МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение
высшего образования
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ
ГИДРОМЕТЕОРОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Экономики и управления на предприятии природопользования»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

(бакалаврская работа)
по направлению подготовки 09.03.03 Прикладная информатика
(квалификация – бакалавр)

На тему«Разработка информационной системы учета сотрудников на предприятии»

Исполнитель Симон Никита Витальевич

Руководитель ст. преподаватель, Сафонова Татьяна Владимировна

«К защите допускаю»

Руководитель кафедры _____

кандидат экономических наук

Майборода Е.В

«23» января 2024 г.

Филиал Российского государственного
гидрометеорологического университета в г. Туапсе
НОРМОКОНТРОЛЬ ПРОЙДЕН
«22» января 2024 г.
Марчинова Т.В.
подпись расшифровка подписи

Туапсе
2024

ОГЛАВЛЕНИЕ

Введение

Актуальность: Разработка информационной системы для учета сотрудников позволяет автоматизировать и оптимизировать процессы управления персоналом. Система может содержать базу данных с информацией о сотрудниках, их персональных данных, должностях, зарплатах и иных релевантных данных. Такая система позволит упростить процессы поиска, найма, увольнения и учета сотрудников, что приведет к оптимизации работы HR-отдела и улучшению общей эффективности работы предприятия.

Улучшение прозрачности и доступности информации: Информационная система для учета сотрудников сделает информацию о сотрудниках более доступной для всех заинтересованных сторон. Это позволит руководству, HR-отделу и другим сотрудникам быстро получать актуальные данные о сотрудниках, такие как их контактная информация, рабочий график, отпуска и больничные листы. Прозрачность и доступность информации способствуют более эффективному принятию управленческих решений и повышению коммуникации внутри предприятия.

Усиление безопасности данных: Информационная система для учета сотрудников также способствует повышению уровня безопасности информации о сотрудниках. Программные решения могут предоставлять функции защиты данных, такие как управление доступом и шифрование. Это позволяет предотвращать несанкционированный доступ к информации, сохраняя конфиденциальность и сохранность персональных данных сотрудников.

Упрощение аналитики и отчетности: Разработка информационной системы для учета сотрудников позволяет создавать различные статистические отчеты и аналитические данные. Это дает возможность для более точной и обоснованной аналитики данных о сотрудниках, таких как их производительность, обучение и требования к повышению квалификации. Отчеты, сгенерированные системой, могут быть полезными для HR-отдела и вышестоящего руководства при принятии стратегических решений, связанных

с управлением персоналом.

Таким образом, разработка информационной системы для учета сотрудников на предприятии является актуальной и важной темой, которая привносит множество преимуществ и оптимизирует процессы управления персоналом, прозрачность и доступность информации, безопасность и аналитику данных.

Объект исследования:Бизнес-процессы по реализации деятельности предприятия кадрового отдела.

Предмет исследования:Оптимизация и автоматизация деятельности предприятия кадрового отдела.

Цели и задачи работы:

Целью работы является разработка информационной системы для учета сотрудников на предприятии, которая позволит эффективно управлять кадровыми процессами и обеспечить максимально возможную автоматизацию учета персонала.

В процессе исследования и разработки информационной системы будут решаться следующие задачи:

- Изучить теоретические основы разработки информационных систем
- Провести анализ существующих информационных систем для учета сотрудников на предприятиях
- Выбор оптимальной архитектуры и структуры разрабатываемой системы.
- Разработка функциональных и нефункциональных требований к системе.
- Проектирование базы данных и интерфейса пользователя системы.

В результате выполнения данной работы ожидается получение полноценной информационной системы для учета сотрудников, которая будет способствовать оптимизации кадровых процессов, повышению оперативности обработки информации о персонале и улучшению управленческих решений в области кадрового делопроизводства предприятия.

Инструменты, используемые в процессе разработки:

Инструменты, используемые в процессе разработки стали такие как:Python3 a

так же библиотеки sqlite3, Tkinter,WEBBROWSER.

Python 3 – последняя версия одного из наиболее популярных языков программирования в мире.

Плюсы Python 3. Простота и понятность: Python имеет простой и понятный синтаксис, который делает его легким для изучения и использования, особенно для начинающих разработчиков.

Интеграция библиотек: Python имеет огромное количество сторонних библиотек и фреймворков, что делает его мощным инструментом для разработки приложений практически любого назначения.

Широкий спектр применения: Python используется в различных областях, таких как веб-разработка, анализ данных, научные вычисления, искусственный интеллект, разработка игр, автоматизация задач, разработка информационных систем и многое другое.

Используемы библиотеки:

1) sqlite3: Библиотека sqlite3 предоставляет простой и удобный способ взаимодействия с базами данных SQLite из приложений Python. Она позволяет выполнять SQL-запросы, работать с данными, создавать и изменять таблицы и многое другое.

2) tkinter: Tkinter – это стандартная библиотека для создания графического интерфейса пользователя в Python. Она обладает простым API и позволяет создавать разнообразные виджеты, кнопки, окна, меню и многое другое для создания интерактивных пользовательских приложений.

3) webbrowser: Модуль webbrowser предоставляет простой способ открывать веб-браузер и загружать веб-страницы из приложений Python. Это удобно для автоматизации открытия определенных URL-адресов или веб-страниц.

1 Теоретическая основа разработки информационной системы

1.1 Теоретические исследования в области разработки информационных систем

Теоретическая основа разработки информационной системы включает в себя ряд ключевых концепций и подходов, которые являются основой для разработки и реализации ИС. Вот несколько важных аспектов теоретической основы:

Разработка ИС должна основываться на системном подходе, который представляет систему как совокупность взаимосвязанных компонентов, взаимодействующих с окружающей средой. Этот подход позволяет анализировать целостную систему, ее элементы и взаимодействие между ними.

ИС часто включает в себя программное обеспечение. При разработке ПО применяется концепция жизненного цикла разработки, которая охватывает фазы, такие как анализ, проектирование, разработка, тестирование и внедрение [1, с.104].

Существует ряд методологий разработки ПО, таких как водопадная модель, гибкие методологии разработки (например, Scrum или Kanban) и спиральная модель разработки. Они предлагают структурированные подходы к организации и выполнению задач разработки ИС.

Моделирование процессов позволяет анализировать и оптимизировать рабочие процессы в ИС. Важные концепции в этой области включают диаграммы потоковданных (DFD), диаграммы активностей и моделирование бизнес-процессов.

ИС часто требуют управления большим объемом данных. Важным аспектом является разработка и использование баз данных, которые позволяют хранить, организовывать и обрабатывать данные эффективным способом, обеспечивая целостность и безопасность данных [1, с.156].

Защита информации в ИС является критическим аспектом. Теоретическая основа включает понимание принципов и методов обеспечения безопасности

информации, включая механизмы аутентификации, авторизации, шифрования и контроля доступа.

Разработка ИС часто включает управление проектами. Это включает в себя планирование, координацию ресурсов, управление временем и бюджетом для достижения поставленных целей проекта.

1.2Общий обзор различных информационных систем

Информационные системы (ИС) - это комплексное объединение аппаратных и программных средств, процессов и процедур, а также людей, предназначенных для сбора, хранения, обработки, анализа, передачи и использования информации в организации.

Компоненты информационных систем могут включать:

Аппаратные устройства: Это физическая инфраструктура, такая как компьютеры, серверы, сетевое оборудование, накопители данных и другие технические средства, которые обеспечивают функционирование информационной системы.

Базы данных: Они представляют собой организованный набор данных, которые используются для хранения информации и обеспечения ее доступности. Базы данных позволяют структурировать и организовать данные таким образом, чтобы они были эффективно хранены, управлялись и обрабатывались.

Сетевая инфраструктура: Это совокупность сетей, протоколов и коммуникационного оборудования, которые обеспечивают связь и передачу данных между различными устройствами в информационной системе. Сетевая инфраструктура может включать локальные сети, глобальные сети, облачные сервисы и другие формы связи [21].

Процессы и процедуры: Это описание действий, которые должны быть выполнены для сбора, обработки и использования информации. Процессы и процедуры могут быть формализованы в виде бизнес-процессов или

руководящих принципов, которые определяют последовательность действий и ролей сотрудников в информационной системе.

Люди: Люди - это пользователи и специалисты, занимающиеся управлением и использованием информационной системы. Они выполняют различные роли, от администраторов системы и аналитиков до конечных пользователей, и взаимодействуют с другими компонентами системы для достижения определенных целей.

Информационные системы могут применяться в различных сферах деятельности, от бизнеса и финансов до здравоохранения и образования. Применение информационных систем позволяет автоматизировать бизнес-процессы, улучшить доступность и надежность информации, улучшить принятие решений и обеспечить конкурентное преимущество. Однако разработка, внедрение и управление информационными системами также представляют собой сложные задач.

Существует множество различных систем информационных систем для учета сотрудников, каждая из которых обладает своими особенностями и функциональными возможностями. Общий обзор сравнения в виде таблицы (таблица 1.1):

1) Система учета персонала (HRMS): Это полнофункциональная информационная система для управления и учета сотрудников. Она включает в себя функции, такие как учет рабочего времени, оплату труда, кадровый учет, набор и отбор персонала, подготовку отчетов и аналитику. HRMS-системы обычно предлагают широкий набор инструментов для автоматизации и оптимизации процессов управления персоналом.

2) Система учета рабочего времени (TimeandAttendanceSystem): Это система, предназначенная для отслеживания и учета рабочего времени сотрудников. Она может включать функции, такие как регистрация прихода/ухода, отчеты о рабочем времени, учет отпусков и больничных, расчет заработной платы на основе отработанных часов и другие схожие функции.

3) Система управления зарплатой и выплатами (PayrollSystem): Эта

система предназначена для автоматизации расчета и выплаты заработной платы сотрудникам. Она может включать в себя функции расчета налогов и социальных отчислений, формирование финансовых отчетов и документов, связанных с выплатами, обработку запросов о выплатах и т. д.

Таблица – 1.1 Сравнение информационных систем

Системы	Функции	Ключевые функции	Достоинства	Интерграция	Подходит для	Примеры программного обеспечения/систем
Time and Attendance System	Отслеживает и регистрирует посещаемость сотрудников и отработанное время	Учет рабочего времени, включение / выключение часов, управление расписанием, расчет сверхурочных	Точный учет рабочего времени, мониторинг производительности, управление отсутствием	Может быть интегрирован с HRMS или системой расчета заработной платы для беспрепятственного обмена данными	Организации, которым необходимо отслеживать посещаемость сотрудников и управлять процессами, связанными со временем	ADP, Kronos, TSheets
HRMS	Управляет различными кадровыми функциями, такими как информирование сотрудников, набор персонала, оценка эффективности, обучение и т.д	Информация о сотрудниках, адаптация, оценка эффективности, управление обучением, управление отпусками	Оптимизированные кадровые процессы, улучшенное управление персоналом, повышенное соответствие требованиям	Может интегрироваться с другими системами, такими как учет рабочего времени и посещаемости или контроль доступа	Организации, которым требуется комплексное решение для управления кадровыми процессами	SAP SuccessFactors, Oracle HCM Cloud
Payroll System	Рассчитывает заработную плату сотрудников и управляет процессами расчета заработной платы	Расчет заработной платы, налоговые вычеты, прямой депозит, формирование отчетов	Эффективное управление заработной платой, точный расчет заработной платы, упрощенные налоговые вычеты	Может интегрироваться с HRMS или системой учета рабочего времени и посещаемости	Организации, которым необходимо автоматизировать расчеты заработной платы и управлять компенсациями	QuickBooks Payroll, ADP Workforce Now, Paychex
Human Resource Management System	Комплексная система, интегрирующая различные кадровые функции и процессы	Управление данными сотрудников, портал самообслуживания сотрудников, аналитика	Централизованные кадровые данные, улучшенное принятие кадровых решений, расширенный опыт сотрудников	Может интегрироваться с другими кадровыми системами для обмена данными	Организации, ищущие комплексное решение для всех кадровых функций	SAP ERP HCM, BambooHR, Namely
Access Control and Security System	Контролирует доступ к физическим зонам и обеспечивает их безопасность	Карты доступа, биометрические системы, управление посетителями,	Повышенная безопасность, ограниченный доступ к чувствительным зонам,	Может интегрироваться с другими системами, такими как HRMS или учет	Организации, требующие усиленных мер контроля доступа и	HID Global, Honeywell Access Control, Avigilon

4) Система управления персоналом (HumanResourceManagementSystem): Эта система объединяет различные аспекты управления персоналом, включая учет, набор и отбор, развитие и обучение, учет рабочего времени и т. д. Она

может предлагать различные инструменты для управления персоналом, чтобы обеспечить эффективное функционирование и развитие организации.

5) Система учета доступа и безопасности (AccessControlandSecuritySystem): Система обеспечивает контроль доступа и безопасность сотрудников на рабочем месте. Она может включать в себя механизмы идентификации, такие как биометрические сканеры, карточки доступа, системы видеонаблюдения и другие средства для обеспечения безопасности и контроля доступа [2, с.184].

1.3Анализ функциональности

Функциональность информационной системы для учета сотрудников обычно включает в себя следующие основные элементы:

1) Регистрация и управление профилем сотрудника. Создание и регистрация нового профиля сотрудника с вводом базовой информации, такой как ФИО, контактные данные, должность, отдел и т.д.

Возможность обновления профиля сотрудника при изменении информации, такой как адрес, контактная информация, статус занятости и т.д.

Аутентификация сотрудников для обеспечения безопасного доступа к системе учета.

2) Управление рабочими часами и отпусками. Ввод и отслеживание рабочего времени сотрудников, включая начало и конец рабочего дня, перерывы и выходные.

Учет и управление отпусками сотрудников, включая возможность запроса отпуска, утверждения, отслеживание оставшегося отпуска и т.д.

3) Кадровые процессы и управление штатом. Отслеживание и управление информацией о всех сотрудниках, включая персональные данные, профессиональный опыт, образование, квалификацию и т.д.

Управление процессами найма и увольнения сотрудников, включая создание вакансий, обзор резюме, проведение собеседований, заявление об

увольнении и т.д. [20].

Учет и управление организационной структурой, включая разделение на отделы и команды, назначение руководителей и подчиненных.

4) Вознаграждения и выгоды для сотрудников. Учет и расчет заработной платы для сотрудников, включая основную заработную плату, выплаты бонусов и премий, удержания и т.д.

Учет и управление дополнительными выгодами и льготами, такими как медицинская страховка, пенсионные планы, корпоративные скидки и т.д.

5) Отчетность и аналитика. Генерация отчетов и аналитика связанных с данными о сотрудниках, таких как статистика рабочего времени, использование отпусков, структура штата, среднее время работы и т.д.

Создание настраиваемых отчетов для поддержки различных требований и нужд компании, таких как аудит, планирование бюджета, мониторинг производительности и т.д.

6) Безопасность и защита данных. Обеспечение конфиденциальности и безопасности данных сотрудников с помощью подходящих мер безопасности, таких как шифрование, доступ по ролям и правам, резервное копирование данных и т.д.

Соблюдение требований законодательства о защите персональных данных и конфиденциальной информации.

7) Интеграция с другими системами. Если компания использует другие информационные системы, такие как система учета клиентов или система управления проектами, важно, чтобы система учета сотрудников могла интегрироваться с ними и обмениваться данными [3, с.124].

1.4Преимущества и недостатки существующих решений

Преимущества:

1) Увеличение эффективности бизнес-процессов: Информационные системы позволяют автоматизировать многие рутинные задачи и процессы, что

повышает эффективность бизнес-процессов. Это может привести к сокращению времени выполнения задач, повышению производительности и улучшению качества работы.

2) Улучшение принятия решений: Информационные системы обеспечивают централизованный доступ к данным, что позволяет представлять информацию в удобной форме для анализа и принятия решений. Это приводит к более точным и обоснованным бизнес-решениям.

3) Рост конкурентоспособности: Использование информационных систем может усилить конкурентные позиции компании путем улучшения клиентского опыта, оптимизации процессов и повышения качества продукции.

4) Улучшение взаимодействия и коммуникации: Информационные системы позволяют улучшить внутренние коммуникации и взаимодействие между сотрудниками, что способствует росту производительности и снижению ошибок [22].

Недостатки:

1) Высокие затраты на внедрение: внедрение информационных систем часто связано с большими финансовыми затратами, особенно при необходимости разработки или приобретения специализированного программного обеспечения.

2) Сложности интеграции: информационные системы могут столкнуться с проблемами интеграции с существующими рабочими процессами, техническими системами и другими интерфейсами.

3) Управление изменениями и обучение персонала: внедрение новых информационных систем может потребовать значительной обучения и адаптации персонала, а также изменений в корпоративной культуре.

4) Уязвимость к кибератакам: информационные системы подвержены киберугрозам, таким как вирусы, хакерские атаки и утечки данных. Это требует дополнительных усилий в обеспечении безопасности и конфиденциальности.

5) Ограничения в аналитике и обработке больших данных: некоторые информационные системы могут сталкиваться с ограничениями в

аналитических возможностях и обработке больших объемов данных из-за технических ограничений или недостаточной масштабируемости.

6) Потенциальные ограничения производительности: некоторые информационные системы могут столкнуться с ограничениями в производительности при работе с большим объемом данных или при выполнении сложных вычислительных операций. Это может привести к замедлению работы системы и необходимости дополнительной оптимизации.

7) Сложности в масштабировании: при увеличении общего объема работ и бизнес-процессов организации, некоторые информационные системы могут столкнуться с проблемами в масштабировании, что потребует дополнительных усилий по расширению и оптимизации системы.

8)Ограничения в пользовательской поддержке: некоторые информационные системы могут иметь ограниченные возможности в поддержке пользователей, особенно в случае сложных или индивидуализированных запросов. Это может создавать трудности для пользователей в решении проблем и вопросов, связанных с использованием системы.

9) Трудности передачи и управления данными: некоторые информационные системы могут столкнуться с трудностями в передаче и управлении данными, особенно в случае их большого объема и необходимости обмена между различными системами и платформами.

10) Возможные недостатки в аналитике и представлении данных: отдельные информационные системы могут оказаться недостаточно гибкими в аналитике и представлении данных, что может затруднять процесс принятия решений и эффективное использование информации.

Поэтому, при выборе и использовании информационных систем важно учитывать все вышеперечисленные аспекты, чтобы эффективно использовать их преимущества и смягчать существующие недостатки [23].

Преимущества существующих решений:

1) TimeandAttendanceSystem (Система учета рабочего времени):

– Автоматизация процесса учета рабочего времени сотрудников, что упрощает и ускоряет процесс расчета заработной платы.

– Возможность отслеживания рабочего времени сотрудников в режиме реального времени, что позволяет контролировать и оптимизировать их производительность.

– Сокращение риска ошибок и мошенничества, связанных с учетом рабочего времени.

2) HRMS (Система управления человеческими ресурсами):

– Централизованное хранение и управление информацией о сотрудниках, включая персональные данные, контактную информацию, информацию о занятости и т. д.

– Автоматизация процессов рекрутинга, подбора персонала, управления производительностью, обучения и развития сотрудников.

– Упрощение административных задач, связанных с управлением человеческими ресурсами, и улучшение эффективности работы HR-отдела.

3) PayrollSystem (Система расчета заработной платы):

– Автоматизация процесса расчета и выплаты заработной платы сотрудникам, что уменьшает риск ошибок и позволяет сэкономить время и усилия.

– Соблюдение законодательства и правил расчета заработной платы, так как система обычно учитывает налоги, страховые взносы и другие факторы при расчете зарплаты.

– Предоставление сотрудникам доступа к электронным платежным ставкам, выплатам и налоговым формам, упрощает процесс управления финансами и повышает прозрачность.

4) HumanResourceManagementSystem (Система управления человеческими ресурсами):

– Обеспечение более эффективного управления рекрутингом, наймом, управлением производительностью и развитием сотрудников.

– Централизованное хранение и управление информацией о

сотрудниках, что облегчает процессы управления персоналом.

– Улучшение коммуникации и сотрудничества между HR-отделом, руководством компании и сотрудниками.

5) AccessControlandSecuritySystem (Система контроля доступа и безопасности):

– Защита важных областей и ресурсов компании от несанкционированного доступа.

– Мониторинг и запись активности пользователей, обеспечивающий безопасность и ответственность.

– Упрощение и автоматизация процесса контроля доступа, что увеличивает безопасность и удобство использования.

Недостатки существующих решений:

1) TimeandAttendanceSystem:

– Высокие затраты на внедрение и обслуживание системы.

– Необходимость обновления и модернизации системы для поддержки новых технологий и требований.

2) HRMS:

– Сложность внедрения и конфигурации системы, особенно для крупных компаний со сложными структурами и процессами управления персоналом.

– Необходимость поддержки системы IT-специалистами или внешними поставщиками услуг.

3) PayrollSystem:

– Возможность ошибок и некорректных расчетов в случае неправильной конфигурации или неправильного ввода данных.

– Зависимость от точности и своевременности предоставления информации.

4) HumanResourceManagementSystem:

– Сложность внедрения и адаптации системы, особенно для компаний с устаревшими системами управления HR.

– Необходимость постоянного обновления и поддержки системы.

5) Access Control and Security System:

– Высокие затраты на внедрение и обслуживание системы.

– Возможность сбоев или проблем совместимости, что может создать уязвимости в безопасности [5, с.304].

1.5 Принципы и методы учета сотрудников

Идентификация сотрудников: один из основных принципов учета сотрудников - это уникальная идентификация каждого сотрудника в информационной системе. Первоначально каждому сотруднику назначается уникальный идентификатор, который может быть номером сотрудника или другим идентификационным кодом. Этот идентификатор помогает однозначно определить каждого сотрудника в системе учета [24].

Регистрация присутствия: для учета присутствия сотрудников используются различные методы. Это может включать использование бейджей или карт доступа, которые сотрудники используют для регистрации своего входа и выхода с рабочего места. Также распространены биометрические считыватели, такие как сканеры отпечатков пальцев или системы распознавания лица, которые позволяют идентифицировать сотрудника и регистрировать его присутствие.

Учет рабочего времени: системы учета времени позволяют отслеживать время, проведенное сотрудниками на работе. Это включает в себя регистрацию начала и окончания рабочего дня, а также перерывов и отсутствия. Виды учета времени могут варьироваться от ручного ввода данных в электронные журналы до использования автоматических систем, которые определяют время входа и выхода сотрудников.

Учет производительности: отслеживание и учет производительности сотрудников является важным аспектом управления персоналом. Системы учета могут включать оценку выполнения задач, сбор данных о

продуктивности, качестве работы и других показателях производительности. Эти данные могут использоваться для оценки работы сотрудников, выявления недостатков и улучшения производительности.

Управление персоналом: информационные системы учета сотрудников также могут включать функции управления персонал

Автоматизация отчетности: информационные системы учета сотрудников позволяют автоматизировать процесс создания и генерации различных отчетов. Они могут быть предназначены для составления отчетов о присутствии, рабочем времени, производительности, использовании отпусков и других аспектах учета сотрудников. Автоматизация отчетности упрощает процесс анализа и мониторинга данных по сотрудникам, а также обеспечивает своевременную и точную информацию для принятия управленческих решений.

Планирование персонала: системы учета сотрудников также могут включать функции планирования персонала. Они позволяют определять потребности в кадрах, составлять графики работы, учет отпусков и больничных, а также прогнозировать возможные нехватки или перегрузки персонала. Это помогает организации эффективно распределять ресурсы и обеспечивать оптимальную рабочую силу.

Безопасность данных: принцип безопасности данных является критическим при учете сотрудников. Информационные системы должны обеспечивать защиту конфиденциальности и целостности данных о сотрудниках. Это может включать ограничение доступа к информации только уполномоченными лицами, использование шифрования данных и резервное копирование для предотвращения потери информации.

Соблюдение законодательства: при учете сотрудников необходимо соблюдать соответствующее законодательство, включая правила и нормы охраны труда, приватности данных и дискриминации. Информационные системы должны быть настроены таким образом, чтобы соответствовать требованиям законодательства и международным стандартам в области учета персонала [6, с.54].

Рисунок 1.1 - Схема последовательности основных компонентов метода учета сотрудников

На схеме (рисунок 1.1) показана последовательность основных компонентов метода учета сотрудников. На первом этапе происходит идентификация сотрудников с помощью уникальных идентификаторов. Затем идет регистрация присутствия, где используются средства, такие как бейджи или карты доступа, для регистрации входа и выхода сотрудников. После этого происходит учет рабочего времени, включая запись времени работы, перерывов и отсутствия сотрудников.

Далее идет учет производительности сотрудников, где осуществляется оценка выполнения задач и производительности. Затем идет управление персоналом, включающее ведение записей о подразделениях, должностях,

квалификации и отпусках сотрудников. Наконец, осуществляется автоматизация отчетности для генерации отчетов о присутствии, времени работы и производительности сотрудников.

Вся система учета сотрудников основана на безопасности данных и соблюдении соответствующего законодательства. Важными аспектами также являются обучение и развитие сотрудников, учет продуктивности и выхода сотрудников, а также интеграция с другими системами организации. Непрерывное улучшение является неотъемлемой частью метода учета сотрудников, позволяя постоянно повышать эффективность системы и улучшать процессы управления персоналом.

Точность и надежность: автоматизированный учет сотрудников позволяет снизить вероятность ошибок в данных и повысить точность расчетов.

Эффективность: автоматический учет сотрудников позволяет автоматизировать процессы, ускоряя время выполнения задачи и улучшая общую эффективность предприятия.

Легкость использования: система учета может быть разработана с учетом удобства использования и простоты интерфейса, что облегчает процесс обучения персонала.

Аналитические возможности: система автоматизированного учета может предоставить различные аналитические отчеты и статистику, помогая принимать более обоснованные решения в области управления персоналом [7, с.209].

2 Выбор оптимальных технологий, языка программирования и архитектурной модели

2.1 SWOT анализ учета сотрудников

SWOT-анализ (анализ сильных и слабых сторон, возможностей и угроз) — метод стратегического анализа, который используется для оценки текущего положения компании или организации. Он основывается на идее того, что для

разработки эффективной стратегии необходимо учитывать внутренние сильные и слабые стороны компании, а также внешние возможности и угрозы, с которыми она сталкивается.

SWOT-анализ помогает идентифицировать факторы, которые могут повлиять на успешность компании или организации. Этот анализ обычно представляется в виде матрицы, разделенной на 4 квадранта: сильные стороны (strengths), слабые стороны (weaknesses), возможности (opportunities) и угрозы (threats).

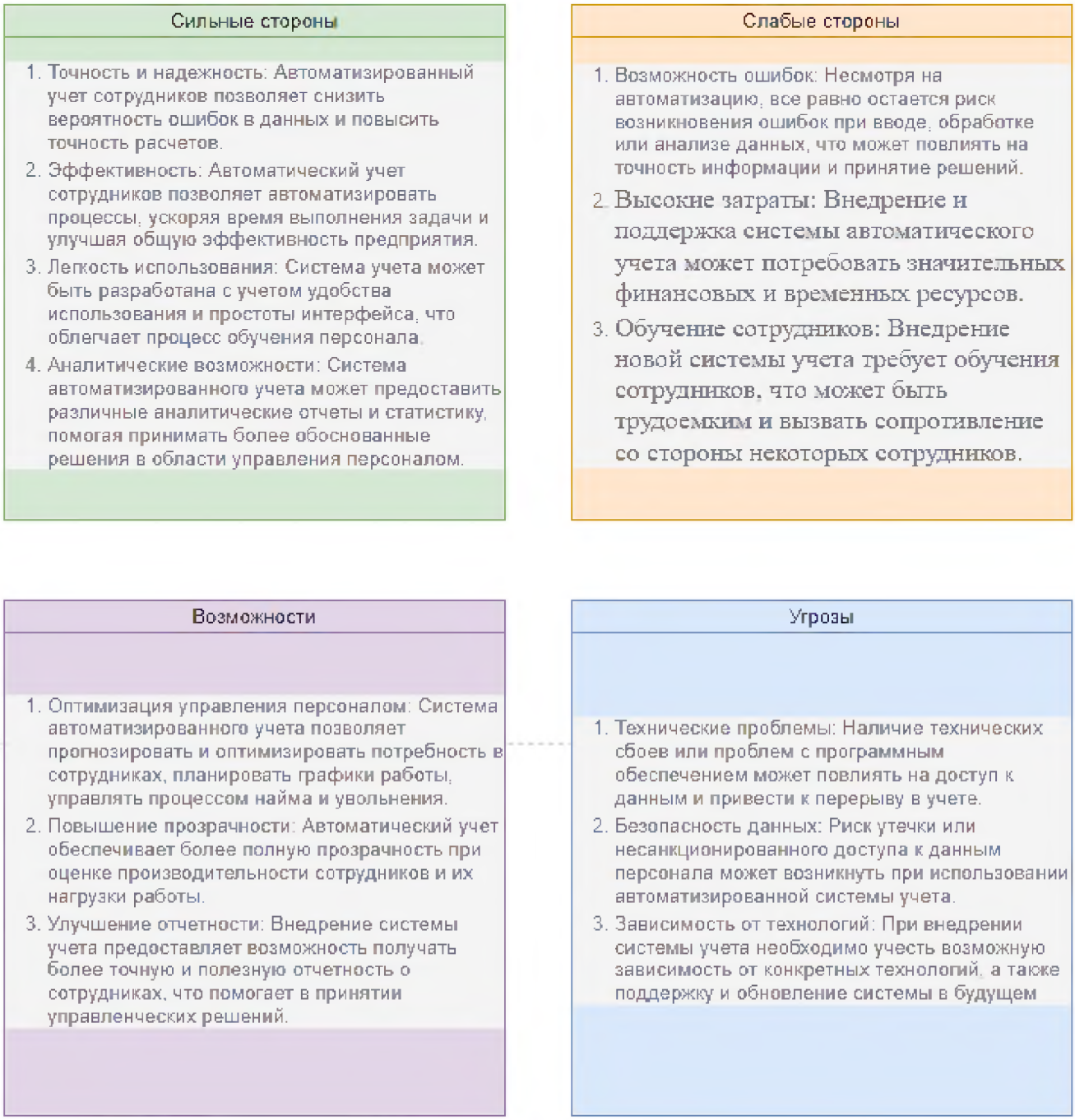


Рисунок 2.1 - SWOTанализ учета сотрудников на предприятии

На рисунке 2.1 собран SWOT-анализ учета сотрудников на предприятии.

Сильные стороны компании включают в себя ее преимущества, уникальные ресурсы и навыки, которые помогают ей отличаться от конкурентов. Слабые стороны указывают на те аспекты, где компания нуждается в улучшении или развитии.

Возможности отражают внешние факторы, которые могут создать новые возможности для роста и развития компании. Угрозы указывают на внешние факторы, которые могут негативно повлиять на компанию, ее конкурентоспособность или успех [25].

SWOT-анализ может быть полезным инструментом, который помогает компании понять свои преимущества и слабости, а также определить наиболее перспективные возможности и пути их реализации, а также предотвратить и справиться с возможными угрозами [9, с.123].

2.2 Выбор оптимальных технологий

Выбор оптимальных технологий для разработки информационной системы зависит от различных факторов, таких как требования к функциональности и производительности, доступные ресурсы (бюджет, персонал), предполагаемая масштабируемость и интеграция. Однако, есть несколько ключевых технологий, которые часто используются при разработке информационных систем:

Языки программирования: выбор языка зависит от множества факторов. Популярные языки, такие как Java, Python, Ruby и C#, являются надежными вариантами и широко используются в разработке информационных систем.

Веб-фреймворки: для разработки веб-приложений можно использовать фреймворки, такие как Django (Python), RubyonRails (Ruby), ASP.NET (C#), Spring (Java) и Laravel (PHP). Они предоставляют готовые компоненты и инструменты для ускорения разработки [26].

Базы данных: выбор базы данных также зависит от требований к системе.

Популярные реляционные базы данных, такие как MySQL, PostgreSQL и Oracle, часто используются для информационных систем.

Интеграция с внешними системами: для интеграции с внешними системами можно использовать API и протоколы взаимодействия, такие как REST, SOAP или GraphQL.

DevOps инструменты: для автоматизации процессов разработки, развертывания и поддержки информационной системы можно использовать инструменты DevOps, такие как Docker, Kubernetes, Jenkins и Ansible.

Важно учесть, что выбор оптимальных технологий может изменяться в зависимости от конкретного проекта и его требований [10, с.92].

2.3 Выбор языка программирования и инструментов

Для разработки информационной системы можно выбрать различные языки программирования и инструменты в зависимости от требований и целей проекта.

Java: Java - мощный и универсальный язык программирования. Он широко используется для разработки корпоративных приложений и масштабных информационных систем.

Имеет обширную библиотеку классов и фреймворков, таких как Spring, Hibernate, JavaFX, которые облегчают разработку и повышают производительность.

Python: Python - простой, понятный и эффективный язык программирования. Он обладает широким спектром библиотек и фреймворков, таких как Django, Flask, NumPy, Pandas, которые облегчают разработку информационных систем. Python также популярен в области анализа данных и машинного обучения [30].

JavaScript: JavaScript - мощный язык программирования, который широко используется для разработки интерактивных пользовательских интерфейсов и веб-приложений.

Благодаря фреймворкам и библиотекам, таким как React, Angular, Vue.js, разработка информационных систем на JavaScript становится проще и эффективнее.

C#: C# - объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он популярен для разработки приложений под платформу Windows и интеграцией с различными Microsoft-технологиями, такими как .NET Framework и ASP.NET. Для разработки информационных систем на C# есть множество фреймворков и инструментов, например, ASP.NET Core.

Так же, в таблице 2.1 сделано сравнение языков, по следующим критериям: сильные стороны , слабые стороны , функционал , доступность для пользователя и возможности.

Кроме выбора языка программирования, также важно определить инструменты для разработки информационной системы:

Интегрированная среда разработки (IDE): Популярные IDE включают VisualStudio (для разработки на C#), Eclipse или IntelliJ IDEA (для разработки на Java), PyCharm или JupyterNotebook (для разработки на Python), WebStorm (для разработки на JavaScript).

База данных: Выбор базы данных зависит от требований и сценариев использования информационной системы. Некоторые популярные базы данных включают MySQL, PostgreSQL, Oracle, MongoDB.

Фреймворки и библиотеки: Использование фреймворков (например, Spring, Django, React) и библиотек (например, jQuery, Bootstrap) поможет ускорить разработку и повысить эффективность.

Средства контроля версий: Для управления кодовой базой и совместной работы разработчиков рекомендуется использовать системы контроля версий, такие как Git или SVN.

Независимо от выбранного языка программирования и инструментов, важно учитывать требования проекта, опыт команды разработчиков и доступные ресурсы.

Таблица – 2.1 Сравнение языков программирования

Язык программирования	Сильные стороны	Слабые стороны	Функционал	Доступность для пользователя	Возможности
JavaScript	Подходит для разработки веб-приложений и динамической веб-страницы	Имеет проблемы с обработкой ошибок и типизацией данных	Мощная библиотека под названием Node.js, которая открывает доступ к серверной разработке на JavaScript	HTML и CSS работают в паре с JavaScript, что делает его доступным для веб-разработчиков	Создание интерактивных веб-сайтов и веб-приложений
Python	Простота и читаемость кода	Низкая производительность по сравнению с некоторыми другими языками	Широкий спектр библиотек и фреймворков в области науки данных, машинного обучения и веб-разработки	Легкий язык, который можно быстро изучить и использовать	Машинное обучение и анализ данных
Java	Популярность и широкое распространение	Разработка в Java обычно требует больше времени и намного больше кода, чем в некоторых других языках	Кроссплатформенность - возможность запуска программ на разных операционных системах	Большое количество доступной документации и ресурсов для изучения и разработки на java	Создание больших и сложных систем
C#	Широкий набор инструментов разработки и поддержка интеграции с другими технологиями	Язык C# ориентирован преимущественно на разработку для платформы .NET, что может быть ограничивающим для определенных проектов	JIT (Just-In-Time) компиляция для повышения производительности	Возможность создавать Windows-приложения	Разработка игр и приложений для Windows

Основным выбором из всех перечисленных языков программирования и инструментов стал Python3

Плюсы Python 3:

1) Современный язык программирования Python 3. предлагает множество новых возможностей и усовершенствований по сравнению с Python 2, включая более современный синтаксис, улучшенную поддержку Unicode, асинхронное программирование.

2) Активная поддержка и развитие: Python 3 активно развивается и имеет широкое сообщество разработчиков, которые поддерживают и улучшают язык, обеспечивая его актуальность и современность.

3) Поддержка новых технологий: Python 3 включает в себя поддержку

современных технологий, таких как асинхронное программирование, улучшенная работа с сетевыми протоколами, а также расширенные возможности в области научных вычислений и машинного обучения.

4) Улучшенная производительность: Python 3 обычно работает немного быстрее, чем Python 2, особенно в отношении некоторых операций со строками и списками.

Минусы Python 3:

1) Совместимость с предыдущими версиями: Некоторые старые библиотеки и модули до сих пор не полностью адаптированы к Python 3, что может вызвать проблемы с совместимостью.

2) Изменения в синтаксисе: Несколько аспектов синтаксиса Python3, могут потребовать переосмысления и изменения прежнего кода на Python 2.

3) Общий выбор использования Python 3 обоснован необходимостью использования современных возможностей, улучшенной производительности и безопасности, а также активной поддержкой сообщества [25].

Описание, библиотек, которые будут использоваться в Python 3

1) Библиотека sqlite3 в Python 3 предоставляет интерфейс для работы с БДSQLite, которая используется в небольших встраиваемых ИС.

SQLite является легковесной и простой в использовании базой данных, что делает ее удобной для хранения структурированных данных в информационной системе.

2) Tkinter - это стандартная библиотека для создания графического интерфейса пользователя (GUI) в Python.

Tkinter предоставляет инструменты для создания оконных приложений, виджетов, элементов управления и работы с событиями, что может быть полезным для создания пользовательского интерфейса информационной системы.

3) Библиотека Pillow предоставляет инструменты для работы с изображениями, включая чтение, запись, обработку и редактирование изображений.

Pillow может использоваться для работы с графическими данными в информационной системе, например, для отображения изображений в интерфейсе или обработки графических данных.

4) Модуль sys предоставляет доступ к некоторым переменным и функциям, связанными с интерпретатором Python и операционной системой.

Модуль sys может использоваться для управления параметрами запуска программы, работы с путями к файлам, доступа к аргументам командной строки и других системных функций.

5) Модуль webbrowser предоставляет простой интерфейс для работы с веб-браузерами из Python-приложений.

В информационной системе webbrowser может использоваться для открытия веб-страниц, ссылок или документации из приложения [9, с.127].

2.4 Выбор архитектурной модели

Выбор архитектурной модели для разработки информационной системы зависит от различных факторов, таких как требования к системе, ее размер и сложность, а также доступные ресурсы и сроки разработки.

На схеме (рисунок 2.2) отображена модель клиент-сервер: это одна из самых распространенных моделей, в которой клиентская часть взаимодействует с сервером для получения или отправки данных. Эта модель обеспечивает централизованное управление и обработку данных.

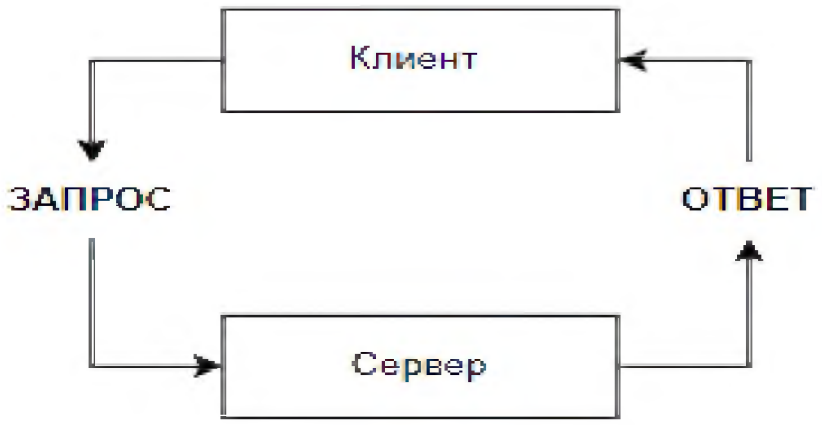


Рисунок 2.2 - Схема клиент-сервер

1) Клиент является конечной точкой, откуда инициируется запрос к серверу.

2) Запрос от клиента передается серверу.

3) Сервер обрабатывает запрос и генерирует ответ.

4) Ответ сервера передается обратно клиенту.

Таким образом, модель клиент-сервер обеспечивает взаимодействие между клиентскими приложениями и сервером, где клиент отправляет запросы, а сервер обрабатывает их и возвращает ответы.

На схеме (рисунок 2.3) модель многоуровневой архитектуры (например, 3-уровневая архитектура): в этой модели система разделяется на несколько слоев, каждый из которых выполняет определенные функции и взаимодействует с другими слоями. Например, уровень представления, бизнес-логики и доступа к данным.

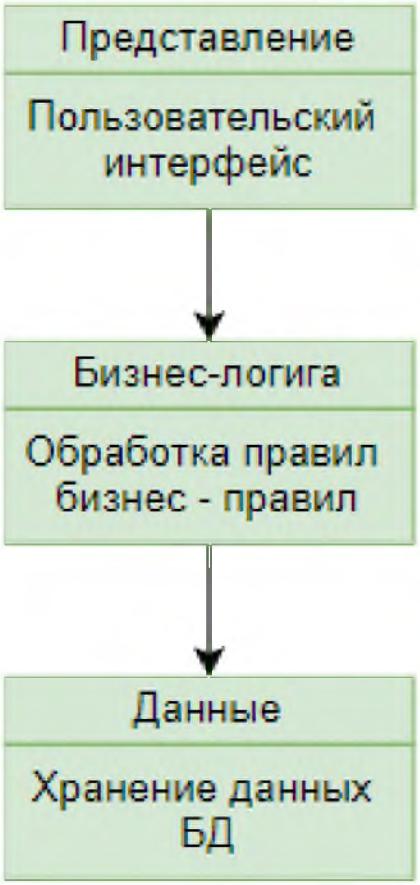


Рисунок 2.3–Схема многоуровневой архитектуры

Представление: этот уровень отвечает за пользовательский интерфейс и

взаимодействие с пользователем. Здесь могут использоваться веб-страницы, мобильные приложения или другие интерфейсы для отображения информации и получения ввода от пользователя.

Бизнес-логика: этот уровень отвечает за обработку данных, бизнес-правила и логику приложения. Здесь происходит манипуляция данными, применение бизнес-правил, обработка запросов от пользователя и формирование результатов.

Данные: этот уровень отвечает за хранение данных, как правило, в базе данных. Здесь происходит сохранение и извлечение данных для обеспечения работы бизнес-логики.

Эта архитектура позволяет разделить приложение на отдельные компоненты и слои, что делает его более гибким, модульным и легко поддерживаемым [27].

Модель событийной архитектуры: Схеме изображенная на рисунке 2.4 реагирует на события и выполняет соответствующие операции. Это позволяет системе быть гибкой и отзывчивой на изменения внешней среды.

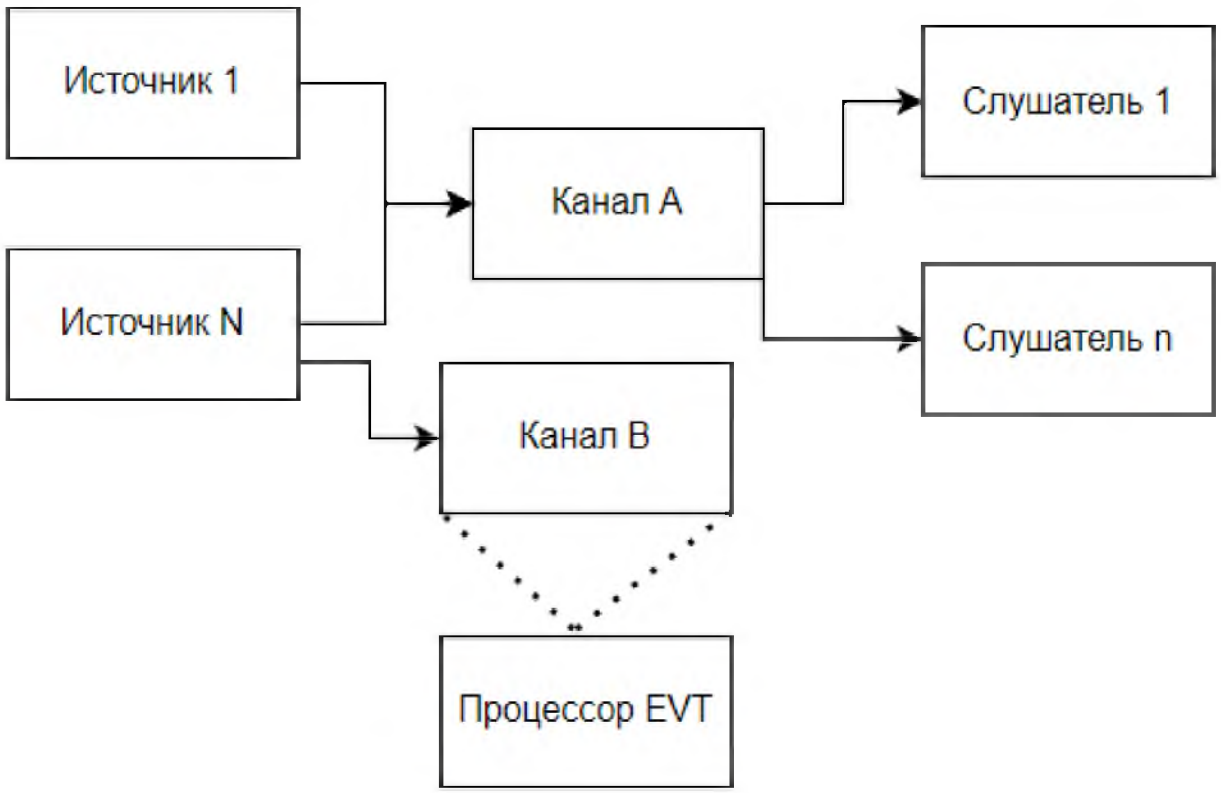


Рисунок 2.4 – Схема событийной архитектуры

Источники генерируют события, которые часто представляют собой реакцию на определённые действия или оповещают систему об изменениях состояния. Каналы обеспечивают среду для передачи событий. Они могут быть очередями сообщений, топиками паб/саб-систем или другими механизмами доставки. Слушатели подписываются на каналы и реагируют на поступающие через них события, инициируя их обработку.

Процессоры событий - службы или компоненты, которые выполняют логику обработки событий, могут изменять данные, вызывать другие сервисы, инициировать новые события и т.п.

Простая модель EDA прекрасно подходит для систем с реактивным поведением, где асинхронная обработка событий может вести к более отзывчивым и масштабируемым приложениям.

На схеме (рисунок 2.5) показана модель распределенной архитектуры: в этой модели система выполняется на распределенных компьютерах или серверах, совместно обрабатывая данные и ресурсы. Это может быть полезно для систем, требующих масштабируемости и отказоустойчивости.

Рисунок 2.5 - Схема распределенной архитектуры

В данной схеме интернет соединяется с балансировщиком нагрузки (LoadBalancer), который распределяет запросы от клиентов между несколькими серверами приложений (ApplicationServer). Затем приложение может получать или сохранять данные в распределенном кластере базы данных (DatabaseCluster), состоящем из нескольких баз данных (DatabaseServers). Благодаря распределенной архитектуре, система может обрабатывать большие объемы запросов и обеспечивать отказоустойчивость, так как каждый компонент имеет свои резервные варианты[6, с.101].

Объектно-ориентированнаямодель позволяет структурировать программное обеспечение как набор взаимодействующих объектов, которые являются экземплярами классов, обрабатывать данные и выполнять операции в соответствии с логикой, определенной в методах. Это облегчает управление сложными системами и способствует повторному использованию кода [9, с.162].

Изображенная схема на рисунке (рисунок 2.6) представляет собой пример объектно-ориентированной структуры, где каждый класс отражает сущность предметной области и имеет соответствующие атрибуты и методы. Например, сотрудники могут быть частью отделов, и их учет будет вестись в классе 4 учет и управление [29].

Объекты класса 1 сотрудники содержат информацию о самих сотрудниках. Каждый сотрудник принадлежит к определенному отделу и имеет свою должность. В классе 2 Отделы учитываются все сотрудники, работающие в данном отделе, а также возглавляющий его руководитель. Класс 3 должность содержит информацию о должностях, которые занимают сотрудники.

Класс 4 позволяет описывать и управлять операциями, связанными с учетом и управлением, такими как создание, чтение, обновление и удаление данных.

Объектно-ориентированная архитектура позволяет легко масштабировать систему, добавлять новые виды сущностей и функциональность без необходимости изменения существующего кода.Класс 5 архив является контейнером, в котором могут храниться функции различного назначения.

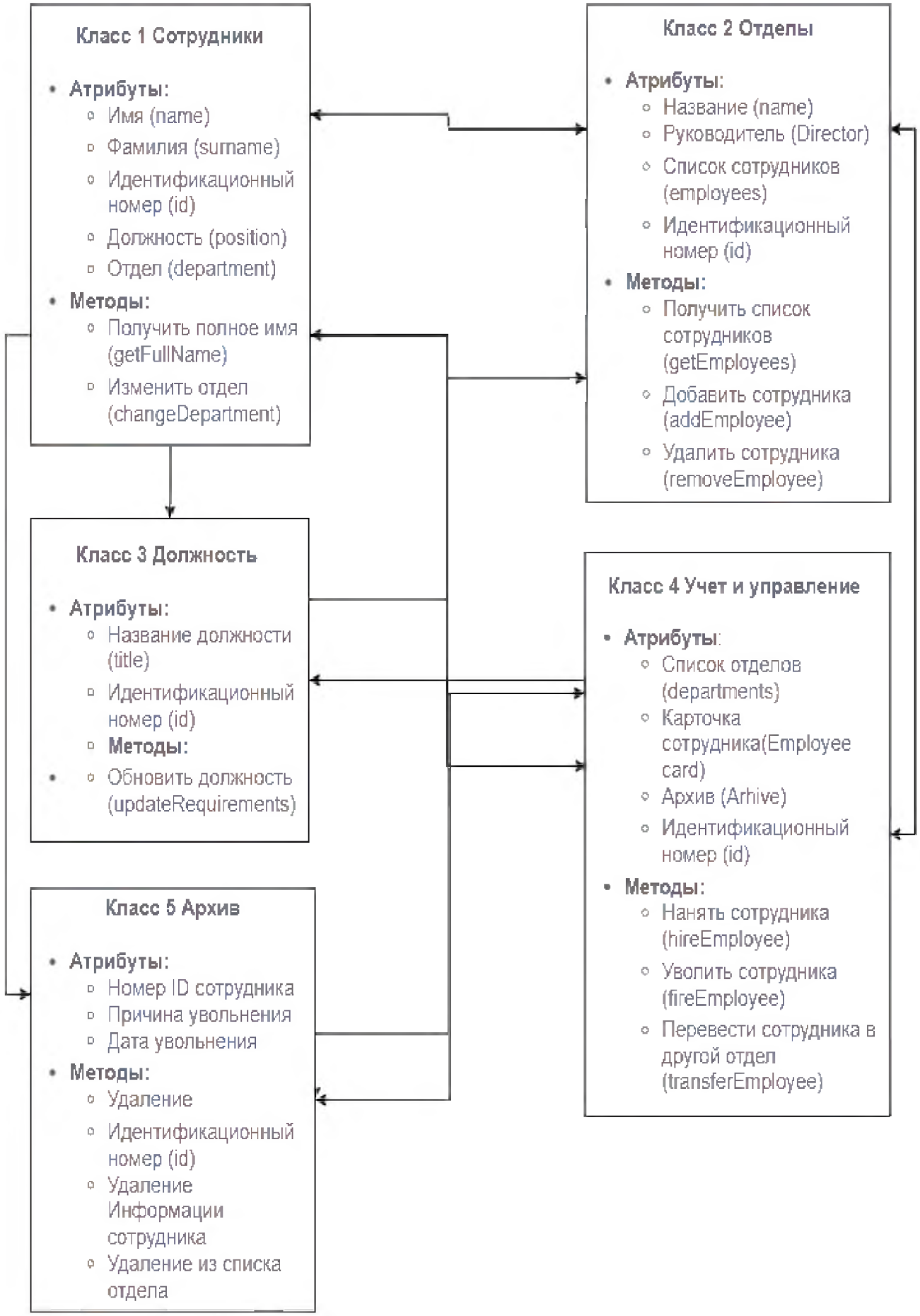


Рисунок 2.6 – Схема объектно-ориентированной архитектуры

2.5 Проектирование функциональных и нефункциональных требований к информационной системе.

Проектирование функциональных и нефункциональных требований к информационной системе для учета сотрудников включает определение различных функций и возможностей системы, а также ограничений, которые должны быть соблюдены.

Функциональные требования:

1) Аутентификация и авторизация: система должна проверять подлинность пользователей и предоставлять различные уровни доступа в зависимости от их роли.

2) Поиск и фильтрация: система должна предоставлять возможность поиска и фильтрации данных в соответствии с определенными параметрами.

3) Создание и хранение данных: система должна позволять пользователям создавать, редактировать и хранить различные типы данных, такие как текст, изображения, видео и др.

4) Генерация отчетов: система должна предоставлять возможность создавать и выводить на печать различные отчеты и статистику, основанную на доступных данных.

5) Интеграция с другими системами: система должна быть способна интегрироваться с другими внешними системами или сервисами для передачи и обмена данными.

6) Регистрация сотрудников: система должна предоставлять возможность регистрации новых сотрудников с указанием их персональных данных, таких как ФИО, должность, отдел и контактная информация.

7) Управление персональными данными: система должна предоставлять возможность управления и обновления персональных данных сотрудников, а также обеспечивать безопасность хранения этих данных.

Нефункциональные требования:

1) Удобство использования: система должна быть простой и интуитивно

понятной для пользователей.

2) Поддерживаемость: система должна быть легко обслуживаема, с возможностью обновления и исправления ошибок.

3) Совместимость: система должна быть совместима с другими существующими системами и стандартами.

4) Надежность: система должна быть надежной и стабильной, обеспечивая доступность и сохранность данных о сотрудниках в течение длительного времени.

5) Безопасность: Система должна обеспечивать защиту данных и предотвращать несанкционированный доступ. Иметь механизмы аутентификации и авторизации для предотвращения несанкционированного доступа к персональным данным сотрудников.

6) Производительность: система должна быть достаточно быстрой и эффективной, чтобы обрабатывать большое количество данных о сотрудниках и выполнять запросы пользователей в режиме реального времени.

7) Масштабируемость: система должна предоставлять возможность масштабирования для учета роста числа сотрудников и объема данных.

8) Интуитивный интерфейс: система должна иметь простой и понятный пользовательский интерфейс, чтобы обеспечить удобство работы и минимизировать время обучения пользователей.

3 Разработка информационной системы для учета сотрудников

3.1 Проектирование базы данных

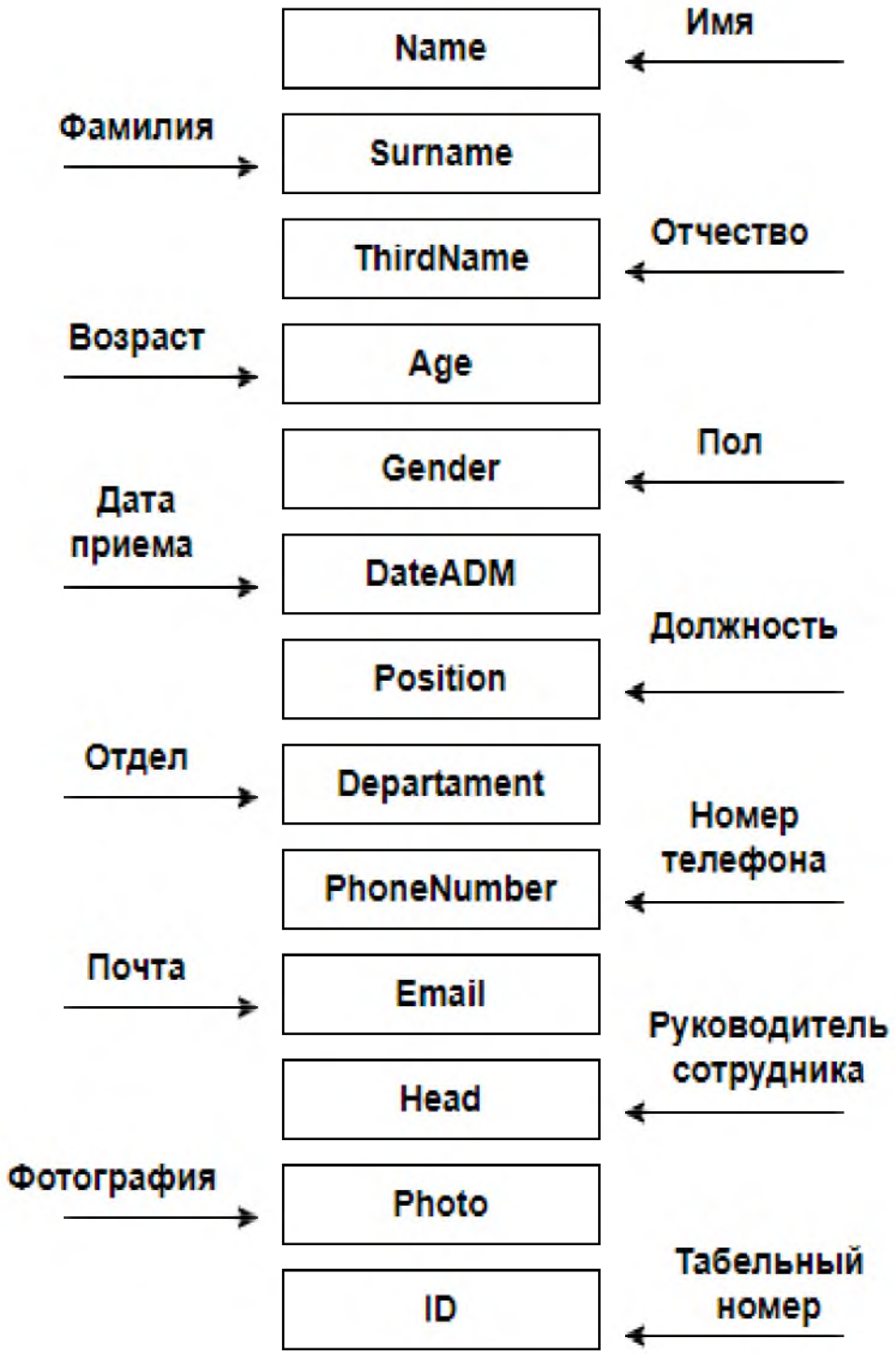


Рисунок 3.1 –Схема проектирования базы данных

На рисунке 3.1 показан проект базы данных, состоящий из следующих

пунктов:

1) Имя (name): Содержит имя сотрудника. Используется для идентификации сотрудника в базе данных.

2) Фамилия (surname): Содержит фамилию сотрудника. Используется для идентификации сотрудника в базе данных.

3) Возраст (age): Содержит возраст сотрудника. Используется для хранения информации о возрасте сотрудника.

4) Отчество (patronymic): Содержит отчество сотрудника. Используется для полного имени сотрудника или для формального обращения.

5) Пол (gender): Содержит пол сотрудника. Используется для хранения информации о поле сотрудника.

6) Дата приема на работу (hire_date): Содержит дату, когда сотрудник был принят на работу. Используется для отслеживания информации о стаже работы сотрудника.

7) Должность (position): Содержит должность, занимаемую сотрудником. Используется для хранения информации о должности сотрудника.

Отдел (department): Содержит информацию о том, в каком отделе работает сотрудник. Используется для организационной структуры компании.

8) Номер телефона (phone_number): Содержит номер контактного телефона сотрудника. Используется для связи с сотрудником.

9) Почта (email): Содержит адрес электронной почты сотрудника. Используется для связи и внутренней коммуникации.

10) Руководитель сотрудника (manager): Содержит информацию о руководителе сотрудника. Используется для организации иерархии в компании.

11)Связь между сотрудниками и руководителями с помощью идентификационного номера: Каждый сотрудник имеет руководителя, которому он подчиняется. Связь может быть установлена с помощью идентификатора руководителя, который ссылается на другую строку в базе.

Связь между отделами и сотрудниками: Каждый отдел имеет сотрудников, которые работают в этом отделе. Связь может быть установлена с

помощью идентификатора отдела, который ссылается на другую строку в базе данных отделов или сотрудников.

Связь между отделами и руководителями: Каждый отдел имеет руководителя, который управляет отделом. Связь может быть установлена с помощью идентификатора руководителя, который ссылается на другую строку в базе данных руководителей или отделов.

ID (идентификатор) - это уникальный идентификатор, присвоенный каждому объекту или сущности в базе данных. Он используется для идентификации и обслуживания связей между различными записями.

ID в базе данных может быть представлен в виде числа, строки или любого другого формата, обеспечивающего уникальность идентификатора. При создании нового объекта ему присваивается новый уникальный ID.

Использование ID позволяет эффективно организовать связи между различными записями в базе данных.

ID также используется для обеспечения уникальности идентификации каждой записи в базе данных. Благодаря этому, при поиске, обновлении или удалении записей можно точно указать на конкретную запись, обратившись к ее уникальному ID.

Связь между различными таблицами может быть установлена с помощью значений ID.

3.2 Разработка базы данных.

На рисунке (рисунок 3.2) изображена база данных ARCHIVE:

id: Уникальный идентификатор записи в архиве (INTEGER). Параметр UNIQUE обеспечивает уникальность каждой записи.

prichina: Поле, описывающее причину архивации данных (TEXT).

date: Текстовое поле, содержащее дату архивации (TEXT).

Здесь, будет хранится информация о причинах, по которым определённые действия или данные помещаются в архив, а также даты этих действий.

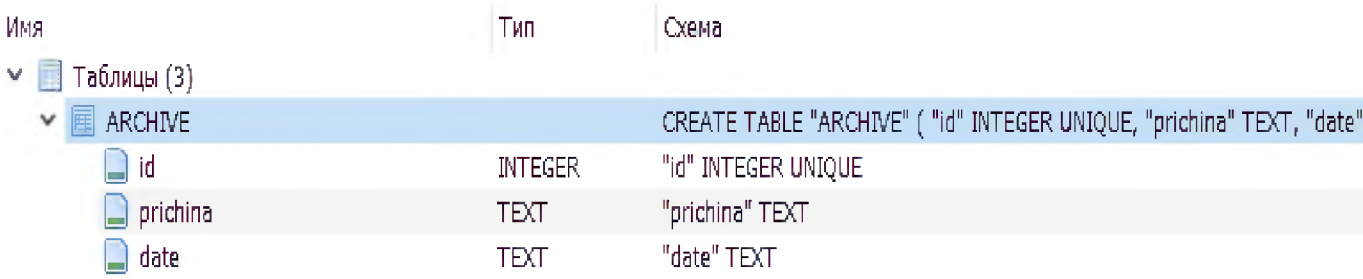
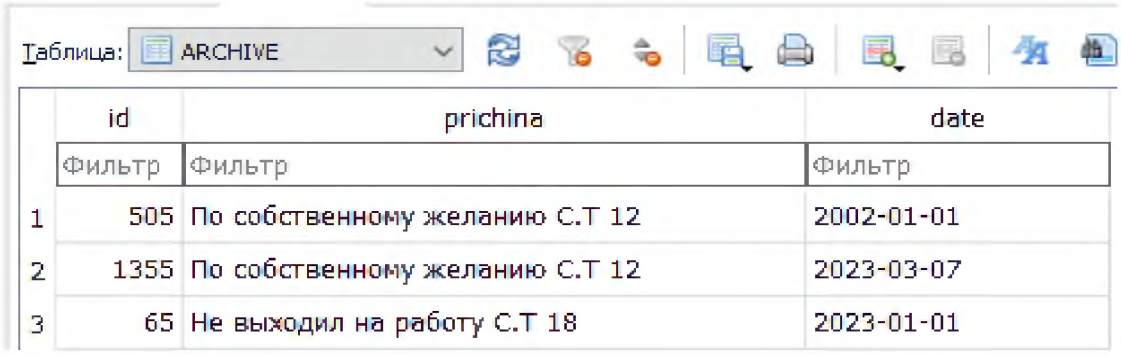


Рисунок 3.2 - База данныхархив с заданными параметрами



	id	prichina	date
	Фильтр	Фильтр	Фильтр
1	505	По собственному желанию С.Т 12	2002-01-01
2	1355	По собственному желанию С.Т 12	2023-03-07
3	65	Не выходил на работу С.Т 18	2023-01-01

Рисунок 3.3- Таблица базы данных архив

На рисунке 3.4 показана база данныхEmployee (сотрудники):

Name, Surname, ThirdName: Поля содержат текстовую информацию о имени, фамилии и отчестве сотрудника соответственно (TEXT).

Age: Возраст сотрудника (INTEGER).

Gender: Пол сотрудника (TEXT).

DateAdm: Дата приёма на работу (TEXT).

Position: Должность сотрудника (TEXT).

Department: Отдел, в котором работает сотрудник (TEXT).

PhoneNumber: Контактный номер телефона сотрудника (TEXT).

EMail: Электронная почта сотрудника (TEXT).

Head: Руководитель сотрудника (TEXT).

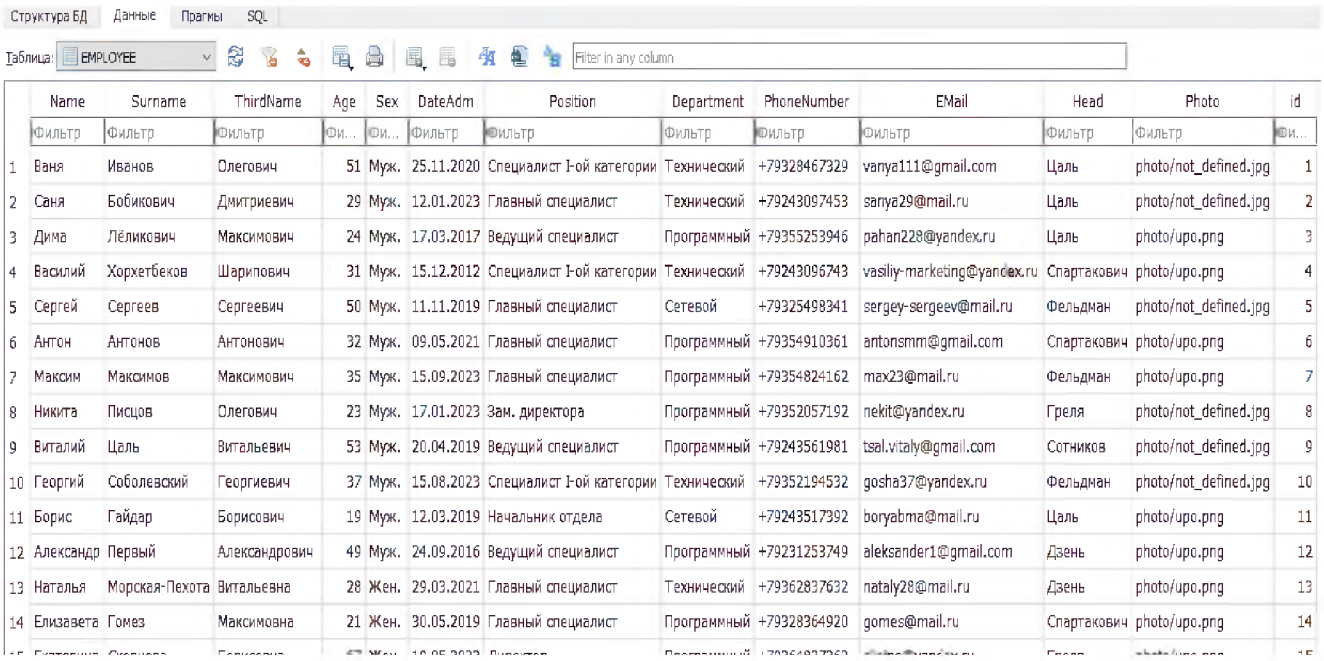
Photo: Ссылка на фотографию сотрудника. По умолчанию устанавливается 'photo/not_defined.jpg' (TEXT).

ID: Уникальный идентификатор сотрудника (INTEGER). Параметр NOT NULL обязывает поле быть заполненным; UNIQUE гарантирует уникальность для каждого сотрудника.

На рисунке 3.5 изображена таблица Employee которая хранит данные о сотрудниках организации, их личные и контактные данные, а также информацию, связанную с их работой в компании.

EMPLOYEE		CREATE TABLE "EMPLOYEE" ("Name" TEXT, "Surname" TEXT, "ThirdName" TEXT, "Age" INTEGER, "Sex" TEXT, "DateAdm" TEXT, "Position" TEXT, "Department" TEXT, "PhoneNumber" TEXT, "EMail"
Name	TEXT	"Name" TEXT
Surname	TEXT	"Surname" TEXT
ThirdName	TEXT	"ThirdName" TEXT
Age	INTEGER	"Age" INTEGER
Sex	TEXT	"Sex" TEXT
DateAdm	TEXT	"DateAdm" TEXT
Position	TEXT	"Position" TEXT
Department	TEXT	"Department" TEXT
PhoneNumber	TEXT	"PhoneNumber" TEXT
EMail	TEXT	"EMail" TEXT
Head	TEXT	"Head" TEXT
Photo	TEXT	"Photo" TEXT DEFAULT 'photo/not_defined.jpg'
id	INTEGER	"id" INTEGER NOT NULL UNIQUE

Рисунок 3.4-База данных сотрудники с заданными параметрами



Структура БД | Данные | Прагмы | SQL

Таблица: EMPLOYEE — Filter in any column

	Name	Surname	ThirdName	Age	Sex	DateAdm	Position	Department	PhoneNumber	EMail	Head	Photo	id
	Фильтр	Фильтр	Фильтр	Фи...	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр	Фи...
1	Ваня	Иванов	Олегович	51	Муж.	25.11.2020	Специалист I-ой категории	Технический	+79328467329	vanya111@gmail.com	Цаль	photo/not_defined.jpg	1
2	Саня	Бобикович	Дмитриевич	29	Муж.	12.01.2023	Главный специалист	Технический	+79243097453	sanya29@mail.ru	Цаль	photo/not_defined.jpg	2
3	Дима	Лёликович	Максимович	24	Муж.	17.03.2017	Ведущий специалист	Программный	+79355253946	pahan228@yandex.ru	Цаль	photo/upo.png	3
4	Василий	Хорхетбеков	Шарипович	31	Муж.	15.12.2012	Специалист I-ой категории	Технический	+79243096743	vasiliy-marketing@yandex.ru	Спартакович	photo/upo.png	4
5	Сергей	Сергеев	Сергеевич	50	Муж.	11.11.2019	Главный специалист	Сетевой	+79325498341	sergey-sergeev@mail.ru	Фельдман	photo/not_defined.jpg	5
6	Антон	Антонов	Антонович	32	Муж.	09.05.2021	Главный специалист	Программный	+79354910361	antonsmm@gmail.com	Спартакович	photo/upo.png	6
7	Максим	Максимов	Максимович	35	Муж.	15.09.2023	Главный специалист	Программный	+79354824162	max23@mail.ru	Фельдман	photo/upo.png	7
8	Никита	Писцов	Олегович	23	Муж.	17.01.2023	Зам. директора	Программный	+79352057192	nekit@yandex.ru	Греля	photo/not_defined.jpg	8
9	Виталий	Цаль	Витальевич	53	Муж.	20.04.2019	Ведущий специалист	Программный	+79243561981	tsal.vitaly@gmail.com	Сотников	photo/not_defined.jpg	9
10	Георгий	Соболевский	Георгиевич	37	Муж.	15.08.2023	Специалист I-ой категории	Технический	+79352194532	gosha37@yandex.ru	Фельдман	photo/not_defined.jpg	10
11	Борис	Гайдар	Борисович	19	Муж.	12.03.2019	Начальник отдела	Сетевой	+79243517392	boryabma@mail.ru	Цаль	photo/upo.png	11
12	Александр	Первый	Александрович	49	Муж.	24.09.2016	Ведущий специалист	Программный	+79231253749	aleksander1@gmail.com	Дзень	photo/upo.png	12
13	Наталья	Морская-Пехота	Витальевна	28	Жен.	29.03.2021	Главный специалист	Технический	+79362837632	nataly28@mail.ru	Дзень	photo/upo.png	13
14	Елизавета	Гомез	Максимовна	21	Жен.	30.05.2019	Главный специалист	Программный	+79328364920	gomes@mail.ru	Спартакович	photo/upo.png	14

Рисунок 3.5- Таблица база данных сотрудники

База данных изображенная на рисунке 3.6 sqlite_sequence:

name: Название объекта в базе данных (неопределённого типа).

seq: Последовательность, применяемая к соответствующему объекту (неопределённого типа).

sqlite_sequence — на рисунке 3.8 служебная таблица, используемая SQLite для отслеживания последних использованных значений AUTOINCREMENT для каждой таблицы, которая определяет такой

столбец.

Рисунок 3.6- База данных sqlite_sequence

с заданными параметрами

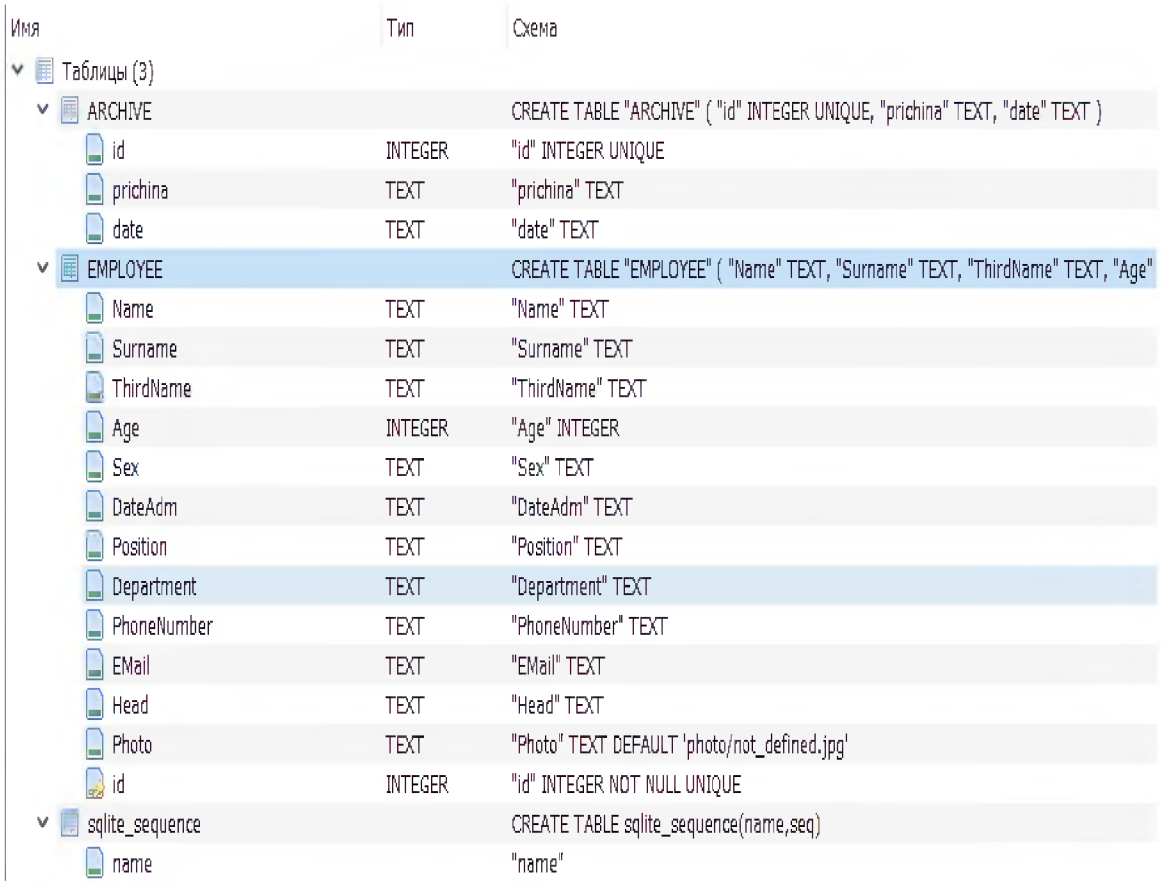


Рисунок 3.7 – Общий вид базы данных

Рисунок 3.8- Таблица база данных sqlite_sequence

заполненная с установленными параметрами

3.3 Создание программного кода для приложения

Данный элементакода, изображенный на рисунке 3.9 представляет собой окно авторизации с помощью библиотеки Tkinter в Python. Окно задается с размерами 320x240 пикселей и заголовком «Авторизация»

Далее в окне создаются следующие элементы интерфейса:

1) Label«Логин» - текстовая метка, которая отображается перед полем ввода логина пользователя.

2) Entry«login_entry» - поле ввода текста, куда пользователь должен ввести свой логин.

3) Label«Пароль» - текстовая метка, которая отображается перед полем ввода пароля пользователя.

4) Entry«password_entry» - поле ввода текста, куда пользователь должен ввести свой пароль.

5) Button«auth_but» - кнопка «Войти», которая будет вызывать функцию «auth» при нажатии.

В конце кода вызывается метод mainloop(), который запускает окно и ожидает взаимодействия пользователя.Предоставленный код использует библиотеку Tkinter для создания окна авторизации.

```
#check log n pass
login_window = Tk()
login_window.geometry("320x240")
login_window.title("Employee++ | Авторизация")
login_label = Label(login_window, text="Логин:")
login_label.pack(anchor='c')
login_entry = Entry(login_window)
login_entry.pack(anchor='c')

password_label = Label(login_window, text="Пароль:", )
password_label.pack(anchor='c')
password_entry = Entry(login_window)
password_entry.pack(anchor='c')
auth_but = ttk.Button(login_window, text="Войти", command=auth)
auth_but.pack(anchor='c', pady=10)
login_window.mainloop()
```

Рисунок 3.9- Код элемента создание настройка окна авторизации

Функция - auth() является частью кода изображенного на рисунке 3.10, предназначенной для аутентификации пользователей. Она проверяет введенные пользователем данные (логин и пароль) и в зависимости от их корректности либо открывает главное окно приложения, либо выводит сообщение об ошибке.

Получить значение, введенное пользователем в поле «login_entry» (элемент интерфейса для ввода логина) и сохранить в переменной entered_login. Получить значение, введенное пользователем в поле «password_entry» (элемент интерфейса для ввода пароля) и сохранить в переменной entered_password.

Сравнить значения переменных entered_login и entered_password с соответствующими значениями «admin» и «12345». Если оба значения совпадают, закрыть окно аутентификации (login_window) и открыть главное окно приложения (main_window()).

В противном случае, показать предупреждающее окно с сообщением «Неправильный логин или пароль!».

Эта функция используется вместе с графическим интерфейсом пользователя (GUI) и внутри нее доступны элементы интерфейса, такие как поле ввода логина (login_entry), поле ввода пароля (password_entry), окно аутентификации (login_window) и функция main_window() для открытия главного окна приложения.

Функция auth() использует простую проверку логина и пароля для аутентификации пользователя.

```
def auth():
    entered_login = login_entry.get()
    entered_password = password_entry.get()
    if entered_login == "admin" and entered_password == "12345":
        login_window.destroy()
        main_window()
    else:
        messagebox.showwarning("Ошибка", "Неправильный логин или пароль!")

    #check login and pass
```

Рисунок 3.10 - Код элемента авторизация

Создание пользователя. Сначала идет серия условий (if ... or ...): для проверки, не пусты ли переменные db_name, db_surname, db_thirdname, db_age, db_gender, db_dateadm, db_position, db_department, db_phone, db_email, и db_head.

Эти переменные, содержат информацию, которую пользователь должен ввести в форму: имя, фамилию, отчество, возраст, пол, дату принятия на работу, должность, отдел, телефонный номер, электронную почту и руководителя.

Если хотя бы одна из этих переменных пустая, появляется всплывающее предупреждение с помощью messagebox.showwarning, уведомляющее пользователя о необходимости заполнить все поля.

Добавление записи в базу данных: если все поля заполнены, код проверяет, пуста ли переменная db_photo, которая содержит информацию о фотографии сотрудника.

Добавление без фотографии: Если db_photo пуста, то осуществляется добавление записи в таблицу базы данных EMPLOYEE без фотографии сотрудника.

Используется SQL-оператор INSERT для добавления данных в таблицу. Переменные передаются в запрос через заполнители '?', чтобы избежать SQL-инъекций.

Добавление с фотографией: если переменная db_photo не пуста, код подразумевает, что фотография будет добавлена в таблицу.

В данном случае к запросу добавляются еще один столбец и соответствующая переменная - путь к фотографии сотрудника.

Выполнение запроса и закрытие соединения: после выполнения соответствующего SQL-запроса данные сохраняются в базе данных с помощью connection.commit(), что гарантирует их постоянство.

После, уничтожается окно add_window, которое, служило формой для ввода данных нового сотрудника.

```
if(db_name == '') or (db_surname == '') or (db_thirdname == '') or (db_age == '') or (db_sex == '') or (db_dateadm == '') or (db_position == '')
    messagebox.showwarning("Ошибка", "Заполните все поля!")
else:
    if db_photo == '':
        cursor.execute('INSERT INTO EMPLOYEE (Name, Surname, ThirdName, Age, Sex, DateAdm, Position, Department, PhoneNumber, EMail, Head) VALUES
        connection.commit()
        add_window.destroy()
    else:
        #db_photo = "photo/" + db_photo
        cursor.execute('INSERT INTO EMPLOYEE (Name, Surname, ThirdName, Age, Sex, DateAdm, Position, Department, PhoneNumber, EMail, Head, Photo)
        connection.commit()
        add_window.destroy()

 or (db_department == '') or (db_phone == '') or (db_email == '') or (db_head == ''):S



 (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)', (db_name, db_surname, db_thirdname, db_age, db_s



  VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)', (db_name, db_surname, db_thirdname, db
```

Рисунок 3.11- Код элемента Создание пользователя

Основное предназначение данного элемента кода, показанного на рисунке 3.11 - функционал добавления новых сотрудников в базу данных через графический интерфейс. Пользователь должен заполнить форму, если все поля заполнены корректно - информация подтверждается и добавляется в базу данных, после чего интерактивное окно закрывается.

Данный элемент кода, изображенный на рисунке 3.12, представляет функцию обработки информацию о сотруднике из базы данных и предоставляет возможность редактирования этих данных в новом окне.

Определение функции on_item_click(event): эта функцияпоказанная на рисунке 3.12 вызывается при клике на элемент, на элемент дерева (treeview) в GUI.C помощью tree.focus(), функция устанавливает фокус на выбранный элемент дерева.Из выбранного элемента извлекается text и value, после выполняется запрос к базе данных для извлечения подробной информации о

сотруднике по его ID. Создается новое окно window_clicked, где будут отображаться и можно будет редактировать подробные данные о сотруднике. И спользуется модуль Pillow для обработки фотографии сотрудника, а также созд аются виджеты Label и Entry для каждого поля данных (имя, фамилия, отчество и т.д.), которые заполняются текущими значениями для редактирован ия.Для выбора должности и отдела используются виджетыCombobox, которые являются списками dolzhnosti и otdelsi устанавливают текущее значение с данными из базы.

Функция save_emp() используется для сохранения измененных данных о сотруднике в базе данных и закрытия окна редактирования.

Кнопки «Сохранить изменения» и «Обновить фотографию»: Создаются две кнопки для сохранения данных о сотруднике и для обновления его фотографии соответственно.

Код обрабатывает действие пользователя для редактирования данных сотрудника в программе с графическим интерфейсом.

Пользователь может изменять информацию в отдельном окне, а затем сохранять её в базу данных.

```
def on_item_click(event):
    #frame3 - карточка сотрудника
    #polychenie
    #try:

    def save_emp():
        global photopath
        up_name = entry_set_name.get()
        up_surname = entry_set_surname.get()
        up_thirdname = entry_set_thirdname.get()
        up_age = entry_set_age.get()
        up_sex = entry_set_sex.get()
        up_dateadm = entry_set_date.get()
        up_position = entry_set_pos.get()
        up_department = entry_set_otdel.get()
        up_phone = entry_set_nomer.get()
        up_email = entry_set_pochta.get()
        up_head = entry_set_head.get()
        if photopath == '':
            photopath = name_of_photo

        up_photo = photopath
        photopath= ''
        cursor.execute('UPDATE EMPLOYEE SET Name = ?, Surname = ?, ThirdName = ?, Age = ?, Sex = ?, DateAdm = ?, Position = ?, Department = ?, PhoneNumber = ?, EMail = ?, Head = ?, Photo = ? WHERE id = ?'
        connection.commit()
        window_clicked.destroy()
```

Рисунок 3.12 - Код элемента обновление информации о сотруднике в базе данных

Этот элемент кода изображенный на рисунке 3.13 начинается с изменения переменной up_photo, которая представляет путь к фотографии сотрудника, затем переменная photopath очищается (устанавливается в пустую строку).

cursor.execute отправляет SQL-запрос на обновление записи в таблице EMPLOYEE. Здесь обновляются значения таких полей, как имя, фамилия, отчество, возраст, пол, дата трудоустройства, должность, отдел, номер телефона, адрес электронной почты, наличие руководителя и путь к фотографии для строк в таблице, которая соответствует указанному id.

Обеспечивая безопасное взаимодействие с базой данных. Переменные, такие как up_name, up_surname, up_thirdname, и так далее, представляют значения, которые будут вставлены в место маркеров.

connection.commit() подтверждает изменения в базе данных, делая обновление данных постоянным.

В конце идет взаимодействие с виджетом tree, который является элементом управления TreeView, используемым для представления таблицы или списка. С помощью tree.focus() получают выделенный элемент в этом дереве.

```
up_photo = photopath
photopath= ''
cursor.execute('UPDATE EMPLOYEE SET Name = ?, Surname = ?, ThirdName = ?, Age = ?, Sex = ?, DateAdm =
connection.commit()
window_clicked.destroy()
```

Рисунок 3.13- Код элемента обновление информации в карточке сотрудника

Функция delete_employee предназначена для удаления записи о сотруднике из базы данных и добавления информации о его увольнении в архив.

Создается новое окно с использованием библиотеки Tkinter

del_window = Tk() – создается экземпляр класса Tk, который представляет новое окно.

del_window.geometry(«650x300») – устанавливается размер окна 650 на 300 пикселей.

del_window.title(«Удаление сотрудника») – задается заголовок окна.

Функция del_exec – это функция, которая содержит логику удаления сотрудника из базы данных: Сначала получаются данные из полей ввода (entry_del, entry_prichina, entry_datau), которые должны быть определены ранее в функции delete_employee. Это идентификатор сотрудника (em_id), причина увольнения (em_prichina) и дата увольнения (em_date).

Далее происходит взаимодействие с базой данных через курсор (cursor), который также должен быть определен вне этой функции.

cursor.execute('DELETE FROM EMPLOYEE WHERE id= ?', (em_id,)) – выполняется SQL-запрос на удаление сотрудника по его идентификатору из таблицы EMPLOYEE.

cursor.execute('INSERT INTO ARCHIVE (id, prichina, date) VALUES (?, ?, ?)', (em_id, em_prichina, em_date)) – выполняется SQL-запрос на добавление записи в архивную таблицу ARCHIVE с идентификатором сотрудника, причиной увольнения и датой.

После выполнения SQL-запросов, изменения подтверждаются командой connection.commit(), где connection — это объект соединения с базой данных, который должен быть также определен где-то в вышестоящем коде.

После, вызывается метод del_window.destroy() для закрытия окна удаления сотрудника после того, как все действия выполнены.

```
# Удаление сотрудника
def delete_employee():
    del_window = Tk()
    del_window.geometry("650x300")
    del_window.title("Employee++ | Удаление сотрудника")


    #логика удаления
    def del_exec():
        em_id = entry_del.get()
        em_prichina = entry_prichina.get()
        em_date = entry_datau.get()
        cursor.execute('DELETE FROM EMPLOYEE WHERE id = ?', (em_id,))
        cursor.execute('INSERT INTO ARCHIVE (id, prichina, date) VALUES (?, ?, ?)', (em_id, em_prichina, em_date))

        connection.commit()
        del_window.destroy()
```

Рисунок 3.14- Код элемента функция удаления информации о сотруднике

Функция логики добавления сотрудника в базу данных.

Данный элемент кода, изображенный на рисунке 3.15 представляет функцию add_emp_osnova(), которая отвечает за логику добавления сотрудника в базу данных. В начале функции происходит считывание значений полей из соответствующих полей ввода (например, entry_name.get()). Имя, фамилия, отчество, возраст, пол, дата принятия на работу, должность, отдел, номер телефона и электронная почта в соответствующие переменные (например, db_name, db_surname). Затем происходит проверка на заполнение всех полей. Если хотя бы одно из полей осталось незаполненным, то выводится сообщение «Заполните все поля!» с помощью messagebox.show_warning(). Если все поля заполнены, то происходит вставка данных в таблицу EMPLOYEE базы данных. Если не указана фотография сотрудника (db_photo == ''), тоиспользуетсязапрос INSERTINTOEMPLOYEE (Name, Surname, ThirdName, Age, Sex, DateAdm, Position, Department, PhoneNumber, EMail, Head) VALUES. Еслиуказанпутькфотографиисотрудника (db_photo != ''), тоиспользуетсязапрос INSERT INTO EMPLOYEE (Name, Surname, ThirdName, Age, Sex, DateAdm, Position, Department, PhoneNumber, EMail, Head, Photo).

После выполнения запроса происходит сохранение изменений в базе данных с помощью connection.commit(). Затем окно добавления сотрудника (add_window) закрывается с помощью add_window.destroy().

```
# Логика добавления
def add_emp_osnova():
    global photopath
    db_name = entry_name.get()
    db_surname = entry_surname.get()
    db_thirdname = entry_thirdname.get()
    db_age = entry_age.get()
    db_sex = entry_sex.get()
    db_dateadm = entry_date.get()
    db_position = entry_position.get()
    db_department = entry_otdel.get()
    db_phone = entry_phone.get()
    db_email = entry_email.get()
    db_head = entry_head.get()
    db_photo = photopath#entry_photo.get()
    photopath = ''
    #print(db_photo)

    if(db_name == '') or (db_surname == '') or (db_thirdname == '') or (db_age == '') or (db_sex == '') or (db_dateadm == ''):
        messagebox.showwarning("Ошибка", "Заполните все поля!")
    else:
        if db_photo == '':
            cursor.execute('INSERT INTO EMPLOYEE (Name, Surname, ThirdName, Age, Sex, DateAdm, Position, Department, PhoneNur
            connection.commit()
            add_window.destroy()
        else:
            #db_photo = "photo/" + db_photo
            cursor.execute('INSERT INTO EMPLOYEE (Name, Surname, ThirdName, Age, Sex, DateAdm, Position, Department, PhoneNur
            connection.commit()
            add_window.destroy()
```

Рисунок 3.15 -Код элемента функция логики добавления в базу данных

Структура списка пользователей с функцией sort_column() и lambda изображенной на рисунке 3.16 представляет собой набор инструкций и правил, позволяющих работать с данными, представленными в виде структуры списка. Он имеет специальную функцию sort_column(), которая позволяет сортировать данные в этой структуре по определенной колонке. Для определения порядка сортировки используется лямбда-функция.

Структура списка: Язык позволяет создавать и обрабатывать структуру данных в виде списка, где каждый элемент может иметь собственные свойства и подсписки.

Функция sort_column(): Язык предоставляет встроенную функцию sort_column(), которая позволяет сортировать данные в структуре списка по указанной колонке. Сортировка может быть осуществлена по возрастанию или по убыванию значений.

Лямбда-функции: Для определения порядка сортировки используются лямбда-функции. Лямбда-функция - это анонимная функция, которая может быть объявлена внутри другой функции. Она позволяет определить критерий сортировки и использовать его в sort_column().

```
#treeview таблица с пользователями
tree.heading("id", text="Табельный номер",command=sorting)
tree.heading("Name", text="Имя", command=lambda: sort_column(tree, 'Name', False))
tree.heading("Surname", text="Фамилия", command=lambda: sort_column(tree, 'Surname', False))
tree.heading("ThirdName", text="Отчество", command=lambda: sort_column(tree, 'ThirdName', False))
tree.heading("Position", text="Должность", command=lambda: sort_column(tree, 'Position', False))
tree.heading("EMail", text="Почта", command=lambda: sort_column(tree, 'EMail', False))

tree.column("#1", stretch=YES, width=100, anchor='c')
tree.column("#2", stretch=YES, width=170, anchor='c')
tree.column("#3", stretch=YES, width=170, anchor='c')
tree.column("#4", stretch=YES, width=170, anchor='c')
tree.column("#5", stretch=YES, width=180, anchor='c')
tree.column("#6", stretch=YES, width=230, anchor='c')
```

Рисунок 3.16- Код элемента таблица с пользователями

Этот часть кода, изображенная на рисунке 3.17 представляет собой

функцию show_photo, которая использует базу данных (SQLite) для извлечения фотографий сотрудников и библиотеки TKinter в сочетании с Pillow для отображения изображений в графическом пользовательском интерфейсе.

defshow_photo(x): - это определение функции show_photo, которая принимает один параметр x, который o является идентификатором сотрудника (id).

cursor.execute('SELECT Photo FROM EMPLOYEE whereid= ?', (x,)) - здесь используется объект cursor для выполнения SQL-запроса к базе данных для извлечения поля Photo из таблицы EMPLOYEE для того сотрудника, у которого id соответствует переданному параметру x.

data = cursor.fetchone()[0] - попытка извлечь первую строку результатов запроса с помощью метода fetchone() и получение первого элемента этой строки, который должен быть фотографией.

photo_ph0 = Image.open(data) - использование класса Image из библиотеки или Pillow для открытия изображения. Data содержит путь к файлу изображения.

photo_ph1 = photo_ph0.resize((293, 293)) - изменение размера открытого изображения до 293x293 пикселей.

photo_ph2 = ImageTk.PhotoImage(photo_ph1, master=window_clicked) - создание объекта PhotoImage, который преобразует изображение Pillow в формат, который можно использовать в TKinter. Параметр master указывает, что window_clicked является родительским окном для этого изображения.

return photo_ph2 - возвращает объект photo_ph2, который теперь можно использовать для отображения в пользовательском интерфейсе приложения.

Для работы с полным функционалом данной функции нужны компоненты которые будут присутствовать и импортированы соответствующим образом в вашем скрипте:

Адекватная инициализация объекта cursor для взаимодействия с базой данных.

Модуль Image из библиотеки Pillow для обработки изображений.

Модуль ImageTk из Pillow для интеграции изображений Pillow с TKinter.

Объект window_clicked должен быть предварительно определен и относиться к экземпляру TKinter окна.

```
# Достать указатель на фотографию через id
def show_photo(x):
    cursor.execute('SELECT Photo FROM EMPLOYEE where id = ?', (x,))
    data = cursor.fetchone()[0]
    photo_ph0 = Image.open(data)
    photo_ph1 = photo_ph0.resize((293, 293))
    photo_ph2 = ImageTk.PhotoImage(photo_ph1, master=window_clicked)
    return photo_ph2
```

Рисунок 3.17 -Код элемента функция извлечения фотографии из библиотеки TKinter

Создание списка отделов,для этого требуется список отделов, где каждому отделу соответствует определенный флаг. Будет использоваться следующий список otdels, где otdels[0] представляет сетевой отдел, otdels[1] – технический отдел, otdels[2] – будет представлять из себя программный отдел, а otdels[3] – будет содержать в себе все отделы сразу под названием “все сотрудники”. На рисунке 3.18 показан элемента кода, в котором показан наглядный расположения отделов.

Установка флага: При нажатии на определенный отдел пользователем, устанавливается соответствующий флаг для выбранного отдела. При нажатии на отдел «Сетевой», устанавливается флаг «Сетевой».

Фильтрация данных: После установки флага, можно использовать эту информацию для фильтрации данных о сотрудниках. То есть, с помощью флага «Сетевой» можно отобразить только сотрудников, которые относятся к сетевому отделу. Это позволяет ограничить работу с данными только с конкретными сотрудниками из выбранного отдела.

Обновление, поиск, сортировка и т.д.: За счет фильтрации по установленному флагу, можно осуществлять операции обновления, поиска, сортировки и другие действия только с данными сотрудников из выбранного отдела. Например, можно обновлять информацию только о сотрудниках

сетевого отдела, искать среди них по определенным критериям или сортировать по имени или должности.

```
def btn_click_set():
    global flag_otdel
    flag_otdel = otdels[0]
def btn_click_tech():
    global flag_otdel
    flag_otdel = otdels[2]
def btn_click_prog():
    global flag_otdel
    flag_otdel = otdels[1]
def btn_click_vse():
    global flag_otdel
    flag_otdel = otdels[3]
```

Рисунок 3.18- Код элемента создание отделов с помощью flags

Функция сортировки столбцов. Функция sort_column принимает аргументы tree (дерево или иной вид иерархической структуры данных), col (номер столбца, по которому должна быть отсортирована структура), и reverse (флаг, указывающий нужно ли сортировать в обратном порядке). Функция сортирует структуру данных по указанному столбцу и меняет порядок дочерних элементов в соответствии с отсортированным порядком.

Строка data = [(tree.set(child, com), child) for child in tree.get_children('')] создаетсписок data, состоящийизкортежей. Каждый кортеж содержит два элемента: значение элемента структуры данных по указанному столбцу com (извлекается с помощью tree.set(child, com)) и сам элемент child.

Строка data.sort(reverse=reverse) сортирует список data в порядке, определенном флагом reverse. Если reverse равно True, то сортировка будет происходить в обратном порядке.

В цикле forindex, (val, child) inenumerate(data): происходит перебор элементов списка data. Переменная index хранит текущий индекс элемента, а переменные val и child хранят соответственно значение и сам элемент.

В строке trie.move(child index) происходит перемещение элемента child на позицию index в структуре данных trie.

Наконец, строка true.heading(call, command=Lambda: sort_column(tree, cool, notreverse)) вызывает функцию sort_column рекурсивно, меняя значение

флага reverse на противоположное. Это позволяет сортировать структуру данных в противоположном направлении при каждом вызове функции.

В итоге, функция sort_column сортирует указанную структуру данных по заданному столбцу и меняет порядок элементов в соответствии с отсортированным порядком, включая возможность сортировки в обратном порядке.

```
def sort_column(tree, col, reverse):
    data = [(tree.set(child, col), child) for child in tree.get_children('')]
    data.sort(reverse=reverse)
    for index, (val, child) in enumerate(data):
        tree.move(child, '', index)

    tree.heading(col, command=lambda: sort_column(tree, col, not reverse))
```

Рисунок 3.19 -Код элемента функции сортировки столбцов

На рисунке 3.20 показана функция поиска по списку

Функция search() выполняет поиск и отображение результатов в таблице tree.

tree.delete(*tree.get_children()): Очищает все строки в таблице перед началом нового поиска.

users = []: Создается пустой список users для хранения информации о сотрудниках.

users = show_info_employee(): Переменной users присваивается результат вызова функции show_info_employee(), которая, возвращает список с информацией о сотрудниках.

search_value = entry_text.get(): Получает значение из текстового поля entry_text, которое содержит текст для поиска.

ifsearch_value == refresh(): Если значение поиска равно «refresh()», вызывается функция refresh()

if(Len(search_value) > 1):: Если длина значения поиска больше 1 символа, выполняется следующий блок кода. s = search_value[0].upper() +

search_value[1:]: В переменную s записывается первый символ значения поиска, приведенный к верхнему регистру, и остальная часть значения без изменений. Используется для поиска с учетом регистра.

search_lower = search_value[0] + search_value[1:].lower():В переменную search_lower записывается первый символ значения поиска без изменений, а остальная часть значения приводится к нижнему регистру. Используется для поиска без учета регистра.

print(search_lower): Выводит значение переменной search_lower.

else: Если длина значения поиска меньше или равна 1 символу, выполняется следующий блок кода.

search_lower = ««: Присваивает пустую строку переменной search_lower.

foruserinusers:: Для каждого сотрудника в списке users выполняется следующий блок кода.

if (search_valueinuse) or (signuser) or (search_lowerinuser):: Если значение поиска содержится в имени сотрудника или фамилии сотрудника или в переменной search_lower содержится в строке сотрудника, то выполняется следующий блок кода.

tree.insert('', 'end', values=user): Вставляет строку с информацией о сотруднике в таблицу tree.

isorno = search_value.isdigit(): Проверяет, является ли значение поиска числом.

if(isopen == True):: Если isopen равно True выполняется следующий блок кода.

foruserinusers:: Для каждого сотрудника в списке users выполняется следующий блок кода.

ifint(search_value) inuse:: Если значение поиска является числом и присутствует в сотруднике (идентификаторе сотрудника), выполняется следующий блок кода.

tree.insert('', 'end', values=user): Вставляет строку с информацией о сотруднике в таблицу tree.

else: Если не выполняется ни одно из условий в блоке кода, продолжается следующая итерация цикла.

```
def search():
    tree.delete(*tree.get_children())
    users = []
    users = show_info_employee()


    search_value = entry_text.get()
    if search_value == '':
        refresh()
    if(len(search_value) > 1):
        s = search_value[0].upper() + search_value[1:]
        search_lower = search_value[0] + search_value[1:].lower()
        print(search_lower)
    else:
        s = ''
        search_lower = ''
    for user in users:
        if (search_value in user) or (s in user) or (search_lower in user):
            tree.insert('', 'end', values=user)

    isorno = search_value.isdigit()
    if(isorno == True):
        for user in users:
            if int(search_value) in user:
                tree.insert('', 'end', values=user)
            else:
                continue
```

Рисунок 3.20- Код элемента функции поиска по списку

Функция refresh.

Эта функция показанная на рисунке 3.21 предназначена для обновления данных, отображаемых в Treeview. Treeview используется для отображения информации о сотрудниках, где каждая строка дерева соответствует одному сотруднику.

users = [] - инициализирует пустой список users.

tree.delete(*tree.get_children()) - удаляет все узлы или строки из виджета tree. tree здесь - это переменная, которая ссылается на экземпляр Treeview. get_children() возвращает идентификаторы всех узлов, которые затем удаляются функцией delete().

users = show_info_employee() show_info_employee() - это функция, которая возвращает список информации о сотрудниках. Этот список присваивается переменной users.

foruserinusers: - итерация по каждому элементу списка users.

tree.insert(«», END, values=user) - вставка каждой записи пользователя в конец виджета tree как нового узла. »» и END определяют, что новые узлы должны быть добавлены к корню дерева и в его конец соответственно, а values=user указывает данные, которые должны быть вставлены в узел.

Функция sorting

Эта функция отвечает за сортировку данных в виджете tree.

globalflag - объявляет, что переменная flag является глобальной, что позволяет изменять её значение внутри функции.

ifflag: - проверяет значение flag. Если flag истина (True), то вызывается функция refresh(), которая, обновляет данные в tree.

else: - в случае, если flag ложь (False), вызывается функция sort_column2(). Эта функция сортирует данные в tree на основе значений во втором столбце.

flag = notflag - переключает значение flag на противоположное. Если flag был True, он станет False, и наоборот.

В данном коде элемента описано что , flag используется для переключения между двумя режимами сортировки сортировка по возрастанию и убыванию или между сортировкой и стандартным отображением данных. При каждом вызове функции sorting() состояние переключается, что ведет к вызовову refresh() и sort_column2().

```
def refresh():
    users = []
    tree.delete(*tree.get_children())
    users = show_info_employee()
    for user in users:
        tree.insert("", END, values=user)

def sorting():
    global flag
    if flag:
        refresh()
    else:
        sort_column2()
    flag = not flag
```

Рисунок 3.21- Код элемента функция обновления списка

3.4Интерфейс приложения

Авторизация в системе. На рисунке 3.22 окно авторизации, представляет из себя поле «Логин»: это текстовое поле, где пользователи вводят, имя пользователя, при авторизации в системе.

Поле «Пароль»: Это поле для ввода пароля, для авторизации в системе.

Кнопка «Войти»: Это кнопка, которую пользователь нажимает после ввода логина и пароля. После клика система проверяет введённые данные, если они введены верно, предоставляет доступ к системе.

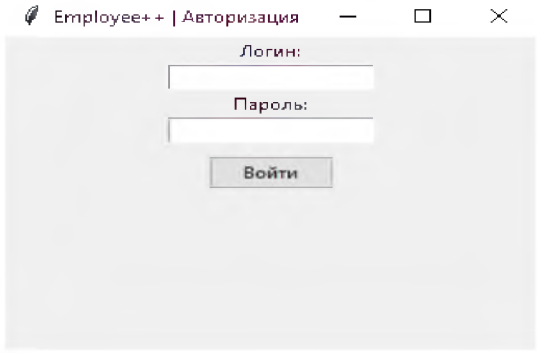


Рисунок 3.22- Окно авторизации пользователя

На рисунке 3.23 отображается диалоговое окно с сообщением об ошибке. Это окно возникает при попытке пользователя войти в систему и ввести логин и пароль.

«Ошибка: Неправильный логин или пароль!». Это сообщение, информирует пользователя о том, что введенные данные для входа в систему не соответствуют введенным данным.

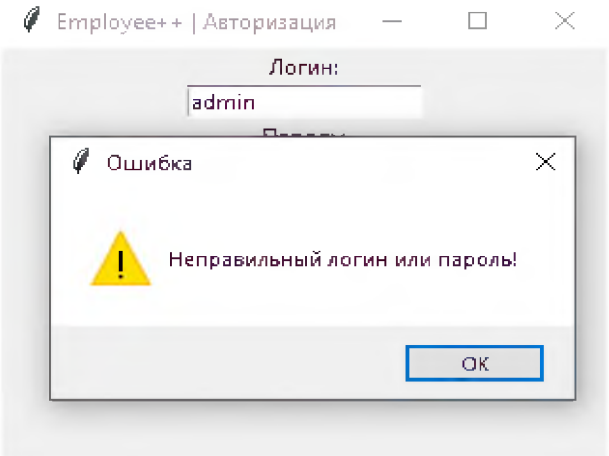


Рисунок 3.23- Окно авторизации пользователя неверный логин или пароль

На рисунке 3.24 изображено главное меню программы, после авторизации. Содержит в себе следующие кнопки и функционал:

– Кнопка - Добавить сотрудника: При нажатии открывается окно для ввода данных нового сотрудника.

– Кнопка - Удалить сотрудника: Эта функция позволяет удалять информацию об уже существующем сотруднике из системы.

– Кнопка - Техническая информация: Это кнопка-ссылка с переходом на онлайн ресурсс технической поддержкой.

– Кнопка - Обновление информации: Позволяет обновлять текущуюстраницу и информацию о сотрудниках в системе.

– Кнопка - Сетевой отдел: Переход к списку сотрудников, которые работают в сетевом отделе организации.

– Кнопка - Программный отдел: Аналогичная кнопка для просмотра или управления сотрудниками программного отдела.

– Кнопка - Технический отдел: Этот элемент интерфейса выполняет ту же функцию, что и предыдущие две, но для сотрудников технического отдела.

– Кнопка - Все сотрудники: Отображает всю базу сотрудников, в виде списка, где можно просматривать и редактировать информацию.

– Кнопка - Найти сотрудника: Осуществляет поиск сотрудников по базе данных с использованием различных критериев, указанных пользователем (например, по фамилии, должности и т.д.).

Поле поиска информации: текстовое поле, где можно ввести запрос для поиска информации внутри базы данных сотрудников, по табельному номеру или фамилии.

Столбец - Табельный номер: в таблице с данными этот столбец предназначен для уникального идентификатора сотрудника в системе учета.

Столбцы - Фамилия, Имя, Отчество: эти столбцы содержат личные данные сотрудника.

Столбец - Должность: в этом столбце указывается занимаемая

сотрудником должность в организации.

Столбец–Электронная почта: в этом столбце отображается электронный адрес сотрудника.

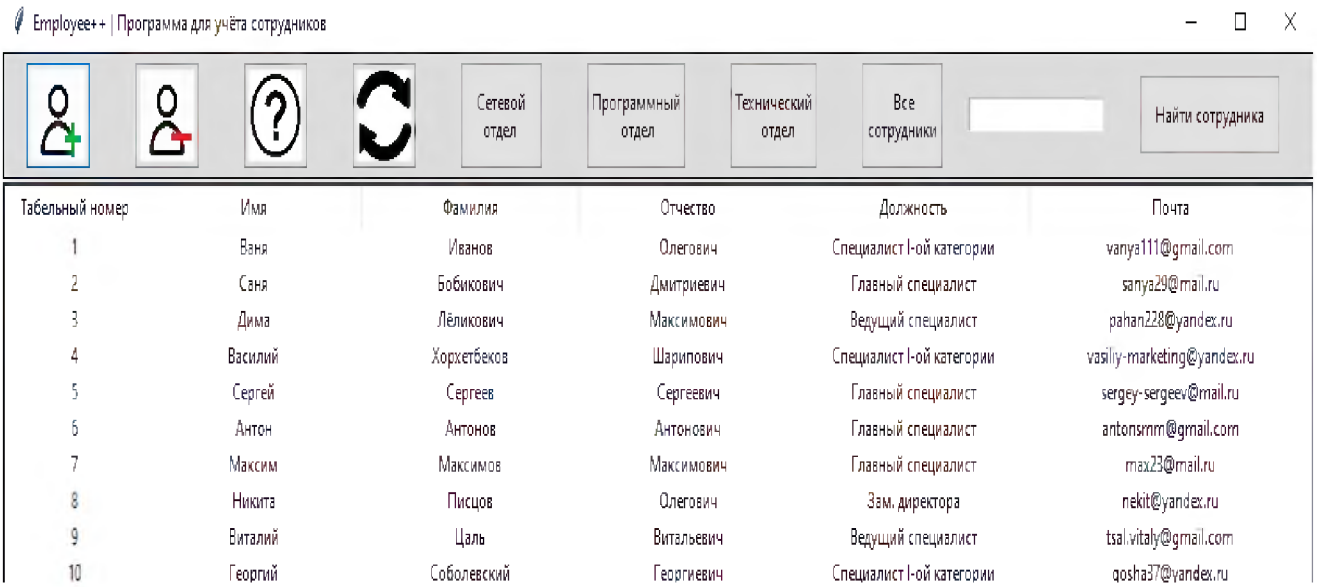


Рисунок 3.24- Главное меню программы

На рисунке 3.25 изображена кнопка «Добавление сотрудника в систему»

Рисунок 3.25 - Кнопка добавление сотрудника

После взаимодействия с кнопкой «добавить сотрудника», появляется окно, содержащее в себе поля, необходимые для заполнения.

Поле для ввода - Имя: это текстовое поле, где пользователь должен ввести имя сотрудника.

Поле для ввода - Фамилия: в этом текстовом поле вводится фамилия сотрудника.

Поле для ввода - Отчество: здесь требуется ввести отчество сотрудника.

Поле для ввода - Возраст: это поле для ввода возраста сотрудника.

На рисунке 3.26 показано поле для выбора Пола: это выпадающий, где

можно выбрать мужской либо женский пол сотрудника.

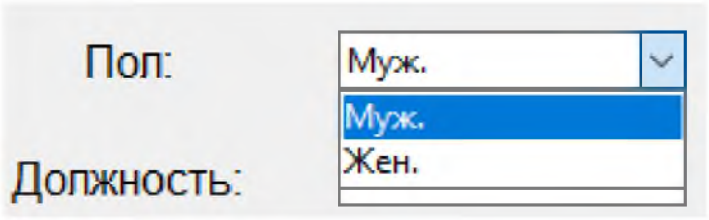


Рисунок 3.26 - Выбор пола

Поле для выбора даты - Принятие на работу: элемент управления, который позволяет выбрать дату из календаря.

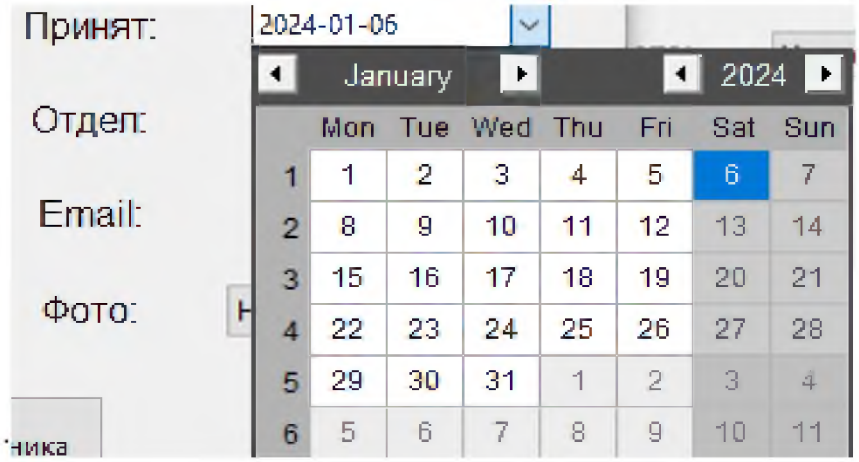


Рисунок 3.27 -Выбор даты

Поле для выбора - Должность: это выпадающий список с семью заранее определенными должностями. Пользователь выбирает одну из них, соответствующую должности нового сотрудника.

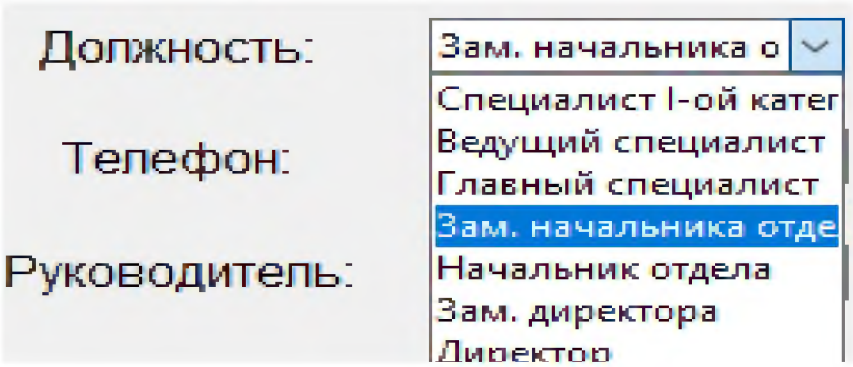


Рисунок 3.28-Выбор должности

Поле для выбора - Отдел: пользователь должен выбрать отдел, к которому будет прикреплен сотрудник.

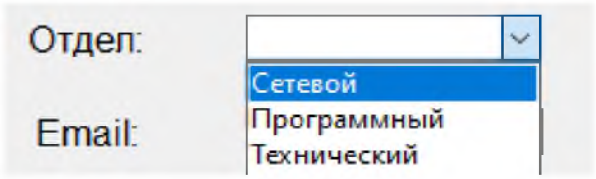


Рисунок 3.29- Выбор отдела

Поле для ввода - Телефон: текстовое поле для ввода номера телефона сотрудника.

Поле для ввода - Email: поле для ввода электронной почты сотрудника.

Поле для ввода - Руководитель: вводится Ф.И.О руководителя сотрудника.

Кнопка - Выбор фотографии сотрудника: кнопка, вызывающая диалоговое окно для загрузки фотографии сотрудника.

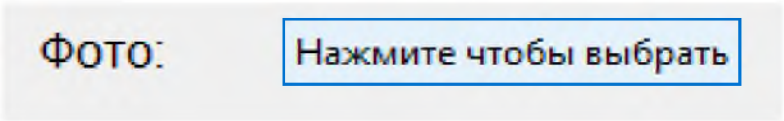


Рисунок 3.30- Выбор и сохранение фотографии

Кнопка - Добавить сотрудника: по нажатию этой кнопки информация будет сохранена в базе данных системы.

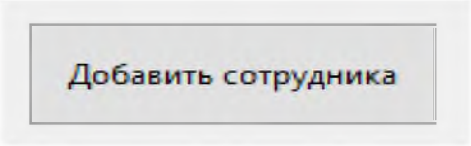


Рисунок 3.31- Кнопка добавление сотрудника

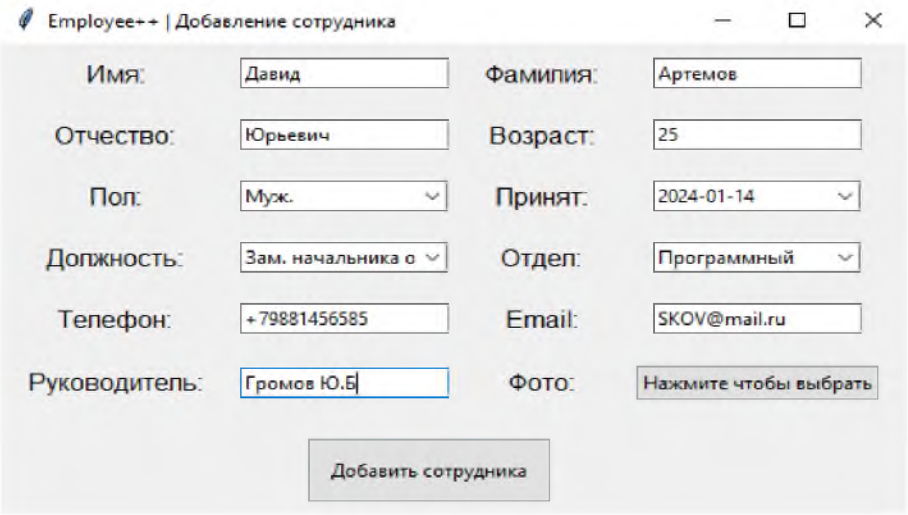


Рисунок 3.32- Окно добавление сотрудник

На рисунке 3.33 показана карточка сотрудника.

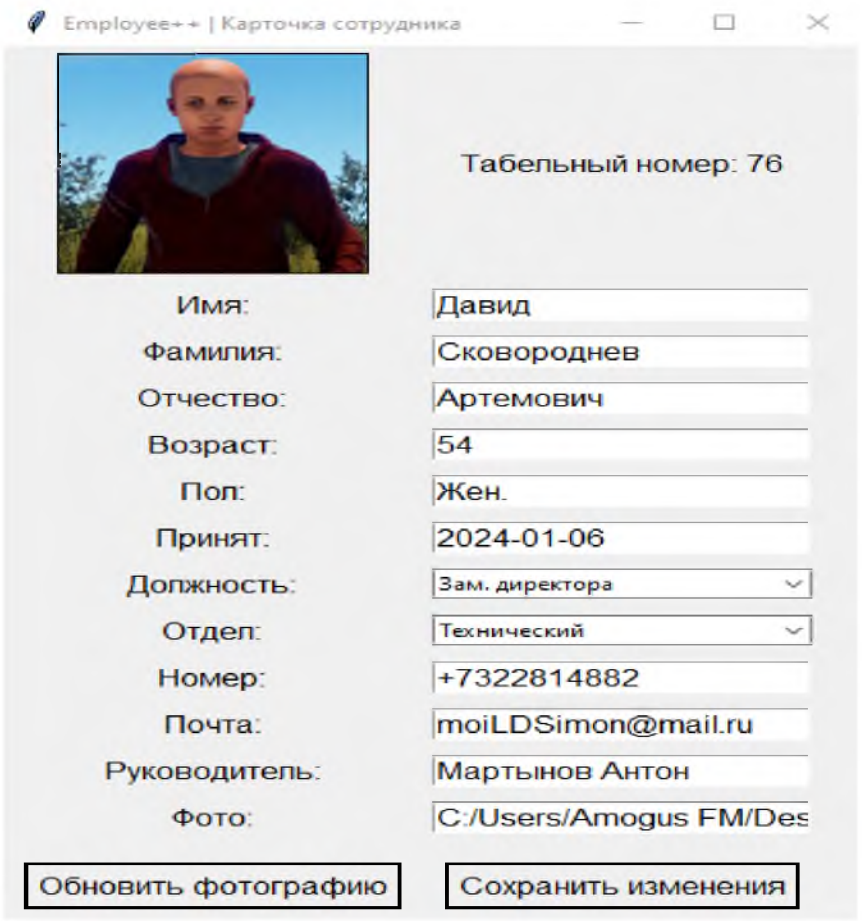


Рисунок 3.33- Карточка сотрудника

Карточка сотрудника содержит в себе такие поля как:

1) Имя, Фамилия, Отчество: пользователь может редактировать эти поля для обновления имени, фамилии и отчества сотрудника в базе данных.

2) Возраст: это поле для обновления возраста сотрудника.

3) Пол: здесь можно выбрать пол сотрудника из предложенных вариантов.

4) Принят: в этом поле, указывается дата принятия на работу.

5) Должность: здесь указывается занимаемая сотрудником должность в организации.

6) Отдел: это поле позволяет определить, в каком отделе работает сотрудник или перевести его в другой отдел.

7) Номер телефона: номер телефона сотрудника.

8) Почта: электронная почта сотрудника.

9) Руководитель: здесь указывается имя непосредственного руководителя данного сотрудника.

10) Фото: в текущее поле вставляется фотография сотрудника для лучшей идентификации.

11) Кнопка - изменить фотографию: при нажатии на эту кнопку пользователь может загрузить и заменить текущую фотографию сотрудника.

12) Кнопка - сохранить изменения: после внесения всех необходимых изменений в карточку сотрудника, нажатие этой кнопки сохранит все сделанные изменения в базе данных.

На рисунке 3.34 изображена информация о сотрудниках в «сетевом отделе».

Управления информацией о сотрудниках на предприятии Табельный номер - уникальный идентификационный номер сотрудника в системе учёта персонала.

Имя - личное имя сотрудника.

Фамилия - фамилия сотрудника.

Отчество - отчество сотрудника

Должность - занимаемая сотрудником должность отделе.

Почта - электронный почтовый адрес сотрудника, используемый для корпоративной связи.

функции:

1) Фильтрация по отделам: пользователь может фильтровать сотрудников по определённому отделу.

2) Поиск/сортировка: так же есть возможность поиска по ключевым словам, таким как имя или фамилия, а также сортировки списка сотрудников, например, по табельному номеру или должности.

3) Редактирование информации: ответственные пользователи могут добавлять новых сотрудников, редактировать информацию о текущих сотрудниках или удалять уходящих сотрудников.

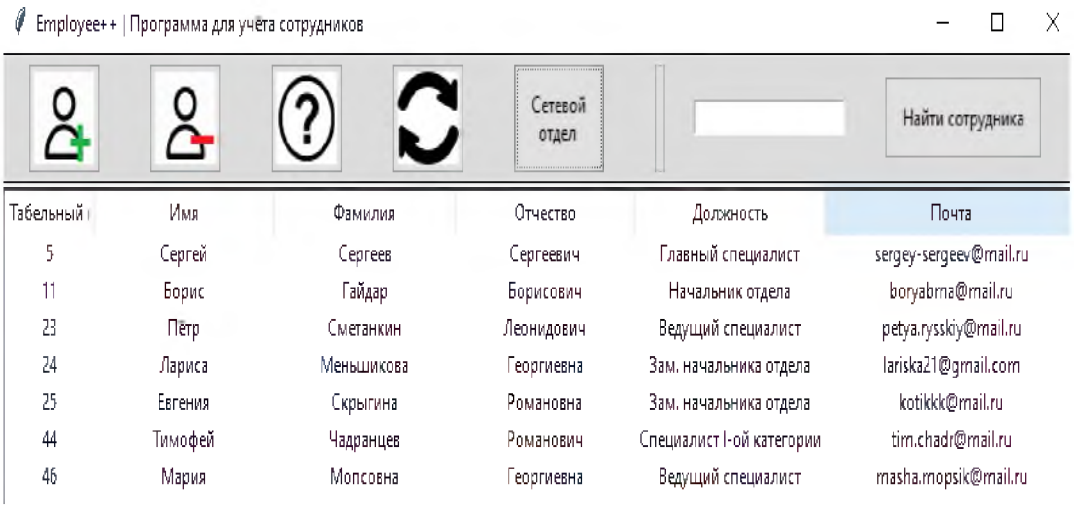


Рисунок 3.34- Сетевой отдел

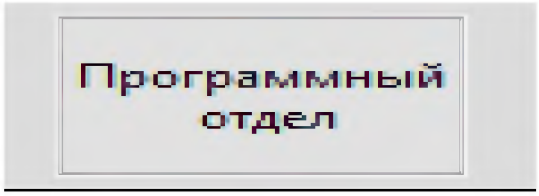


Рисунок 3.35- Кнопка для перехода в программный отдел

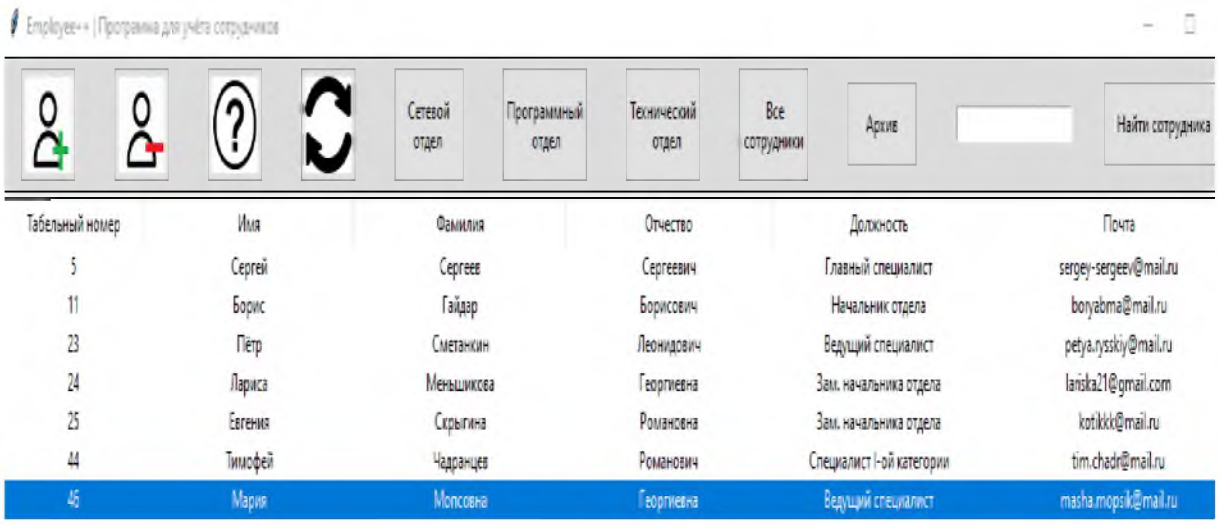


Рисунок 3.36- Программный отдел

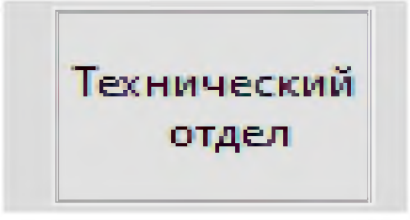


Рисунок 3.37 - Кнопка для перехода в технический отдел

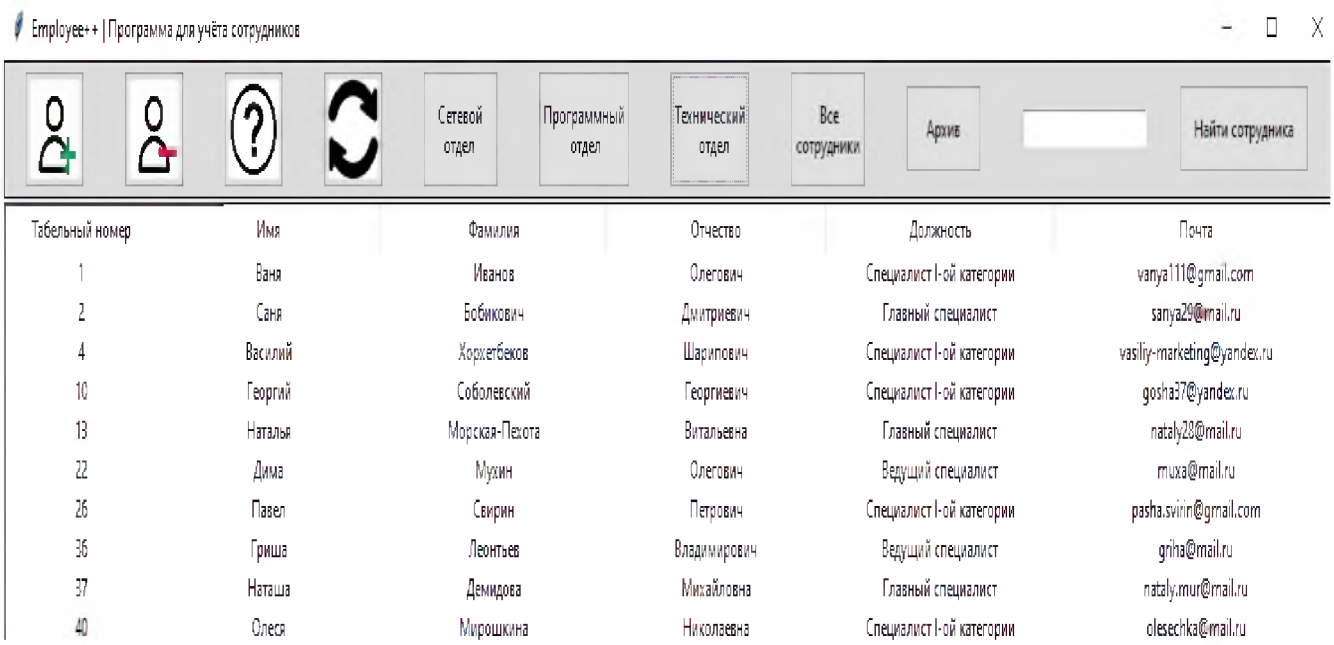


Рисунок 3.38 - Программный отдел

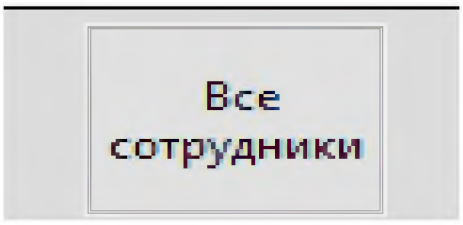


Рисунок 3.39 - Кнопка для просмотра всех сотрудников

Поиск сотрудников и информации:

1) Поле для ввода информации показанное на рисунке 3.40 представляет из себя текстовое поле, в которое пользователь вводит ключевые слова для поиска сотрудника. Эти ключевые слова могут включать имя, фамилию, отчество, должность, адрес электронной почты сотрудника и т. д.

2) Кнопка для поиска: после ввода информации в поле поиска, пользователь нажимает кнопку для срабатывания процесса поиска, система обрабатывает запросы с учетом регистра букв (то есть нечувствительна к регистру).

3) Поиск информации: при нажатии на кнопку поиска система просматривает базу данных на предмет соответствия запросу. Поиск нечувствителен к регистру, что означает, что не имеет значения, ввел ли

пользователь имя большими или маленькими буквами — результаты будут одинаковыми.

4) Вывод результатов поиска: если система находит совпадения, она отображает информацию о сотрудниках в таблице. В таблице будут поля с данными каждого найденного сотрудника, такие как имя, фамилия, отчество, должность, адрес электронной почты и т. д.

5) Возможность редактирования информации: каждая карточка сотрудника, которая появляется в результате поиска, может быть открыта для просмотра или редактирования информации. Реализовано для обновления данных о сотрудниках, таких как изменение должности, контактной информации и т. д.

Когда пользователь вводит данные в поле поиска и нажимает кнопку поиска, происходит следующее:

1) Система преобразует запрос в стандартный формат (например, приводит все символы к нижнему регистру), чтобы обеспечить нечувствительность поиска к регистру.

2) С помощью алгоритмов поиска система сверяет введенные данные с информацией в базе данных.

3) При обнаружении совпадений система собирает всю связанную информацию о сотрудниках и отображает в интерфейсе пользователя.

4) Пользователь может через интерфейс выбрать карточку с интересующим сотрудником и перейти к просмотру или редактированию информации.

Рисунок 3.40 - Кнопка для поиска сотрудников

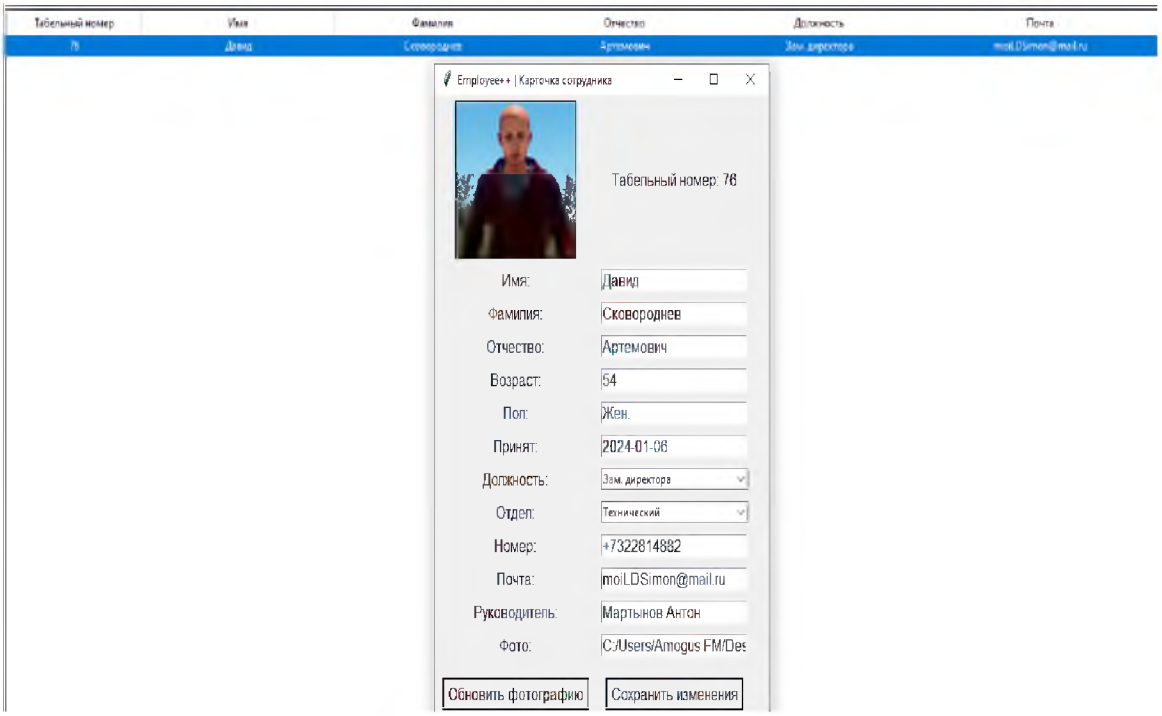


Рисунок3.41- Результат поиска сотрудника

Рисунок 3.42-Кнопка удаление сотрудника

На рисунке 3.43 показано окно «удаления сотрудника» содержащие в себе:

1) Поле для ввода - табельный номер (ID). Это текстовое поле, куда необходимо ввести уникальный идентификационный номер сотрудника, который подлежит увольнению. Табельный номер используется системой для точной идентификации записи сотрудника в базе данных.

2) Поле для ввода - причина увольнения сотрудника: это текстовое поле, в котором указывается причина увольнения.

3) Поле для ввода - дата увольнения: здесь вводится дата, на которую запланировано увольнение.

4) Кнопка – удалить: эта кнопка инициирует процесс удаления записи сотрудника из системы.

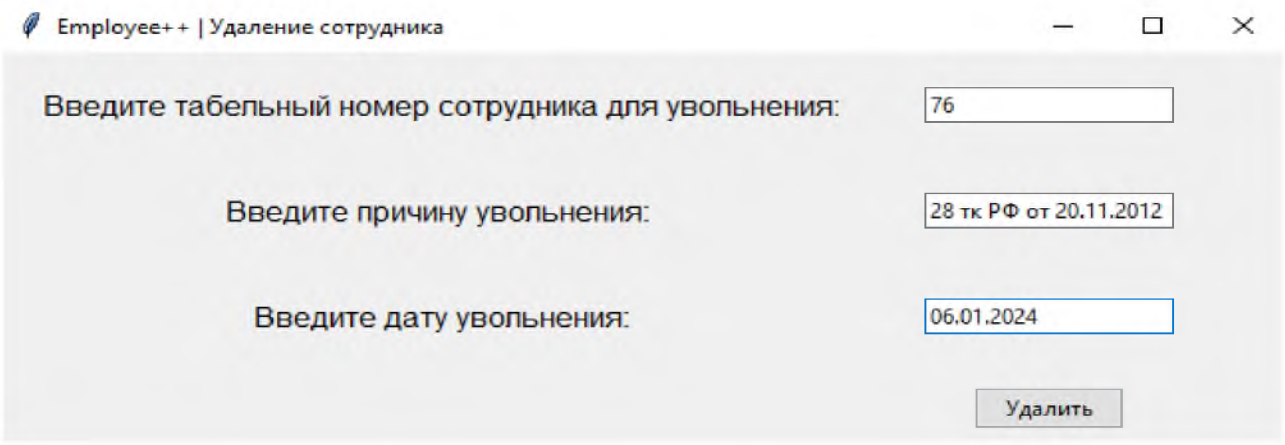


Рисунок 3.43- Окно для заполнение данных для удаления сотрудника

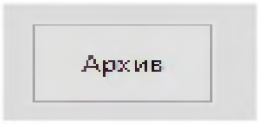


Рисунок 3.44 - Кнопка архив

Окно показанное на рисунке 3.45 «Архив» предназначен для отображения и просмотра информации об уволенных сотрудниках.

Табельный номер (id): Этот столбец содержит уникальный идентификационный номер каждого сотрудника. Каждому сотруднику присваивается уникальный табельный номер при приеме на работу.

Причина увольнения: Этот столбец предоставляет информацию о причинах, по которым сотрудник покинул организацию.

Дата: Этот столбец фиксирует точную дату увольнения сотрудника из организации.

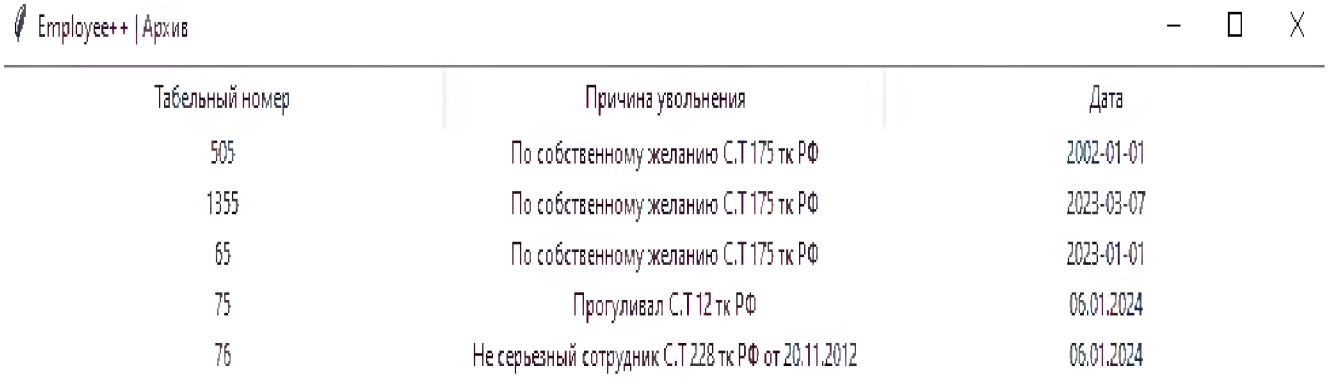


Рисунок 3.45- Окно с архивом удаленных сотрудников

Заключение

Разработка информационной системы для учета сотрудников на предприятии представляет собой значимый этап, который требует профессиональных знаний и тщательного подхода. В ходе процесса разработки информационной системы учета сотрудников на предприятии было сделано значительное количество работы, результатом которой стало создание функциональной и эффективной системы. Основная цель данного проекта заключалась в автоматизации и оптимизации процесса учета сотрудников, что позволяет предприятию эффективно управлять своим персоналом.

Важным этапом в работе над информационной системой стало изучение архитектурных моделей, которые являются основой разработки подобных систем. Архитектурные модели обеспечивают структурирование системы и определяют ее компоненты, взаимодействия и функциональность. Изучение архитектурных моделей позволило построить подходящую систему учета сотрудников на предприятии и обеспечить ее эффективное функционирование.

Для разработки информационной системы был проведен тщательный анализ и исследование DB Browser (SQLite). Это инструмент, который обеспечивает надежное хранение и управление информацией в базе данных. Изучение данного инструмента позволило создать структурированную базу данных для хранения информации о сотрудниках на предприятии.

Был изучен язык программирования Python 3. Этот язык программирования предоставляет широкие возможности для разработки информационных систем. Благодаря изучению и применению Python 3, было разработано приложение с графическим интерфейсом, что обеспечило удобное взаимодействие пользователей с информационной системой. Графический интерфейс предоставляет простоту использования и удобство пользователю, что было достигнуто благодаря использованию Python 3.

В ходе работы были выполнены поставленные задачи:

– Изучены теоретические основы разработки информационных

систем;

– Проведён анализ существующих информационных систем для учета сотрудников на предприятиях;

– Использована оптимальная архитектура и структура разрабатываемой системы;

– Разработаны функциональные и нефункциональные требования к системе;

– Спроектированы база данных и интерфейс пользователя системы.

В результате разработки информационной системы для учета сотрудников на предприятии, предприятие получило эффективный инструмент для контроля и управления персоналом. Благодаря созданной системе, можно эффективно отслеживать и управлять информацией о сотрудниках. Это позволяет предприятию оптимизировать процессы учета, повысить производительность и улучшить общую эффективность бизнеса.

В заключение, разработка информационной системы для учета сотрудников на предприятии является важным шагом для современного управления персоналом. Изучение и применение DB Browser (SQLite) и языка программирования Python 3 позволило создать функциональную и удобную систему, которая обеспечивает эффективный учет и управление сотрудниками предприятия. Полученное решение имеет широкий спектр применения и может быть адаптировано под различные потребности и бизнес-процессы.

Список литературы

1) Лашина, М.В. Информационные системы и технологии в экономике и маркетинге: учеб. пособие / М.В. Лашина, Т.Г. Соловьев. - М.: КноРус, 2018. - 480 с.

2) Остроух, А.В. Интеллектуальные информационные системы и технологии: Монография / А.В. Остроух. - СПб.: Лань, 2019. - 308 с.

3) Сулейманова, Д.Ю. Информационные системы управления инновационными процессами / Д.Ю. Сулейманова. - М.: Русайнс, 2018. - 224 с.

4) Уткин, В.Б. Информационные системы в экономике / В.Б. Уткин. - М.: Academia, 2018. - 189 с.

5) Федорова, Г.Н. Информационные системы: учеб. / Г.Н. Федорова. - М.: Academia, 2018. - 384 с.

6) Федотова, Е.Л. Информационные технологии и системы: учеб.пособие / Е.Л. Федотова. - М.: Форум, 2018. - 149 с.

7) Чистов, Д.В. Информационные системы в экономике: учеб. пособие / Д.В. Чистов. - М.: Инфра-М, 2019. - 248 с.

8) Хорольский, В.Я. Проектирование и эксплуатация энергоустановок телекоммуникационных систем: учеб. пособие / В.Я. Хорольский, А.Б. Ершов. - М.: Форум, 2019. - 285 с.

9) Перлова, О.Н. Проектирование и разработка информационных систем: Учеб. / О.Н. Перлова. - М.: Академия, 2018. - 272 с

10) Коршак, А.А. Проектирование систем газораспределения: учеб. пособие / А.А. Коршак. - Рн/Д: Феникс, 2018. - 192 с.

11) Зырянов, Ю.Т. Проектирование радиопередающих устройств для систем подвижной радиосвязи: учеб. пособие / Ю.Т. Зырянов, П.А. Федюнин, О.А. Белоусов. - СПб.: Лань, 2018. - 116 с.

12) Конюхова, Е.А. Проектирование систем электроснабжения промышленных предприятий (теория и примеры) / Е.А. Конюхова. – М.: Русайнс, 2018. – 224 с.

13) Дыбская, В.В. Проектирование системы распределения в логистике: Монография / В.В Дыбская. – М.: Инфра-М, 2019. – 277с.

14) Белов, В.В. Проектирование информационных систем: учеб. / В.В. Белов. – М.: Академия, 2018. – 144с.

15) Сайт Код-Академии. [Электронный ресурс]. URL:https:/ /www.codecademy.com (дата обращения 2.12.2023).

16) Официальный сайт компании «Новософт». [Электронный ресурс]. URL: http://www.novosoft.ru (дата обращения 21.11.2023).

17) Официальный сайт компании «Инфоком». [Электронный ресурс]. URL:https://infocom-s.ru/ (дата обращения 02.12.2023).

18) Официальный сайт группы компаний «АйТи». [Электронный ресурс]. URL: http://www.it.ru (дата обращения: 10.11.2023).

19) Сайт Professorweb. [Электронный ресурс]. URL: https://www.professprweb.ru

20) Сайт hackerrank. [Электронный ресурс]. URL: https://www.hackerrank.com/ (дата обращения 20.12.2023).

21) Сайт sqlfiddle. [Электронный ресурс]. URL: https://sqlfiddle.com/ (дата обращения 15.12.2023).

22) Сайт stepik. [Электронный ресурс]. URL: https://stepik.org/ (дата обращения 1.12.2023).

23) Сайт learnpython. [Электронный ресурс]. URL: https://learnpython.org/ (дата обращения 16.12.2023).

24) Сайт tutorialspoint. [Электронный ресурс]. URL: https://www.tutorialspoint.com/ (дата обращения 15.12.2023).

25) Сайт codecademy. [Электронный ресурс]. URL: https://www.codecademy.com (дата обращения 07.12.2023).

26) Официальный сайт компании «Инфоком». [Электронный ресурс]. URL: https://infocom-s.ru/ (дата обращения 010.12.2023).

27) Официальный сайт компании «Basecamp». [Электронный ресурс]. URL: https://basecamp.com (дата обращения 20.11.2023).

28) Официальный сайт компании «Новософт». [Электронный ресурс]. URL: http://www.novosoft.ru (дата обращения 29.11.2023).

29) Сайтworkspace. [Электронный ресурс].URL: https://workspace.ru /tools/cms/(дата обращения 16.11.2023).

30) Сайтpython. [Электронный ресурс].URL: https://www.python.org /downloads/(дата обращения 05.09.2023).