Содержание

[Введение 5](#_Toc200150043)

[1 Анализ предметной области 6](#_Toc200150044)

[1.1 Особенности предметной области 6](#_Toc200150045)

[1.2 Описание бизнес-процессов 11](#_Toc200150046)

2 Проектирование базы данных для системы “Контроль исполнения поручений” ……………………………………………………………………….18

[2.1 Проектирование концептуальной модели 18](#_Toc200150047)

[2.2 Проектирование логической модели базы данных 20](#_Toc200150048)

[2.3 Проектирование физической модели базы данных 23](#_Toc200150049)

[3 Разработка и реализация клиентского приложения “Контроль исполнения поручений” 25](#_Toc200150050)

[3.1 Выбор средств реализации 25](#_Toc200150051)

[3.2 Архитектура программного обеспечения 27](#_Toc200150052)

[3.2 Функциональные возможности 30](#_Toc200150053)

[Заключение 39](#_Toc200150054)

[Список использованной литературы 41](#_Toc200150055)

[Приложение А 42](#_Toc200150056)

[Приложение Б 44](#_Toc200150057)

[Приложение В 46](#_Toc200150058)

# Введение

Контроль исполнения поручений представляет собой ключевой элемент эффективного управления в организациях различного профиля и масштаба. Он обеспечивает системное отслеживание исполнения задач, поставленных руководством, способствует своевременному реагированию на отклонения и позволяет повысить исполнительскую дисциплину. В условиях возрастающей сложности бизнес-процессов, увеличения объёма поручений и числа исполнителей становится особенно актуальной задача автоматизации процедур контроля исполнения.

Целью настоящего курсового проекта является разработка информационной подсистемы, направленной на автоматизацию процессов контроля исполнения поручений в организации. Актуальность данной разработки обусловлена необходимостью минимизации рисков, связанных с несвоевременным или неполным выполнением заданий, а также потребностью в обеспечении прозрачности и подотчётности всех этапов выполнения поручений. Внедрение подобной системы позволит значительно сократить временные затраты на мониторинг выполнения задач, устранить дублирование функций, повысить управляемость и ответственность исполнителей, а также обеспечить оперативный доступ к актуальной информации о статусе выполнения поручений. Всё это в совокупности будет способствовать повышению эффективности управления организацией в целом.

# 1 Анализ предметной области

## Особенности предметной области

Контроль исполнения поручений является важнейшим элементом эффективного управления в любой организации. Он представляет собой систематический процесс отслеживания, проверки и анализа выполнения задач, поставленных руководителями различным уровням подчиненных. Данный процесс позволяет обеспечить соблюдение сроков, качество выполнения заданий, а также формирует дисциплину и прозрачность в управлении.

Предметная область контроля исполнения поручений охватывает широкий круг участников и процессов. В первую очередь это руководители, которые формируют поручения, ставят задачи и отслеживают их выполнение. Далее — исполнители, которые непосредственно отвечают за выполнение тех или иных задач. Кроме того, существует вспомогательный персонал, обеспечивающий документооборот, регистрацию и сопровождение поручений. Каждый участник этого процесса должен четко понимать свои функции и зоны ответственности, что требует формализации взаимодействия и наличия прозрачных правил.

Поручения в организации могут носить различный характер: от краткосрочных и оперативных до стратегических и долгосрочных. Они могут быть связаны с подготовкой отчетности, проведением мероприятий, согласованием документов, выполнением технических или производственных задач и многим другим. Общей чертой всех поручений является наличие определенного срока исполнения, конкретного результата и ответственного лица. Эффективное управление этими параметрами и составляет суть системы контроля исполнения.

Особенность предметной области заключается в высокой плотности информации и необходимости постоянного обновления данных. Каждый день в организациях генерируется большое количество поручений, каждое из которых требует регистрации, отслеживания и отчетности. В условиях ручного ведения учета, например в бумажных журналах или электронных таблицах, происходит быстрое накопление несистематизированной информации, что приводит к потере контроля, нарушению сроков и снижению управляемости. Отсутствие прозрачности в процессе исполнения поручений часто приводит к дублированию задач, перекладыванию ответственности и затягиванию сроков, что негативно сказывается на общем результате работы организации.

Еще одной важной особенностью предметной области является влияние человеческого фактора. Исполнители могут забыть о полученном задании, неправильно понять его суть или отложить выполнение на последний момент. Руководители, в свою очередь, не всегда имеют возможность вручную отслеживать сотни поручений, находящихся в работе. Это делает необходимым наличие вспомогательных инструментов — таких как напоминания, уведомления, автоматическое формирование отчетов и визуализация сроков исполнения. Подобные инструменты особенно актуальны в крупных организациях с разветвленной структурой и множеством подразделений, где поручения часто передаются по вертикали и горизонтали управления.

Современные требования к оперативности и точности исполнения задач требуют внедрения автоматизированных информационных систем. В рамках предметной области контроля исполнения поручений такие системы позволяют не только отслеживать статус поручений, но и анализировать эффективность работы сотрудников, выявлять узкие места в процессах, прогнозировать риски несвоевременного исполнения. Использование информационных технологий обеспечивает единый стандарт учета поручений, их регистрация в единой базе данных, автоматическое уведомление исполнителей о новых задачах и приближающихся сроках, а также формирование аналитических и итоговых отчетов для руководства.

Автоматизация контроля исполнения поручений позволяет исключить излишнюю бумажную нагрузку, минимизировать риски потери информации, повысить уровень прозрачности и управляемости процессов. Особенно актуально это в условиях высокой динамики внешней среды, когда организация должна быстро адаптироваться, а руководители принимать решения на основе достоверной и актуальной информации. Таким образом, основным вектором развития предметной области является переход от ручных и полуручных методов контроля к интегрированным цифровым платформам, способным поддерживать эффективное управление поручениями на всех уровнях организационной структуры.

В то же время при внедрении автоматизированных систем необходимо учитывать особенности конкретной организации: структуру, объем документооборота, уровень ИТ-поддержки, а также подготовленность персонала к использованию новых технологий. Любая система контроля исполнения поручений должна быть гибкой, масштабируемой и адаптируемой под изменяющиеся бизнес-потребности. Только при соблюдении этих условий возможно добиться значительного повышения эффективности работы организации, минимизации рисков и достижения высоких показателей управленческой дисциплины.

Для проектирования такой системы необходимо выделить и формализовать основные информационные объекты, которые будут отражать структуру бизнес-процессов и послужат основой для построения моделей базы данных. Эти объекты определяют логику работы системы, позволяют организовать хранение, обработку и анализ информации, связанной с исполнением поручений.

К ключевым объектам системы относятся:

**- Пользователь (логин)** — содержит данные об учетных записях всех участников системы: логин, пароль, роль пользователя (например, администратор, исполнитель, заказчик), а также поля для одобрения регистрации и комментариев. Это основной объект для аутентификации и разграничения прав доступа.

**- Исполнитель** — описывает сотрудников, назначаемых для выполнения поручений.

**- Заказчик** —лица, формирующие поручения. Аналогично, содержатся ФИО, подразделение, должность и привязка к логину, что позволяет отследить, кто именно создал то или иное поручение.

**- Администратор** —содержит информацию о сотрудниках с правами управления системой: внесение изменений, контроль пользователей, распределение ролей.

**- Поручение** — основной управляемый объект системы, представленный. Содержит идентификатор, наименование поручения, текстовое описание, дату создания, крайний срок.

- **Статус исполнения** — таблица используется для описания текущего состояния поручения или его этапа. Содержит наименование статуса (например, «выполняется», «на проверке», «выполнено»), дату изменения и возможный комментарий. Это позволяет отслеживать прогресс по каждому заданию.

**- Исполнение поручения конкретным исполнителем** — служит для учета связи между поручением и его непосредственными исполнителями. Хранит идентификаторы поручения, исполнителя и текущего статуса, что позволяет реализовать схему «многие ко многим» между задачами и сотрудниками.

**- Регистрация заявок на выполнение** — используется для фиксации заявок новых пользователей или исполнителей на участие в системе. Поля включают ФИО, логин, должность, подразделение и ссылки на связанные записи. Эта таблица может быть использована для согласования доступа и начального формирования учетной записи.

**- Должности** — является справочной и содержит перечень всех должностей, используемых в организации. Связывается с исполнителями, администраторами и заказчиками.

- **Подразделения** — представляет организационную структуру, включающую отделы, управления, департаменты. Используется для группировки сотрудников и построения иерархии.

Каждый из этих объектов имеет четкую роль в бизнес-процессе и связан с другими сущностями через внешние ключи. Например, один заказчик может создавать множество поручений; поручение может быть направлено нескольким исполнителям; каждый исполнитель относится к определённому подразделению и должности; статус исполнения может меняться во времени и фиксироваться в отдельных таблицах.

Подобная структура обеспечивает хорошую масштабируемость системы и возможность построения гибкой модели данных, на которой в дальнейшем можно реализовать функционал автоматизированного контроля исполнения поручений: фильтрацию и поиск задач, напоминания о сроках, отчеты по эффективности, анализ загруженности и прочее.

## 1.2 Описание бизнес-процессов

Проектирование информационной системы контроля исполнения поручений требует тщательного анализа текущих бизнес-процессов организации и формализации их в виде моделей, отражающих функциональные, информационные и объектные взаимодействия участников. Для этого используются как функциональные методы моделирования, такие как IDEF0 и DFD, так и объектно-ориентированные нотации UML. На основе этих моделей проектируются интерфейсы, логика работы и структура базы данных будущей системы.

Базовым бизнес-процессом в предметной области является жизненный цикл поручения — от его создания до завершения. Этот процесс включает следующие этапы:

**- Формирование поручения** — заказчик (руководитель или администратор) создаёт новое поручение, указывая его содержание, срок исполнения и назначая ответственных исполнителей.

**- Регистрация и распределение** — поручение сохраняется в системе и становится доступным для просмотра назначенным исполнителям.

**- Исполнение поручения** — исполнитель знакомится с поручением, приступает к выполнению и при необходимости оставляет промежуточные комментарии или отчёты.

**- Контроль статуса** — система фиксирует изменения статуса поручения, уведомляет о приближении сроков или просрочке.

**- Приёмка выполнения** — заказчик проверяет результат, оставляет комментарии, при необходимости возвращает на доработку или утверждает выполнение.

**- Аналитика и отчётность** — руководители могут просматривать статус всех задач, отслеживать загруженность сотрудников, формировать отчёты о выполнении поручений по подразделениям и пользователям.

Этот бизнес-процесс включает в себя как ручные действия пользователей, так и автоматические функции системы, такие как уведомления, проверка сроков, фильтрация и формирование отчетов.

На рисунке 1 представлена контекстная диаграмма IDEF0 для разработки программного обеспечения по осуществлению контроля исполнения поручений.

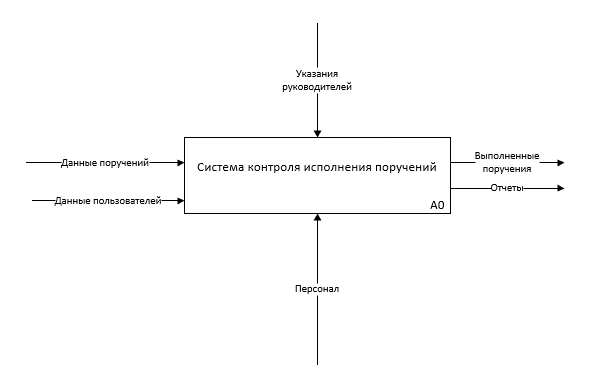


Рисунок 1 – Контекстная диаграмма описания системы контроля исполнения поручений

Контекстная диаграмма демонстрирует общее описание системы “Контроль исполнения поручений”. На вход подаются данные пользователей и поручений для этих пользователей, регламентирующие стандартами компании. На выходе получаем набор выполненных поручений, а также отчеты о выполненных задачах.

Теперь рассмотрим декомпозицию процесса системы контроля исполнения поручений, представленную на рисунке 2.

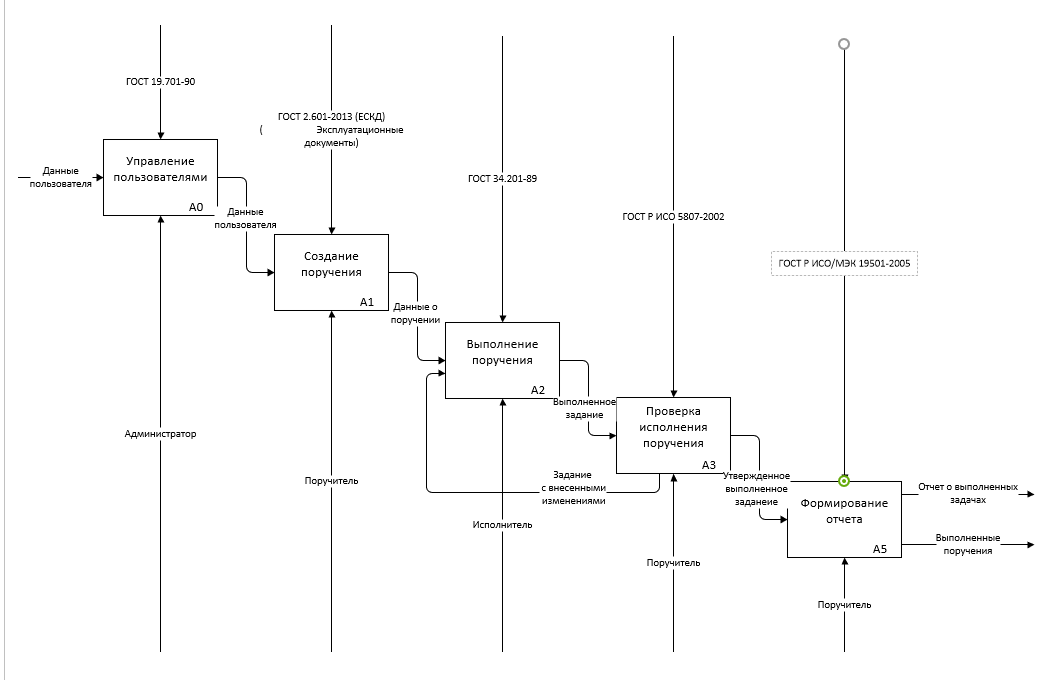


Рисунок 2 – Диаграмма декомпозиции описания системы контроля выполнения поручений

Декомпозиция показывает, как общее описание системы расщепляется на конкретные процессы.

Процесс управления арендными помещениями начинается с этапа **управления пользователями (A0)**. На этом этапе формируется профиль пользователя, включающий ключевые параметры, такие как тип, контактная информация и предпочтения по аренде.

На следующем этапе осуществляется **создание поручения (A1)**. Здесь формируются задачи, связанные с подготовкой и оформлением договоров аренды, с учётом требований пользователей. После создания поручения система фиксирует необходимые данные и передаёт их на выполнение.

После этого система переходит к этапу **выполнения поручения (A2)**. Исполнитель обрабатывает задачи, связанные с регистрацией помещений, включая их каталогизацию, описание характеристик (площадь, адрес, состояние) и подготовку к сдаче в аренду.

На следующем этапе происходит **проверка исполнения поручения (A3)**. Поручитель оценивает качество выполнения задач, включая предоставление услуг (например, техническое обслуживание, ремонт), с учётом стандартов и требований.

Завершающим этапом является **формирование отчёта (A5)**. Здесь система обрабатывает данные об утверждённых задачах и формирует отчёт по выполненным поручениям, включая информацию о финансовых операциях: платежах, комиссиях и сопутствующих расходах.

Data Flow Diagram (DFD) показывает потоки данных между этими элементами, что позволяет четко понять, как информация передается и обрабатывается на каждом этапе. DFD диаграмма представлена на рисунке 3

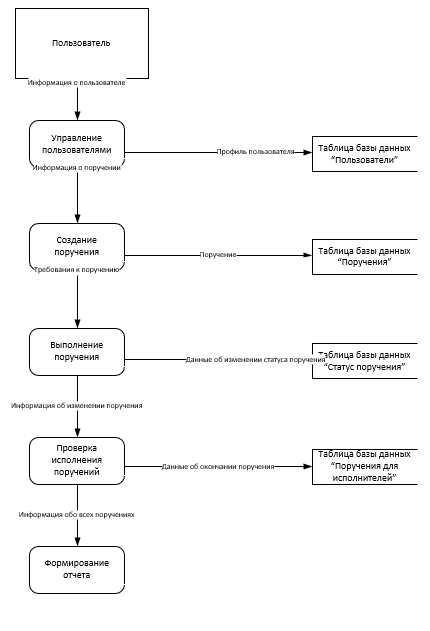


Рисунок 3 – диаграмма потоков данных

Процесс начинается с участия пользователя, который предоставляет информацию о себе. Эти данные поступают в процесс **«Управление пользователями»**, где происходит создание или обновление профиля пользователя. Профиль сохраняется в **таблице базы данных «Пользователи».**

Затем формируется задание или поручение. Пользовательские требования передаются в процесс **«Создание поручения»,** где поручение обрабатывается и сохраняется в **таблице базы данных «Поручения»**.

Далее поручение направляется в процесс **«Выполнение поручения»,** где исполнитель приступает к выполнению задания. Все изменения статуса выполнения фиксируются и сохраняются в **таблице базы данных «Статус поручения».**

После выполнения задания информация об этом передается в процесс **«Проверка исполнения поручений»**, где поручение проверяется и утверждается. Информация о завершении поручения сохраняется в **таблице базы данных «Поручения для исполнителей».**

На последнем этапе выполняется **«Формирование отчета»**, где на основе данных о всех выполненных поручениях создается итоговый отчет. Этот отчет служит подтверждением завершения задач и может быть использован для анализа эффективности исполнения поручений.

1.3 Постановка цели и задач системы

**Целью разрабатываемой информационной системы является автоматизация процессов контроля исполнения поручений**, что позволит упростить и ускорить взаимодействие между участниками процесса, повысить прозрачность выполнения задач, обеспечить эффективное отслеживание статусов и сроков исполнения, а также предоставить удобные средства фильтрации и экспорта данных для анализа и отчетности.

Для достижения поставленной цели система должна решать следующие задачи:

**- Автоматизация учета поручений**, включая возможность их создания, редактирования, удаления и поиска, а также хранение информации о поручителях, исполнителях, сроках выполнения, приоритетах и связанных документах;

**- Обеспечение управления жизненным циклом поручений**, с возможностью назначения ответственных, постановки сроков, изменения статуса выполнения и фиксации комментариев, что позволяет отслеживать прогресс по каждой задаче в режиме реального времени;

**- Реализация механизма проверки исполнения поручений**, включая подтверждение выполнения, возврат на доработку, внесение замечаний и формирование обратной связи от поручителя, а также последующее закрытие поручения;

**- Предоставление инструментов для управления пользователями и ролями**, включая регистрацию и авторизацию пользователей, распределение прав доступа в зависимости от роли (поручитель, исполнитель, администратор) и контроль истории действий в системе;

**- Реализация возможностей фильтрации информации** по различным параметрам (дата, статус, исполнитель, приоритет и т. д.), а также экспорт отфильтрованных данных в форматы Word, Excel для подготовки отчетов и аналитики;

**- Обеспечение безопасности и целостности данных** за счет внедрения механизмов аутентификации, авторизации, валидации вводимой информации и разграничения доступа к функциональности в соответствии с ролью пользователя;

**- Создание современного и удобного пользовательского интерфейса**, выполненного в едином лаконичном стиле, обеспечивающего быструю навигацию, понятные формы ввода и визуализацию текущего статуса выполнения поручений для всех участников процесса. 2 Проектирование базы данных для системы “Контроль исполнения поручений”

## 2.1 Проектирование концептуальной модели

Проектирование базы данных (БД) – одна из наиболее сложных и ответственных задач, связанных с созданием информационной системы (ИС). В результате её решения должны быть определены содержание БД, эффективный для всех её будущих пользователей способ организации данных и инструментальные средства управления данными.

В рамках сформулированной цели и решения конкретных задач можно выделить следующие типы сущностей:

**Пользователи.** В системе хранятся сведения о сотрудниках, работающих с системой. Для каждого пользователя фиксируются уникальный идентификатор, логин, фамилия, имя, отчество, номер телефона, электронная почта и дата регистрации. Также каждому пользователю соответствует определённое подразделение и должность.

**Подразделения и должности.** Сотрудники могут быть распределены по различным структурным единицам. Для каждого подразделения и должности сохраняется уникальный идентификатор и наименование.

**Поручения.** Система предусматривает механизм создания поручений. Для каждого поручения указывается наименование, описание, дата выдачи и выполнения, а также статус выполнения. Каждое поручение связано с автором и получателем (исполнителем).

**Исполнители.** Для отслеживания выполнения поручений используется отдельная сущность, фиксирующая связи между поручением, исполнителем и текущим статусом выполнения задачи.

**Статусы выполнения.** Статусы отражают текущий этап выполнения поручений. У каждого статуса есть уникальный идентификатор, наименование и дата последнего изменения.

Концептуальная модель представлена на рисунке 4.

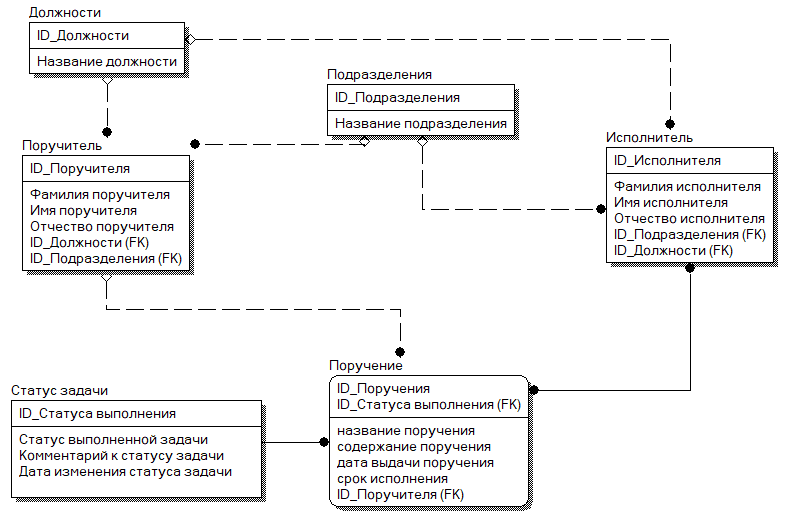


Рисунок 4 – концептуальная модель базы данных

Приведённая концептуальная модель базы данных будет использована как основа для создания логической модели, Концептуальная модель предоставляет общее представление о структуре системы и взаимосвязях между сущностями, что является важным этапом в процессе разработки информационной системы.

## 2.2 Проектирование логической модели базы данных

На основе концептуальной модели была разработана логическая модель базы данных, которая описывает основные сущности системы и их связи более детально. Структура модели представлена на рисунке 5

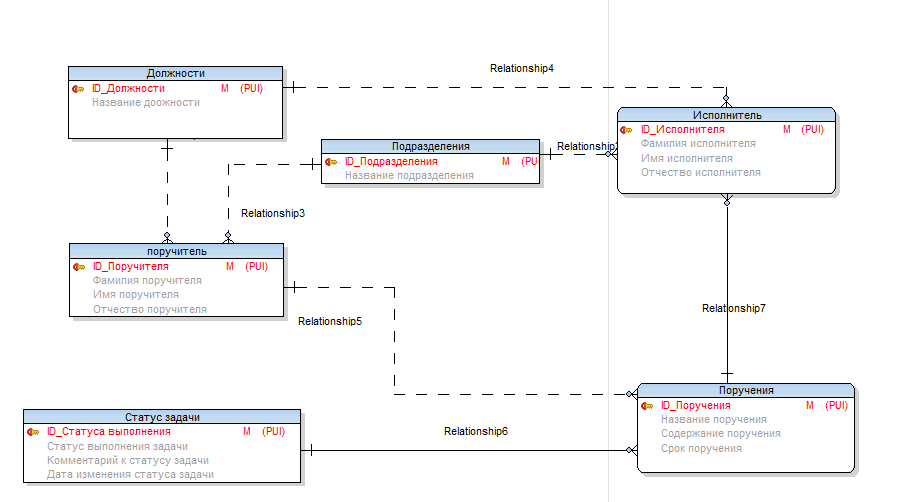


Рисунок 5 – логическая модель базы данных

В системе реализованы следующие связи между основными сущностями:

**Поручатель** связан с **должностью** и **подразделением** отношением многие-к-одному, поскольку несколько поручателей могут занимать одну и ту же должность и входить в одно подразделение.

**Исполнитель** также связан с **подразделением** и **должностью** по аналогичной логике — несколько исполнителей могут принадлежать к одному подразделению и занимать одну должность.

**Поручения** находятся в отношении **многие-к-одному** с исполнителями, так как один исполнитель может выполнять несколько поручений.

Каждое **поручение** связано со своим **статусом задачи** в отношении многие-к-одному, поскольку множество поручений могут иметь один и тот же статус (например, "выполняется" или "выполнено").

Таким образом, модель отражает структуру управления задачами в организации, где поручатели и исполнители являются сотрудниками различных подразделений и должностей, поручения назначаются исполнителям, а прогресс отслеживается по статусам выполнения.

Нормализация базы данных необходима для минимизации избыточности данных, устранения аномалий при вставке, изменении и удалении данных, а также обеспечения целостности и согласованности информации. Это достигается путем разделения данных на связанные таблицы, где каждый фрагмент информации хранится только один раз. В результате нормализации данные становятся более организованными, легче поддаются изменениям и менее подвержены ошибкам, что значительно упрощает управление базой данных и повышает её эффективность. Например, без нормализации одни и те же данные могли бы храниться в разных местах, что привело бы к дублированию и возможным несоответствиям при обновлении. Нормализация помогает избежать таких проблем, делая систему более надежной и удобной для работы.

Убедимся, что построенная логическая модель базы данных удовлетворяет условиям нормализации:

- Все сущности логической модели находятся в первой нормальной форме (1NF), так как каждое поле является атомарным, содержит только одно значение, и отсутствуют повторяющиеся группы или списки. Каждая таблица имеет уникальный первичный ключ, обеспечивающий однозначную идентификацию записей.

- Сущности также соответствуют второй нормальной форме (2NF), поскольку все неключевые атрибуты полностью функционально зависят от целого первичного ключа, а частичные зависимости отсутствуют. В модели нет составных первичных ключей, где бы неключевые атрибуты зависели только от части ключа.

- Модель находится в третьей нормальной форме (3NF), так как все неключевые атрибуты напрямую зависят от первичного ключа, исключая транзитивные зависимости. Таким образом, структура базы данных обеспечивает минимизацию избыточности и поддержание целостности данных.

## 2.3 Проектирование физической модели базы данных

Физическая модель создана для реализации логической структуры данных в конкретной системе управления базами данных. Она определяет точные типы данных, ограничения целостности, индексы и связи между таблицами, что обеспечивает эффективное хранение и управление данными. Данная модель является конечным этапом проектирования базы данных и представлена на рисунке 6

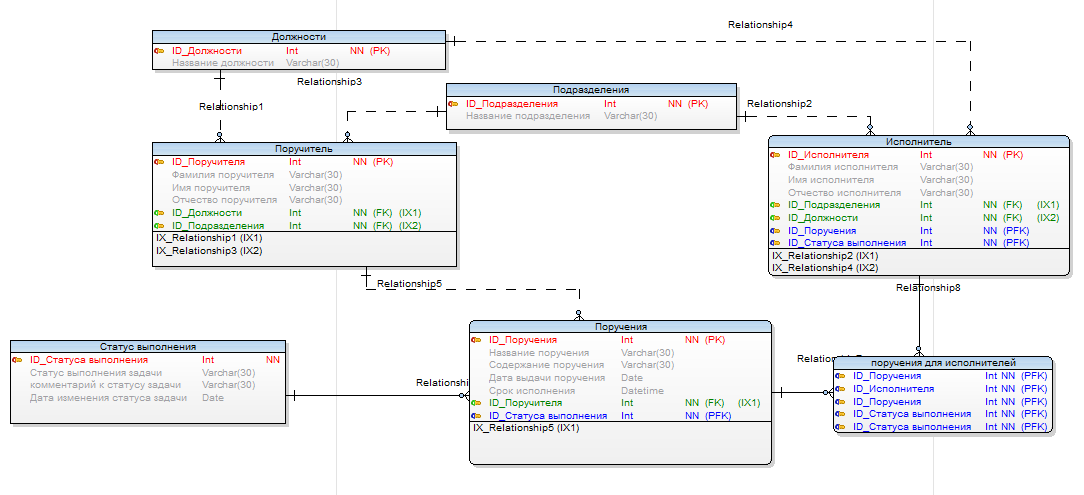


Рисунок 6 – физическая модель базы данных

**Таблица** Поручитель предназначена для хранения информации о сотрудниках, которые создают поручения. В таблице фиксируются:

- ID\_Поручителя (тип Int) — первичный ключ,

- Фамилия, Имя, Отчество (тип Varchar(30)) — персональные данные,

- ID\_Должности (внешний ключ) — ссылается на таблицу Должности,

- ID\_Подразделения (внешний ключ) — ссылается на таблицу Подразделения.

**Таблица** Должности содержит список должностей в организации:

- ID\_Должности (тип Int) — первичный ключ,

- Название должности (тип Varchar(30)).

**Таблица** Подразделения предназначена для хранения информации о структурных подразделениях:

- ID\_Подразделения (тип Int) — первичный ключ,

- Название подразделения (тип Varchar(30)).

**Таблица** Исполнитель содержит сведения об исполнителях поручений:

- ID\_Исполнителя (тип Int) — первичный ключ,

- Фамилия, Имя, Отчество (тип Varchar(30)),

- ID\_Подразделения и ID\_Должности (внешние ключи) — связи с таблицами Подразделения и Должности,

- ID\_Поручения (внешний ключ) — ссылается на таблицу Поручения,

- ID\_Статуса выполнения (внешний ключ) — ссылка на таблицу Статус выполнения.

**Таблица** Поручения используется для хранения информации о созданных поручениях:

- ID\_Поручения (тип Int) — первичный ключ,

- Название поручения, Содержание поручения (тип Varchar(30)),

- Дата выдачи, Дата исполнения (тип Date),

- ID\_Поручителя (внешний ключ) — связь с таблицей Поручитель,

- ID\_Статуса выполнения (внешний ключ) — ссылка на таблицу Статус выполнения.

**Таблица** Статус выполнения отражает текущий этап исполнения поручения:

- ID\_Статуса выполнения (тип Int) — первичный ключ,

- Статус выполнения задачи (тип Varchar(30)),

- Комментарий к статусу,

- Дата изменения статуса (тип Date).

**Таблица** Поручения для исполнителей реализует связь многие-ко-многим между поручениями и исполнителями:

- ID\_Поручения, ID\_Исполнителя, ID\_Статуса выполнения — составной первичный ключ и внешние ключи, связывающие исполнителей с поручениями и текущим статусом исполнения.

# 3 Разработка и реализация клиентского приложения “Контроль исполнения поручений”

## 3.1 Выбор средств реализации

В современной практике разработки программного обеспечения грамотный выбор программно-инструментальных средств имеет решающее значение для успешной реализации проекта. От правильного подбора инструментов зависит не только эффективность выполнения задач, но и качество создаваемого программного продукта, его удобство в обслуживании, а также возможность масштабирования в будущем. Использование соответствующих средств позволяет существенно сократить сроки разработки, снизить вероятность ошибок и обеспечить стабильную и производительную работу системы.

В рамках настоящего проекта для достижения поставленных целей был применён комплекс взаимодополняющих инструментов, каждый из которых был выбран с учётом специфики предметной области и требований к создаваемой информационной системе. Ниже представлено описание используемых решений и их значимости в процессе проектирования и разработки.

**- Для управления базами данных** было выбрано программное обеспечение **Microsoft SQL Server Management Studio 20 (SSMS)** — мощная среда администрирования для Microsoft SQL Server. SSMS предоставляет разработчику широкий спектр возможностей: создание и редактирование объектов базы данных, работа с запросами на языке T-SQL, настройка прав доступа и мониторинг состояния системы. Надёжность, гибкость и масштабируемость делают данный инструмент оптимальным выбором для реализации базы данных в рамках рассматриваемой системы.

**- Разработка пользовательского интерфейса и логики приложения** осуществлялась с помощью **Microsoft Visual Studio 2022** — современной интегрированной среды разработки, предоставляющей полный набор инструментов для работы с языком C#. Среди преимуществ Visual Studio — интеллектуальный редактор кода, мощный отладчик, средства сборки и тестирования, а также поддержка актуальных фреймворков и библиотек, что позволяет создавать надёжные и функциональные приложения с удобным интерфейсом.

**- Моделирование бизнес-процессов** выполнялось с применением **BPWin** — специализированного инструмента, поддерживающего стандарты описания процессов, такие как IDEF0 и DFD. Этот инструмент помогает формализовать требования, выявить логические связи и потенциальные точки оптимизации в работе системы, а также обеспечить прозрачность и согласованность всех этапов разработки.

**- Для проектирования структуры базы данных** использовался **ERwin Data Modeler** — профессиональное средство моделирования, предназначенное для создания как логических, так и физических моделей данных. Инструмент позволяет определить структуры таблиц, их связи, атрибуты, провести нормализацию данных и сгенерировать SQL-код для их создания. Поддержка обратного проектирования делает ERwin особенно полезным при необходимости интеграции или модификации существующих БД.

Таким образом, применяемый в проекте набор программных решений обеспечивает комплексный подход ко всем основным этапам жизненного цикла разработки: от формализации требований и моделирования процессов до реализации базы данных и интерфейса приложения. Выбранные средства не только обеспечивают высокий уровень качества создаваемого продукта, но и гарантируют его надёжность, расширяемость и удобство дальнейшего сопровождения, что особенно важно при работе с системами длительного срока эксплуатации.

## 3.2 Архитектура программного обеспечения

Разрабатываемая система представляет собой приложение на платформе **Windows Forms**, реализованное с использованием языка **C#** и построенное по принципам многоуровневой архитектуры. Интерфейс пользователя представлен набором форм, каждая из которых отвечает за определённую функциональность: авторизация, управление договорами аренды, объектами недвижимости, оказанными услугами, выставленными счетами, а также отслеживание и обработку платежей.

Взаимодействие с базой данных осуществляется через специализированный компонент **DatabaseHelper**, который инкапсулирует всю работу с **SQL Server** и обеспечивает безопасный и эффективный доступ к данным. Это решение позволило абстрагировать логику работы с хранилищем информации от остальной части приложения, повысив тем самым его гибкость и сопровождаемость.

Система использует **ролевую модель управления доступом**, согласно которой каждому пользователю присваивается определённая роль. В зависимости от уровня доступа, пользователю предоставляется определённый набор функций, что позволяет ограничить действия в системе и обеспечивает соблюдение политики безопасности.

Бизнес-логика приложения разделена на несколько модулей, каждый из которых реализует отдельную область функциональности: модуль аутентификации и управления пользователями, модуль работы с помещениями и договорами, блок учета оказываемых услуг, а также модуль финансовых операций и анализа платежей. Такой подход способствует логической структуризации кода и облегчает дальнейшее развитие и поддержку системы.

Для улучшения взаимодействия с пользователем в системе реализованы **вспомогательные компоненты**, включая расширения стандартных элементов управления, единый стиль оформления (темизация интерфейса), а также базовая форма, на основе которой создаются все прочие формы. Это позволило унифицировать внешний вид и поведение интерфейса, сделав работу с системой более удобной и интуитивно понятной.

Особое внимание было уделено **вопросам безопасности**. В системе реализовано **хеширование паролей** с использованием стойких алгоритмов, ведётся **журналирование действий пользователей** (логирование), а также реализованы меры по **защите от SQL-инъекций**, включая параметризацию запросов и валидацию пользовательского ввода.

Архитектура системы построена на основе проверенных шаблонов проектирования, включая **MVC (Model-View-Controller)** для разделения представления, логики и данных, **Repository** — для инкапсуляции доступа к данным, а также **Unit of Work** — для обеспечения согласованности при выполнении нескольких связанных операций. Применение данных паттернов обеспечивает **модульность, масштабируемость** и **лёгкость сопровождения** программного кода.

Диаграмма классов является важным инструментом в процессе разработки программного обеспечения, позволяющим визуально представить структуру системы, взаимодействие между компонентами и их функциональность. Она играет ключевую роль на этапах проектирования и документирования, обеспечивая понимание архитектуры системы. Диаграмма классов представлена на рисунке 7

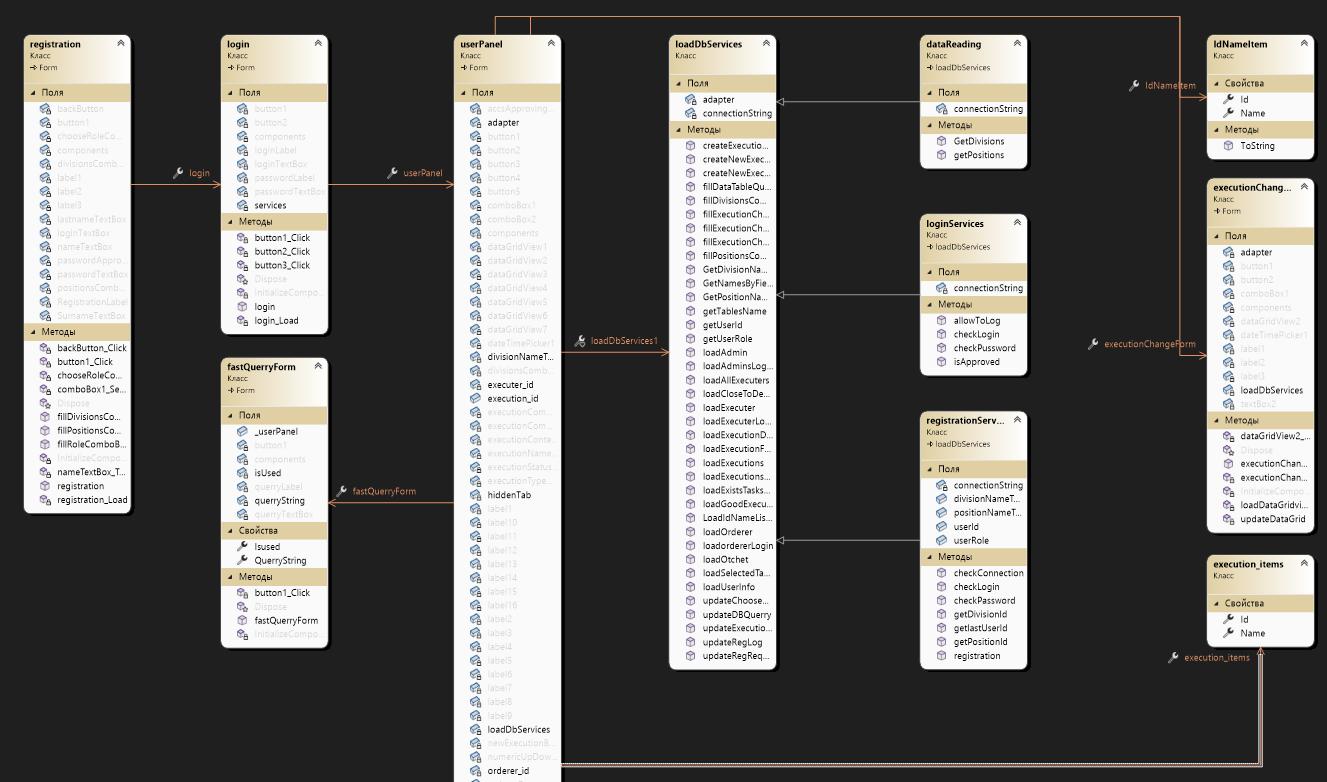


Рисунок 7 – диаграмма классов

Краткое описание модулей:

**Формы интерфейса (UI):**

- login – форма входа пользователя, взаимодействует с loginServices.

- registration – форма регистрации, использует registrationServices.

- userPanel – основная панель пользователя, подключает loadDbServices.

- executionChangeForm – форма редактирования поручений.

- fastQueryForm – форма для быстрого поиска, подключена к userPanel.

**Сервисные классы (логика и доступ к данным):**

- loadDbServices – основной класс для работы с базой данных, содержит методы загрузки данных, обновлений, выборок.

- loginServices – отвечает за авторизацию, проверку логина и пароля.

- registrationServices – реализует регистрацию новых пользователей и работу со справочниками.

- dataReading – обеспечивает загрузку подразделений и должностей.

- IdNameItem – вспомогательный класс для представления пар "ID – Название".

- execution\_items – представляет элементы исполнения поручений (например, статусы и действия).

**Связи между модулями:**

- UI-формы напрямую связаны с сервисами.

- Сервисы используют один или несколько вспомогательных классов.

- userPanel агрегирует наиболее важные зависимости (loadDbServices, данные о пользователе и выполнении поручений).

Таким образом, диаграмма демонстрирует модульную структуру приложения с чётким разделением логики, интерфейса и доступа к данным.

## 3.2 Функциональные возможности

При запуске программы пользователь видит меню авторизации, в котором он может либо ввести уже существующие данные о своем аккаунте, либо зарегистрироваться. Меню входа представлено на рисунке 8

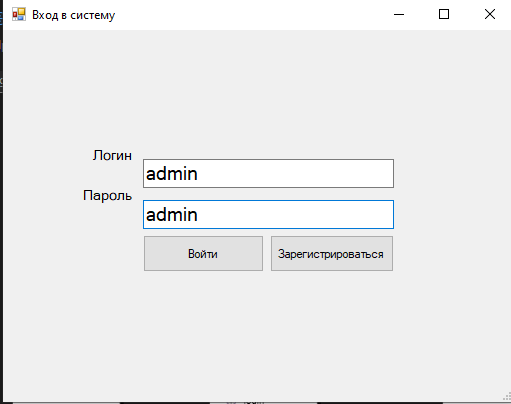


Рисунок 8 – меню входа

Для входа используется запрос “If exists (select 1 from loginEmployees where login = @login) select 1 else select 0”

Меню регистрации представлено на рисунке 9

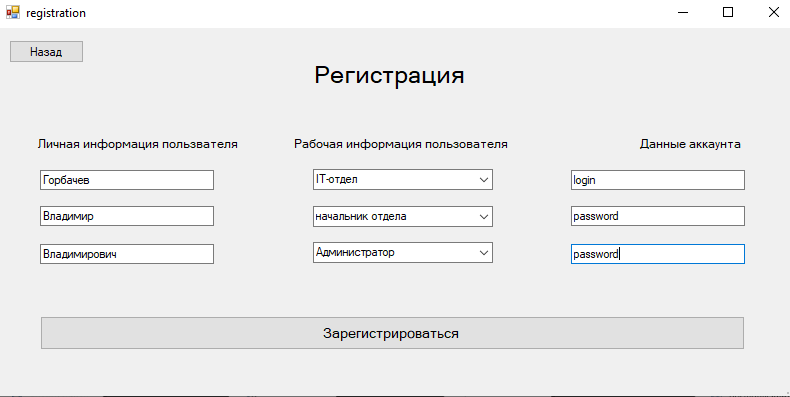


Рисунок 9 – меню регистрации

Запрос на создание нового пользователя: "insert into admins (admin\_surname, admin\_name, admin\_lastname, division\_id, position\_id, login\_id\_fk) values(@surname, @name, @lastname, @division\_id\_fk, @position\_id\_fk, @login\_id\_FK)"

Для полной регистрации пользователя администратор должен подтвердить данные только что зарегистрированного аккаунта.

При аунтификации пользователя происходит проверка прав доступа, и в зависимости от наделенных привилегий пользователю выводятся различные окна для взаимодействия.

Рассмотрим весь возможный функционал приложения.

На рисунке 10 изображено меню подтверждения аккаунтов.

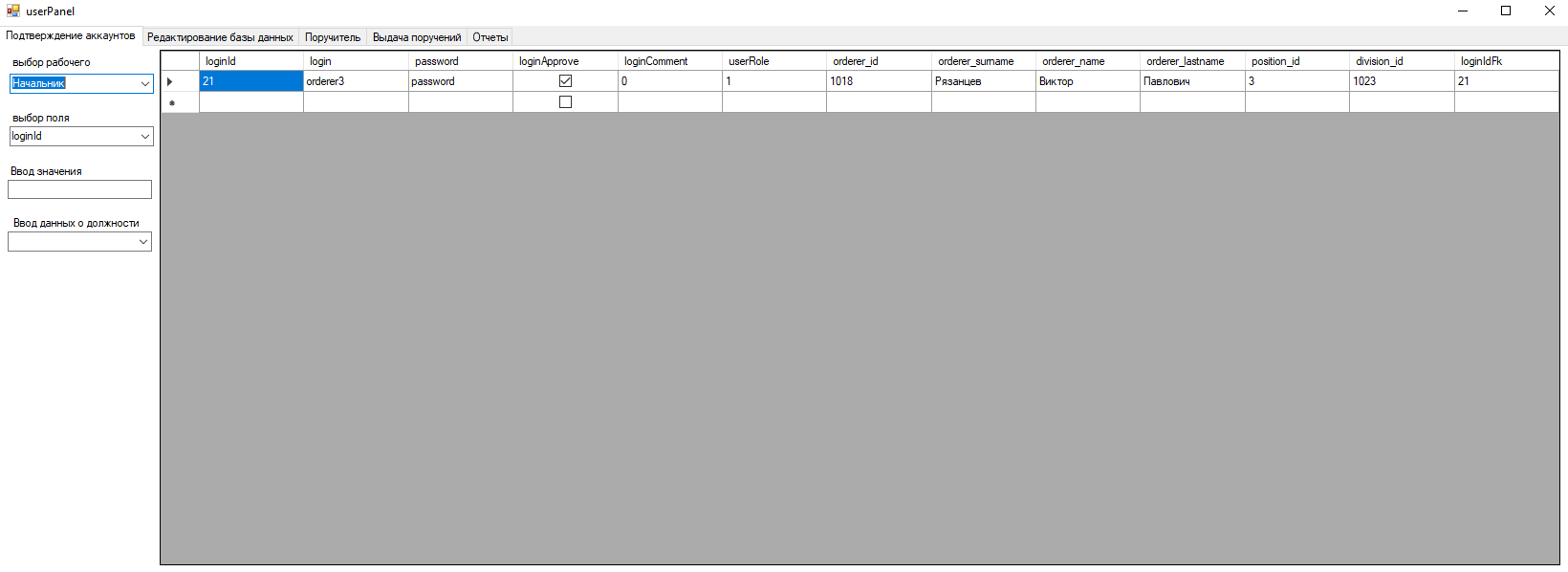


Рисунок 10 – окно подтверждения зарегистрированных пользователей

Для вывода используется функция loadAdmin() с запросом select \* from loginEmployees join registration\_requests on loginEmployees.loginId = registration\_requests.loginIdFk

После проверки администратором данных об аккаунте, администратор изменяет данные о регистрации пользователя и делает профиль активным.

На рисунке 11 изображен полный вывод базы данных для администратора

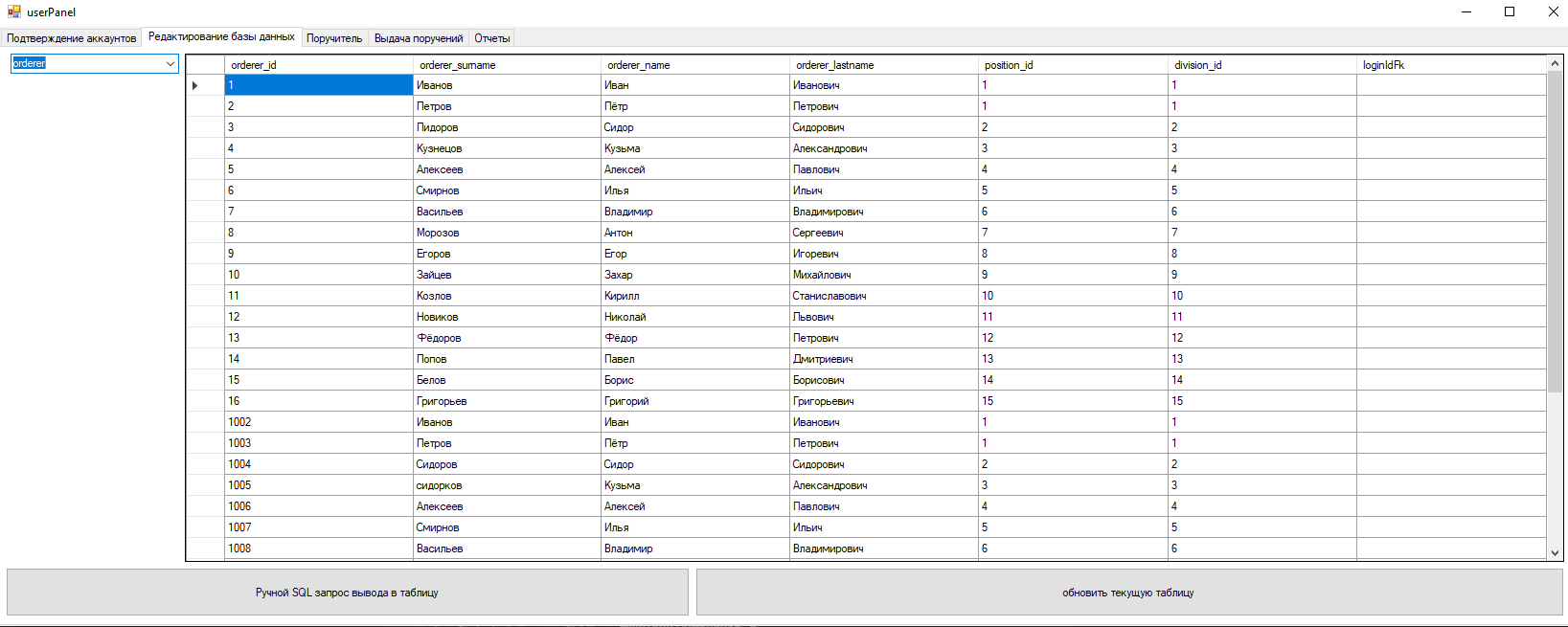


Рисунок 10 – полный вывод бд для администратора

Вывод всех бд происходит динамически.

В данной форме администратор имеет право редактировать БД, а также вводить SQL запросы на вывод данных в таблицу вручную.

Возможность ввода запросов изображена на рисунке 11.

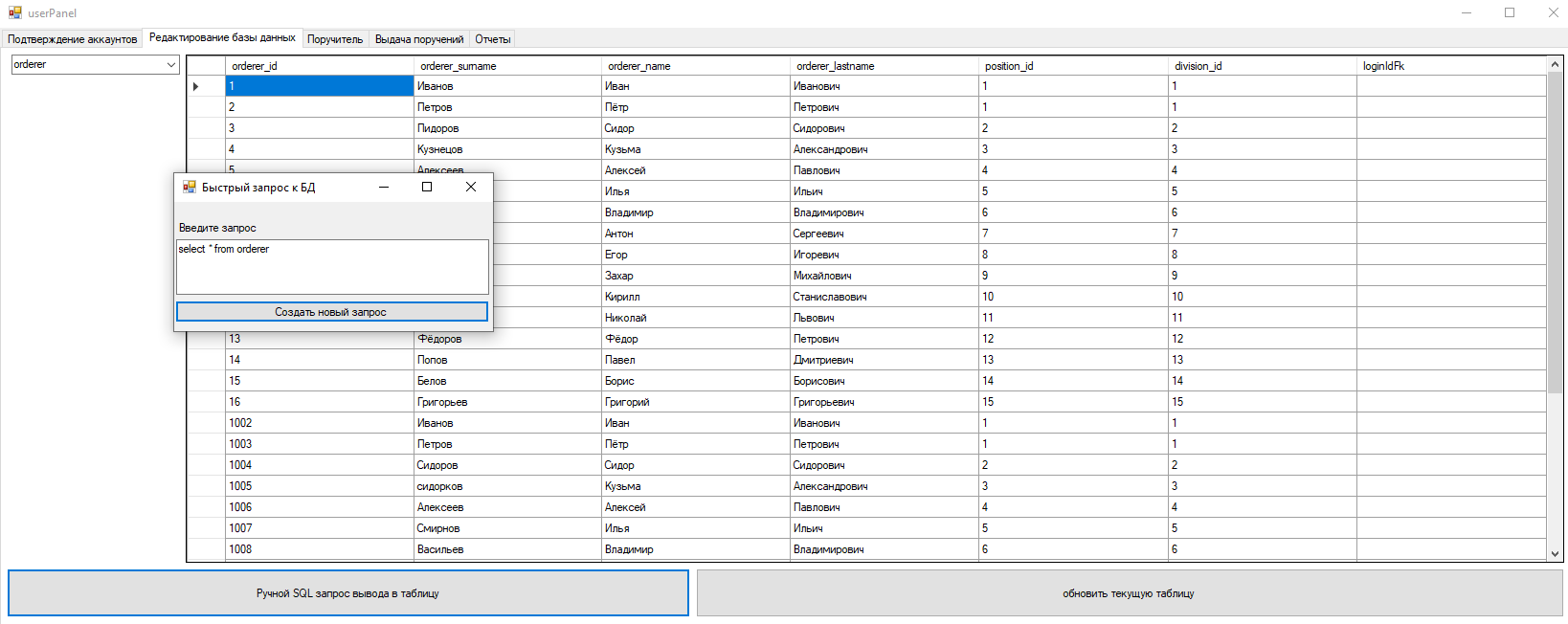


Рисунок 11 – возможность ручных SQL запросов

На рисунке 12 изображен вывод данных обо всех поручениях, доступных администратору и поручителю.

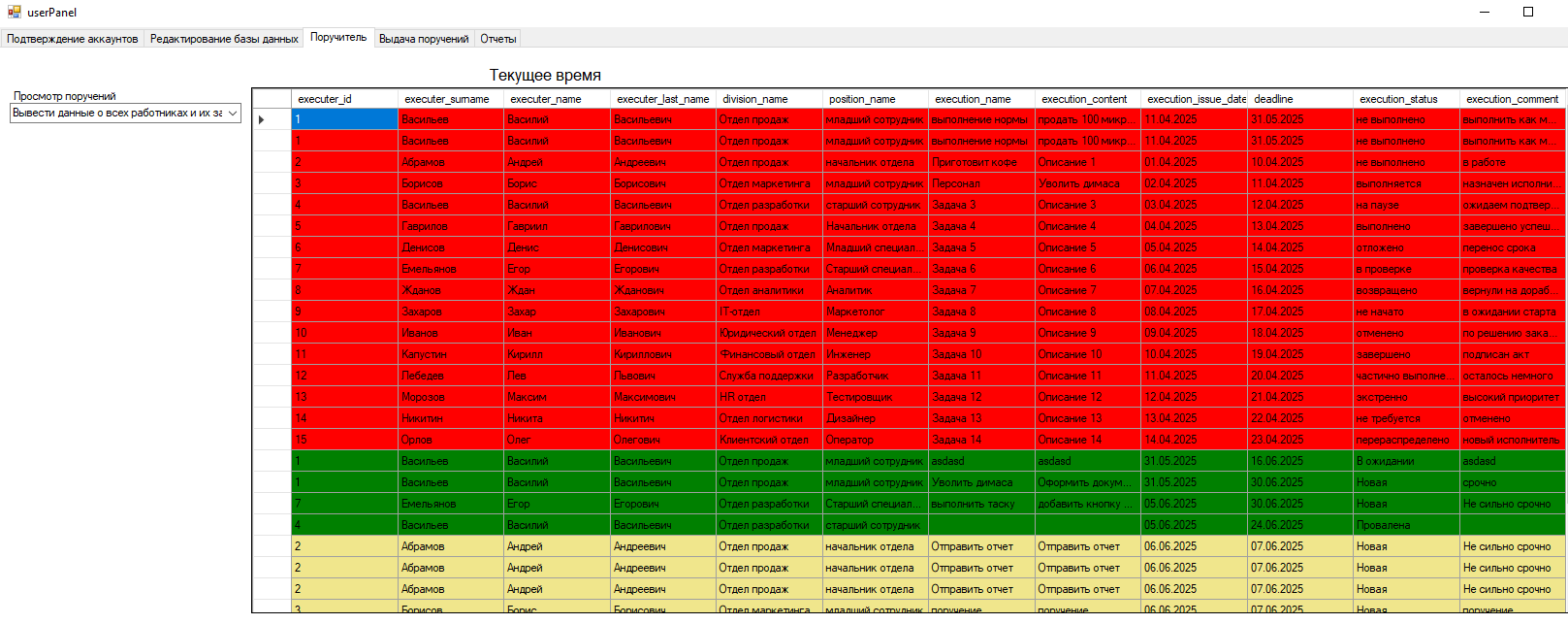


Рисунок 12 – данные о поручениях

На рисунке 13 изображен вывод просроченных поручений

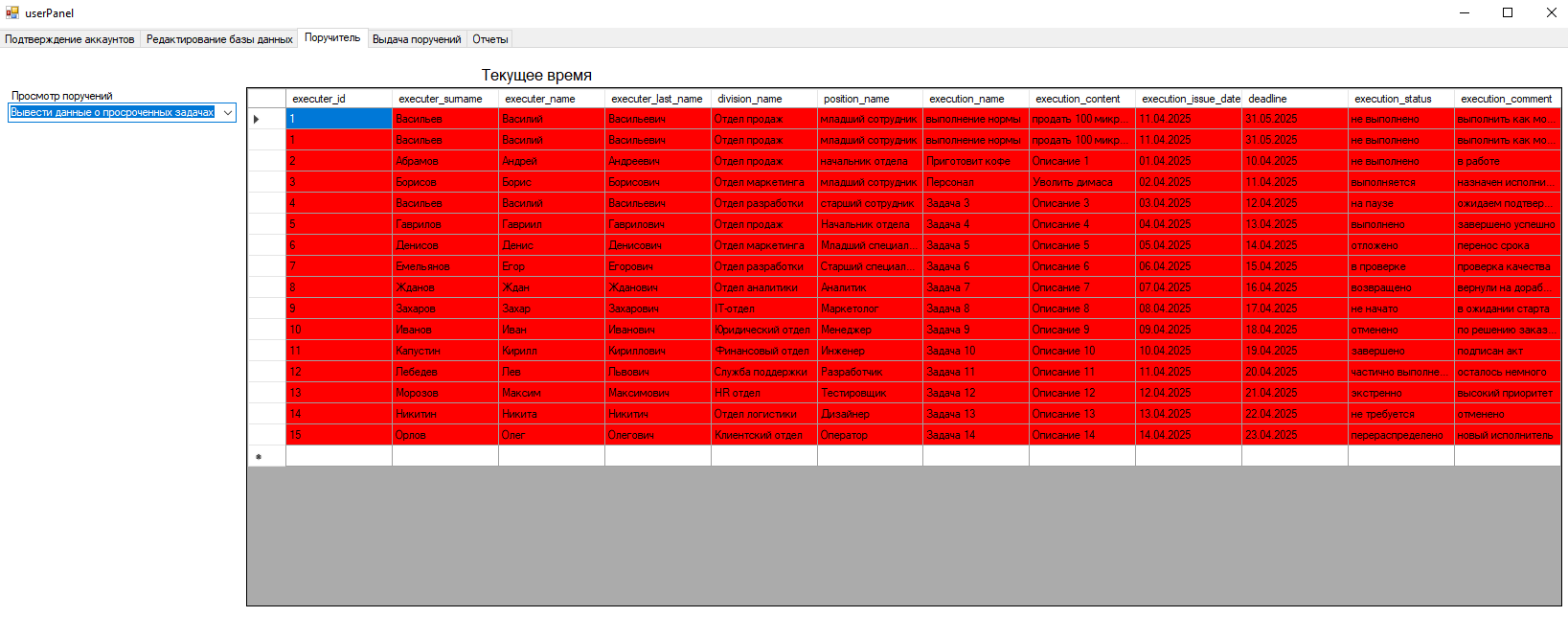


Рисунок 13 – вывод просроченных поручений

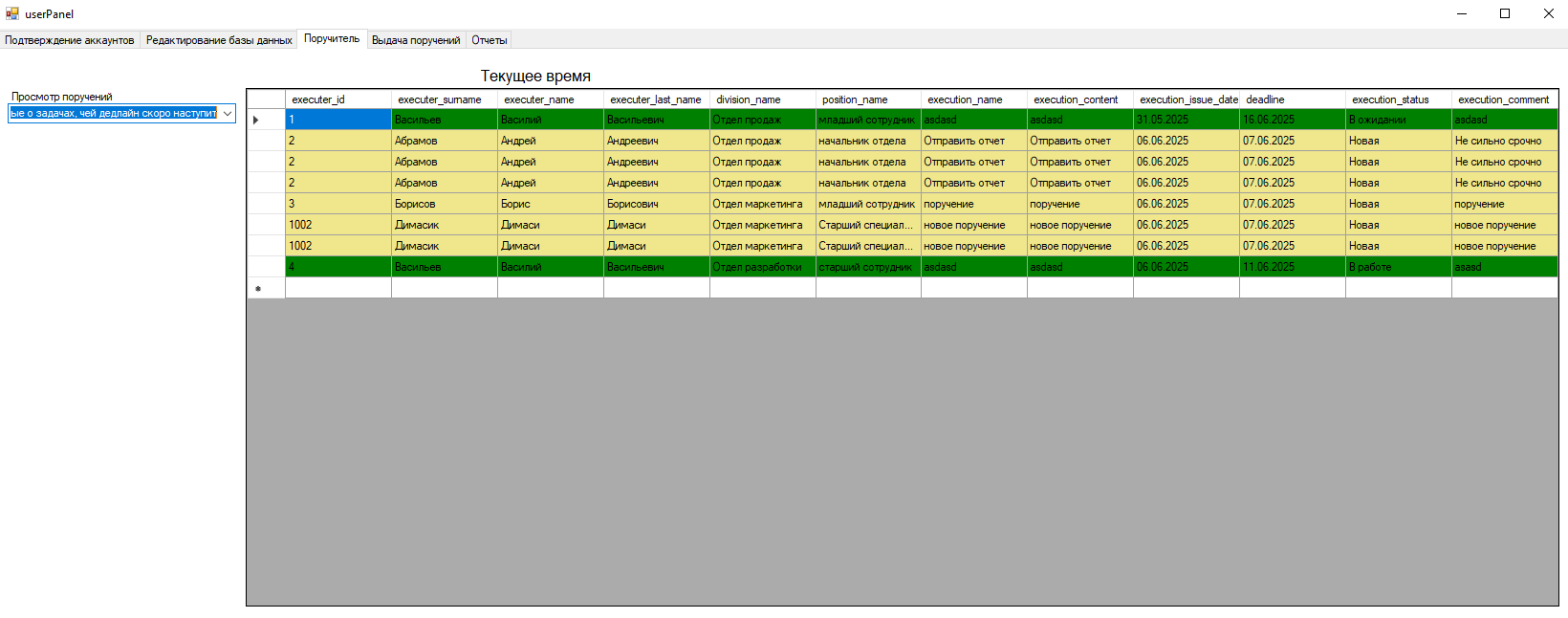


Рисунок 14 – вывод поручений, чей дедлайн скоро наступит

На рисунке 15 изображена панель выдачи поручений.

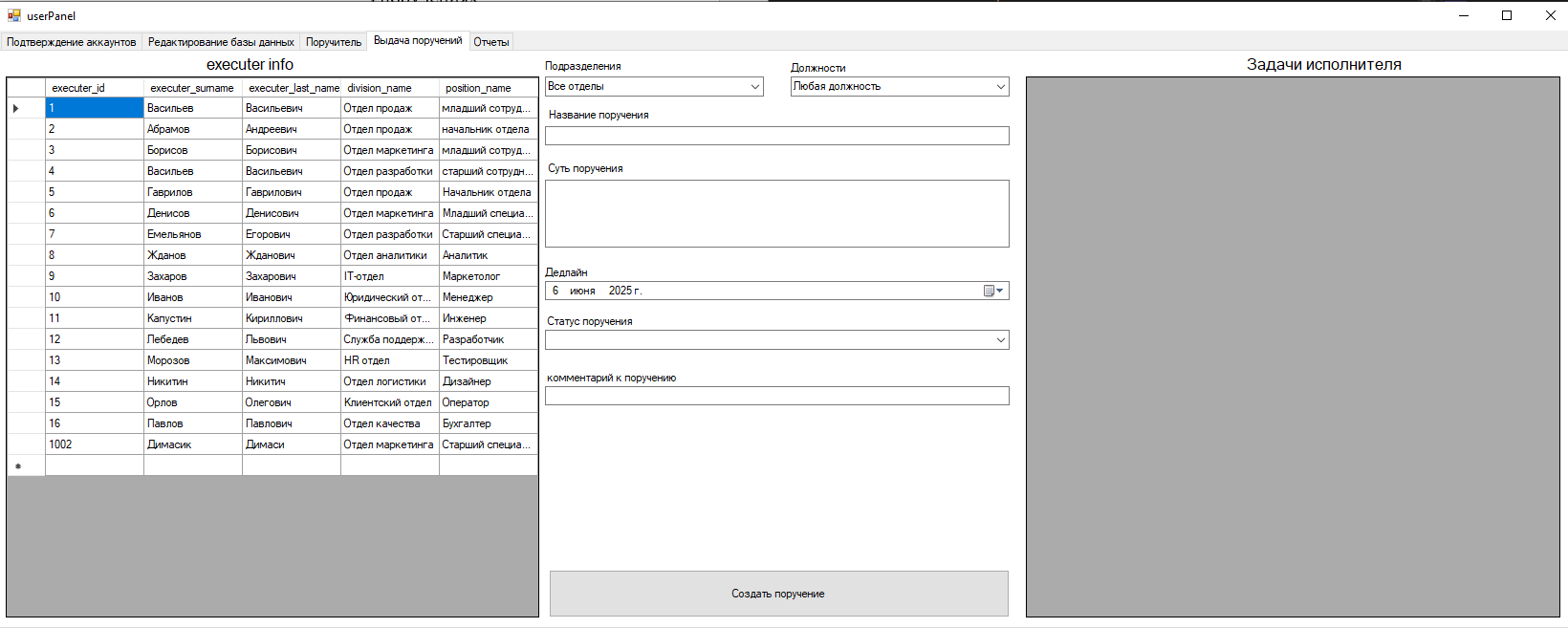


Рисунок 15 – панель ввода поручений

На рисунке 16 изображен принцип ввода данных в панель задач

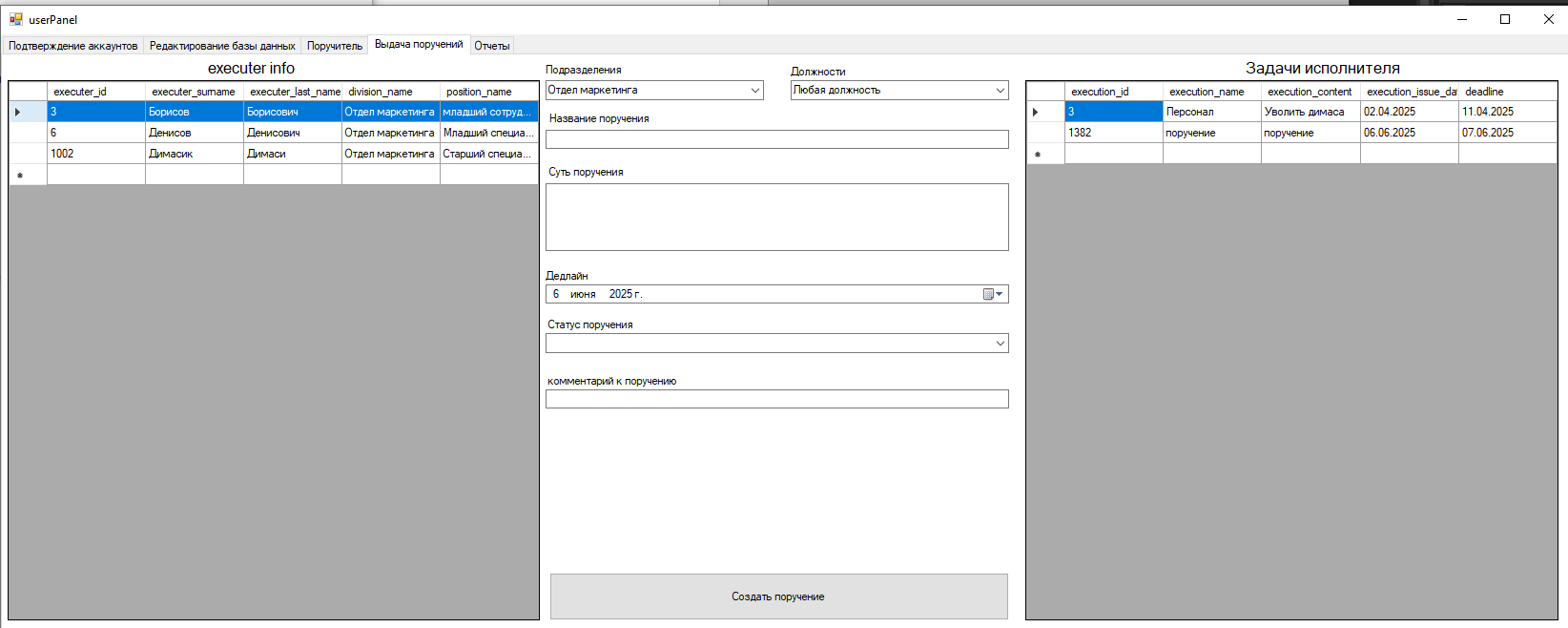


Рисунок 16 – принцип ввода данных о поручении

Поручитель вводит данные о поручении, после в левой таблице выбирает исполнителя и выдает ему поручение на кнопку.

На рисунке 17 изображена панель вывода отчетах о людях и их поручениях.

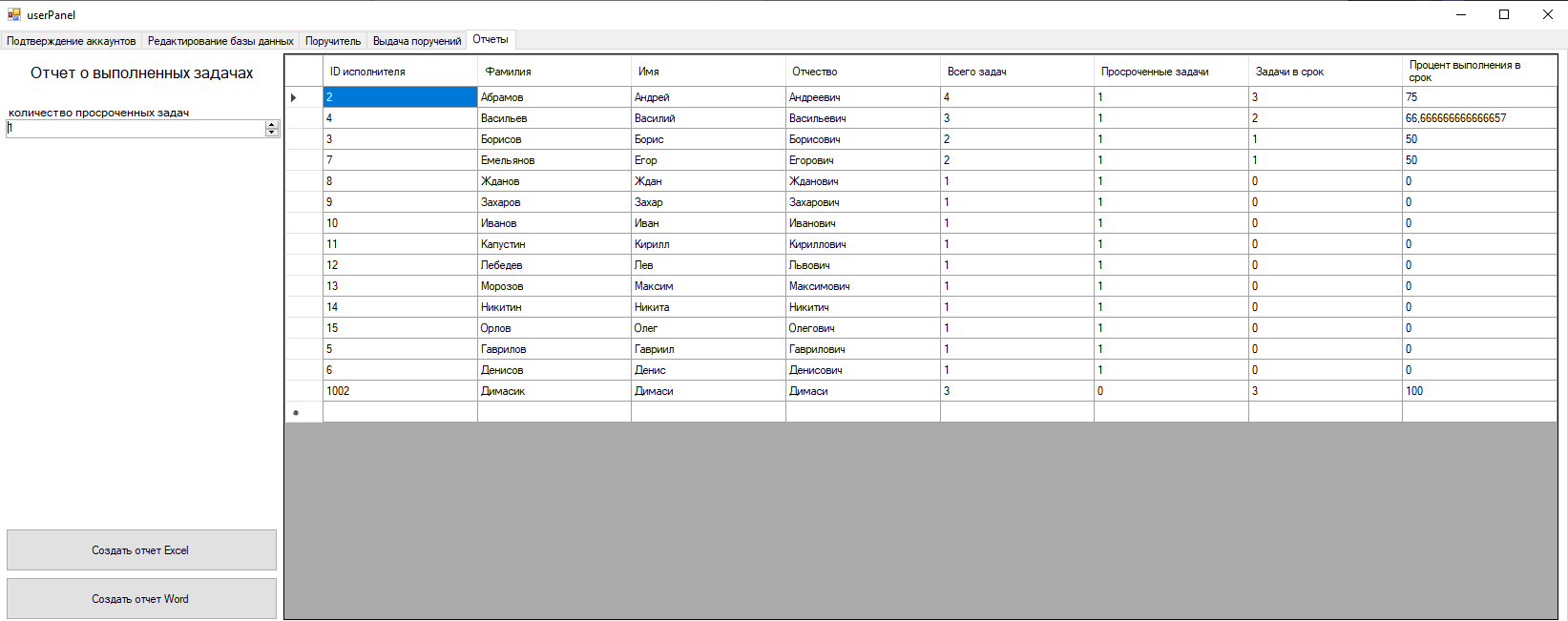


Рисунок 17 – вывод отчетах о людях и их поручениях

Также в меню вывода отчета можно делать выборку по количеству просроченных поручений

Отчет использует следующий SQL запрос: @"

-- Получаем ID статуса 'Выполнена'

DECLARE @CompletedStatusId INT;

SELECT @CompletedStatusId = execution\_status\_id

FROM [ordersExecutions].[dbo].[execution\_status]

WHERE execution\_status = 'Выполнена';

-- Если статус 'Выполнена' не найден, используем 0 (не будет совпадений)

IF @CompletedStatusId IS NULL SET @CompletedStatusId = 0;

SELECT

e.executer\_id AS [ID исполнителя],

e.executer\_surname AS [Фамилия],

e.executer\_name AS [Имя],

e.executer\_last\_name AS [Отчество],

COUNT(ex.execution\_id) AS [Всего задач],

SUM(CASE WHEN ex.deadline < GETDATE() AND

(ex.executiom\_status\_id\_fk != @CompletedStatusId OR ex.executiom\_status\_id\_fk IS NULL)

THEN 1 ELSE 0 END) AS [Просроченные задачи],

SUM(CASE WHEN ex.deadline >= GETDATE() OR

ex.executiom\_status\_id\_fk = @CompletedStatusId

THEN 1 ELSE 0 END) AS [Задачи в срок],

CAST(SUM(CASE WHEN ex.deadline >= GETDATE() OR

ex.executiom\_status\_id\_fk = @CompletedStatusId

THEN 1 ELSE 0 END) AS FLOAT) / NULLIF(COUNT(ex.execution\_id), 0) \* 100 AS [Процент выполнения в срок]

FROM

[ordersExecutions].[dbo].[executer] e

LEFT JOIN

[ordersExecutions].[dbo].[execution\_for\_executers] efe ON e.executer\_id = efe.executer\_id\_fk

LEFT JOIN

[ordersExecutions].[dbo].[execution] ex ON efe.execution\_id\_fk = ex.execution\_id

GROUP BY

e.executer\_id,

e.executer\_surname,

e.executer\_name,

e.executer\_last\_name

HAVING

COUNT(ex.execution\_id) > 0"; из функции loadOtchet();

На рисунке 18 изображен вид сформированного отчета в эксель



Рисунок 18 – сформированных отчет эксель

На рисунке 19 изображен WORD отчет

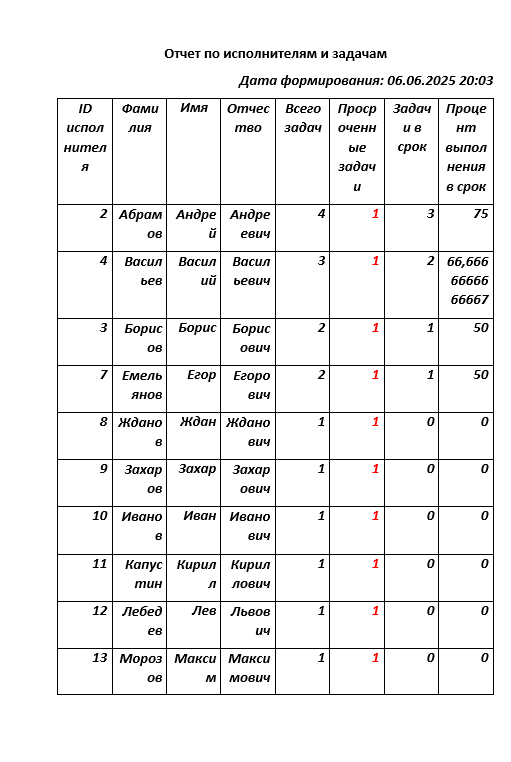


Рисунок 19 – вид отчета в WORD

Так же программа имеет меню для исполнителя, наделенного меньшим количеством прав, которое выводит исполнителю все его поручения

На рисунке 20 изображена панель пользователя с поручениями

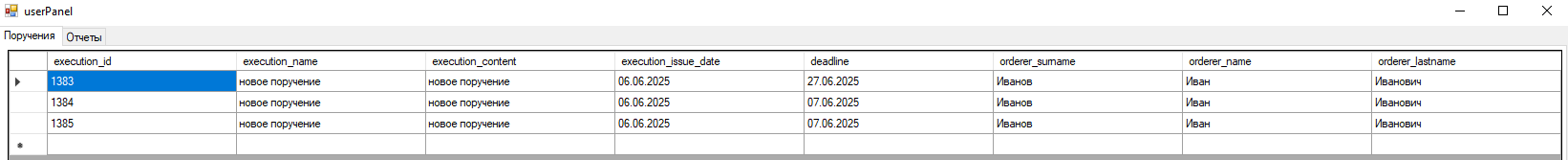


Рисунок 20 – панель пользователя

# Заключение

В ходе выполнения курсового проекта была спроектирована и реализована база данных для информационной системы контроля исполнения поручений. Система предназначена для автоматизации процессов управления поручениями в организациях, где требуется контроль исполнения задач, распределённых между подразделениями и сотрудниками.

В рамках проекта были достигнуты все поставленные цели и решены следующие задачи:

- Разработана логическая модель базы данных в соответствии с требованиями предметной области контроля поручений;

- Создана физическая модель с учётом нормализации данных до третьей нормальной формы (3NF);

- Реализованы основные таблицы базы данных, заданные связи между сущностями и ограничения целостности;

- Написаны SQL-запросы для операций вставки, обновления, выборки и удаления данных, используемых в приложении.

Для более полного представления структуры и работы системы были разработаны следующие диаграммы:

**-** Диаграмма классов, описывающая архитектуру клиентской части Windows Forms-приложения;

- Логическая диаграмма базы данных, отражающая связи между сущностями предметной области;

- Модульная диаграмма, демонстрирующая структуру приложения и взаимодействие между его основными компонентами.

Система предоставляет следующие возможности:

- Регистрация и аутентификация пользователей с учётом их ролей;

- Формирование и назначение поручений сотрудникам;

- Мониторинг сроков исполнения задач;

- Обновление статусов исполнения;

- Быстрый поиск и фильтрация поручений по различным критериям.

Применение данной системы позволяет повысить прозрачность и управляемость бизнес-процессов, улучшить контроль за исполнением поручений, а также минимизировать человеческий фактор и снизить риск просрочек. Автоматизация учёта поручений способствует более эффективному распределению ресурсов и повышению общей производительности организации.

# Список использованной литературы

1. Кенер, Джозеф. SQL Server 2019: Полное руководство / Джозеф Кенер; Пер. с англ. — М.: ДМК Пресс, 2022. — 1024 с.

2. Адамасек, Славомир. Изучаем SQL Server для разработчиков C# / Славомир Адамасек; Пер. с англ. — М.: Вильямс, 2021. — 512 с.

3. Литвин, Андрей. Entity Framework Core: Практическое руководство / Андрей Литвин. — М.: Питер, 2023. — 480 с.

4. Макконнелл, Стив. Базы данных. Современный подход / Стив Макконнелл; Пер. с англ. — СПб.: Питер, 2020. — 672 с.

5. Хабибуллин, Р. Р. Проектирование и администрирование баз данных SQL Server / Р. Р. Хабибуллин. — М.: Издательский дом "Вильямс", 2021. — 448 с.

6. Джойс, Бенджамин. ADO.NET для профессионалов / Бенджамин Джойс; Пер. с англ. — М.: ДМК Пресс, 2022. — 384 с.

7. Фленов, Ю. В. Разработка приложений с использованием SQL Server и C# / Ю. В. Фленов. — СПб.: БХВ-Петербург, 2020. — 528 с.

8. Робертсон, Эрик. Оптимизация производительности SQL Server / Эрик Робертсон; Пер. с англ. — М.: Вильямс, 2023. — 416 с.

9. Гарсиа-Молина, Хector. Базы данных: системный подход / Хector Гарсиа-Молина [и др.]; Пер. с англ. — М.: Вильямс, 2021. — 1280 с.

10. Петров, А. Н. Практикум по проектированию баз данных для Windows-приложений / А. Н. Петров. — М.: Горячая линия — Телеком, 2022. — 368 с.

# Приложение А

(Обязательное)

Листинг программного кода для окна входа

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Text;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace kpCRM

{

public partial class login : Form

{

private loginServices services;

public login()

{

services = new loginServices();

InitializeComponent();

}

public userPanel userPanel

{

get => default;

set

{

}

}

private void login\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

registration regForm = new registration();

this.Hide();

regForm.ShowDialog();

this.Show();

}

private void button2\_Click(object sender, EventArgs e)

{

int userId;

loadDbServices loadDbServices = new loadDbServices();

if (services.isApproved(loginTextBox.Text, passwordTextBox.Text) == true)

{

MessageBox.Show("Вход выполнен успешно");

userPanel userForm = new userPanel(loadDbServices.getUserRole(loginTextBox.Text), loadDbServices.getUserId(loginTextBox.Text, loadDbServices.getUserRole(loginTextBox.Text)));

userForm.ShowDialog();

this.Close();

}

else MessageBox.Show("Аккаунта с такими данными не существует");

}

private void button3\_Click(object sender, EventArgs e)

{

}

}

}

# Приложение Б

(Обязаельное)

Листинг программного кода для окна

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace kpCRM

{

public partial class registration : Form

{

public registration()

{

InitializeComponent();

fillDivisionsCombo();

fillPositionsCombo();

fillRoleComboBox();

}

public login login

{

get => default;

set

{

}

}

public void fillDivisionsCombo()

{

dataReading dataReading = new dataReading();

List<string> divisionList = dataReading.GetDivisions();

divisionsComboBox.Items.Clear();

divisionsComboBox.Items.AddRange(divisionList.ToArray());

}

public void fillPositionsCombo()

{

dataReading dataReading = new dataReading();

List<string> divisionList = dataReading.getPositions();

positionsComboBox.Items.Clear();

positionsComboBox.Items.AddRange(divisionList.ToArray());

}

public void fillRoleComboBox()

{

chooseRoleComboBox.Items.Add("Администратор");

chooseRoleComboBox.Items.Add("Начальник");

chooseRoleComboBox.Items.Add("Рабочий");

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void registration\_Load(object sender, EventArgs e)

{

}

private void backButton\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

registrationServices newReg = new registrationServices();

bool isConnectef = newReg.registration(loginTextBox.Text, passwordTextBox.Text, SurnameTextBox.Text, nameTextBox.Text, lastnameTextBox.Text, divisionsComboBox.SelectedItem.ToString(), positionsComboBox.SelectedItem.ToString(), chooseRoleComboBox.SelectedItem.ToString()); //chooseRoleComboBox.SelectedValue);

if (isConnectef)

{

MessageBox.Show("Можно регать");

}

else MessageBox.Show("Пользователь с таким именем уже существует");

}

private void nameTextBox\_TextChanged(object sender, EventArgs e)

{

}

private void chooseRoleComboBox\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

# Приложение В

(Обязательное)

Листинг программного кода окна ввода ручных SQL запросов

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace kpCRM

{

public partial class fastQuerryForm : Form

{

private string querryString;

private bool isUsed;

public userPanel \_userPanel;

public fastQuerryForm(userPanel userPanel)

{

InitializeComponent();

\_userPanel = userPanel;

Isused = false;

}

public string QuerryString { get { return querryString; } set { querryString = value; } }

public bool Isused { get { return isUsed; } set { isUsed = value; } }

private void button1\_Click(object sender, EventArgs e)

{

\_userPanel.fillTableByQuerry(querryTextBox.Text);

}

private void fastQuerryForm\_Load(object sender, EventArgs e)

{

}

}

}