был выбран мануальный (ручной) вариант тестирования.

Отдельно следует отметить, что приложение постоянно осуществляет обработку SQL запросов, поэтому необходимо было осуществить проверку на правильность выполнения запросов.

Итак, работа приложения начинается с запуска сервера, а затем клиента. При неправильной последовательности действий, то есть первоначальном запуске клиента, приложение информирует пользователя об ошибке с помощью специального окна. Кроме самого сообщения об ошибке пользователю также представляется причина возникновения ошибки, а именно «Сервер не запущен» (см. рис. 8.1).

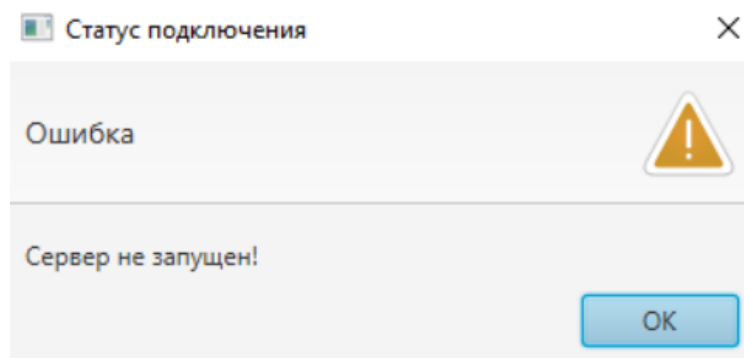


Рисунок 8.1 – Ошибка при незапущенном сервере

Далее пользователь совершает вход в систему. Если же в поля для ввода (логин и пароль) были введены неверные данные, система его информирует об ошибке, однако не с помощью диалогового окна, а в виде дополнительной строки текста на странице входа (см. рис. 8.2).

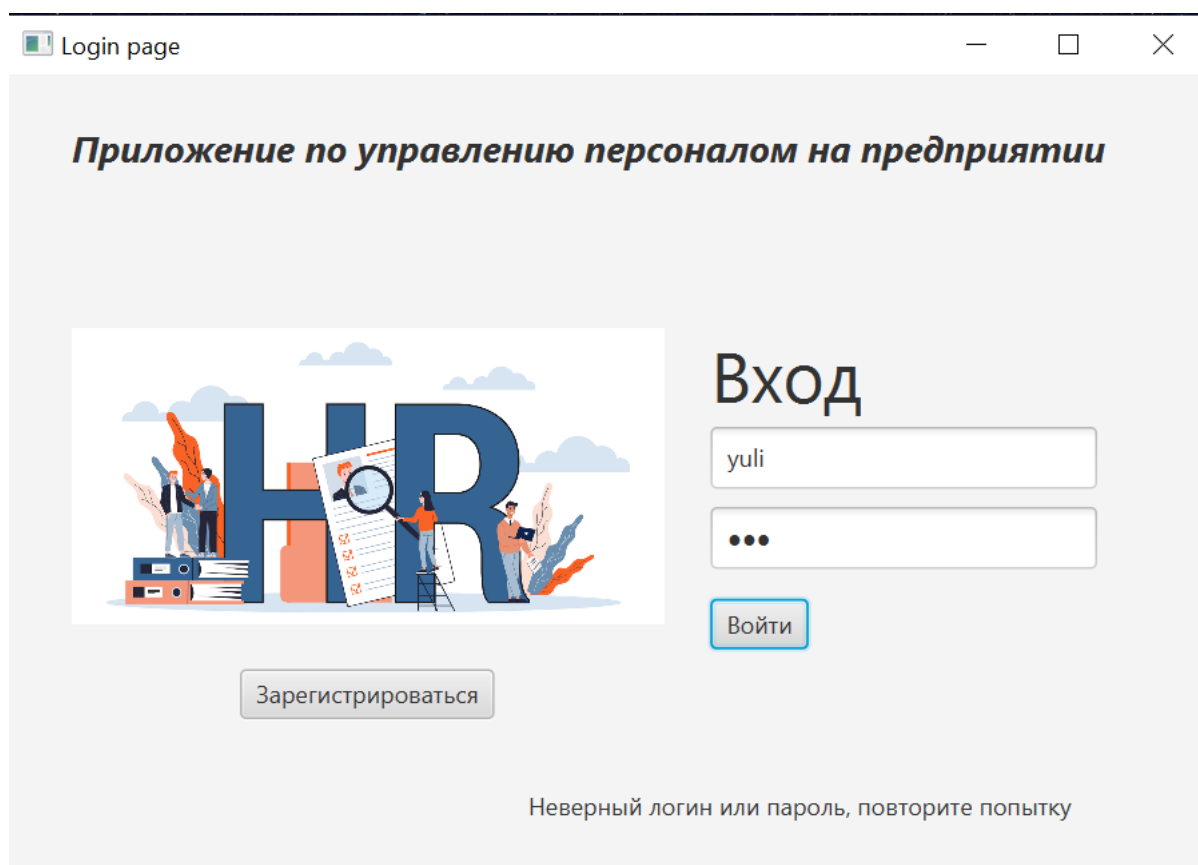


Рисунок 8.2 – Неверные данные при входе

После успешного входа на странице управления сотрудниками пользователь выполняет поиск необходимого сотрудника по фамилии либо по табельному номеру. При этом, если пользователь ввел пустое значение либо не выбрал критерий, система также информирует его об ошибке в виде появляющейся строки, объясняющей причину невыполнения поиска. Для удобства пользователя строка, информирующая об ошибке выделена красным цветом.

Результат поиска по пустому значению представлен на рисунке 8.3.

The screenshot shows a window titled "Управление кадрами". On the left, there is a search panel with a "Назад" button, a search bar containing "Поиск", and an "Ок" button. Below the search bar are two radio buttons: "по ФИО" (selected) and "по табельному номеру". Further down is a "Сортировать сотрудников" section with two radio buttons: "по алфавиту" and "по дате рождения", and an "Ок" button. A red error message "Строка поиска не может быть пустой!" is displayed. On the right, there is a table with columns: ID, Фамилия, Имя, Отчество, and Дата рождения. Above the table are buttons "Добавить", "Выбрать", and "Удалить". The table contains 10 rows of employee data.

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...
38	лорп	орп	орп	2021-12-03T0...
39	орп	орп	лор	2021-12-03T0...
40	ОРП	НПА	РПА	2021-12-01T0...
41	Иванова	Кира	Анатолевна	2002-12-07T0...

Рисунок 8.3 – Поиск по пустому значению

Результат поиска без выбранного критерия представлен на рисунке 8.4.

The screenshot shows the same "Управление кадрами" window. The search bar now contains the number "3". The error message "Выберите параметр поиска!" is displayed in red. The table of employees remains the same as in the previous screenshot.

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
31	Воронцов	Александр	Анатольевич	2021-12-02T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...
38	лорп	орп	орп	2021-12-03T0...
39	орп	орп	лор	2021-12-03T0...
40	ОРП	НПА	РПА	2021-12-01T0...
41	Иванова	Кира	Анатолевна	2002-12-07T0...

Рисунок 8.4 – Поиск без выбранного критерия

При осуществлении сортировки также возможен вариант, что пользователь не выбрал параметр сортировки. В этом случае пользователь получит

такое же уведомление об ошибке, как и при осуществлении поиска (см. рис. 8.5).

Управление кадрами

Назад

Поиск

☐ по ФИО
☐ по табельному номеру

Сортировать сотрудников

☐ по алфавиту
☐ по дате рождения

Выберите параметр сортировки!

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
31	Воронцов	Александр	Анатолевич	2021-12-02T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...
38	лорп	орп	орп	2021-12-03T0...
39	орп	орп	лор	2021-12-03T0...
40	ОРП	НПА	РПА	2021-12-01T0...
41	Иванова	Кира	Анатолевна	2002-12-07T0...

Рисунок 8.5 – Сортировка без выбранного параметра

Для просмотра полной информации о сотруднике предприятия необходимо сначала выбрать нужного сотрудника, а затем нажать на кнопку «Выбрать». Если же пользователь нажал на кнопку не выбрав сотрудника, он получит уведомление об ошибке и причине ее возникновения, расположенное слева и выделенное красным цветом (см. рис. 8.6).

Управление кадрами

Назад

Поиск

☐ по ФИО
☐ по табельному номеру

Сортировать сотрудников

☐ по алфавиту
☐ по дате рождения

Выберите работника!

ID	Фамилия	Имя	Отчество	Дата рождения
5	Воронова	Юлиана	Александровна	1998-08-08T0...
7	Лодис	Юлия	Сергеевна	2017-12-02T0...
31	Воронцов	Александр	Анатолевич	2021-12-02T0...
37	Васильев	Денис	Сергеевич	2002-03-15T0...
38	лорп	орп	орп	2021-12-03T0...
39	орп	орп	лор	2021-12-03T0...
40	ОРП	НПА	РПА	2021-12-01T0...
41	Иванова	Кира	Анатолевна	2002-12-07T0...

Рисунок 8.6 – Не выбран работник для просмотра профиля

Пользователь системы также может добавить нового сотрудника, нажав на кнопку «Добавить». При этом, если он не введет необходимую информацию о новом сотруднике, а сразу попытается добавить его, в ответ он получит окно, уведомляющее об ошибке (см. рис. 8.7).

The screenshot shows a web application window titled "Добавить сотрудника" (Add Employee). The form contains several input fields: "Фамилия" (Surname), "Имя" (Name), "Отчество" (Patronymic), "Дата рождения" (Date of birth), "Гражданство" (Citizenship), "Семейное положение" (Marital status), "Номер социального страхования" (Social security number), "Пол" (Gender) with radio buttons for "Женский" (Female) and "Мужской" (Male), "Подразделение" (Department) and "Должность" (Position) dropdown menus, "Дата приема" (Date of hire), and "Дата окончания контракта" (Contract end date). A "Добавить" (Add) button is at the bottom. An error dialog box titled "Статус редактирования" (Editing status) is overlaid on the form. It contains the text "Ошибка" (Error) and "Ошибка операции" (Operation error), a red "X" icon, and an "OK" button.

Рисунок 8.7 – Пустые поля при добавлении нового сотрудника

Следующая возможная ошибка может возникнуть на этапе формирования отчета. На этом этапе пользователь может забыть ввести дату для составления отчета. В таком случае на странице появится уведомление о необходимости выбрать дату (см. рис. 8.8).

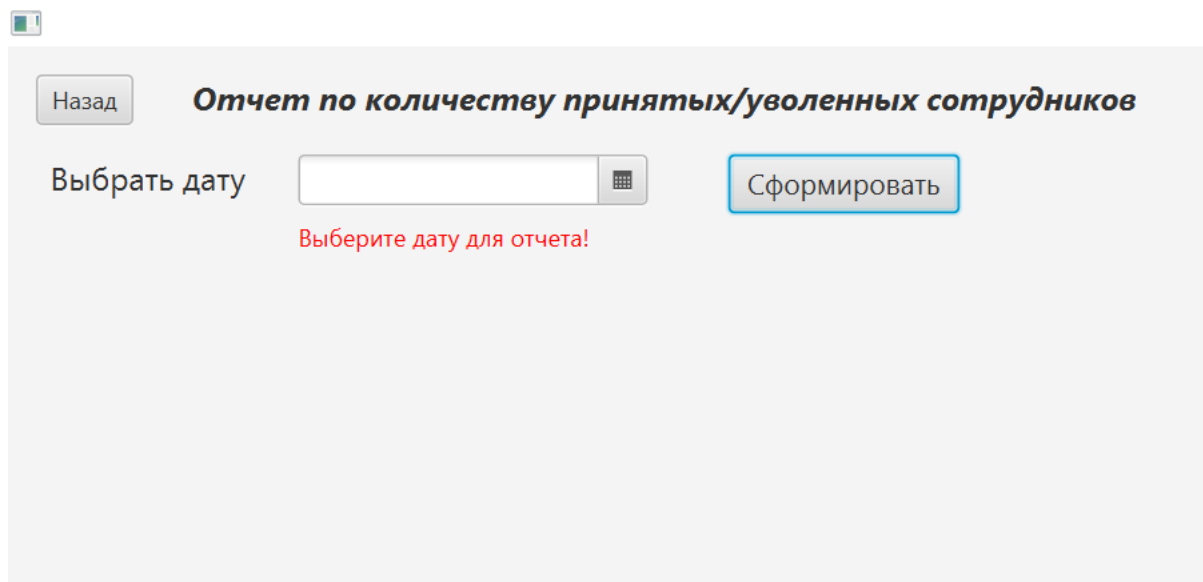


Рисунок 8.8 – Отсутствие даты для построения отчетов

Следует отметить, что окна уведомлений возникают не только по причине ошибки, совершенной пользователем, а также и в качестве уведомлений об успешном совершении действия. Пример такого окна, появляющегося при успешном удалении сотрудника, приведен на рисунке 8.9.

Это сделано для того, чтобы пользователь, совершая какие-либо действия в системе, был уверен в том, что действие совершилось и он не допустил никаких ошибок.

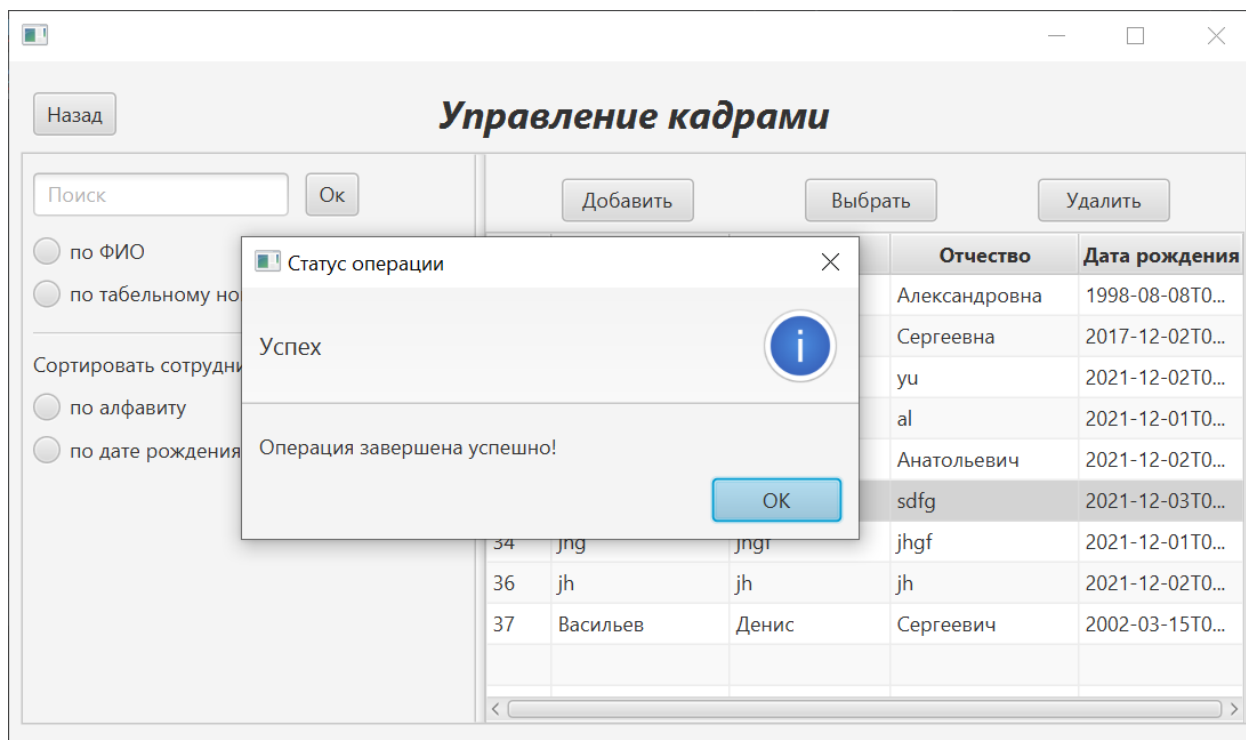


Рисунок 8.9 – Информирование об успешно завершенной операции

ЗАКЛЮЧЕНИЕ

В современном мире создание приложений для автоматизации процессов предприятия огромную роль. Эта необходимость связана с повышением производительности и эффективности работы предприятия.

Наблюдается тенденция к существенным переменам во всех областях, где присутствуют человеческие ресурсы. Руководители компаний стараются максимально оптимизировать все процессы и увеличить получаемую прибыль.

В результате разработки была создано программное средство по управлению кадровой политикой предприятия и программной поддержке HR-отдела.

Приложение разработано на объектно-ориентированном языке программирования Java. Для создания графического пользовательского интерфейса была использована библиотека JavaFX.

Данные, с которыми работает приложение, записываются и хранятся в базе данных, которая связана с СУБД MySQL с помощью менеджера баз данных JDBC.

Приложение является клиент-серверным и поддерживает многопоточность, поэтому несколько пользователей могут одновременно работать в приложении.

Реализованное программное средство соответствует всем поставленным всем поставленным задачам:

- 1 Изучена предметная область кадровой политики предприятия, управления персоналом и деятельности HR-службы.
- 2 Выделены требования к разрабатываемому программному средству.
- 3 Выбраны средства проектирования и разработки.
- 4 Разработан графический интерфейс приложения.
- 5 Создана база данных для хранения информации.
- 6 Разработано приложение
- 7 Описано руководство пользователя
- 8 Произведено тестирование приложения.

Также была построена информационная модель системы с использованием IDEF0, которая была декомпозирована до 4 уровня.

Кроме того, для понимания логической и физической модели программы были построены UML-диаграммы, а именно диаграмма компонентов, диаграмма развертывания, диаграмма классов, диаграмма вариантов использования, диаграмма состояний и диаграмма последовательности.

В результате разработанная программа по управлению кадрами на предприятии позволит оптимизировать процессы отдела кадров, а также повысить эффективность работы отдела кадров и предприятия в целом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- [1] MySQL [Электронный ресурс]. – Режим доступа: <https://metanit.com/sql/mysql>.
- [2] Руководство по языку программирования Java [Электронный ресурс]. – Режим доступа: <https://metanit.com/java/tutorial/>
- [3] Управление персоналом [Электронный ресурс]. – Режим доступа: https://www.audit-it.ru/terms/trud/upravlenie_personalom.html.
- [4] Unified Modeling Language [Электронный ресурс]. – Режим доступа: <https://www.visual-paradigm.com/guide/>.
- [5] Введение в MVC и HMVC [Электронный ресурс]. – Режим доступа: <https://ruseller.com/>
- [6] Основы баз данных [Электронный ресурс]. – Режим доступа: <https://www.site-do.ru/db/db1.php>
- [7] Проектирование базы данных [Электронный ресурс]. – Режим доступа: http://www.bseu.by/it/tohod/lekcii4_3.htm. – Дата доступа: 20.11.2021
- [8] Ивановская, Л.В. Управление персоналом: Теория и практика. Организация, нормирование и регламентация труда персонала: Учебнопрактическое пособие / Л.В. Ивановская. - М.: Проспект, 2013. - 64 с.
- [9] Знакомство с нотацией IDEF0 и пример использования [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/company/trinion/blog/322832/>.
- [10] Паттерны проектирования на Java [Электронный ресурс]. – Режим доступа: <https://refactoring.guru/ru/design-patterns/java/>.
- [11] Роберт Лафоре. Структуры данных и алгоритмы на Java. -- Питер, 2017. – 704 с.
- [12] Веснин, В.Р. Управление персоналом в схемах: Учебное пособие / В.Р. Веснин. - М.: Проспект, 2015. - 96 с.
- [13] Алавердов, А.Р. Управление персоналом: Учебное пособие / А.Р. Алавердов, Е.О. Куроедова, О.В. Нестерова. - М.: МФПУ Синергия, 2013. - 192.
- [14] Виды диаграмм и принципы их использования [Электронный ресурс]. – Режим доступа: https://flexberry.github.io/ru/fd_landing_page.html.
- [15] Голенищев, Э. П. Информационное обеспечение систем управления : справ. пособие / Э. П. Голенищев, И. В. Клименко – Ростов н/Д : «Феникс», 2003 – 352 с.
- [16] Enterprise Architect 14 User Guide [Электронный ресурс]. – Режим доступа: https://sparxsystems.com/enterprise_architect_user_guide/14.0/index/indy.html/

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг алгоритмов, реализующих бизнес-логику

```
chooseBtn.setOnAction(actionEvent -> {
    errorLabel.setText("");
    try
    {
        selectedUser=table.getSelectionModel().getSelectedItem();
        UserHolder holder = UserHolder.getInstance();
        User user = new User(selectedUser.get(0).getId(), se-
lectedUser.get(0).getFirstName(),selectedUser.get(0).getLastName(),
        selectedUser.get(0).getPatronymic(), se-
lectedUser.get(0).getBirthDay());
        holder.set(user);
        navigate("/sample/UI/employee.fxml");
    }
    catch (IndexOutOfBoundsException ex)
    {
        errorLabel.setText("Выберите работника!");
    }
});
```

@FXML

```
void initialize() throws IOException, SQLException {
    SocketHolder socketHolder = SocketHolder.getInstance();
    socket = socketHolder.get();
    out = new BufferedWriter(new OutputStreamWrit-
er(socket.getOutputStream()));
    in = new BufferedReader(new InputStreamRead-
er(socket.getInputStream()));
```

```

setData();
saveButtonEdit.setOnAction(actionEvent -> {
    User edited_user = getData();
    try {
        DataHandler.Serialize(edited_user);
        out.write("edit\n");
        out.flush();
        String status = in.readLine();
        ReactStatus.react(status);
    } catch (IOException e) {
        e.printStackTrace();
    }
    navigate("/sample/UI/hr_management.fxml");
});
backButtonEdit.setOnAction(actionEvent -> {
    navigate("/sample/UI/hr_management.fxml");
});
}

```


ПРИЛОЖЕНИЕ Б

(обязательное)

Листинг основных элементов программы

```
void initialize(){
    try {
        socket = new Socket(InetAddress.getByName("127.0.0.1"), 4004);
        SocketHolder socketHolder = SocketHolder.getInstance();
        socketHolder.set(socket);

        in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

        out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));

    } catch (IOException x) {
        System.out.println("Input/output error");
        x.printStackTrace();
    }

    signButton.setOnAction(actionEvent -> {
        login=loginEntry.getText().trim();
        password=passwordEntry.getText().trim();
        if (socket == null) {
            return;
        }
        try {
            out.write(login+" "+password + " signUp\n"); // отправляем сообщение
на сервер
            out.flush();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    });
}
```

```

    } finally {
        try {
            RoleHolder roleHolder = RoleHolder.getInstance();
            roleHolder.set("Администратор");
            signButton.getScene().getWindow().hide();
            navigate("/sample/UI/main_page.fxml");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
});
}

void initialize() {
    RoleHolder roleHolder = RoleHolder.getInstance();
    String currentUser = roleHolder.get();
    roleLabel.setText(currentUser);
    if (currentUser.equals("Пользователь"))
    {
        configureSystem.setVisible(false);
    }
    SimpleDateFormat formatter = new SimpleDateFormat("dd.MM.yyyy");
    Date date = new Date();
    currentDateLabel.setText(formatter.format(date));
    manageEmployees.setOnAction(actionEvent ->
    {
        navigate("/sample/UI/hr_management.fxml");
    });
}

```

```

buildReport.setOnAction(actionEvent -> {
    navigate("/sample/UI/report.fxml");
});
configureSystem.setOnAction(actionEvent ->
{
    navigate("/sample/UI/setting.fxml");
});
exit.setOnAction(actionEvent -> {
    SocketHolder socketHolder = SocketHolder.getInstance();
    Socket socket = socketHolder.get();
    if (socket == null) {
        return;
    }
    try {
        out = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
        out.write("stop\n"); // отправляем сообщение на сервер
        out.flush();
        SceneController sceneController = SceneController.getInstance();
        sceneController.getStage().close();
    }
    catch (IOException e) {
        e.printStackTrace();
    }
});
}

```

ПРИЛОЖЕНИЕ В

(обязательное)

Листинг скрипта генерации базы данных

```
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS,
UNIQUE_CHECKS=0;

SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0;

SET @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLES,NO_ZERO_IN_DATE,NO_ZERO_DATE,ERROR_FOR_DIVISION_BY_ZERO,NO_ENGINE_SUBSTITUTION';

CREATE SCHEMA IF NOT EXISTS `hrmanagerdb` DEFAULT CHARACTER
SET utf8mb4 COLLATE utf8mb4_0900_ai_ci ;

USE `hrmanagerdb` ;

CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`hrmanagers` (
  `Табельный_номер` INT NOT NULL AUTO_INCREMENT,
  `Роль` VARCHAR(45) NOT NULL,
  `Имя_пользователя` VARCHAR(45) NOT NULL,
  `Пароль` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`Табельный_номер`))
ENGINE = InnoDB
AUTO_INCREMENT = 5
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

INSERT INTO `hrmanagerdb`.`hrmanagers`
(`Роль`,
`Имя_пользователя`,
`Пароль`)
VALUES
```

```
('Администратор',  
  'yuli',  
  '1111');
```

```
CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`вид_образования` (  
  `КодВидаОбразования` INT NOT NULL AUTO_INCREMENT,  
  `ВидОбразования` VARCHAR(45) NULL DEFAULT NULL,  
  PRIMARY KEY (`КодВидаОбразования`))
```

```
ENGINE = InnoDB
```

```
AUTO_INCREMENT = 3
```

```
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;
```

```
INSERT INTO `hrmanagerdb`.`вид_образования`  
(`ВидОбразования`)
```

```
VALUES
```

```
('Очное'),
```

```
('Заочное');
```

```
CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`гражданство` (  
  `КодГражданства` INT NOT NULL AUTO_INCREMENT,  
  `Гражданство` VARCHAR(45) NULL DEFAULT NULL,  
  PRIMARY KEY (`КодГражданства`))
```

```
ENGINE = InnoDB
```

```
DEFAULT CHARACTER SET = utf8mb4
```

```
COLLATE = utf8mb4_0900_ai_ci;
```

```
INSERT INTO `hrmanagerdb`.`гражданство`  
(`Гражданство`)
```

```
VALUES
```

```
('Республика Беларусь'),
```

('Российская Федерация'),

('Украина'),

('Польша'),

('Латвия'),

('Литва');

CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`должности` (

 `КодДолжности` INT NOT NULL AUTO_INCREMENT,

 `Должность` VARCHAR(45) NULL DEFAULT NULL,

 PRIMARY KEY (`КодДолжности`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

INSERT INTO `hrmanagerdb`.`должности`

(`Должность`)

VALUES

('Директор по персоналу'),

('Специалист по кадрам'),

('Экономист'),

('Финансовый директор'),

('Продукт-менеджер'),

('Менеджер по проектам'),

('Java разработчик'),

('C# разработчик');

CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`форма_обучения` (

 `КодФормыОбучения` INT NOT NULL AUTO_INCREMENT,

 `ФормаОбучения` VARCHAR(45) NULL DEFAULT NULL,

 PRIMARY KEY (`КодФормыОбучения`))

```

ENGINE = InnoDB

AUTO_INCREMENT = 11

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4_0900_ai_ci;

INSERT INTO `hrmanagerdb`.`форма_обучения`
(`ФормаОбучения`)
VALUES
('Платная'),
('Бюджетная');

CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`образование` (
  `КодОбразования` INT NOT NULL AUTO_INCREMENT,
  `ДатаПоступления` DATETIME NULL DEFAULT NULL,
  `ДатаОкончания` DATETIME NULL DEFAULT NULL,
  `Специальность` VARCHAR(45) NULL DEFAULT NULL,
  `КодФормыОбучения` INT NULL DEFAULT NULL,
  `КодВидаОбразования` INT NULL DEFAULT NULL,
  PRIMARY KEY (`КодОбразования`),
  INDEX `кодФормыОбучения_INDEX` (`КодФормыОбучения` ASC)
  VISIBLE,
  INDEX `кодВидаОбразования_INDEX` (`КодВидаОбразования` ASC)
  VISIBLE,
  CONSTRAINT `ВидОбразованияFK`
    FOREIGN KEY (`КодВидаОбразования`)
    REFERENCES `hrmanagerdb`.`вид_образования` (`КодВидаОбразования`),
  CONSTRAINT `ФормаОбученияFK`
    FOREIGN KEY (`КодФормыОбучения`)
    REFERENCES `hrmanagerdb`.`форма_обучения` (`КодФормыОбучения`))
ENGINE = InnoDB

```

```

AUTO_INCREMENT = 3
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`подразделение` (
  `КодПодразделения` INT NOT NULL AUTO_INCREMENT,
  `Подразделение` VARCHAR(45) NULL DEFAULT NULL,
  PRIMARY KEY (`КодПодразделения`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;
INSERT INTO `hrmanagerdb`.`подразделение`
(`Подразделение`)
VALUES
('Отдел кадров'),
('Экономический отдел'),
('Финансовый отдел'),
('Отдел управления продуктом'),
('Отдел управления проектом'),
('Отдел разработки');
CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`семейное_положение` (
  `КодСемейногоПоложения` INT NOT NULL AUTO_INCREMENT,
  `СемейныйСтатус` VARCHAR(45) NULL DEFAULT NULL,
  `КоличествоДетей` INT NULL DEFAULT NULL,
  PRIMARY KEY (`КодСемейногоПоложения`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8mb4
COLLATE = utf8mb4_0900_ai_ci;

```



```

INSERT INTO `hrmanagerdb`.`семейное_положение`
(`СемейныйСтатус`,
`КоличествоДетей`)
VALUES
('Есть супруг(а)', 1),
('Есть супруг(а)', 2),
('Есть супруг(а)', 3),
('Есть супруг(а)', 4),
('Нет супруг(а)', 0),
('Нет супруг(а)', 1),
('Нет супруг(а)', 2),
('Нет супруг(а)', 3),
('Нет супруг(а)', 4),
CREATE TABLE IF NOT EXISTS `hrmanagerdb`.`работники` (
  `Табельный_номер` INT NOT NULL AUTO_INCREMENT,
  `Фамилия` VARCHAR(45) NULL DEFAULT NULL,
  `Имя` VARCHAR(45) NULL DEFAULT NULL,
  `Отчество` VARCHAR(45) NULL DEFAULT NULL,
  `КодГражданства` INT NULL DEFAULT NULL,
  `КодДолжности` INT NULL DEFAULT NULL,
  `КодСемейногоПоложения` INT NULL DEFAULT NULL,
  `КодПодразделения` INT NULL DEFAULT NULL,
  `ДатаПриема` DATETIME NULL DEFAULT NULL,
  `ДатаУвольнения` DATETIME NULL DEFAULT NULL,
  `Пол` VARCHAR(45) NULL DEFAULT NULL,
  `ДатаРождения` DATETIME NULL DEFAULT NULL,

```

```

`Фото` VARCHAR(45) NULL DEFAULT NULL,
`НомерСоциальногоСтрахования` INT NULL DEFAULT NULL,
`КодОбразования` INT NULL DEFAULT NULL,
PRIMARY KEY (`Табельный_номер`),
INDEX `КодОбразования_INDEX` (`КодОбразования` ASC) VISIBLE,
INDEX `КодПодразделения_INDEX` (`КодПодразделения` ASC) VISIBLE,
INDEX `КодСемейногоПоложения_INDEX` (`КодСемейногоПоложения`
ASC) VISIBLE,
INDEX `КодДолжности_INDEX` (`КодДолжности` ASC) VISIBLE,
INDEX `КодГражданства_INDEX` (`КодГражданства` ASC) VISIBLE,
CONSTRAINT `hrmanagersFK`
    FOREIGN KEY (`Табельный_номер`)
    REFERENCES `hrmanagerdb`.`hrmanagers` (`Табельный_номер`),
CONSTRAINT `ГражданствоFK`
    FOREIGN KEY (`КодГражданства`)
    REFERENCES `hrmanagerdb`.`гражданство` (`КодГражданства`),
CONSTRAINT `ДолжностиFK`
    FOREIGN KEY (`КодДолжности`)
    REFERENCES `hrmanagerdb`.`должности` (`КодДолжности`),
CONSTRAINT `ОбразованиеFK`
    FOREIGN KEY (`КодОбразования`)
    REFERENCES `hrmanagerdb`.`образование` (`КодОбразования`),
CONSTRAINT `ПодразделениеFK`
    FOREIGN KEY (`КодПодразделения`)
    REFERENCES `hrmanagerdb`.`подразделение` (`КодПодразделения`),
CONSTRAINT `СемейноеПоложениеFK`
    FOREIGN KEY (`КодСемейногоПоложения`)

```

```
REFERENCES `hrmanagerdb`.`семейное_положение`  
(`КодСемейногоПоложения`))  
  
ENGINE = InnoDB  
  
AUTO_INCREMENT = 3  
  
DEFAULT CHARACTER SET = utf8mb4  
  
COLLATE = utf8mb4_0900_ai_ci;  
  
SET SQL_MODE=@OLD_SQL_MODE;  
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;  
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```