Оглавление

[ВВЕДЕНИЕ 2](#_Toc193328083)

[ГЛАВА 1. ИНСРУМЕНТЫ РАЗРАБОТКИ 4](#_Toc193328084)

[1.1. Общие сведения 4](#_Toc193328085)

[1.2. Понятие информационной системы 4](#_Toc193328086)

[1.3. Система контроля и управления доступом 5](#_Toc193328087)

[1.4. Обзор и выбор СУБД 7](#_Toc193328088)

[1.5. Обзор и выбор языка программирования 16](#_Toc193328089)

[1.6. Обзор и выбор среды разработки 21](#_Toc193328090)

[1.7. Обзор и выбор системы сборки 24](#_Toc193328091)

[1.8. Этапы разработки 26](#_Toc193328092)

[ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДКУТА 28](#_Toc193328093)

[2.1. Проектирование базы данных 28](#_Toc193328094)

[2.2. Разработка пользовательского интерфейса 31](#_Toc193328095)

[2.3. Тестирование программного обеспечения 39](#_Toc193328096)

[ЗАКЛЮЧЕНИЕ 40](#_Toc193328097)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 41](#_Toc193328098)

# ВВЕДЕНИЕ

Современные компьютерные технологии стремительно развиваются. Если раньше на предприятии все контракты хранились в бумажном виде, то сейчас они хранятся на серверах. Были охранники, которые контролировали вход на определённую территорию – в современном мире их заменяет система контроля и управлением доступом.

Актуальность темы заключается в том, что информационная система позволяет автоматизировать сбор, хранение и обработку информации о сотрудниках, их перемещения по предприятию и времени, проведённом на рабочем месте; обеспечить пользователей системы справочной рабочей информацией, работой с базой данных предприятия.

Целью данной работы является разработка информационной системы для системы контроля и управления доступом предприятия.

Задачи курсового проекта:

1. рассмотреть понятие информационной системы и её классификацию;
2. обозреть популярные системы управления базами данных, языки программирования, среды разработки, системы сборки и выбрать наиболее подходящие;
3. описать этапы разработки информационной системы;
4. создать информационную систему с помощью выбранных технологий;
5. протестировать информационную систему;
6. сделать выводы после окончания работ.

Объектом исследования данной курсовой работы является информационная система предприятия.

Предмет исследования данной курсовой работы – проектирование и создание информационной системы для предприятия.

Курсовой проект имеет следующую структуру:

* введение, в котором обозначена актуальность, цель, задачи, объект и предмет исследования;
* теоретическая глава, в которой рассмотрены основные понятия, СУБД и языки программирования, их преимущества и недостатки, и выбор подходящих;
* практическая часть, описывающая создание информационной системы и взаимодействия её компонентов;
* заключение, которое содержит выводы о проделанной работе.

Методы исследования: теоретические, статистические и эмпирические.

Практическая значимость работы заключается в том, что государственному учреждению необходимо иметь информационную систему, чтобы с помощью неё и физической СКУД контролировать доступ на территорию предприятия, в помещения, требующие определённый уровень доступа; получать информацию о времени нахождения того или иного сотрудника на рабочем месте; просматривать список подчинённых сотрудников и проектов, которыми они занимаются; добавлять новые заказы; просматривать и редактировать базу данных предприятия.

# ГЛАВА 1. ИНСРУМЕНТЫ РАЗРАБОТКИ

* 1. **Общие сведения**

Каждому государственному учреждению военного сектора страны важны эффективность и безопасность. Всё это может обеспечить информационная система, которая заменит людей, храня и обрабатывая данные с высокой скоростью за них. Но человеческий фактор всё равно необходим, чтобы контролировать систему, вносить новые данные, и в случае её неисправности принимать необходимые меры.

Одно из таких предприятий предложило контракт на создание такого программного обеспечения. Его функционал заключается в эффективной и безопасной работе с информацией базы данных, хранении заказов, отображении рабочей информации авторизовавшегося сотрудника или подчинённого, где находился работник в определённую дату и время. Под эффективностью подразумевается быстрый обмен данными между СУБД и программным обеспечением, а под безопасностью – использование минимального количества ресурсов устройства, уничтожение неиспользуемой конфиденциальной информации из памяти операционной системы, шифрование данных при передаче и хранении.

## Понятие информационной системы

Информационная система (ИС) – организационно-техническая система, которая предназначена для выполнения информационно-вычислительных работ или предоставления информационно-вычислительных услуг, удовлетворяющих потребности системы управления и ее пользователей – управленческого персонала внешних пользователей путем использования или создания информационных продуктов.

Информационные системы следует разделять по следующим характеристикам: структура, классификация.

Структура информационной системы состоит из совокупности подсистем.

Подсистема – это часть системы, выделенная по какому-либо признаку. Подсистемы бывают функциональными и обеспечивающими.

Функциональная часть состоит из ряда подсистем, которые решают конкретные задачи планирования, контроля, анализа, учета и управления деятельностью объектов. Для обеспечения нормальной деятельности основной части информационной системы в ее составе должна быть обеспечивающая подсистема.

Подсистемы являются обеспечивающими, если общая структура информационной системы рассматривается как совокупность подсистем независимо от сферы применения. Они бывают информационными, техническими, математическими и программными, организационными, правовыми.

Классификация ИС обычно разделена на категории:

* по степени автоматизации (ручные, автоматические, автоматизированные);
* по типу используемых данных (фактографические, документальные);
* по архитектуре (локальные, файл-серверные, клиент-серверные);
* по сфере применения (экономические, медицинские, географические);
* по охвату задач (персональные, групповые, корпоративные);
* по характеру обработки данных (информационно-справочные, решающие).

## 1.3. Система контроля и управления доступом

Система контроля и управления доступом (СКУД) - комплекс технических средств и программного обеспечения, предназначенный для создания контроля доступа в определённых помещениях или на ограниченных территориях.

Основная задача СКУД — управление доступом на заданную территорию (кого пускать, в какое время и на какую территорию).

Некоторые дополнительные задачи:

* учёт рабочего времени;
* расчёт заработной платы;
* ведение базы персонала или посетителей.

История СКУД началась на заре развития человечества – с того момента, как произошло расслоение общества по имущественному признаку. В первобытно-общинном строе все было общее, запирать что-либо не имело смысла. Но при переходе к следующей общественно-экономической формации появилась частная собственность, которую надо было охранять от тех, у кого этой собственности не было.

Сейчас эта система состоит из двух составляющих: физической и программной. Физическая часть – комплект взаимосвязанных устройств, таких как турникеты и считыватели магнитных карт доступа. Но подробно рассмотрим вторую составляющую.

В информационную систему СКУД входит база данных учреждения и программное обеспечение, взаимодействующее с ней. Программа должна соответствовать стандартам безопасности: обмен данными с базой осуществляется только после шифрования перед отправкой и соответствующей расшифровкой после получения, рекомендуемый алгоритм – SHA-256 и выше. Данные для аутентификации в системе должны хранится и пересылаться всегда в зашифрованном виде с помощью симметричного или асимметричного ключа; исключением является ПО СКУД, которое расшифровывает их для сравнения с введёнными данными пользователя для входа в систему, после чего удаляет расшифрованные данные из своей памяти.

Пример работы данной системы: сотрудник входит на предприятие, использовав свой пропуск с магнитным ключом, СКУД считывает его и, если аутентификация прошла успешно, пропускает дальше. В то же время она заносит данные о посещении этого сотрудника в базу данных.

## 1.4. Обзор и выбор СУБД

Система управления базой данных (СУБД) – программное обеспечение для создания и работы с базами данных, их последующего сопровождения: поддержка актуальности и настройка безопасности.

Главная функция СУБД – это управление данными (которые могут быть как во внешней, так и в оперативной памяти). СУБД обязательно поддерживает языки баз данных, отслеживает и сохраняет записи об изменениях, а также отвечает за копирование и восстановление информации после каких-либо сбоев.

Компоненты СУБД:

* язык запросов – используется для создания запросов к БД, например, SQL.
* Ядро СУБД – основной компонент, который обеспечивает выполнение запросов и доступ к данным.
* Драйверы – программное обеспечение, необходимое для обеспечения взаимодействия между СУБД и приложениями.
* Основное приложение – графический интерфейс для управления БД, включая создание таблиц, пользователей и управление правами доступа.
* Консоль – уже подключённый экземпляр терминала операционной системы к БД.
* Библиотеки – наборы программных модулей, которые можно использовать для создания приложений, использующих базу данных.
* Хранилище данных – физическое устройство или набор устройств, где содержится вся информация, например, жесткий диск.

Основные виды СУБД:

* по расположению (локальные, централизованные, распределённые);
* по хранению и обработке данных и запросов (встраиваемые, файл-серверные, клиент-серверные);
* по языку запросов (SQL-ориентированные, NoSQL-ориентированные);
* по структуре и организации данных (колоночные, графовые, документные, ключ-значение, реляционные).

Хоть все системы управления базами данных схожи между собой: предоставляют возможность просмотра и управления базами данных, таблицами, могут иметь один и тот же язык запросов, отличия всё-таки есть. Каждая СУБД должна иметь свою документацию по использованию для пользователей, которая может различаться уровнем доступности информации. Для больших государственных учреждений, часто использующих платные, более продвинутые версии СУБД, должна предоставляться техническая поддержка со стороны компании-разработчика ПО, которая в свою очередь имеет свои особенности и условия.

При выборе СУБД стоит учитывать, насколько большой кластер данных имеется или может появиться, цену ПО, совместимость с установленной операционной системой, удобно ли подключаться к ней из стороннего приложения. Если предприятие или малый бизнес использует бюджетную систему с базовым функционалом, но в будущем может расширяться, то и возможностей программного обеспечения для работы с базами данных уже может не хватить и тогда будет удобно установить расширенную версию, что нужно рассмотреть заранее, так как, если её не будет, то переход на другую СУБД будет проблематичным.

Рассмотрим несколько популярных СУБД для предприятий, которые могут иметь и платную, и бесплатную версии.

* MySQL является одной из самых популярных и распространенных СУБД, которая используется во многих компаниях (например, Facebook, Wikipedia, Twitter, LinkedIn, Alibaba и других). MySQL представляет собой реляционную СУБД, которая относится к свободному программному обеспечению: она распространяется на условиях GNU Public License. Как правило, эту систему управления базами данных определяют как хорошую, быструю и гибкую, рекомендованную к применению в небольших или средних проектах.

Принцип работы MySQL основан на клиент-серверной модели: база данных и система управления ею хранятся на сервере, а пользователи подключаются к ней с помощью своих устройств.

Основные компоненты MySQL:

* MySQL Server - центральная часть MySQL, которая управляет всеми базами данных. Сервер обрабатывает SQL-запросы, принимает, обрабатывает, отправляет данные и обеспечивает взаимодействие с клиентами.
* Хранилище данных - отвечает за фактическое хранение и организацию данных в таблицах.

У MySQL есть множество различных преимуществ. Например, она поддерживает различные типы таблиц – как известные MyISAM и InnoDB, так и более экзотичные HEAP и MERGE. Кроме того, количество поддерживаемых типов постоянно растет. MySQL выполняет все команды быстро – возможно, сейчас это самая быстрая СУБД из всех существующих. С этой системой управления базами данных может одновременно работать неограниченное количество пользователей, а число строк в таблицах может достигать 50 миллионов.

Так как в сравнении с некоторыми другими системами MySQL поддерживает меньшее количество возможностей, то и работать с ней значительно проще, чем, к примеру, с PostgreSQL, о которой будет рассказано ниже.

Для работы с MySQL используется не только текстовый, но и графический режим. Это становится реальным благодаря приложению phpMyAdmin: для работы в приложении вам даже не нужно знать SQL-команды, а администрировать свою базу данных можно прямо через браузер.

MySQL – это выбор тех, кому необходима СУБД для проекта небольшого или среднего размера, быстрая и удобная в работе и без сложностей с администрированием.

* SQLite - На данный момент это одна из самых компактных СУБД. Также она является встраиваемой и реляционной.

Она позволяет хранить все данные в одном файле и, благодаря своему небольшому объему, отличается завидным быстродействием. SQLite значительно отличается от MySQL и PostgreSQL своей структурой: движок и интерфейс этой СУБД находятся в одной библиотеке – и именно это позволяет выполнять все запросы очень быстро. Другие СУБД (MySQL, PostgreSQL, Oracle и т.д.) используют парадигму «клиент-сервер», когда взаимодействие происходит через сетевой протокол.

Некоторые сферы применения SQLite:

* мобильные приложения: SQLite позволяет приложениям хранить и управлять данными, такими как настройки, кэшированные данные, локальные базы данных и другие.
* Десктопные приложения: SQLite может быть встроен в десктопные приложения для хранения данных, таких как локальные настройки, журналы, история и многое другое.
* Веб-браузеры: некоторые веб-браузеры используют эту БД для хранения истории посещений, закладок и кэша веб-страниц.
* Встроенные системы: SQLite может использоваться во встроенных системах и устройствах, таких как медицинское оборудование, автомобильные информационные и развлекательные системы, умные дома и многое другое, где требуется управление данными.
* Веб-серверы: БД иногда используется веб-серверами для временного хранения данных или кэширования, особенно в тестовых и разработочных средах.
* Аналитика и инструменты отладки: СУБД также может быть использован для анализа и обработки данных, а также в инструментах отладки и тестирования.

Некоторые недостатки SQLite:

* ограниченная поддержка типов данных: SQLite поддерживает только четыре типа данных, которые реализованы в SQL: INTEGER — целое число, REAL — дробное число, TEXT — текст, BLOB — двоичные данные.
* Отсутствие хранимых процедур: так называются блоки кода на SQL, которые сохраняются в базу данных. SQLite их не поддерживает из-за особенностей архитектуры.
* Ограничения в применении: отсутствие сервера — преимущество и недостаток одновременно. Например, к одной базе не смогут обращаться несколько разных устройств.
* PostgreSQL - свободно распространяемая система управления базами данных относится к объектно-реляционному типу СУБД. Как и в случае с MySQL, работа с PostgreSQL основывается на языке SQL, однако, в отличие от MySQL, PostgreSQL поддерживает стандарт SQL-2011. Эта СУБД не имеет ограничений ни по максимальному размеру базы данных, ни по максимуму записей или индексов в таблице.

Если говорить о преимуществах PostgreSQL, то в первую очередь это надежность транзакций и репликаций, возможность наследования и легкая расширяемость. PostgreSQL поддерживает различные расширения и варианты языков программирования, такие как PL/Perl, PL/Python и PL/Java. Также есть возможность загружать C-совместимые модули.

Некоторые особенности PostgreSQL:

* поддержка сложных типов данных: СУБД позволяет хранить и обрабатывать не только традиционные структурированные данные, но и более сложные, такие как документы и мультимедиа.
* Высокая производительность: PostgreSQL поддерживает параллельную обработку запросов, индексы и механизмы, которые ускоряют поиск и манипуляцию данными.
* Масштабируемость и гибкость: СУБД поддерживает вертикальное и горизонтальное масштабирование, что позволяет эффективно работать как с малыми, так и с очень большими объёмами данных.
* Высокий уровень безопасности: PostgreSQL предоставляет многочисленные механизмы для защиты данных, такие как шифрование, аутентификация и авторизация, а также поддерживает строгие права доступа.

PostgreSQL используется в разных сферах, например:

* веб-разработка: многие веб-приложения и онлайн-сервисы, такие как социальные сети, интернет-магазины и системы для обработки пользовательских данных, используют PostgreSQL как основную базу данных.
* Финансовые и аналитические системы: для финансовых учреждений, которые обрабатывают огромные объёмы транзакционных данных, PostgreSQL представляет собой мощное решение для построения надёжных и безопасных аналитических платформ.
* Геоинформационные системы: PostgreSQL, с расширением PostGIS, активно используется для создания геоинформационных систем. Это позволяет обрабатывать и хранить географические данные, такие как карты, координаты и другие пространственные объекты.

Многие отмечают, что в отличие от MySQL данная СУБД имеет хорошую и подробную документацию, которая дает ответы практически на все вопросы.

О том, что это более масштабная, чем MySQL, СУБД, говорит и тот факт, что PostgreSQL периодически сравнивают с такой мощной системой управления данных, как Oracle. Все это позволяет говорить о PostgreSQL как об одной из самых продвинутых СУБД на данный момент.

* Oracle - система, отличающаяся стабильностью уже не один десяток лет, поэтому ее выбирают корпорации, для которых важна надежность восстановления после сбоев, отлаженная процедура бэкапа, возможность масштабирования и другие ценные возможности. К тому же эта СУБД обеспечивает отличную безопасность и эффектную защиту данных.

Эта СУБД относится к объектно-реляционному типу. Название произошло от названия разработавшей эту систему фирмы Oracle. Наравне с SQL СУБД использует процедурное расширение под названием PL/SQL, а также язык Java.

В отличие от других СУБД, стоимость покупки и использования Oracle достаточно высока, и именно это зачастую является значимым препятствием к ее использованию в небольших фирмах. Вероятно, именно это также является причиной того, что в рейтинге лучших СУБД на 2016 год в России Oracle находится лишь на 6-м месте.

Некоторые возможности Oracle Database:

* различные варианты хранения данных: локальная, облачная и гибридная модель. Система может хранить данные в реляционной, документальной, графовой, объектной модели, а также в модели «ключ — значение».
* Автоматическая репликация данных.
* Автоматическое восстановление повреждённых данных.
* Автоматическое масштабирование базы данных при увеличении нагрузки.
* Поддержка транзакций — последовательностей операторов, которые рассматриваются как неделимое действие.

Некоторые преимущества:

* возможности сжатия данных и поддержка ограничений ссылочной целостности, которые обеспечивают порядок и сохранение производительности в управлении большими массивами данных.
* Функция Database In-Memory, с которой Oracle Database может обрабатывать миллиарды строк данных в секунду.
* Технология машинного обучения с анализом данных в реальном времени — Oracle Machine Learning.
* Поддержка всех популярных языков программирования.
* Наличие многофункционального инструмента управления базами данных — Enterprise Manager.

Некоторые недостатки Oracle Database:

* Сложность настройки и администрирования. Oracle — мощная и сложная система, требующая высокой квалификации от администратора.
* Требовательность к ресурсам. Oracle потребляет значительные вычислительные ресурсы, особенно в больших установках.
* Длительное обучение. Поскольку Oracle предлагает множество функций и возможностей, для полного освоения системы требуется значительное время.
* MongoDB - эта СУБД отличается тем, что она предназначена для хранения иерархических структур данных, и поэтому ее называют документно-ориентированной (она представляет собой документное хранилище без использования таблиц или схем). MongoDB имеет открытый исходный код.

Используя идентификатор, вы можете производить быстрые операции над объектом. Также эта СУБД хорошо показывает себя и при сложных взаимодействиях. В первую очередь речь идет о быстродействии – в некоторых случаях приложение, написанное на MongoDB, будет работать быстрее, чем такое же приложение, использующее SQL, т.к. MongoDB относится к классу СУБД NoSQL и пользуется объектным языком запросов, который значительно легче SQL.

Однако этот язык имеет и свои ограничения, и потому MongoDB следует использовать в случаях, когда нет необходимости в сложных и нетривиальных выборках.

Некоторые особенности MongoDB:

* гибкость данных: не требуется строгая схема;
* поддержка вложенности: легко работать с вложенными объектами и массивами;
* масштабируемость: горизонтальное масштабирование через шардинг;
* высокая производительность: встроенные индексы ускоряют запросы;
* поддержка транзакций: начиная с версии 4.0, MongoDB поддерживает многофазные транзакции;

Некоторые области применения MongoDB:

* разработка веб-приложений: хранение данных пользователей, контента, логов и т. д.
* Аналитика больших данных (Big Data): хранение и анализ больших объёмов неструктурированных данных.
* Работа с IoT-технологией: содержит данные, получаемые от IoT-устройств, может обрабатывать огромные объёмы данных с высокой скоростью.
* Работа с системой управления контентом (CMS): хранение мультимедиа, метаданных и другой информации.
* Разработка мобильных приложений: благодаря возможности офлайн-синхронизации (с использованием MongoDB Realm) базу данных часто применяют при создании кроссплатформенных приложений.

Из широкого спектра СУБД была выбрана PostgreSQL, так как она имеет:

* лёгкую настройку безопасности: наличие специальных расширений, таких как pg\_crypto, которое предоставляет набор функций для создания симметричных и асимметричных ключей шифрования; настройка параметров подключения к базе данных через текстовый файл, а именно указания пользователя, протокол шифрования при передаче данных, адрес(а) подключения.
* Простой и популярный язык запросов SQL.
* Драйвер (psql) для работы с базой данных из консоли операционной системы, который можно скачать отдельно от полного комплекта PostgreSQL, что удобно, если нужно взаимодействовать с ней удалённо с устройства, на котором нет ПО СУБД, так как время установки psql будет в разы меньше, чем время для всего комплекта программ.

## 1.5. Обзор и выбор языка программирования

Для хорошей работы приложения нужен производительный язык программирования, так как оно работает с большой базой данных предприятия и должно быстро обрабатывать информацию.

Рассмотрим некоторые варианты из данной категории.

* Язык Go был разработан для создания высокоэффективных программ. Его можно рассматривать как более современного конкурента для языков C и C++, однако Go сначала появился как язык программирования компании Google для решения проблем разработки в самом учреждении:
* медленная компиляция программ;
* неконтролируемые зависимости;
* затруднения с читаемостью кода;
* сложность понимания документации;
* дублирование разработок;
* высокая стоимость обновлений;
* отсутствие синхронизации обновления при дублировании кода;
* трудоёмкость разработки инструментария;
* проблемы межъязыкового взаимодействия.

Создатели Go отказались от обработчиков исключений, потому что, по их мнению, из-за них разработчики чаще ошибаются при написании кода. На смену им был создан специальный интерфейс error для проверки кодов возвратов функций. Некоторые затратные операции, такие как вставка элемента в середину массива, были вырезаны, так как они, как считают авторы, может провоцировать программиста на создание неэффективного кода.

Особенности разработки программы:

* алфавит: язык является регистрозависимым.
* Система модулей: чтобы не было путаницы при обновлении пакета, специально разработанная система сборки проверяет удовлетворённость всех зависимостей, и, если такого не будет, то она блокирует внесение изменений.
* Псевдопеременная «\_»: в Go объявление локальной переменной без дальнейшего её использования является синтаксической ошибкой и для этого была предопределена псевдопеременная «\_», и значения, которые в неё попадут просто исчезнут. Так программист указывает компилятору, что данные сознательно проигнорированы.
* Отложенные функции: благодаря ключевому слову defer разработчик может указать функцию, которую необходимо вызвать перед тем, как компилятор покинет текущую зону видимости.

Встроенные типы данных:

* целые числа (разновидности int и uint) со знаком и без знака;
* числовой тип данных byte для работы с неформатированными бинарными данными;
* числовой тип данных rune для представления символа в кодировке Unicode;
* числа с плавающей запятой;
* комплексные числа (complex64 и complex128);
* строки в кодировке UTF-8 (string);
* логические значения истины и лжи (bool).
* C++ - универсальный язык программирования. Он подходит для индивидуальной и коммерческой разработки, создания операционных систем и игровых движков. Является объектно-ориентированным и сочетает свойства высокоуровневых и низкоуровневых языков, имеет огромную стандартную библиотеку с функционалом от работы с динамическими массивами до поддержки многопоточности программы. Сообщество по сей день выпускает не менее «слабые» библиотеки, такие как boost, которые вполне могут конкурировать со стандартной, но зачастую их используют вместе, объединяя достоинства и устраняя недостатки. На нём можно написать как прикладную программу, так и драйверы устройств. В основном именно на нём работает большая часть военного сектора страны, либо на языках, основанных на нём. Он перенял синтаксис от языка C и до сих пор авторы хотят сохранить их совместимость.

Данный язык славится своей возможностью тонко работать с памятью. Он позволяет выделять в ней место под какие-либо данные, можно явно указать расположить их последовательно с определённым размером под каждый блок данных. Это позволяет создать очень хорошую оптимизацию для приложения. Обращаться и изменять такие данные можно через указатели – переменные, которые хранят их адрес в памяти. Но «большая сила – большая ответственность»: выделяя память, нужно не забывать её своевременно освобождать, ведь, если этого не сделать, то может произойти утечка памяти – программа займёт всю возможную память устройства, и тогда аварийно завершится либо сама программа, либо операционная система.

Некоторые отличия от других языков:

* виртуальные функции: разработчик библиотеки может написать ключевое слово virtual перед методом класса (функцию принято называть методом, если она находится внутри класса) для указания того, что программист, использующий эту библиотеку, может сам переопределить её, то есть написать свой алгоритм работы этого метода. Также существуют чисто виртуальные функции – функции, которые необходимо переопределить, иначе использование экземпляра этого класса невозможно. Обычно они используются в классе модели какого-либо элемента пользовательского интерфейса. Это явно указывает программисту, что он должен сам описать, как элемент будет получать и обрабатывать данные, но, так как этот метод уже объявлен в классе модели элемента, то нет необходимости говорить программе, когда его вызывать, ведь она знает про его существование, а этого достаточно.
* Перегрузка функций: в случае, если в зависимости от количества или типов аргументов, функция должна выполнять разные действия, то можно её перегрузить – объявить и описать две и более функции с одним именем, но с разными аргументами. Например, можно реализовать две функции умножения: первая будет принимать два числа и перемножать их, а вторая, которая будет вызвана вместо первой, если будет передано только одно число, умножит это число само на себя.
* Шаблоны: если программисту нужно указать, что в алгоритме может использоваться любой тип данных, он может сделать это через структуру шаблонов. Например, чтобы не перегружать функцию сложения для каждого возможного типа данных, перед ней нужно создать шаблон и указать его как тип данных аргументов. Но нужно не забывать защищать такие функции от передачи в них не подходящих по типу данных, так как компилятор не сможет предупредить разработчика о несоответствии типов данных и операций с ними, а программа аварийно завершится.
* Rust – язык программирования, совмещающий в себе функциональное и процедурное программирование с объектно-ориентированным (ООП). Если программист пишет именно на нём, то скорее всего его волнует скорость и безопасность работы. Его главное отличие от C++ - безопасная работа с памятью, которая достигается встроенной в компилятор системой статической проверки ссылок.

Особенности языка программирования:

* ООП: в Rust немного изменена суть объектно-ориентированного программирования, так как основана на типажах – абстрактных типах данных, содержащих только методы. Присутствует наследование типажей – типаж может зависеть от других и нуждаться в их реализации. Шаблоны из C++ заменены обобщенными типами данных, конструкторы которых могут возвращать не свой тип данных. Они эффективно обрабатываются компилятором и имеют почти ту же производительность, что и в упомянутом языке.
* Параллелизм вычислений: асинхронные операции – операции, которые могут выполняться одновременно и даже обмениваться данными. Определение и вызов асинхронных операций осуществляется с помощью ключевого слова async, которое определяет асинхронную функцию или блок; обычный вызов такой функции возвращает объект с типажом Future — дескриптор ленивой асинхронной операции (операции, выполнение которой откладывается до нужды в её работе или результате). Вызов .await позволяет одной асинхронной операции ждать, пока не завершится другая асинхронная операция.

Для разработки приложения был выбран язык программирования C++, так как, благодаря возможностям работы с памятью, он один из самых производительных и позволяет обеспечить высокий уровень безопасности, ограничивая доступ к каким-либо значениям из вне. Его инструментарий для работы с многопоточностью позволяет реализовать чётко структурированное программное обеспечение, которое может одновременно выполнять множество операций, а в случае простаивания потока с функциями без дела или его возможного переполнения, поставить поток на ожидание или очистить его соответственно. Он имеет обширный каталог библиотек для работы с любой базой данных и обработкой большого объёма информации.

## 1.6. Обзор и выбор среды разработки

Интегрированная среда разработки (IDE) — это комплект взаимосвязанного программного обеспечения для написания и отлаживания кода.

В основном IDE состоит из:

* текстового редактора для написания кода, также помогает составить удобную иерархию файлов в проекте, изначально предлагая один из удобных вариантов.
* Конструктора пользовательского интерфейса, который позволяет программисту не писать код для создания и перемещения того или иного элемента, например, кнопки, а просто перетащить его из общего списка в нужное место на форму – макет приложения.
* Компилятора или интерпретатора, то есть программы-переводчика, позволяющей перевести текст, написанный на языке программирования, в набор машинных кодов.
* Отладчика, который проверяет код и помогает устранить ошибки, предоставляя возможность приостановить работу программы на определённой строке кода.

Во многих средах разработки в процессе сборки программы встраивается сборщик мусора – специальная утилита для поиска не использующихся данных в приложении и освобождении места в памяти, которое они занимают, путём их удаления, что повышает безопасность и оптимизацию. В современных версиях дополнительно можно установить искусственный интеллект, который поможет дописать блок кода, но пока он не совершенен и часто приходится переписывать после него.

Большое значение при выборе IDE – то, что она использует для перевода написанного кода на машинный: компилятор или интерпретатор.

Различия в их работе:

* + интерпретатор запускает программу сразу, сколько бы строк она не содержала, но её работа будет не самой быстрой, так как код обрабатывается только тогда, когда до него дойдёт ход выполнения приложения. Компилятор, напротив, тратит больше времени на обработку и перевод кода, сборку бинарного исполняемого файла, и только потом можно запустить программу, но данная подготовка оправдывается, ведь приложение теперь всегда будет работать на машинном коде, не тратя время на повторный перевод.
  + Благодаря тому, что компилятор сначала собирает программу, можно обнаруживать ошибки до запуска и исправлять их, иначе сборка не будет успешной, а в случае с интерпретатором они дадут о себе знать, только если пользователь сам наткнётся на них в ходе выполнения программы.
  + Для крупных решений определённо нужна платформенная независимость, чтобы приложение работало на любой операционной системе. С одной стороны, программист может написать только программный код, и в зависимости от ОС клиенту нужно скачать подходящий интерпретатор для его запуска. С другой, чтобы пользователям не доставлять лишних забот, можно воспользоваться системой сборки для компилятора, которая представляет собой инструмент со своим собственным алгоритмическим языком, выучив который, можно написать инструкцию по сборке приложения с указанием использующихся компонентов. После этого такой инструмент по написанным алгоритмам поможет компилятору сделать программу, которая будет запускаться практически на любой платформе.

Итоговое приложение должно иметь графический интерфейс, работать максимально быстро и на большинстве операционных систем. Выбранный язык программирования C++ отлично подходит для этих целей. Остаётся выбрать IDE, которое поспособствует разработке.

* + Microsoft Visual Studio имеет несколько полезных функций, дополняющих возможности написания кода: IntelliSense – технология, дописывающая имена функций и блоки кода при вводе начальных символов, также содержит ссылки на документации функций и классов; Copilot – искусственный интеллект, помогающий написать код: предлагает варианты для автозаполнения функций соответствующими аргументами, либо составляет целые ветвления и циклы за программиста.

Предоставляет инструменты для разработки как консольных приложений, так и полноценных с графическим интерфейсом. Второй вариант поддерживается технологиями WPF и Windows Forms.

* + Windows Forms — это технология, позволяющая создавать приложения с графическим интерфейсом пользователя и являющаяся частью Microsoft .NET Framework. Она упрощает доступ к элементам интерфейса Microsoft Windows за счёт создания обёртки для существующего Win32 API в управляемом коде.

Существенным минусом данной технологии является отсутствие кроссплатформенности, то есть программа будет работать только на операционной системе Windows.

* + Windows Presentation Foundation (WPF) — это библиотека для создания кроссплатформенных программ с пользовательским интерфейсом.

WPF имеет большой набор функций разработки приложений: элементы управления, макет, двумерную и трёхмерную графику, стили, документы, мультимедиа. Для описания свойств элементов и их расположения в интерфейсе используется язык разметки XAML. WPF использует векторную графику для визуализации интерфейса, что обеспечивает высокое качество изображения на любых устройствах и при любом разрешении экрана.

Недостаток этой библиотеки – для разработки программы обычно используется язык программирования C#. Можно писать и на C++, но на специальной версии для этого от компании Microsoft: Visual C++/CLI, однако в последнее время он плохо поддерживается.

* + Qt — фреймворк для разработки кроссплатформенного программного обеспечения. Он предоставляет инструменты и библиотеки для создания графических пользовательских интерфейсов, а также для разработки сетевых и консольных приложений, работы с базами данных, многопоточности и других задач. В данной среде можно создавать настольные и мобильные приложения, даже некоторые игры.

Основные компоненты Qt:

* + Qt Creator – редактор кода (IDE).
  + Qt Designer – конструктор графического интерфейса, позволяющий создавать приложения, не прописывая соответствующий код вручную, а перетаскивая нужные элементы (кнопки, поля для текстового ввода) из общего списка на форму.

Среда разработки содержит свои аналоги стандартных библиотек C++, которые учитывают ошибки своих предшественников, а также драйверы для взаимодействия с базами данных.

Для написания кода и графического интерфейса приложения была выбрана среда Qt Creator, так как предоставляет возможность использовать для этого язык программирования C++. Форма окон программы будет практически одинакова отображаться на всех операционных системах, исключение составляют кнопки минимизации, максимизации и выхода из приложения, которые могут быть настроены непосредственно в настройках ОС. В параметрах IDE можно удобно переключаться между комплектами, состоящими из компилятора, отладчика и системы сборки, что позволяет определить более подходящий вариант для создания программного обеспечения.

## 1.7. Обзор и выбор системы сборки

Это система, которая автоматизирует сборку проекта по прописанным правилам: версия стандарта языка программирования, целевые операционные системы, подключаемые модули и т. д. Она определяет зависимости между файлами с исходным кодом и программой на выходе и выполняет компиляцию.

Существует три популярных системы сборки под выбранные язык и IDE.

* + CMake – программное обеспечение для автоматизации сборки программы из исходного кода. Параметры прописываются на специальном языке сценариев схожим с языком программирования в созданном разработчиком документе CMakeLists.txt и, если все файлы с кодом и подключаемыми модулями указаны верно, то при запуске компиляции он сгенерирует файлы сборки для создания приложения. Главная особенность – размещение выходных файлов (конфигурационные, файлы запуска) вне каталога-проекта, что позволяет компилировать программу с разными параметрами сборки.

Синтаксис языка более сложен, чем у его аналогов, но в отличие от них, имеет больше возможностей, так как позволяет делать программы кроссплатформенными и запускать на любой операционной системе. Cmake поддерживают многие среды разработки: Visual Studio, Qt Creator и др., а также большое количество компиляторов, в которое входят Clang, GNU GCC, MSVC, SunPro и Intel C++.

* + Более простой в освоении вариант – QMake. Это утилита, которая устанавливается вместе с Qt Creator. Принцип работы схож: генерирует файлы сборки по правилам, прописанным в файле \*.pro, отличается своим языком сценариев, который легче, чем у CMake. Благодаря интеграции с Qt, подключение модулей среды разработки, такие как QSQL для работы с базами данных, становится очень простой задачей, как и обработка ресурсов и форм.

Но существует и ряд недостатков. Начиная с шестой версии фреймворка, QMake почти не поддерживается и считается устаревшим. Также, хоть и встроенные библиотеки проще подключать этой системой сборки, сторонние могут оказаться проблемой для интеграции, так как она менее гибкая по сравнению с другими.

* + Для решений, где важны скорость и эффективность сборки, была создана система Ninja. Она хорошо показывает себя при повторных сборках проекта, когда разработчики часто вносят изменения в код. Главная особенность – низкоуровневость: нуждается в дополнительной системе сборки для описания целей, что является как плюсом, ведь её можно использовать для ускорения процесса, так и минусом, предоставляющим дополнительную сложность написания сценария сборки.

В ходе работы с Ninja могут возникнуть следующие трудности:

* + Отсутствие информации о прогрессе. Система не ведёт подробный журнал процесса сборки, из-за чего могут появиться проблемы с поиском ошибок.
  + Сложность синтаксиса. Он очень сложен и не рекомендуется для ручного написания. Желательно использовать Ninja вместе с другими системами сборки, например, с CMake.
  + Переполнение памяти. Ninja собирает приложение в параллельном режиме, используя все ресурсы устройства, поэтому лучше ограничивать количество потоков сборки.

Для проекта был выбран комплект из систем сборок CMake и Ninja, где первая собирает программу и даёт ей свойство кроссплатформенности, а вторая ускоряет этот процесс.

## 1.8. Этапы разработки

Информационная система должна работать с базой данных, где хранятся сведения для аутентификации пользователя, список заказов, отделы, рабочая информация о сотрудниках. Приложение должно отображать эту информацию и предоставлять возможность работать со списком проектов, подчинённых и БД. Во время обмена данными этими двумя компонентами системы они должны быть зашифрованы алгоритмом sha-256. Пароли в базе должны быть также зашифрованы с помощью симметричного ключа.

Для создания приложения был выбран следующий набор технологий: PostgreSQL для создания базы данных; язык программирования C++, среда разработки Qt Creator и система сборки из CMake и Ninja, которые обеспечат всё для написания оптимизированной программы для любой операционной системы.

На этапе разработки была создана база данных. Написан функционал приложения, создан его графический интерфейс, осуществлено подключение к БД. Для удобной разработки была использована система контроля версий Git, взаимодействие с которой производилось через ПО Git Bash.

Тестирование осуществлялось с помощью написания тест-кейсов. В них была отображена качественная обработка исключений приложением в случаях, если аутентификация не была пройдена или отсутствует соединение с БД.

# ГЛАВА 2. РАЗРАБОТКА ПРОГРАММНОГО ПРОДКУТА

## 2.1. Проектирование базы данных

Для хранения информации была создана база данных ACMS (access control and management system) с помощью программного обеспечения PostgreSQL 17, которое использует язык запросов SQL. Использовано расширение pg\_crypto для шифрования паролей.

В конфигурационном файле СУБД pg\_hba.conf были прописаны параметры удалённого подключения, а именно IP-адрес хоста, пользователь базы данных, алгоритм шифрования при обмене информацией.



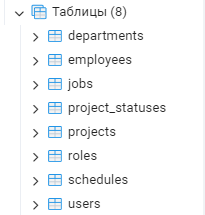


Рисунок 1 - База данных «ACMS»

Данные хранятся в восьми таблицах, описанных ниже.

* + departments.

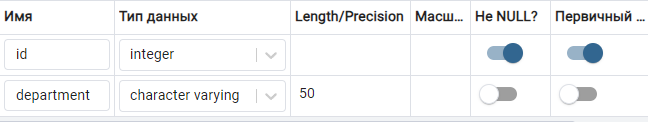


Рисунок 2 - Таблица «departments»

Данная таблица отображает отделы в учреждении.

* + employees.

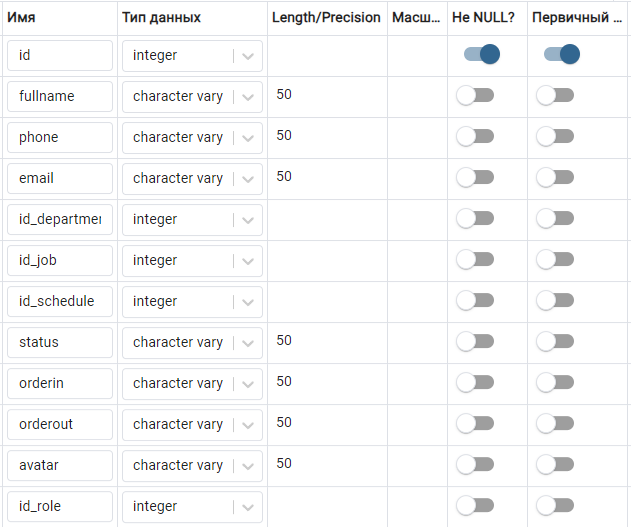


Рисунок 3 - Таблица «employees»

Эта таблица содержит список сотрудников и всю их рабочую информацию.

* + jobs.



Рисунок 4 - Таблица «jobs»

Данная таблица хранит должности в учреждении.

* + project\_statuses.



Рисунок 5 - Таблица «project\_statuses»

Эта таблица отображает все возможные статусы разработки проекта.

* + projects.

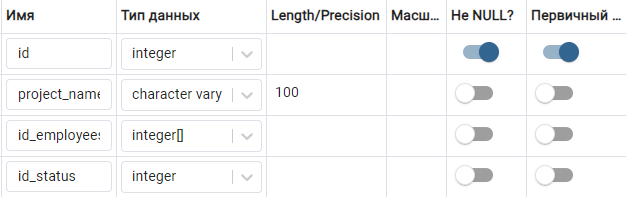


Рисунок 6 - Таблица «projects»

Данная таблица содержит список заказов и их статусы.

* + roles.

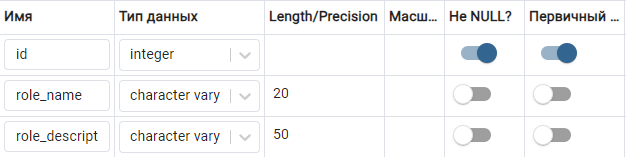


Рисунок 7 - Таблица «roles»

Эта таблица хранит информацию о всех ролях БД и их описания.

* + schedules.



Рисунок 8 - Таблица «schedules»

Данная таблица отображает информацию о возможных графиках работы на предприятии.

* + users.

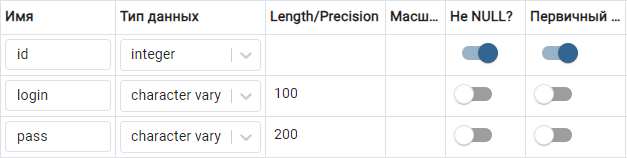


Рисунок 9 - Таблица «users»

Эта таблица содержит логины и пароли сотрудников для авторизации в приложении.

В базу данных были внесены роли пользователей: обычные сотрудники предприятия db\_user с привилегией SELECT ко всем таблицам кроме таблицы projects (SELECT, INSERT, UPDATE) и администратор БД manager со всеми возможными правами.

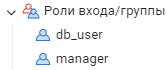


Рисунок 10 – Роли в базе данных

Так как в PostgreSQL 17 отсутствует тип данных для счётчика serial, столбцы id имеют тип int, а их автоинкремент реализован с помощью последовательностей для каждой таблицы, имеющие названия <таблица>\_id\_seq.

После всех настроек и расстановки связей между таблицами база данных приобрела следующий вид.

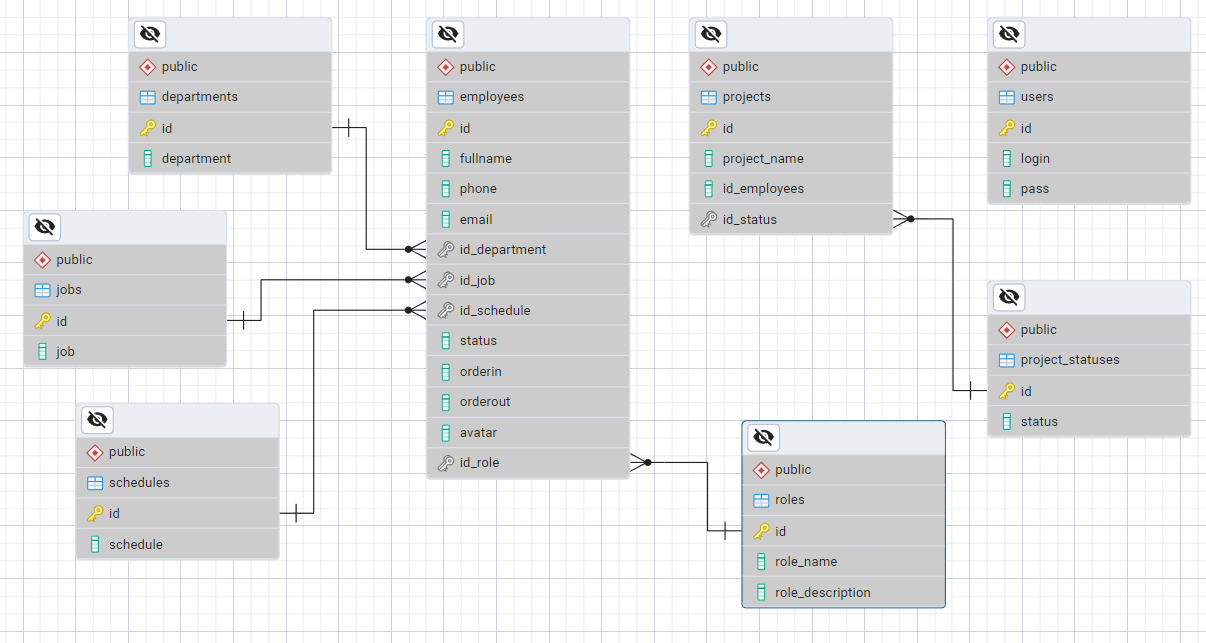


Рисунок 11 – Схема базы данных

## 2.2. Разработка пользовательского интерфейса

Во время разработки приложения не стоит забывать про безопасность доступа к данным. Это можно гарантировать, если подключаться к базе данных с разным уровнем доступа в зависимости от требований программы. Об этом будет упомянуто в описании работы приложения. В процессе дизайна было выбрано использование градиентов: задний фон – от тёмно-фиолетового к светло-фиолетовому, цвет кнопок – оттенки оранжевого или голубого в отдельном случае, цвет текста просто белый, если он отображает какую-то информацию, если его вводят с клавиатуры – чёрный.

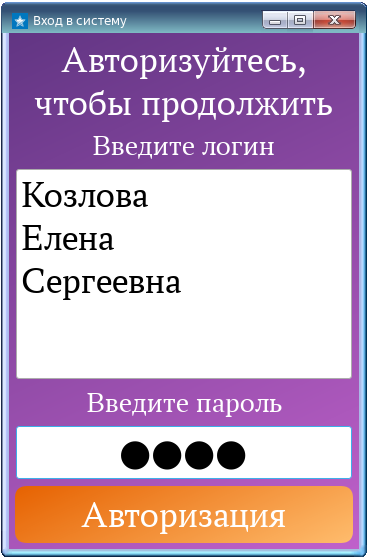


Рисунок 12 – Окно авторизации

При запуске приложения пользователю необходимо авторизоваться для дальнейшей работы. Для этого нужно ввести свой логин, который является ФИО сотрудника, и пароль, который он узнаёт после принятия на работу в организацию и внесения работника в базу данных. Чтобы было легче читать логин, например, для проверки правильности введённых данных, каждый раз, когда пользователь ставит пробел, он заменяется переносом курсора на следующую строчку. На этом этапе система имеет максимальный уровень доступа для аутентификации сотрудника. Сразу после успешной проверки программа определяет должность пользователя в учреждении и в зависимости от этого составляет SQL-запрос для получения только необходимой информации; отсоединяется от БД и уничтожает из своей памяти введённые данные для авторизации.

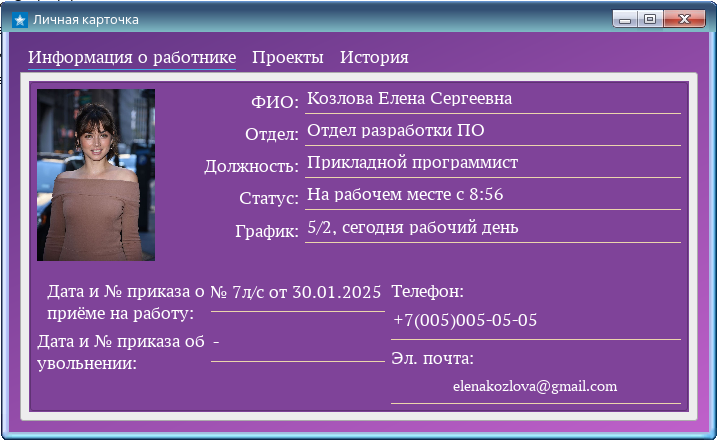


Рисунок 13 – «Личная карточка» сотрудника

Если система определила, что пользователь является обычным работником предприятия без подчинённых, то она подключится к базе данных, используя роль db\_user, имеющую минимальные привилегии. Программа откроет окно «Личная карточка» с тремя вкладками «Информация о работнике», «Проекты» и «История». Информация из ранее полученного результата запроса отобразится на первой вкладке, в том числе и фотография сотрудника.

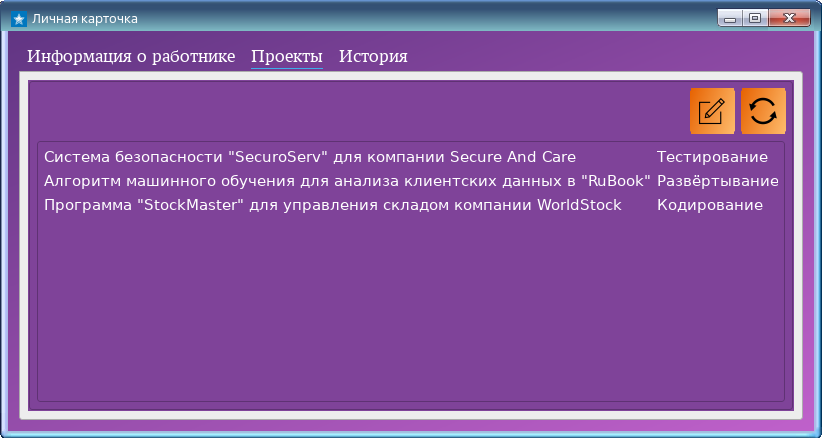


Рисунок 14 – Вкладка «Проекты»

На следующей странице работник увидит список проектов, в которых он участвует. Он может обновить информацию, нажав на соответствующую кнопку, или изменить статус проекта, включив режим редактирования.

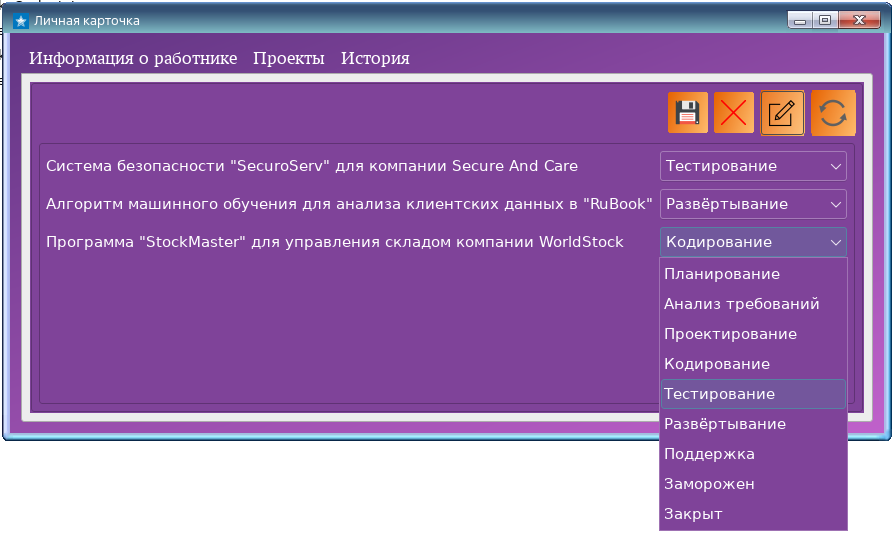


Рисунок 15 – Режим редактирования статусов проектов

В данном режиме сотруднику на выбор предоставляется список возможных статусов проекта. Он может указать любой и сохранить изменения.

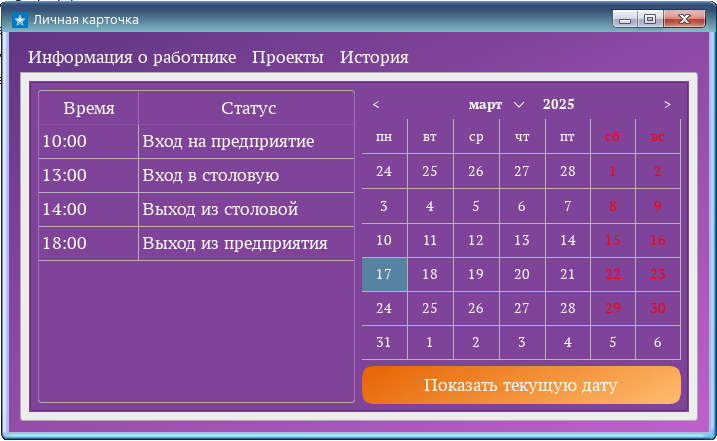


Рисунок 16 – История посещения сотрудника 17.03.2025

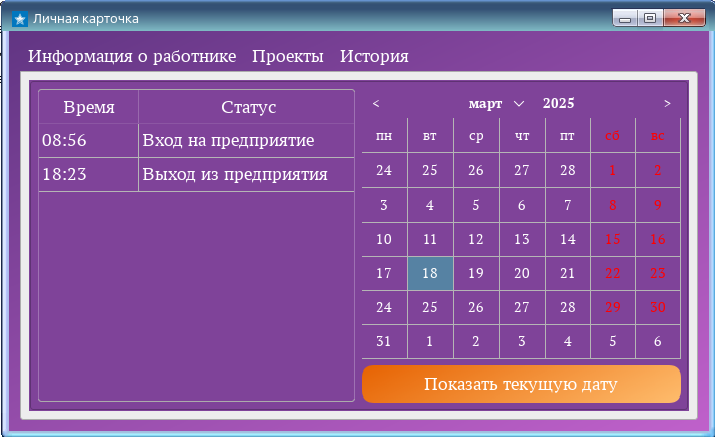


Рисунок 17 – История посещения сотрудника 18.03.2025

На последней вкладке работник может просмотреть свою историю посещения предприятия и его некоторых зон, переключаясь между датами с помощью удобного встроенного календаря.

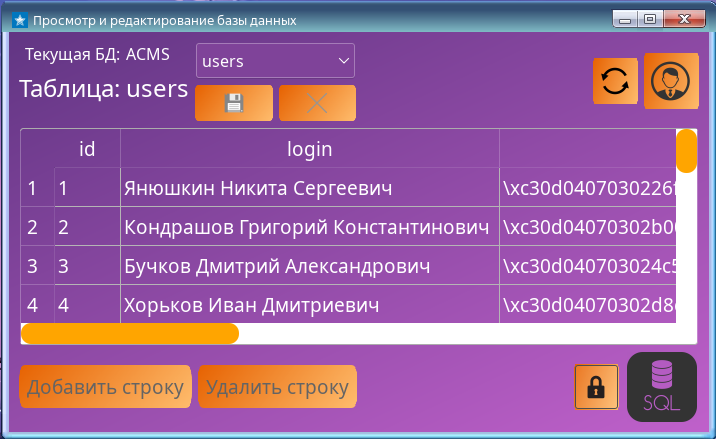


Рисунок 18 – Окно просмотра и редактирования базы данных

Если в системе авторизовался администратор БД, то на экран выводится соответствующее окно, предоставляющее возможности работы с таблицами базы данных учреждения. Как и любую другую форму, её можно растягивать вплоть до размера полного экрана, что только облегчит читаемость текста. Данное окно позволяет переключаться между таблицами, принудительно обновить информацию, если в базе данных могли быть изменения. Администратор может открыть свою «Личную карточку», диалоговое окно для создания и выполнения какого-либо SQL-запроса. Открытую таблицу можно редактировать, переключив режим с чтения на изменение, нажав на кнопку с изображением замочка.

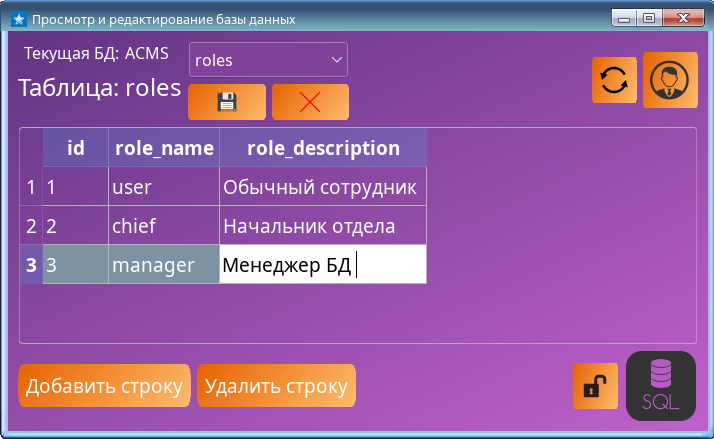


Рисунок 19 – Редактирование таблицы roles

В режиме редактирования есть возможность добавить строку со значениями по умолчанию или удалить выбранную, изменить текст отдельной ячейки, после чего применить или отклонить изменения, нажав на соответствующую кнопку и подтвердив действие в появившемся информационном диалоговом окне.

Если текущая таблица «грязная», то есть содержит несохранённые изменения, а пользователь нажмёт на кнопку переключения режима на «только чтение», то приложение спросит, что делать с этими данными: сохранить и продолжить или отклонить.

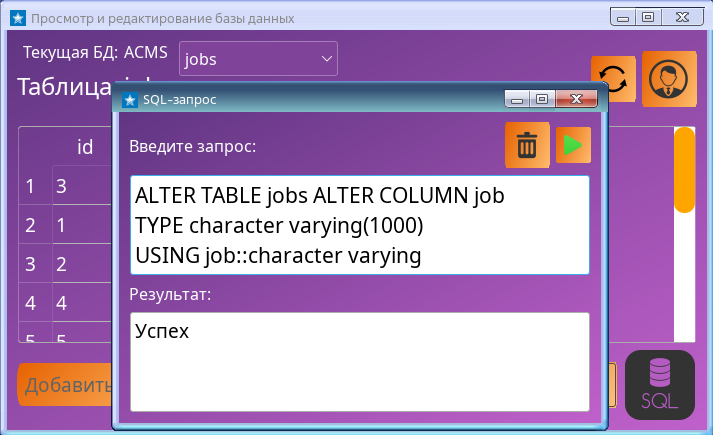


Рисунок 20 – Создание и выполнение SQL-запроса

При необходимости выполнить какой-нибудь SQL-запрос, администратор может перейти в подходящее окно, нажав на кнопку «SQL». Данная форма имеет всего два поля для запроса и вывода результата: успех или ошибка, а также кнопки очистки поля ввода и запуска скрипта. Окно сохраняет своё состояние даже после своего закрытия, что значит, если администратор снова его откроет, последний введённый запрос останется для повторного запуска или изменения.

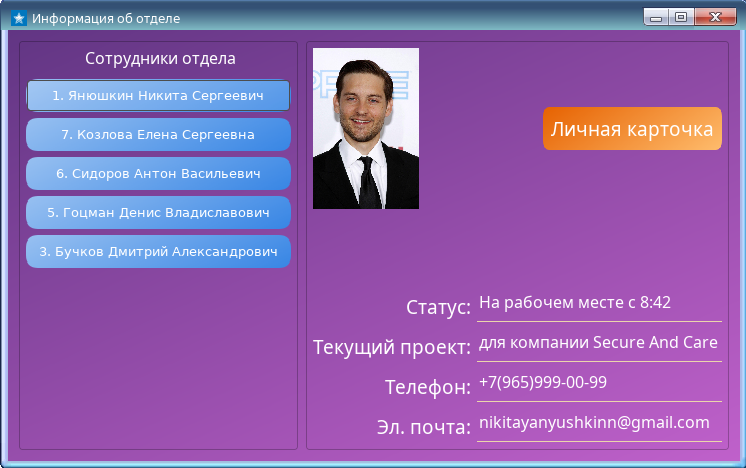


Рисунок 21 – Окно начальника отдела

Если система определяет, что авторизуется начальник какого-либо отдела, она отображает окно со списком, который состоит из него и подчинённых, и их основной информацией, а также предоставляет возможность открыть «личную карточку» работника.

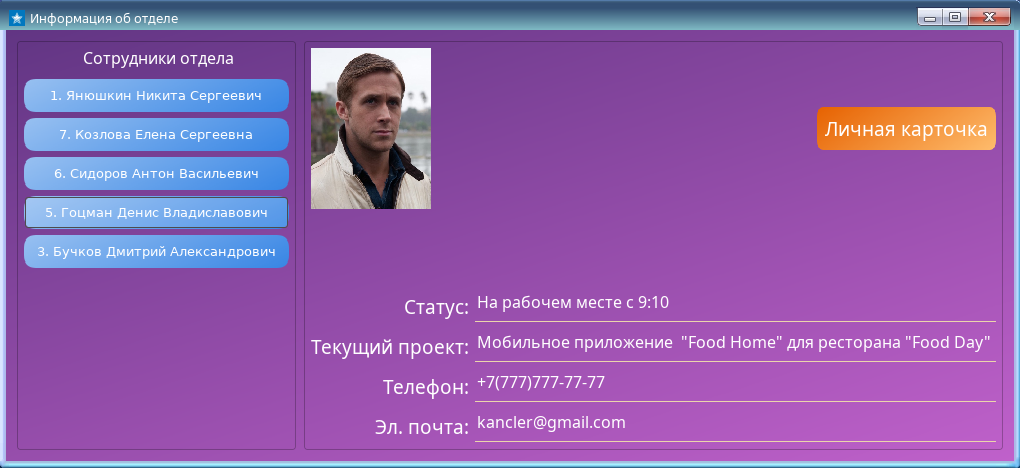


Рисунок 22 – Информация о подчинённом

Чтобы посмотреть сведения о любом подчинённом, достаточно нажать на его ФИО. Число, стоящее рядом с именем, является его личным идентификатором в базе данных, который может пригодиться в будущем. Это окно также растягивается на экране, что может пригодиться для полного отображения текста.

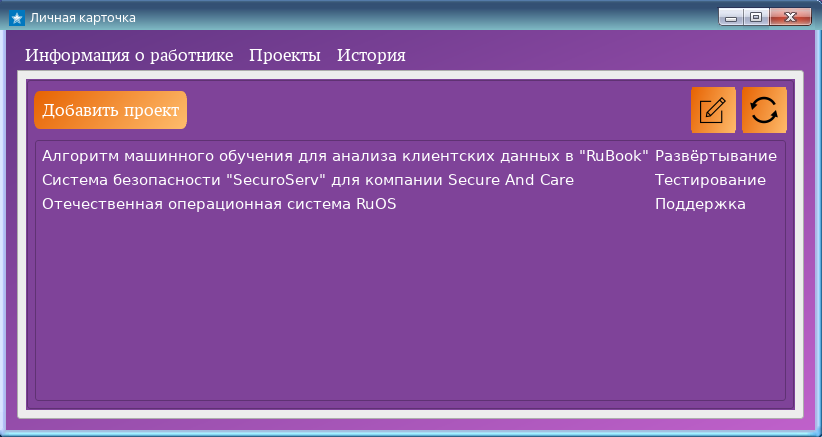


Рисунок 23 – Список проектов начальника отдела

В «личной карточке» начальника отдела есть небольшое изменение – кнопка «Добавить проект», где он может внести в БД новый проект.

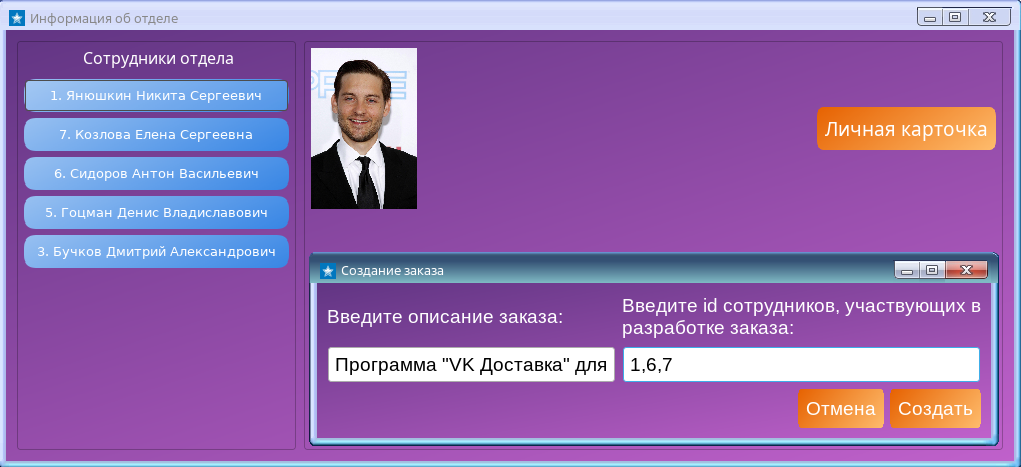


Рисунок 24 – Окно «Создание заказа»

В появившейся форме сотрудник должен указать описание заказа и личные идентификаторы, отображающиеся возле имени, работников отдела через запятую, которые будут участвовать в его разработке.

## 2.3. Тестирование программного обеспечения

Чтобы предотвратить аварийное завершение программы по какой-либо причине, обработка исключений в исходном коде – частая практика, она встречается везде, где может произойти ошибка.

Далее будут разобраны два самых частых случая: указан неправильный логин или пароль; отсутствует соединение с базой данных.

Таблица 1

|  |  |
| --- | --- |
| Идентификатор тест-кейса | TC001 |
| Название | Проверка авторизации пользователя |
| Описание | Тест проверяет работоспособность программы при указании неверных данных для авторизации |
| Шаги выполнения | В окне «Вход в систему» ввести неправильный логин и пароль, нажать кнопку «Авторизация» |

Продолжение таблицы 1

|  |  |
| --- | --- |
| Ожидаемый результат | Появляется окно с уведомлением, что пользователь ввёл неправильный логин или пароль |
| Фактический результат | Появляется окно с уведомлением, что пользователь ввёл неправильный логин или пароль |
| Статус | Пройден |

Также был разработан юнит-тест для функции connectToDBWithManager класса Connection, которая устанавливает соединение с базой данных, используя роль администратора. В функции открывается файл host.txt, который содержит IP-адрес хоста. Метод возвращает значение true, если подключение установлено, в противном случае – false.

В первом случае файл host.txt есть и тестирование выводит следующее:

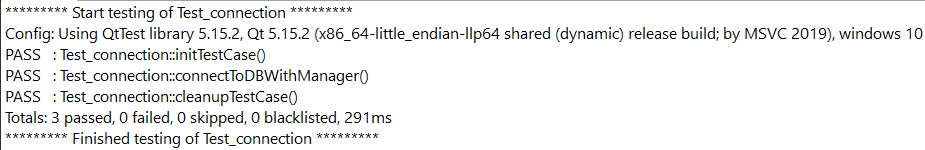


Рисунок 25 – Успешное прохождение юнит-теста

Во втором файл отсутствует и результат уже другой:

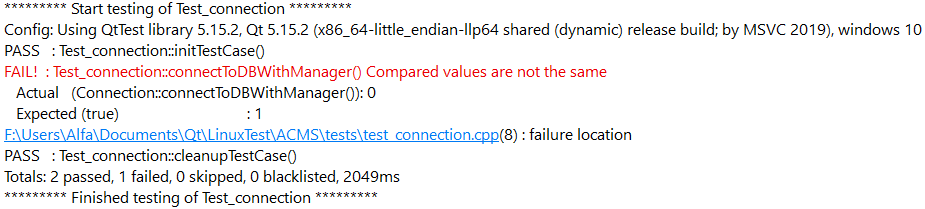


Рисунок 26 – Неуспешное прохождение юнит-теста

# ЗАКЛЮЧЕНИЕ

В процессе работы были выполнены следующие задачи:

1. рассмотреть понятие информационной системы и её классификацию;
2. обозреть популярные системы управления базами данных, языки программирования, среды разработки, системы сборки и выбрать наиболее подходящие;
3. описать этапы разработки информационной системы;
4. создать информационную систему с помощью выбранных технологий;
5. протестировать информационную систему;
6. сделать выводы после окончания работ.

Была достигнута поставленная цель разработки информационной системы для системы контроля и управления доступом предприятия.

В программу планируется встроить свою кэш-систему для большей оптимизации, добавить чат между работниками, который будет иметь высокий уровень защиты данных и работать только в пределах учреждения.

С помощью разработанной информационной системы можно получать информацию о времени нахождения того или иного сотрудника на рабочем месте; просматривать список подчинённых сотрудников и проектов, которыми они занимаются; добавлять новые заказы; просматривать и редактировать базу данных предприятия.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Огнева М. В., Кудрина Е. В. Программирование на языке C++ практический курс / Огнева М. В., Кудрина Е. В, 2022 — 336 c.
2. Герберт, Шилдт C++. Базовый курс / Шилдт Герберт. - М.: Диалектика / Вильямс, 2022 – 564 с.
3. Дейтел, Пол Как программировать на С / Пол Дейтел, Харви Дейтел. - М.: Бином, 2022 – 858 с.
4. Джамса К. Учимся программировать на языке C++ / К. Джамса. - М.: Мир, 2022 – 320 с.
5. Культин, Н. C/C++ в задачах и примерах / Н. Культин. - М.: БХВ-Петербург, 2022 – 368 с.
6. Адам Фримен ASP.NET 4.5 с примерами на C# 5.0 для профессионалов / Фримен Адам. - М.: Диалектика / Вильямс, 2021 – 2792 с.
7. Биллиг, В. А. Основы программирования на С# / В.А. Биллиг. - М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2021 – 488 с.
8. Ватсон, Б. С# 4.0 на примерах (C# 4.0. How-To) / Б. Ватсон. - М.: БХВ-Петербург, 2021 – 608 с.
9. Вагнер Билл С# Эффективное программирование / Билл Вагнер. - М.: ЛОРИ, 2021 – 320 с.
10. Культин, Н. С# в задачах и примерах / Н. Культин. - М.: БХВ-Петербург, 2020. - 1293 c.
11. Тим Макнамара Rust в действии / Тим Макнамара, 2022 – 528 с.
12. Клабник С., Николс К. Программирование на Rust / Клабник С., Николс К., 2021 – 252 с.
13. Брайан, Джонсон Основы Microsoft Visual Studio .NET 2003 / Джонсон Брайан. - М.: Русская Редакция, 2022 – 773 с.
14. Понамарев, Вячеслав Программирование на C++/C# в Visual Studio .NET 2003 / Вячеслав Понамарев. - М.: БХВ-Петербург, 2021 – 322 с.
15. Постолит Анатолий Visual Studio .NET: разработка приложений баз данных / Постолит, Анатолий. - М.: СПб: БХВ, 2021 – 544 с.
16. Свидзиньски Р. CMake для С++. Современный подход / Свидзиньски Р., 2025 – 442 с.
17. Алан Донован, Брайан Керниган Язык программирования Go / Алан Донован, Брайан Керниган, 2020 – 432 с.
18. Джей Макгаврен Изучаем Go / Джей Макгаврен, 2020 – 544 с.
19. Мэтью Титмус Облачный Go / Мэтью Титмус, 2022 – 418 с.
20. Джон Боднер Go: идиомы и паттерны проектирования / Джон Боднер, 2022 – 416 с.
21. Михалис Цукалос Golang для профи / Михалис Цукалос, 2022 – 624 с.