**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 7](#_Toc167798101)

[**1.** **Общая часть** 8](#_Toc167798102)

[**1.1. Исследование предметной области** 8](#_Toc167798103)

[**1.2. Анализ требований** 9](#_Toc167798104)

[**2.** **Специальная часть** 11](#_Toc167798105)

[**2.1. Проектирование программного продукта** 11](#_Toc167798106)

[**2.2. Программирование программного продукта** 25](#_Toc167798107)

[**2.3. Тестирование и отладка программного продукта** 31](#_Toc167798108)

[**2.4. Документирование** 42](#_Toc167798109)

[**2.5. Эксплуатация и сопровождение программного продукта** 55](#_Toc167798110)

[**3.** **Охрана труда и техника безопасности при работе на персональном компьютере** 57](#_Toc167798111)

[**ЗАКЛЮЧЕНИЕ** 58](#_Toc167798112)

[**СПИСОК ЛИТЕРАТУРЫ** 59](#_Toc167798113)

# 

# **ВВЕДЕНИЕ**

В любой организации есть свои задачи и обязанности. К сожалению, не все люди в организации могут запомнить все свои обязанности, и очень часто это приводит к нелепым ситуациям и задержкам на производстве. Такая проблема встречается и в создании программного продукта. Программное обеспечение «Дневник разработчика» поможет решить эти неприятные проблемы. С помощью этого программного продукта можно будить следить за разработкой проектов, выдавать задачи сотрудникам, а также вести заметки.

Предметом исследования является программный продукт «Дневник разработчика».

Объектом исследования является разработка приложений.

Целью дипломного проекта является разработка и создание программного продукта под названием «Дневник разработчика», который предназначен для упрощения организации работы в организации или команде разработчиков. Этот инструмент поможет контролировать процесс разработки проектов, распределять задачи сотрудникам, а также вести заметки и отслеживать выполнение задач.

Таким образом, разработка программного продукта «Дневник разработчика» позволит упростить процесс управления проектами, повысить эффективность работы команды разработчиков и избежать ненужных задержек и недоразумений в процессе разработки программного обеспечения.

Задачами дипломного проекта является:

1. Исследование предметной области.
2. Анализ требований необходимых для реализации проекта.
3. Проектирование программного продукта.
4. Программирование программного продукта.
5. Тестирование и отладка программного продукта.
6. Документирование программного продукта.
7. Создание базы данных.
8. Эксплуатация и сопровождение программного продукта.

# **Общая часть**

# **1.1. Исследование предметной области**

Предметная область дипломного проекта «Дневник разработчика» относится к области информационных технологий и создания программных продуктов.

В качестве предметной области была взята работа по организации процесса разработки программных продуктов, а именно процесс распределения задач необходимых для создания программного продукта. Разработка программного продукта – это процесс разработки, уточнения, проектирования, программирования, документирования, тестирования и исправления ошибок, связанных с созданием и поддержкой приложений, фреймворков или других программных компонентов.

Объектом учета информации служат задачи и обязанности сотрудников, а также сам программный продукт, над которым ведётся разработка.

Программа «Дневник разработчика» предназначена для автоматизации нескольких процессов, а именно: для автоматизации процесса распределения задач между работниками, а также для автоматизации организации работы над проектами.

Под учетом подразумевается хранение и обработка информации, присутствующая в базе данных, а также возможность добавления и нахождения любых данных по запросу.

Под обработкой подразумевается любое взаимодействие с данными. А именно добавление данных об проекте.

Основные функции и возможности дипломного проекта:

1. Заявка на создание проекта или проведения технических работ: это информация, предоставляемая заказчиком.
2. Создание проекта, исходя из требований заявки.
3. Определение и создание необходимых задач для выполнения проекта.
4. Распределение задач между сотрудниками.
5. Мониторинг выполнения задач и проектов.
6. Взаимосвязь между сотрудниками по средством внутренней системы связи, для обеспечения более качественного выполнения работ.
7. Создание заметок для сотрудников.
8. Отчётность и информирование администраторов системы о проделанных работах.

Основные аспекты работы дипломного проекта:

* Один работник может выполнять несколько поставленных задач.
* Над одной определённой задачей работает один сотрудник.
* Один проект может содержать несколько задач.
* Работник сам имеет право выбрать задачу, но в тоже время менеджер может назначить на задачу сотрудника.
* Заметки может создавать любой пользователь.

# **1.2. Анализ требований**

Данный продукт наделяет организацию возможностью удобного распределения обязанностей среди сотрудников, удобным учетом проделанных работ над проектом, а также с возможность ведения заметок и взаимосвязи между сотрудниками организации.

Цели – автоматизирование учета данных о сотрудниках, а также автоматизация контроля за проектами, находящимися в разработке. Кроме того, ИС будет реализована функция взаимосвязи сотрудников, а также функция ведения заметок.

Требования к функциональным характеристикам:

1. Программный продукт должен иметь форму авторизации для сотрудников.
2. Не зарегистрированный пользователь может только оставлять заявки на создание проекта или проведения технических работ, оставляя при этом уточняющую информацию о проекте, а также свою личную информацию для обратной связи.
3. В программе должен реализован разный уровень доступа к информации, разграниченный ролями «Сотрудник», «ТимЛид», «Админ». Роль «Сотрудник» должна иметь доступ к просмотру свободных задач, к просмотру забранных задач, к функции написания заметок, а также к функции взаимодействия сотрудников. Роль «ТимЛид» должна иметь доступ к любой информации, связанной с проектами, а именно к самим проектом, к задачам, к функции написания заметок, а также к функции взаимодействия сотрудников, также роль должна иметь право на создание, удаление, редактирование любой информации о проекте. Роль «Админ» должна иметь доступ к информации о сотрудниках, возможность создания нового сотрудника, а также его удаления или редактирования.
4. Программа должна иметь простой и понятный графический интерфейс.
5. Программное обеспечение "Дневник разработчика" должно иметь удобный интерфейс, позволяющий пользователям быстро и эффективно создавать новые проекты, назначать задачи со сроками выполнения, а также делиться информацией с другими членами команды.
6. Должна быть реализована навигационная панель для перехода по разделам программы.

Требования к составу и параметрам технических устройств и библиотек:

1. Операционная система Windows 10 и выше.
2. Системная библиотека .NET 4.7.2 и выше.
3. Процессор: 2.0 ГГц Pentium 4 и выше.
4. Оперативная память 4 ГБ и выше.
5. Место на жестком диске 100 Мб.
6. Доступ к MSSQL серверу.

# **Специальная часть**

# **2.1. Проектирование программного продукта**

Проектирование программного обеспечения - процесс создания проекта [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) (ПО). Проектирование ПО является частным случаем [проектирования](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) продуктов и процессов.

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе выданных заказчиком [требований к ПО](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E) (исходные условия задачи). Эти требования подвергаются анализу.

Проектирование ПО включает следующие основные виды деятельности:

1. выбор метода и стратегии решения;
2. выбор представления внутренних данных;
3. разработка основного [алгоритма](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC);
4. [документирование ПО](https://ru.wikipedia.org/wiki/%D0%94%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D1%86%D0%B8%D1%8F_%D0%BD%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5);
5. [тестирование](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%9F%D0%9E) и подбор тестов;
6. выбор представления входных данных.

В процессе проектирования программного обеспечения для выражения его характеристик используются различные нотации – блок – схемы.

1. ER – диаграммы.
2. UML – диаграммы.
3. DFD – диаграммы, а также макеты.

Диаграмма претендентов играет основную роль в моделировании поведения системы, подсистемы или моделировании класса. Каждая диаграмма показывает множество прецедентов, актёров и отношений между ними.

Диаграммы прецедентов применяются для моделирования вида системы с точки зрения прецедентов (или вариантов использования). Чаще всего это предполагает моделирование контекста системы, подсистемы или класса либо моделирование требований, предъявляемых к поведению указанных элементов.

Для данного проекта была разработана диаграмма претендентов (рисунок 1), которая отображает функционал пользователей в программе.

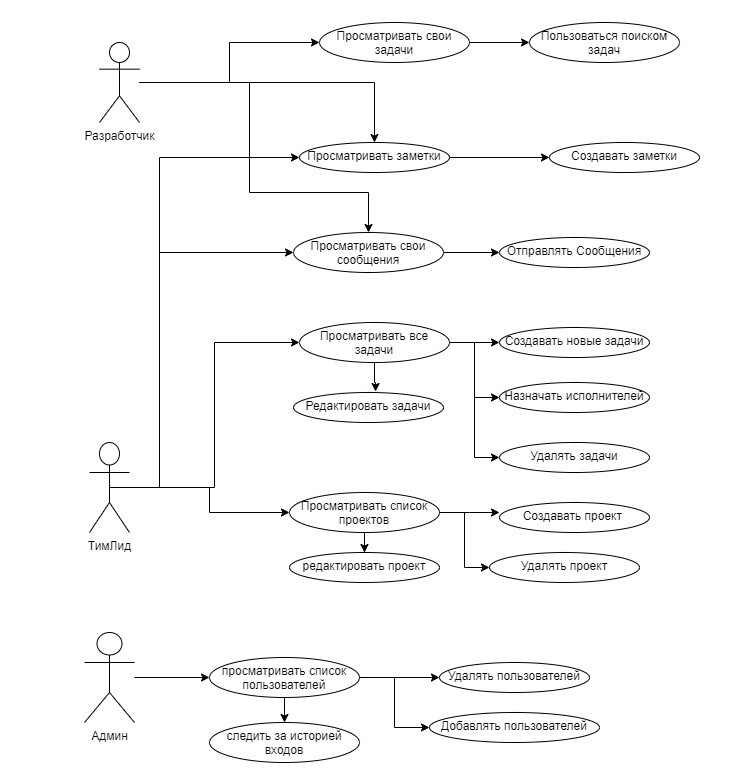


Рисунок 1 – Диаграмма претендентов.

Диаграмма последовательности — диаграмма, на которой показаны взаимодействия объектов, упорядоченные по времени их проявления.

Особенности взаимодействия элементов моделируемой системы могут быть представлены на диаграммах кооперации и последовательности. Диаграммы кооперации используются для спецификации динамики поведения систем, хотя время в явном виде в них отсутствует.

Для данного программного продукта была также разработана диаграмма последовательности на примере процесса создания нового пользователя (рисунок 2).

Диаграммы деятельности представляют собой графическое представление рабочих процессов поэтапных действий и действий с поддержкой выбора.

Они описывают поток управления целевой системой, такой как исследование сложных бизнес-правил и операций, а также описание прецедентов и бизнес-процессов.

Для данного программного продукта была также разработана диаграмма деятельности (рисунок 3).

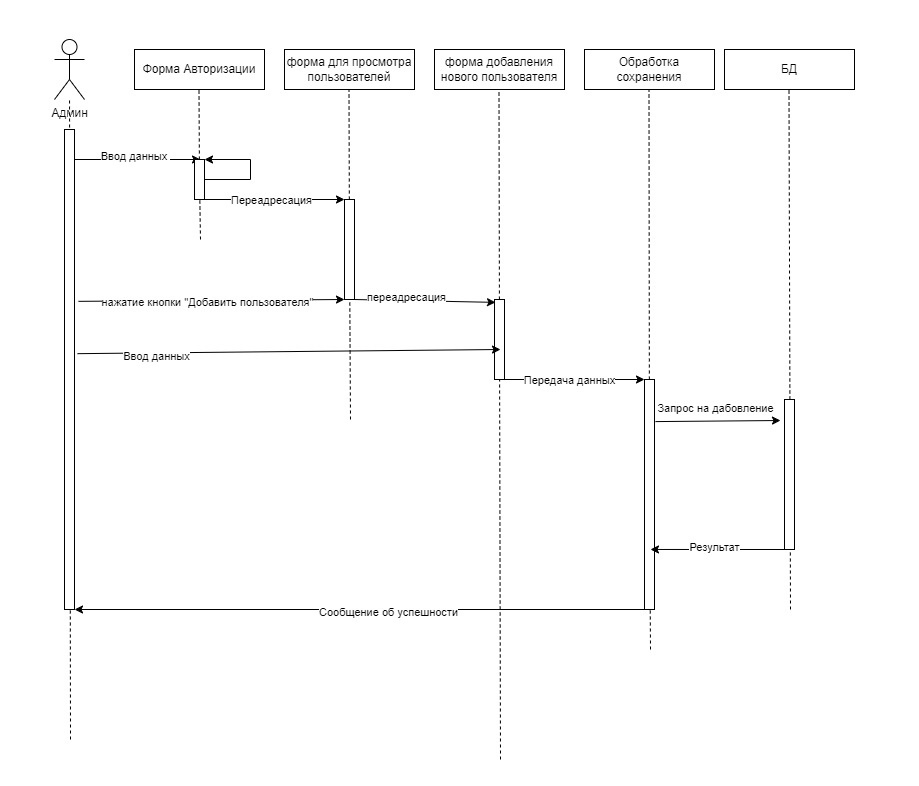


Рисунок 2 – Диаграмма последовательности.

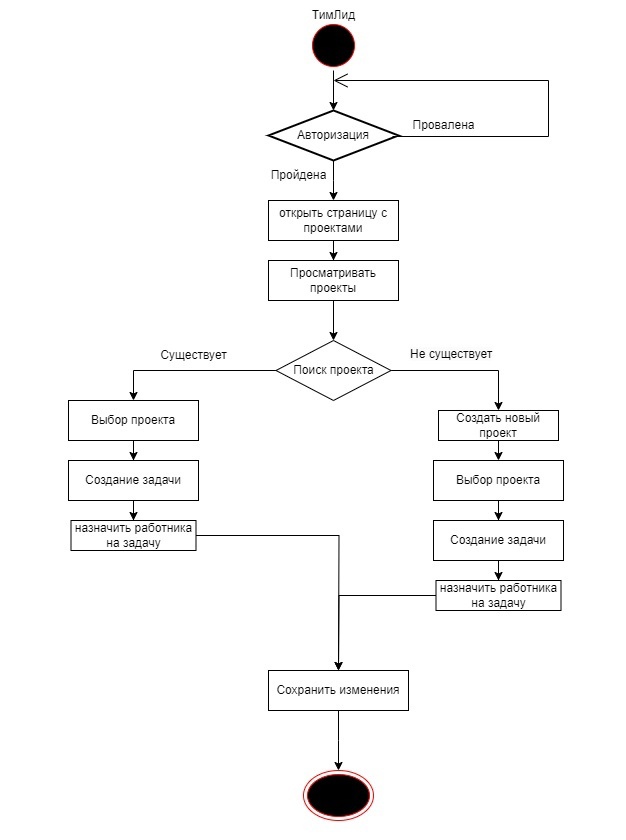


Рисунок 3 – Диаграмма деятельности.

Макетирование – это процесс создания модели требуемого программного продукта. Допустим, у заказчика есть некие требования к работе ПО. По его пожеланиям создаётся макет, который постепенно доводится до степени готовности внесением поправок и дополнительных пожеланий заказчика.

Макет интерфейса – это документ, содержащий в себе информацию о приложении.

В макете представлены страницы интерфейса (рисунок 4 - рисунок 17.)

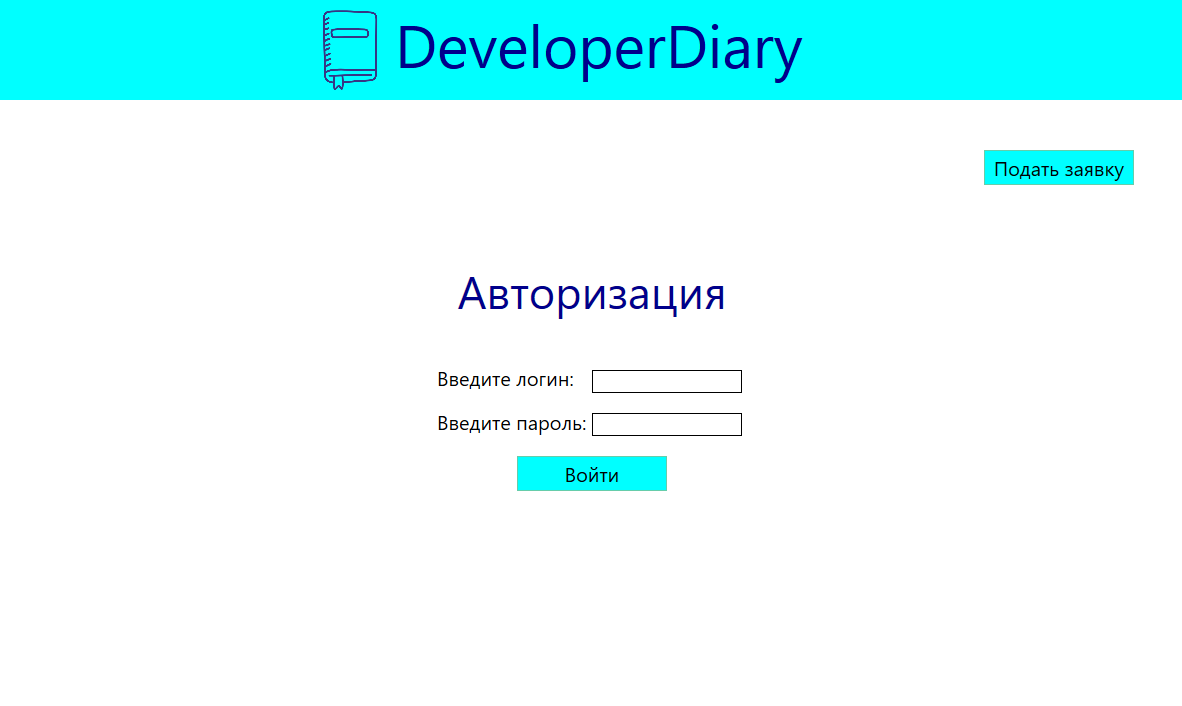


Рисунок 4 –Окно авторизации.

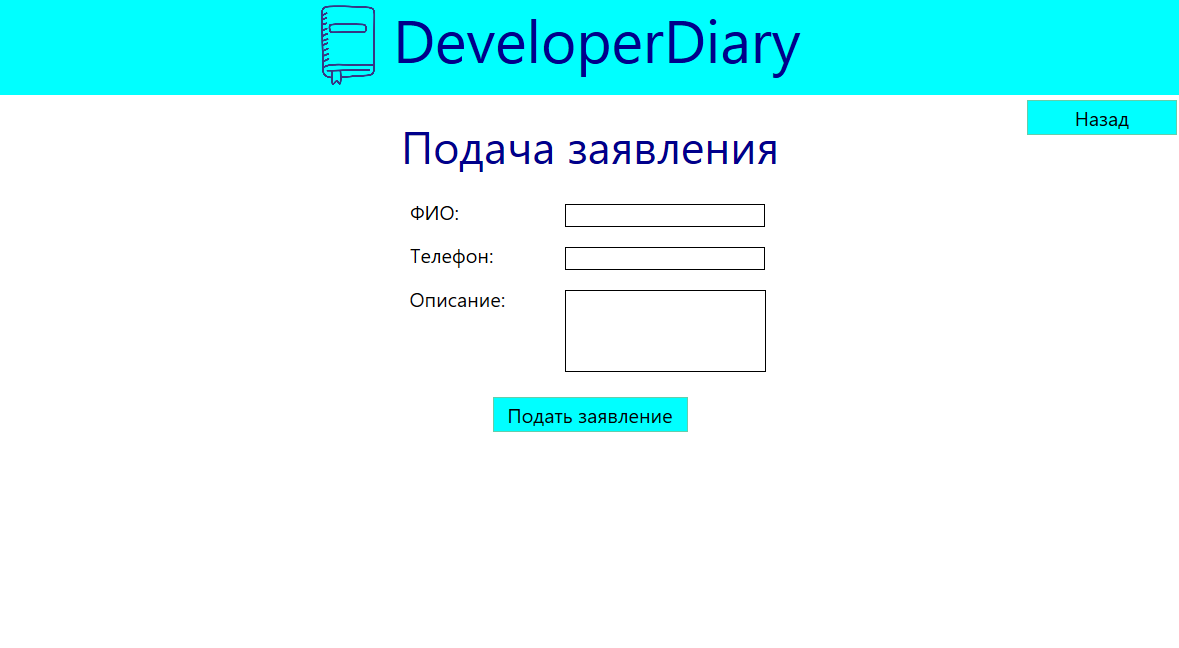


Рисунок 5 –Окно подачи заявления.

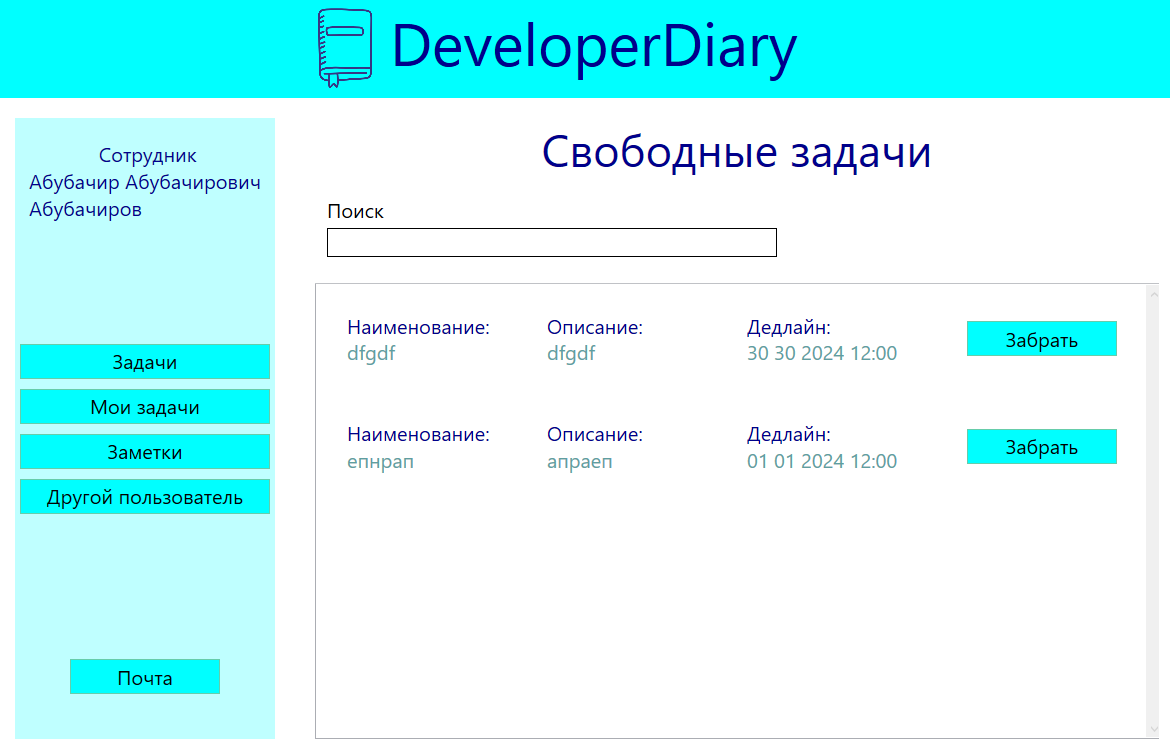


Рисунок 6 –Страница свободных задач у роли «Сотрудник».

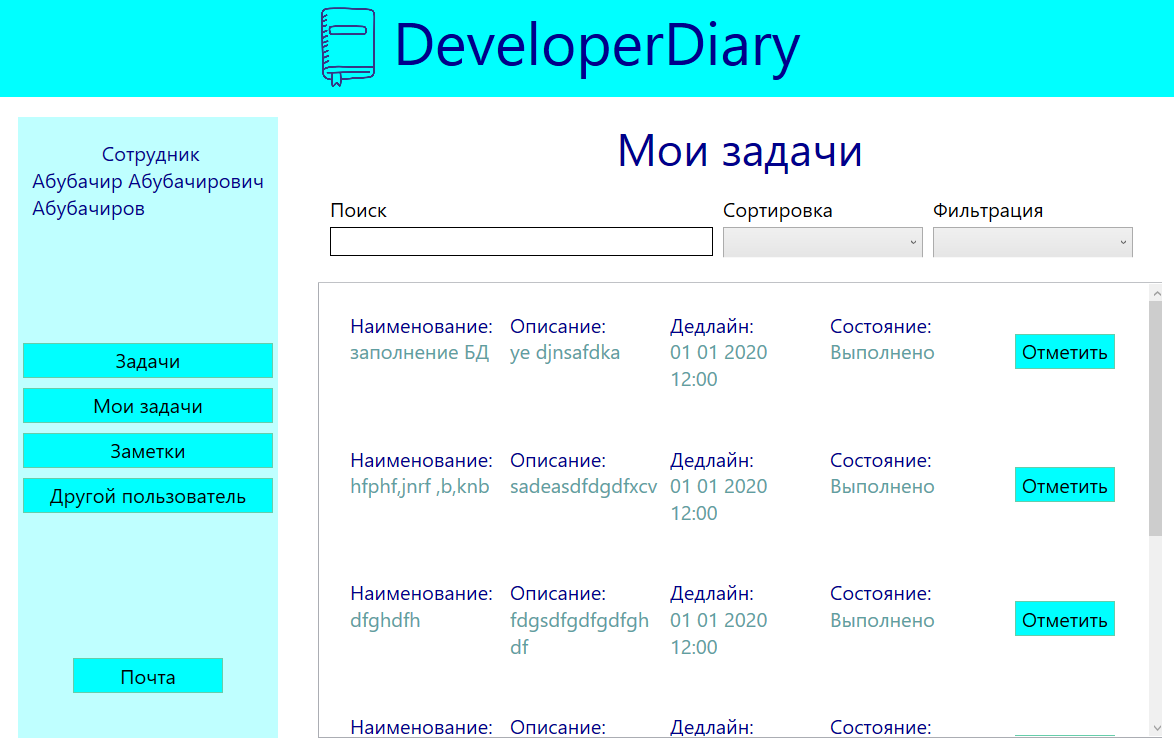


Рисунок 7 –Страница задач «Сотрудника».

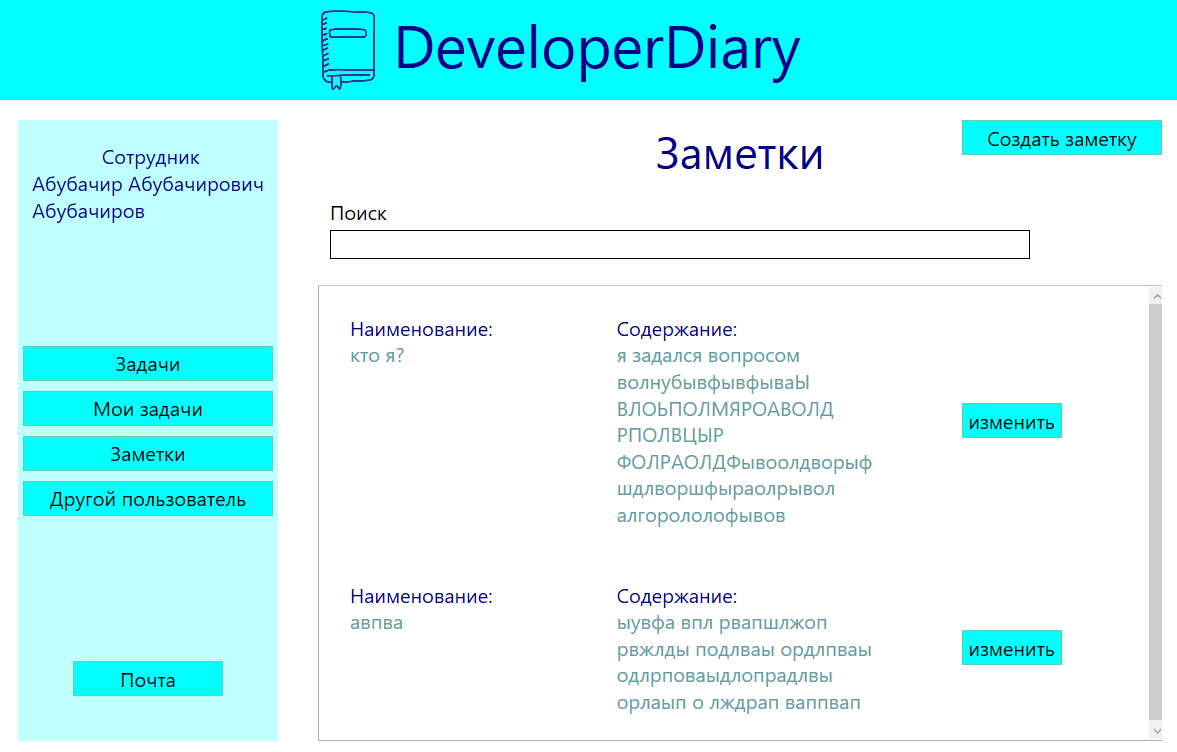


Рисунок 8 –Страница заметок.

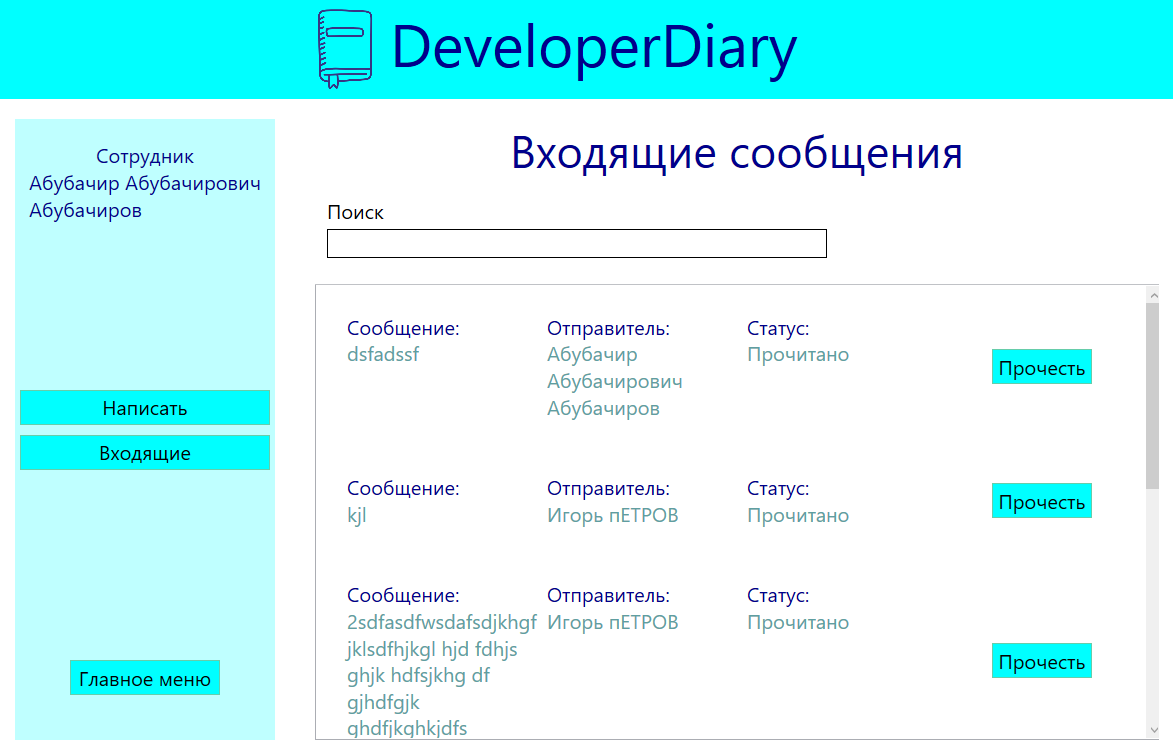


Рисунок 9 –Страница входящих сообщений.

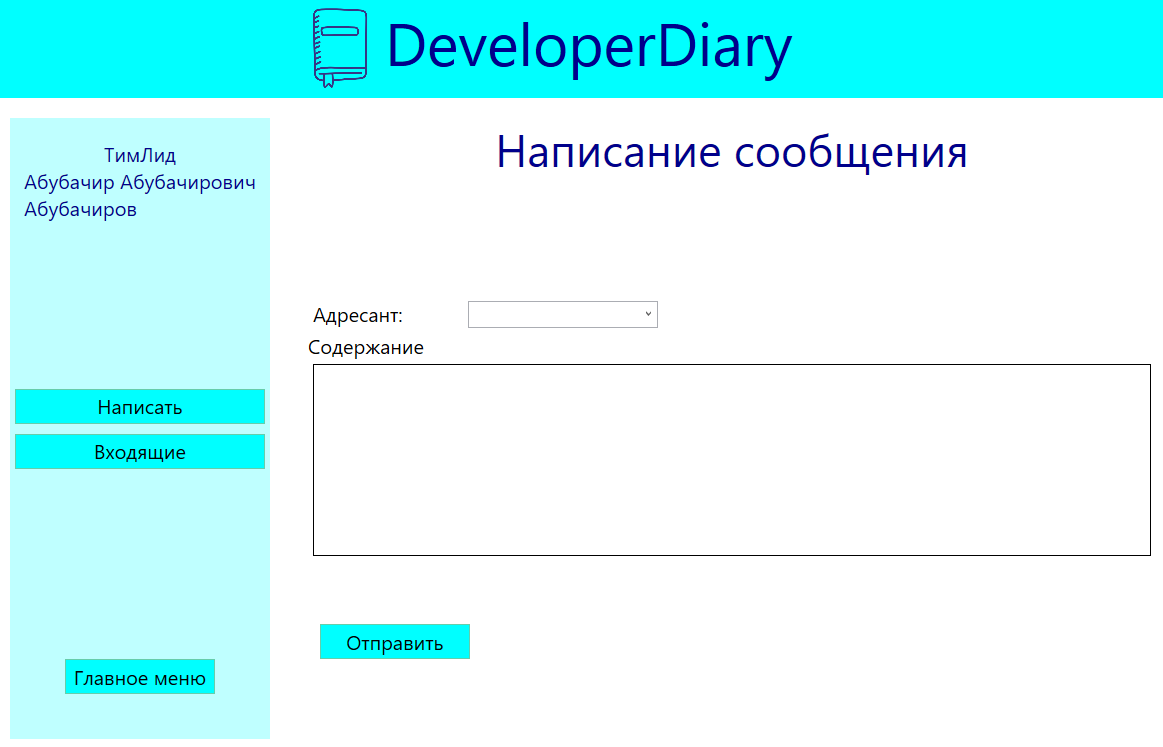


Рисунок 10 –Страница написания сообщений.

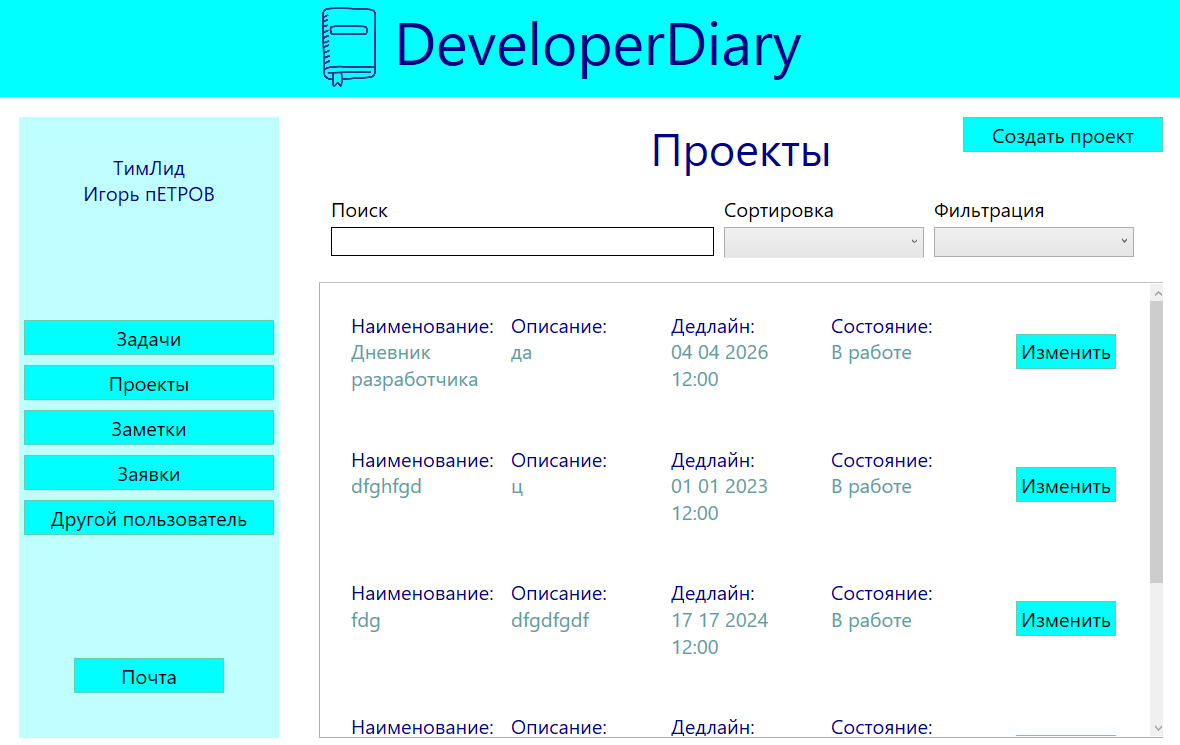


Рисунок 11 –Страница проектов у роли «ТимЛид».

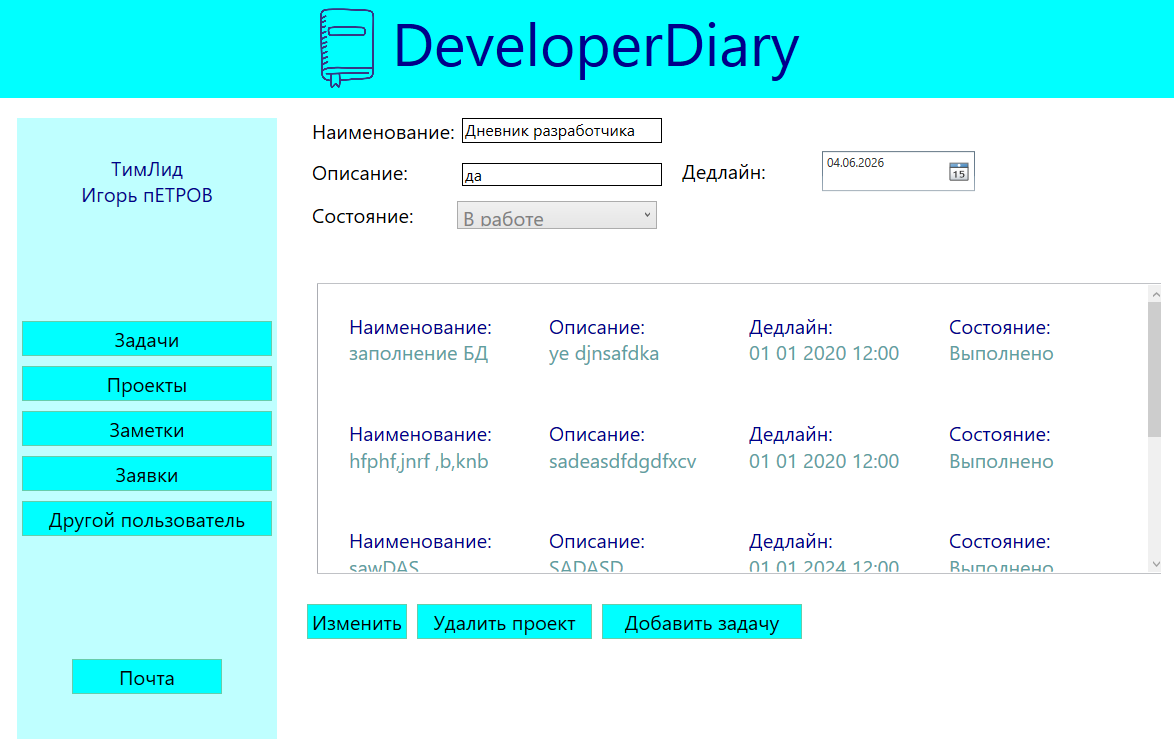


Рисунок 12 –Страница редактирования проекта у роли «ТимЛид».

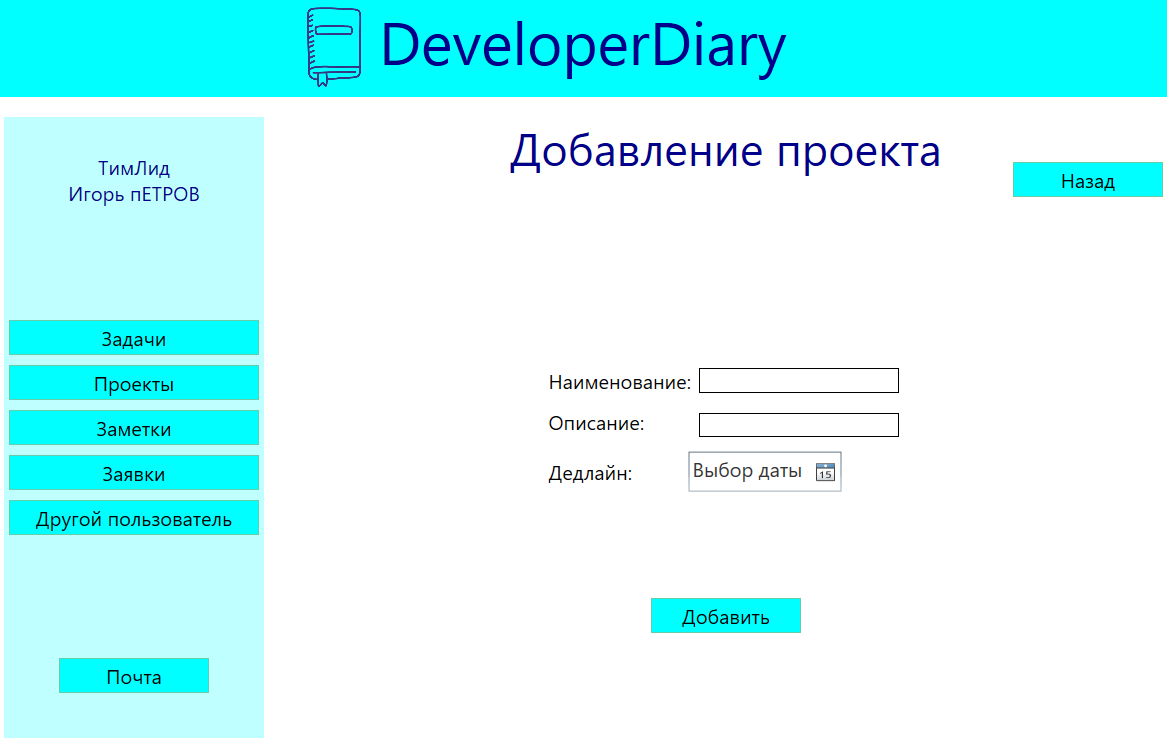


Рисунок 13 –Страница добавления проекта у роли «ТимЛид».

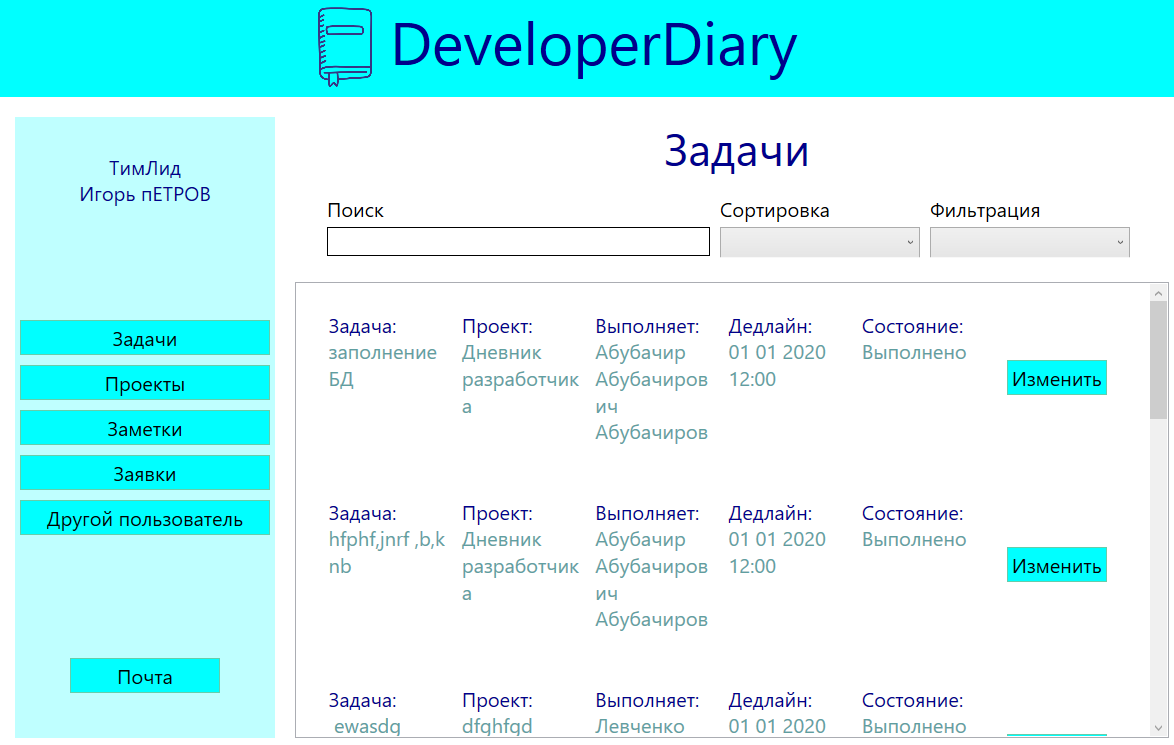


Рисунок 14 –Страница задач у роли «ТимЛид».

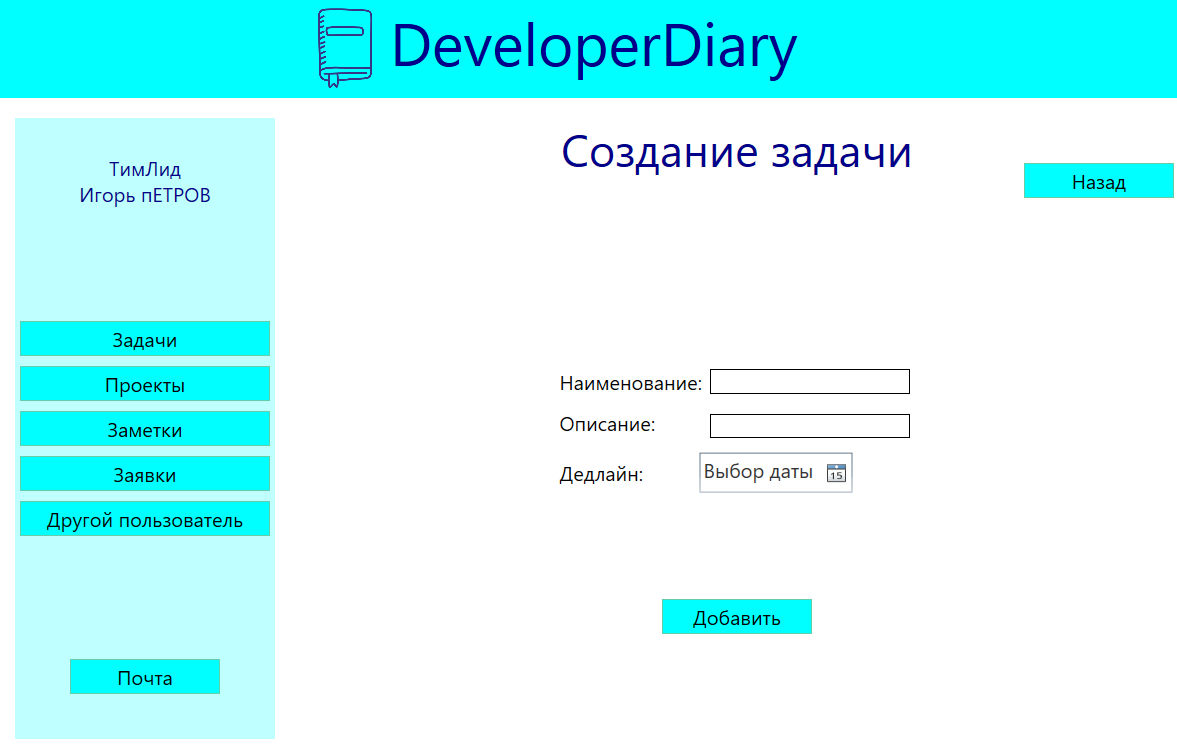


Рисунок 15 –Страница добавления задачи у роли «ТимЛид».

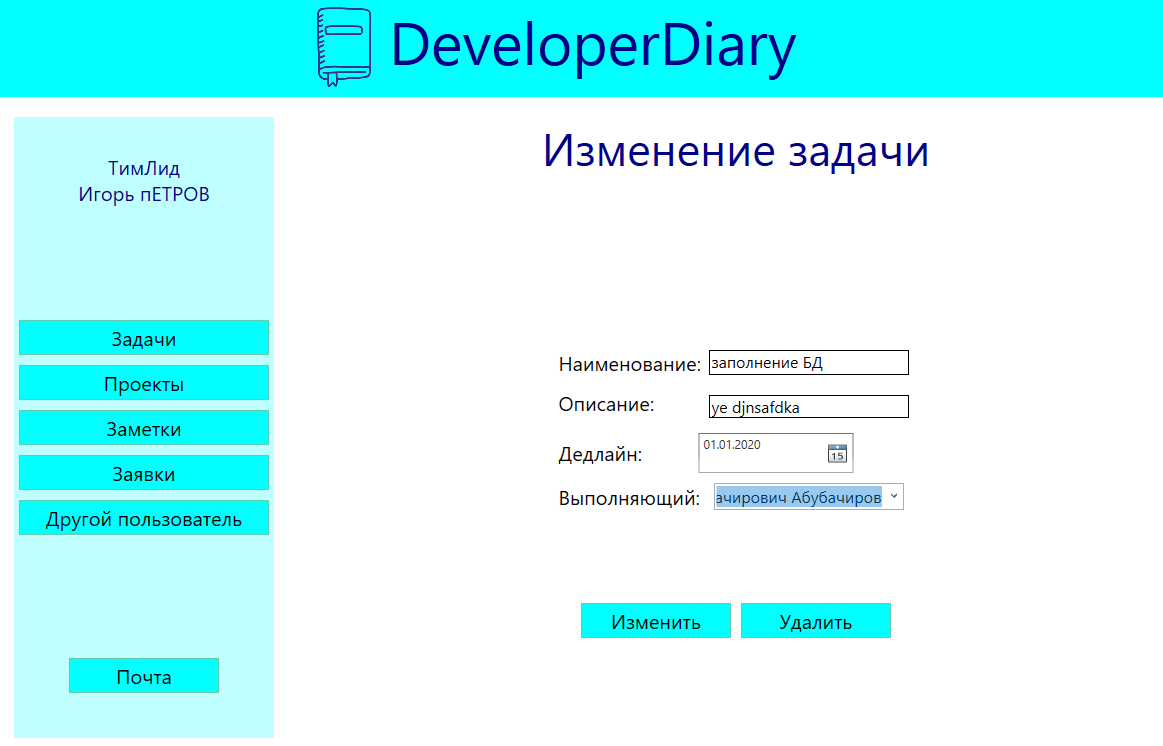


Рисунок 16 –Страница редактирования задачи у роли «ТимЛид».

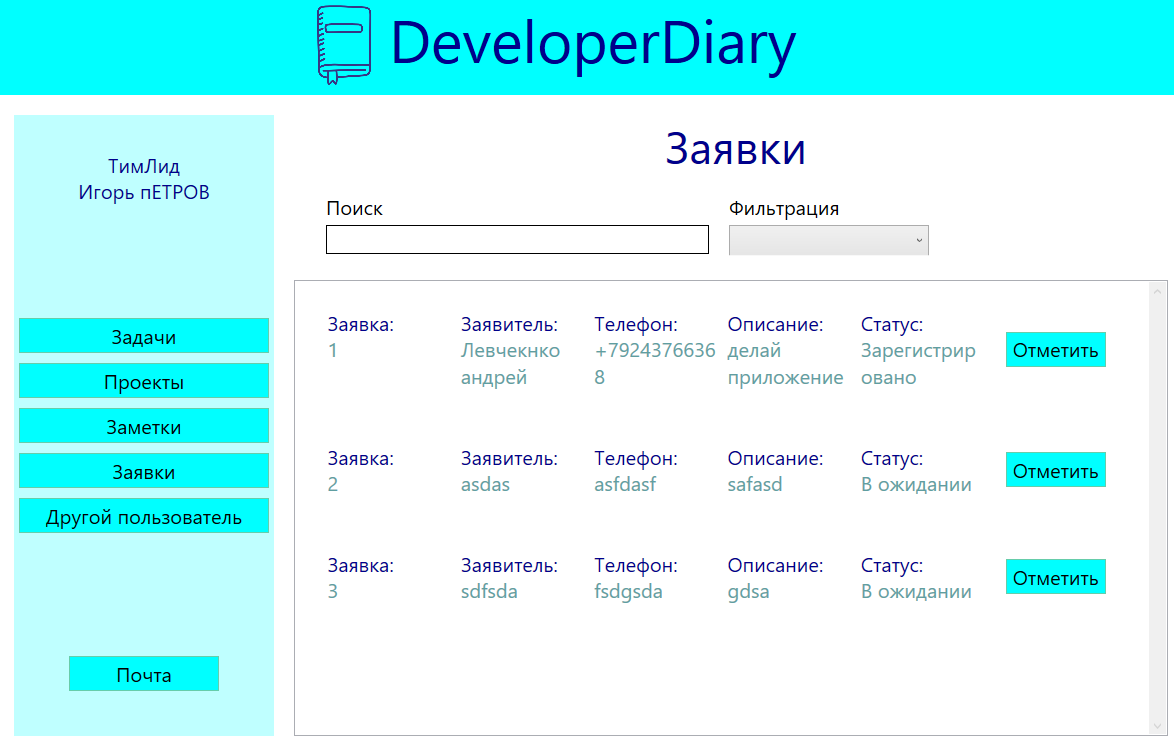


Рисунок 17 –Страница заявок у роли «ТимЛид».

Модульная структура программы – это подход к организации кода, при котором программа разбивается на отдельные модули или компоненты, каждый из которых выполняет определенную функцию или решает определенную задачу.

Модули являются независимыми и могут быть разработаны и отлажены отдельно от остальной программы. Они могут быть написаны на разных языках программирования и могут быть использованы в различных проектах.

Преимущества модульной структуры программы:

1. Модульность: модули позволяют разделить программу на более мелкие и понятные части, что упрощает разработку, отладку и поддержку кода.
2. Повторное использование: модули могут быть использованы в различных проектах, что позволяет избежать дублирования кода и экономит время и усилия разработчика.
3. Улучшенная читаемость: модульная структура программы делает код более организованным и понятным, что облегчает его чтение и понимание.
4. Улучшенная отладка: при наличии модульной структуры программы отладка становится более простой, поскольку ошибки могут быть локализованы и исправлены в отдельных модулях, без необходимости изменения всей программы.

Модульная структура программы может быть реализована с использованием различных подходов, таких как объектно-ориентированное программирование (ООП), где каждый модуль представляет класс или объект, или с использованием функционального программирования, где каждый модуль представляет функцию или процедуру.

Важно отметить, что модульная структура программы требует хорошего планирования и проектирования, чтобы определить, какие функции или задачи должны быть выделены в отдельные модули и как они будут взаимодействовать друг с другом. модель данных, позволяющая описывать концептуальные схемы предметной области.

Для данного программного продукта была также разработана модульная структура, которая была представлена ввиду диаграммы. (рисунок 18).

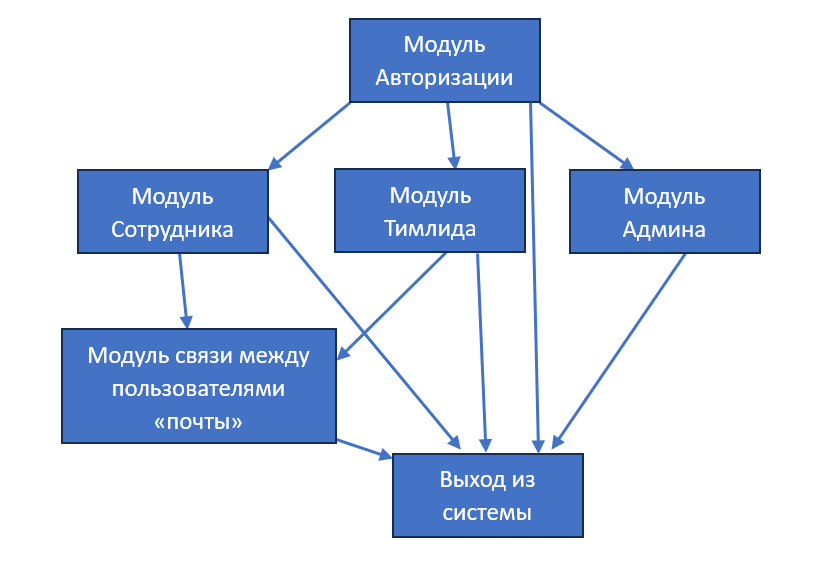


Рисунок 18 – Модульная структура.

ER-модель (от англ. Entity-Relationship model, модель «сущность – связь») – модель данных, позволяющая описывать концептуальные схемы предметной области.

ER-модель используется при высокоуровневом (концептуальном) проектировании баз данных. С её помощью можно выделить ключевые сущности и обозначить связи, которые могут устанавливаться между этими сущностями.

Во время проектирования баз данных происходит преобразование схемы, созданной на основе ER-модели, в конкретную схему базы данных на основе выбранной модели данных (реляционной, объектной, сетевой или др.).

ER-модель представляет собой формальную конструкцию, которая сама по себе не предписывает никаких графических средств её визуализации. В качестве стандартной графической нотации, с помощью которой можно визуализировать ER-модель, была предложена диаграмма «сущность-связь» (англ. Entity-Relationship diagram, ERD, ER-диаграмма).

Понятия «ER-модель» и «ER-диаграмма» часто не различают, хотя для визуализации ER-моделей могут быть использованы и другие графические нотации, либо визуализация может вообще не применяться (например, использоваться текстовое описание).

Выделим отношения данной базы данных, которые образуют структуру базы данных:

* **Проект**. Основные атрибуты проекта — наименование, описание, состояние.
* **Задача**. Основные атрибуты задачи — исполнитель, наименование, описание.
* **Пользователь**. Основные атрибуты пользователя—ФИО, роль.
* **Заметки**. Основные атрибуты заметки —наименование, содержание.
* **Сообщения**. Основные атрибуты сообщения— отправитель, получатель, сообщение.
* **Роль**. Основные атрибуты роли — наименование.

Спроектированная ER-модель, представлена ниже (рисунок 19).

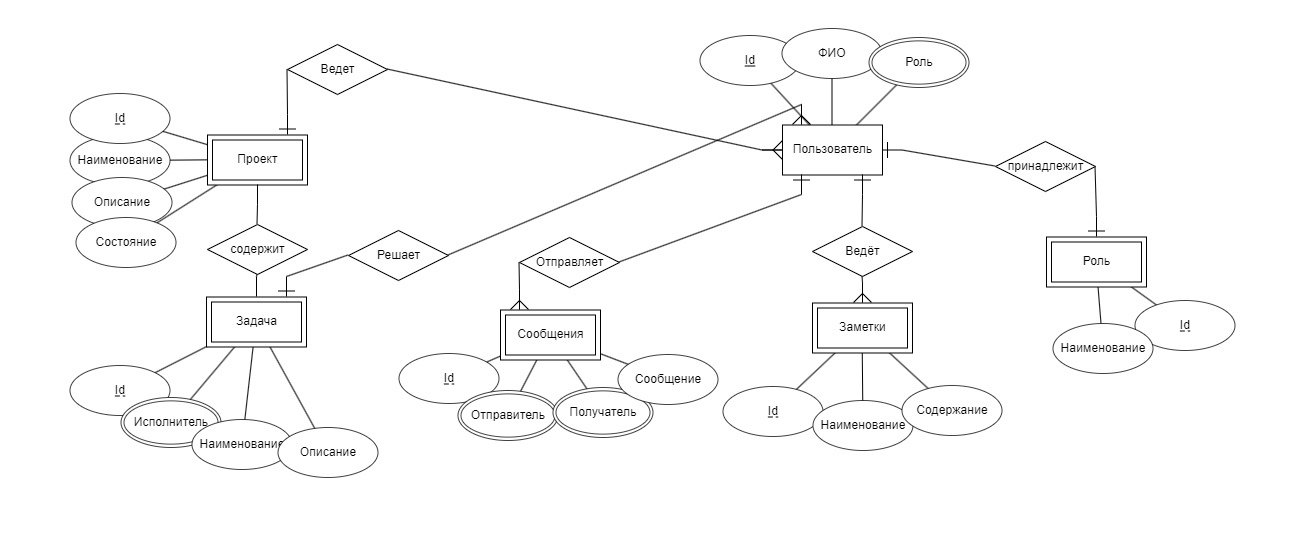


Рисунок 19 –Er-модель.

Физическое проектирование БД — это процесс подготовки описания реализации БД на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Пример физического проектирования БД в среде разработке MSSMS (рисунок 20).

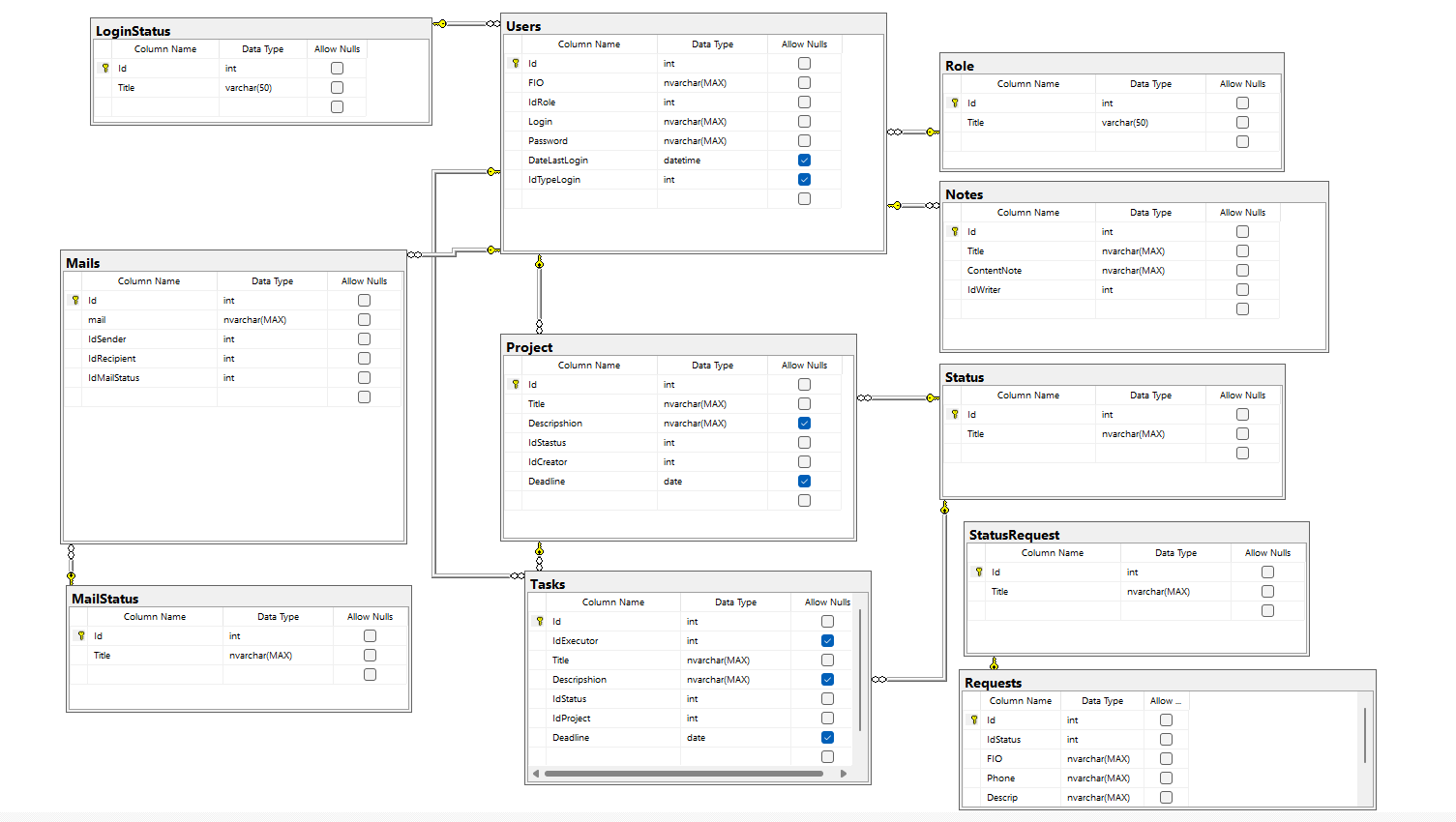


Рисунок 20 – Физическое проектирование.

# **2.2. Программирование программного продукта**

Программирование — это процесс создания программ (программного обеспечения). Для этого программисты пишут исходный код на одном из языков программирования.

Для реализации программного продукта был выбран способ объектно-ориентированного программирования.

Объектно-ориентированное программирование, или ООП — это одна из парадигм разработки. Парадигмой называют набор правил и критериев, которые соблюдают разработчики при написании кода. Если представить, что код — это рецепт блюда, то парадигма — то, как рецепт оформлен в кулинарной книге. Парадигма помогает стандартизировать написание кода. Это снижает риск ошибок, ускоряет разработку и делает код более читабельным для других программистов.

Суть понятия объектно-ориентированного программирования в том, что все программы, написанные с применением этой парадигмы, состоят из объектов. Каждый объект — это определённая сущность со своими данными и набором доступных действий.

Для реализации данного продукта был выбран метод каскадного программирования с промежуточным контролем

Каскадная модель с промежуточным контролем («водоворот», или итерационная модель) является модификацией предыдущей модели. Промежуточный контроль осуществляется с помощью циклов обратной связи между этапами. Этапы, как и в модели «водопад» расположены последовательно, но каждый следующий этап имеет связь с предыдущими (рисунок 21).

Каскадная модель жизненного цикла программного обеспечения имеет следующие достоинства:

1. Стабильность требований в течение всего жизненного цикла разработки.
2. На каждой стадии формируется законченный набор документации.
3. Определенность и понятность шагов модели и простота её применения.

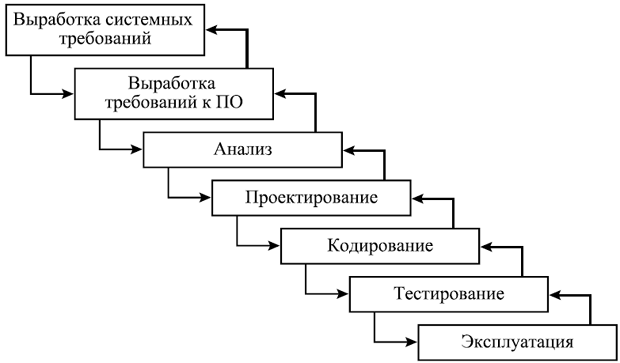


Рисунок 21 – Каскадная модель с промежуточным контролем.

Такая модель жизненного цикла ПС реализует возможность внесения изменений на предыдущем этапе, если уже был начат следующий. Промежуточный контроль осуществляется путем исправления ошибок на каждом из этапов непосредственно в момент выявления проблемы.

Преимуществом каскадной модели с промежуточным контролем благодаря промежуточным корректировкам является снижение риска получения некачественного продукта и повышение надежности системы. Промежуточные корректировки обеспечивают возможность вернуться на один или несколько уровней назад до внесения изменений. Однако здесь проявляются и недостатки: время жизни каждого из этапов равно периоду разработки, само время, отведенное на разработку, может быть увеличено, следом возрастают и затраты на реализацию проекта, а согласование результатов происходит после внедрения, что может повлечь за собой моральное устаревание системы и низкую востребованность.

Диаграмма классов — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов (полей), методов, интерфейсов и взаимосвязей (отношений) между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

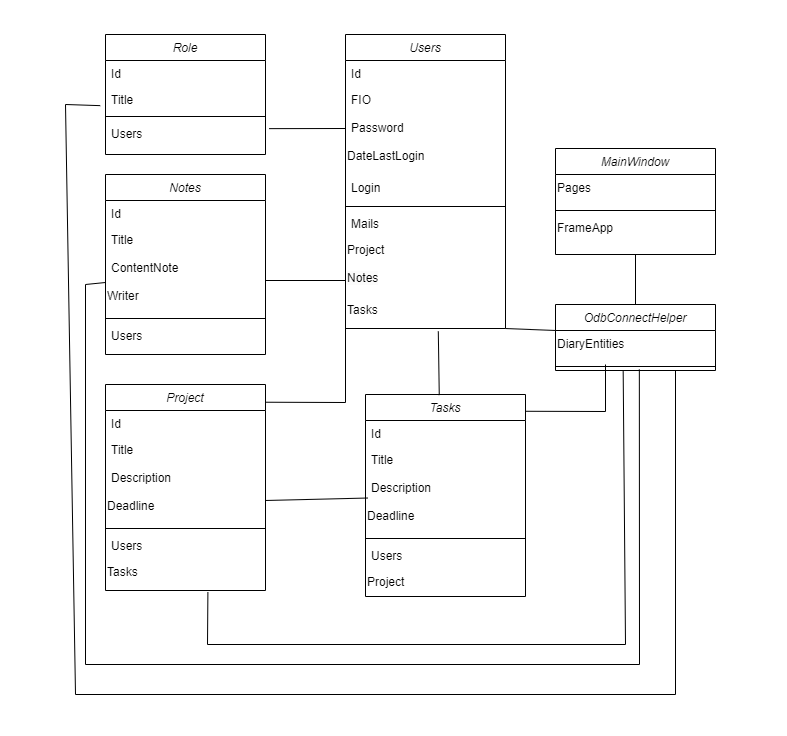
Для данной программы была разработана диаграмма классов (рисунок 22). 

Рисунок 22 – диаграмма классов.

Представленный программный продукт был разработан на платформе WPF (Windows Presentation Foundation) – это система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. Графическая подсистема в составе .NET Framework, использующая язык XAML.

XAML – это язык разметки на основе XML, который позволяет определить внешний вид приложения в декларативной форме. Обычно он используется для создания окон, страниц и пользовательских элементов управления, а также их заполнения элементами управления, фигурами и графическими элементами.

Код программы написан на языке С#. Данный язык программирования имеет множество достоинств и является полностью объектно-ориентированным языком. Прост и надёжен в использовании.

Проектирование и разработка базы данных была выполнена в MS SQL Server. SQL – декларативный язык программирования, применяемый для создания и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Основные методы, классы и функции программного продукта:

1. Метод, организовывающий поиск, сортировку и фильтрацию объектов в списке (рисунок 23).

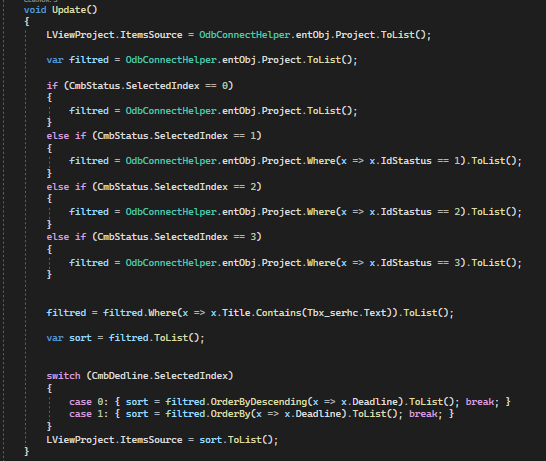


Рисунок 23 – Метод, организовывающий поиск, сортировку и фильтрацию объектов.

1. Класс для подключения к базе данных.

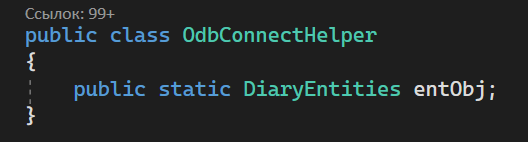


Рисунок 24 – Класс для подключения к базе данных.

1. Класс для организации перехода по страницам (рисунок 25).

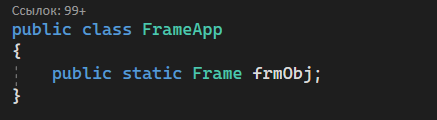
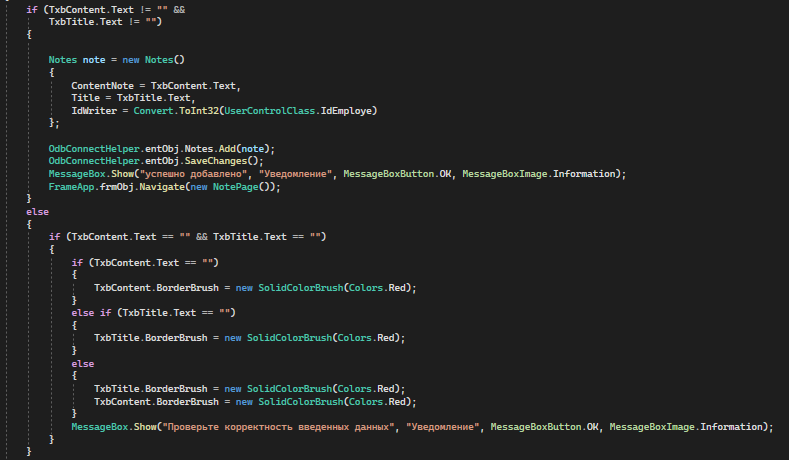


Рисунок 25 – Класс для организации перехода по страницам.

1. Функция добавления информации в базу данных (рисунок 26).

 Рисунок 26 – Функция добавления информации в базу данных.

1. Функция удаления информации из базы данных (рисунок 27).

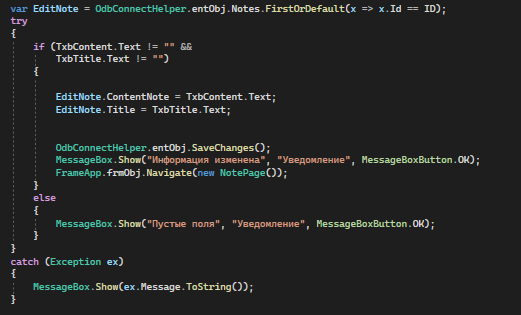


Рисунок 27 – Функция удаления информации из базы данных

В дальнейшем все функции, методы и классы были определены аналогичным образом. Следует продолжать разработку программного продукта, придерживаясь уже существующего стиля и структуры кода. Такой подход поможет обеспечить единообразие, улучшить читаемость и обслуживаемость кода.

# **2.3. Тестирование и отладка программного продукта**

Тестирование программного обеспечения — процесс испытания программного продукта с целью проверки соответствия между реальным поведением программы и её ожидаемым поведением. Проверяется это соотношение на конечном наборе тестов, выбранных определённым образом.

Виды тестирования ПО, в зависимости от преследуемых целей:

1. Функциональные.
2. Нефункциональные.
3. Связанные с изменениями.

Функциональные тесты базируются на функциях и особенностях, а также взаимодействии с другими системами, и могут быть представлены на всех уровнях тестирования: компонентном или модульном (Component/Unit testing), интеграционном (Integration testing), системном (System testing) и приемочном (Acceptance testing). Функциональные виды тестирования рассматривают внешнее поведение системы.

Нефункциональное тестирование описывает тесты, необходимые для определения характеристик программного обеспечения, которые могут быть измерены различными величинами. В целом, это тестирование правильности работы системы.

Связанные с проведения необходимых изменений, таких как исправление бага/дефекта, программное обеспечение должно быть пере тестировано для подтверждения того факта, что проблема была действительно решена.

Основные виды тестирования:

1. Методом черного ящика – это функциональное и нефункциональное тестирование без доступа к внутренней структуре компонентов системы. Метод тестирования «черного ящика» – процедура получения и выбора тестовых случаев на основе анализа спецификации (функциональной или нефункциональной), компонентов или системы без ссылки на их внутреннее устройство.
2. Методом белого ящика – это метод тестирования программного обеспечения, который предполагает, что внутренняя структура, устройство и реализация системы известны тестировщику. Традиционно тестирование белого ящика выполняется на уровне модулей, однако, используется для тестирования интеграции систем и системного тестирования, тестирования внутри устройства и путей между устройствами.
3. Методом серого ящика – это метод тестирования программного продукта или приложения с частичным знанием его внутреннего устройства. Для выполнения тестирования «серого ящика» нет необходимости в доступе тестировщика к исходному коду. Тесты пишутся на основе знания алгоритма, архитектуры, внутренних состояний.

Тестирование проводилось в таблицах (таблица 1-10).

Таблица 1 - Test case №1: Авторизация под ролью «сотрудник».

|  |  |
| --- | --- |
| **Test Case #** | Тест\_1 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Авторизация под ролью «сотрудник». |
| **Резюме испытания** | Выявление ошибок при попытке авторизации. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «1» и «1» соответственно.  Шаг 3 – Нажатие кнопки «Войти». |
| **Данные тестирования** | Данные таблицы «User». |
| **Ожидаемый результат** | Открытие страницы со свободными задачами. |
| **Фактический результат** | Открытие страницы со свободными задачами. |
| **Предпосылки** | Запустить программу осуществить вход. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Ошибок не выявлено. |

Таблица 2 - Test case №2: Подача заявления.

|  |  |
| --- | --- |
| **Test Case #** | Тест\_2 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Подача заявления. |
| **Резюме испытания** | Выявление ошибок при подаче заявления. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 – нажатие кнопки «Подать заявку».  Шаг 3 – заполнить поле «ФИО» данными: «Алексей Петров»  Шаг 4 – заполнить поле «Телефон» данными: «89245645858»  Шаг 5 – заполнить поле «Описание» данными: «Учёт ПК»  Шаг 6 – нажатие кнопки «Подать заявление». |
| **Данные тестирования** | Функция добавления новой заявки. |
| **Ожидаемый результат** | Появление уведомления о успешности операции, добавления заявления в базу данных. |
| **Фактический результат** | Появление уведомления о успешности операции, добавления заявления в базу данных. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Таблица 3 - Test case №3: Авторизация под ролью «ТимЛид».

|  |  |
| --- | --- |
| **Test Case #** | Тест\_1 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Авторизация под ролью «ТимЛид». |
| **Резюме испытания** | Выявление ошибок при попытке авторизации. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти». |
| **Данные тестирования** | Данные таблицы «User». |
| **Ожидаемый результат** | Открытие страницы с проектами. |
| **Фактический результат** | Открытие страницы с проектами. |

Продолжение таблицы 3 - Test case №3: Авторизация под ролью «ТимЛид».

|  |  |
| --- | --- |
| **Предпосылки** | Запустить программу осуществить вход. |
| **Постусловия** | Стабильная работа программы |
| **Статус (Pass/Fail)** | Pass |
| **Комментарии** | Ошибок не выявлено. |

|  |  |
| --- | --- |
| **Test Case #** | Тест\_4 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Создание проекта. |
| **Резюме испытания** | Выявление ошибок при создании проекта. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Нажатие кнопки «Создать проект».  Шаг 5 – заполнить поле «Наименование» данными: «ItService»  Шаг 6 – заполнить поле «Описание» данными: «Учёт ПК»  Шаг 7 – Выбрать дату в поле «Дедлайн» «20.07.2024».  Шаг 8 – нажатие кнопки «Добавить». |
| **Данные тестирования** | Функция добавления нового проекта. |
| **Ожидаемый результат** | Появление уведомления о успешности операции, добавления проекта в базу данных. |
| **Фактический результат** | Появление уведомления о успешности операции, добавления проекта в базу данных. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Таблица 4 - Test case №4: Создание проекта.

Таблица 5 - Test case №5: Написание сообщения.

|  |  |
| --- | --- |
| **Test Case #** | Тест\_2 |
| **Приоритет теста** | Высокий. |

Продолжение таблицы 5 - Test case №5: Написание сообщения.

|  |  |
| --- | --- |
| **Название тестирования/Имя** | Написание сообщения. |
| **Резюме испытания** | Выявление ошибок при написании сообщения. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Нажатие кнопки «Почта».  Шаг 5 – Нажатие кнопки «Написать».  Шаг 6 – Выбрать Адресанта в поле «Адресант» «Левченко Андрей Александрович».  Шаг 7 – заполнить поле «Содержание» данными: «Привет, ты сделал вчерашнюю задачу».  Шаг 8 – нажатие кнопки «Отправить». |
| **Данные тестирования** | Функция отправки сообщения. |
| **Ожидаемый результат** | Появление уведомления о успешности операции, добавления сообщения в базу данных. |
| **Фактический результат** | Появление уведомления о успешности операции, добавления сообщения в базу данных. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Таблица 6 - Test case №6: Редактирование проекта.

|  |  |
| --- | --- |
| **Test Case #** | Тест\_6 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Редактирование проекта. |
| **Резюме испытания** | Выявление ошибок при редактировании проекта. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Найти проект «Дневник разработчика» в списке проектов.  Шаг 5 – Нажатие кнопки «Изменить» в строке с описанием проекта.  Шаг 6 – Изменить поле «Наименование» данными: «ItService»  Шаг 7 – Выбрать дату в поле «Дедлайн» «20.07.2024».  Шаг 8 – нажатие кнопки «Изменить». |

Продолжение таблицы 6 - Test case №6: Редактирование проекта.

|  |  |
| --- | --- |
| **Данные тестирования** | Функция изменения проекта. |
| **Ожидаемый результат** | Появление уведомления о успешности операции, изменение проекта в базе данных. |
| **Фактический результат** | Появление уведомления о успешности операции, изменение проекта в базе данных. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

|  |  |
| --- | --- |
| **Test Case #** | Тест\_7 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Создание новой задачи у проекта. |
| **Резюме испытания** | Выявление ошибок при создании проекта. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Найти проект «Дневник разработчика» в списке проектов.  Шаг 5 – Нажатие кнопки «Изменить.  Шаг 6 – Нажатие кнопки «Добавить задачу».  Шаг 7 – заполнить поле «Наименование» данными: «Заполнить БД»  Шаг 8 – заполнить поле «Описание» данными: «Заполнить БД»  Шаг 9 – Выбрать дату в поле «Дедлайн» «20.07.2024».  Шаг 10 – нажатие кнопки «Добавить». |
| **Данные тестирования** | Функция добавления новой задачи у проекта. |
| **Ожидаемый результат** | Появление уведомления о успешности операции. |
| **Фактический результат** | Появление уведомления о успешности операции. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Таблица 7 - Test case №7: Создание новой задачи у проекта.

Таблица 8 - Test case №8: Ответ на сообщение.

|  |  |
| --- | --- |
| **Test Case #** | Тест\_7 |
| **Приоритет теста** | Средний. |
| **Название тестирования/Имя** | Ответ на сообщение. |
| **Резюме испытания** | Выявление ошибок при ответе на сообщение. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Нажатие кнопки «Почта».  Шаг 5 – Нажатие кнопки «Входящие».  Шаг 6 – Найти сообщение «Привет» в списке сообщений.  Шаг 7 – Нажатие кнопки «Прочесть».  Шаг 8 – нажатие кнопки «Ответить».  Шаг 9 – заполнить поле «Содержание» данными: «Привет»  Шаг 10 – нажатие кнопки «Ответить». |
| **Данные тестирования** | Функция ответа на сообщение. |
| **Ожидаемый результат** | Появление уведомления о успешности операции. |
| **Фактический результат** | Появление уведомления о успешности операции. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Таблица 9 - Test case №9: Удаление проекта.

|  |  |
| --- | --- |
| **Test Case #** | Тест\_9 |
| **Приоритет теста** | Высокий. |
| **Название тестирования/Имя** | Удаление проекта. |
| **Резюме испытания** | Выявление ошибок при далении проекта. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Найти проект «Дневник разработчика» в списке проектов. |

Продолжение таблицы 9 - Test case №9: Удаление проекта.

|  |  |
| --- | --- |
| **Шаги тестирования** | Шаг 5 – Нажатие кнопки «Удалить». |
| **Данные тестирования** | Функция добавления новой задачи у проекта. |
| **Ожидаемый результат** | Появление уведомления о успешности операции. |
| **Фактический результат** | Появление уведомления о успешности операции. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Таблица 10 - Test case №10: Создание заметки.

|  |  |
| --- | --- |
| **Test Case #** | Тест\_10 |
| **Приоритет теста** | Средний. |
| **Название тестирования/Имя** | Создание заметки. |
| **Резюме испытания** | Выявление ошибок при создании замиетки. |
| **Шаги тестирования** | Шаг 1 – Запуск программы.  Шаг 2 –Заполнение полей Логин и пароль данными «2» и «2» соответственно.  Шаг 3 – Нажатие кнопки «Войти».  Шаг 4 – Нажатие кнопки «Заметки».  Шаг 5 – Нажатие кнопки «Создать заметку».  Шаг 6 – заполнить поле «Наименование» данными: «Закончить проект 2».  Шаг 7 – заполнить поле «Содержание» данными: «Закончить проект 2 скорее».  Шаг 8 – Нажатие кнопки «Создать». |
| **Данные тестирования** | Функция создания заметки. |
| **Ожидаемый результат** | Появление уведомления о успешности операции. |
| **Фактический результат** | Появление уведомления о успешности операции. |
| **Предпосылки** | Запустить программу. |
| **Постусловия** | Стабильная работа программы. |
| **Статус (Pass/Fail)** | Pass. |
| **Комментарии** | Ошибок не выявлено. |

Отладка – это процесс поиска и исправления ошибок или неполадок в исходном коде какого-либо программного обеспечения. Когда программное обеспечение не работает, как ожидалось, компьютерные программисты изучают код, чтобы выяснить причину появления ошибок. Они используют инструменты отладки для запуска программного обеспечения в контролируемой среде, пошаговой проверки кода, а также анализа и поиска проблем.

**Цели отладки в разработке программного обеспечения включают:**

1. **Устранение ошибок:** основная и самая очевидная цель отладки – исправление обнаруженных в программе ошибок. Это может касаться как простых синтаксических ошибок, так и более сложных проблем логики или дизайна программы.
2. **Повышение стабильности и надежности:** удаление найденных багов помогает сделать программное обеспечение более стабильным и надежным в работе. Это, в свою очередь, улучшает пользовательский опыт и способствует большему доверию пользователей к продукту.
3. **Оптимизация производительности:** в процессе отладки можно выявить и устранить не только ошибки, но и узкие места в производительности программы, что приведет к увеличению скорости работы и эффективности использования ресурсов системы.
4. **Повышение качества кода:** анализ и исправление ошибок помогают разработчикам лучше понять код и потенциально пересмотреть или улучшить части программы для предотвращения будущих ошибок.
5. **Обучение разработчиков:** процесс отладки дает разработчикам возможность учиться на своих ошибках, понимать их причины и избегать аналогичных проблем в будущем, что повышает их компетенции и улучшает практики кодирования.
6. **Документирование и предупреждение:** В процессе отладки разработчики могут документировать обнаруженные баги и способы их решения, что становится важной информацией для будущего анализа и помогает предотвратить повторение ошибок.

**Отладка** – это не просто поиск и исправление ошибок, это комплексный процесс, который влияет на многие аспекты разработки программного обеспечения, направленный на улучшение продукта за счет повышения его качества, надежности, стабильности и удобства использования. Процесс отладки программного обеспечения может варьироваться в зависимости от сложности проекта, используемых инструментов и предпочтений разработчиков, но в целом он включает **следующие шаги**:

1. Обнаружение ошибки: Отладка начинается с обнаружения ошибки, которое может произойти через автоматическое тестирование, отчеты пользователей или наблюдения разработчиков. Ошибка может проявляться как сбой программы, неправильное поведение или неожиданные результаты.
2. Воспроизведение ошибки: Дальнейшим шагом является попытка воспроизвести ошибку в контролируемой среде, чтобы точно определить условия, при которых она возникает. Это включает в себя сбор информации о конфигурации системы, входных данных, настройках пользователя и других переменных.
3. Локализация проблемы: после того как удалось воспроизвести ситуацию, вызывающую ошибку, следующий шаг – локализация проблемы в исходном коде. Для этого могут использоваться различные инструменты отладки, такие как отладчики, которые позволяют выполнять код пошагово, просматривать состояние переменных и изменение выполнения программы в реальном времени.
4. Анализ причин возникновения ошибки: определив местоположение ошибки в коде, разработчики анализируют логику работы программы, чтобы понять причину ошибки. Это может включать в себя анализ потоков данных, условий выполнения и взаимодействий между компонентами системы.
5. Исправление ошибки: после определения причины ошибки разработчики вносят изменения в код для ее устранения. Это может потребовать простого исправления.
6. Тестирование исправления: исправленный код должен быть протестирован, чтобы убедиться, что ошибка была устранена. Тестирование обычно включает повторное воспроизведение исходной ошибки.
7. Документирование: важным шагом является документирование ошибки, процесса ее обнаружения, анализа и устранения. Это помогает в будущем предотвратить повторение аналогичных ошибок и служит ценным ресурсом для команды разработчиков.
8. Подтверждение исправлений пользователями или тестировщиками: после внутреннего тестирования решения ошибки, важно получить подтверждение от пользователей или профессиональных тестировщиков, что проблема действительно устранена в их среде и при их сценариях использования.

Эти шаги не обязательно следуют друг за другом строго последовательно; в зависимости от ситуации разработчики могут вернуться на один или несколько шагов назад, если в процессе отладки обнаруживаются новые подробности или возникают дополнительные проблемы.

Отладка проводилось в таблице (таблица 11).

Таблица 11 - Отладка.

|  |  |  |
| --- | --- | --- |
| **Ошибка** | **Пояснение** | **Решение** |
|  | Не верно указана логическая функция вместо «или» должно быть «и». |  |
|  | Переменная не должна являться массивом, нужно более точно определить переменую. |  |
|  | Чтоб не выводит все задачи сразу нужно более точно прописать условия выборки. |  |
|  | В данном случаи нужно использовать метод Contains. |  |

Продолжение таблицы 11 - Отладка.

|  |  |  |
| --- | --- | --- |
| **Ошибка** | **Пояснение** | **Решение** |
|  | Неправильно инициализирована переменная класса FrameApp. |  |
|  | Не верный порядок параметров у MessageBox. |  |
|  | Не обходимо прерывание после case. |  |

Отладка и тестирование – это взаимодополняющие процессы, которые гарантируют, что программы работают должным образом. После написания полного раздела или части кода программисты выполняют тестирование, чтобы обнаружить недочеты и ошибки. Как только они обнаружены, кодировщики могут начать процесс отладки и работать над устранением ошибок в программном обеспечении.

# **2.4. Документирование**

Руководство пользователя, предназначено для оказания помощи пользователям в использовании определенного продукта, услуги или приложения. Обычно его пишет технический специалист, разработчик продукта или сотрудник службы поддержки клиентов компании.

Большинство руководств пользователя содержат как письменное руководство, так и связанные с ним изображения. В случае компьютерных приложений обычно включаются скриншоты UI интерфейса, а руководства по оборудованию часто включают четкие, упрощенные диаграммы.

Руководство пользователя АИС «Дневник разработчика»

При запуске приложения пользователь увидит первую страницу (рисунок 23). На ней представлена функция авторизации и переход на страницу подачи заявления.

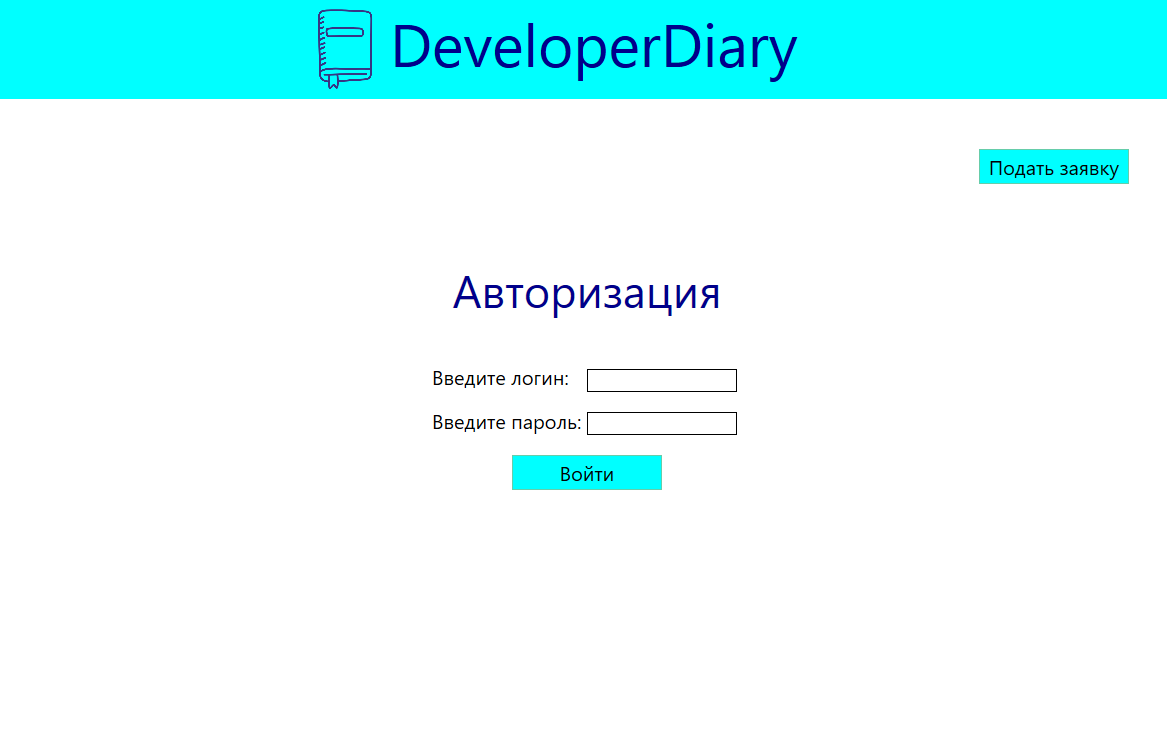


Рисунок 23 – Окно авторизации.

Инструкция для подачи заявлений.

Для подачи заявления пользователь должен нажать на кнопку «Подать заявку», которая находится в правом верхнем углу программы. После чего пользователя перебросит на страницу подачи заявлений (рисунок 28), на данной странице находятся поля для ввода личных данных (ФИО и телефон), а также поля для описания проблемы или же проекта, кнопка додачи заявления и кнопка возврата на страницу авторизации. Далее пользователь должен заполнить все поля и нажать на кнопку «Подать заявление». После нажатия кнопки, если все поля были заполнены появится уведомление о успешной подаче заявления и программа автоматически перебросит пользователя на страницу авторизации.

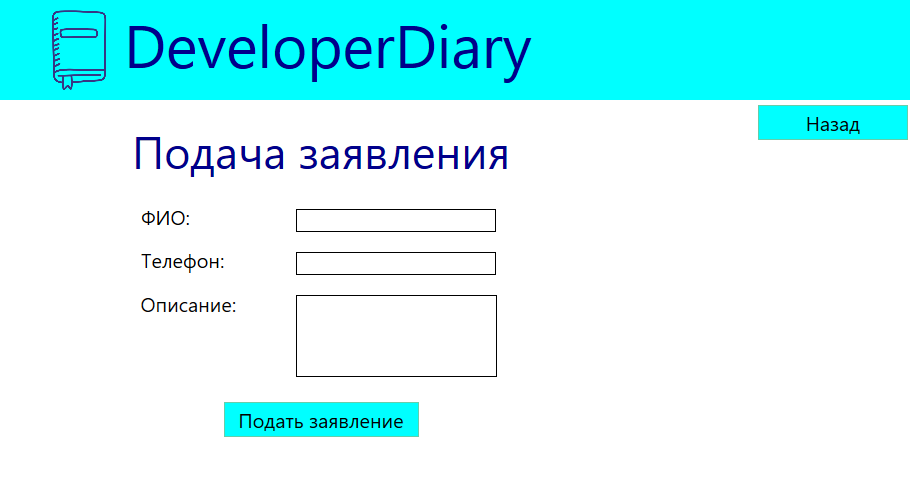


Рисунок 28 – Окно подачи заявления.

Инструкция пользователя с уровнем допуска «Сотрудник».

После успешной авторизации пользователя с ролью «Сотрудник» программа перекинет пользователя на страницу свободных задач (рисунок 29).

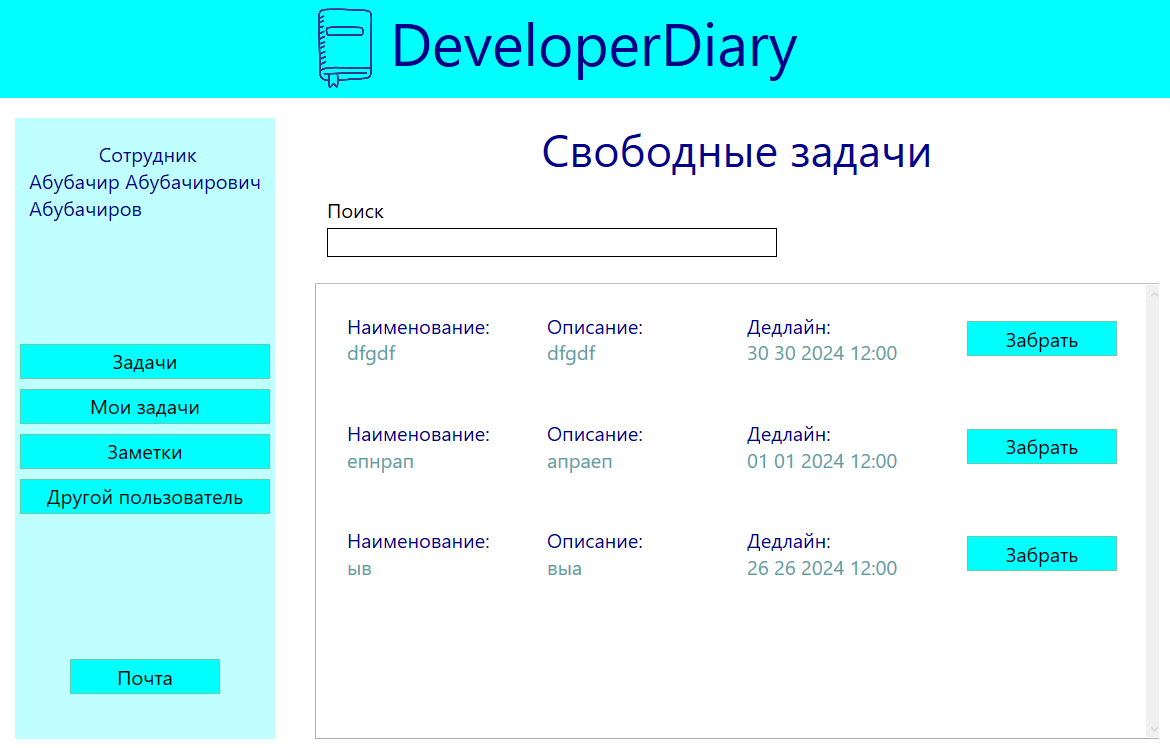


Рисунок 29 – страница свободных задач.

На ней находятся навигационная панель для перехода по страницам программы, надпись, которая будет содержать роль и ФИО авторизованного пользователя, функция поиска по наименованию задачи, а также сам список всех свободных задач, которые никто не выполняет. На данной странице пользователь может забрать любую, доступную из списка задачу, нажав на кнопку «Забрать» рядом с самой задачей. После чего выведется уведомление о выполнении данной операции, и задача удалится из данного списка также переместится в список задач пользователя.

При нажатии на кнопку «Мои задачи», которая находится на навигационной панели, пользователь переместится на страницу «Мои задачи» (рисунок 30).

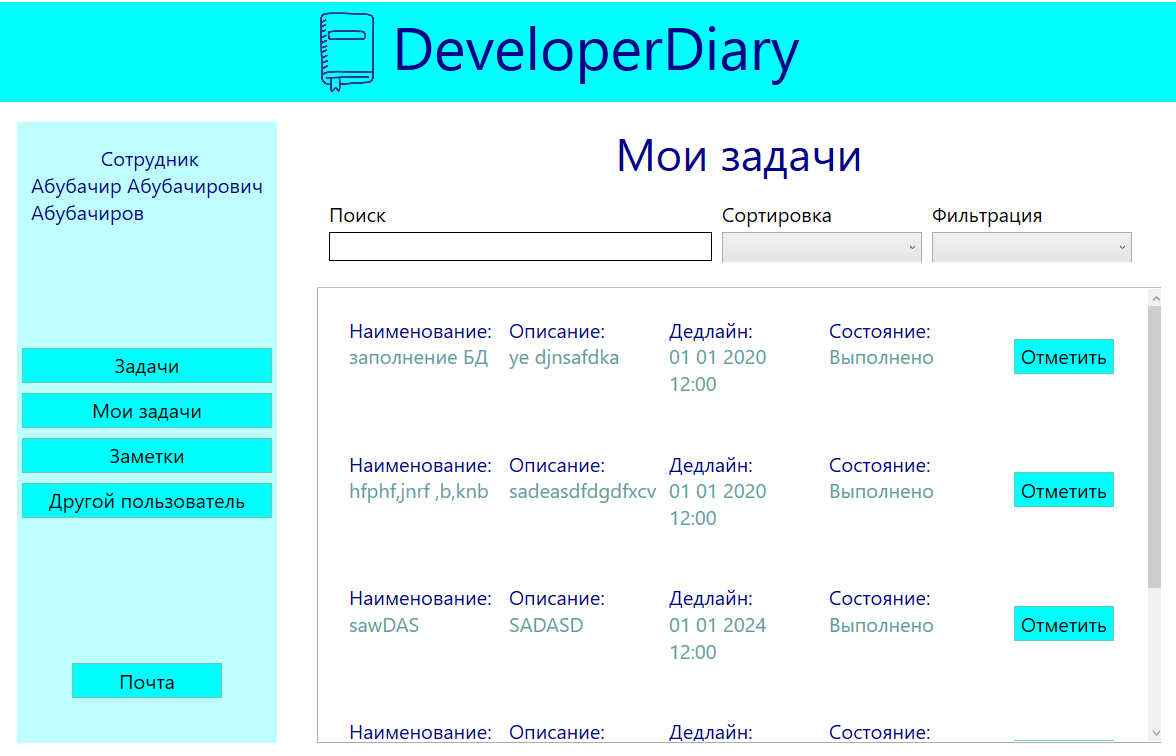


Рисунок 30 – страница «Мои задачи».

Здесь также реализована функция поиска по наименованию задачи, однако на данной странице присутствует сортировка по дедлайну и фильтрация по статусу выполнения. В данном списке отображаются задачи, которые пользователь взял на странице свободных задач. При выполнении задачи пользователь может отметить задачу, нажав на кнопку «Отметить» рядом с задачей, и после уведомления об успешности операции задача поменяет своё состояние.

При нажатии на кнопку «Заметки», которая также находится на навигационной панели, пользователя переместит на страницу заметок (рисунок 31).

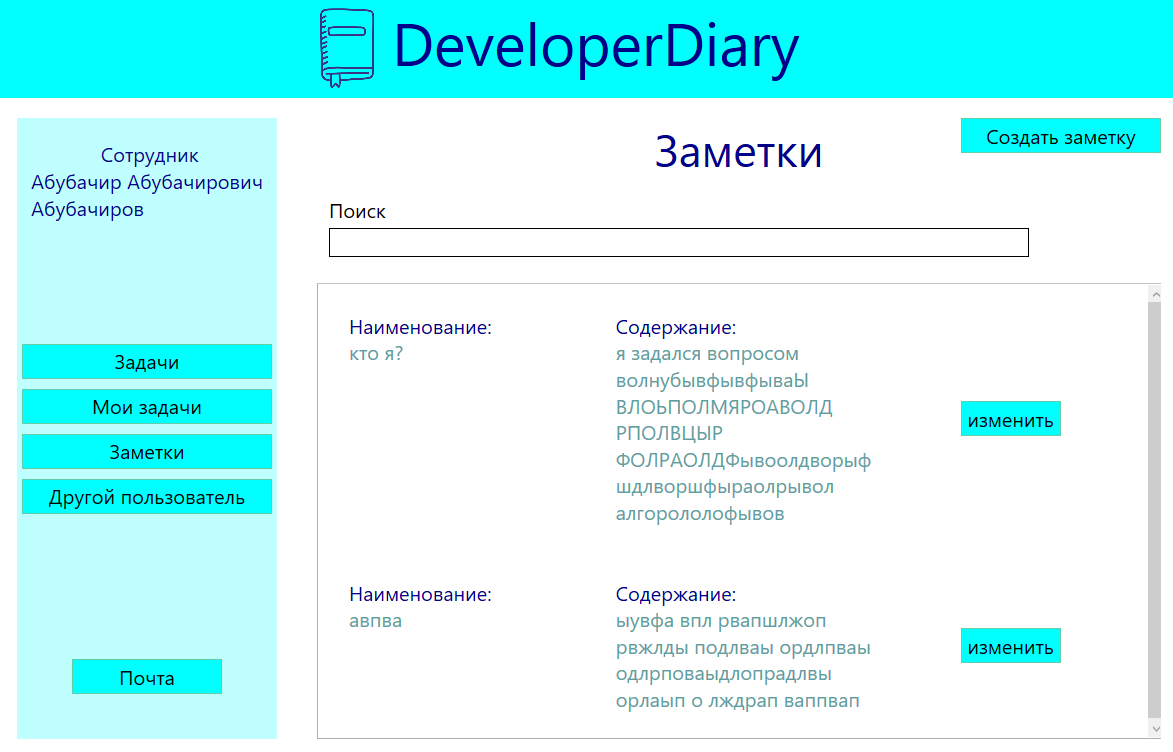


Рисунок 31 – страница заметок.

На этой странице также присутствует навигационная панель, функция поиска и список, однако теперь это список личных заметок пользователя.

Чтобы добавить новую заметку пользователь должен нажать на кнопку «Создать заметку». После чего его перебросит на страницу создание новой заметки (рисунок 32), на которой будет представлены поля для заполнения данных (наименование, содержание). После заполнение всех полей пользователь должен нажать на кнопку «Создать», при нажатии на кнопку появится уведомление о добавлении заметки и пользователя перекинет на страницу заметок.

Для изменения или удаления заметки пользователь должен нажать на кнопку «Изменить» рядом с заметкой. После чего пользователя перекинет на страницу редактирования заметки (рисунок 33). На данной странице будут те же поля для заполнения что и на странице создания заметок, однако они будут уже заполнены. Для изменения данных нужно редактировать поля и нажать на кнопку «Изменить», после чего данные обновятся, а для удаления кнопку «Удалить», после чего данные удалятся и пользователя перекинет на страницу заметок.

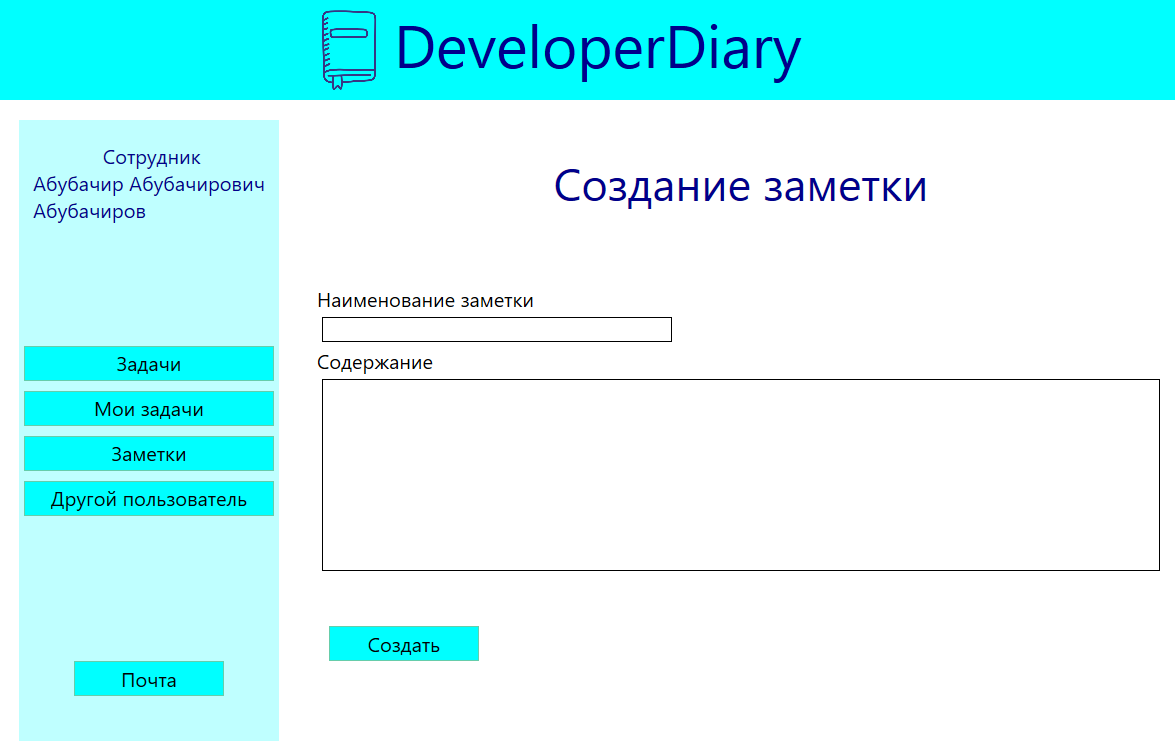


Рисунок 32 – страница создания заметок.

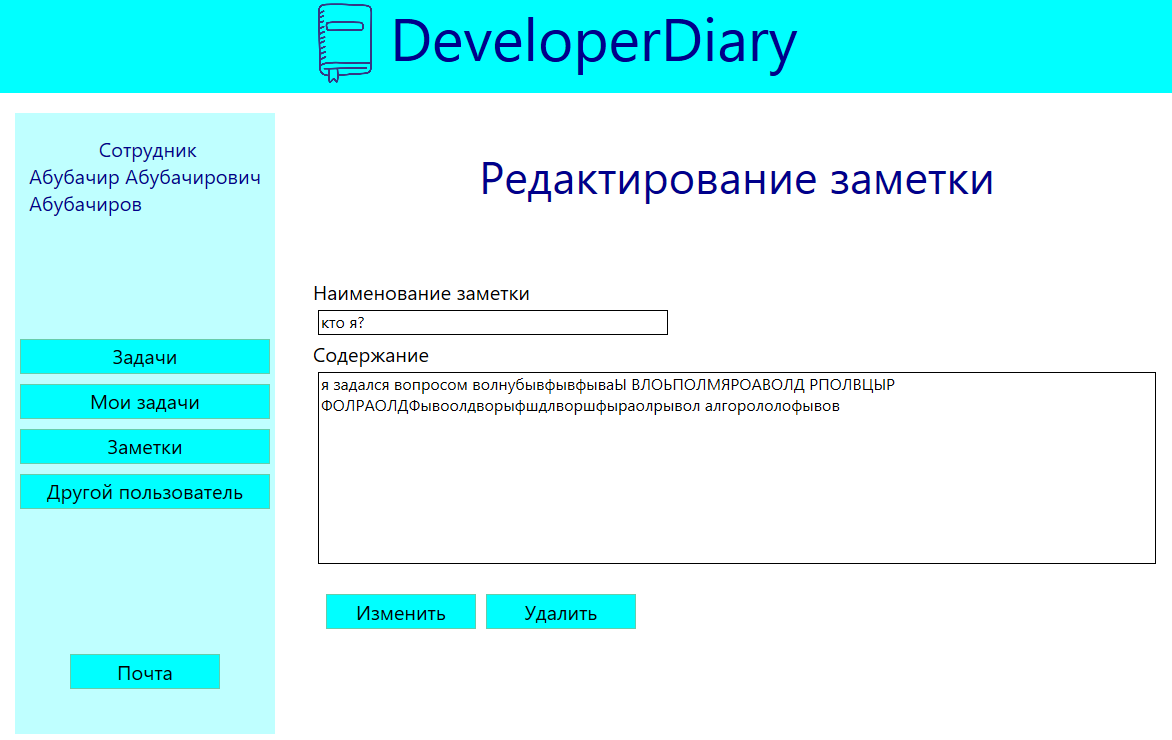


Рисунок 33 – страница редактирования заметок.

Инструкция пользователя с уровнем допуска «ТимЛид».

После успешной авторизации пользователя с ролью «ТимЛид» программа перекинет пользователя на страницу проектов (рисунок 34).

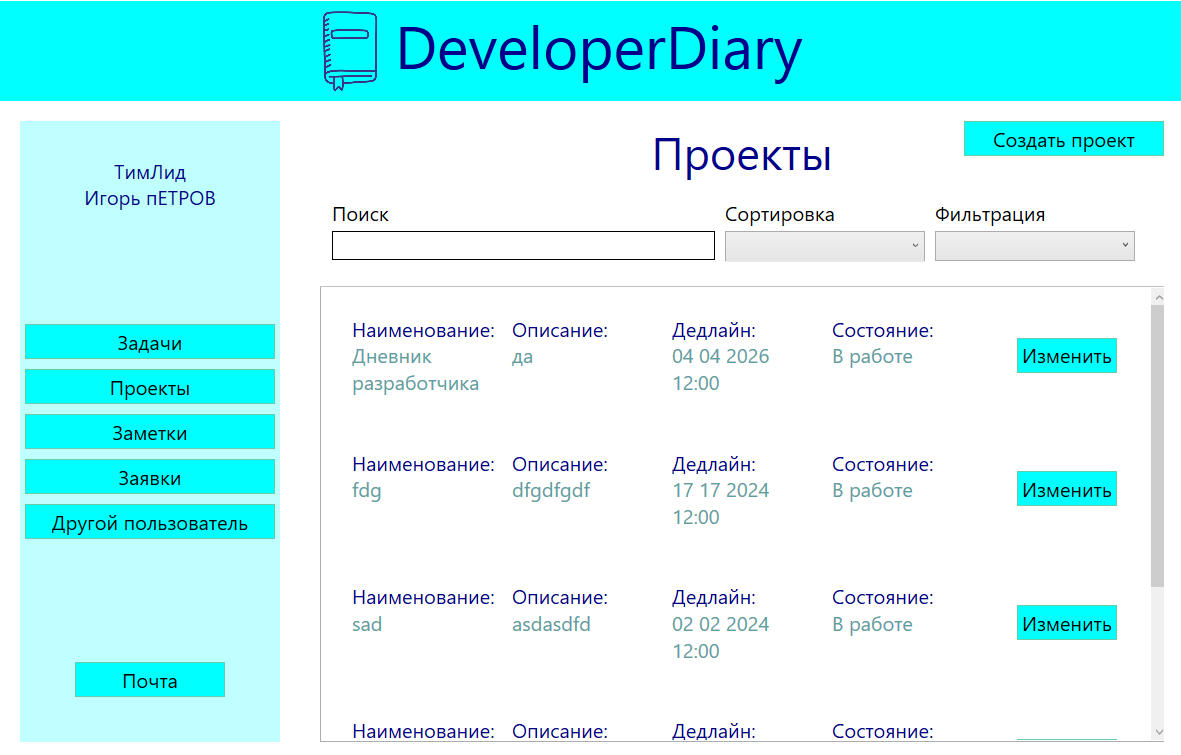


Рисунок 34 – страница проектов.

Так же, как и у пользователя «Сотрудник» на всех страницах присутствует навигационная панель, однако содержание этой панели отличается.

На странице проектов присутствует поиск по наименованию, сортировка по дедлайну, фильтрация по состоянию проекта и список всех проектов.

Для создания нового проекта необходимо нажать на кнопку «Создать проект», при нажатии на кнопку пользователя перебросит на страницу добавления проекта (рисунок 35), на которой будут поля для ввода (Наименование, описание), выпадающий список для выбора даты, кнопка «Добавить» и Кнопка «Назад», возвращающая пользователя на страницу проектов. После заполнения всех полей и выбора даты пользователь должен нажать на кнопку «Добавить», появится уведомление об успешности операции, добиться новый проект в список проектов и пользователя автоматически перебросит на страницу проектов.

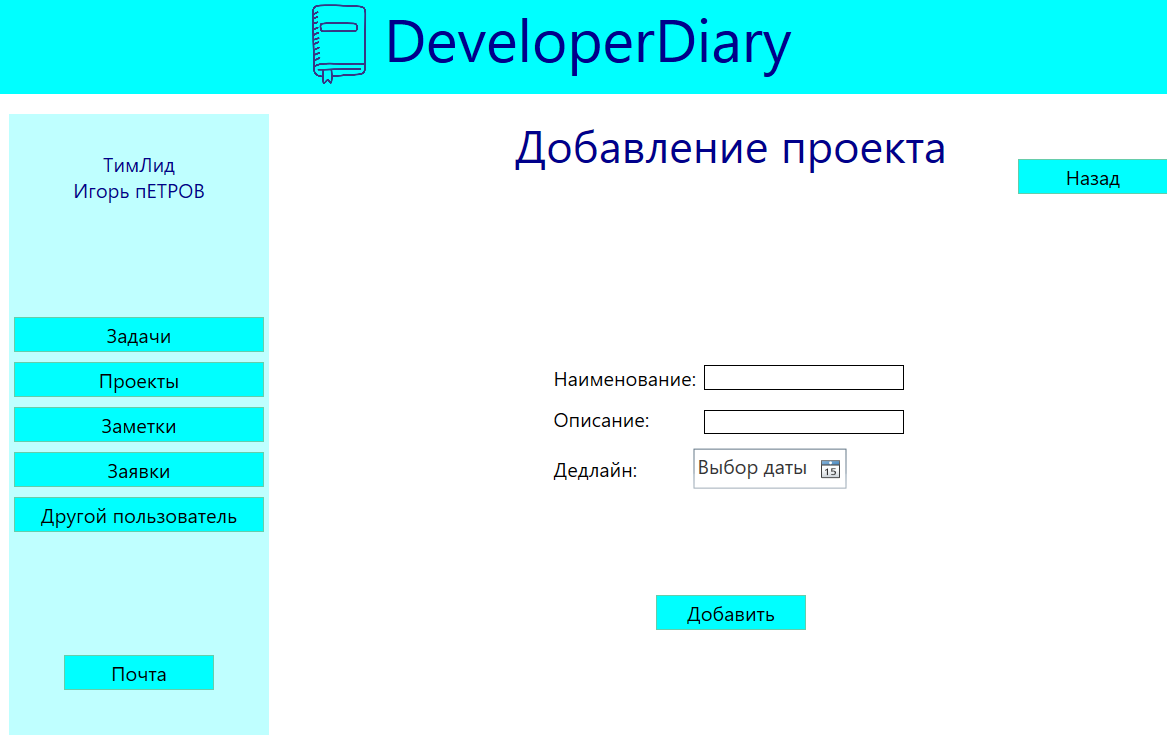


Рисунок 35 – страница создания проектов.

Для изменения или удаления проекта пользователь должен нажать на кнопку «Изменить» рядом с проектом. После чего пользователя перекинет на страницу редактирования проекта (рисунок 36). На данной странице будут те же поля для заполнения что и на странице создания проекта помимо поля состояния и списка задач данного проекта. Для изменения данных нужно редактировать поля и нажать на кнопку «Изменить», после чего данные обновятся, а для удаления кнопку «Удалить проект», после чего данные удалятся и пользователя перекинет на страницу проектов.

Чтобы создать новую задачу для проекта нужно на странице редактирования проекта нажать на кнопку «Добавить задачу», после чего пользователь окажется на странице создания задачи (рисунок 37). На данной странице находятся 2 поля для заполнения данными (Наименование, описание), выпадающий список для выбора даты. После заполнения данных полей и выбора даты, пользователь должен нажать на кнопку добавить, появится уведомление о добавлении

задачи и пользователя перекинет на страницу изменения проекта

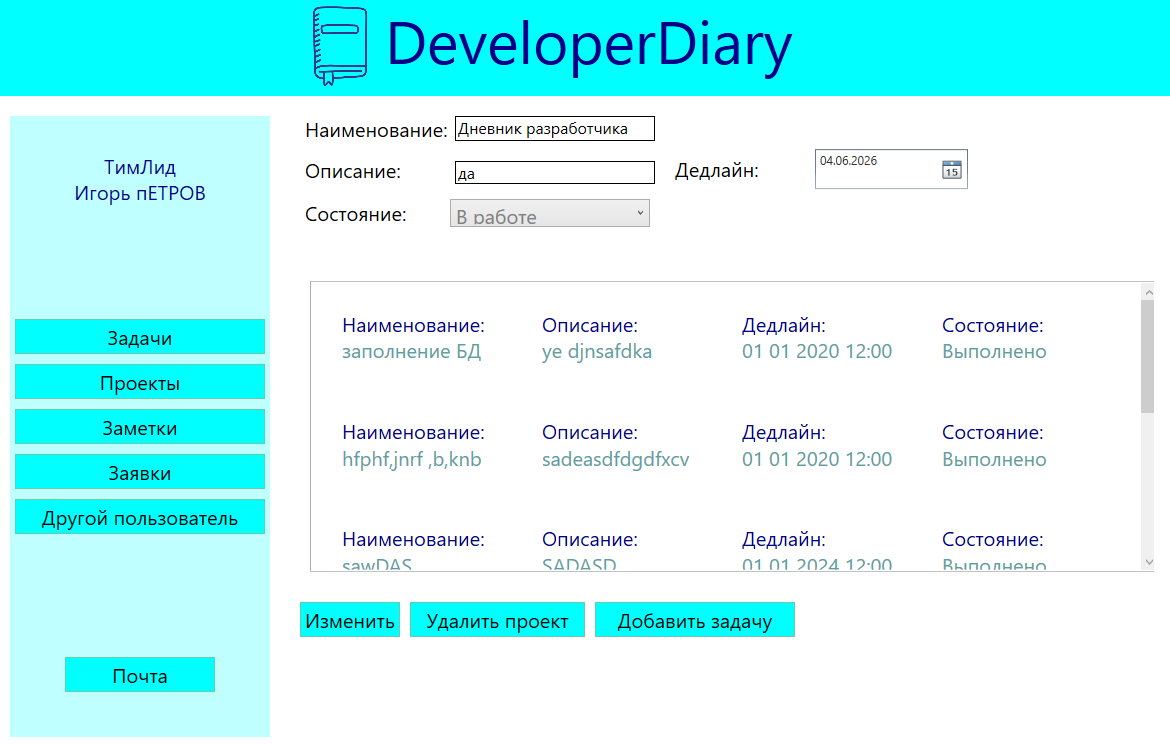


Рисунок 36 – страница редактирования проекта.

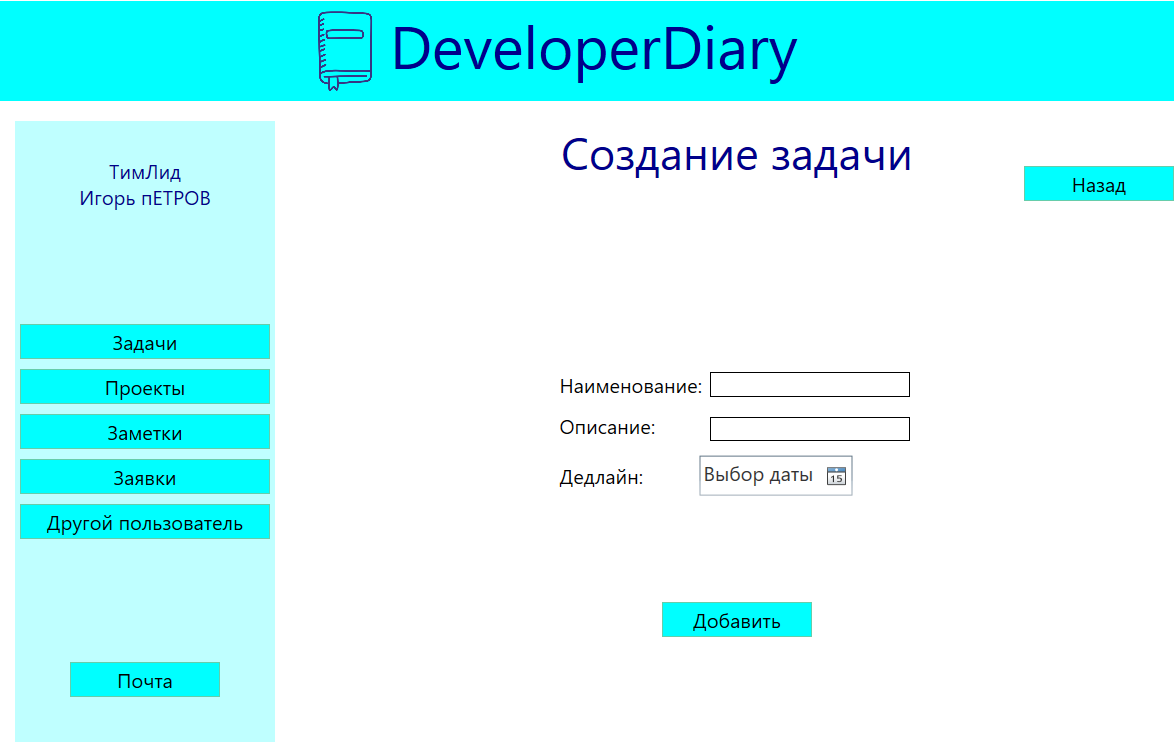


Рисунок 37 – страница создания задачи для проекта.

При нажатии на кнопку «Задачи» пользователя перекинет на страницу задач (рисунок 38).

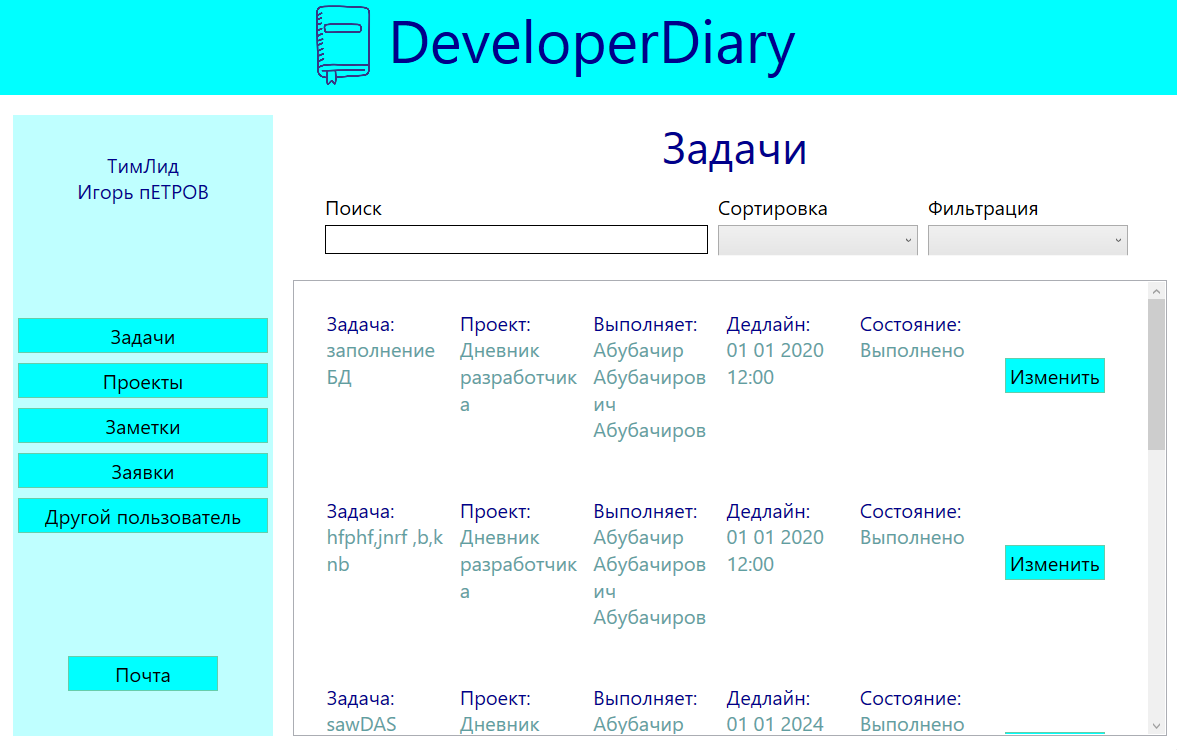


Рисунок 38 – страница создания задачи для проекта.

На странице задач присутствует поиск по наименованию, сортировка по дедлайну, фильтрация по состоянию задачи и список всех задач.

Для изменения или удаления задачи пользователь должен нажать на кнопку «Изменить» рядом с задачей. После чего пользователя перекинет на страницу редактирования задач (рисунок 39). На данной странице будут те же поля для заполнения что и на странице создания задачи помимо поля состояния и выполняющего задачу. Для изменения данных нужно редактировать поля, выбрать выполняющего задачу и нажать на кнопку «Изменить», после чего данные обновятся, а для удаления нужно нажать на кнопку «Удалить», после чего данные удалятся и пользователя перекинет на страницу задач.

Страницы заметок, у пользователя «ТимЛид», выполнены так же, как и у пользователя «Сотрудник».

При нажатии на кнопку «Заявки» пользователя перекинет на страницу заявок (рисунок 40).

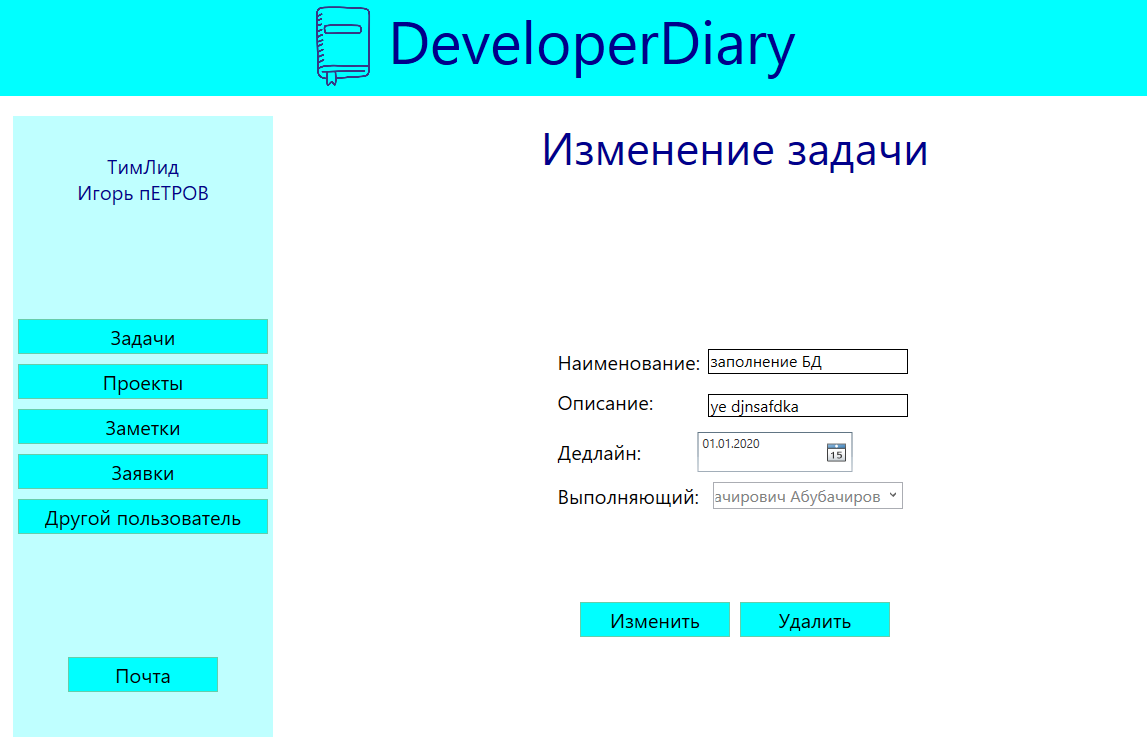


Рисунок 39 – страница изменения задачи.

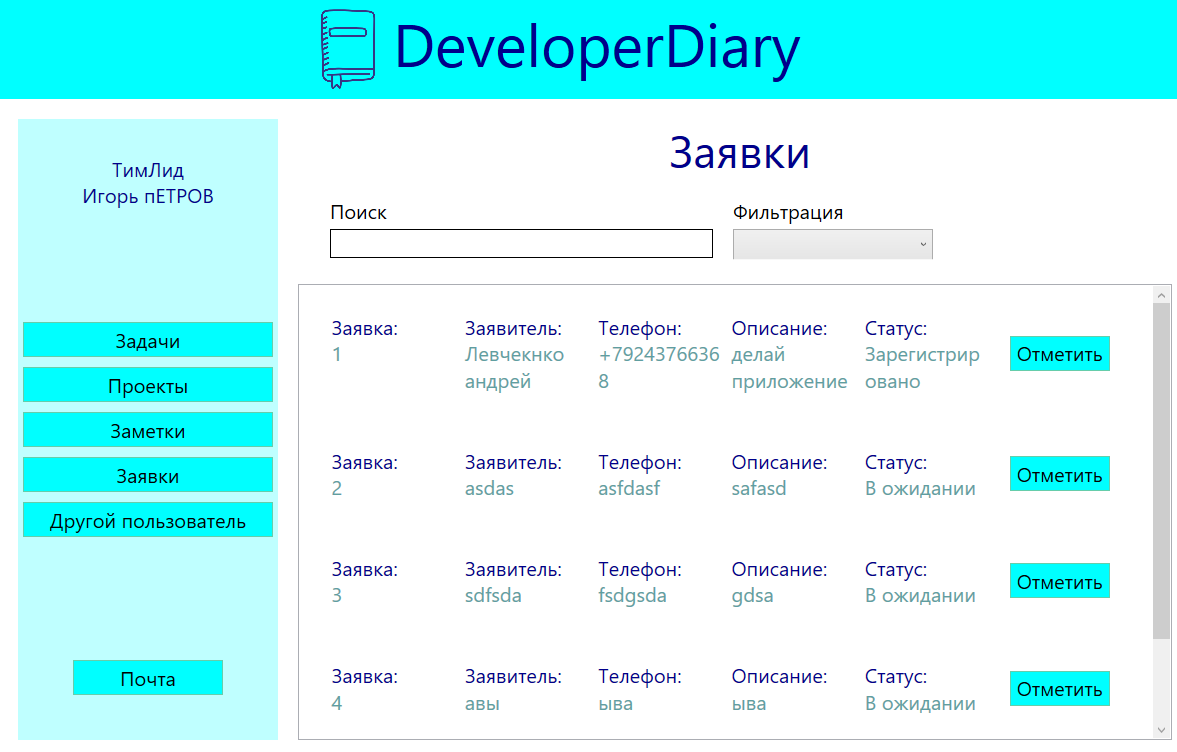


Рисунок 40 – страница заявок.

На странице заявок присутствует поиск по наименованию, фильтрация по состоянию заявок и список всех заявок.

При нажатии на кнопку «Отметить», заявка поменяет свой статус и пользователя перекинет на страницу создания проекта.

Инструкция пользователя по использованию «почты».

Для того чтобы пользоваться почтой пользователь должен авторизироваться и нажать на кнопку «Почта», после чего его перекинет на страницу с входящими сообщениями (рисунок 41).

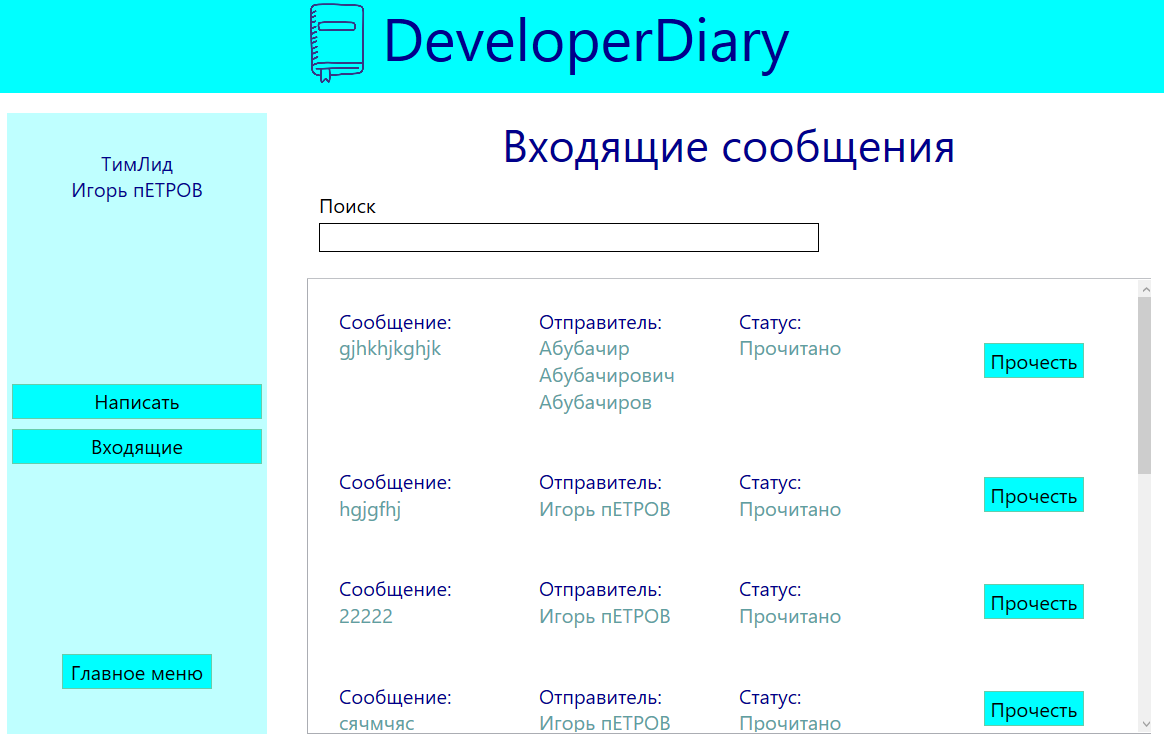


Рисунок 41 – страница входящих сообщений.

На данной странице находится список с входящими сообщениями пользователя, поиск.

Пользователь может прочесть сообщение. Для этого нужно нажать на кнопку «Прочесть», после чего откроется страница прочтения сообщения (рисунок 42), на которой можно увидеть отправителя и само сообщение.

При нажатии на этой странице кнопку «Ответить» открывается страница для ответа на сообщение (рисунок 43) на которой нужно заполнить поле и нажать на кнопку «Ответить». После чего пользователь ответит на сообщение

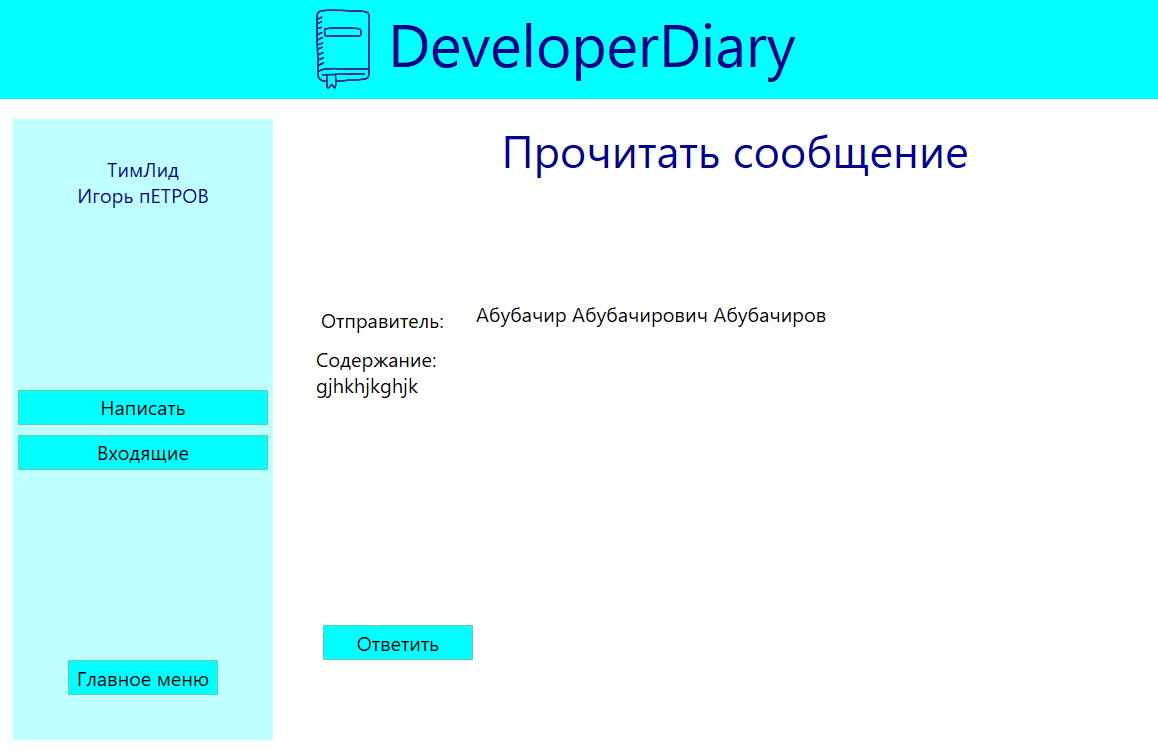


Рисунок 42 – страница просмотра сообщения.

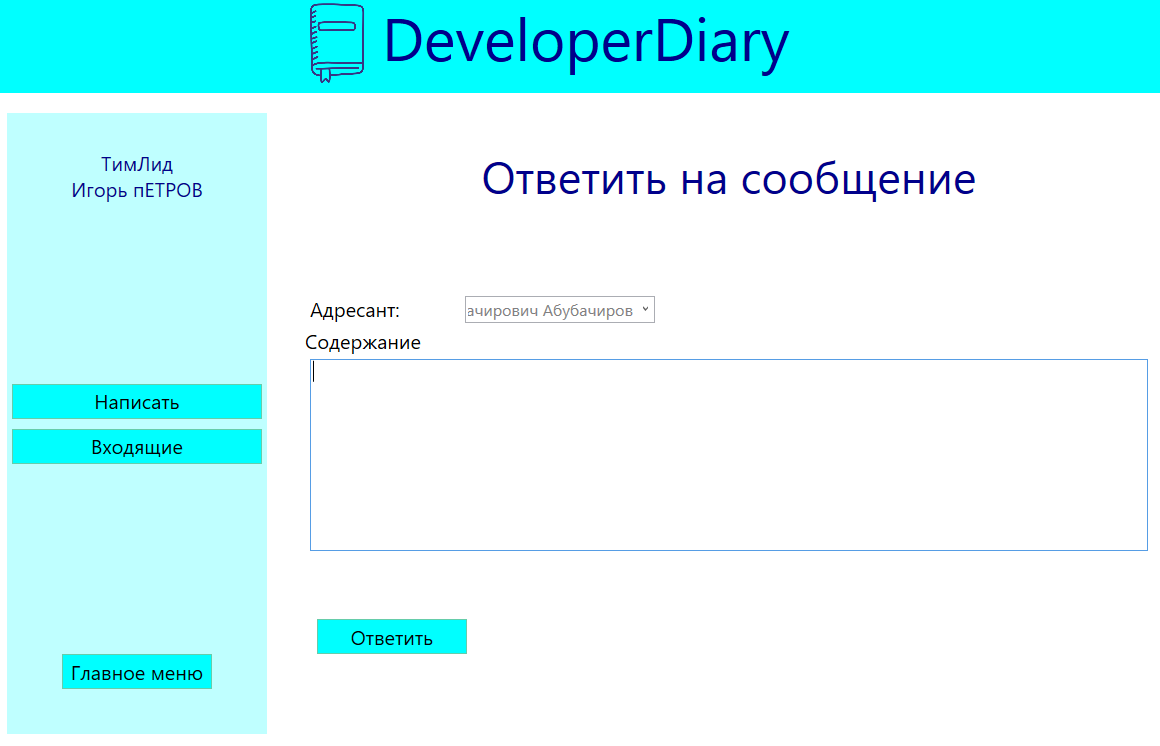


Рисунок 43 – страница ответа на сообщение.

Чтобы написать сообщение пользователь должен нажать на кнопку «Написать», откроется окно Создания сообщения (страница 44), необходимо заполнить все поля и нажать на кнопку «Отправить», после чего пользователь напишет сообщение другому пользователю.

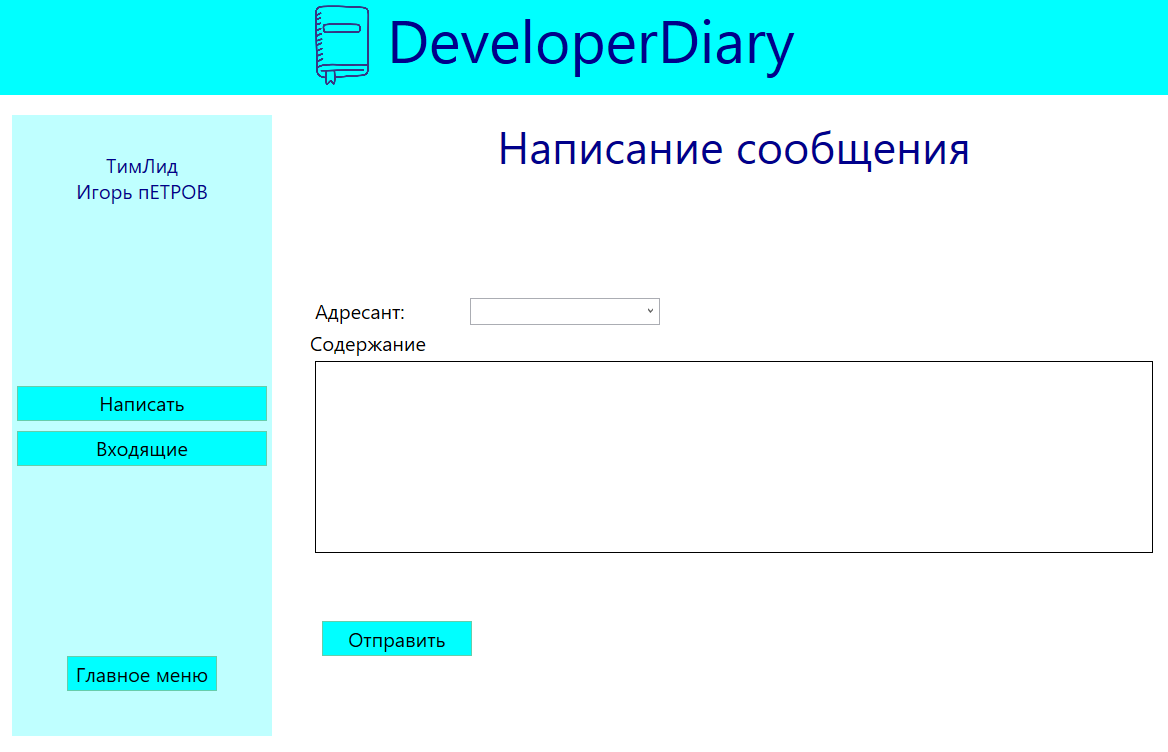


Рисунок 44 – страница создания сообщение.

# **2.5. Эксплуатация и сопровождение программного продукта**

Эксплуатация и сопровождение программного продукта — это процессы, связанные с обеспечением нормальной работы и поддержкой программного продукта после его выпуска и внедрения.

Эксплуатация программного продукта включает в себя следующие основные этапы:

* 1. Установка и настройка программного продукта на компьютерах или серверах пользователя.
  2. Предоставление доступа пользователям к программному продукту с учетом их ролей и прав.
  3. Мониторинг работы программного продукта, обнаружение и устранение ошибок и сбоев.
  4. Резервное копирование данных и обеспечение их безопасности.
  5. Оказание поддержки и консультаций пользователям по вопросам эксплуатации программного продукта.

Сопровождение программного продукта включает в себя следующие основные этапы:

* 1. Разработка и выпуск обновлений программного продукта для исправления ошибок, улучшения функционала или совместимости с новыми технологиями.
  2. Поддержка старых версий программного продукта по запросам от пользователей.
  3. Обновление документации и инструкций по использованию программного продукта.
  4. Обучение пользователей новым функциям и возможностям программного продукта.

Эффективная эксплуатация и сопровождение программного продукта важны для обеспечения его стабильной работы, удовлетворения потребностей пользователей и увеличения продуктивности бизнеса.

Этот программный продукт будет установлен на все компьютеры в организации и подключен к локальному серверу MSSQL. На этом сервере будет храниться база данных, необходимая для работы продукта. Сам MSSQL сервер продукт также будет размещён на локальном сервере предприятия.

Обучением сотрудников будет заниматься руководство предприятия. В этом им поможет руководство пользователя, описанное выше.

# **Охрана труда и техника безопасности при работе на персональном компьютере**

Техника безопасности на начало работы:

1. Осмотреть рабочее место, при необходимости привести его в порядок.
2. Отрегулировать яркость экрана компьютера.
3. Проконтролировать, подключены ли электрические части компьютера к сети и отсутствие оголённых частей проводки.
4. Провести проверку целостности стола, стула, подставки для ног, выдвижной части стола для клавиатуры и т. д.

Требования техники безопасности во время работы:

1. Выполнять работу, за которую вы ответственны.
2. Следить за чистотой своего рабочего места.
3. Не закрывать вентиляционные окна компьютера.
4. Правильно и по назначению эксплуатировать компьютер.
5. Следить за соблюдением своего графика работы и отдыха.
6. Корректно прекращать работу компьютера и приложений.

Памятка что нужно сделать в конце рабочего дня:

1. Правильно завершать работу всех запущенных программ и устройств.
2. Проверить отсутствие в дисководах дисков и USB-флэш-накопителей в USB-портах.
3. Выключить компьютер.
4. Отключить устройства от электросети.
5. Осмотреть рабочее место и привести его в порядок при необходимости.

# **ЗАКЛЮЧЕНИЕ**

В результате дипломного проекта был разработан программный продукт «Дневник разработчика». Оперативность работы повышена за счёт автоматизации процессов по организации работ и распределения задач над проектами.

Были выполнены следующие задачи:

1. Приобретение опыта в разработке программы.
2. Исследование предметной области.
3. Проектирование программного продукта.
4. Разработан программный продукт.
5. Описание методов программирования.
6. Составление документации к проекту.
7. Проведение отладки и тестирования проекта.
8. Спроектирована и разработана БД.

Готовый программный продукт может активно применятся в работе по организации процесса работ и распределения задач среди сотрудников, а также обновляться.

# **СПИСОК ЛИТЕРАТУРЫ**

1. ГОСТ 2.104-2006. Единая система конструкторской документации. Основные надписи. – Москва: Изд-во стандартов, 2007. - 14 с.
2. ГОСТ 2.105-95. Единая система программной документации. Общие требования к текстовым документам. – Минск, 1995. – 28 с.
3. ГОСТ 7.12-93. Единая система конструкторской документации. Сокращение слов в тексте и подписях под иллюстрациями. – Минск, 1995. – 28 с.
4. ГОСТ 7.32-2017. Единая система программной документации. Оформление иллюстраций и рисунков. – Москва: Изд-во стандартов, 2017. - 28 с.
5. ГОСТ 7.82-2001. Единая система программной документации. Библиографическая запись. Библиографическое описание электронных ресурсов. – Минск, 2001. – 24 с.
6. ГОСТ 19.001-77. Единая система программной документации. Общие положения. – Москва: Изд-во стандартов, 2010. - 5 с.
7. ГОСТ ISO/IEC 23270:2003. Определение формы и интерпретация программ, написанных на языке программирования C#. – Технический комитет: ИСО/МЭК СТК 1/ПК 22, 2003-04. -471с.
8. C# для чайников: изд. Диалектика-Вильямс / Мюллер Джон Поль, Семпф Билл – 2019. – 608 с.
9. C# 7 и .NET Core. Кроссплатформенная разработка для профессионалов: изд. Питер / Прайс Марк – 2018. – 640 с.
10. Сайт вопросов и ответов для программистов - https://ru.stackover- flow.com/ (24.05.2024)
11. Сайт статьей по IT-сфере - https://habr.com/ (04.05.2024)
12. Зыков Н.В. Единые требования к оформлению курсового и дипломного проекта (работы): методические указания для студентов очного и заочного обучения всех специальностей: изд.4, испр. и доп./ Н.В. Зыков - Чита: ЗабГК, 2020. - 53 с.
13. А
14. А
15. А
16. А
17. А
18. А
19. А
20. А
21. А
22. А
23. А
24. А
25. А
26. А