**ОТЧЁТ**

*ОКЭИ 09.02.07 9024 11 ПП*

*ПП 02.01 Произовдственная практика*

По модулю *ПМ.02 Инструментальные средства разработки программного обеспечения*

Количество листов: *33*

Дата готовности: *24 сентября 2025*

Разработал: *Косырев А. В.*

Руководитель: *Лукьяненко Ольга Владимировна*

Соответствие отчета с заданием на практику *Лукьяненко О. В.*

Оренбург 2025

**Содержание**

[Введение 3](#_Toc209427937)

[1 Характеристика предприятия 4](#_Toc209427938)

[2 Техническое задание 6](#_Toc209427939)

[3 Информационное обеспечение 8](#_Toc209427940)

[4 Функционирование программы 11](#_Toc209427941)

[5 Логика работы программы 13](#_Toc209427942)

[6 Руководство системного программиста 14](#_Toc209427943)

[7 Руководство оператора 16](#_Toc209427944)

[8 Техника безопасности и пожарная безопасность 18](#_Toc209427945)

[Заключение 20](#_Toc209427946)

[Список используемых источников 22](#_Toc209427947)

[Приложение А 23](#_Toc209427948)

[Приложение Б 24](#_Toc209427951)

[Приложение В 25](#_Toc209427954)

[Приложение Г 27](#_Toc209427957)

[Приложение Д 29](#_Toc209427958)

[Приложение Е 30](#_Toc209427961)

[Приложение Ё 31](#_Toc209427964)

# Введение

Современные компании, работающие в сфере анализа и мониторинга рынка, нуждаются в эффективных инструментах управления внутренними процессами. Одним из ключевых факторов успешной деятельности предприятия является прозрачность и контроль выполнения задач сотрудниками.

Компания ООО «Джи Эм Трейд» активно развивается в области предоставления аналитических решений и сервисов для клиентов, однако в процессе внутренней организации работы сотрудников возникает потребность в специализированной системе управления задачами. На данный момент для этих целей используется Bitrix, функционал которого избыточен для предприятия и не соответствует его экономическим и организационным особенностям.

В связи с этим актуальной является разработка собственного веб-приложения, которое позволит упростить постановку и контроль выполнения задач, повысить эффективность работы менеджеров, сократить количество просроченных заданий и обеспечить оперативное взаимодействие между руководителями и исполнителями.

Предлагаемая система должна обладать удобным интерфейсом, поддержкой реальных уведомлений и календарным отображением задач. Она будет включать функционал управления пользователями, назначение ролей, гибкую настройку статусов задач и экспорт статистики. Таким образом, проектируемое решение позволит компании повысить внутреннюю производительность, снизить затраты на сопровождение сторонних систем и адаптировать инструмент под реальные потребности предприятия.

К входным данным относятся сведения, вводимые пользователями, т.е. регистрационные данные, параметры создаваемых задач — название, описание, исполнитель, срок выполнения, а также изменения статусов и комментарии.

К выходным данным относятся результаты обработки информации системой, уведомления о создании и изменении задач, статусы выполнения, отчёты, статистика и экспорт данных в формате CSV, а также календарное представление задач.

# Характеристика предприятия

Компания ООО «Джи Эм Трейд» — это динамично развивающаяся организация, специализирующаяся на анализе и мониторинге рынка. Основное направление деятельности предприятия — сбор, обработка и предоставление данных о компаниях, товарных позициях и их характеристиках. Это позволяет клиентам принимать обоснованные бизнес-решения на основе достоверной и актуальной информации. Компания работает в нескольких направлениях: исследование и анализ рыночной ситуации, предоставление клиентам инструментов для управления характеристиками товаров и компаний, а также разработка решений для интеграции и управления данными.

Организационная структура предприятия включает руководство, финансовый отдел, IT отдел, отдел продаж и маркетинга. В руководстве находятся генеральный директор и финансовый директор. Финансовый отдел состоит из специалистов по обработке данных и экспертов по рыночному анализу. Технический отдел включает разработчиков программного обеспечения, администраторов баз данных и специалистов по DevOps. Отдел продаж и маркетинга представлен менеджерами по работе с клиентами и специалистами по продвижению услуг. Структура компании представлена на рисунке далее.

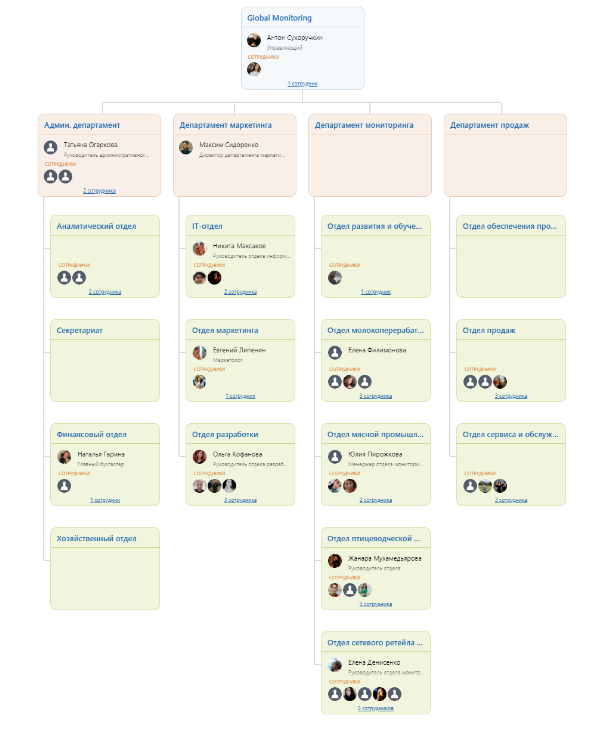


Рисунок 1 – Структура компании

# Техническое задание

Веб-приложение для управления задачами и мониторинга статусов выполнения задач сотрудниками. Разработка ведется на основании поручения руководства предприятия с целью оптимизации процессов постановки и выполнения задач сотрудниками.

Полное наименование: Веб-приложение для управления задачами и мониторинга статусов выполнения задач сотрудников.

Условное обозначение: Система «BananaTasks».

Система позволит руководителям, в удобной форме выставлять новые задачи для менеджеров, и следить за их выполнением.

Начало разработки: 01.09.2025 г.

Завершение разработки и ввод в опытную эксплуатацию: 26.09.2025 г.

Функции:

1. Создавать и редактировать задачи.
2. Назначать исполнителей и устанавливать дедлайны.
3. Отслеживать статус выполнения задач.
4. Получать уведомления в реальном времени о прогрессе (например, о выполнении задачи, задача просрочена, все задачи выполнены).
5. Отображать задачи в календарной форме.

Разработанное веб-приложение позволяет систематизировать задачи менеджеров, обеспечить контроль сроков выполнения и прозрачность рабочих процессов. Благодаря календарю и оповещению в реальном времени снижается количество просроченных задач, ускоряется обмен информацией, что повышает эффективность работников.

До сих пор сотрудники используют Bitrix, его использование для компании является избыточным и экономически нецелесообразным. Система включает CRM, телефонию и множество модулей, которые не принимаются в деятельности предприятия, однако увеличивают сложность освоения и стоимость. В то время как разработка специализированного веб-приложения позволяет учесть конкретные потребности компании, минимизировать стоимость владения и повысить удобство использования для сотрудников.

Оптимизация процесса постановки задач менеджерам и контроля их исполнения и снижение количества просроченных задач являются главными целями создания системы.

Кейсы возможные на сайте представлены ниже.

1. Регистрация: хэшируем пароль, создаем пользователя, назначаем роль.
2. Авторизация: сверяем пароли, выдаем jwt и refresh токен, создаем запись в refresh\_tokens.
3. Управление пользователями: получение всех пользователей. Изменение роли, создаем notifications.
4. Создание задачи: создание задачи, создание уведомления, публикация через websocket.
5. Редактирование задачи: изменение описания, названия, дедлайна
6. Изменения статуса задачи: изменения статуса, создание уведомления в бд для менеджера, публикация в websocket.
7. Удаление задачи.
8. Просмотр всех задач, руководителем.
9. Просмотр всех задач, администратором с помощью elasticsearch(user, name, дедлайн).
10. Просмотр всех задач, которые прендалежат менеджеру.
11. Просмотр календаря.
12. Экспорт статистики в csv.
13. Уведомление выполняющему, что задача создана.
14. Уведомление руководителю, что статус задачи изменен.
15. Уведомление выполняющему, что дедлайн изменен.
16. Крон, чтобы выставлять статус Просрочена, у задач, которых кончился дедлайн .
17. RBAC чтобы, если что изменить роль с мененджера на руководителя или наоборот.

Входные данные предаствлены ниже:

* регистрационные данные пользователей (ФИО, email, пароль, роль);
* учетные данные для авторизации (логин, пароль, токен);
* данные о задачах (название, описание, исполнитель, дедлайн, приоритет, статус);
* команды администратора (назначение ролей, изменение прав доступа);
* параметры поиска и фильтрации (ФИО пользователя, название задачи, дедлайн);
* данные для изменения задачи (новое описание, дедлайн, статус);
* команды по экспорту статистики (выбор периода, тип отчёта).

Выходные данные выглядят так:

* уведомления в реальном времени (создание задачи, изменение дедлайна, изменение статуса, просрочка);
* список пользователей с ролями;
* карточка задачи с полной информацией;
* персональный список задач для каждого менеджера;
* агрегированный список задач для руководителя и администратора;
* календарное представление задач;
* статистические отчёты по задачам (выполненные, просроченные, активные);
* экспортированные файлы в формате CSV.

В первую очередь должны быть реализованы базовые функции: регистрация и авторизация, создание и редактирование задач, назначение исполнителей, система уведомлений. После этого — функции календаря, поиска, экспорта статистики. Такая очередность позволяет быстрее ввести систему в опытную эксплуатацию и постепенно расширять функционал.

# Информационное обеспечение

Разрабатываемое веб-приложение использует реляционную базу данных PostgreSQL, в которой определены несколько ключевых сущностей. Каждая сущность представлена таблицей с набором атрибутов, а между таблицами установлены связи через внешние ключи. Такая организация данных позволяет обеспечивать целостность информации и эффективно обрабатывать пользовательские запросы.

Пользователи являются центральной сущностью системы, так как именно они формируют задачи, назначают исполнителей и контролируют выполнение. Для обеспечения безопасности в таблице хранится зашифрованный пароль, а также определяется роль пользователя (администратор, руководитель, менеджер), от которой зависят его права в системе. Имеет связь с таблицей отделов. Таблицу можно наблюдать ниже.

Таблица 1 – Структра модели пользователей

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Наименование поля | Тип данных | Размер |
| 1 | 2 | 3 | 4 |
| ID пользователей (PK) | id | number | 4 байт |
| Адрес электронной почты | email | string | 65 535 байт |
| Пароль | passwordHash | string | 65 535 байт |
| Часовой пояс | timezone | string | 65 535 байт |
| Имя и фамилия | fullName | string | 65 535 байт |
| Аквность | isActive | boolean | 1 байт |
| Отдел (FK) | departmentId | number | 4 байт |

Задачи являются основной сущностью веб-приложения. Каждая задача имеет уникальный идентификатор, название, описание, сроки выполнения и статус. Кроме того, она связана с пользователем-постановщиком и пользователем-исполнителем, что позволяет контролировать ответственность за выполнение. Структура представлена ниже.

Таблица 2 – Структра модели задач

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Наименование поля | Тип данных | Размер |
| 1 | 2 | 3 | 4 |
| ID задач (PK) | id | number | 4 байт |
| Название задачи | title | string | 65 535 байт |
| Описание задачи | description | string | 65 535 байт |
| Статус | status | string | 65 535 байт |
| Приоритет | priority | number | 4 байт |
| 1 | 2 | 3 | 4 |
| Дедлайн | deadline | datetime | 4 байт |
| Исполнитель | assignedTo | number | 4 байт |
| Постановщик | createdBy | number | 4 байт |

Таблица ролец используется для хранения перечня возможных ролей в системе (администратор, руководитель, менеджер). С её помощью реализуется механизм ролевого управления доступом (RBAC). Описание таблица далее.

Таблица 3 – Структра модели ролей

|  |  |  |  |
| --- | --- | --- | --- |
| Идентификатор | Наименование поля | Тип данных | Размер |
| 1 | 2 | 3 | 4 |
| ID роли (PK) | id | number | 4 байт |
| Наименование роли | name | string | 65 535 байт |
| Описание | description | string | 65 535 байт |

Для функционирования разрабатываемого веб-приложения требуется сервер с установленной СУБД PostgreSQL, а также клиентские устройства пользователей. Минимальные характеристики аппаратного обеспечения следующие. Для серверной части:

* 4-ядерный процессор;
* оперативная память от 8 гб;
* дисковое пространство от 100 гб;
* операционная система Windows.

А для клиентской части:

* устройство с доступом в Интернет;
* современный веб-браузер;
* минимальное разрешение экрана 1366x768.

Такие характеристики обеспечивают корректную работу системы при одновременном использовании приложения десятками пользователей и позволяют масштабировать её при необходимости. Полную схему можно увиедеть в [Приложении А](#_Приложение_А).

К информации, вводимой пользователями с клавиатуры, относятся:

* регистрационные данные (ФИО, email, пароль, роль);
* учетные данные при авторизации (логин, пароль);
* параметры создаваемых задач (название, описание, исполнитель, дедлайн, приоритет);
* команды изменения статусов задач (например, «Выполнена» или «Просрочена»).

Формы ввода реализуются в интерфейсе веб-приложения и в заполненном виде помещаются в [Приложение В](#_Приложение_В).

Следует отметить, что часть вводимой информации используется для обработки, но не сохраняется в БД. К примеру, фильтры поиска, временные параметры экспорта или временные токены сеанса не подлежат хранению на постоянных носителях.

К выходной информации относятся:

* списки задач (персональные для сотрудников и для руководителей);
* карточки задач с детальной информацией;
* уведомления в реальном времени (о создании, изменении дедлайна, смене статуса, просрочке);
* статистические отчеты по задачам (количество выполненных, активных и просроченных)

Каждый выходной документ имеет свою структуру, разработанную на основании требований заказчика. Например, CSV-отчет включает заголовки: id задачи, название, исполнитель, статус, дедлайн, приоритет, комментарий. Структуры выходных файлов представлены в [Приложении Г](#_Приложение_Г).

# Функционирование программы

Разрабатываемое веб-приложение предназначено для управления задачами и мониторинга их выполнения сотрудниками компании. Основная цель программы — обеспечить прозрачность рабочих процессов, контроль сроков выполнения задач, а также своевременное информирование пользователей о событиях в системе. Программа функционирует как интегрированная система, включающая серверную часть, клиентский интерфейс и базу данных.

Особое внимание при проектировании было уделено построению системы в соответствии с принципами чистой архитектуры, что позволило добиться высокой модульности и удобства сопровождения. В рамках данной концепции выделяются несколько уровней: Domain, где сосредоточена бизнес-логика, сущности, перечисления и value objects; Application, содержащий сценарии работы с доменной моделью (use cases); Infrastructure, включающая репозитории, сервисы и адаптеры для взаимодействия с внешними системами и базой данных; а также Interface, представленный контроллерами и резолверами, которые выполняют роль входных точек (HTTP/GraphQL). Благодаря такому разделению обеспечивается слабая связанность модулей и высокая степень повторного использования кода, что особенно важно при масштабировании и дальнейшем развитии проекта.

Особое место в системе занимает база данных, реализованная с использованием PostgreSQL и описанная при помощи Prisma ORM. Она содержит ключевые сущности, которые отражают логику предметной области: пользователи, роли, задачи, уведомления, токены обновления и департаменты. Между таблицами установлены чёткие связи, позволяющие поддерживать целостность и консистентность данных. Так, каждый пользователь может относиться к конкретному департаменту, быть назначенным исполнителем или автором задачи, обладать несколькими ролями и получать уведомления о событиях, связанных с его деятельностью.

Каждый пользователь системы принадлежит к определённому департаменту и обладает конкретной ролью. В модели предусмотрены следующие роли: администратор, менеджер, руководитель департамента, разработчик и продавец. Возможности пользователей по ролям далее:

* администратор управляет пользователями, может изменять роли, контролировать все задачи и просматривать статистику;
* руководитель департамента создаёт задачи для сотрудников своего подразделения, назначает исполнителей и отслеживает прогресс;
* менеджеры, разработчики и продавцы выступают исполнителями задач и взаимодействуют с системой в части выполнения поручений.

Ролевое разграничение обеспечивает гибкость системы и реализует RBAC-модель (Role-Based Access Control), что гарантирует доступ только к тем функциям, которые соответствуют должностным обязанностям сотрудника.

Основной бизнес-процесс приложения строится вокруг сущности Task. Руководитель или администратор формирует задачу, указывая название, описание, приоритет и дедлайн. В момент создания задача получает статус NEW. В базе данных фиксируется идентификатор автора (createdById) и назначенного исполнителя (assignedToId).

Если руководитель указывает конкретного сотрудника, система сохраняет связь между пользователем и задачей. Таким образом, в дальнейшем пользователь получает доступ к своей «личной» подборке задач.

Исполнитель имеет возможность изменять статус задачи: IN\_PROGRES (в работе), DONE (выполнена) или OVERDUE (просрочена). Статус меняется как вручную, так и автоматически. Например, при наступлении дедлайна специальный крон-процесс переводит невыполненные задачи в статус OVERDUE.

Руководитель может изменять условия задачи (описание, дедлайн, приоритет), а также удалять задачи при необходимости.

Таким образом, система отражает полный жизненный цикл задачи — от момента её постановки до завершения или перевода в категорию просроченных.

Цепочка действий представлена далее:

1. Пользователь вводит данные через веб-форму.
2. Клиентский интерфейс отправляет запрос на сервер.
3. Сервер проверяет права доступа и валидирует данные.
4. Данные сохраняются или обновляются в базе данных.
5. Генерируются уведомления и события WebSocket.
6. Клиент получает обновлённые списки задач и уведомления в реальном времени.
7. Фоновые процессы (cron) проверяют дедлайны и автоматически обновляют статусы.
8. Пользователь просматривает обновленную информацию через интерфейс или экспортирует данные в CSV.

Существенную роль в бизнес-логике играет система уведомлений. Каждое значимое событие фиксируется в таблице Notification, что обеспечивает прозрачность процессов и оперативное информирование сотрудников.

Список возможных уведомлений в системе выглядит так:

* при создании задачи назначенному исполнителю направляется уведомление о том, что ему поручено новое задание;
* при изменении дедлайна или описания исполнитель получает соответствующее уведомление;
* при завершении задачи или изменении её статуса руководитель уведомляется о прогрессе работы подчинённых;
* непрочитанные уведомления хранятся в базе данных с флагом isRead = false и отображаются пользователю при входе в систему;

Кроме того, уведомления передаются в реальном времени с использованием WebSocket, что позволяет сотрудникам мгновенно получать информацию о происходящих изменениях без необходимости обновлять страницу.

# Логика работы программы

Логика работы веб-приложения строится вокруг взаимодействия пользователей с задачами и системой уведомлений, обеспечивая прозрачный контроль процессов внутри организации. После авторизации каждый сотрудник получает доступ к системе в зависимости от своей роли: администратор управляет пользователями и назначает права, руководитель формулирует задачи и отслеживает их выполнение, менеджеры и исполнители видят порученные им задания и фиксируют изменения статусов.

Создание новой задачи сопровождается автоматическим формированием уведомления для назначенного исполнителя. Система сохраняет все параметры задачи — описание, дедлайн, приоритет, инициатора — и привязывает её к конкретному пользователю. Если исполнитель меняет статус задачи, например переводит её в состояние «выполнена» или «в работе», уведомление создаётся уже для руководителя, чтобы тот своевременно получил информацию о прогрессе. Аналогичным образом при изменении дедлайна система оповещает исполнителя, чтобы избежать недопонимания и задержек.

Уведомления функционируют в двух режимах: они сохраняются в базе данных, что позволяет пользователю просматривать историю взаимодействия, и параллельно доставляются в режиме реального времени через WebSocket. Такой подход обеспечивает как надёжность фиксации событий, так и оперативность реакции. Кроме того, в системе предусмотрен периодический анализ сроков выполнения с помощью механизма cron-задач. Если дата окончания задания превышена, статус автоматически переводится в «просрочено», а соответствующие уведомления отправляются как руководителю, так и исполнителю.

Руководитель и администратор имеют возможность наблюдать общую картину в виде списка всех задач или календарного представления, что значительно облегчает планирование. При необходимости они могут воспользоваться поиском и фильтрацией, включая интеграцию с Elasticsearch для быстрого доступа к данным. Менеджеры же видят только свои задачи и получают удобный инструмент для личной организации рабочего времени.

Таким образом, бизнес-логика программы объединяет несколько взаимосвязанных процессов: регистрацию и авторизацию пользователей, назначение ролей и прав доступа, формирование и сопровождение задач на всех этапах их жизненного цикла, а также автоматизированную систему уведомлений, поддерживающую постоянную обратную связь между участниками. Центральным элементом является именно согласованность действий: каждое событие — от создания до изменения или завершения задачи — немедленно отражается в системе и в информационных потоках между пользователями. Это позволяет минимизировать риски просрочек, сделать процесс работы более прозрачным и повысить эффективность взаимодействия сотрудников.

# Руководство системного программиста

Разрабатываемая система управления задачами построена на модульной архитектуре с использованием фреймворка NestJS, что обеспечивает чёткое разграничение обязанностей между уровнями и удобство в сопровождении. Основная концепция построена на принципах чистой архитектуры, благодаря чему код остаётся легко расширяемым, тестируемым и поддающимся изменению без необходимости затрагивать всю систему целиком.

В рамках архитектуры выделяются четыре ключевых уровня. Уровень Domain включает бизнес-логику и основные сущности: пользователи, задачи, уведомления, департаменты, роли. Здесь же определяются перечисления, value objects и основные правила предметной области, такие как статус задачи, права доступа или формат хранения времени. Уровень Application реализует сценарии использования (use cases), которые управляют взаимодействием доменной модели с внешними процессами. Примером таких сценариев является создание новой задачи, назначение исполнителя или смена статуса. Уровень Infrastructure отвечает за интеграцию с внешними компонентами: базой данных PostgreSQL через Prisma ORM, системой уведомлений в реальном времени (WebSocket), а также фоновыми cron-процессами, автоматически изменяющими статусы задач при наступлении дедлайнов. Наконец, уровень Interface представлен контроллерами и резолверами, которые реализуют входные точки системы — HTTP- и GraphQL-запросы.

С точки зрения хранения данных в PostgreSQL предусмотрено несколько взаимосвязанных таблиц. Таблица пользователей связана с таблицей департаментов и таблицей ролей, что обеспечивает ролевое разграничение доступа. Таблица задач фиксирует идентификаторы автора и исполнителя, дедлайн, приоритет и статус. В таблице уведомлений хранится информация обо всех событиях системы с отметкой о прочтении. Отдельно предусмотрена таблица refresh\_tokens для организации безопасной авторизации с использованием JWT. Поддержание целостности обеспечивается связями «один ко многим» и «многие ко многим», а также каскадными ограничениями на уровне Prisma ORM.

Для системного программиста важной задачей является сопровождение и развитие системы. В процессе эксплуатации необходимо контролировать корректность работы фоновых задач, которые по cron-сценарию изменяют статусы задач на «просроченные». Необходимо отслеживать стабильность соединений WebSocket, поскольку уведомления в реальном времени являются критически важным элементом работы программы. В случае ошибок или перегрузки необходимо проверять логи серверной части, анализировать метрики и, при необходимости, изменять параметры конфигурации.

Программа предусматривает возможность расширения. Добавление новых типов задач или статусов осуществляется через модификацию доменной модели и последующую миграцию базы данных с помощью встроенного механизма Prisma. Аналогично возможно добавление новых ролей, для чего потребуется изменить ролевую модель RBAC, обновить схемы доступа в коде и внести коррективы в таблицу roles. Важно учитывать, что любое изменение в доменном уровне требует внесения соответствующих правок в сценарии Application и может затронуть адаптеры уровня Infrastructure.

Для обеспечения отказоустойчивости системный программист должен следить за состоянием базы данных и периодически выполнять резервное копирование. Настоятельно рекомендуется развернуть систему мониторинга производительности PostgreSQL, а также предусмотреть стратегию отката миграций. В случае критических ошибок приложения необходимо уметь анализировать трассировки запросов в Prisma и логи NestJS.

Таким образом, руководство системного программиста включает понимание логики взаимодействия уровней архитектуры, поддержку целостности базы данных, контроль работы фоновых процессов и уведомлений, а также знание механизмов расширения и адаптации системы под новые требования. Программный код структурирован таким образом, что основные изменения сосредоточены в пределах конкретных модулей, что облегчает сопровождение и позволяет развивать систему без риска нарушения общей работоспособности.

# Руководство оператора

Оператор системы представляет собой конечного пользователя веб-приложения, который взаимодействует с интерфейсом для выполнения своих должностных обязанностей. В зависимости от роли, закреплённой за конкретным пользователем, оператор может обладать различными возможностями и уровнем доступа к функциям системы. Роли реализуются на основе RBAC-модели (Role-Based Access Control) и определяют, какие операции доступны: от просмотра задач до их создания и управления пользователями.

Работа оператора начинается с авторизации. После ввода учётных данных система выполняет проверку пароля и выдаёт токены доступа. В случае успешного входа пользователь перенаправляется в личный кабинет, где отображается список назначенных задач, уведомления и календарь. Важным элементом интерфейса является панель уведомлений, которая позволяет оперативно отслеживать изменения статусов задач или новые назначения.

В роли исполнителя оператор получает доступ к персональному списку задач. Каждая задача сопровождается названием, описанием, датой дедлайна и текущим статусом. Исполнитель может открыть карточку задачи, внести комментарии и изменить статус — например, перевести задачу в категорию «В работе» или «Выполнено». При изменении статуса автоматически создаётся уведомление для руководителя, что позволяет последнему контролировать процесс выполнения поручений. Если срок выполнения задачи истекает, система автоматически переводит её в статус «Просрочена», и оператор получает соответствующее уведомление.

В роли руководителя оператор имеет расширенный набор функций. Он может создавать новые задачи, указывая исполнителя, приоритет, сроки и описание. После сохранения задачи система фиксирует её в базе данных и отправляет уведомление назначенному сотруднику. Руководитель также может редактировать условия задачи, изменять сроки или назначать других исполнителей. Для удобства управления предусмотрен календарный режим, позволяющий визуально распределять задачи по дням, неделям и месяцам. Это особенно важно при планировании работы подразделения.

Оператор-администратор получает наибольший доступ к функциям системы. Он управляет пользователями, изменяет их роли, контролирует все задачи и имеет возможность выгружать статистику в формате CSV. Для администратора также предусмотрена возможность поиска по задачам с использованием Elasticsearch, что позволяет быстро находить задания по ключевым параметрам, таким как исполнитель, название или дата завершения.

Особое внимание уделено работе с системой уведомлений. Оператор видит список непрочитанных сообщений в личном кабинете, а также получает мгновенные оповещения через механизм WebSocket. Это гарантирует, что информация о новых назначениях, изменениях условий или завершении задач не будет упущена. Прочитанные уведомления сохраняются в архиве и могут быть просмотрены повторно при необходимости.

Интерфейс системы построен таким образом, чтобы минимизировать количество ошибок оператора. Все критические действия сопровождаются подтверждением: например, при удалении задачи система запрашивает подтверждение, чтобы исключить случайное удаление данных. При некорректном вводе данных отображаются подсказки и сообщения об ошибках, что снижает вероятность нарушения целостности информации.

Для корректной эксплуатации приложения оператору необходимо соблюдать несколько правил. Во-первых, рекомендуется регулярно проверять панель уведомлений и календарь, чтобы не допускать накопления просроченных задач. Во-вторых, необходимо своевременно обновлять статусы задач, чтобы руководитель имел актуальное представление о ходе работы. В-третьих, при изменении личных данных или смене пароля оператор должен использовать встроенные средства системы, а не обращаться к сторонним сервисам, чтобы избежать нарушения безопасности.

Таким образом, руководство оператора включает последовательность действий для работы в системе, описание доступных функций в зависимости от роли, правила корректного ввода данных и взаимодействия с интерфейсом. Программа разработана с учётом принципов удобства и прозрачности, что позволяет операторам быстро осваивать систему и эффективно выполнять свои должностные обязанности. Благодаря интеграции календаря, уведомлений в реальном времени и ролевой модели доступа оператор получает мощный инструмент для организации и контроля рабочих процессов.

# Техника безопасности и пожарная безопасность

Разработка и эксплуатация программного обеспечения, несмотря на отсутствие прямого физического контакта с производственным оборудованием, также связана с соблюдением определённых норм техники безопасности и требований пожарной безопасности. Эти правила направлены на создание безопасных условий труда для операторов, программистов и системных администраторов, а также на предотвращение чрезвычайных ситуаций, связанных с использованием вычислительной техники и организацией рабочего пространства.

Работа с веб-приложением предполагает использование персональных компьютеров, серверов и сетевого оборудования. В этой связи важным является обеспечение безопасных условий эксплуатации компьютерной техники. Рабочее место оператора должно быть оборудовано исправным компьютером с корректно работающей системой электропитания. Подключение оборудования допускается только через исправные розетки и сертифицированные кабели. Использование самодельных удлинителей и перегрузка электрических сетей категорически запрещается, так как это может привести к перегреву и возгоранию.

С точки зрения техники безопасности особое внимание уделяется эргономике рабочего места. Длительная работа за компьютером может привести к перенапряжению зрения и нарушению осанки. В этой связи оператор обязан соблюдать режим труда и отдыха: каждые 60 минут работы за экраном необходимо делать короткий перерыв для восстановления зрения и выполнения упражнений для снятия мышечного напряжения. Монитор должен располагаться на уровне глаз, а клавиатура и мышь — на удобной высоте, что снижает риск профессиональных заболеваний.

Важным фактором безопасности является защита данных и недопущение несанкционированного доступа. Пользователи обязаны использовать только свои учётные записи и не передавать пароли третьим лицам. Несоблюдение этих правил может привести к утечке данных и компрометации системы. Администратор должен регулярно обновлять программное обеспечение и антивирусные средства, что снижает риск заражения вредоносными программами и возможных кибератак.

Отдельное место занимает пожарная безопасность. Компьютерная техника и серверное оборудование в процессе эксплуатации выделяют тепло, что требует надлежащей вентиляции и исключения перегрева. В помещениях, где размещается оборудование, необходимо наличие исправной системы вентиляции и кондиционирования. Кроме того, категорически запрещается хранение легковоспламеняющихся жидкостей и материалов вблизи рабочих мест и серверных стоек.

В случае возгорания оператор обязан немедленно отключить оборудование от сети электропитания и сообщить о происшествии ответственному лицу. В серверных помещениях необходимо предусмотреть наличие первичных средств пожаротушения, таких как огнетушители углекислотного типа (ОУ), которые позволяют тушить электрооборудование без риска повреждения аппаратуры. Все сотрудники должны быть проинструктированы о месте нахождения огнетушителей и порядке их применения.

Кроме того, необходимо регулярно проводить инструктажи по технике безопасности и пожарной безопасности для всех сотрудников, работающих с системой. При проведении инструктажей особое внимание уделяется порядку действий в случае обнаружения задымления, запаха гари или неисправности оборудования. Работник обязан немедленно прекратить использование неисправного устройства и сообщить об этом системному администратору или ответственному за охрану труда.

Таким образом, техника безопасности и пожарная безопасность при эксплуатации веб-приложения сводятся к совокупности мер, направленных на сохранение здоровья сотрудников, защиту оборудования от повреждений и предотвращение чрезвычайных ситуаций. Соблюдение данных требований обеспечивает надёжное функционирование системы, минимизацию рисков и создание безопасных условий труда для операторов и администраторов.

# Заключение

В ходе выполнения данной работы была разработана и описана концепция веб-приложения для управления задачами и мониторинга их выполнения сотрудниками компании ООО «Джи Эм Трейд». Актуальность разработки обусловлена потребностью предприятия в специализированной системе, которая учитывает особенности его организационной структуры и позволяет повысить эффективность внутренних процессов.

В отличие от громоздких и избыточных решений, таких как Bitrix, проектируемая система ориентирована на решение конкретных прикладных задач: постановка и контроль выполнения поручений, назначение ответственных сотрудников, отслеживание сроков и оперативное уведомление участников о ключевых изменениях. Такой подход позволяет минимизировать затраты на сопровождение, облегчить освоение системы и обеспечить прозрачность взаимодействия между руководителями и исполнителями.

В работе были детально рассмотрены все основные аспекты разработки системы:

* сформулированы цели и задачи проекта, определены входные и выходные данные, отражающие специфику бизнес-процессов предприятия;
* описана архитектура системы, основанная на принципах чистой архитектуры и разделении на уровни Domain, Application, Infrastructure, Interface, что обеспечивает модульность, удобство сопровождения и возможность масштабирования в будущем;
* разработана информационная модель базы данных, включающая сущности «Пользователь», «Задача», «Уведомление», «Департамент» и другие, а также показаны связи между ними, обеспечивающие целостность и согласованность данных;
* подробно изложено функционирование программы, включая бизнес-логику взаимодействия пользователей, постановку и выполнение задач, обработку статусов, а также систему уведомлений в реальном времени;
* рассмотрены вопросы эксплуатации программного обеспечения: руководство системного программиста, руководство оператора, техника безопасности и пожарная безопасность.

Результаты анализа и проектирования позволяют сделать вывод, что предлагаемая система способна значительно повысить внутреннюю продуктивность предприятия, снизить количество просроченных задач, обеспечить оперативный обмен информацией и прозрачный контроль деятельности сотрудников. Использование календарного отображения, экспорт статистики и гибкая настройка ролей обеспечивают универсальность и адаптивность решения.

Таким образом, разработанное веб-приложение является не просто инструментом для постановки задач, а полноценной системой управления внутренними процессами, способной стать важным звеном в цифровой трансформации компании. Перспективы развития проекта заключаются в дальнейшем расширении функционала — интеграции с внешними сервисами аналитики, внедрении мобильного приложения, использовании технологий машинного обучения для прогнозирования выполнения задач и персональных рекомендаций для сотрудников.

Выполненная работа подтверждает, что использование современных подходов проектирования и технологий (NestJS, PostgreSQL, Prisma ORM, WebSocket, RBAC) позволяет создавать гибкие и надёжные решения, отвечающие актуальным потребностям бизнеса.

# Список используемых источников

1. NestJS Documentation / [Электронный ресурс]: NestJS – Режим доступа: <https://docs.nestjs.com/> - 15.09.2025.
2. Prisma Documentation / [Электронный ресурс]: Prisma – Режим доступа: [https://www.prisma.io/docs](https://www.prisma.io/docs%20-%2016.01.2025)  - 16.09.2025.
3. TypeScript Handbook / [Электронный ресурс]: Microsoft – Режим доступа: <https://www.typescriptlang.org/docs/> - 17.09.2025.
4. MDN Web Docs: JavaScript / [Электронный ресурс]: Mozilla – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/JavaScript> - 18.09.2025.
5. MDN Web Docs: Web APIs / [Электронный ресурс]: Mozilla – Режим доступа: https://developer.mozilla.org/ru/docs/Web/API - 19.09.2025.
6. WebSocket API / [Электронный ресурс]: MDN – Режим доступа: <https://developer.mozilla.org/ru/docs/Web/API/WebSocket> - 20.09.2025.
7. REST API Design Guide / [Электронный ресурс]: Microsoft – Режим доступа: <https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design> - 21.09.2025.
8. JWT Authentication Guide / [Электронный ресурс]: Auth0 – Режим доступа: [https://auth0.com/docs/secure/tokens - 22.09.2025](https://auth0.com/docs/secure/tokens%20-%2022.09.2025).
9. Role-Based Access Control (RBAC) / [Электронный ресурс]: OWASP – Режим доступа: <https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html> - 23.09.2025.
10. Prisma Client API Reference / [Электронный ресурс]: Prisma – Режим доступа: <https://www.prisma.io/docs/orm/prisma-client> - 24.09.2025.

# Приложение А

# *(обязательное)*

# Информационная модель

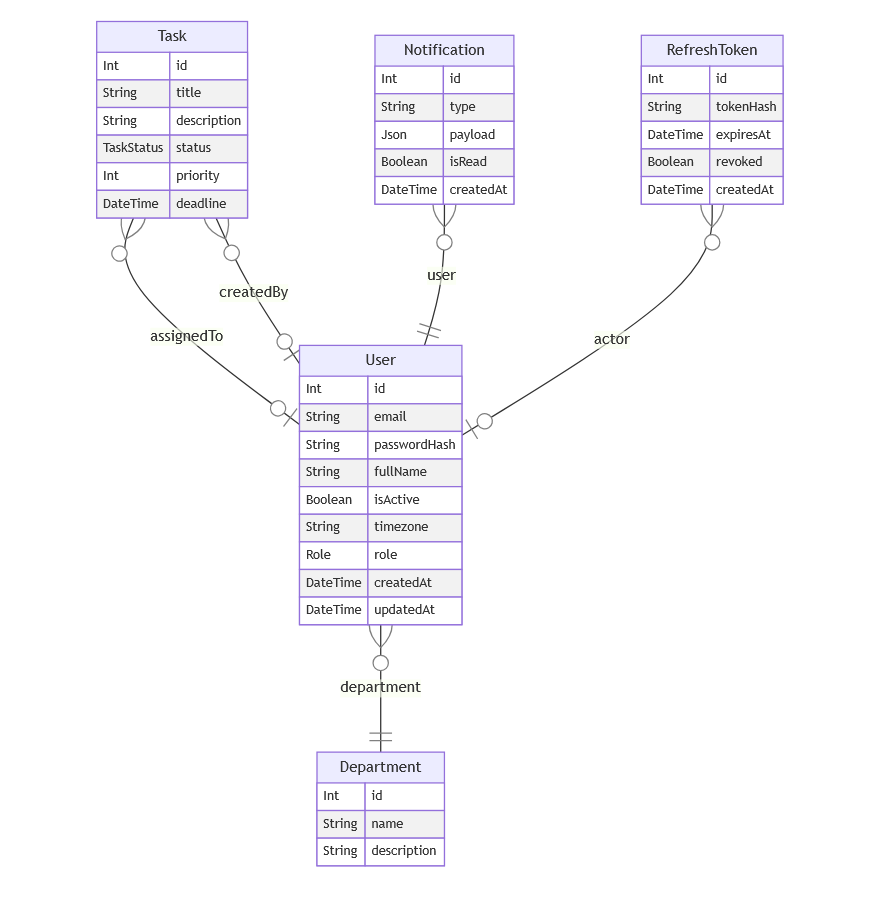


Рисунок А.1 – Информационная модель

# Приложение Б

# *(обязательное)*

# Функциональная модель

(дома вставить)

Рисунок Б.1 – Функциональная модель

(дома вставить)

Рисунок Б.2 – Детализированная функциональная модель

# Приложение В

# *(обязательное)*

# Входные документы



Рисунок В.1 – Входные данные для регистрации

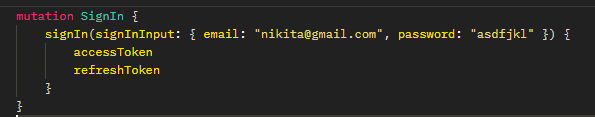


Рисунок В.2 – Входные данные для аутентификации



Рисунок В.3 – Входные данные для создания задачи

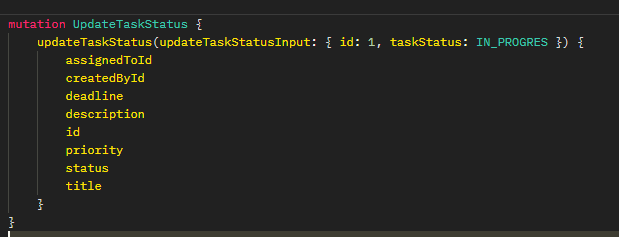


Рисунок В.4 – Входные данные для обновления статуса задачи

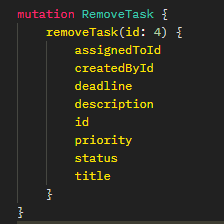


Рисунок В.4 – Входные данные для удаления задачи

# Приложение Г

***(обязательное)***

**Выходные документы**



Рисунок Г.1 – Выходные данные для получение текущего пользователя



Рисунок Г.2 – Выходные данные для задач выполняещего



Рисунок Г.1 – Выходные данные для задач создающего



Рисунок Г.1 – Выходные данные для уведомлений

# Приложение Д

# *(обязательное)*

# Диаграмма прецедентов

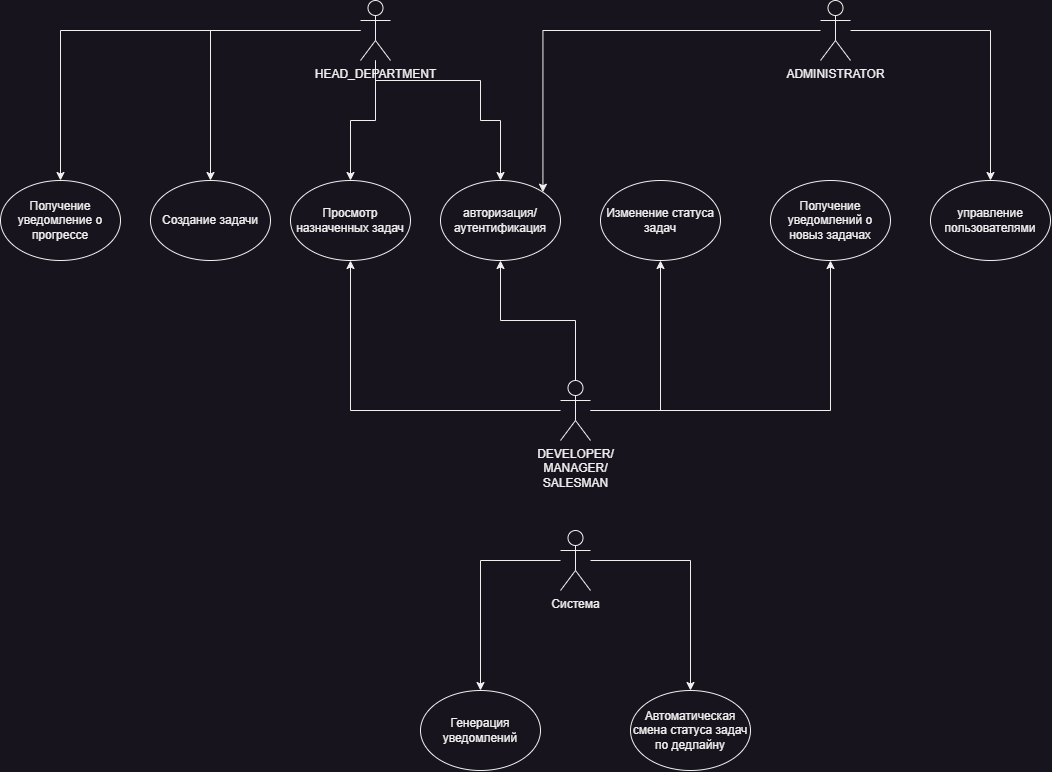


Рисунок Д.1 - Диаграмма прецедентов

# Приложение Е

# *(обязательное)*

# Диаграмма деятельности

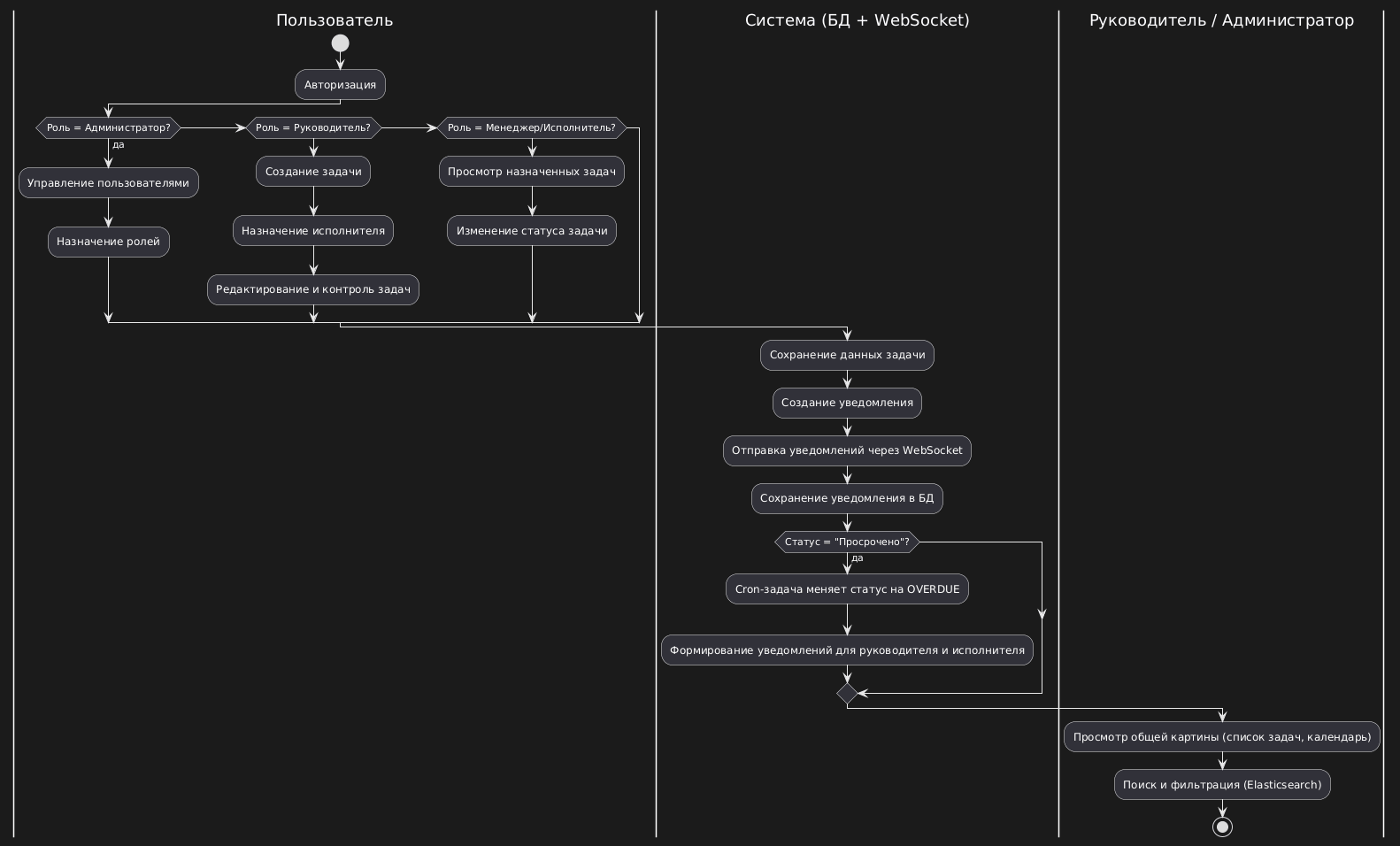


Рисунок Е.1 – Диагрмма деятельности

# Приложение Ё

# *(обязательное)*

# Листинг программы

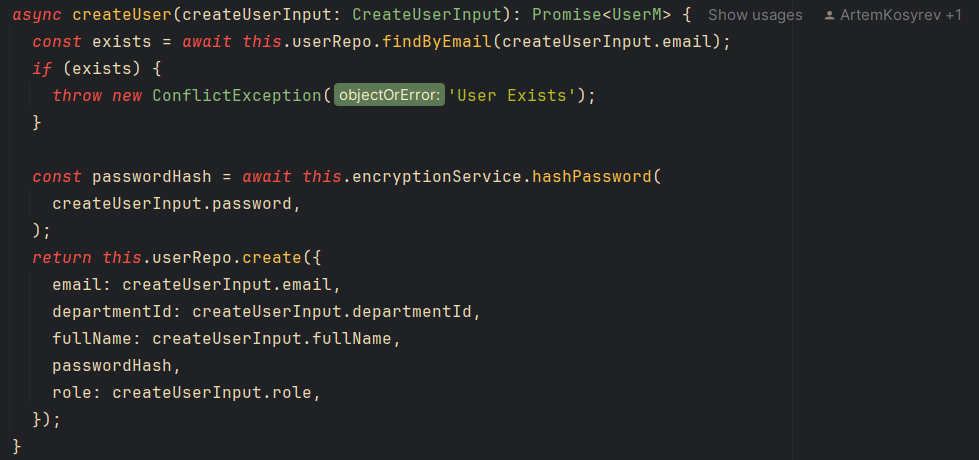


Рисунок Ё.1 – Бизнес логика регистрации пользователя



Рисунок Ё.2 – Бизнес логика входа в систему



Рисунок Ё.3– Бизнес логика создания задачи

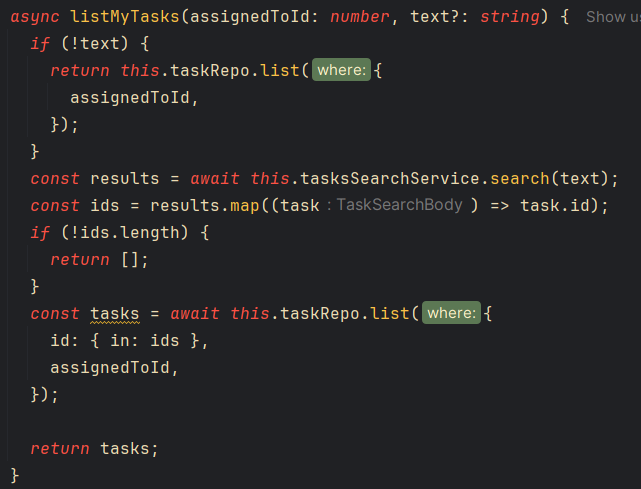


Рисунок Ё.4– Бизнес логика получения назначенных задач

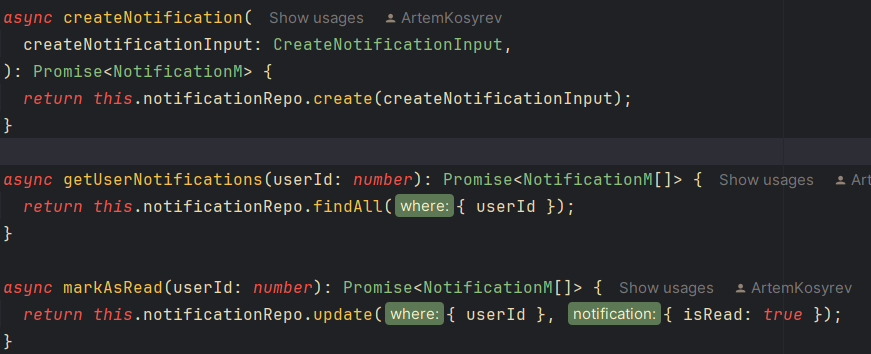


Рисунок Ё.5– Бизнес логика работы уведомлений