



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

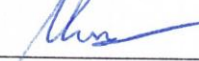
«МИРЭА – Российский технологический университет»
РТУ МИРЭА


Институт кибербезопасности и цифровых технологий
Кафедра КБ-14 «Цифровые технологии обработки данных»

КУРСОВАЯ РАБОТА

по дисциплине «Технологии программирования»
(наименование дисциплины)

Тема курсовой работы Разработка программы для управления IT-проектами

Студент группы Митина М. Д., БСБО-04-22
(Ф.И.О., учебная группа) 
(подпись студента)

Руководитель
курсовой работы Кашкин Е. В., к.т.н., доцент каф. КБ-14
(Ф.И.О., должность, ученое звание,
ученая степень) 
(подпись руководителя)

Рецензент
(при наличии) _____
(Ф.И.О., должность, ученое звание, ученая
степень) _____
(подпись рецензента)

Курсовая работа
представлена
к защите « 7 » 06 2023г.

Допущена
к защите « 7 » 06 2023г.

Москва 2023 г.



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт кибербезопасности и цифровых технологий
Кафедра КБ-14 «Цифровые технологии обработки данных»

Утверждаю
Заведующий кафедрой
Иванова И.А.
(подпись) (Ф.И.О.)
« 14 » 02 20 23 г.

ЗАДАНИЕ

на выполнение курсовой работы по дисциплине
«Технологии программирования»

Тема курсовой работы Разработка программы для управления IT-проектами

Студент Митина Марина Денисовна Группа БСБО-04-22

Исходные данные

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

Титульный лист; Содержание; Введение; Глава 1. Исследование предметной области;

Глава 2. Проектирование архитектуры программы; Глава 3. Разработка программы;

Заключение; Список использованной литературы; Приложение А. Листинг кода;

Приложение Б. Интерфейс приложения

Срок предоставления к защите курсовой работы

до « 31 » 05 20 23 г.

Задание на курсовую работу выдал

(подпись руководителя)

Кашкин Е.В.

(Ф.И.О. руководителя)

Задание на курсовую работу получил

(подпись обучающегося)

Митина М.Д.

(Ф.И.О. обучающегося)

Москва 2023 г.

Содержание

Введение.....	7
Глава 1. Исследование предметной области	8
1.1. Методики управления проектами в сфере IT	8
1.1.1. Каскадная модель Waterfall.....	8
1.1.2. Гибкая модель Agile.....	9
1.2. Анализ существующих программных решений	11
1.2.1. «Trello»	11
1.2.2. «Jira»	12
1.2.3. «Asana»	14
1.2.4. Сравнительный анализ	16
1.3. Формирование требований.....	18
Выводы	18
Глава 2. Проектирование архитектуры программы	20
2.1. ER-диаграмма базы данных	20
2.2. Функциональная схема системы	23
2.3. Описание модулей и алгоритмов системы	23
2.3.1. Модуль авторизации.....	24
2.3.2. Модуль регистрации.....	24
2.3.3. Модуль изменения пользователя	25
2.3.4. Модуль создания проекта	26
2.3.5. Модуль изменения проекта.....	26
2.3.6. Модуль создания этапа.....	27
2.3.7. Модуль изменения этапа	28
2.3.8. Модуль создания задачи.....	29
2.3.9. Модуль изменения задачи.....	29
2.3.10. Модуль взаимодействия с метками.....	30
2.3.11. Модуль перемещения элементов	31
Выводы	32

Глава 3. Разработка программы	33
3.1. Описание структуры системы	33
3.1.1. Класс DB	34
3.1.2. Класс User	34
3.1.3. Класс Project	36
3.1.4. Класс Stage	39
3.1.5. Класс Task	41
3.1.6. Класс Tag	43
3.1.7. Класс Notification	46
3.1.8. Класс Attachment	47
3.1.9. Класс TagBoxElem	48
3.2. Описание работы системы	48
3.2.1. Форма LoginForm	49
3.2.2. Форма SignUpForm	51
3.2.3. Форма MainForm	53
3.2.4. Форма UserForm	56
3.2.5. Форма CreateProjectForm	59
3.2.6. Форма ProjectForm	60
3.2.7. Форма InviteUserForm	66
3.2.8. Форма TaskForm	68
3.2.9. AddDateForm	74
3.2.10. AddRespForm	75
3.2.11. LabelForm	78
3.2.12. NewLabelForm	80
Выводы	81
Заключение	83
Список использованной литературы	84
Приложение А. Листинг кода	85
Приложение А.1. Программный код класса DB	85
Приложение А.2. Программный код класса User	85

Приложение А.3. Программный код класса Project	88
Приложение А.4. Программный код класса Stage.....	92
Приложение А.5. Программный код класса Task.....	96
Приложение А.6. Программный код класса Tag	105
Приложение А.7. Программный код класса Attachment	108
Приложение А.8. Программный код класса Notification	109
Приложение А.9. Программный код класса TagBoxElem	111
Приложение А.10. Программный код класса Program.....	111
Приложение А.11. Программный код формы MainForm.....	111
Приложение А.12. Программный код формы LoginForm.....	113
Приложение А.13. Программный код формы SignUpForm	114
Приложение А.14. Программный код формы UserForm.....	116
Приложение А.15. Программный код формы ChangePass.....	118
Приложение А.16. Программный код формы CreateProjectForm	119
Приложение А.17. Программный код формы InviteUserForm	120
Приложение А.18. Программный код формы ProjectForm.....	121
Приложение А.19. Программный код формы TaskForm	126
Приложение А.20. Программный код формы LabelForm	131
Приложение А.21. Программный код формы NewLabelForm	132
Приложение А.22. Программный код формы AddRespForm	133
Приложение А.23. Программный код формы AddDateForm.....	134
Приложение Б. Интерфейс приложения	136
Приложение Б.1. Окно авторизации	136
Приложение Б.2. Окно регистрации	137
Приложение Б.3. Окно главной страницы.....	137
Приложение Б.4. Окно личного кабинета	138
Приложение Б.5. Окно изменения пароля	138
Приложение Б.6. Окно нового проекта.....	139
Приложение Б.7. Окно пустого проекта	139
Приложение Б.8. Окно проекта. Вкладка «Доска Kanban»	140

Приложение Б.9. Окно проекта. Вкладка «Календарь»	141
Приложение Б.10. Окно проекта. Вкладка «Метки проекта»	142
Приложение Б.11. Окно приглашения нового участника.....	142
Приложение Б.12. Окно задачи.....	143
Приложение Б.12. Окно делегирования задачи	143
Приложение Б.13. Окно присвоения даты.....	143
Приложение Б.14. Окно присвоения метки.....	144
Приложение Б.15. Окно создания новой метки	144

Введение

Современный рынок информационных технологий ставит перед компаниями высокие требования к эффективному управлению IT-проектами. Как правило, управление такими проектами связано с большими объемами информации, высокими требованиями к производительности, а также необходимостью своевременного выполнения задач и достижения целей проекта. В таких условиях становятся незаменимы инструменты, позволяющие контролировать процесс и управлять ресурсами проекта максимально эффективно.

Для решения этих задач существует большое количество различных программных решений. Каждое из них обладает своими сильными и слабыми сторонами, среди недостатков особенно часто выделяется сложный интерфейс, требующий определенных навыков и опыта.

Таким образом целью курсовой работы является разработка программы, которая поможет более результативно управлять IT-проектами, и в то же время будет интуитивно понятна даже неопытному пользователю.

Курсовая работа состоит из трех глав. В них будут рассмотрены и решены следующие задачи:

Разбор современных методик управления и существующих готовых программных решений для управления IT-проектами. Выделение и анализ преимуществ и недостатков, отличительных особенностей, методов и инструментов.

Определение основных функций и объектов программы, отсеивание необязательных инструментов во избежание перегруженности интерфейса. Проектирование архитектуры, в том числе построение связей между объектами и составление алгоритмов функций.

Разработка программы с учетом всех вышеуказанных целей: исчерпывающего функционала и интуитивно понятно интерфейса.

Глава 1. Исследование предметной области

В данной главе будут рассмотрены наиболее популярные подходы к управлению проектами, а затем проведен обзор существующих программных продуктов с последующим сравнением, с целью выявления их достоинств и недостатков.

1.1. Методики управления проектами в сфере IT

За всё время существования IT-индустрии выработались различные подходы к управлению проектами, каждый из которых характеризуется собственными принципами и процедурами. Именно они определяют ход работы над проектом и взаимодействия между его участниками. Обзор этих методик поможет лучше понять цели конечных пользователей, а также наиболее эффективные способы их реализации.

1.1.1. Каскадная модель Waterfall

Методология делится на пять основных этапов, представленных на рисунке 1.1. Подход включает в себя следующие стадии: анализ требований, проектирование архитектуры, разработка, тестирование и внедрение [1].



Рисунок 1.1 – этапы каскадной модели Waterfall

Главная особенность каскадного подхода – последовательность. Переход от одной фазы к другой происходит только после полного завершения предыдущей. Для работы по этой методике необходимо иметь строгое представление о задачах проекта даже на самых ранних стадиях, поскольку эта модель не предполагает возвращения к предыдущим этапам.

Из этого следует главное преимущество и главный недостаток подхода – жёсткая структура. Благодаря строгому планированию становится легче организовывать работу и делегировать задачи. Однако такой неповоротливый подход не подойдет для управления крупным проектом, не имеющим чёткого технического задания или предполагающим постоянные изменения в нём.

Основным инструментом визуализации хода проекта в каскадной модели является диаграмма Ганта, изображенная на рисунке 1.2. Диаграмма представляет собой временную шкалу, на которой последовательно отображаются полосы - задачи проекта.

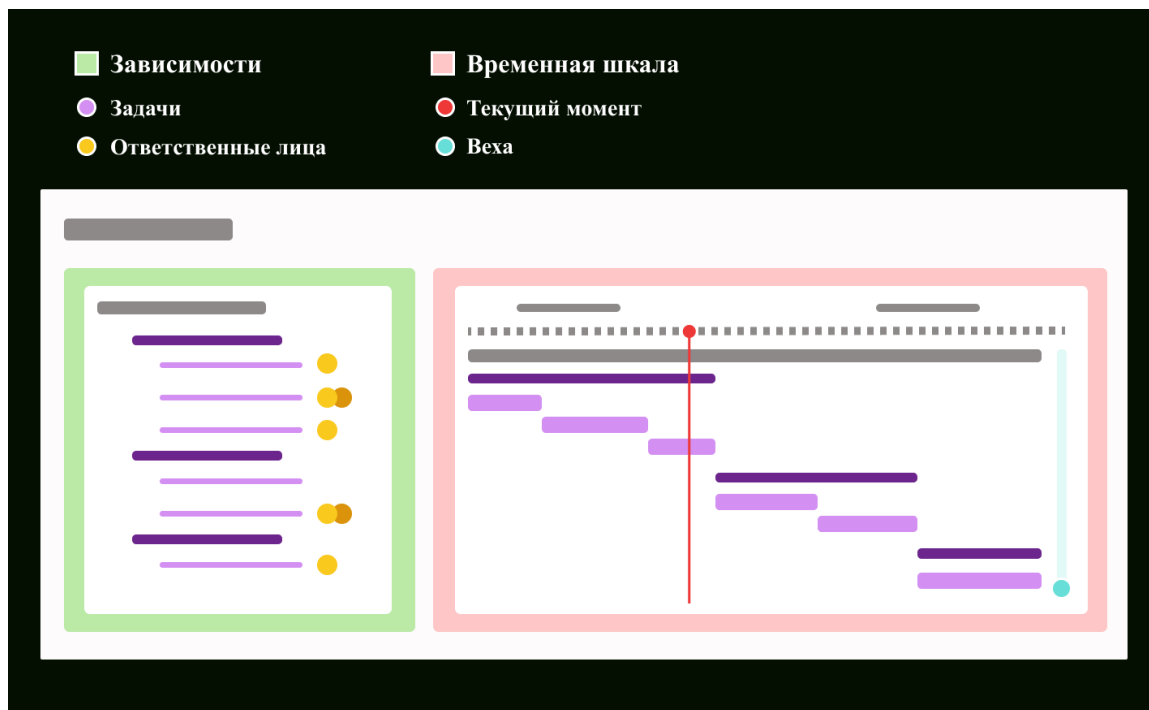


Рисунок 1.2 – основные составляющие диаграммы Ганта

1.1.2. Гибкая модель Agile

Один из главных принципов модели Agile – деление проекта на спринты (итерации) [2]. Каждый спринт длится определенное количество времени (от пары недель до пары месяцев) и состоит из стадий: анализ, проектирование, разработка, тестирование, внедрение и поддержка (см. рис. 1.3). Результатом каждого спринта является полноценная часть проекта.

Гибкая модель ценится командами за возможность быстрого реагирования на изменения требований к проекту, поскольку требования и планы оцениваются непрерывно. Однако, постоянные изменения и отсутствие четкого плана развития зачастую могут приводить к сдвигам сроков завершения проекта и незапланированным расходам.



Рисунок 1.3 – этапы гибкой модели Agile

Наглядное изображение хода проекта в гибкой модели обычно представляется в виде досок Kanban. Основными составляющими доски являются видимые сигналы, столбцы, ограничения WIP (незавершенной работы), commitment point (точка принятия обязательств) и delivery point (точка поставки продукта) (см. рис. 1.4). С помощью карточек и столбцов на доске команды могут понять какой объем работы следует взять на себя, по мере выполнения задач карточки перемещаются по столбцам и отражают ход работы.

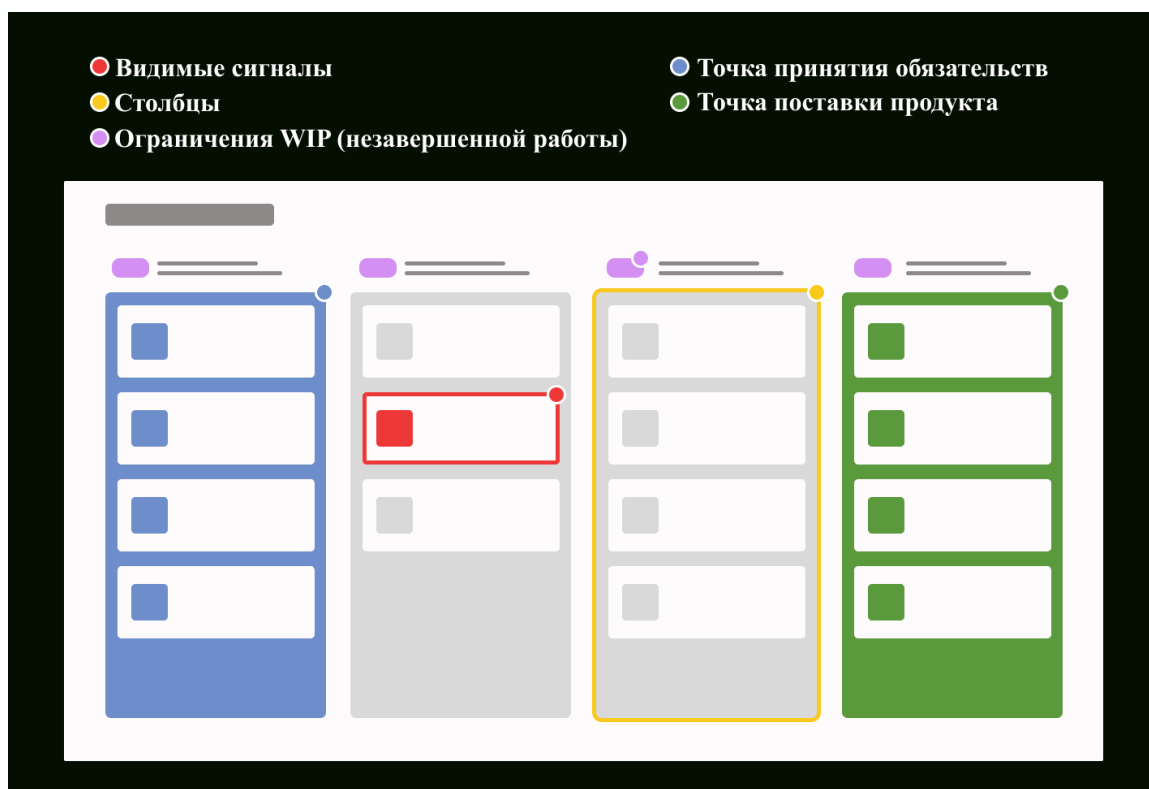


Рисунок 1.4 – основные составляющие Kanban-доски

1.2. Анализ существующих программных решений

В следующем разделе будут рассмотрены наиболее популярные сервисы для управления проектами: «Trello», «Jira» и «Asana» [3]. Результатом станет сравнительная таблица программ, отражающая их функционал и особенности. Она поможет сформировать список требований к собственной программе и избежать возможных недоработок.

1.2.1. «Trello»

Основными элементами «Trello» являются доски, колонки и карточки. Доска представляет собой пространства, где можно следить за информацией по проектам. Они могут быть как приватными для использования в личных проектах, так и публичными для командной работы. На доске создаются колонки, в которых содержатся карточки с заданиями (см. рис. 1.5). С помощью колонок можно создать процесс, при котором карточки будут продвигаться по каждому этапу от начала до конца или же просто служить хранилищем идей и информации.

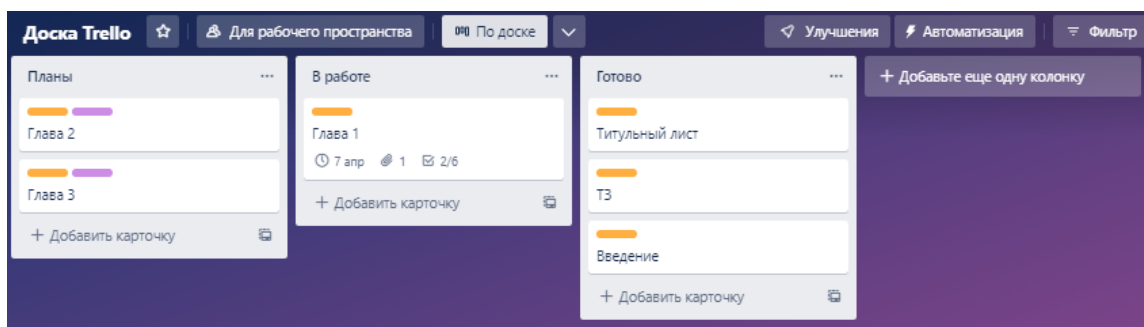


Рисунок 1.5 – Kanban-доска «Trello»

Конкретные задания и идеи формулируются на карточках. Помимо текстовой информации в карточку можно добавлять чек-листы, вложения и ссылки, присваивать метки и даты (см. рис. 1.6).

На обзорной платформе Capterra сервис «Trello» занимает 1 место по количеству отзывов в разделе Project Planning Software (программное обеспечение для планирования и управления проектами). Программа получила среднюю оценку 4.5 из 5 на основании 22368 отзывов [4].

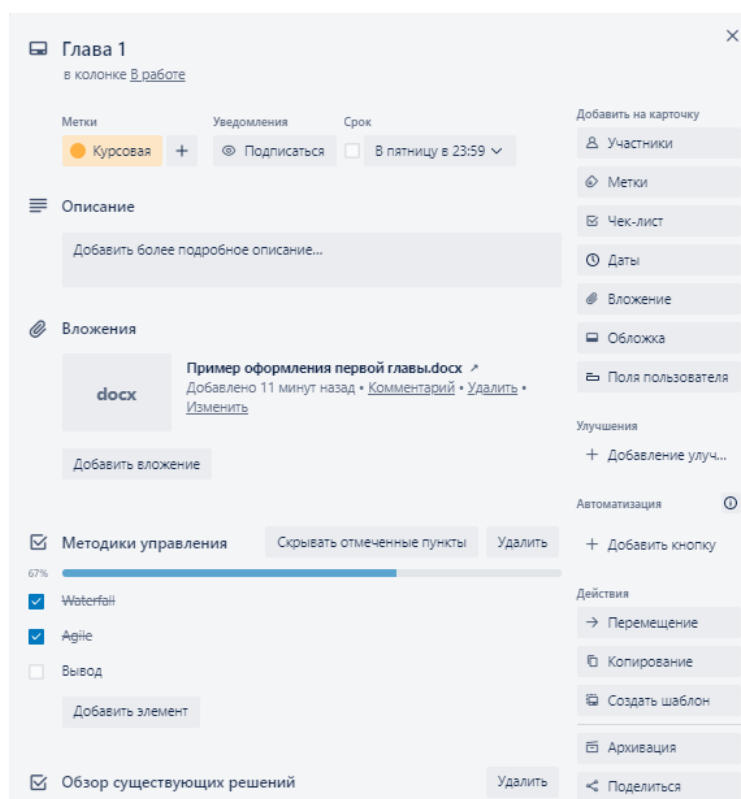


Рисунок 1.6 – карточка «Trello»

К числу недостатков можно отнести ограниченность функционала в бесплатной версии, привязку к сети, отсутствие уведомлений и базы данных сотрудников. Также на текущий момент десктопная и мобильная версии приложения недоступны на территории РФ.

1.2.2. «Jira»

Приложение «Jira» построено по принципам досок Kanban и использует гибкую методику разработки Agile. Но эти принципы дополняются множеством вспомогательных механизмов, таких как полный список задач, дашборды (документы со статистическими данными и инфографикой), дорожные карты (базируются на основе диаграммы Ганта), календари и т.д.

При создании доски проекта пользователь получает готовый шаблон Kanban-доски с 3 колонками-этапами (см. рис. 1.7). При необходимости пользователь может добавлять колонки, устанавливать ограничения на количество задач. В колонки добавляются карточки, по умолчанию есть 2 вида: задачи и эпика (крупные задачи, которые будут подразделяться на дочерние). Помимо этого, пользователь имеет возможность создавать

собственные макеты карточки, содержащие только необходимые блоки информации, такие как дата, время, метки, люди, URL-ссылки и т.д.

Доска JIRA

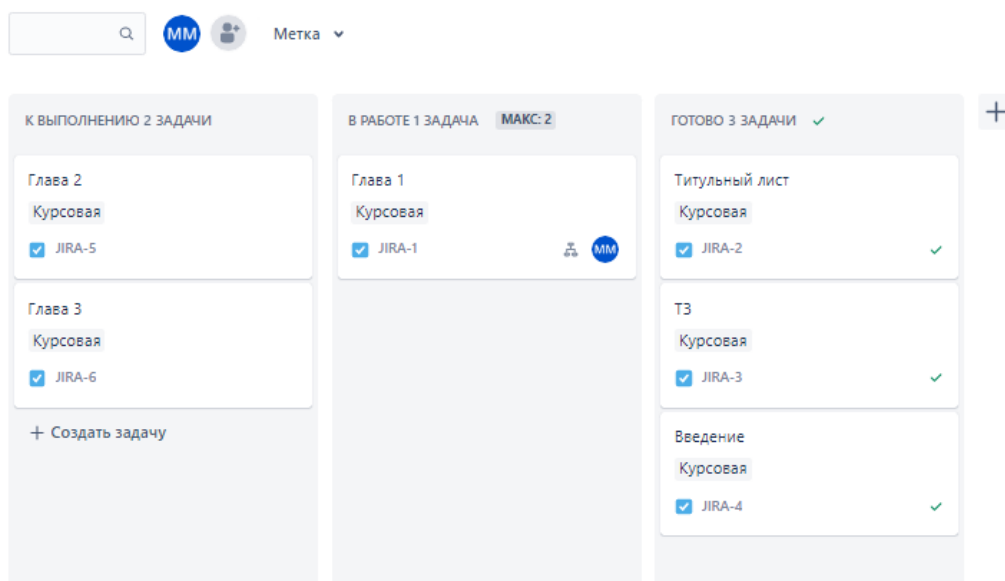


Рисунок 1.7 – Kanban-доска «Jira»

В карточке есть полноценный текстовый редактор для описания, возможность добавлять вложения и списки дочерних задач, инструменты для делегирования (см. рис. 1.8).

На обзорной платформе Capterra сервис «Jira» занимает 2 место по количеству отзывов в разделе Project Planning Software. Программа получила среднюю оценку 4.4 из 5 на основании 13148 отзывов [5].

К достоинствам «Jira» относят большие возможности для масштабирования, интеграцию с другими сервисами (GitHub, Figma, Microsoft 365 и т.д.), автоматизацию процессов, удобную инфографику для аналитики.

К недостаткам относят привязку к сети, невозможность отдать задачу нескольким членам команды, высокий порог вхождения и достаточно сложный интерфейс. Сервис «Jira» на данный момент недоступен на территории РФ, но несмотря на это многие команды продолжают использовать программу, используя VPN (виртуальные частные сети).

Глава 1

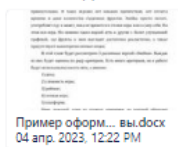
Прикрепить Добавить дочернюю задачу Добавить ссылку на задачу

Дедлайн 07 апр. 2023 г.

Описание

Добавить описание...

Вложения (1)



Дочерние задачи

Сортировка: ... +

Готово 33 %

JIRA-7	Методики управления	Готово
JIRA-8	Обзор существующих решений	К ВЫПОЛНЕНИЮ
JIRA-9	Формирование требований	К ВЫПОЛНЕНИЮ

Рисунок 1.8 – карточка «Jira»

1.2.3. «Asana»

Сервис «Asana» сочетает в себе множество различных инструментов и видов представления. Пользователь добавляет карточки заданий, которые затем представляются в виде списка, доски (см. рис. 1.9), хронологии или календаря (см. рис. 1.10). Переключение между видами выполняется с помощью верхней панели управления, что многие пользователи находят очень удобным.

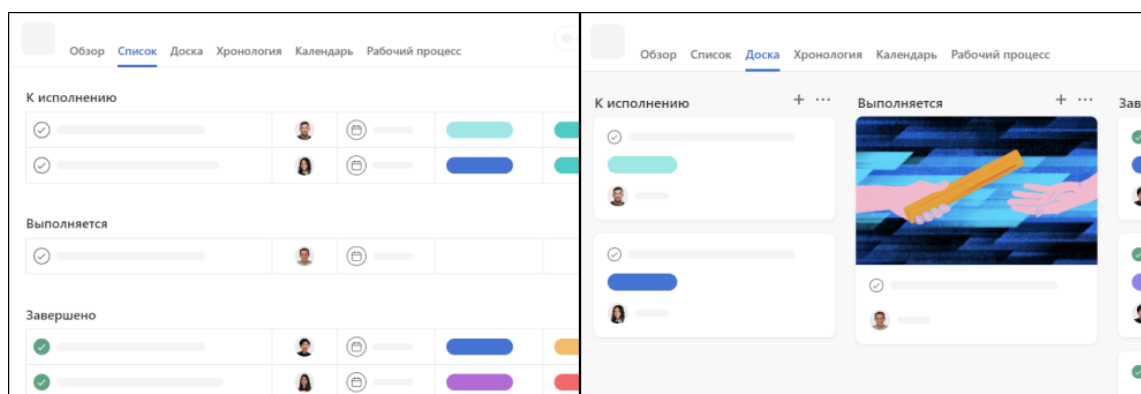


Рисунок 1.9 – список и доска «Asana»

Карточки «Asana» поддерживают добавление вложений, присваивание статуса и приоритета, разделение на подзадачи (см. рис. 1.11).

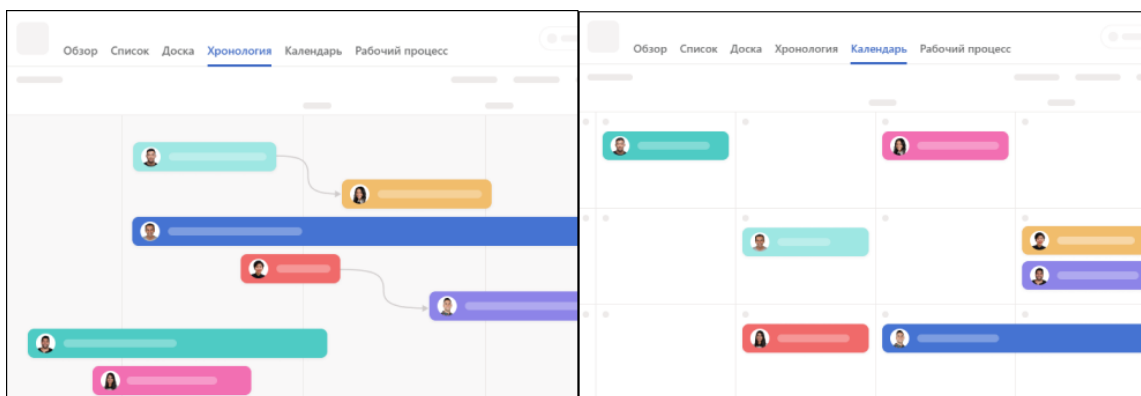


Рисунок 1.10 – хронология и календарь «Asana»

На обзорной платформе Capterra сервис «Asana» занимает 3 место по количеству отзывов в разделе Project Planning Software. Программа получила общую среднюю оценку 4.5 из 5 на основании 11958 отзывов [6]. Простота использования оценена на 4.3 из 5.

Глава 1

Исполнитель ММ Марина Митина ✕ Сделать сегодня ▼

Срок выполнения 📅 24 мар – 7 апр ✕

Проекты 🟢 Проект Asana В процессе ▼
Добавить в проекты

Зависимые элементы + Добавить зависимые элементы

🕒 Приоритет Высокий

🕒 Статус Под угрозой

Описание

What is this task about?

Подзадачи

- ✅ Методики управления
- ✅ Анализ существующих решений
- ✅ Формирование требований

+ Добавить подзадачу

Рисунок 1.11 – карточка «Asana»

Среди рассмотренных сервисов отличительной особенностью «Asana» являются элементы геймификации, в программе предусмотрена система достижений и рейтинга. Некоторые команды отмечают, что это помогает им

повышать продуктивность как команды в целом, так и отдельных разработчиков.

К достоинствам «Asana» можно отнести множество сервисов для интеграции (Google Drive, Slack, GitHub и т.д.), относительно простой и понятный интерфейс, возможность работать без сети (хоть и с ограниченным функционалом).

К недостаткам же относятся ограниченные возможности бесплатной версии, отсутствие Push-уведомлений и возможности ограничивать права доступа для пользователей (любой исполнитель может менять задачу и сроки).

1.2.4. Сравнительный анализ

Теперь, рассмотрев каждую из программ по отдельности, можно выделить ключевые критерии для их сравнения:

1. цена (нижняя и верхняя границы);
2. оценка на обзорной платформе Capterra;
3. наличие инструментария досок Kanban;
4. визуализация проекта в виде календаря или дорожной карты;
5. наличие инструментов для командной работы;
6. простота в использовании;
7. кроссплатформенность;
8. возможность работы offline;
9. русификация;
10. доступность на территории РФ.

Для критериев оценки выбраны следующие шкалы:

1. абсолютная – для критериев 1-3 и 7;
2. бинарная – для критериев 4-6 и 8-12.

На основании критериев представлено суммарное изложение обзора существующих программ (см. таблицу 1.1).

Таблица 1.1 - сравнительный анализ существующих продуктов

Программа Критерий	«Trello»	«Jira»	«Asana»
Нижняя граница цены	0 Р	0 Р	0 Р
Цена за полный функционал (в месяц)	17.50 \$ (≈1350 Р)	14.50 \$ (≈950 Р)	13.49 \$ (≈1050 Р)
Оценка (из 5)	4.5	4.4	4.5
Kanban-доски	Да	Да	Да
Представление в виде календаря	Только по подписке	Да	Да
Инструменты для совместной работы	Да	Да	Да
Простота в использовании (из 5)	4.5	4.0	4.3
Элементы геймификации	Нет	Нет	Да
Кроссплатформенность	Да	Да	Да
Offline режим (работа без сети)	Нет	Нет	Да
Русификация	Да	Да	Да
Доступность на территории РФ	Только веб-версия	Нет	Да

Проанализировав сравнительную таблицу, можно сделать вывод, что основными характеристиками программ являются: наличие Kanban-доски, представление в виде календаря или временной шкалы и инструменты для совместной работы. На основании этих данных будут сформулированы требования к разрабатываемой программе.

1.3. Формирование требований

В результате проведенного анализа ведущих сервисов для проектного менеджмента выявлены наиболее значимые функции, которыми должна обладать программа для управления проектами. Таким образом, разрабатываемой программе необходимо отвечать следующим требованиям:

1. в программе должна быть функция создания проекта и задания ему сроков, а также добавления участников;
2. каждый проект должен иметь функцию создания и изменения колонок-этапов проекта;
3. в программе должна быть функция создания карточек-задач;
4. в программе должна быть возможность присваивать и изменять свойства задач, такие как название, сроки, исполнитель, метки, вложения, описание;
5. программа должна содержать функцию создания меток и присваивания их карточкам;
6. программа должна представлять задачи в различных видах: доске-
Kanban и календаре;

В качестве языка для написания программы был выбран язык программирования C#. Поскольку визуализация является важнейшей частью программы, в качестве интерфейса программирования будет использован Windows Forms.

Выводы

В данной главе был проведен обзор основных методик управления проектами: Waterfall и Agile, а также разбор 3 ведущих сервисов для управления проектами: «Trello», «Jira» и «Asana».

В обзоре были разобран функционал и интерфейс каждого из решений, выделены особенности, преимущества и недостатки. Вся информация была собрана в сравнительную таблицу для наглядного представления и более быстрого анализа.

Информация о методиках управления необходима для понимания потребностей будущих пользователей, а обзор существующих программных решений позволил сформировать четкие требования к разрабатываемому продукту. Списка требований для создания собственной программы необходимо придерживаться в следующих этапах разработки.

Глава 2. Проектирование архитектуры программы

Перед началом разработки необходимо определить, какие объекты и модули должна содержать программа, а также как они будут взаимодействовать между собой. Правильно спроектированная архитектура программы обеспечит более эффективную разработку и обслуживание приложения, а также позволит упростить внесение изменений и дополнений в будущем.

В данной главе будет представлена модель базы данных, описывающая сущности программы и связи между ними, а затем определена функциональная схема системы, разбивающая программу на модули, каждый из которых будет описан в виде блок-схемы.

2.1. ER-диаграмма базы данных

Поскольку программа будет состоять из множества связанных между собой элементов, для её функционирования необходима база данных. Устройство базы данных показано на рисунке 2.1 в виде ER-диаграммы (Entity-Relationship diagram - диаграмма «сущность-связь»).

ER-диаграмма показывает какие объекты содержатся в базе данных, их атрибуты, а также связи между объектами. Связи могут быть разных типов, в данной работе представлены следующие:

1. «к нулю, одному или многим» - сущность может быть связана с нулем, одной или несколькими другими сущностями, на диаграмме представлена как линия с кругом и тремя лапками на конце;
2. «к одному или многим» - сущность должна быть связана как минимум с одной другой сущностью, на диаграмме представлена как линия с поперечной чертой и тремя лапками на конце;
3. «к одному» - сущность должна быть связана с одной и только одной другой сущностью, на диаграмме представлена как линия с поперечной чертой на конце.

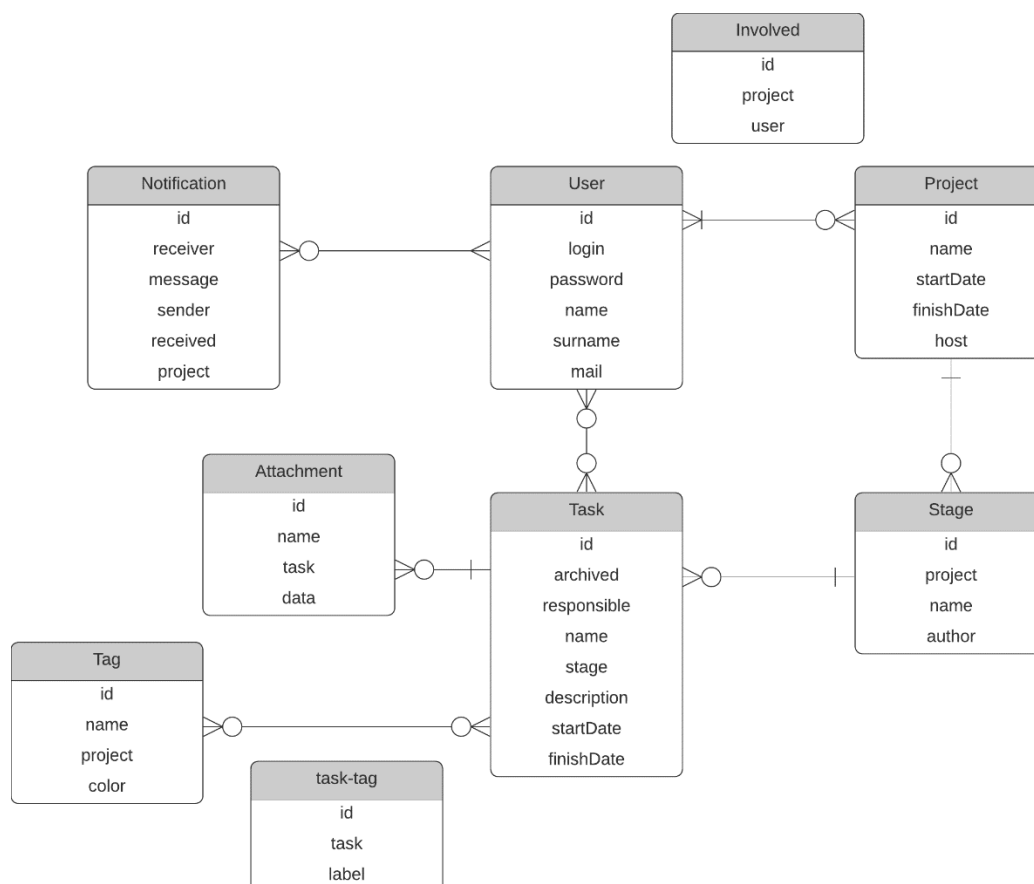


Рисунок 2.1 – ER-диаграмма базы данных

В данном проекте определены следующие основные сущности: «User» (пользователь), «Project» (проект), «Stage» (этап), «Task» (задача), «Attachment» (вложение), «Tag» (метка) и «Notification» (уведомление). Также присутствуют 2 вспомогательные таблицы, обеспечивающие связь «многие ко многим»: «Task-tag» и «Involved» (участники).

Сущность «User» имеет следующие свойства: id (идентификатор), login (логин), password (пароль), name (имя), surname (фамилия), mail (почта). «User» связана с сущностями «Project» (связь «к нулю, одному или многим» – пользователь может иметь несколько проектов), «Task» (связь «к нулю, одному или многим» – пользователю могут быть назначены ноль, одна или несколько задач).

Сущность «Project» содержит свойства id, name (название), startDate (дата начала работы), finishDate (дата окончания работы) и host (создатель проекта). «Project» связана с сущностями «User» (связь «к одному или многим» – в проекте должен участвовать минимум один пользователь),

«Stage» (связь «к нулю или многим» – проект может содержать от нуля до нескольких этапов).

Сущность «Stage» содержит свойства id, name, project и author. «Stage» связана с сущностями «Project» (связь «к одному» – доска может принадлежать одному и только одному проекту), «Stage» (связь «к нулю, одному или многим» – на доске может быть от нуля до нескольких этапов-колонок).

Сущность «Task» содержит свойства id, name, description (описание), startDate, finishDate, archived (статус выполнения), responsible (ответственное лицо), stage. «Task» связана с сущностями «User» (связь «к нулю, одному или многим» – задача может быть присвоена одному или нескольким пользователям или еще не быть присвоена вообще), «Stage» (связь «к одному» – задача принадлежит одному и только одному этапу), «Tag» (связь «к нулю, одному или многим» - у задачи может быть от нуля до нескольких меток), «Attachment» (связь «к нулю, одному или многим» - у задачи может быть от нуля до нескольких вложений).

Сущность «Tag» содержит свойства id, name, project и color (цвет). «Tag» связана с сущностями «Task» (связь «к нулю, одному или многим» – метка может быть присвоена одной или нескольким задачам, либо еще не быть присвоена вообще), «Stage» (связь «к нулю, одному или многим» – метка может быть присвоена одному или нескольким этапам, либо еще не быть присвоена вообще).

Сущность «Attachment» содержит свойства id, name, tasks и data (данные). «Attachment» связана с сущностями «Task» (связь «к одному» – вложение обязательно относится к одной задаче).

Сущность «Notification» содержит свойства id, receiver (получатель), message (сообщение), sender (отправитель), received (статус доставки), project. Сущность связана с «User» связью «к одному или многим» и с «Project» связью «к одному».

Таким образом, были рассмотрены все сущности, их атрибуты и связи. Далее на их основе будет создана функциональная схема системы.

2.2. Функциональная схема системы

В данном разделе будет рассмотрена функциональная схема системы (см. Рис. 2.2). Она состоит из модулей, которые представляют собой отдельный функциональный блок, и связей между ними, связи могут быть как однонаправленными, так и двунаправленными.

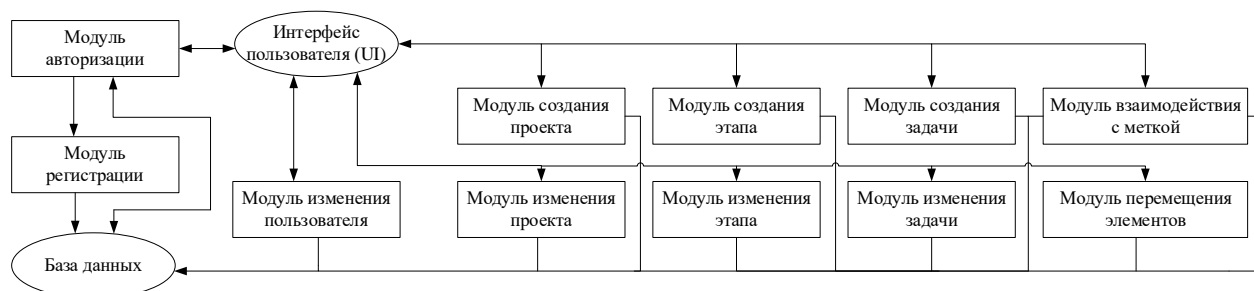


Рисунок 2.2 – функциональная схема системы

При запуске пользователь активирует модуль авторизации. Если у пользователя нет аккаунта, он может его создать с помощью модуля регистрации. Если авторизация пройдена успешно (логин и пароль найдены в базе данных), пользователь попадает в основное рабочее пространство, где может взаимодействовать со своими проектами и их элементами. Пользователь может создавать новые элементы с помощью модулей создания элементов (проекта, этапа, задачи и метки), а также изменять уже существующие с помощью модулей изменения и перемещения. Кроме того, пользователь может изменять информацию о самом себе (рабочую почту, имя или фамилию) через соответствующий модуль.

2.3. Описание модулей и алгоритмов системы

В данном разделе представлено проектирование главных функциональных модулей системы в виде блок-схем, а также описывается принцип их работы и рассматриваются алгоритмы, являющиеся основой модулей.

2.3.1. Модуль авторизации

Первым окном, встречающим пользователя, является окно авторизации, отображающая его работу блок-схема представлена на рисунке 2.3. Окно представляет собой 2 поля для ввода логина и пароля и 2 кнопки «Войти» и «Регистрация». При нажатии на кнопку «Регистрация» активируется модуль регистрации, а при введенном логине и пароле и нажатии кнопки «Войти» данные пользователя сверяются с базой данных. Если данные найдены открывается рабочее пространство (пустое если у пользователя нет проектов, либо заполненное проектами вошедшего пользователя), если нет – выводится сообщение «Пользователь не найден», и модуль начинает работу с начала.

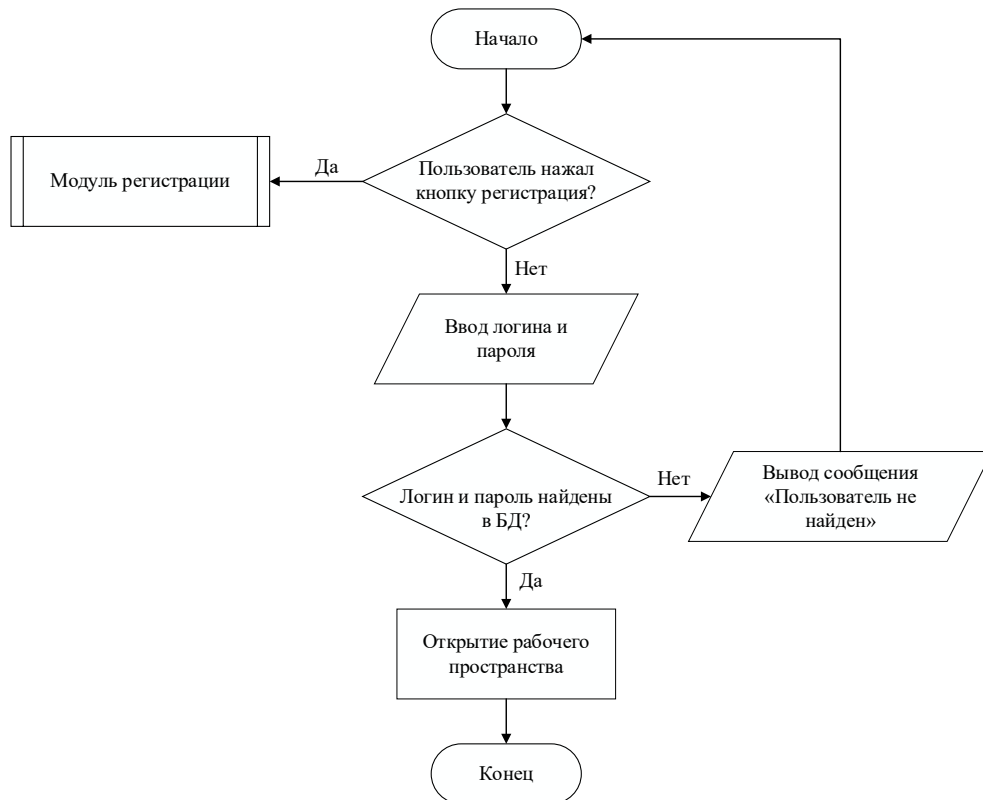


Рисунок 2.3 – блок-схема модуля авторизации

2.3.2. Модуль регистрации

Если пользователь использует программу впервые, ему необходимо пройти регистрацию. За это отвечает модуль регистрации, схематично изображенный на рисунке 2.4.

У пользователя открывается новое окно, представляющее собой форму с полями, некоторые из которых являются обязательными для заполнения (логин и пароль), а также кнопку «Зарегистрироваться». Когда кнопка нажата, проводится несколько проверок на длину и надежность пароля, уникальность логина и заполнение обязательных полей. Если все условия соблюдены данные передаются в базу данных, а пользователь перенаправляется в модуль авторизации. Если какое-то из условий не выполнено, выводится соответствующее сообщение об ошибке.

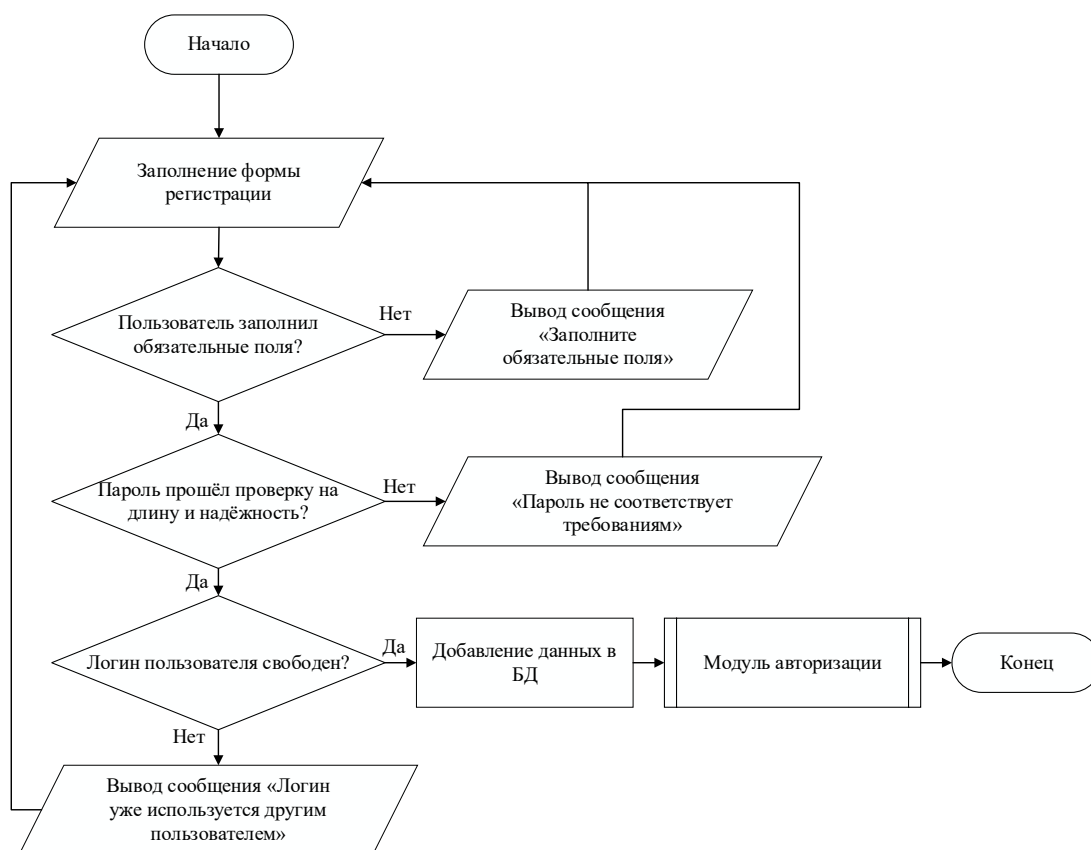


Рисунок 2.4 – блок-схема модуля регистрации

2.3.3. Модуль изменения пользователя

Если пользователь хочет добавить или изменить опциональные данные, указанные при регистрации, то через меню настроек он активирует модуль изменения пользователя, блок-схему которого можно увидеть на рисунке 2.5.

Настройки пользователя представляют собой поля с данными и кнопку «Сохранить». При нажатии на кнопку данные пользователя обновляются в

базе данных. После успешного обновления пользователю выводится сообщение «Данные изменены».

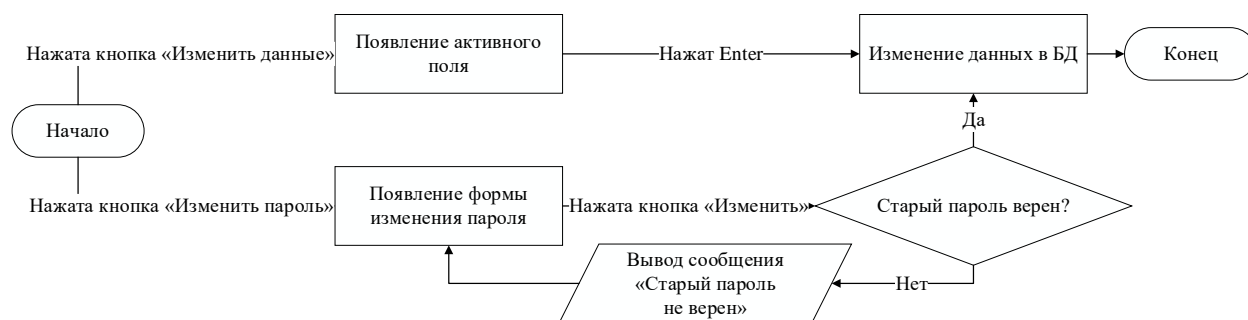


Рисунок 2.5 – блок-схема модуля изменения пользователя

2.3.4. Модуль создания проекта

После успешной авторизации пользователь попадает в рабочее пространство. Здесь он видит сводную статистику, свои проекты и кнопку «Создать проект» (либо только кнопку, если проектов еще нет). При нажатии на кнопку активируется модуль создания проекта, схематически представленный на рисунке 2.6, – открывается новое диалоговое окно, которое представляет собой поля и кнопку «Создать». Если кнопка нажата и обязательные поля заполнены, данные о проекте передаются в базу данных. Затем в рабочем пространстве создаётся проект и шаблон доски, и пользователь перенаправляется на его страницу.

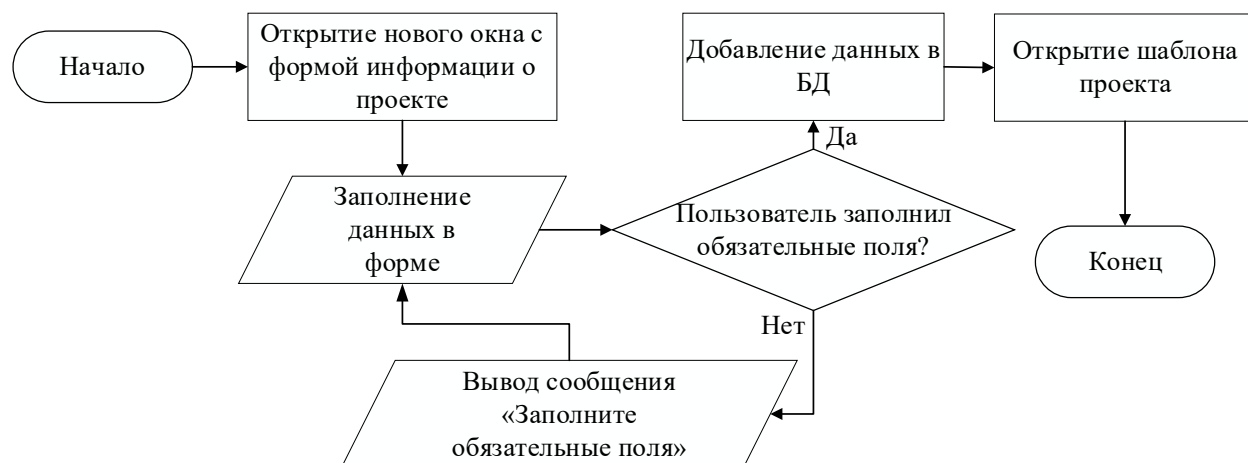


Рисунок 2.6 – блок-схема модуля создания проекта

2.3.5. Модуль изменения проекта

При необходимости изменить, удалить или добавить информацию о проекте пользователь нажимает на одну из отвечающих за это кнопок и

активирует модуль изменения проекта, блок-схема которого изображена на рисунке 2.7. При нажатии на кнопку удаления, вызывается окно с подтверждением удаления. Если оно подтверждено, проект удаляется из базы данных. Затем проект удаляется из рабочего пространства. Если пользователь изменил данные в полях и нажал на кнопку «Сохранить», данные обновляются в базе данных, затем изменяется отображение проекта на рабочем пространстве.

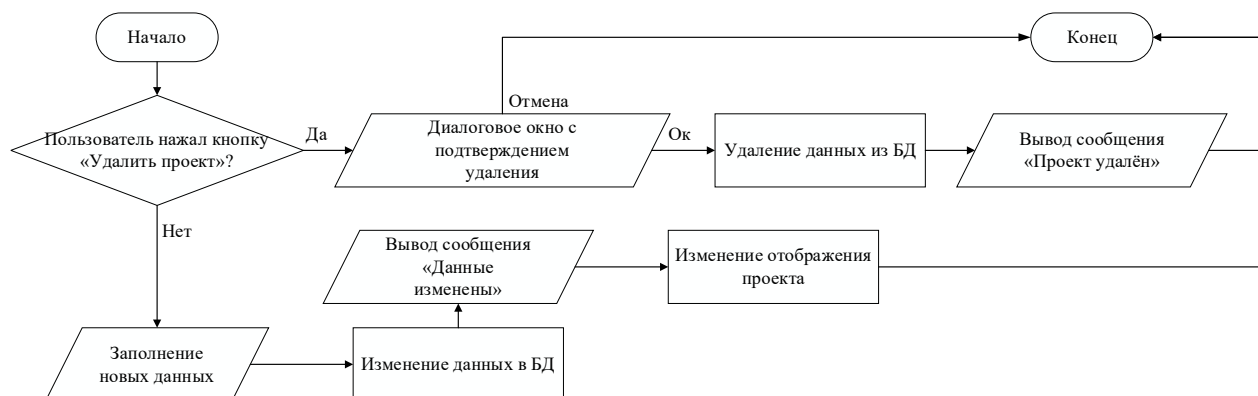


Рисунок 2.7 – блок-схема модуля изменения проекта

2.3.6. Модуль создания этапа

В рабочем пространстве проекта пользователя встречаются этапы доски проекта, и кнопка «Добавить этап». При нажатии на кнопку активируется модуль создания этапа, схематично изображенный на рисунке 2.8. Открывается поле, в которое пользователь вводит название этапа. После нажатия кнопки Enter на доске создается этап с выбранным названием, и информация о нем передается в базу данных.

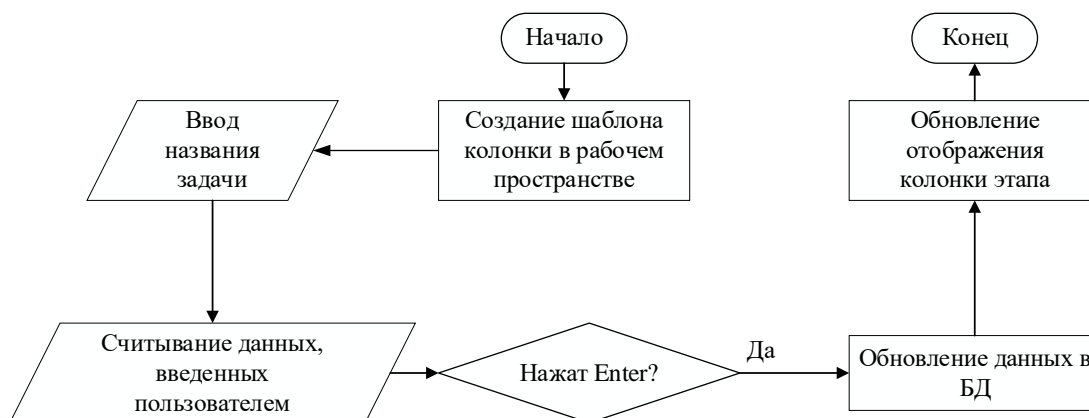


Рисунок 2.8 – блок-схема модуля создания этапа

2.3.7. Модуль изменения этапа

У каждого этапа имеется кнопка настроек. При нажатии на нее, перед пользователем появляется выпадающее меню с тремя опциями: переименовать этап, изменить лимит активных задач и удалить этап. Нажатие на кнопки активируют модуль изменения этапа, изображенный на рисунке 2.9.

При нажатии на кнопку «Переименовать» название этапа становится активным полем, и пользователь вводит новое название. После нажатия Enter данные проходят через модуль учета изменений и обновляют базу данных.

При нажатии на кнопку «Изменить лимит активных задач» число у названия этапа становится активным полем, и пользователь вводит новое значение. После нажатия Enter данные проходят через модуль учета изменений и обновляют базу данных.

При нажатии кнопки «Удалить» если этап не содержит задач, то он мгновенно удаляется из рабочего пространства доски, активирует модуль учета изменений и удаляется из базы данных. Если в этапе есть задачи, перед пользователем появляется всплывающее окно с подтверждением удаления: вместе с этапом будут также удалены его задачи. При нажатии ОК этап и задачи удаляются из базы данных и обновляется отображение рабочего пространства.

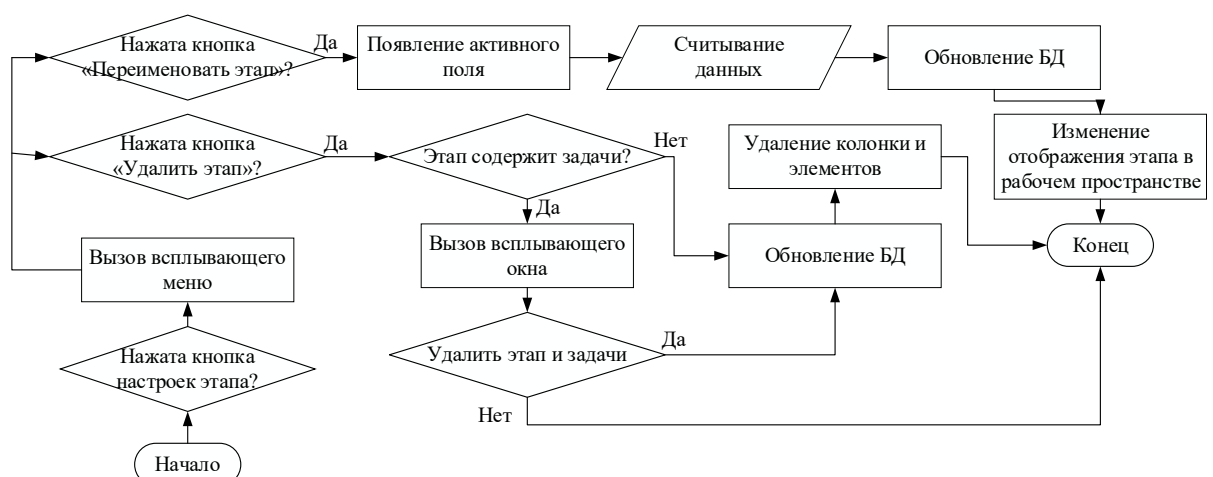


Рисунок 2.9 – блок-схема модуля изменения этапа

2.3.8. Модуль создания задачи

В колонке этапа помимо задач находится кнопка «Добавить задачу». При нажатии на нее активизируется модуль создания задачи, блок-схема которого представлена на рисунке 2.10. Кнопка смещается вниз, а на ее месте появляется карточка задачи с активными полем для названия и двумя кнопками: «Назначить задачу» и «Установить сроки».

Если пользователь нажимает кнопку назначения задачи, появляется выпадающий список, где можно выбрать исполнителя. При нажатии на кнопку сроков, появляется форма с календарем. После ввода данных и нажатия Enter карточка обновляется, а информация передается в базу данных.

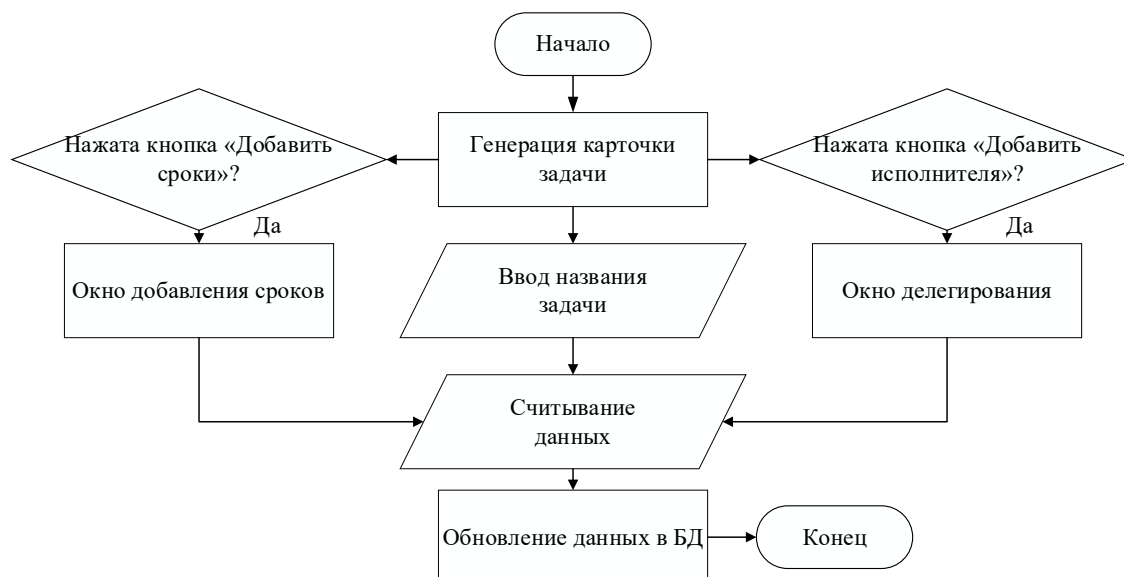


Рисунок 2.10 – блок-схема модуля добавления задачи

2.3.9. Модуль изменения задачи

При нажатии на кнопки задачи или саму существующую задачу активируется модуль изменения задачи, схематически изображенный на рисунке 2.11. Появляется новое окно, в котором представлена вся информация о задаче. Здесь пользователь может изменить исполнителя, сроки и статус, добавить метки, описание и вложения, а также удалить задачу.

Те же шаги проходятся при изменении карточки через рабочее пространство (например, при клике на кнопку «Выполнено», находящейся на карточке).

При удалении задачи последовательность действий во многом аналогична, но отличительной особенностью является всплывающее окно с кнопкой «Отменить», при нажатии на которую не происходит обновления базы данных, а отображение карточки возвращается к предыдущему виду.

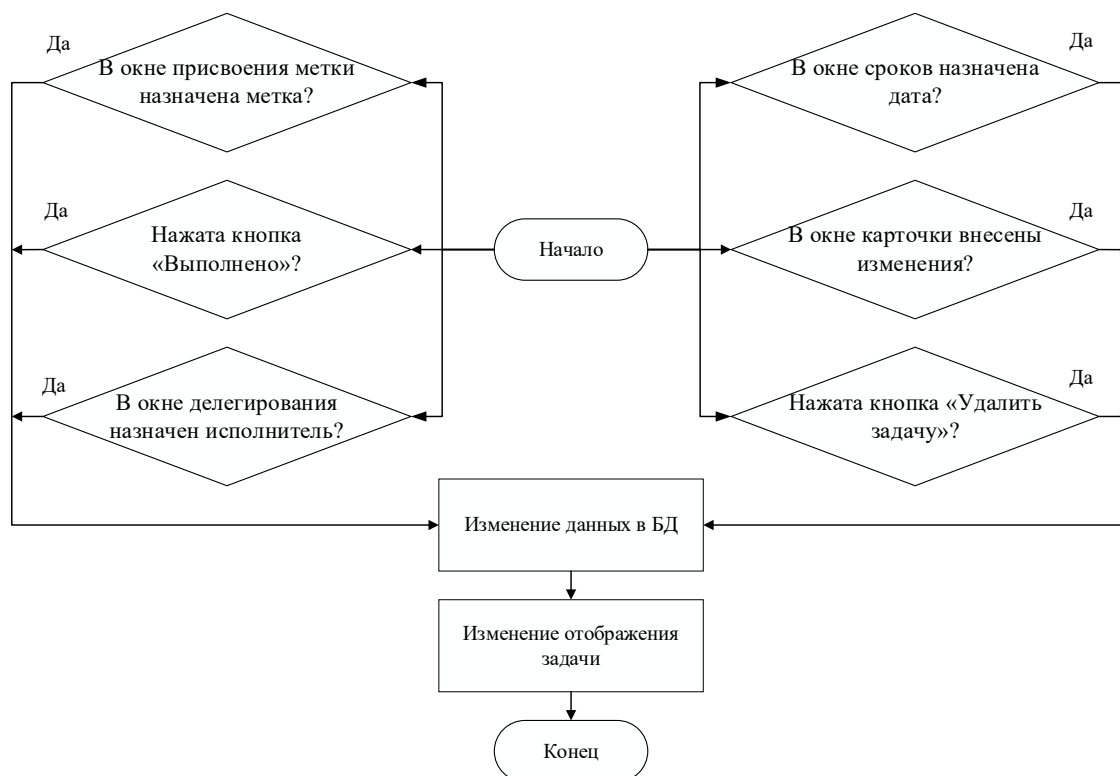


Рисунок 2.11 – блок-схема модуля изменения задачи

2.3.10. Модуль взаимодействия с метками

Для лучшей визуализации в программе предусмотрено создание и присваивание меток задачам. Этот функционал обеспечивается модулем взаимодействия с метками, блок-схема которого представлена на рисунке 2.12. Модуль активируется из раздела карточки задачи при присваивании новой метки. При нажатии на соответствующую кнопку на странице карточки вызывается всплывающее меню, содержащее уже существующие метки и кнопку добавления новой.

При добавлении новой метки пользователь вводит ее название и выбирает цвет, затем активируется модуль учета изменений, и обновляется база данных, метка присваивается задаче, со страницы которой была создана. Модуль также активируется при нажатии на метку в рабочем пространстве,

вызывая всплывающее меню с кнопками «Переименовать», «Удалить метку», «Удалить метку с задачи» и «Изменить цвет».

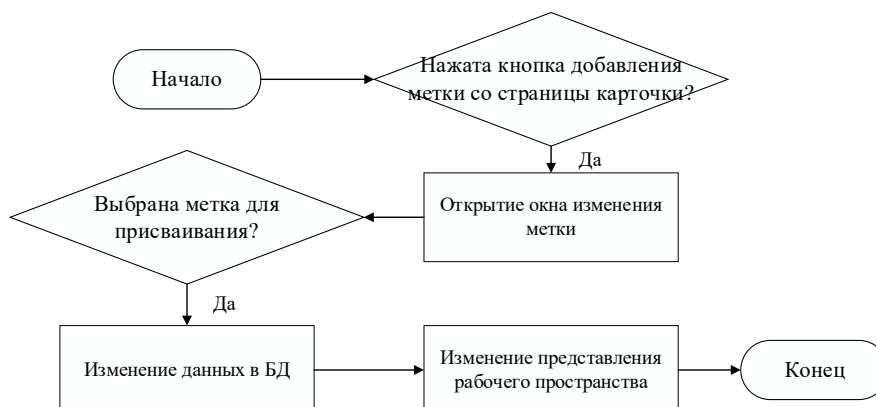


Рисунок 2.12 – блок-схема модуля взаимодействия с метками

2.3.11. Модуль перемещения элементов

Данные этапов и заданий могут быть изменены не только посредством внесения их через поля, но и с помощью механики «Drag&drop». Например, карточка задачи может быть перетащена с помощью мыши в другую колонку, что изменит принадлежность карточки к этапу. За эту функцию отвечает модуль перемещения элементов, блок-схема которого представлена на рисунке 2.13.

Если пользователь зажимает мышь на элементе карточки, создается копия этого элемента (она будет показывать место, куда переместится элемент после отпускания мыши). Если элемент может быть перемещен в зону, где стоит курсор, то на это место перемещается копия элемента. Так происходит до тех пор, пока мышь не будет отпущена. Как только это произошло, элемент перемещается и фиксируется, обновляется база данных.

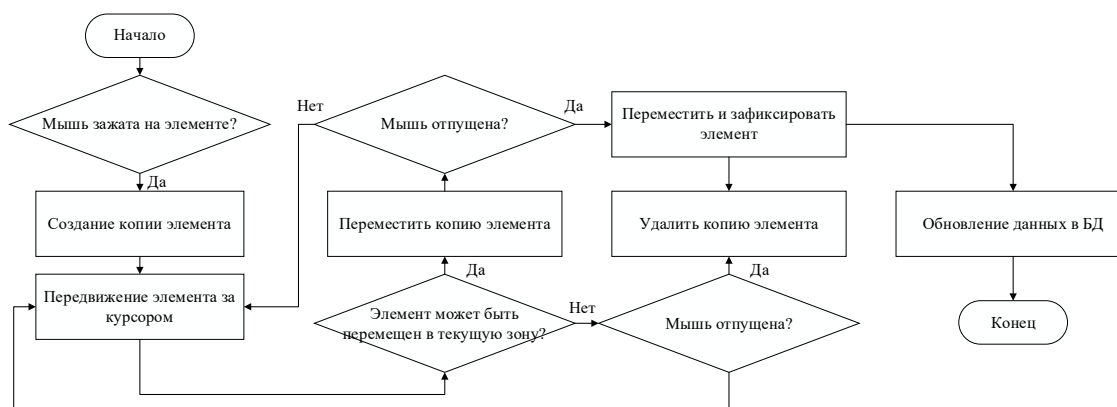


Рисунок 2.13 – блок-схема модуля перемещения элементов

Выводы

В данной главе была спроектирована архитектура системы. Была составлена ER-диаграмма, содержащая сущности «User», «Project», «Stage», «Task», «Attachment», «Tag» и «Notification». Диаграмма ляжет в основу базы данных программы. Также была продумана функциональная схема системы, определены основные модули: авторизации, регистрации, создания и изменения проекта, создания и изменения задачи, создания и изменения этапа, взаимодействия с метками, перемещения элементов. Все модули решают определенные задачи для реализации функционала, каждый модуль был представлен в виде блок-схемы, описывающей алгоритм работы модуля, что упростит написание программного кода в дальнейшем и обеспечит более эффективную разработку.

Глава 3. Разработка программы

В рамках данной главы разработана программа для управления проектами. Её задачей является предоставление пользователю возможности создавать проекты, этапы и задачи внутри них, отслеживать выполнение задач, а также визуализировать ход проекта в удобной для восприятия форме.

3.1. Описание структуры системы

Программа представляет из себя Windows приложение. Она реализована с использованием интерфейса программирования приложений Windows Forms и языка программирования C#. Для хранения данных программы была выбрана база данных MySQL. В процессе реализации программных модулей были разработаны следующие классы:

1. **DB** – класс, отвечающий за соединение с сервером и подключение к базе данных;
2. **User** – класс, содержащий методы для работы с данными пользователя и хранящий информацию о нем;
3. **Project** – класс, содержащий методы для работы с данными проектов и хранящий информацию о них;
4. **Stage** - класс, содержащий методы для работы с данными этапов и хранящий информацию о них;
5. **Task** - класс, содержащий методы для работы с данными задач и хранящий информацию о них;
6. **Tag** – класс, содержащий методы для работы с данными меток и хранящий информацию о них;
7. **Notification** – класс, содержащий методы для работы с данными уведомлений и хранящий информацию о них;
8. **Attachment** – класс, содержащий методы для работы с данными приложений и хранящий информацию о них;
9. **TagBoxElem** – класс объектов из которых состоят списки элемента ComboBox.

В данном разделе будет подробно рассмотрен каждый класс и приведен листинг кода основных методов.

3.1.1. Класс DB

Класс DB предоставляет базовый функционал для работы с базой данных MySQL, включая открытие и закрытие соединения. Класс содержит одно поле connection объекта MySqlConnection из библиотеки MySql.Data. Оно инициализируется строкой подключения.

Методы класса openConnection и closeConnection изменяют состояние соединения с помощью методов Open() и Close(), а метод getConnection() представляет текущее соединение, что позволяет другим частям кода получить доступ к соединению с базой данных и выполнению запросов. Код класса показан на листинге 3.1.

```
MySqlConnection connection = new
MySqlConnection("server=localhost;port=3306;username=root;password=root;database=tik
task");

    public void openConnection()
    {
        if (connection.State == System.Data.ConnectionState.Closed) {
connection.Open(); }
        }
    public void closeConnection()
    {
        if (connection.State == System.Data.ConnectionState.Open) {
connection.Close(); }
        }
    public MySqlConnection getConnection()
    {
        return connection;
    }
```

Листинг 3.1 – класс DB

3.1.2. Класс User

Класс User был разработан для более удобного взаимодействия с данными пользователей и уменьшения количества запросов к базе данных. Класс содержит следующие поля: id, name, surname, login, mail, все они являются публичными переменными типа string. Заполнение происходит при помощи конструктора либо напрямую, либо через перегрузку, когда на вход передается только id пользователя, и конструктор сам делает запрос в базу данных. Код конструктора представлен на листинге 3.2.

```

public User(string id = null, string name = null, string surname = null, string
login = null, string mail = null)
{
    this.id = id;
    this.name = name;
    this.surname = surname;
    this.login = login;
    this.mail = mail;
}
public User(string id)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE
'id' = @userID", db.getConnection());
    command.Parameters.Add("@userID", MySqlDbType.Int32).Value = id;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    db.openConnection();
    adapter.Fill(table);
    db.closeConnection();
    this.id = id;
    foreach (DataRow row in table.Rows)
    {
        this.mail = row["mail"].ToString();
        this.name = row["name"].ToString();
        this.surname = row["surname"].ToString();
        this.login = row["login"].ToString();
    }
}

```

Листинг 3.2 – конструкторы класса User

Ряд методов класса User отвечает за получение данных из базы данных, например метод GetUserProject() возвращает таблицу типа DataTable, содержащую строки всех проектов, в которых задействован пользователь. Код метода представлен на листинге 3.3. Подобный функционал выполняют методы GetUserTasks() (возвращает таблицу задач пользователя), GetUserTasksByDate() (возвращает таблицу задач пользователя, назначенных на определенную дату) и GetUserNotifications() (возвращает таблицу уведомлений, адресованных пользователю). Код этих методов приведен в приложении А.2.

```

public DataTable GetUserProjects() {
    DB db = new DB();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `involved` WHERE
'user' = @user", db.getConnection());
    command.Parameters.Add("@user", MySqlDbType.Int32).Value = id;
    adapter.SelectCommand = command;
    adapter.Fill(table);
    return table;
}

```

Листинг 3.3 – метод GetUserProjects()

Следующая группа методов отвечает за обновление данных пользователя. Метод ChangeName() создает и выполняет запрос к базе данных, изменяя столбец «name» в таблице «users». Код метода представлен на листинге 3.4. Методы ChangeSurname() и ChangePassword() работают аналогично, их код представлен в приложении А.2.

```
public void ChangeName(string newName)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `users` SET `name` =
@newName WHERE `id` = @userID", db.getConnection());
    command.Parameters.Add("@newName", MySqlDbType.VarChar).Value = newName;
    name = newName;
    command.Parameters.Add("userID", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
}
```

Листинг 3.4. – метод ChangeName()

3.1.3. Класс Project

Как и класс User класс Project был разработан для более удобного взаимодействия с данными проектов и уменьшения количества запросов к базе данных. Класс содержит публичные строковые поля id, name, host, поля типа DateTime? startDate и FinishDate, поле previewCard типа Panel и поле main типа MainForm. Данные заполняются через конструктор либо напрямую, либо, если передается только id проекта, конструктор сам выполняет запрос к базе данных и заполняет данные из полученной таблицы. Код конструктора и его перегрузки представлен на листинге 3.5.

```
public Project(MainForm main, string name = null, DateTime? startDate =
null, DateTime? finishDate = null, string host = null, string id = null)
{
    this.id = id;
    this.name = name;
    this.startDate = startDate;
    this.finishDate = finishDate;
    this.host = host;
    this.main = main; }
public Project(string id)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `projects` WHERE
`id` = @pI", db.getConnection());
    command.Parameters.Add("@pI", MySqlDbType.Int32).Value = id;
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    DataTable table = new DataTable();
```

```

        adapter.Fill(table);

        foreach (DataRow row in table.Rows)
        {
            this.id = id;
            this.name = row["name"].ToString();
            this.startDate = DateTime.Parse(row["startDate"].ToString());
            this.finishDate = DateTime.Parse(row["finishDate"].ToString());
            this.host = row["host"].ToString();
        }
    }
}

```

Листинг 3.5 – конструктор Project

Функцию отправки данных проекта в базу данных выполняет метод PushToDB(). Его код представлен на листинге 3.6. При создании нового проекта обновляются 2 таблицы: «projects», хранящая все данные о проектах и «involved», хранящая пары участник-проект.

```

public void PushToDB()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO `projects` (`name`, `startDate`, `finishDate`, `host`) VALUES (@pN, @pSD, @pED, @pH);", db.getConnection());

    command.Parameters.Add("@pN", MySqlDbType.VarChar).Value = name;
    command.Parameters.Add("@pSD", MySqlDbType.DateTime).Value = startDate.Value;
    command.Parameters.Add("@pED", MySqlDbType.DateTime).Value = finishDate.Value;
    command.Parameters.Add("@pH", MySqlDbType.Int32).Value = host;

    db.openConnection();
    command.ExecuteNonQuery();
    this.id = command.LastInsertedId.ToString();
    command = new MySqlCommand("INSERT INTO `involved` (`project`, `user`) VALUES (@pI, @uI);", db.getConnection());
    command.Parameters.Add("@pI", MySqlDbType.Int32).Value = this.id;
    command.Parameters.Add("@uI", MySqlDbType.Int32).Value = this.host;
    command.ExecuteNonQuery();
    db.closeConnection();
}

```

Листинг 3.6 – метод PushToDB()

После создания нового проекта создается карточка его превью на главной странице. За это отвечает метод GetPreviewCard(), который возвращает объект типа Panel. Данный метод является обширной процедурой, содержащей множество операций инициализации графических элементов, а также задания значений их свойств, поэтому представлен в приложении А.3.

Следующая группа методов отвечает за получение информации о деталях проекта. GetProjectStages() возвращает таблицу всех этапов проекта, его код представлен на листинге 3.7. Аналогично работают методы GetProjectTasks() (получает таблицу, содержащую все задачи проекта) и GetProjectLabels() (получает таблицу меток проекта), их код представлен в приложении А.4.

```
public DataTable GetProjectsStages()
{
    DB db = new DB();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();

    MySqlCommand command = new MySqlCommand("SELECT * FROM `stages` WHERE
`project` = @CUI", db.getConnection());
    command.Parameters.Add("@CUI", MySqlDbType.Int32).Value = id;

    adapter.SelectCommand = command;
    adapter.Fill(table);

    return table;
}
```

Листинг 3.7 – метод GetProjectStages()

При удалении проекта вызывается метод Delete(), код которого представлен на листинге 3.8. Метод посылает несколько запросов к базе данных, удаляя строку проекта из таблицы projects, все строки все этапов проекта из таблицы stages, все строки пар пользователь-проект из таблицы involved, все задачи проекта из таблицы tasks.

```
public void Delete()
{
    foreach (DataRow taskRow in GetProjectTasks().Rows)
    {
        Task task = new Task(taskRow);
        task.Delete();
    }
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("DELETE FROM `involved` WHERE
`project` = @projectID", db.getConnection());
    command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    command = new MySqlCommand("DELETE FROM `stages` WHERE `project` =
@projectID", db.getConnection());
    command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
    command.ExecuteNonQuery();
    command = new MySqlCommand("DELETE FROM `projects` WHERE `id` =
@projectID", db.getConnection());
    command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
    command.ExecuteNonQuery();
}
```

```

        command = new MySqlCommand("DELETE FROM `labels` WHERE `project` =
@projectID", db.getConnection());
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
        command.ExecuteNonQuery();
        db.closeConnection();
    }
}

```

Листинг 3.8 – метод Delete()

3.1.4. Класс Stage

Данный класс содержит поля id, name, author и project, они являются публичными и строковыми. Данные заполняются либо напрямую, либо через собственный запрос из конструктора, если в него передано только строковое значение id, либо из значений строки таблицы, если в него передан объект типа DataRow. Код конструкторов представлен на листинге 3.9.

```

public Stage(MainForm main, ProjectForm parentProject, string name, string
author, string id)
{
    this.author = author;
    this.name = name;
    this.mainProj = main;
    this.parentProject = parentProject;
    this.id = id;
}

public Stage(string id)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `stages` WHERE
'id' = @sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    DataTable table = new DataTable();
    adapter.Fill(table);

    foreach (DataRow row in table.Rows)
    {
        this.id = row["id"].ToString();
        this.name = row["name"].ToString();
        this.author = row["author"].ToString();
        this.project = row["project"].ToString();
    }
}

public Stage(DataRow row)
{
    this.id = row["id"].ToString();
    this.name = row["name"].ToString();
    this.author = row["author"].ToString();
    this.project = row["project"].ToString();
}

```

Листинг 3.9 – конструкторы Stage

У класса Stage, как и у класса Project есть метод, создающий карточку этапа, состоящий из инициализации графических элементов, их параметров и привязанных событий. Код этого метода приведен в приложении А.4.

Карточки этапов участвуют в механике DragAndDrop, являясь целевыми объектами для переносимых курсором карточек задач. За это отвечают методы Stage_DragEnter() и Stage_DragOver(), код которых приведен на листинге 3.10. Первый метод вызывается при входе перетаскиваемого элемента на область, которая принимает перетаскивание. Если перетаскиваемый элемент представляет собой панель, то устанавливается эффект перетаскивания DragDropEffects.Move, указывающий на возможность перемещения элемента. Второй метод вызывается во время перемещения перетаскиваемого элемента над областью, которая принимает перетаскивание. Если все необходимые условия выполняются, то панель taskPanel добавляется в список контроллеров целевого столбца, что фактически перемещает панель из одного столбца в другой.

```
private void Stage_DragEnter(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(typeof(Panel)))
    {
        e.Effect = DragDropEffects.Move;
    }
}
private void Stage_DragOver(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(typeof(Panel)))
    {
        Panel taskPanel = (Panel)e.Data.GetData(typeof(Panel));
        Panel targetColumn = (Panel)sender;

        if (taskPanel.Parent != targetColumn)
        {
            targetColumn.Controls.Add(taskPanel);
        }
    }
}
```

Листинг 3.10 – методы Stage_DragEnter() и Stage_DragOver()

За удаление этапа из базы данных отвечает метод Delete(), его код представлен на листинге 3.11. Он удаляет строку, содержащую удаляемую колонку из таблицы stages и все строки задач, принадлежащих этапу из таблицы tasks.


```

public void Delete()
{
    DB db = new DB();

    MySqlCommand command = new MySqlCommand("DELETE FROM `tasks` WHERE
`stage` = @sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    command = new MySqlCommand("DELETE FROM `stages` WHERE `id` = @sI",
db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
    command.ExecuteNonQuery();
    db.closeConnection();
}

```

Листинг 3.11 – метод Delete()

Также класс содержит метод GetStageTasks(), который возвращает таблицу всех задач проекта, делая запрос в базу данных. Код метода представлен на листинге 3.12.

```

public DataTable GetStagesTasks()
{
    DB db = new DB();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();

    MySqlCommand command = new MySqlCommand("SELECT * FROM `tasks` WHERE
`stage` = @tS", db.getConnection());
    command.Parameters.Add("@tS", MySqlDbType.Int32).Value = id;

    adapter.SelectCommand = command;
    adapter.Fill(table);

    return table;
}

```

Листинг 3.12 – метод GetStageTasks()

3.1.5. Класс Task

Класс Task был разработан для более удобного взаимодействия с данными задач и уменьшения количества запросов к базе данных. Класс содержит следующие строковые публичные поля: id, name, stage, responsible, description. А также публичное поле archived типа bool. Все данные заполняются через конструктор, либо напрямую, либо через запрос к базе данных если в конструктор передано только значение id, либо получая данные из строки таблицы, если передан объект типа DataRow. Код конструкторов представлен на листинге 3.13.

```

    public Task(string id, string name, string description, string stage,
DateTime startDate, DateTime finishDate, Stage parentStage = null)
    {
        this.id = id;
        this.name = name;
        this.description = description;
        this.stage = stage;
        this.startDate = startDate;
        this.finishDate = finishDate;
        this.parentStage = parentStage;
    }

    public Task(DataRow row)
    {
        this.id = row["id"].ToString();
        this.name = row["name"].ToString();
        if (row["description"] != null) this.description =
row["description"].ToString();
        this.stage = row["stage"].ToString();
        if (row["startDate"] != null) this.startDate =
DateTime.Parse(row["startDate"].ToString());
        if (row["startDate"] != null) this.finishDate =
DateTime.Parse(row["finishDate"].ToString());
        if (row["responsible"] != null) this.responsible =
row["responsible"].ToString();
        this.archived = Convert.ToBoolean(row["archived"]);
    }

    public Task(string id)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `tasks` WHERE
`id` = @tI", db.getConnection());
        command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        adapter.SelectCommand = command;
        DataTable table = new DataTable();
        adapter.Fill(table);

        foreach (DataRow row in table.Rows)
        {
            this.id = row["id"].ToString();
            this.name = row["name"].ToString();
            if (row["description"] != null) this.description =
row["description"].ToString();
            this.stage = row["stage"].ToString();
            if (row["startDate"] != null) this.startDate =
DateTime.Parse(row["startDate"].ToString());
            if (row["startDate"] != null) this.finishDate =
DateTime.Parse(row["finishDate"].ToString());
            if (row["responsible"] != null) this.responsible =
row["responsible"].ToString();
            this.archived = Convert.ToBoolean(row["archived"]);
        }
    }
}

```

Листинг 3.13 – конструкторы Task

У класса Tasks, как и у класса Stage есть метод, создающий карточку задачи, состоящий из инициализации графических элементов, их параметров и привязанных событий. Отличительной особенностью является переменная

типа bool, передаваемая на вход методу, она определяет разрешена ли механика DragAndDrop для карточки (если карточка отображается на главной странице или вкладке календаря механика запрещена). Код этого метода приведен в приложении А.5.

В событии перетаскивания карточки участвует метод TaskPanel_MouseDown(), он активируется когда на карточке задачи нажимается левая кнопка мыши. Код метода представлен на листинге 3.14.

```
private void TaskPanel_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButton.Left)
    {
        Panel taskPanel = (Panel)sender;
        taskPanel.DoDragDrop(taskPanel, DragDropEffects.Move);
    }
}
```

Листинг 3.14 – метод TaskPanel_MouseDown()

Часть методов класса отвечает за получение элементов, связанных с задачей. Так метод GetTaskTags() возвращает таблицу, заполненную всеми метками, присвоенными задаче. Код метода представлен на листинге 3.15. Аналогично метод GetAttachment() возвращает таблицу вложений задачи, его код представлен в приложении А.5.

```
public DataTable GetTaskTags()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `label` FROM `task-
label` WHERE `task` = @tI", db.getConnection());
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    db.openConnection();
    adapter.Fill(table);
    db.closeConnection();
    return table;
}
```

Листинг 3.15 – метод GetTaskTags()

3.1.6. Класс Tag

Данный класс отвечает за хранение данных о метках и выполнений действий с ними. Аналогично столбцам таблицы labels базы данных, класс содержит публичные строковые поля id, project, name и color. Данные заполняются через конструктор либо напрямую, либо через запрос в базу

данных по id, либо из таблицы. Код конструкторов представлен на листинге 3.16.

```
public Tag(string id, string name, string color, string project)
{
    this.id = id;
    this.name = name;
    this.color = color;
    this.project = project;
}
public Tag(DataRow labelRow)
{
    id = labelRow["id"].ToString();
    name = labelRow["name"].ToString();
    color = labelRow["color"].ToString();
    project = labelRow["project"].ToString();
}
public Tag(string id)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `labels` WHERE
`id` = @LI", db.getConnection());
    command.Parameters.Add("@LI", MySqlDbType.Int32).Value = id;
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    DataTable table = new DataTable();
    adapter.Fill(table);

    foreach (DataRow row in table.Rows)
    {
        this.id = id;
        this.name = row["name"].ToString();
        this.color = row["color"].ToString();
        this.project = row["project"].ToString();
    }
}
```

Листинг 3.16 – конструкторы Tag

Метод GetTasksByTag() позволяет получить из базы данных таблицу, заполненную данными всех задач, содержащих данную метку, его код представлен на листинге 3.17.

```
public DataTable GetTasksByTag()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `task` FROM `task-label`
WHERE `label` = @labelID", db.getConnection());
    command.Parameters.Add("@labelID", MySqlDbType.Int32).Value = id;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    db.openConnection();
    adapter.SelectCommand = command;
    adapter.Fill(table);
    db.closeConnection();
    return table;
}
```

Листинг 3.17 – метод GetTasksByTag()

Метод RemoveFromTask() удаляет присвоенную задаче метку, обновляет данные в базе данных с помощью запроса, а затем обновляет все открытые окна, которые содержат задачу. Код метода представлен на листинге 3.18.

```
private void RemoveFromTask()
{
    TaskForm openedTF =
Application.OpenForms.OfType<TaskForm>().FirstOrDefault();
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("DELETE FROM `task-label` WHERE
`task` = @t AND `label` = @l", db.getConnection());
    command.Parameters.Add("@t", MySqlDbType.Int32).Value =
openedTF.task.id;
    command.Parameters.Add("@l", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
    openedTF.UpdateTags();

    if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
}
```

Листинг 3.18 – метод RemoveFromTask()

Метод GetColor() преобразовывает численное значение цвета метки, хранящееся в базе данных к объекту типа Color, которое можно использовать для назначения цвета панелей. Код метода представлен на листинге 3.19.

```
public Color GetColor()
{
    Color tagColor = Color.FromArgb(Convert.ToInt32(color));
    return tagColor;
}
```

Листинг 3.19 – метод GetColor()

Также класс Tag содержит несколько методов, создающих панели метки. Метод GetBigLabelTagPreview() создает и заполняет элементами панель метки, которая отображается в окне задачи, метод GetSmallTag() создает панель, отображаемую в окне проекта на карточках задач, а метод GetLabelPanel() создает панель, которую пользователь видит в окне проекта на вкладке метки проекта. Код данных методов достаточно объемн, поэтому представлен в приложении А.6.

3.1.7. Класс Notification

Класс Notification содержит поля, аналогичные столбцам таблицы базы данных notification: id, receiver, message, sender, project и received. Данные заполняются с помощью конструктора который принимает на вход значение id уведомления. Код конструктора представлен на листинге 3.20.

```
public Notification(string id)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `notification`
WHERE `id` = @notID", db.getConnection());
    command.Parameters.Add("@notID", MySqlDbType.Int32).Value = id;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    db.openConnection();
    adapter.Fill(table);
    db.closeConnection();
    this.id = id;
    foreach (DataRow row in table.Rows)
    {
        this.receiver = row["receiver"].ToString();
        this.message = row["message"].ToString();
        this.sender = row["sender"].ToString();
        this.project = row["project"].ToString();
        this.received = Convert.ToBoolean(row["received"]);
    }
}
```

Листинг 3.20 – конструктор Notification

Метод GetInvitationPanel создает панель уведомления приглашенного пользователя, которая добавляется на панель уведомлений личного кабинета. Большая часть метода отвечает за инициализацию графических элементов и задачу их свойств и событий. События графических элементов обеспечивают логику работы приглашения участников в проект. После приглашения нового участника в базе данных создается новая строка уведомления, которая загружается в виде панели в личном кабинете приглашенного пользователя. При нажатии кнопки «Принять» столбец receiver изменяет свое значение и в таблицу involved добавляется пара участник-проект. У приглашенного пользователя на главной странице теперь будет отображаться проект, в который его пригласили. Код метода представлен в приложении А.8.

За изменение статуса отвечает метод SetReceived(), его код представлен на листинге 3.21.

```

public void SetReceived()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `notification` SET
`received` = @status WHERE `id` = @notID", db.getConnection());
    command.Parameters.Add("@status", MySqlDbType.Int32).Value = 1;
    this.received = true;

    command.Parameters.Add("@notID", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
}

```

Листинг 3.21 – метод SetReceived()

3.1.8. Класс Attachment

Класс Attachment хранит данные о вложении и определяет методы работы с ними. Класс содержит поля, аналогичные столбцам таблицы attachments базы данных: id, name, tasks, data. Первые 3 имеют строковый тип данных, последнее является массивом байтов. Данные заполняются через конструктор, принимающий на вход строку таблицы из базы данных. Код конструктора представлен на листинге 3.22.

```

public Attachment(DataRow attachmentRow)
{
    this.id = attachmentRow["id"].ToString();
    this.name = attachmentRow["name"].ToString();
    this.task = attachmentRow["task"].ToString();
    this.data = (byte[])attachmentRow["data"];
}

```

Листинг 3.22 – конструктор Attachment

За создание превью вложения, отображающееся на панели вложений окна задачи, отвечает метод GetAttachmentPreview(). Код метода представлен в приложении А. На превью помимо названия вложения находится кнопка загрузки, при нажатии на нее вызывается метод Save_Click(). Он вызывает всплывающее окно файловой системы и предоставляет пользователю возможность выбрать путь сохранения вложения. Код метода представлен на листинге 3.23.

```

private void Save_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog = new SaveFileDialog();
    saveFileDialog.Filter = "All Files (*.*)|*.*";
    saveFileDialog.RestoreDirectory = true;
    saveFileDialog.FileName = name;
    if (saveFileDialog.ShowDialog() == DialogResult.OK)

```

```
{  
    string filePath = saveFileDialog.FileName;  
    File.WriteAllBytes(filePath, data);  
}
```

Листинг 3.23 – метод Save_Click()

3.1.9. Класс TagBoxElem

Класс TagBoxElem состоит из 2 полей: id и name. Из экземпляров этого класса состоят все списки проекта, используемые в качестве источника элемента ComboBox, когда нужно чтобы у пользователя отображалось только имя сущности, а при ее выборе передавалось значение id. Для этого был переопределен метод ToString(), который возвращает только значение name. Код метода представлен на листинге 3.24.

```
public override string ToString()  
{  
    return name;  
}
```

Листинг 3.24 – переопределение метода ToString()

3.2. Описание работы системы

В данном разделе будут описаны формы, созданные при разработке программы, показан их интерфейс и приведен листинг основных методов с описанием работы. Всего в программе представлено 13 форм-окон:

1. **LoginForm** – окно авторизации;
2. **SignUpForm** – окно регистрации;
3. **MainForm** – окно главной страницы;
4. **UserForm** – окно личного кабинета пользователя;
5. **ChangePassForm** – окно смены пароля;
6. **NewProjectForm** – окно создания нового проекта;
7. **ProjectForm** – окно проекта;
8. **InviteUserForm** – окно приглашения нового участника проекта;
9. **TaskForm** – окно задачи;
10. **AddDateForm** – окно добавления даты;
11. **AddRespForm** – окно делегирования задачи;

- 12. **LabelForm** – окно присвоения метки;
- 13. **NewLabelForm** – окно создания метки.

3.2.1. Форма LoginForm

Форма авторизации - первое окно, которое видит пользователь при запуске программы. Окно состоит из 2 полей: логин и пароль, и 2 кнопок: «Войти» и «Зарегистрироваться», а также верхней панели, которая отвечает за перемещение формы и содержит в себе кнопки закрытия и сворачивания окна. Внешний вид окна представлен на рисунке 3.1.

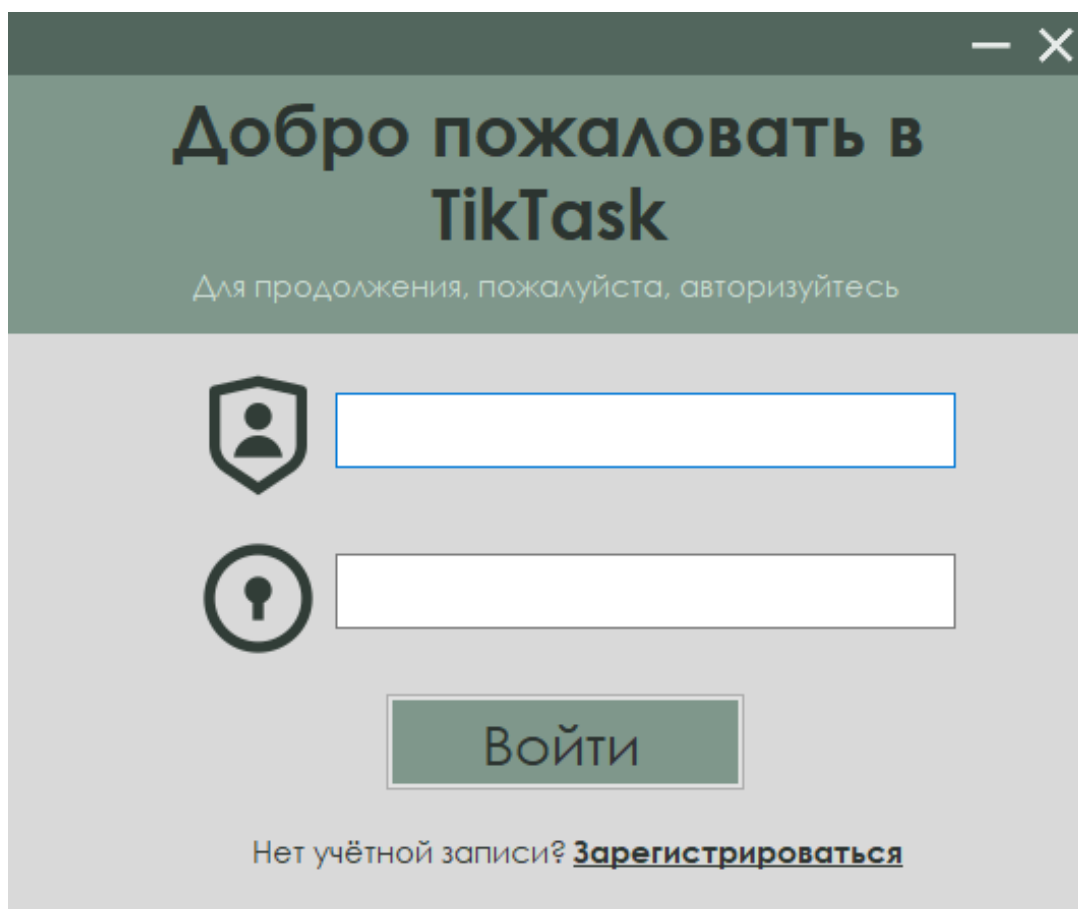


Рисунок 3.1 – окно авторизации

За перемещение окна отвечают методы-события `topPanel_MouseClick()` и `topPanel_MouseMove()`, их код представлен на листинге 3.24. При нажатии левой кнопки мыши в области панели позиция курсора считывается в переменную `clickPoint`, а затем при перемещении мыши новые координаты окна рассчитываются исходя из координат курсора. На этой панели также находятся кнопки сворачивания и закрытия окна, нажатие на которые

вызывает методы `minimizeButton_Click()` и `closeButton_Click()` соответственно. Такие же методы есть у каждого окна программы, которое имеет верхнюю панель, поэтому при рассмотрении остальных форм они упоминаться не будут.

```
private void minimizeButton_Click(object sender, EventArgs e) {
this.WindowState = FormWindowState.Minimized; }
private void closeButton_Click(object sender, EventArgs e) {
Application.Exit(); }
Point clickPoint;
private void topPanel_MouseClick(object sender, MouseEventArgs e)
{
    clickPoint = new Point(e.X, e.Y);
}
private void topPanel_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - clickPoint.X;
        this.Top += e.Y - clickPoint.Y;
    }
}
```

Листинг 3.24 – методы верхней панели

Если пользователь использует программу впервые, ему необходимо пройти регистрацию. Нажатие кнопки «Зарегистрироваться» вызывает метод `signUpButton_Click()`, код которого представлен на листинге 3.25. Окно авторизации скрывается, создается новый экземпляр класса регистрации `SignUpForm`, и форма выводится на экран.

```
private void signUpButton_Click(object sender, EventArgs e)
{
    this.Hide();
    SignUpForm signUpForm = new SignUpForm();
    signUpForm.Show();
}
```

Листинг 3.25 – метод `signUpButton_Click()`

Если аккаунт у пользователя уже есть, он вводит свои логин и пароль и нажимает кнопку «Войти», что вызывает метод `Login_Click()`, код которого представлен на листинге 3.26. Введенные пользователем данные считываются в переменные `login` и `password`, затем формируется запрос к базе данных, выбирающий все строки таблицы «users», где значения столбцов «login» и «password» равны введенным. Запрос возвращает таблицу, после чего происходит проверка на число рядов в ней. Если оно равно нулю, значит пользователя с такими данные не существует, о чем пользователь

уведомляется всплывающим окном с сообщением «Неверный логин или пароль». Если же число рядов не равно нулю, создается экземпляр класса User, хранящий информацию о всех данных пользователя, который передается в конструктор нового экземпляра класса MainForm. Открывается новое окно главной страницы, а окно авторизации скрывается.

```
private void Login_Click(object sender, EventArgs e)
{
    string login = loginField.Text;
    string pass = passField.Text;

    DB db = new DB();
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE
`login` = @uL AND `password` = @uP", db.getConnection());
    command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = login;
    command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = pass;
    adapter.SelectCommand = command;
    adapter.Fill(table);

    if (table.Rows.Count > 0)
    {
        string i = table.Rows[0][0].ToString();
        string l = table.Rows[0][1].ToString();
        string n = table.Rows[0][3].ToString();
        string s = table.Rows[0][4].ToString();
        string m = table.Rows[0][5].ToString();
        User user = new User(i, n, s, l, m);
        MainForm mainForm = new MainForm(user);
        mainForm.Show();
        this.Hide();
    }
    else
        MessageBox.Show("Неверный логин или пароль");
}
```

Листинг 3.26 – метод Login_Click()

3.2.2. Форма SignUpForm

Окно регистрации состоит из верхней панели, 5 полей для ввода имени, фамилии, логина, пароля и почты, и 2 кнопок: «Зарегистрироваться» и «Назад к авторизации». Внешний вид окна представлен на рисунке 3.2.

При нажатии на кнопку «Назад к авторизации» вызывается метод backToLoginButton_Click(), код которого представлен в листинге 3.27. Окно регистрации скрывается, затем выполняется проверка на наличие скрытой формы авторизации, и либо открывается скрытое окно, либо создается новый экземпляр класса LoginForm.

```

private void backToLoginButton_Click(object sender, EventArgs e)
{
    this.Hide();
    if (Application.OpenForms.OfType<LoginForm>().Count() > 0)
        Application.OpenForms.OfType<LoginForm>().FirstOrDefault().Show();
    else
    {
        LoginForm loginForm = new LoginForm();
        loginForm.Show();
    }
}

```

Листинг 3.27 – метод backToLoginButton_Click()

Рисунок 3.2 – окно регистрации

При нажатии на кнопку «Зарегистрироваться» вызывается метод Register_Click(), код которого представлен на листинге 3.28. Сначала выполняется ряд проверок: заполнены ли обязательные поля и уникальны ли логин и почта, введенные пользователем. При нарушении хотя бы одного условия, пользователю выводится соответствующее сообщение об ошибке. Проверки уникальности логина и почты вынесены в отдельные методы IsUserExist() и IsMailExist(), и представляют собой запрос к базе данных, аналогичный проверке данных при авторизации.

```

private void Register_Click(object sender, EventArgs e){
    if (loginField_SA.Text == "" || passField_SA.Text == "" ||
        mailField_SA.Text == "" )

```

```

{
    MessageBox.Show("Пожалуйста, заполните обязательные поля");
    return;
}
if (IsUserExist() || IsMailExist()) { return; }
User newUser = new User();
if (newUser.AddUser(loginField_SA, passField_SA, nameField_SA,
surnameField_SA, mailField_SA))
{
    MessageBox.Show("Аккаунт создан.");
    this.Hide();
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
}
else
    MessageBox.Show("Что-то пошло не так");
}

```

Листинг 3.28 – метод Register_Click()

Если все проверки пройдены успешно, создается новый экземпляр класса User, заполненный введенными данными через конструктор, и у него вызывается метод AddUser(), формирующий запрос к базе данных, добавляющий нового пользователя. Затем форма регистрации скрывается, и открывается окно авторизации.

3.2.3 Форма MainForm

После успешного прохождения авторизации перед пользователем появляется главная форма, внешний вид которой представлен на рисунке 3. Окно состоит из верхней панели, панели пользователя, панели приветствия, панели проектов и панели задач.

У класса формы есть 1 переменная curUser класса User, которая передается из формы авторизации через конструктор. Код конструктора представлен на листинге 3.

```

public User curUser;
public MainForm(User curUser) {
    this.curUser = curUser;
    InitializeComponent();
    greetingLabel.Text = Greeting();
    loginLabelMain.Text = curUser.Login;
}

```

```

        dataLabel.Text = $"{DateTime.Now.ToString("dddd")},
{DateTime.Now.ToString("M")}"
        UpdateTasks();
        UpdateProjects();
    }

```

Листинг 3 – конструктор MainForm

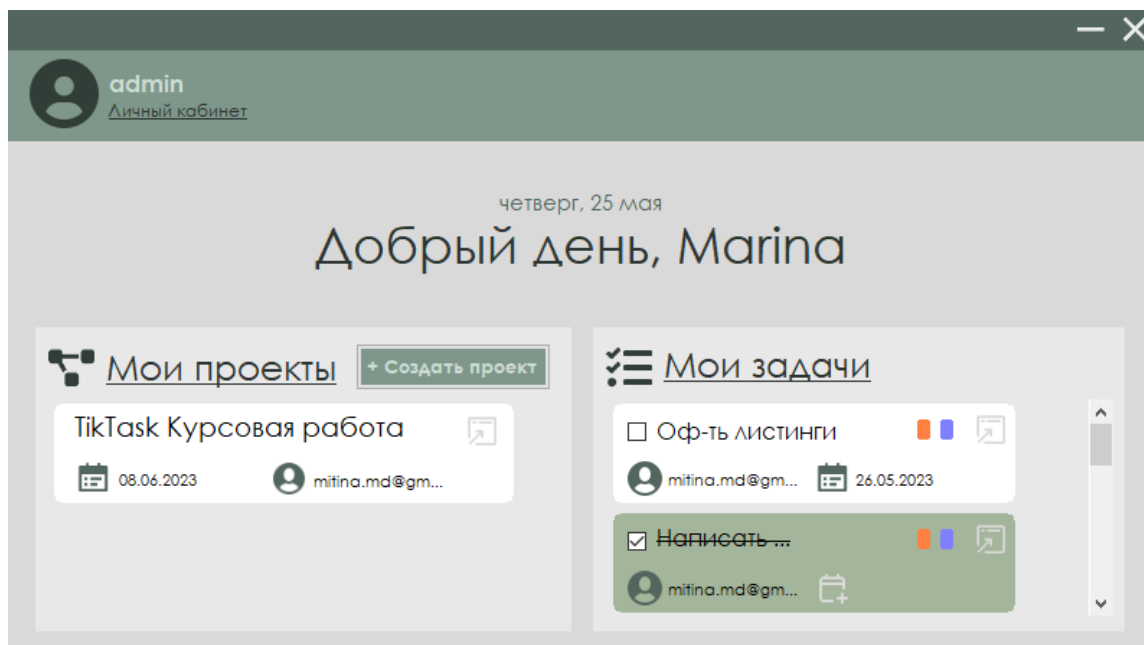


Рисунок 3.3 – главное окно

Помимо этого, в конструкторе присваивается значение тексту приветствия в зависимости от времени суток (генерация приветствия вынесена в отдельный метод Greeting(), представленный на листинге 3.29), тексту логина пользователя, тексту текущей даты.

```

private string Greeting()
{
    string name;
    if (curUser.name != "") { name = curUser.name; }
    else name = curUser.login;
    int time = Convert.ToInt32(DateTime.Now.ToString("HH"));
    if (time <= 4)
        return $"Доброй ночи, {name}";
    else if (time <= 11)
        return $"Доброе утро, {name}";
    else if (time <= 17)
        return $"Добрый день, {name}";
    else return $"Добрый вечер, {name}";
}

```

Листинг 3.29 – метод Greeting()

Также в конструкторе вызываются 2 метода UpdateTasks() и UpdateProjects(). Они отвечают за заполнение панелей проектов и задач. В

методе заполнения проектов создается объект типа `DataTable`, которому присваивается возвращаемое значение метода `GetUserProjects()` класса `User`. Объект представляет собой таблицу, заполненную строками проектов, в которых задействован пользователь. Для каждой строки создается экземпляр класса `Project` и вызывается метод `GetTaskPreviewPanel()`, который возвращает карточку проекта. Карточка добавляется на панель проектов. Метод заполнения панели задач работает аналогично. Код обоих методов представлен на листинге 3.30.

```
public void UpdateTasks()
{
    tasksFlowLayoutPanel.Controls.Clear();
    DataTable curUserTasks = curUser.GetUserTasks();
    foreach (DataRow task in curUserTasks.Rows)
    {
        Task userTask = new Task(task);
        userTask.GetTaskPreviewPanel();

tasksFlowLayoutPanel.Controls.Add(userTask.GetExistingTaskPreviewPanel(false));
    }
}
public void UpdateProjects()
{
    projectFlowLayout.Controls.Clear();
    DataTable curUserProjects = curUser.GetUserProjects();
    foreach (DataRow row in curUserProjects.Rows)
    {
        Project userProject = new Project(row["project"].ToString());
        projectFlowLayout.Controls.Add(userProject.GetPreviewCard());
    }
}
```

Листинг 3.30 – методы `UpdateTasks()` и `UpdateProjects()`

При нажатии на кнопку «Личный кабинет» вызывается метод `userAccountButton_Click()`, он создает экземпляр класса `UserForm`, показывает его форму и скрывает окно главной страницы. Код метода представлен на листинге 3.31.

```
private void userAccountButton_Click(object sender, EventArgs e)
{
    UserForm userForm = new UserForm(curUser);
    userForm.Show();
    this.Hide();
}
```

Листинг 3.31 – метод `userAccountButton_Click()`

При нажатии на кнопку «Создать проект» вызывается метод `createProjectButton_Click()`. Он проверяет существует ли открытая форма

создания проекта, и, если нет, создает и выводит новую. Код метода представлен на листинге 3.32.

```
private void createProjectButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<CreateProjectForm>().FirstOrDefault()
    != null) return;
    CreateProjectForm createProjectForm = new CreateProjectForm(this);
    createProjectForm.Show();
}
```

Листинг 3.32 – метод createProjectButton_Click()

3.2.4. Форма UserForm

Форма UserForm представляет личный кабинет пользователя, где он может изменять свои данные, просматривать уведомления, а также выйти из аккаунта. Внешний вид формы представлен на рисунке 3.4.

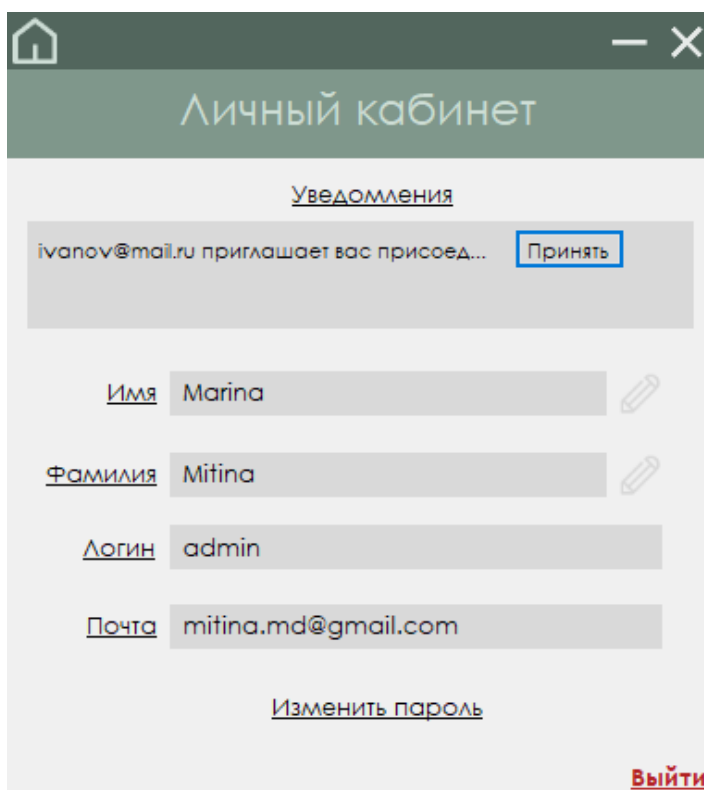


Рисунок 3.4 – окно личного кабинета

Форма состоит из верхней панели, панели уведомлений, 4 полей, содержащих имя, фамилию, логин и почту пользователя, 2 кнопок для изменения имени и фамилии, кнопки «Изменить пароль» и кнопки «Выйти».

Класс формы содержит 1 поле curUser типа User, значение которой присваивается через конструктор. Код конструктора представлен на листинге 3.33


```

User curUser;
public UserForm(User curUser)
{
    this.curUser = curUser;
    InitializeComponent();
    foreach (DataRow notRow in curUser.GetUserNotifications().Rows)
    {
        Notification not = new Notification(notRow["id"].ToString());
        if (not.received == false)
        {
            notifFlowLayout.Controls.Add(not.GetInvitationNotificationPanel());
        }
    }
    nameLabel.Text = curUser.name;
    surnameLabel.Text = curUser.surname;
    loginLabel.Text = curUser.login;
    mailLabel.Text = curUser.mail;
}

```

Листинг 3.33 – конструктор UserForm

Также в конструкторе происходит заполнение панели уведомлений. Вызываемый метод GetUserNotifications() извлекает из базы данных все уведомления текущего пользователя, для каждого уведомления создается экземпляр класса Notification(), и, если уведомление еще не было отмечено как полученное, создается его карточка. Карточка добавляется на панель уведомлений.

Кроме того, в конструкторе присваиваются значения текста всех полей формы, значения достаются из экземпляра класса User, созданного на этапе авторизации.

При необходимости изменить имя или фамилию, пользователь может воспользоваться соответствующими кнопками. Нажатие на них активирует один из методов editNameButton_Click() или editSurnameButton_Click(). Алгоритм их работы идентичен, поэтому будет представлен код только первого метода. Он представлен на листинге 3.34.

```

private void editNameButton_Click(object sender, EventArgs e)
{
    TextBox nameField = new TextBox();
    nameField = new TextBox();
    nameField.Size = namePanel.Size;
    nameField.Location = namePanel.Location;
    nameField.Text = curUser.name;
    nameField.Font = new Font("Century Gothic", 10);
    nameField.KeyPress += (sender2, e2) =>
    {
        if (e2.KeyChar == (char)Keys.Enter)
        {
            curUser.ChangeName(nameField.Text);
        }
    }
}

```

```

        Controls.Remove(nameField);
        Controls.Add(namePanel);
        editNameButton.Click += editNameButton_Click;
        Update();
    }
};
editNameButton.Click -= editNameButton_Click;
Controls.Remove(namePanel);
Controls.Add(nameField);
}

```

Листинг 3.34 – метод editNameButton_Click()

При нажатии на кнопку изменения поля создается новый экземпляр класса `TextBox`, ему задается размер, местоположение и шрифт, а также добавляется анонимный метод, реагирующий на нажатие клавиши `Enter`. Затем текущее имя скрывается и заменяется на активное поле, куда пользователь может ввести новые данные. Они сохранятся при нажатии клавиши `Enter`, с помощью метода `ChangeName()` класса `User`. После обновления данных происходит обратная замена, активное поле скрывается, а текст с новыми данными добавляется обратно на форму.

Также пользователь может при желании изменить пароль, нажав на кнопку «Изменить пароль» и вызвав метод `changePasswordButton_Click()`. Изменение пароля происходит в отдельном окне, поэтому метод проверяет приложение на наличие уже открытых форм изменения пароля. Если их нет, создает новую и выводит ее на экран. Код метода представлен на листинге 3.35.

```

private void changePasswordButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<ChangePass>().FirstOrDefault() != null)
        return;
    ChangePass changePassForm = new ChangePass(curUser);
    changePassForm.Show();
}

```

Листинг 3.35 – метод changePasswordButton_Click()

Также в окне личного кабинета пользователь может закончить текущую сессию, выйдя из аккаунта. За это отвечает метод `endSessionButton_Click()`, вызываемый нажатием кнопки «Выйти». Метод закрывает все открытые окна и выводит новую форму авторизации. Код представлен на листинге 3.36.

```
private void endSessionButton_Click(object sender, EventArgs e)
{
    LoginForm loginForm = new LoginForm();
    loginForm.Show();
    (Application.OpenForms.OfType<MainForm>().FirstOrDefault()).Close();
    (Application.OpenForms.OfType<UserForm>().FirstOrDefault()).Close();
    if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() != null)
        Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().Close();
}
```

Листинг 3.36 – метод endSessionButton_Click()

3.2.5. Форма CreateProjectForm

Окно нового проекта появляется при нажатии кнопки «Создать проект». Оно состоит из верхней панели, поля для названия, 2 выпадающих календарей для выбора сроков и кнопки «Создать». Внешний вид окна представлен на рисунке 3.5.

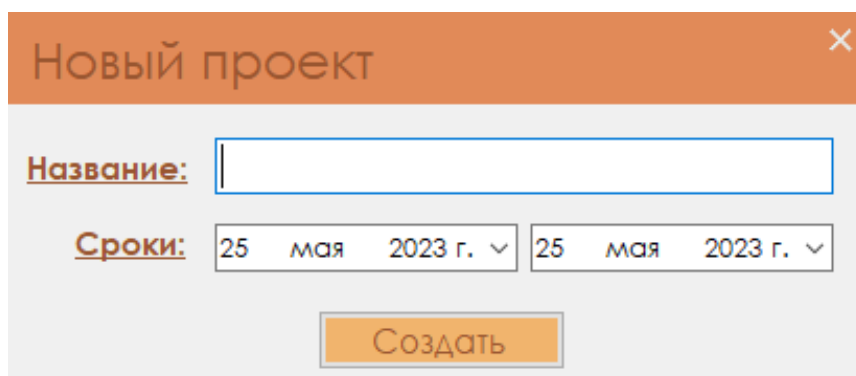


Рисунок 3.5 – окно создания проекта

Конструктор формы принимает на вход объект главной страницы, для её обновления после добавления проекта, а также устанавливает точку появления окна, которая принимает значение местоположения курсора. Код конструктора представлен на листинге 3.37.

```
MainForm main;
public CreateProjectForm(MainForm main)
{
    this.main = main;
    InitializeComponent();
    Location = new Point(Cursor.Position.X, Cursor.Position.Y); }
}
```

Листинг 3.37 – конструктор CreateProjectForm()

При нажатии на кнопку «Создать» вызывается метод createProjectButton_Click(), он создает экземпляр класса Project, заполняя его данными из объектов формы. Затем вызывается метод PushToDB(), который

добавляет проект в базу данных, и обновляется главная страница. Код метода представлен на листинге 3.38.

```
private void createProjectButton_Click(object sender, EventArgs e)
{
    Project newProject = new Project(main, nameField.Text,
        startDatePicker.Value, finishDatePicker.Value, main.curUser.id);
    newProject.PushToDB();
    main.UpdateProjects();
    this.Close();
}
```

Листинг 3.38 – createProjectButton_Click()

3.2.6. Форма ProjectForm

При нажатии на кнопку открытия проекта вызывается окно проекта. Оно состоит из верхней панели, панели редактирования проекта, и рабочей области, разделенной на 3 вкладки: «Доска Kanban», «Календарь» и «Метки проекта». Внешний вид окна с первой открытой вкладкой представлен на рисунке 3.6

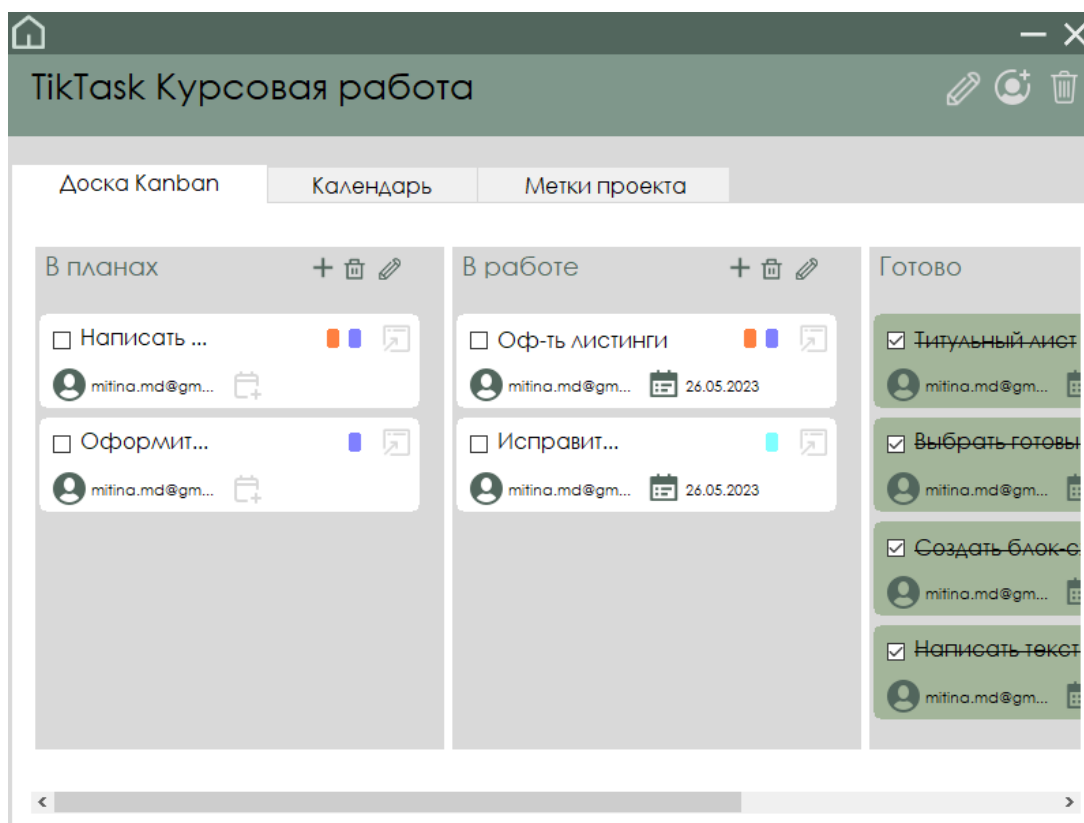


Рисунок 3.6 – окно проекта. Вкладка «Доска Kanban»

Конструктор класса принимает на вход экземпляр класса Project, который присваивается полю curProject, затем, получая из него данные о проекте, изменяет текст, отвечающий за названия проекта. Также конструктор

отвечает за заполнение вкладки рабочей области, открываемой по умолчанию – Kanban доски. Сначала вызывается метод `GetProjectStages()`, возвращающий таблицу всех этапов проекта. Затем для каждой строки создается экземпляр класса `Stage`, и с помощью метода `GetStagePanel()` для каждого этапа генерируется панель, которая добавляется на рабочую область. Таким же образом для каждого этапа вызывается метод `GetStageTasks()`, и создаются экземпляры класса `Task`. На панели этапа добавляются панели карточек. Код конструктора представлен на листинге 3.39.

```
public Project curProject;
public ProjectForm(Project project)
{
    InitializeComponent();
    this.curProject = project;
    nameLabel.Text = curProject.name;
    stagesPanel.Controls.Remove(addStageButton);
    foreach (DataRow stage in curProject.GetProjectsStages().Rows)
    {
        Stage new_stage = new Stage(stage);
        FlowLayoutPanel new_stagePanel =
            new_stage.GetStagePanel(stagesPanel.Height - 50);
        foreach (DataRow task in new_stage.GetStagesTasks().Rows)
        {
            Task new_task = new Task(task);
            Panel new_taskPanel =
                new_task.GetExistingTaskPreviewPanel(true);
            new_stagePanel.Controls.Add(new_taskPanel);
        }
        stagesPanel.Controls.Add(new_stagePanel);
    }
    stagesPanel.Controls.Add(addStageButton);
}
```

Листинг 3.39 – конструктор `ProjectForm`

Помимо панелей этапов в рабочей области также находится кнопка добавления нового этапа. При нажатии на нее вызывается метод `addStageButton_Click()`. Он убирает кнопку из рабочей области и создает на ее месте новое тестовое поле, в которое пользователь вводит название этапа.

После нажатия клавиши `Enter` создается экземпляр класса `Stage`, данные о нем передаются в базу данных посредством метода `PushToDB()`, а текстовое поле заменяется на полноценную карточку этапа, сгенерированную методом `GetStagePanel()`. Кнопка «Добавить этап» возвращается в рабочую область. Код метода представлен на листинге 3.40.

```

        TextBox stageName;
        private void addStageButton_Click(object sender, EventArgs e) {
            stageName = new TextBox();
            stageName.KeyPress += (sender2, e2) => {
                if (e2.KeyChar == (char)Keys.Enter) {
                    Stage stage = new Stage(curProject.main, this, stageName.Text,
                                            curProject.host, null);
                    stage.PushToDB(stageName);
                    stagesPanel.Controls.Add(stage.GetStagePanel(stagesPanel.Height
- 50));

                    stagesPanel.Controls.Remove(stageName);
                    stagesPanel.Controls.Add(addStageButton);
                    stagesPanel.Controls.Add(addStageButton);
                }
            };
            stageName.Width = 300;
            stageName.Height = 30;
            stageName.Font = new Font("Century Gothic", 12);
            stagesPanel.Controls.Remove(addStageButton);
            stagesPanel.Controls.Add(stageName); }

```

Листинг 3.40 – метод addStageButton_Click()

Панель изменения проекта представлена 3 кнопками. Первая отвечает за изменение названия проекта, вызывает метод editProjectButton_Click() при нажатии, код которого представлен на листинге 3.41. Метод заменяет название проекта на активное поле, в которое пользователь может ввести новое название. При нажатии Enter, вызывается анонимный метод: поле заменяется обратно на уже новое название, а метод editName() класса Project обновляет информацию в базе данных. Также с помощью метода OpenForms получается экземпляр формы главного меню, который обновляется с помощью метода UpdateProject().

```

        private void editProjectButton_Click(object sender, EventArgs e) {
            if (projectPanel.Controls.OfType<TextBox>().Count() != 0) return;
            TextBox editProjectField = new TextBox();
            editProjectField.Size = nameLabel.Size;
            editProjectField.Location = nameLabel.Location;
            editProjectField.Font = nameLabel.Font;
            projectPanel.Controls.Remove(nameLabel);
            projectPanel.Controls.Add(editProjectField);
            editProjectField.KeyPress += (sender2, e2) => {
                if (e2.KeyChar == (char)Keys.Enter){
                    curProject.EditName(editProjectField.Text);
                    nameLabel.Text = editProjectField.Text;
                    projectPanel.Controls.Remove(editProjectField);
                    projectPanel.Controls.Add(nameLabel);
                    MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
                    main.UpdateProjects();}}; }

```

Листинг 3.41 – метод editProjectButton_Click

Еще одна кнопка на панели изменения проекта позволяет пригласить другого пользователя как участника проекта. Нажатие вызывает метод `inviteUserButton_Click()`, код которого представлен на листинге 3.41. Метод создает новый экземпляр формы `InviteUserForm()`, передавая ему данные о текущем пользователе и проекте.

```
private void inviteUserButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<InviteUserForm>().FirstOrDefault() != null)
        return;
    MainForm main = Application.OpenForms.OfType<MainForm>().FirstOrDefault();
    InviteUserForm inviteForm = new InviteUserForm(main.curUser, curProject);
    inviteForm.Show();
}
```

Листинг 3.41 – метод `inviteUserButton_Click()`

Также проект можно удалить, нажав на кнопку удаления проекта. При нажатии вызывается метод `deleteProjectButton_Click()`, его код представлен на листинге 3.42. На экране появляется всплывающее окно с сообщением «Вы уверены, что хотите удалить проект?» и 2 кнопками: ОК и Отменить. При нажатии на кнопку ОК вызывается метод `Delete` у объекта текущего проекта `curProject`, он передает запрос на удаления базе данных. Затем закрывается окно проекта, и открывается главная страница, обновленная методами `UpdateProjects()` и `UpdateTasks()`.

```
private void deleteProjectButton_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить проект?", "Подтверждение удаления", MessageBoxButtons.OKCancel);

    if (result == DialogResult.OK)
    {
        curProject.Delete();
        MainForm main =
            Application.OpenForms.OfType<MainForm>().FirstOrDefault();
        main.UpdateProjects();
        main.UpdateTasks();
        main.Show();
        this.Close();
    }
    else if (result == DialogResult.Cancel)
    {
        return;
    }
}
```

Листинг 3.42 – метод `deleteProjectButton_Click()`

Следующим большим функциональным блоком является рабочее пространство с вкладками. Оно представлено формой `TabPage`, которая

позволяет переключаться между вкладками. На рисунках 3 и 3 представлен внешний вид вкладок «Календарь» и «Метки проекта».

Вкладка «Календарь» представляет собой календарь и 2 панели. На первой панели отображаются задачи, которые должны быть выполнены в выбранный пользователем на календаре день. На второй панели отображаются задачи, которым еще не присвоены сроки.

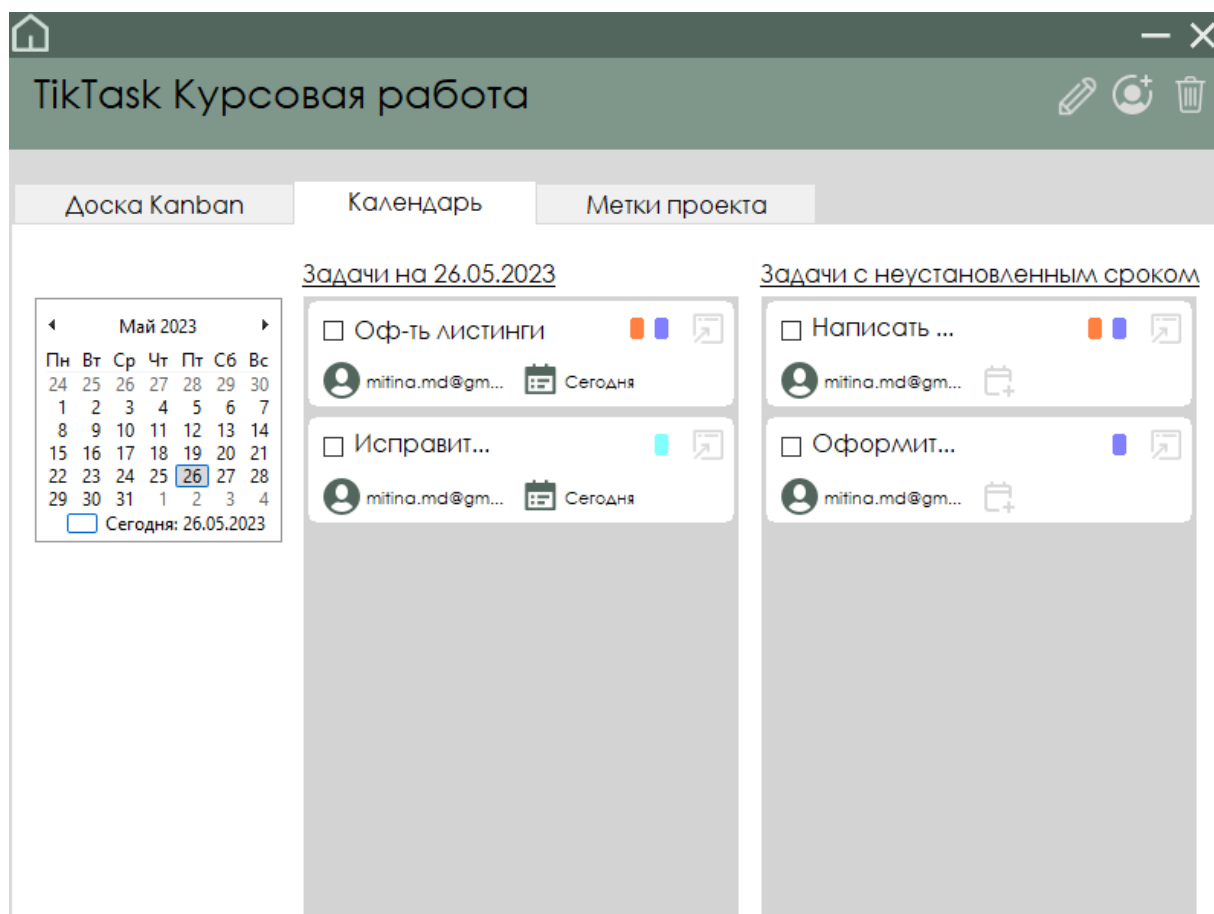


Рисунок 3.7 – окно проекта. Вкладка «Календарь»

Вкладка «Метки проекта» состоит из 2 панелей. На первой перечислены все метки, которые пользователь создавал в рамках проекта. Панели каждой метки содержат чекбокс, при отметке которого на вторую вкладку добавляются задачи, содержащие эту метку. На этой вкладке также можно создать новую метку или удалить ненужные.

За заполнение пространства каждой вкладки актуальной информацией отвечает метод `visualTabs_Selected()`, его код представлен на листинге 3.43. Активная вкладка определяется с помощью конструкции условий. При выборе

вкладки «Календарь» определяется выбранная пользователем дата. Затем обновляется заголовок первой панели и вызывается метод класса User `GetUserTaskByDate()`, который возвращает таблицу, заполненную всеми задачами текущего пользователя, относящимися к переданной дате. Если таблица пуста на панель добавляется текст «На этот день у вас нет активных задач», если же задачи есть, для каждой создается экземпляр класса Task и генерируется карточка с помощью метода `GetExistingTaskPreview()`. Карточки добавляются на панель.

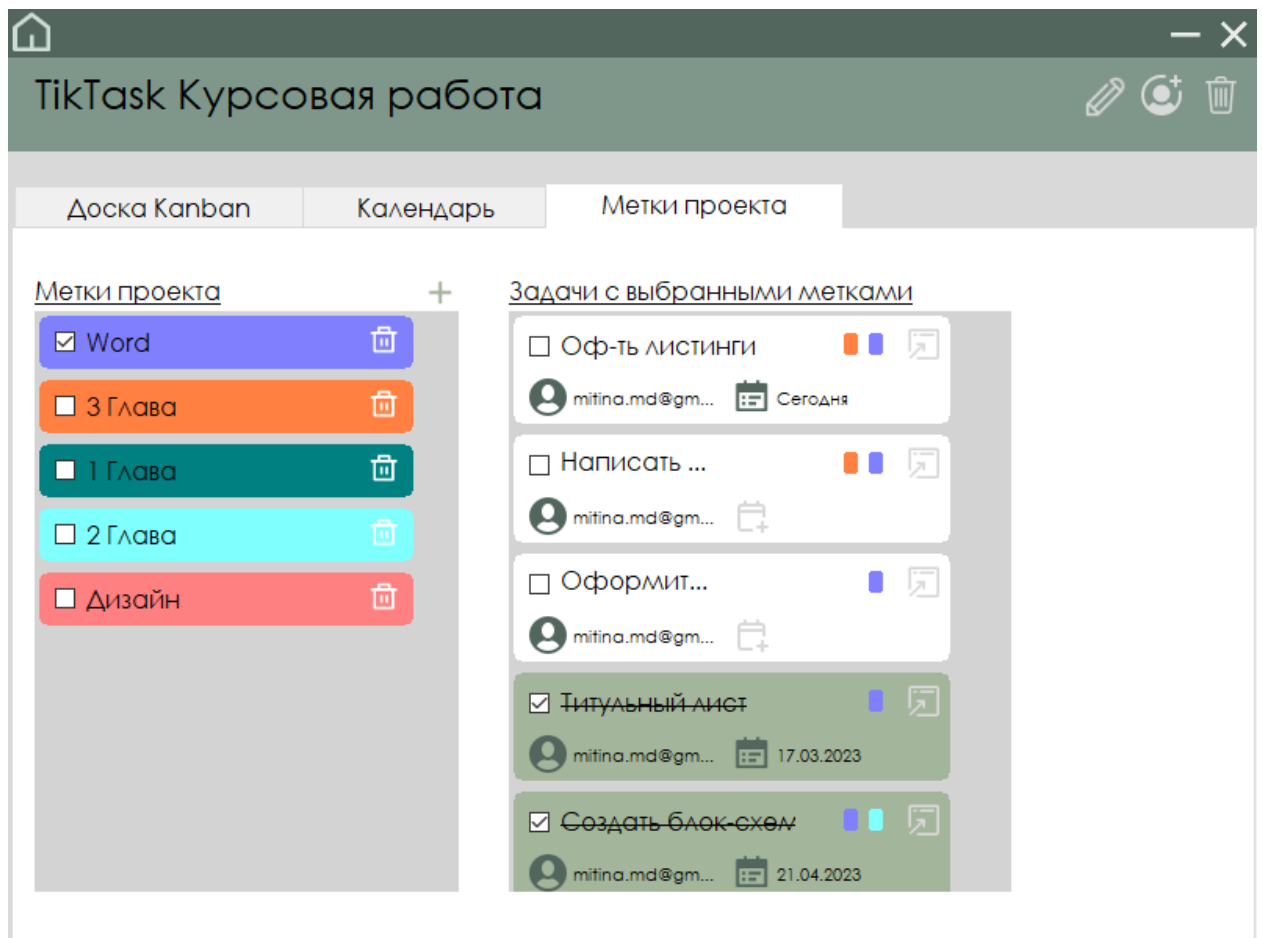


Рисунок 3.8 – окно проекта. Вкладка «Метки проекта»

Если выбрана вкладка «Метки проекта», то вызывается метод класса Project `GetProjectLabels()`, возвращающий таблицу всех меток проекта. Для каждой метки создается экземпляр класса Tag и создается карточка с помощью метода `GetLabelPanel()`. Карточки добавляются на соответствующую панель.

```
private void visualTabs_Selected(object sender, TabControlEventArgs e)
{
    if (visualTabs.SelectedTab == calendarTab)
    {
```

```

        dayTaskFlowLayout.Controls.Clear();
        taskDateLabel.Text = $"Задачи на
            {DateTime.Today.ToString().Split()[0]}";
        MainForm main =
            Application.OpenForms.OfType<MainForm>().FirstOrDefault();
        DataTable tasks = main.curUser.GetUserTasksByDate(DateTime.Today);
        if (tasks.Rows.Count == 0)
            dayTaskFlowLayout.Controls.Add(zeroTaskLabel);
        foreach (DataRow row in tasks.Rows)
        {
            Task newTask = new Task(row);
            dayTaskFlowLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
        }
        noDeadlineTasksLayout.Controls.Clear();
        tasks = main.curUser.GetUserTasksByDate(DateTime.MinValue);
        foreach (DataRow row in tasks.Rows)
        {
            Task newTask = new Task(row);
            noDeadlineTasksLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
        }
    }
    if (visualTabs.SelectedTab == labelsTab )
    {
        tagsFlowLayout.Controls.Clear();
        tasksFlowLayout.Controls.Clear();
        DataTable tags = curProject.GetProjectLabels();
        foreach (DataRow tagRow in tags.Rows)
        {
            Tag tag = new Tag(tagRow);
            tagsFlowLayout.Controls.Add(tag.GetLabelPanel());
        }
    }
}
}
}

```

Листинг 3.43 – метод visualTabs_Selected()

3.2.7. Форма InviteUserForm

Работа над крупными проектами обычно включает в себя командную работу, поэтому важно чтобы проект был доступен для всех его участников. Добавление нового участника вынесено в отдельную форму, которая вызывается нажатием иконки добавления на странице проекта. Форма представляет собой небольшое всплывающее окно с полем и кнопкой «Пригласить». Внешний вид формы показан на рисунке 3.9.

Конструктор формы принимает на вход объекты классов User и Project. Код конструктора представлен на листинге 3.44. Помимо присвоения данных полям класса, конструктор определяет место генерации окна, совпадающее с текущим положением курсора при вызове.

```

User curUser;
Project curProject;
public InviteUserForm(User curUser, Project project)
{

```

```

        this.curUser = curUser;
        this.curProject = project;
        InitializeComponent();
        Location = new Point(Cursor.Position.X - this.Width, Cursor.Position.Y);
    }

```

Листинг 3.44 – конструктор InviteUserForm

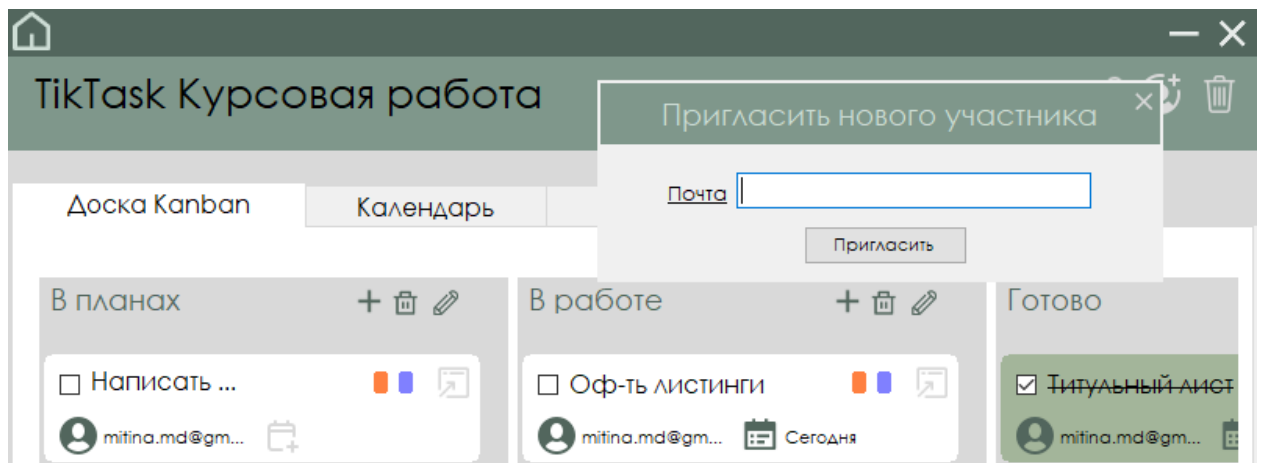


Рисунок 3.9 – окно InviteUserForm

При нажатии кнопки пригласить вызывается метод `inviteButton_Click()`. Он формирует запрос к базе данных, который проверяет существование пользователя с указанной почтой. Если пользователь не найден выводится соответствующее сообщение, если найден, то формируется новый запрос к базе данных, создающий новую строку в таблице «notifications». Запрос заполняет данные об отправителе, получателе, проекте и статусе получения. После успешного выполнения пользователю выводится сообщение «Приглашение отправлено», и окно приглашения скрывается. Код метода представлен на листинге 3.45.

```

private void inviteButton_Click(object sender, EventArgs e)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `id` FROM `users` WHERE
        `mail` = @userMail", db.getConnection());
    command.Parameters.Add("@userMail", MySqlDbType.VarChar).Value =
        userMailField.Text;
    db.openConnection();
    if (command.ExecuteScalar() == null)
    {
        MessageBox.Show("Пользователь не найден");
        db.closeConnection();
    }
    else
    {
        string id = command.ExecuteScalar().ToString();
        db.closeConnection();
        command = new MySqlCommand("INSERT INTO `notification` (`receiver`,
            `message`, `sender`, `received`, `project`) VALUES (@receiver,

```

```

        @message, @sender, @received, @project)", db.getConnection());
        command.Parameters.Add("@receiver", MySqlDbType.Int32).Value = id;
        command.Parameters.Add("@message", MySqlDbType.LongText).Value =
            "{sender} приглашает вас присоединиться к проекту";
        command.Parameters.Add("@sender", MySqlDbType.Int32).Value =
            curUser.id;
        command.Parameters.Add("@received", MySqlDbType.Int32).Value = 0;
        command.Parameters.Add("@project", MySqlDbType.Int32).Value =
            curProject.id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
        MessageBox.Show("Приглашение отправлено");
        this.Close();
    }
}

```

Листинг 3.45 – метод inviteButton_Click()

3.2.8. Форма TaskForm

Каждая карточка задачи на странице проекта имеет кнопку открытия. При нажатии на нее открывается новое окно задачи, где пользователь может посмотреть более подробную информацию о ней и изменить или добавить данные. Внешний вид окна представлен на рисунке 3.10.

Окно представляет собой заголовок с названием задачи и ряд полей с кнопками для их редактирования. Были реализованы следующие поля: исполнитель, сроки, проект, этап, описание, метки и вложения. Также на верхней панели находятся кнопки изменения названия задачи и удаления.

Конструктор формы принимает на вход объект класса Task и заполняет данными из него каждое из полей. Код конструктора представлен на листинге 3.46.

```

public Task task;
Project project;
public TaskForm(Task task)
{
    this.task = task;
    InitializeComponent();
    taskNameLabel.Text = task.name;
    Stage stage = new Stage(task.stage);
    project = new Project(stage.project);
    if (task.responsible != null) label7.Text = task.responsible;
    if (task.startDate != DateTime.MinValue) label12.Text =
        task.startDate.ToString().Split()[0];
    if (task.finishDate != DateTime.MinValue) label13.Text =
        task.finishDate.ToString().Split()[0];
    stageNameLabel.Text = stage.name;
    projectNameLabel.Text = project.name;
    if (task.description != null) descriptionBox.Text = task.description;
    DataTable tags = task.GetTasksTags();
}

```

```

foreach (DataRow tagRow in tags.Rows)
{
    Tag tag = new Tag(tagRow["Label"].ToString());
    Panel tagPanel = tag.GetBigLabelPreview();
    tagsFlowLayout.Controls.Add(tagPanel);
}
UpdateAttachedFiles();
labelForm = new LabelForm(task);
}

```

Листинг 3.46 – конструктор TaskForm

Оф-тЬ ЛИСТИНГИ

Исполнитель: mitina.md@gmail.com

Сроки: 19.05.2023 — 26.05.2023

Проект: TikTask Курсовая работа | Этап: В работе

Описание: Листинг может быть цветной

Метки: Word, 3 Глава

Вложения: Пример оформления третьей главы.pdf

Рисунок 3.10 – окно TaskForm

При нажатии на иконку изменения названия проекта вызывается метод `editNameButton_Click()`, он заменяет название на активное поле, где пользователь может изменить название. Затем после нажатия клавиши `Enter` вызывается метод класса `Task UpdateName()`, обновляющий базу данных и поле заменяется обратно на новое название. Код конструктора представлен на листинге 3.47.

```

private void editNameButton_Click(object sender, EventArgs e)
{
    Controls.Remove(taskNameLabel);
    TextBox taskNameField;
    taskNameField = new TextBox();
}

```

```

taskNameField.Font = new Font("Century Gothic", 16);
taskNameField.Location = new Point(58, 5);
taskNameField.Size = new Size(286, 37);
taskNameField.Text = task.name;
taskNameField.KeyPress += (sender2, e2) =>
{
    if (e2.KeyChar == (char)Keys.Enter)
    {
        task.UpdateName(taskNameField.Text);
        taskNameLabel.Text = task.name;
        Controls.Add(taskNameLabel);
        Controls.Remove(taskNameField);
        if (Application.OpenForms.OfType<ProjectForm>().Count() != 0)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
    }
};
this.Controls.Add(taskNameField);
}

```

Листинг 3.47 – метод editNameButton_Click()

Рядом с кнопкой изменения названия находится кнопка удаления задачи. При нажатии вызывается метод deleteButton_Click(). Его код представлен на листинге 3.48. Метод вызывает новое диалоговое окно для подтверждения удаления, а затем при нажатии кнопки ОК с помощью метода Delete класса Task удаляет задачу и все связанные с ней элементы из базы данных, все открытые окна обновляются.

```

private void deleteButton_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить задачу?", "Подтверждение удаления", MessageBoxButtons.OKCancel);

    if (result == DialogResult.OK)
    {
        task.Delete();
        ProjectForm openedPF =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
        openedPF.UpdateProjectView();
        this.Close();
    }
    else if (result == DialogResult.Cancel)
    {
        return;
    }
}

```

Листинг 3.48 – метод deleteButton_Click()

При нажатии на иконку изменения исполнителя вызывается метод addResponsButton_Click() и открывается новое окно AddRespForm. Код метода представлен на листинге 3.49.

```
private void addResponsButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<AddRespForm>().Count() > 0) return;
    AddRespForm form = new AddRespForm(task, this);
    form.Show();
}
```

Листинг 3.49 – метод addResponsButton_Click()

Аналогичный метод addFinishDateButton_Click() вызывается при нажатии кнопки изменения сроков. Его код представлен на листинге 3.50.

```
private void editFinishDateButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<AddDateForm>().Count() > 0) return;
    AddDateForm form = new AddDateForm(task, this);
    form.Show();
}
```

Листинг 3.50 – метод addFinishDateButton_Click()

При нажатии на иконку изменения этапа вызывается метод editStageButton_Click(). Метод заменяет поле этапа на элемент ComboBox, который представляет собой выпадающий список. Элементы списка являются объектами класса TagBoxElem и создаются после запроса к базе данных. Пользователь выбирает новый этап из выпадающего списка, и после нажатия клавиши Enter обновляются все открытые окна. Код метода представлен на листинге 3.51.

```
private void editStageButton_Click(object sender, EventArgs e)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `project` FROM `stages`
        WHERE `id` = @sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = task.stage;
    db.openConnection();
    object projId = command.ExecuteScalar();
    db.closeConnection();
    Project project = new Project(projId.ToString());
    DataTable stagesTable = project.GetProjectsStages();
    List<TagBoxElem> stagesList = new List<TagBoxElem>();
    foreach (DataRow row in stagesTable.Rows)
    {
        stagesList.Add(new TagBoxElem(row));
    }
    stages = new ComboBox();
    stages.ValueMember = "id";
    stages.Location = new Point(313, 143);
    stages.Size = new Size(117, 24);
    stages.DataSource = stagesList;
    stages.Text = stageNameLabel.Text;
    stages.Font = new Font("Century Gothic", 10);
    Controls.Add(stages);
    stages.BringToFront();
    stages.KeyPress += (sender2, e2) =>
    {
```

```

        if (e2.KeyChar == (char)Keys.Enter)
        {
            task.UpdateStage(((TagBoxElem)stages.SelectedValue).id);
            stageNameLabel.Text = stages.SelectedItem.ToString();
            ProjectForm projectForm =
                Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
            projectForm.UpdateProjectView();
            Controls.Remove(stages);
        }
    };
}

```

Листинг 3.51 – метод editStageButton_Click()

У поля описания задачи есть 2 иконки: первая делает поле активным для редактирования, а вторая сохраняет результат и посылает его в базу данных. Эти действия происходят благодаря методам editDescription_Click() и saveDescription_Click(). Их код представлен на листинге 3.52.

```

private void editDescription_Click(object sender, EventArgs e)
{
    descriptionBox.ReadOnly = false;
    descriptionBox.Enabled = true;
}

private void saveDescription_Click(object sender, EventArgs e)
{
    descriptionBox.ReadOnly = true;
    descriptionBox.Enabled = false;
    task.description = descriptionBox.Text;
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET
`description` = @tD WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("@tD", MySqlDbType.LongText).Value =
descriptionBox.Text;
    command.Parameters.Add("@tI", MySqlDbType.LongText).Value = task.id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
}

```

Листинг 3.52 – методы editDescription_Click() и saveDescription_Click()

За добавление новых меток задаче отвечает иконка на панели меток. При нажатии на нее вызывается метод AddLabel_Click(). Он открывает новое окно добавление метки, если таких открытых окон нет. Код метода представлен на листинге 3.53.

```

private void addLabel_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<LabelForm>().Count() > 0) return;
    labelForm = new LabelForm(task);
    labelForm.Show();
}

```

Листинг 3.53 – метод addLabel_Click()

За работу с вложениями отвечают несколько методов: `addAttachmentButton_Click()`, `SaveFileToDatabase()` и `UpdateAttachedFiles()`. При нажатии на иконку добавления вложения открывается новое диалоговое окно, где пользователь выбирает необходимые для прикрепления файлы. Затем каждый файл преобразуется в массив байтов и, если не превышен допустимый размер - сохраняется в базу данных. Метод `UpdateAttachedFiles()` создает для каждого вложения экземпляр класса `Attachment` и создает его панель, которая добавляется в поле вложений окна. Код методов представлен на листинге 3.54.

```
private void addAttachmentButton_Click(object sender, EventArgs e)
{
    openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Multiselect = true;

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        foreach (string filePath in openFileDialog1.FileNames)
        {
            byte[] fileData = File.ReadAllBytes(filePath);
            SaveFileToDatabase(filePath.Split('\\').Last(), fileData);
        }
        UpdateAttachedFiles();
    }
}

private void SaveFileToDatabase(string fileName, byte[] fileData)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO `attachments`
    (`name`, `data`, `task`) VALUES (@aN, @aD, @aT)", db.getConnection());
    command.Parameters.Add("@aN", MySqlDbType.VarChar).Value = fileName;
    command.Parameters.Add("@aD", MySqlDbType.LongBlob).Value = fileData;
    command.Parameters.Add("@aT", MySqlDbType.Int32).Value = task.id;
    db.openConnection();
    try { command.ExecuteNonQuery();
    catch
    {
        MessageBox.Show("Превышен допустимый размер файла");
    }
    db.closeConnection();
}

private void UpdateAttachedFiles()
{
    attachmentsFlowLayout.Controls.Clear();
    foreach (DataRow attachmentRow in task.GetAttachments().Rows)
    {
        Attachment file = new Attachment(attachmentRow);
        attachmentsFlowLayout.Controls.Add(file.GetAttachmentPreview());
    }
}
```

Листинг 3.54 – методы работы с вложениями

3.2.9. AddDateForm

Всплывающее окно изменения даты может быть вызвано из карточки или окна задачи. Оно представляет собой 2 элемента типа DateTimePicker для выбора начальной и конечной даты и кнопки «Применить». Внешний вид окна представлен на рисунке 3.11.

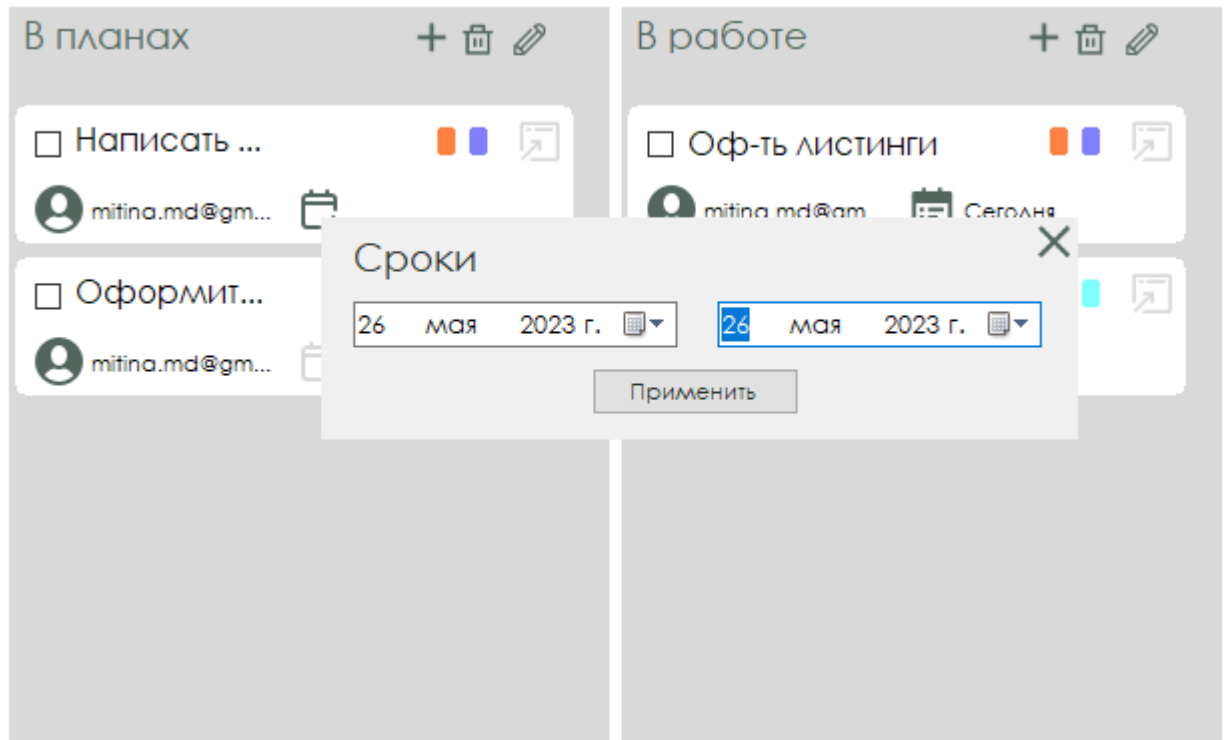


Рисунок 3.11 – окно AddDateForm

Класс формы имеет конструктор с 2 перегрузками. Одна из них принимает на вход только объект редактируемой задачи, а вторая еще открытое окно задачи. Код конструктора представлен на листинге 3.55.

```
public Task task;
public TaskForm taskForm;
public AddDateForm(Task task)
{
    InitializeComponent();
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    this.task = task;
    if (task.startDate != DateTime.MinValue) startTimePicker.Value =
        task.startDate;
    if (task.finishDate != DateTime.MinValue) finishTimePicker.Value =
        task.finishDate;
}
public AddDateForm(Task task, TaskForm taskForm)
{
    InitializeComponent();
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    this.task = task;
    this.taskForm = taskForm;
}
```

```

        if (task.startDate != DateTime.MinValue) startTimePicker.Value =
            task.startDate;
        if (task.finishDate != DateTime.MinValue) finishTimePicker.Value =
            task.finishDate;
    }

```

Листинг 3.55 – конструктор AddDateForm()

При нажатии на кнопку «Применить» вызывается метод `confirmButton_Click()`. Его код представлен на листинге 3.56. Выбранная дата присваивается полям задачи и затем обновляется в базе данных. Если есть открытые окна, на которых отображена задача, они также обновляются. Окно изменения даты закрывается.

```

private void confirmButton_Click(object sender, EventArgs e)
{
    task.startDate = startTimePicker.Value;
    task.finishDate = finishTimePicker.Value;

    DB db = new DB();
    db.openConnection();

    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET `startDate`
        = @tSD WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("tSD", MySqlDbType.Date).Value = task.startDate;
    command.Parameters.Add("tI", MySqlDbType.Int32).Value = task.id;
    command.ExecuteNonQuery();

    command = new MySqlCommand("UPDATE `tasks` SET `finishDate` = @tFD WHERE
        `id` = @tI", db.getConnection());
    command.Parameters.Add("tFD", MySqlDbType.Date).Value = task.finishDate;
    command.Parameters.Add("tI", MySqlDbType.Int32).Value = task.id;
    command.ExecuteNonQuery();
    db.closeConnection();
    task.updateFinishDate();
    if (taskForm != null) taskForm.updateDates();
    if (Application.OpenForms.OfType<MainForm>().FirstOrDefault() != null)
        Application.OpenForms.OfType<MainForm>().FirstOrDefault().updateTasks();
    if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
        Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().updateProjectView(); ;
    this.close();
}

```

Листинг 3.56 – метод ConfirmButton_Click()

3.2.10. AddRespForm

Окно делегирования задачи представляет собой форму с элементом класса `ComboBox` с выпадающим списком и кнопку «Назначить». Внешний вид окна представлен на рисунке 3.12.

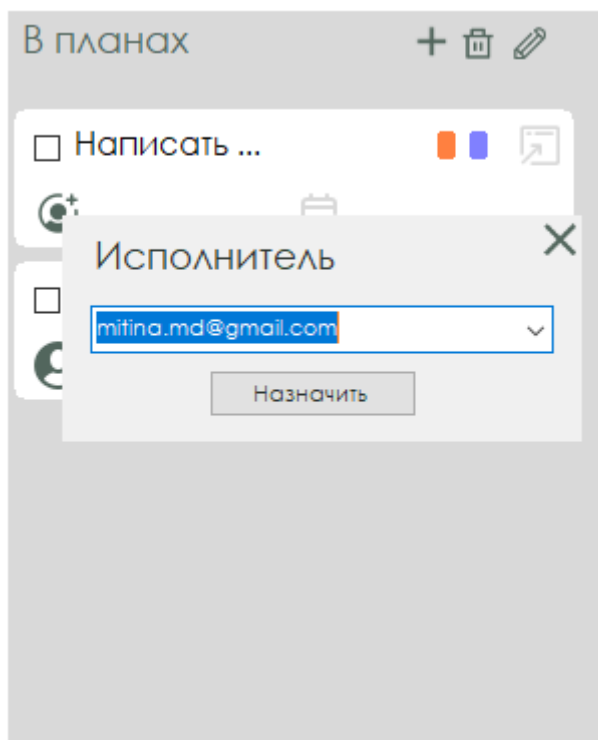


Рисунок 3.12 – окно назначения исполнителя

Как и окно изменения сроков, окно изменения исполнителя так же может вызываться из окна задачи либо карточки задачи в окне проекта, поэтому класс формы так же имеет перегрузку конструктора, он представлен на листинге 3.57.

```
public Task task;
public TaskForm taskForm;
public AddRespForm(Task task)
{
    InitializeComponent();
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    this.task = task;
    responsibleBox.DataSource = GetProjectMembers();
}
public AddRespForm(Task task, TaskForm taskForm)
{
    InitializeComponent();
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    this.task = task;
    this.taskForm = taskForm;
    responsibleBox.DataSource = GetProjectMembers();
}
```

Листинг 3.57 – конструктор AddRespForm()

В конструкторе происходит заполнение выпадающего списка, он состоит из почт всех пользователей, которые участвуют в проекте. Список возвращается методом `GetProjectMembers()`, его код представлен на

листинге 3.58. Сперва из базы данных достается значение проекта задачи, которое передается в новый запрос к таблице «involved», выбирая всех пользователей, подключенных к проекту. Затем для каждого пользователя считывается его почта и добавляется в список. Также добавляется пустое значение, чтобы пользователь мог снять назначение с задачи.

```
private List<string> GetProjectMembers()
{
    List<string> members = new List<string>();
    string stage, project, mail;
    DB db = new DB();

    db.openConnection();

    MySqlCommand command = new MySqlCommand("SELECT `stage` FROM `tasks`
WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = task.id;
    object result = command.ExecuteScalar();
    stage = result.ToString();

    command = new MySqlCommand("SELECT `project` FROM `stages` WHERE `id` =
@sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = stage;
    project = command.ExecuteScalar().ToString();

    command = new MySqlCommand("SELECT * FROM `involved` WHERE `project` =
@pI", db.getConnection());
    command.Parameters.Add("@pI", MySqlDbType.Int32).Value = project;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    adapter.Fill(table);
    foreach (DataRow row in table.Rows)
    {
        command = new MySqlCommand("SELECT `mail` FROM `users` WHERE `id` =
@uI", db.getConnection());
        command.Parameters.Add("@uI", MySqlDbType.Int32).Value =
row["user"].ToString();
        mail = command.ExecuteScalar().ToString();
        members.Add(mail);
    }
    members.Add("");
    db.closeConnection();
    return members;
}
```

Листинг 3.58 – метод GetProjectMembers()

Нажатие кнопки «Назначить» вызывает метод `delegateButton_Click()`. Его код представлен на листинге 3.59. Метод обновляет базу данных, а затем все открытые окна, на которых отображается задача.

```
private void delegateButton_Click(object sender, EventArgs e)
{
    string mail = responsibleBox.Text;
    DB db = new DB();
```

```

        db.openConnection();
        if (mail == "")
        {
            MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET
`responsible` = NULL WHERE `id` = @tI", db.getConnection());
            command.Parameters.Add("@tI", MySqlDbType.Int32).Value = task.id;
            command.ExecuteNonQuery();
            task.responsible = null;
        }
        else
        {
            MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET
`responsible` = @tR WHERE `id` = @tI", db.getConnection());
            command.Parameters.Add("@tR", MySqlDbType.VarChar).Value = mail;
            command.Parameters.Add("@tI", MySqlDbType.Int32).Value = task.id;
            command.ExecuteNonQuery();
            task.responsible = mail;
        }

        db.closeConnection();
        if (taskForm != null) taskForm.UpdateResponsible();
        task.UpdateResponsible();
        if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
        this.Close();
    }

```

Листинг 3.59 – метод delegateButton_Click()

3.2.11. LabelForm

При присуждении задаче новой метки вызывается новое окно, которое представляет собой панель с элементом ComboBox с выпадающим списком, кнопкой «Присвоить» и кнопкой «Новая». Внешний вид окна представлен на рисунке 3.13.

Конструктор класса принимает на вход объект текущей задачи, а также создает список всех доступных меток для назначения. Код конструктора представлен на листинге 3.61.

```

Task task;
public LabelForm(Task task)
{
    this.task = task;
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    InitializeComponent();
    List<TagBoxElem> labelsBoxSource = new List<TagBoxElem>();
    foreach (DataRow labelRow in task.GetProject().GetProjectLabels().Rows)
    {
        labelsBoxSource.Add(new TagBoxElem(labelRow));
    }
    labelsBox.ValueMember = "id";
    labelsBox.DataSource = labelsBoxSource;
}

```

Листинг 3.61 – конструктор LabelForm()

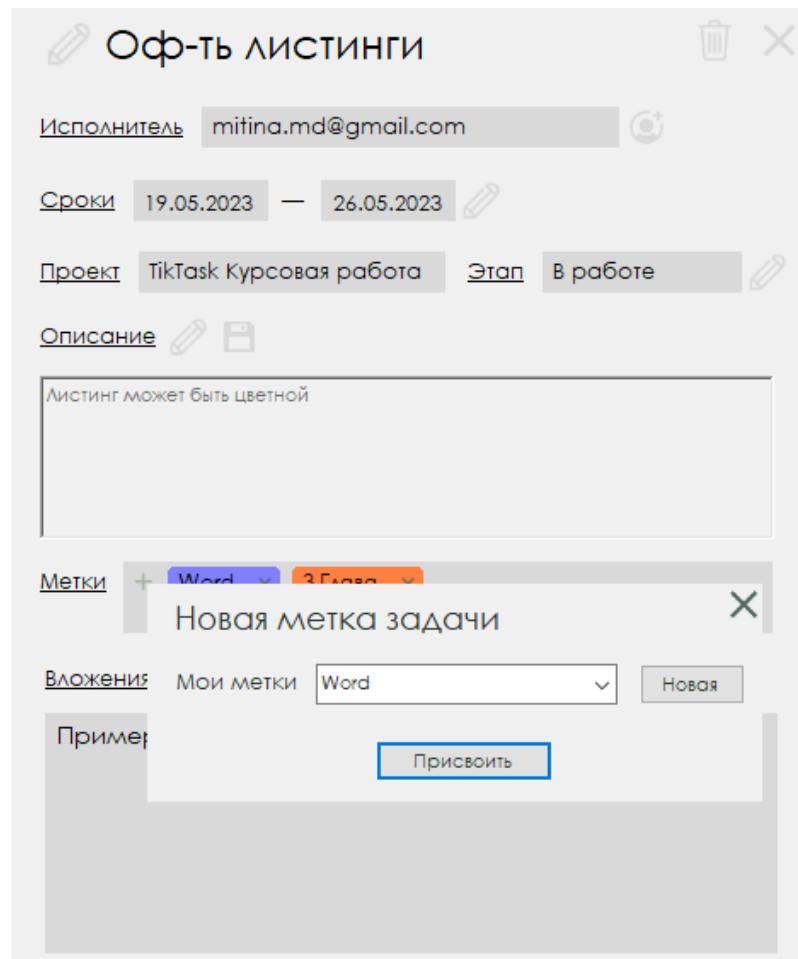


Рисунок 3.13 – окно добавления метки

При нажатии на кнопку «Новая» вызывается метод `newLabelButton_Click()`. Он открывает новое окно создания метки и скрывает текущее. Код метода представлен на листинге 3.62.

```
private void newLabelButton_Click(object sender, EventArgs e)
{
    this.Hide();
    NewLabelForm nLF = new NewLabelForm();
    nLF.Location = this.Location;
    nLF.Show();
}
```

Листинг 3.62 – метод `newLabelButton_Click()`

При нажатии на кнопку «Присвоить» вызывается метод `setLabelButton_Click()`. Он получает выбранное значение метки, обновляет базу данных и все открытые окна с текущей задачей. Код метода представлен на листинге 3.63.

```
private void setLabelButton_Click(object sender, EventArgs e)
{
    DB db = new DB();
```

```

        MySqlCommand command = new MySqlCommand("INSERT INTO `task-label`
(`task`, `label`) VALUES (@t, @l)", db.getConnection());
        command.Parameters.Add("@t", MySqlDbType.VarChar).Value = task.id;
        command.Parameters.Add("@l", MySqlDbType.Int32).Value =
((TagBoxElem)labelsBox.SelectedValue).id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
        this.Close();
        TaskForm openedTF =
Application.OpenForms.OfType<TaskForm>().FirstOrDefault();
        openedTF.UpdateTags();
        if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
    }

```

Листинг 3.63 – метод setLabelButton_Click()

3.2.12. NewLabelForm

При создании новой метки вызывается окно NewLabelForm. Оно представляет собой панель с полем для названия, кнопки «Выбрать», образца выбранного цвета и кнопки «Создать». Внешний вид окна представлен на рисунке 3.14.

При нажатии на кнопку «Выбрать» вызывается метод chooseColorButton_Click(). Метод вызывает всплывающее окно выбора цвета и при нажатии ОК присваивает этот цвет образцу цвета. Значение цвета кладется в переменную color. Код метода представлен на листинге 3.64.

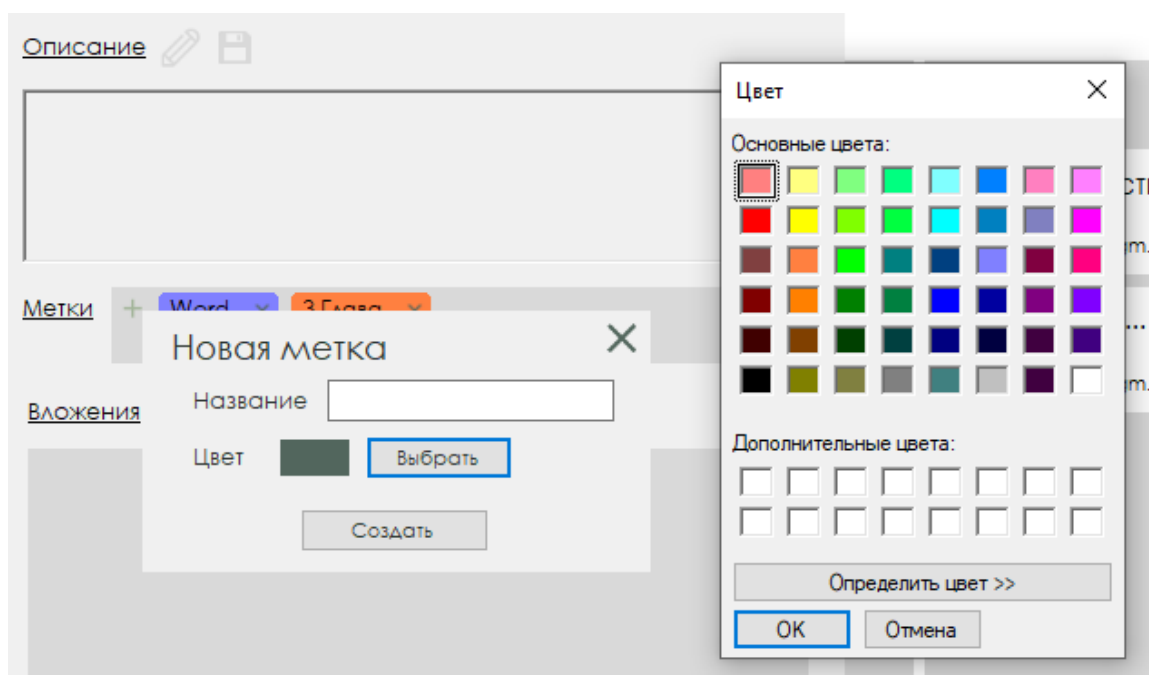


Рисунок 3.14 – окно создания новой метки


```

private void chooseColorButton_Click(object sender, EventArgs e)
{
    if (colorDialog1.ShowDialog() == DialogResult.OK)
    {
        color = colorDialog1.Color;
        colorPreview.BackColor = color;
    }
}

```

Листинг 3.64 – метод chooseColorButton_Click()

При нажатии на кнопку «Создать» вызывается метод setLabelButton_Click(): данные о новой метке добавляются в таблицу «task-label» базы данных. Код метода представлен на листинге 3.65.

```

private void setLabelButton_Click(object sender, EventArgs e)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO `labels` (`name`, `project`, `color`) VALUES (@LN, @LP, @LC)", db.getConnection());
    command.Parameters.Add("@LN", MySqlDbType.VarChar).Value = newTagNameTextBox.Text;
    ProjectForm pForm = Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
    command.Parameters.Add("@LP", MySqlDbType.Int32).Value = pForm.curProject.id;
    command.Parameters.Add("@LC", MySqlDbType.VarChar).Value = color.ToArgb().ToString();
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
    this.Close();
}

```

Листинг 3.65 – метод setLabelButton_Click()

Выводы

В данной главе была описана разработка программы для управления проектами. На основе требований, сформулированных в первой главе, и модулей, спроектированных во второй главе, было разработано 9 классов: DB, User, Project, Stage, Task, Tag, Attachment, Notification, TagBoxElem и создано 13 форм: MainForm (окно главной страницы), LoginForm (окно авторизации), UserForm (окно личного кабинета), SignUpForm (окно регистрации), ProjectForm (окно проекта), TaskForm (окно задачи), AddDateForm (окно добавления даты), AddRespForm (окно добавления исполнителя), LabelForm (окно присвоения метки), NewLabelForm (окно создания новой метки). Для каждого из классов были рассмотрены его поля и методы, описаны их функции

и приведен листинг кода. Также была рассмотрена каждая форма, приведены скриншоты их интерфейса и разобраны методы.

Заключение

В процессе разработки программы для управления IT-проектами был проанализирован и применен на практике большой объем информации по данной теме.

В первой главе был проведен обзор методик управления проектами, таких как Waterfall и Agile, а также сервисов для управления проектами, включая «Trello», «Jira» и «Asana». Был проведен сравнительный анализ функционала, интерфейса, преимуществ и недостатков каждого решения. Этот обзор позволил сформировать четкие требования к разрабатываемому продукту и определить потребности будущих пользователей.

Во второй главе была спроектирована архитектура системы, основываясь на результате обзора и требованиях пользователей. Была составлена ER-диаграмма, определены основные модули и их функциональные схемы, что позволило эффективно разрабатывать программный код в дальнейшем.

Третья глава отведена непосредственно разработке. Были созданы необходимые классы и формы на основе сформулированных требований и моделей системы. Для каждого класса были рассмотрены его поля и методы, описаны их функции, а также приведены примеры кода. Каждая форма также была проанализирована с точки зрения интерфейса и описаны ее основные методы.

Таким образом, результаты работы позволяют сделать вывод об успешной реализации программы для управления проектами, которая соответствует поставленным требованиям и позволяет эффективно организовывать и контролировать процесс выполнения задач в рамках проектов.

Список использованной литературы

1. W. Royce Managing the Development of Large Software Systems, 1970, С. 328-329;
2. Основопологающие принципы Agile-манифеста [электронный ресурс]. URL: <http://agilemanifesto.org/iso/ru/principles.html>. Дата обращения 25.03.23;
3. Capterra [электронный ресурс]. URL: https://www.capterra.com/project-planning-software/?sortOrder=most_reviews. Дата обращения 02.04.23;
4. «Trello» на Capterra [электронный ресурс]. URL: <https://www.capterra.com/p/211559/»Trello»-2/>. Дата обращения 03.04.23;
5. «Jira» на Capterra [электронный ресурс]. URL: <https://www.capterra.com/p/19319/»JIRA»/>. Дата обращения 04.04.23;
6. «Asana» на Capterra [электронный ресурс]. URL: <https://www.capterra.com/p/184581/»Asana»-PM/>. Дата обращения 04.04.23.
7. «Walkthrough: Performing a Drag-and-Drop Operation in Windows Forms» [электронный ресурс]. URL: <https://learn.microsoft.com/en-us/dotnet/desktop/winforms/advanced/walkthrough-performing-a-drag-and-drop-operation-in-windows-forms?view=netframeworkdesktop-4.8>. Дата обращения 04.05.23.
8. «Rounded Corners in C# windows forms» [электронный ресурс]. URL: <https://stackoverflow.com/questions/18822067/rounded-corners-in-c-sharp-windows-forms>. Дата обращения 05.05.23.
9. «ComboBox: Adding Text and Value to an Item» [электронный ресурс]. URL: <https://stackoverflow.com/questions/3063320/combobox-adding-text-and-value-to-an-item-no-binding-source>. Дата обращения: 10.05.23.

Приложение А. Листинг кода

Приложение А.1. Программный код класса DB

```
using MySql.Data.MySqlClient;

namespace TikTask
{
    internal class DB
    {
        MySqlConnection connection = new
        MySqlConnection("server=localhost;port=3306;username=root;password=root;database=tik
        task");
        public void openConnection()
        {
            if (connection.State == System.Data.ConnectionState.Closed) {
            connection.Open(); }
        }
        public void closeConnection()
        {
            if (connection.State == System.Data.ConnectionState.Open) {
            connection.Close(); }
        }
        public MySqlConnection getConnection()
        {
            return connection;
        }
    }
}
```

Приложение А.2. Программный код класса User

```
using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public class User
    {
        public string id, name, surname, login, mail;
        public User(string id = null, string name = null, string surname = null,
        string login = null, string mail = null)
        {
            this.id = id;
            this.name = name;
            this.surname = surname;
            this.login = login;
            this.mail = mail;
        }
        public User(string id)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE
            `id` = @userID", db.getConnection());
            command.Parameters.Add("@userID", MySqlDbType.Int32).Value = id;
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            adapter.SelectCommand = command;
            db.openConnection();
            adapter.Fill(table);
            db.closeConnection();
            this.id = id;
        }
    }
}
```

```

        foreach (DataRow row in table.Rows)
        {
            this.mail = row["mail"].ToString();
            this.name = row["name"].ToString();
            this.surname = row["surname"].ToString();
            this.login = row["login"].ToString();
        }
    }
    public DataTable GetUserTasks()
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `tasks` WHERE
`responsible` = @uM", db.getConnection());
        MainForm mainForm =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
        command.Parameters.Add("@uM", MySqlDbType.VarChar).Value = mail;
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        adapter.SelectCommand = command;
        adapter.Fill(table);

        return table;
    }
    public DataTable GetUserTasksByDate(DateTime date)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `tasks` WHERE
`finishDate` = @finDate AND `responsible` = @userMail", db.getConnection());
        command.Parameters.Add("@finDate", MySqlDbType.DateTime).Value = date;
        command.Parameters.Add("@userMail", MySqlDbType.VarChar).Value = mail;
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        DataTable table = new DataTable();
        db.openConnection();
        adapter.SelectCommand = command;
        adapter.Fill(table);
        db.closeConnection();
        return table;
    }
    public bool AddUser(TextBox login, TextBox pass, TextBox name, TextBox
surname, TextBox mail)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("INSERT INTO `users` (`login`,
`password`, `name`, `surname`, `mail`) VALUES (@uL, @uP, @uN, @uS, @uM);",
db.getConnection());

        command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = login.Text;
        command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = pass.Text;
        command.Parameters.Add("@uN", MySqlDbType.VarChar).Value = name.Text;
        command.Parameters.Add("@uS", MySqlDbType.VarChar).Value = surname.Text;
        command.Parameters.Add("@uM", MySqlDbType.VarChar).Value = mail.Text;

        this.name = name.Text;
        this.surname = surname.Text;
        this.login = login.Text;
        this.mail = mail.Text;

        db.openConnection();
        if (command.ExecuteNonQuery() == 1)
        {
            db.closeConnection();
            id = command.LastInsertedId.ToString();
            return true;
        }
    }

```

```

    }
    else
        db.closeConnection();
        return false;
    }
    public void ChangeName(string newName)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("UPDATE `users` SET `name` =
@newName WHERE `id` = @userID", db.getConnection());
        command.Parameters.Add("@newName", MySqlDbType.VarChar).Value = newName;
        name = newName;
        command.Parameters.Add("userID", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
    }
    public void ChangeSurname(string newSurname)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("UPDATE `users` SET `surname` =
@newSurname WHERE `id` = @userID", db.getConnection());
        command.Parameters.Add("@newSurname", MySqlDbType.VarChar).Value =
newSurname;
        surname = newSurname;
        command.Parameters.Add("userID", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
    }
    public void ChangePassword(string newPass)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("UPDATE `users` SET `password` =
@newPass WHERE `id` = @userID", db.getConnection());
        command.Parameters.Add("@newPass", MySqlDbType.VarChar).Value = newPass;
        command.Parameters.Add("userID", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
    }
    public DataTable GetUserNotifications()
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `notification`
WHERE `receiver` = @userID", db.getConnection());
        command.Parameters.Add("@userID", MySqlDbType.Int32).Value = id;
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        adapter.SelectCommand = command;
        db.openConnection();
        adapter.Fill(table);
        db.closeConnection();
        return table;
    }
    public DataTable GetUserProjects()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();

        MySqlCommand command = new MySqlCommand("SELECT * FROM `involved` WHERE
`user` = @user", db.getConnection());

```

```

        command.Parameters.Add("@user", MySqlDbType.Int32).Value = id;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        return table;
    }
}

```

Приложение А.3. Программный код класса Project

```

using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public class Project
    {
        public string id, name, host;
        public DateTime? startDate, finishDate;
        public Panel previewCard;
        public MainForm main;

        [DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]
        private static extern IntPtr CreateRoundRectRgn
        (
            int nLeftRect,
            int nTopRect,
            int nRightRect,
            int nBottomRect,
            int nWidthEllipse,
            int nHeightEllipse
        );

        public Project(MainForm main, string name = null, DateTime? startDate =
null, DateTime? finishDate = null, string host = null, string id = null)
        {
            this.id = id;
            this.name = name;
            this.startDate = startDate;
            this.finishDate = finishDate;
            this.host = host;
            this.main = main;
        }

        public Project(string id)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("SELECT * FROM `projects` WHERE
`id` = @pI", db.getConnection());
            command.Parameters.Add("@pI", MySqlDbType.Int32).Value = id;
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            adapter.SelectCommand = command;
            DataTable table = new DataTable();
            adapter.Fill(table);

            foreach (DataRow row in table.Rows)
            {
                this.id = id;
            }
        }
    }
}

```



```

        this.name = row["name"].ToString();
        this.startDate = DateTime.Parse(row["startDate"].ToString());
        this.finishDate = DateTime.Parse(row["finishDate"].ToString());
        this.host = row["host"].ToString();
    }
}

public void PushToDB()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO `projects` (`name`,
`startDate`, `finishDate`, `host`) VALUES (@pN, @pSD, @pED, @pH);",
db.getConnection());

    command.Parameters.Add("@pN", MySqlDbType.VarChar).Value = name;
    command.Parameters.Add("@pSD", MySqlDbType.DateTime).Value =
startDate.Value;
    command.Parameters.Add("@pED", MySqlDbType.DateTime).Value =
finishDate.Value;
    command.Parameters.Add("@pH", MySqlDbType.Int32).Value = host;

    db.openConnection();
    command.ExecuteNonQuery();
    this.id = command.LastInsertedId.ToString();
    command = new MySqlCommand("INSERT INTO `involved` (`project`, `user`)
VALUES (@pI, @uI);", db.getConnection());
    command.Parameters.Add("@pI", MySqlDbType.Int32).Value = this.id;
    command.Parameters.Add("@uI", MySqlDbType.Int32).Value = this.host;
    command.ExecuteNonQuery();
    db.closeConnection();
}

public Panel GetPreviewCard()
{
    previewCard = new Panel();
    previewCard.Width = 320;
    previewCard.Height = 70;
    previewCard.BackColor = Color.White;
    previewCard.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0, 320, 70,
10, 10));

    Label projectNameLabel = new Label();
    projectNameLabel.Width = 250;
    projectNameLabel.AutoEllipsis = true;
    projectNameLabel.Text = this.name;
    projectNameLabel.Font = new Font("Century Gothic", 14);
    projectNameLabel.Location = new Point(previewCard.Location.X + 10,
previewCard.Location.Y + 5);

    PictureBox deadline = new PictureBox();
    deadline.Image = Resources.calendarDark32;
    deadline.Height = 24;
    deadline.Width = 24;
    deadline.SizeMode = PictureBoxSizeMode.StretchImage;
    deadline.Location = new Point(previewCard.Location.X + 15,
projectNameLabel.Location.Y + 35);
    Label dateLabel = new Label();
    dateLabel.Text = finishDate.ToString().Split()[0];
    dateLabel.AutoSize = true;
    dateLabel.Font = new Font("Century Gothic", 8);
    dateLabel.Location = new Point(deadline.Location.X + deadline.Width + 3,
deadline.Location.Y + 5);
    if (finishDate == DateTime.Today) dateLabel.Text = "Сегодня";
}

```

```

        if (finishDate < DateTime.Today) dateLabel.ForeColor = Color.Red;

        PictureBox hostImage = new PictureBox();
        hostImage.Image = Resources.userDark24;
        hostImage.Height = 22;
        hostImage.Width = 22;
        hostImage.SizeMode = PictureBoxSizeMode.StretchImage;
        hostImage.Location = new Point(dateLabel.Location.X + dateLabel.Width +
10, projectNameLabel.Location.Y + 36);

        User hostUs = new User(host);
        Label hostLabel = new Label();
        hostLabel.Text = hostUs.mail;
        hostLabel.AutoEllipsis = true;
        hostLabel.Font = new Font("Century Gothic", 8);
        hostLabel.Location = new Point(hostImage.Location.X + hostImage.Width +
3, hostImage.Location.Y + 5);

        PictureBox openProjectButton = new PictureBox();
        openProjectButton.Image = Resources.openLight32;
        openProjectButton.Height = 24;
        openProjectButton.Width = 24;
        openProjectButton.SizeMode = PictureBoxSizeMode.StretchImage;
        openProjectButton.Cursor = Cursors.Hand;
        openProjectButton.Location = new Point(previewCard.Width - 35, 7);
        openProjectButton.Click += (sender, e) =>
        {
            ProjectForm form = new ProjectForm(this);
            MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
            main.Hide();
            form.Show();
        };
        openProjectButton.MouseEnter += (sender, e) =>
        {
            openProjectButton.Image = Resources.openDark32;
        };
        openProjectButton.MouseLeave += (sender, e) =>
        {
            openProjectButton.Image = Resources.openLight32;
        };

        previewCard.Controls.Add(openProjectButton);
        previewCard.Controls.Add(projectNameLabel);
        previewCard.Controls.Add(deadline);
        previewCard.Controls.Add(dateLabel);
        previewCard.Controls.Add(hostImage);
        previewCard.Controls.Add(hostLabel);
        return previewCard;
    }

    public DataTable GetProjectsStages()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();

        MySqlCommand command = new MySqlCommand("SELECT * FROM `stages` WHERE
`project` = @CUI", db.getConnection());
        command.Parameters.Add("@CUI", MySqlDbType.Int32).Value = id;

        adapter.SelectCommand = command;
        adapter.Fill(table);
    }

```

```

        return table;
    }
    public DataTable GetProjectTasks()
    {
        DataTable stages = GetProjectsStages();
        DB db = new DB();
        DataTable tasks = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        MySqlCommand command;
        db.openConnection();
        foreach (DataRow row in stages.Rows)
        {
            command = new MySqlCommand("SELECT * FROM `tasks` WHERE `stage` =
@stageID", db.getConnection());
            command.Parameters.Add("@stageID", MySqlDbType.Int32).Value =
row["id"].ToString();
            adapter.SelectCommand = command;
            adapter.Fill(tasks);
        }
        db.closeConnection();
        return tasks;
    }
    public DataTable GetProjectLabels()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();

        MySqlCommand command = new MySqlCommand("SELECT * FROM `labels` WHERE
`project` = @CUI", db.getConnection());
        command.Parameters.Add("@CUI", MySqlDbType.Int32).Value = id;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        return table;
    }
    public void EditName(string newName)
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("UPDATE `projects` SET `name` =
@projectName WHERE `id` = @projectID ", db.getConnection());
        command.Parameters.Add("@projectName", MySqlDbType.VarChar).Value =
newName;
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
        name = newName;
    }
    public void Delete()
    {
        foreach (DataRow taskRow in GetProjectTasks().Rows)
        {
            Task task = new Task(taskRow);
            task.Delete();
        }
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("DELETE FROM `involved` WHERE
`project` = @projectID", db.getConnection());
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
    }

```

```

        command = new MySqlCommand("DELETE FROM `stages` WHERE `project` =
@projectID", db.getConnection());
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
        command.ExecuteNonQuery();
        command = new MySqlCommand("DELETE FROM `projects` WHERE `id` =
@projectID", db.getConnection());
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
        command.ExecuteNonQuery();
        command = new MySqlCommand("DELETE FROM `labels` WHERE `project` =
@projectID", db.getConnection());
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value = id;
        command.ExecuteNonQuery();
        db.closeConnection();
    }
}
}

```

Приложение А.4. Программный код класса Stage

```

using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public class Stage
    {
        public string id, name, author, project;
        public ProjectForm parentProject;
        public MainForm mainProj;
        public FlowLayoutPanel stage;
        public bool activeTaskPreview = false;
        PictureBox addTaskButton, editStageButton, deleteStageButton;
        Label stageName;
        Panel head;

        public Stage(MainForm main, ProjectForm parentProject, string name, string
author, string id)
        {
            this.author = author;
            this.name = name;
            this.mainProj = main;
            this.parentProject = parentProject;
            this.id = id;
        }

        public Stage(string id)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("SELECT * FROM `stages` WHERE
`id` = @sI", db.getConnection());
            command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            adapter.SelectCommand = command;
            DataTable table = new DataTable();
            adapter.Fill(table);

            foreach (DataRow row in table.Rows)
            {
                this.id = row["id"].ToString();
            }
        }
    }
}

```

```

        this.name = row["name"].ToString();
        this.author = row["author"].ToString();
        this.project = row["project"].ToString();
    }
}
public Stage(DataRow row)
{
    this.id = row["id"].ToString();
    this.name = row["name"].ToString();
    this.author = row["author"].ToString();
    this.project = row["project"].ToString();
}
public bool PushToDB(TextBox name)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO `stages` (`name`,
`author`, `project`) VALUES (@sN, @sA, @sPr);", db.getConnection());

    command.Parameters.Add("@sN", MySqlDbType.VarChar).Value = name.Text;
    this.name = name.Text;
    MainForm mainForm =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
    command.Parameters.Add("@sA", MySqlDbType.Int32).Value =
mainForm.curUser.id;
    this.author = mainForm.curUser.id;
    command.Parameters.Add("@sPr", MySqlDbType.Int32).Value =
parentProject.curProject.id;
    this.project = parentProject.curProject.id;

    db.openConnection();
    if (command.ExecuteNonQuery() == 1)
    {
        db.closeConnection();
        this.id = command.LastInsertedId.ToString();
        return true;
    }
    else
        db.closeConnection();
    return false;
}
public FlowLayoutPanel GetStagePanel(int height)
{
    stage = new FlowLayoutPanel();
    stage.AllowDrop = true;
    stage.DragEnter += Stage_DragEnter;
    stage.DragOver += Stage_DragOver;
    stage.DragDrop += Stage_DragDrop;
    stage.Width = 300;
    stage.Height = height;
    stage.AutoScroll = true;
    stage.BackColor = Color.FromArgb(217, 217, 217);
    head = new Panel();
    head.Width = stage.Width - 30;
    head.Height = 40;

    stageName = new System.Windows.Forms.Label();
    stageName.Width = stage.Width - 110;
    stageName.Height = 20;
    stageName.Text = name;
    stageName.AutoEllipsis = true;
    stageName.Font = new Font("Century Gothic", 14);
    stageName.ForeColor = Color.FromArgb(82, 102, 93);

    addTaskButton = new PictureBox();

```

```

        addTaskButton.Width = 24;
        addTaskButton.Height = 24;
        addTaskButton.Image = Properties.Resources.addDark32;
        addTaskButton.SizeMode = PictureBoxSizeMode.StretchImage;
        addTaskButton.Cursor = Cursors.Hand;
        addTaskButton.Location = new Point(head.Location.X + head.Width - 50 -
addTaskButton.Width, default);
        addTaskButton.Click += AddTaskButton_Click;
        addTaskButton.MouseEnter += AddTaskButton_MouseEnter;
        addTaskButton.MouseLeave += AddTaskButton_MouseLeave;
        deleteStageButton = new PictureBox();
        deleteStageButton.Width = 18;
        deleteStageButton.Height = 18;
        deleteStageButton.Image = Properties.Resources.deleteDark;
        deleteStageButton.SizeMode = PictureBoxSizeMode.StretchImage;
        deleteStageButton.Cursor = Cursors.Hand;
        deleteStageButton.Location = new Point(addTaskButton.Location.X + 26,
addTaskButton.Location.Y + 4);
        deleteStageButton.Click += deleteStageButton_Click;
        deleteStageButton.MouseEnter += (sender, e) => { deleteStageButton.Image
= Resources.deleteBlack; };
        deleteStageButton.MouseLeave += (sender, e) => { deleteStageButton.Image
= Resources.deleteDark; };
        editStageButton = new PictureBox();
        editStageButton.Width = 18;
        editStageButton.Height = 18;
        editStageButton.Image = Properties.Resources.editDark32;
        editStageButton.SizeMode = PictureBoxSizeMode.StretchImage;
        editStageButton.Cursor = Cursors.Hand;
        editStageButton.Location = new Point(deleteStageButton.Location.X + 26,
addTaskButton.Location.Y + 4);
        editStageButton.MouseEnter += EditStageButton_MouseEnter;
        editStageButton.MouseLeave += EditStageButton_MouseLeave;
        editStageButton.Click += EditStageButton_Click;
        head.Controls.Add(stageName);
        head.Controls.Add(editStageButton);
        head.Controls.Add(addTaskButton);
        head.Controls.Add(deleteStageButton);
        stage.Controls.Add(head);
        return stage;
    }

    private void Stage_DragDrop(object sender, DragEventArgs e)
    {
        Panel taskPanel = (Panel)e.Data.GetData(typeof(Panel));
        string taskId = taskPanel.Tag.ToString();
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET `stage` =
@stageID WHERE `id` = @taskId", db.getConnection());
        command.Parameters.Add("@stageID", MySqlDbType.Int32).Value = id;
        command.Parameters.Add("@taskId", MySqlDbType.Int32).Value = taskId;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
        ProjectForm projForm =
Application.OpenForms.Of<ProjectForm>().FirstOrDefault();
        projForm.UpdateProjectView();
    }

    private void Stage_DragEnter(object sender, DragEventArgs e)
    {
        if (e.Data.GetDataPresent(typeof(Panel)))
        {
            e.Effect = DragDropEffects.Move;
        }
    }

```

```

    }
}
private void Stage_DragOver(object sender, DragEventArgs e)
{
    if (e.Data.GetDataPresent(typeof(Panel)))
    {
        Panel taskPanel = (Panel)e.Data.GetData(typeof(Panel));
        Panel targetColumn = (Panel)sender;

        if (taskPanel.Parent != targetColumn)
        {
            targetColumn.Controls.Add(taskPanel);
        }
    }
}
TextBox stageField;
private void EditStageButton_Click(object sender, EventArgs e)
{
    head.Controls.Remove(stageName);
    stageField = new TextBox();
    stageField.Width = stage.Width - 80;
    stageField.Height = 18;
    stageField.Text = name;
    stageField.Font = new Font("Century Gothic", 14);
    stageField.ForeColor = Color.FromArgb(82, 102, 93);
    stageField.KeyPress += StageField_KeyPress;
    head.Controls.Add(stageField);
}
private void deleteStageButton_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Вы уверены что хотите удалить этап? Задачи этапа удалятся вместе с ним.", "Подтверждение удаления", MessageBoxButtons.OKCancel);

    if (result == DialogResult.OK)
    {
        this.Delete();
        ProjectForm openedPF =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
        openedPF.UpdateProjectView();
    }
    else if (result == DialogResult.Cancel)
    {
        return;
    }
}
public void Delete()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("DELETE FROM `tasks` WHERE `stage` = @sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    command = new MySqlCommand("DELETE FROM `stages` WHERE `id` = @sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
    command.ExecuteNonQuery();
    db.closeConnection();
}
private void StageField_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Enter)

```

```

        {
            this.name = stageField.Text;
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("UPDATE `stages` SET `name`
= @pN WHERE `id` = @sI", db.getConnection());
            command.Parameters.Add("@sI", MySqlDbType.Int32).Value = id;
            command.Parameters.Add("@pN", MySqlDbType.VarChar).Value = name;
            db.openConnection();
            command.ExecuteNonQuery();
            db.closeConnection();
            stageName.Text = name;
            head.Controls.Remove(stageField);
            head.Controls.Add(stageName);
        }
    }
    private void EditStageButton_MouseLeave(object sender, EventArgs e)
    {
        editStageButton.Image = Properties.Resources.editDark32;
    }
    private void EditStageButton_MouseEnter(object sender, EventArgs e)
    {
        editStageButton.Image = Properties.Resources.editBlack32;
    }
    private void AddTaskButton_MouseLeave(object sender, EventArgs e)
    {
        addTaskButton.Image = Properties.Resources.addDark32;
    }
    private void AddTaskButton_MouseEnter(object sender, EventArgs e)
    {
        addTaskButton.Image = Properties.Resources.addBlack32;
    }
    public void AddTaskButton_Click(object sender, EventArgs e)
    {
        if (!activeTaskPreview)
        {
            Task new_task = new Task(this);
            Panel panel = new_task.GetTaskPreviewPanel();
            stage.Controls.Add(panel);
        }
    }
    public DataTable GetStagesTasks()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();

        MySqlCommand command = new MySqlCommand("SELECT * FROM `tasks` WHERE
`stage` = @tS", db.getConnection());
        command.Parameters.Add("@tS", MySqlDbType.Int32).Value = id;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        return table;
    }
}
}

```

Приложение А.5. Программный код класса Task

```

using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Linq;

```



```

using System.Runtime.InteropServices;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public class Task
    {
        public string id, name, stage;
        public string responsible, description = null;
        bool archived = false;
        public DateTime startDate, finishDate;

        Panel taskPanel;
        TextBox taskNameField;
        public bool activeField = false;
        Stage parentStage;
        PictureBox addResponsButton;
        PictureBox addDatesButton;
        PictureBox openTaskButton;
        System.Windows.Forms.Label Responsible = new System.Windows.Forms.Label();
        CheckBox taskNameLabel;

        [DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]
        private static extern IntPtr CreateRoundRectRgn
        (
            int nLeftRect,
            int nTopRect,
            int nRightRect,
            int nBottomRect,
            int nWidthEllipse,
            int nHeightEllipse
        );

        public Task(string id, string name, string description, string stage,
            DateTime startDate, DateTime finishDate, Stage parentStage = null)
        {
            this.id = id;
            this.name = name;
            this.description = description;
            this.stage = stage;
            this.startDate = startDate;
            this.finishDate = finishDate;
            this.parentStage = parentStage;
        }

        public Task(DataRow row)
        {
            this.id = row["id"].ToString();
            this.name = row["name"].ToString();
            if (row["description"] != null) this.description =
row["description"].ToString();
            this.stage = row["stage"].ToString();
            if (row["startDate"] != null) this.startDate =
DateTime.Parse(row["startDate"].ToString());
            if (row["finishDate"] != null) this.finishDate =
DateTime.Parse(row["finishDate"].ToString());
            if (row["responsible"] != null) this.responsible =
row["responsible"].ToString();
            this.archived = Convert.ToBoolean(row["archived"]);
        }

        public Task(string id)
        {

```

```

        DB db = new DB();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `tasks` WHERE
`id` = @tI", db.getConnection());
        command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        adapter.SelectCommand = command;
        DataTable table = new DataTable();
        adapter.Fill(table);

        foreach (DataRow row in table.Rows)
        {
            this.id = row["id"].ToString();
            this.name = row["name"].ToString();
            if (row["description"] != null) this.description =
row["description"].ToString();
            this.stage = row["stage"].ToString();
            if (row["startDate"] != null) this.startDate =
DateTime.Parse(row["startDate"].ToString());
            if (row["startDate"] != null) this.finishDate =
DateTime.Parse(row["finishDate"].ToString());
            if (row["responsible"] != null) this.responsible =
row["responsible"].ToString();
            this.archived = Convert.ToBoolean(row["archived"]);
        }
    }
    public Task(Stage parentStage)
    {
        this.parentStage = parentStage;
        this.stage = parentStage.id;
    }

    public Project GetProject()
    {
        Stage stage = new Stage(this.stage);
        Project project = new Project(stage.project);
        return project;
    }
    public void Delete()
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("DELETE FROM `tasks` WHERE `id`
= @tI", db.getConnection());
        command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
        command = new MySqlCommand("DELETE FROM `attachments` WHERE `task` =
@tI", db.getConnection());
        command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
        command.ExecuteNonQuery();
        command = new MySqlCommand("DELETE FROM `task-label` WHERE `task` =
@tI", db.getConnection());
        command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
        command.ExecuteNonQuery();
        db.closeConnection();
    }
    public void PushToDB()
    {
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("INSERT INTO `tasks`
(`archived`, `name`, `stage`, `description`, `startDate`, `finishDate`) VALUES
(@tAr, @tN, @tS, @tD, @tSD, @tFD);", db.getConnection());

        command.Parameters.Add("tAr", MySqlDbType.Int32).Value = this.archived;
        command.Parameters.Add("@tN", MySqlDbType.VarChar).Value = name;
    }

```

```

        command.Parameters.Add("@tS", MySqlDbType.Int32).Value = parentStage.id;
        if (description != null)
        {
            command.Parameters.Add("tD", MySqlDbType.VarChar).Value =
description;
        }
        else
        {
            command.Parameters.Add("tD", MySqlDbType.VarChar).Value = null;
        }
        if (startDate != null)
        {
            command.Parameters.Add("tSD", MySqlDbType.DateTime).Value =
startDate;
        }
        else
        {
            command.Parameters.Add("tSD", MySqlDbType.DateTime).Value = null;
        }
        if (finishDate != null)
        {
            command.Parameters.Add("tFD", MySqlDbType.DateTime).Value =
finishDate;
        }
        else
        {
            command.Parameters.Add("tFD", MySqlDbType.DateTime).Value = null;
        }

        db.openConnection();
        command.ExecuteNonQuery();
        this.id = command.LastInsertedId.ToString();
        db.closeConnection();
    }

    public Panel GetTaskPreviewPanel()
    {
        taskPanel = new Panel();
        taskPanel.Width = 280;
        taskPanel.Height = 70;
        taskPanel.BackColor = Color.White;
        taskPanel.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0,
taskPanel.Width, taskPanel.Height, 10, 10));

        taskNameField = new TextBox();
        taskNameField.Font = new Font("Century Gothic", 12);
        taskNameField.Location = new Point(taskPanel.Location.X + 5,
taskPanel.Location.Y + 5);
        taskNameField.Width = taskPanel.Width - 50;
        taskNameField.KeyPress += TaskName_KeyPress;
        taskPanel.Controls.Add(taskNameField);

        if (parentStage != null) parentStage.activeTaskPreview = true;

        openTaskButton = new PictureBox();
        openTaskButton.Height = 24;
        openTaskButton.Width = 24;
        openTaskButton.SizeMode = PictureBoxSizeMode.StretchImage;
        openTaskButton.Image = Resources.openLight32;
        openTaskButton.Cursor = Cursors.Hand;
        openTaskButton.Location = new Point(taskPanel.Width - 30, 6);
        openTaskButton.MouseEnter += OpenTaskButton_MouseEnter;
        openTaskButton.MouseLeave += OpenTaskButton_MouseLeave;
        openTaskButton.Click += OpenTaskButton_Click;
    }

```

```

        taskPanel.Controls.Add(openTaskButton);

        return taskPanel;
    }
    Label deadline = new Label();
    FlowLayoutPanel tagsPanel;
public Panel GetExistingTaskPreviewPanel(bool allowDND)
{
    taskPanel = new Panel();
    taskPanel.Tag = id;
    taskPanel.Width = 280;
    taskPanel.Height = 70;
    taskPanel.BackColor = Color.White;
    if (allowDND)
    {
        taskPanel.Tag = id;
        taskPanel.MouseDown += TaskPanel_MouseDown;
    }
    taskPanel.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0,
taskPanel.Width, taskPanel.Height, 10, 10));
    if (archived) taskPanel.BackColor = Color.FromArgb(163, 181, 154);
    openTaskButton = new PictureBox();
    openTaskButton.Height = 24;
    openTaskButton.Width = 24;
    openTaskButton.SizeMode = PictureBoxSizeMode.StretchImage;
    openTaskButton.Image = Resources.openLight32;
    openTaskButton.Cursor = Cursors.Hand;
    openTaskButton.Location = new Point(taskPanel.Width - 30, 6);
    openTaskButton.MouseEnter += OpenTaskButton_MouseEnter;
    openTaskButton.MouseLeave += OpenTaskButton_MouseLeave;
    openTaskButton.Click += OpenTaskButton_Click;

    taskNameLabel = new CheckBox();
    taskNameLabel.Text = name;
    taskNameLabel.Size = new Size(200, 30);
    if (archived)
    {
        taskNameLabel.Font = new Font("Century Gothic", 12,
FontStyle.Strikeout);
        taskNameLabel.Checked = true;
    }
    else taskNameLabel.Font = new Font("Century Gothic", 12);
    taskNameLabel.Location = new Point(10, 5);
    taskNameLabel.AutoEllipsis = true;
    taskNameLabel.CheckedChanged += TaskNameLabel_CheckedChanged;

    addResponsButton = new PictureBox();
    addResponsButton.Location = new Point(10, taskPanel.Height - 30);
    addResponsButton.Height = 24;
    addResponsButton.Width = 24;
    addDatesButton = new PictureBox();
    addDatesButton.Location = new Point(140, taskPanel.Height - 30);
    addDatesButton.SizeMode = PictureBoxSizeMode.StretchImage;
    addDatesButton.Height = 24;
    addDatesButton.Width = 24;
    if (responsible == null || responsible == "")
    {
        addResponsButton.Image = Resources.add_user_light;
        addResponsButton.Cursor = Cursors.Hand;
        Responsible.Text = "";
        addResponsButton.MouseEnter += AddResponsButton_MouseEnter;
        addResponsButton.MouseLeave += AddResponsButton_MouseLeave;
        addResponsButton.Click += AddResponsButton_Click;
    }
}

```

```

        ToolTip responsibTip = new ToolTip();
        responsibTip.SetToolTip(addResponsButton, "Назначить задачу");
    }
    else
    {
        addResponsButton.Image = Resources.userDark24;
        Responsible.Text = responsible;
    }

    Responsible.Font = new Font("Century Gothic", 8);
    Responsible.Location = new Point(addResponsButton.Location.X + 25,
addResponsButton.Location.Y + 5);
    Responsible.Width = addDatesButton.Location.X -
addResponsButton.Location.X - 25;
    Responsible.AutoEllipsis = true;
    taskPanel.Controls.Add(Responsible);
    if (finishDate == DateTime.MinValue)
    {
        addDatesButton.Image = Resources.addDate_light;
        addDatesButton.Cursor = Cursors.Hand;
        addDatesButton.MouseEnter += addDatesButton_MouseEnter;
        addDatesButton.MouseLeave += addDatesButton_MouseLeave;
        addDatesButton.Click += AddDatesButton_Click;
        ToolTip dateTip = new ToolTip();
        dateTip.SetToolTip(addDatesButton, "Назначить сроки");
    }
    else
    {
        addDatesButton.Image = Resources.calendarDark32;
        if (finishDate == DateTime.Today) deadline.Text = "Сегодня";
        else deadline.Text = finishDate.ToString().Split()[0];
        deadline.Font = new Font("Century Gothic", 8);
        if (finishDate < DateTime.Today && !archived) deadline.ForeColor =
Color.Red;
        deadline.Location = new Point(addDatesButton.Location.X + 25,
addDatesButton.Location.Y + 5);
        deadline.Width = addDatesButton.Location.X -
addResponsButton.Location.X - 25;
        taskPanel.Controls.Add(deadline);
        addDatesButton.SizeMode = PictureBoxSizeMode.StretchImage;
        addDatesButton.MouseEnter -= addDatesButton_MouseEnter;
        addDatesButton.MouseLeave -= addDatesButton_MouseLeave;
        addDatesButton.Click -= AddDatesButton_Click;
    }

    tagsPanel = new FlowLayoutPanel();
    tagsPanel.FlowDirection = FlowDirection.RightToLeft;
    tagsPanel.Size = new Size(60, 20);
    tagsPanel.AutoScroll = false;
    tagsPanel.Location = new Point(180, 8);
    foreach (DataRow tagRow in GetTasksTags().Rows)
    {
        Tag tag = new Tag(tagRow["label"].ToString());
        Panel tagPanel = tag.GetSmallTag();
        tagsPanel.Controls.Add(tagPanel);
    }
    taskPanel.Controls.Add(tagsPanel);
    taskPanel.Controls.Add(addResponsButton);
    taskPanel.Controls.Add(addDatesButton);
    taskPanel.Controls.Add(openTaskButton);
    taskPanel.Controls.Add(taskNameLabel);
    return taskPanel;
}

```

```

private void TaskPanel_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        Panel taskPanel = (Panel)sender;
        taskPanel.DoDragDrop(taskPanel, DragDropEffects.Move);
    }
}

private void TaskNameLabel_CheckedChanged(object sender, EventArgs e)
{
    if (taskNameLabel.Checked)
    {
        this.archived = true;
        taskPanel.BackColor = Color.FromArgb(163, 181, 154);
        taskNameLabel.Font = new Font("Century Gothic", 12,
FontStyle.Strikeout);
        deadline.ForeColor = Color.Black;
    }
    else
    {
        this.archived = false;
        taskPanel.BackColor = Color.White;
        taskNameLabel.Font = new Font("Century Gothic", 12);
        if (finishDate < DateTime.Today && !archived) deadline.ForeColor =
Color.Red;
    }
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET `archived` =
@tS WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("@tS", MySqlDbType.Binary).Value =
Convert.ToInt32(this.archived);
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = this.id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
}

private void OpenTaskButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<TaskForm>().FirstOrDefault() != null)
Application.OpenForms.OfType<TaskForm>().FirstOrDefault().Close();
    TaskForm taskForm = new TaskForm(this);
    taskForm.Show();
}

private void OpenTaskButton_MouseLeave(object sender, EventArgs e)
{
    openTaskButton.Image = Resources.openLight32;
}

private void OpenTaskButton_MouseEnter(object sender, EventArgs e)
{
    openTaskButton.Image = Resources.openDark32;
}

private void TaskName_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Enter)
    {
        taskPanel.Controls.Remove(taskNameField);

        CheckBox taskNameLabel = new CheckBox();
        taskNameLabel.Text = taskNameField.Text;

```

```

        this.name = taskNameLabel.Text;
        taskNameLabel.Size = new Size(200, 30);
        taskNameLabel.Location = new Point(10, 5);
        taskNameLabel.AutoEllipsis = true;

        addResponsButton = new PictureBox();
        addResponsButton.Height = 24;
        addResponsButton.Width = 24;
        addResponsButton.Image = Resources.add_user_light;
        addResponsButton.Location = new Point(10, taskPanel.Height - 30);
        addResponsButton.Cursor = Cursors.Hand;
        addResponsButton.MouseEnter += AddResponsButton_MouseEnter;
        addResponsButton.MouseLeave += AddResponsButton_MouseLeave;
        addResponsButton.Click += AddResponsButton_Click;
        ToolTip responsibTip = new ToolTip();
        responsibTip.SetToolTip(addResponsButton, "Назначить задачу");

        addDatesButton = new PictureBox();
        addDatesButton.Height = 24;
        addDatesButton.Width = 24;
        addDatesButton.Image = Resources.addDate_light;
        addDatesButton.Location = new Point(140, taskPanel.Height - 30);
        addDatesButton.Cursor = Cursors.Hand;
        addDatesButton.MouseEnter += addDatesButton_MouseEnter;
        addDatesButton.MouseLeave += addDatesButton_MouseLeave;
        addDatesButton.Click += AddDatesButton_Click;
        ToolTip dateTip = new ToolTip();
        dateTip.SetToolTip(addDatesButton, "Назначить сроки");

        taskPanel.Controls.Add(addResponsButton);
        taskPanel.Controls.Add(addDatesButton);
        taskPanel.Controls.Add(taskNameLabel);
        parentStage.activeTaskPreview = false;
        this.PushToDB();
        taskPanel.Tag = id;
        taskPanel.MouseDown += TaskPanel_MouseDown;
    }
}

private void AddDatesButton_Click(object sender, EventArgs e)
{
    AddDateForm form = new AddDateForm(this);
    form.Show();
}

private void AddResponsButton_Click(object sender, EventArgs e)
{
    AddRespForm form = new AddRespForm(this);
    form.Show();
}

public void UpdateResponsible()
{
    if (responsible != null)
    {
        addResponsButton.Image = Resources.userDark24;
        Responsible.Text = responsible;
        addResponsButton.MouseEnter -= AddResponsButton_MouseEnter;
        addResponsButton.MouseLeave -= AddResponsButton_MouseLeave;
    }
    else
    {
        addResponsButton.Image = Resources.add_user_light;
        addResponsButton.Cursor = Cursors.Hand;
    }
}

```

```

        Responsible.Text = "";
        addResponsButton.MouseEnter += AddResponsButton_MouseEnter;
        addResponsButton.MouseLeave += AddResponsButton_MouseLeave;
        addResponsButton.Click += AddResponsButton_Click;
        ToolTip responsibTip = new ToolTip();
        responsibTip.SetToolTip(addResponsButton, "Назначить задачу");
    }

}

public void UpdateFinishDate()
{
    addDatesButton.Image = Resources.calendarDark32;
    if (finishDate == DateTime.Today) deadline.Text = "Сегодня";
    else deadline.Text = finishDate.ToString().Split()[0];
    if (finishDate < DateTime.Today) deadline.ForeColor = Color.Red;
    addDatesButton.MouseEnter -= addDatesButton_MouseEnter;
    addDatesButton.MouseLeave -= addDatesButton_MouseLeave;
}

public void UpdateStage(string newStageId)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET `stage` =
@sI WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = newStageId;
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
    this.stage = newStageId;
}

public void UpdateName(string newName)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET `name` = @nN
WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("@nN", MySqlDbType.VarChar).Value = newName;
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
    this.name = newName;
}

public DataTable GetTasksTags()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `label` FROM `task-
label` WHERE `task` = @tI", db.getConnection());
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    adapter.SelectCommand = command;
    db.openConnection();
    adapter.Fill(table);
    db.closeConnection();
    return table;
}

public DataTable GetAttachments()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT * FROM `attachments`
WHERE `task` = @tI", db.getConnection());
    command.Parameters.Add("@tI", MySqlDbType.Int32).Value = id;

```



```

        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        adapter.SelectCommand = command;
        db.openConnection();
        adapter.Fill(table);
        db.closeConnection();
        return table;
    }

    private void AddResponsButton_MouseEnter(object sender, EventArgs e)
    {
        addResponsButton.Image = Resources.add_user_dark;
    }
    private void AddResponsButton_MouseLeave(object sender, EventArgs e)
    {
        addResponsButton.Image = Resources.add_user_light;
    }
    private void addDatesButton_MouseEnter(object sender, EventArgs e)
    {
        addDatesButton.Image = Resources.addDate_dark;
    }
    private void addDatesButton_MouseLeave(object sender, EventArgs e)
    {
        addDatesButton.Image = Resources.addDate_light;
    }
}
}
}

```

Приложение А.6. Программный код класса Tag

```

using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Runtime.InteropServices;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    internal class Tag
    {
        string id, project, name, color;
        public Tag(string id, string name, string color, string project)
        {
            this.id = id;
            this.name = name;
            this.color = color;
            this.project = project;
        }
        public Tag(DataRow labelRow)
        {
            id = labelRow["id"].ToString();
            name = labelRow["name"].ToString();
            color = labelRow["color"].ToString();
            project = labelRow["project"].ToString();
        }
        public Tag(string id)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("SELECT * FROM `labels` WHERE
`id` = @LI", db.getConnection());
            command.Parameters.Add("@LI", MySqlDbType.Int32).Value = id;
            MySqlDataAdapter adapter = new MySqlDataAdapter();

```

```

        adapter.SelectCommand = command;
        DataTable table = new DataTable();
        adapter.Fill(table);

        foreach (DataRow row in table.Rows)
        {
            this.id = id;
            this.name = row["name"].ToString();
            this.color = row["color"].ToString();
            this.project = row["project"].ToString();
        }
    }
    public Color GetColor()
    {
        Color tagColor = Color.FromArgb(Convert.ToInt32(color));
        return tagColor;
    }
    [DllImport("Gdi32.dll", EntryPoint = "CreateRoundRectRgn")]
    private static extern IntPtr CreateRoundRectRgn
    (
        int nLeftRect,
        int nTopRect,
        int nRightRect,
        int nBottomRect,
        int nWidthEllipse,
        int nHeightEllipse
    );
    public Panel GetBigLabelPreview()
    {
        Label tagNameLabel = new Label();
        tagNameLabel.Font = new Font("Century Gothic", 9);
        tagNameLabel.Text = name;
        tagNameLabel.AutoSize = true;
        FlowLayoutPanel tagBigPanel = new FlowLayoutPanel();
        tagBigPanel.FlowDirection = FlowDirection.LeftToRight;
        tagBigPanel.AutoScroll = false;
        tagBigPanel.AutoSize = true;
        tagBigPanel.BackColor = GetColor();
        PictureBox deleteTag = new PictureBox();
        deleteTag.Image = Resources.closeDark16;
        deleteTag.Size = new Size(14, 14);
        deleteTag.Cursor = Cursors.Hand;
        deleteTag.SizeMode = PictureBoxSizeMode.StretchImage;
        deleteTag.Click += (sender, e) => { RemoveFromTask(); };
        tagBigPanel.Controls.Add(tagNameLabel);
        tagBigPanel.Controls.Add(deleteTag);
        tagBigPanel.SizeChanged += (sender, e) => {
            tagBigPanel.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0,
tagBigPanel.Width, tagBigPanel.Height, 4, 4));
        };
        return tagBigPanel;
    }
    private void RemoveFromTask()
    {
        TaskForm openedTF =
Application.OpenForms.OfType<TaskForm>().FirstOrDefault();
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("DELETE FROM `task-label` WHERE
`task` = @t AND `label` = @l", db.getConnection());
        command.Parameters.Add("@t", MySqlDbType.Int32).Value =
openedTF.task.id;
        command.Parameters.Add("@l", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
    }

```

```

        db.closeConnection();
        openedTF.UpdateTags();

        if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
    }
    public Panel GetSmallTag()
    {
        Panel tagPanel = new Panel();
        tagPanel.BackColor = GetColor();
        tagPanel.Size = new Size(10, 15);
        tagPanel.Cursor = Cursors.Hand;
        tagPanel.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0,
tagPanel.Width, tagPanel.Height, 4, 4));
        ToolTip tip = new ToolTip();
        tip.SetToolTip(tagPanel, $"{name}");
        return tagPanel;
    }
    public Panel GetLabelPanel()
    {
        Panel panel = new Panel();
        panel.Size = new Size(240, 35);
        panel.BackColor = GetColor();
        panel.Region = Region.FromHrgn(CreateRoundRectRgn(0, 0, panel.Width,
panel.Height, 10, 10));
        panel.Tag = id;
        CheckBox tagName = new CheckBox();
        tagName.Text = name;
        panel.Controls.Add(tagName);
        PictureBox deleteTag = new PictureBox();
        deleteTag.Image = Resources.deleteWhite;
        deleteTag.Size = new Size(20, 20);
        deleteTag.SizeMode = PictureBoxSizeMode.StretchImage;
        deleteTag.Tag = id;
        deleteTag.Cursor = Cursors.Hand;
        deleteTag.Click += (sender, e) =>
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("DELETE FROM `labels` WHERE
`id` = @tagID", db.getConnection());
            command.Parameters.Add("@tagID", MySqlDbType.Int32).Value =
((PictureBox)sender).Tag.ToString();
            db.openConnection();
            command.ExecuteNonQuery();
            db.closeConnection();
            ProjectForm projectForm =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
            projectForm.DeleteTagPanel(id);
        };
        panel.Controls.Add(deleteTag);
        deleteTag.Location = new Point(panel.Width - 30, 5);
        tagName.Location = new Point(10, 5);
        tagName.CheckedChanged += (sender, e) =>
        {
            ProjectForm projectForm =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
            DataTable tasks = GetTasksByTag();
            if (tagName.Checked)
            {
                foreach (DataRow taskRow in tasks.Rows)
                {
                    Task task = new Task(taskRow["task"].ToString());

```

```

projectForm.FillTasksPanel(task.GetExistingTaskPreviewPanel(false));
    }
    else
    {
        projectForm.DeleteTasksByTag(tasks);
    }
};
return panel;
}

public DataTable GetTasksByTag()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `task` FROM `task-label`
WHERE `label` = @labelID", db.getConnection());
    command.Parameters.Add("@labelID", MySqlDbType.Int32).Value = id;
    DataTable table = new DataTable();
    MySqlDataAdapter adapter = new MySqlDataAdapter();
    db.openConnection();
    adapter.SelectCommand = command;
    adapter.Fill(table);
    db.closeConnection();
    return table;
}
}
}
}

```

Приложение А.7. Программный код класса Attachment

```

using System;
using System.Data;
using System.Windows.Forms;
using System.IO;
using System.Drawing;
using TikTask.Properties;

namespace TikTask
{
    internal class Attachment
    {
        string id, name, task;
        byte[] data;
        public Attachment(DataRow attachmentRow)
        {
            this.id = attachmentRow["id"].ToString();
            this.name = attachmentRow["name"].ToString();
            this.task = attachmentRow["task"].ToString();
            this.data = (byte[])attachmentRow["data"];
        }
        public Panel GetAttachmentPreview()
        {
            Panel attachmentPanel = new Panel();
            attachmentPanel.Size = new Size(400, 30);
            Label attachmentName = new Label();
            attachmentName.Font = new Font("Century Gothic", 10);
            attachmentName.Size = new Size(370, 30);
            attachmentName.Text = name;
            attachmentName.AutoEllipsis = true;
            PictureBox download = new PictureBox();
            download.Image = Resources.downloadLight1;
            download.Cursor = Cursors.Hand;
            download.MouseEnter += (sender, e) => { download.Image =
Resources.downloadDark; };

```

```

        download.MouseLeave += (sender, e) => { download.Image =
Resources.downloadLight1; };
        download.SizeMode = PictureBoxSizeMode.StretchImage;
        download.Size = new Size(24, 24);
        download.Location = new Point(attachmentName.Location.X +
attachmentName.Width + 10, attachmentName.Location.Y);
        download.Click += Save_Click;
        attachmentPanel.Controls.Add(download);
        attachmentPanel.Controls.Add(attachmentName);
        return attachmentPanel;
    }

    private void Save_Click(object sender, EventArgs e)
    {
        SaveFileDialog saveFileDialog = new SaveFileDialog();
        saveFileDialog.Filter = "All Files (*.*)|*.*";
        saveFileDialog.RestoreDirectory = true;
        saveFileDialog.FileName = name;
        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
            string filePath = saveFileDialog.FileName;
            File.WriteAllBytes(filePath, data);
        }
    }
}

```

Приложение А.8. Программный код класса Notification

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TikTask
{
    public class Notification
    {
        public string id, receiver, message, sender, project;
        public bool received;
        public Notification(string id)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("SELECT * FROM `notification`
WHERE `id` = @notID", db.getConnection());
            command.Parameters.Add("@notID", MySqlDbType.Int32).Value = id;
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            adapter.SelectCommand = command;
            db.openConnection();
            adapter.Fill(table);
            db.closeConnection();
            this.id = id;
            foreach (DataRow row in table.Rows)
            {
                this.receiver = row["receiver"].ToString();
                this.message = row["message"].ToString();
            }
        }
    }
}

```

```

        this.sender = row["sender"].ToString();
        this.project = row["project"].ToString();
        this.received = Convert.ToBoolean(row["received"]);
    }
}
public Panel GetInvitationNotificationPanel()
{
    Panel notPanel = new Panel();
    notPanel.Size = new Size(350, 30);
    Label notLabel = new Label();
    User senUser = new User(sender);
    Project projectObj = new Project(project);
    MessageBox.Show($"{projectObj.name}");
    notLabel.Text = $"{senUser.mail} приглашает вас присоединиться к проекту
    \"{projectObj.name}\"";
    notLabel.AutoEllipsis = true;
    notLabel.Size = new Size(255, 30);
    notLabel.Font = new Font("Century Gothic", 8);
    notLabel.Location = new Point(default, 5);
    notPanel.Controls.Add(notLabel);
    Button acceptButton = new Button();
    acceptButton.Cursor = Cursors.Hand;
    acceptButton.Text = "Принять";
    acceptButton.Font = new Font("Century Gothic", 8);
    acceptButton.Size = new Size(60, 22);
    acceptButton.Click += (sender, e) =>
    {
        received = true;
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("UPDATE `notification` SET
        `received` = @status WHERE `id` = @notID", db.getConnection());
        command.Parameters.Add("@status", MySqlDbType.Int32).Value =
        received;
        command.Parameters.Add("@notID", MySqlDbType.Int32).Value = id;
        db.openConnection();
        command.ExecuteNonQuery();
        command = new MySqlCommand("INSERT INTO `involved` (`project`,
        `user`) VALUES (@projectID, @userID)", db.getConnection());
        command.Parameters.Add("@projectID", MySqlDbType.Int32).Value =
        projectObj.id;
        command.Parameters.Add("@userID", MySqlDbType.Int32).Value =
        receiver;
        command.ExecuteNonQuery();
        db.closeConnection();
    };
    notPanel.Controls.Add(acceptButton);
    acceptButton.Location = new Point(260, 1);
    return notPanel;
}
public void SetReceived()
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `notification` SET
    `received` = @status WHERE `id` = @notID", db.getConnection());
    command.Parameters.Add("@status", MySqlDbType.Int32).Value = 1;
    this.received = true;
    command.Parameters.Add("@notID", MySqlDbType.Int32).Value = id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
}
}
}
}

```

Приложение А.9. Программный код класса TagBoxElem

```
using System.Data;

namespace TikTask
{
    public class TagBoxElem
    {
        public string id, name;
        public TagBoxElem(DataRow source)
        {
            id = source["id"].ToString();
            name = source["name"].ToString();
        }
        public override string ToString()
        {
            return name;
        }
    }
}
```

Приложение А.10. Программный код класса Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Cryptography.X509Certificates;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TikTask
{
    internal static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new LoginForm());
        }
    }
}
```

Приложение А.11. Программный код формы MainForm

```
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public partial class MainForm : Form
    {
        public User curUser;
        public MainForm(User curUser)
        {
            this.curUser = curUser;
            InitializeComponent();
            greetingLabel.Text = Greeting();
            loginLabelMain.Text = curUser.Login;
        }
    }
}
```

```

        dataLabel.Text = $"{DateTime.Now.ToString("dddd")},
{DateTime.Now.ToString("M")}";
        UpdateTasks();
        UpdateProjects();
    }
    public void UpdateTasks()
    {
        tasksFlowLayoutPanel.Controls.Clear();
        DataTable curUserTasks = curUser.GetUserTasks();
        foreach (DataRow task in curUserTasks.Rows)
        {
            Task userTask = new Task(task);
            userTask.GetTaskPreviewPanel();

tasksFlowLayoutPanel.Controls.Add(userTask.GetExistingTaskPreviewPanel(false));
        }
    }
    public void UpdateProjects()
    {
        projectFlowLayout.Controls.Clear();
        DataTable curUserProjects = curUser.GetUserProjects();
        foreach (DataRow row in curUserProjects.Rows)
        {
            Project userProject = new Project(row["project"].ToString());
            projectFlowLayout.Controls.Add(userProject.GetPreviewCard());
        }
    }
    private string Greeting()
    {
        string name;
        if (curUser.name != "") { name = curUser.name; }
        else name = curUser.login;
        int time = Convert.ToInt32(DateTime.Now.ToString("HH"));
        if (time <= 4)
            return $"Доброй ночи, {name}";
        else if (time <= 11)
            return $"Доброе утро, {name}";
        else if (time <= 17)
            return $"Добрый день, {name}";
        else return $"Добрый вечер, {name}";
    }
    private void createProjectButtonFirst_Click(object sender, EventArgs e)
    {
        if (Application.OpenForms.OfType<CreateProjectForm>().FirstOrDefault()
!= null) return;
        CreateProjectForm createProjectForm = new CreateProjectForm(this);
        createProjectForm.Show();
    }
    private void userAccountButton_Click(object sender, EventArgs e)
    {
        UserForm userForm = new UserForm(curUser);
        userForm.Show();
        this.Hide();
    }
    private void closeButton_Click(object sender, EventArgs e)
    {
        Application.Exit();
    }
    Point clickPoint;
    private void topPanel_MouseClick(object sender, MouseEventArgs e)
    {
        clickPoint = new Point(e.X, e.Y);
    }
    private void topPanel_MouseMove(object sender, MouseEventArgs e)

```



```

    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - clickPoint.X;
            this.Top += e.Y - clickPoint.Y;
        }
    }
}

```

Приложение А.12. Программный код формы LoginForm

```

using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;

namespace TikTask
{
    public partial class LoginForm : Form
    {
        public LoginForm()
        {
            InitializeComponent();
        }
        private void minimizeButton_Click(object sender, EventArgs e) {
this.WindowState = FormWindowState.Minimized; }
        private void closeButton_Click(object sender, EventArgs e) {
Application.Exit(); }
        Point clickPoint;
        private void topPanel_MouseClick(object sender, MouseEventArgs e)
        {
            clickPoint = new Point(e.X, e.Y);
        }
        private void topPanel_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                this.Left += e.X - clickPoint.X;
                this.Top += e.Y - clickPoint.Y;
            }
        }
        private void Login_Click(object sender, EventArgs e)
        {
            string login = loginField.Text;
            string pass = passField.Text;

            DB db = new DB();
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE
`login` = @uL AND `password` = @uP", db.getConnection());

            command.Parameters.Add("@uL", MySqlDbType.VarChar).Value = login;
            command.Parameters.Add("@uP", MySqlDbType.VarChar).Value = pass;

            adapter.SelectCommand = command;
            adapter.Fill(table);

            if (table.Rows.Count > 0)
            {
                string i = table.Rows[0][0].ToString();
                string l = table.Rows[0][1].ToString();
                string n = table.Rows[0][3].ToString();
            }
        }
    }
}

```

```

        string s = table.Rows[0][4].ToString();
        string m = table.Rows[0][5].ToString();
        User user = new User(i, n, s, l, m);
        MainForm mainForm = new MainForm(user);
        mainForm.Show();
        this.Hide();
    }
    else
        MessageBox.Show("Неверный логин или пароль");
}
private void signUpButton_Click(object sender, EventArgs e)
{
    this.Hide();
    SignUpForm signUpForm = new SignUpForm();
    signUpForm.Show();
}
}
}

```

Приложение А.13. Программный код формы SignUpForm

```

using MySql.Data.MySqlClient;
using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public partial class SignUpForm : Form
    {
        public SignUpForm()
        {
            InitializeComponent();
        }
        private void closeButton_Click(object sender, EventArgs e)
        {
            Application.Exit(); ;
        }

        Point clickPoint;
        private void topPanel_MouseClick(object sender, MouseEventArgs e)
        {
            clickPoint = new Point(e.X, e.Y);
        }
        private void topPanel_MouseMove(object sender, MouseEventArgs e)
        {
            if (e.Button == MouseButtons.Left)
            {
                this.Left += e.X - clickPoint.X;
                this.Top += e.Y - clickPoint.Y;
            }
        }
        private void Register_Click(object sender, EventArgs e)
        {
            if (loginField_SA.Text == "" || passField_SA.Text == "" ||
mailField_SA.Text == "" )
            {
                MessageBox.Show("Пожалуйста, заполните обязательные поля");
                return;
            }
            if (IsUserExist() || IsMailExist()) { return; }
            User newUser = new User();

```

```

        if (newUser.AddUser(loginField_SA, passField_SA, nameField_SA,
surnameField_SA, mailField_SA))
        {
            MessageBox.Show("Аккаунт создан.");
            this.Hide();
            LoginForm loginForm = new LoginForm();
            loginForm.Show();
        }
        else
            MessageBox.Show("Что-то пошло не так");
    }
    public bool IsUserExist()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE
`login` = @uL", db.getConnection());

        command.Parameters.Add("@uL", MySqlDbType.VarChar).Value =
loginField_SA.Text;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        if (table.Rows.Count > 0)
        {
            MessageBox.Show("Логин уже занят");
            return true;
        }
        else
            return false;
    }

    public bool IsMailExist()
    {
        DB db = new DB();
        DataTable table = new DataTable();
        MySqlDataAdapter adapter = new MySqlDataAdapter();
        MySqlCommand command = new MySqlCommand("SELECT * FROM `users` WHERE
`mail` = @uM", db.getConnection());

        command.Parameters.Add("@uM", MySqlDbType.VarChar).Value =
mailField_SA.Text;

        adapter.SelectCommand = command;
        adapter.Fill(table);

        if (table.Rows.Count > 0)
        {
            MessageBox.Show("Пользователь с такой почтой уже существует.");
            return true;
        }
        else
            return false;
    }
    private void backToLoginButton_Click(object sender, EventArgs e)
    {
        this.Hide();
        if (Application.OpenForms.OfType<LoginForm>().Count() > 0)
            Application.OpenForms.OfType<LoginForm>().FirstOrDefault().Show();
        else
        {
            LoginForm loginForm = new LoginForm();

```

```

        loginForm.Show();
    }
}
}
}

```

Приложение А.14. Программный код формы UserForm

```

using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public partial class UserForm : Form
    {
        User curUser;
        public UserForm(User curUser)
        {
            this.curUser = curUser;
            InitializeComponent();
            foreach (DataRow notRow in curUser.GetUserNotifications().Rows)
            {
                Notification not = new Notification(notRow["id"].ToString());
                if (not.received == false)
                {
                    notifFlowLayout.Controls.Add(not.GetInvitationNotificationPanel());
                }
                nameLabel.Text = curUser.name;
                surnameLabel.Text = curUser.surname;
                loginLabel.Text = curUser.login;
                mailLabel.Text = curUser.mail;
            }
            private void Update()
            {
                nameLabel.Text = curUser.name;
                surnameLabel.Text = curUser.surname;
                loginLabel.Text = curUser.login;
                mailLabel.Text = curUser.mail;
            }
            private void pictureBox3_Click(object sender, EventArgs e)
            {
                this.Close();
                MainForm main = Application.OpenForms.OfType<MainForm>().First();
                main.Show();
            }
            private void editNameButton_Click(object sender, EventArgs e)
            {
                TextBox nameField = new TextBox();
                nameField = new TextBox();
                nameField.Size = namePanel.Size;
                nameField.Location = namePanel.Location;
                nameField.Text = curUser.name;
                nameField.Font = new Font("Century Gothic", 10);
                nameField.KeyPress += (sender2, e2) =>
                {
                    if (e2.KeyChar == (char)Keys.Enter)
                    {
                        curUser.ChangeName(nameField.Text);
                        Controls.Remove(nameField);
                    }
                }
            }
        }
    }
}

```

```

        Controls.Add(namePanel);
        editNameButton.Click += editNameButton_Click;
        Update();
    }
};
editNameButton.Click -= editNameButton_Click;
Controls.Remove(namePanel);
Controls.Add(nameField);
}

private void editSurnameButton_Click(object sender, EventArgs e)
{
    TextBox surnameField = new TextBox();
    surnameField = new TextBox();
    surnameField.Size = surnamePanel.Size;
    surnameField.Location = surnamePanel.Location;
    surnameField.Text = curUser.surname;
    surnameField.Font = new Font("Century Gothic", 10);
    surnameField.KeyPress += (sender2, e2) =>
    {
        if (e2.KeyChar == (char)Keys.Enter)
        {
            curUser.ChangeSurname(surnameField.Text);
            Controls.Add(surnamePanel);
            Controls.Remove(surnameField);
            editSurnameButton.Click += editSurnameButton_Click;
            Update();
        }
    };
    editSurnameButton.Click -= editSurnameButton_Click;
    Controls.Remove(surnamePanel);
    Controls.Add(surnameField);
}

private void editNameButton_MouseEnter(object sender, EventArgs e)
{
    editNameButton.Image = Resources.editDark32;
}

private void editNameButton_MouseLeave(object sender, EventArgs e)
{
    editNameButton.Image = Resources.editLight32;
}

private void editSurnameButton_MouseEnter(object sender, EventArgs e)
{
    editSurnameButton.Image = Resources.editDark32;
}

private void editSurnameButton_MouseLeave(object sender, EventArgs e)
{
    editSurnameButton.Image = Resources.editLight32;
}

private void homeButton_Click(object sender, EventArgs e)
{
    this.Close();
    MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
    main.Show();
}

private void changePasswordButton_Click(object sender, EventArgs e)
{

```

```

        if (Application.OpenForms.OfType<ChangePass>().FirstOrDefault() != null)
return;
        ChangePass changePassForm = new ChangePass(curUser);
        changePassForm.Show();
    }

    private void endSessionButton_Click(object sender, EventArgs e)
    {
        LoginForm loginForm = new LoginForm();
        loginForm.Show();
        (Application.OpenForms.OfType<MainForm>().FirstOrDefault()).Close();
        (Application.OpenForms.OfType<UserForm>().FirstOrDefault()).Close();
        if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null) Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().Close();
    }
}
}
}

```

Приложение А.15. Программный код формы ChangePass

```

using MySql.Data.MySqlClient;
using System;
using System.Drawing;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public partial class ChangePass : Form
    {
        User curUser;
        public ChangePass(User curUser)
        {
            InitializeComponent();
            this.curUser = curUser;
            this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
        }

        private void changePassButton_Click(object sender, EventArgs e)
        {
            if (CheckCondition())
            {
                DB db = new DB();
                MySqlCommand command = new MySqlCommand("UPDATE `users` SET
`password` = @newPass WHERE `id` = @userID", db.getConnection());
                command.Parameters.Add("@newPass", MySqlDbType.VarChar).Value =
newPassField1.Text;
                command.Parameters.Add("@userID", MySqlDbType.Int32).Value =
curUser.id;
                db.openConnection();
                command.ExecuteNonQuery();
                db.closeConnection();
                this.Close();
            }
        }

        private bool CheckCondition()
        {
            if (oldPassField.Text == "" || newPassField1.Text == "" ||
newPassField2.Text == "")
            {
                MessageBox.Show("Пожалуйста, заполните обязательные поля.");
                return false;
            }
            if (newPassField1.Text != newPassField2.Text)

```

```

        {
            MessageBox.Show("Пароли не совпадают.");
            return false;
        }
        DB db = new DB();
        MySqlCommand command = new MySqlCommand("SELECT `password` FROM `users`
WHERE `login` = @uL", db.getConnection());
        command.Parameters.Add("@uL", MySqlDbType.VarChar).Value =
curUser.login;
        db.openConnection();
        if (command.ExecuteScalar().ToString() != oldPassField.Text)
        {
            MessageBox.Show("Старый пароль неверен.");
            return false;
        }
        db.closeConnection();
        return true;
    }

    private void closeButton_Click(object sender, EventArgs e)
    {
        this.Close();
    }

    private void closeButton_MouseEnter(object sender, EventArgs e)
    {
        closeButton.Image = Resources.closeDark16;
    }

    private void closeButton_MouseLeave(object sender, EventArgs e)
    {
        closeButton.Image = Resources.closeLight32;
    }
}
}
}

```

Приложение А.16. Программный код формы CreateProjectForm

```

using System;
using System.Drawing;
using System.Windows.Forms;

namespace TikTask
{
    public partial class CreateProjectForm : Form
    {
        MainForm main;
        public CreateProjectForm(MainForm main)
        {
            this.main = main;
            InitializeComponent();
            Location = new Point(Cursor.Position.X, Cursor.Position.Y);
        }
        private void createProjectButton_Click(object sender, EventArgs e)
        {
            Project newProject = new Project(main, nameField.Text,
startDatePicker.Value, finishDatePicker.Value, main.curUser.id);
            newProject.PushToDB();
            main.UpdateProjects();
            this.Close();
        }
        private void pictureBox1_Click(object sender, EventArgs e)
        {
            this.Close();
        }
    }
}

```

```

    }
}

```

Приложение А.17. Программный код формы InviteUserForm

```

using MySql.Data.MySqlClient;
using System;
using System.Drawing;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public partial class InviteUserForm : Form
    {
        User curUser;
        Project curProject;
        public InviteUserForm(User curUser, Project project)
        {
            this.curUser = curUser;
            this.curProject = project;
            InitializeComponent();
            Location = new Point(Cursor.Position.X - this.Width, Cursor.Position.Y);
        }

        private void closeButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void closeButton_MouseEnter(object sender, EventArgs e)
        {
            closeButton.Image = Resources.closeDark32;
        }

        private void closeButton_MouseLeave(object sender, EventArgs e)
        {
            closeButton.Image = Resources.closeLight32;
        }

        private void inviteButton_Click(object sender, EventArgs e)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("SELECT `id` FROM `users` WHERE
`mail` = @userMail", db.getConnection());
            command.Parameters.Add("@userMail", MySqlDbType.VarChar).Value =
userMailField.Text;
            db.openConnection();
            if (command.ExecuteScalar() == null)
            {
                MessageBox.Show("Пользователь не найден");
                db.closeConnection();
            }
            else
            {
                string id = command.ExecuteScalar().ToString();
                db.closeConnection();
                command = new MySqlCommand("INSERT INTO `notification` (`receiver`,
`message`, `sender`, `received`, `project`) VALUES (@receiver, @message, @sender,
@received, @project)", db.getConnection());
                command.Parameters.Add("@receiver", MySqlDbType.Int32).Value = id;
                command.Parameters.Add("@message", MySqlDbType.LongText).Value =
"Пользователь {sender} приглашает вас присоединиться к проекту";
                command.Parameters.Add("@sender", MySqlDbType.Int32).Value =
curUser.id;
            }
        }
    }
}

```



```

        command.Parameters.Add("@received", MySqlDbType.Int32).Value = 0;
        command.Parameters.Add("@project", MySqlDbType.Int32).Value =
curProject.id;
        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
        MessageBox.Show("Приглашение отправлено");
        this.Close();
    }
}
}
}

```

Приложение А.18. Программный код формы ProjectForm

```

using System;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using TikTask.Properties;

namespace TikTask
{
    public partial class ProjectForm : Form
    {
        public Project curProject;
        TextBox stageName;
        public ProjectForm(Project project)
        {
            InitializeComponent();
            this.curProject = project;
            nameLabel.Text = curProject.name;
            stagesPanel.Controls.Remove(addStageButton);
            foreach (DataRow stage in curProject.GetProjectsStages().Rows)
            {
                Stage new_stage = new Stage(stage);
                FlowLayoutPanel new_stagePanel =
new_stage.GetStagePanel(stagesPanel.Height - 50);
                foreach (DataRow task in new_stage.GetStagesTasks().Rows)
                {
                    Task new_task = new Task(task);
                    Panel new_taskPanel =
new_task.GetExistingTaskPreviewPanel(true);
                    new_stagePanel.Controls.Add(new_taskPanel);
                }
                stagesPanel.Controls.Add(new_stagePanel);
            }
            stagesPanel.Controls.Add(addStageButton);
        }
        private void homeButton_Click(object sender, EventArgs e)
        {
            this.Close();
            MainForm mainForm =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
            mainForm.Show();
        }

        private void closeButton_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void addStageButton_Click(object sender, EventArgs e)
        {

```

```

stageName = new TextBox();
stageName.KeyPress += (sender2, e2) =>
{
    if (e2.KeyChar == (char)Keys.Enter)
    {
        Stage stage = new Stage(curProject.main, this, stageName.Text,
curProject.host, null);
        stage.PushToDB(stageName);
        stagesPanel.Controls.Add(stage.GetStagePanel(stagesPanel.Height
- 50));

        stagesPanel.Controls.Remove(stageName);
        stagesPanel.Controls.Add(addStageButton);
        stagesPanel.Controls.Add(addStageButton);
    }
};
stageName.Width = 300;
stageName.Height = 30;
stageName.Font = new Font("Century Gothic", 12);
stagesPanel.Controls.Remove(addStageButton);
stagesPanel.Controls.Add(stageName);
}

public void UpdateProjectView()
{
    if (visualTabs.SelectedTab == tabPage3)
    {
        dayTaskFlowLayout.Controls.Clear();
        noDeadlineTasksLayout.Controls.Clear();
        MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
        DataTable tasks = main.curUser.GetUserTasksByDate(DateTime.Today);
        if (tasks.Rows.Count == 0)
dayTaskFlowLayout.Controls.Add(zeroTaskLabel);
        foreach (DataRow row in tasks.Rows)
        {
            Task newTask = new Task(row);

dayTaskFlowLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
        }
        tasks = main.curUser.GetUserTasksByDate(DateTime.MinValue);
        foreach (DataRow row in tasks.Rows)
        {
            Task newTask = new Task(row);

noDeadlineTasksLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
        }
    }
    stagesPanel.Controls.Clear();
    foreach (DataRow stage in curProject.GetProjectsStages().Rows)
    {
        Stage new_stage = new Stage(stage);
        FlowLayoutPanel new_stagePanel =
new_stage.GetStagePanel(stagesPanel.Height - 50);
        foreach (DataRow task in new_stage.GetStagesTasks().Rows)
        {
            Task new_task = new Task(task);
            Panel new_taskPanel =
new_task.GetExistingTaskPreviewPanel(true);
            new_stagePanel.Controls.Add(new_taskPanel);
        }
        stagesPanel.Controls.Add(new_stagePanel);
    }
    stagesPanel.Controls.Add(addStageButton);
}
}

```

```

private void inviteUserButton_MouseEnter(object sender, EventArgs e)
{
    inviteUserButton.Image = Resources.inviteDark;
}

private void inviteUserButton_MouseLeave(object sender, EventArgs e)
{
    inviteUserButton.Image = Resources.inviteLight;
}

private void inviteUserButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<InviteUserForm>().FirstOrDefault() !=
null) Application.OpenForms.OfType<InviteUserForm>().FirstOrDefault().Show(); ;
    MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
    InviteUserForm inviteForm = new InviteUserForm(main.curUser,
curProject);
    inviteForm.Show();
}

private void monthCalendar_DateSelected(object sender, DateRangeEventArgs e)
{
    taskDateLabel.Text = $"Задачи на
{monthCalendar.SelectionEnd.ToString().Split()[0]}";
    dayTaskFlowLayout.Controls.Clear();
    MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
    DataTable tasks =
main.curUser.GetUserTasksByDate(DateTime.Parse(monthCalendar.SelectionEnd.ToString()
.Split()[0]));
    if (tasks.Rows.Count == 0)
dayTaskFlowLayout.Controls.Add(zeroTaskLabel);
    foreach (DataRow row in tasks.Rows)
    {
        Task newTask = new Task(row);

dayTaskFlowLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
    }
    noDeadlineTasksLayout.Controls.Clear();
    tasks = main.curUser.GetUserTasksByDate(DateTime.MinValue);
    foreach (DataRow row in tasks.Rows)
    {
        Task newTask = new Task(row);

noDeadlineTasksLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
    }
}

private void visualTabs_Selected(object sender, TabControlEventArgs
e)
{
    if (visualTabs.SelectedTab == tabPage3)
    {
        dayTaskFlowLayout.Controls.Clear();
        taskDateLabel.Text = $"Задачи на
{DateTime.Today.ToString().Split()[0]}";
        MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
        DataTable tasks = main.curUser.GetUserTasksByDate(DateTime.Today);
        if (tasks.Rows.Count == 0)
dayTaskFlowLayout.Controls.Add(zeroTaskLabel);
        foreach (DataRow row in tasks.Rows)
        {
            Task newTask = new Task(row);

```

```

dayTaskFlowLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
    }
    noDeadlineTasksLayout.Controls.Clear();
    tasks = main.curUser.GetUserTasksByDate(DateTime.MinValue);
    foreach (DataRow row in tasks.Rows)
    {
        Task newTask = new Task(row);
    }
noDeadlineTasksLayout.Controls.Add(newTask.GetExistingTaskPreviewPanel(false));
    }
    if (visualTabs.SelectedTab == tabPage4)
    {
        tagsFlowLayout.Controls.Clear();
        tasksFlowLayout.Controls.Clear();
        DataTable tags = curProject.GetProjectLabels();
        foreach (DataRow tagRow in tags.Rows)
        {
            Tag tag = new Tag(tagRow);
            tagsFlowLayout.Controls.Add(tag.GetLabelPanel());
        }
    }
}
public void FillTasksPanel(Panel taskCard)
{
    foreach (Panel panel in tasksFlowLayout.Controls)
    {
        if (taskCard.Tag.ToString() == panel.Tag.ToString()) return;
    }
    tasksFlowLayout.Controls.Add(taskCard);
}
public void DeleteTasksByTag(DataTable tasks)
{
    foreach (DataRow taskRow in tasks.Rows)
    {
        foreach (Panel panel in tasksFlowLayout.Controls)
        {
            if (panel.Tag.ToString() == taskRow["task"].ToString())
            {
                tasksFlowLayout.Controls.Remove(panel);
            }
        }
    }
}
public void DeleteTagPanel(string id)
{
    foreach (Panel panel in tagsFlowLayout.Controls)
    {
        if (panel.Tag.ToString() == id)
        {
            foreach (Panel taskPanel in tasksFlowLayout.Controls)
            {
                if (taskPanel.Tag.ToString() == id)
                {
                    tasksFlowLayout.Controls.Remove(taskPanel);
                }
            }
            tagsFlowLayout.Controls.Remove(panel);
            UpdateProjectView();
        }
    }
}
}

```

```

private void pictureBox1_Click(object sender, EventArgs e)
{
    NewLabelForm labelForm = new NewLabelForm();
    labelForm.Show();
    UpdateProjectView();
}
private void editProjectButton_Click(object sender, EventArgs e)
{
    if (projectPanel.Controls.OfType<TextBox>().Count() != 0) return;
    TextBox editProjectField = new TextBox();
    editProjectField.Size = nameLabel.Size;
    editProjectField.Text = nameLabel.Text;
    editProjectField.Location = nameLabel.Location;
    editProjectField.Font = nameLabel.Font;
    projectPanel.Controls.Remove(nameLabel);
    projectPanel.Controls.Add(editProjectField);
    editProjectField.KeyPress += (sender2, e2) =>
    {
        if (e2.KeyChar == (char)Keys.Enter)
        {
            curProject.EditName(editProjectField.Text);
            nameLabel.Text = editProjectField.Text;
            projectPanel.Controls.Remove(editProjectField);
            projectPanel.Controls.Add(nameLabel);
            MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
            main.UpdateProjects();
        }
    };
}

private void editProjectButton_MouseEnter(object sender, EventArgs e)
{
    editProjectButton.Image = Resources.editDark32;
}

private void editProjectButton_MouseLeave(object sender, EventArgs e)
{
    editProjectButton.Image = Resources.editLight32;
}

private void deleteProjectButton_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить
проект?", "Подтверждение удаления", MessageBoxButtons.OKCancel);

    if (result == DialogResult.OK)
    {
        curProject.Delete();
        MainForm main =
Application.OpenForms.OfType<MainForm>().FirstOrDefault();
        main.UpdateProjects();
        main.UpdateTasks();
        main.Show();
        this.Close();
    }
    else if (result == DialogResult.Cancel)
    {
        return;
    }
}

private void deleteProjectButton_MouseEnter(object sender, EventArgs e)

```

```

{
    deleteProjectButton.Image = Resources.deleteDark321;
}

private void deleteProjectButton_MouseLeave(object sender, EventArgs e)
{
    deleteProjectButton.Image = Resources.deleteLight321;
}
Point clickPoint;
private void topPanel_MouseClick(object sender, MouseEventArgs e)
{
    clickPoint = new Point(e.X, e.Y);
}
private void topPanel_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - clickPoint.X;
        this.Top += e.Y - clickPoint.Y;
    }
}
}
}

```

Приложение А.19. Программный код формы TaskForm

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;
using TikTask.Properties;
using System.IO;

namespace TikTask
{
    public partial class TaskForm : Form
    {
        public Task task;
        Project project;
        LabelForm labelForm;
        public TaskForm(Task task)
        {
            this.task = task;
            InitializeComponent();
            taskNameLabel.Text = task.name;
            Stage stage = new Stage(task.stage);
            project = new Project(stage.project);
            if (task.responsible != null) label7.Text = task.responsible;
            if (task.startDate != DateTime.MinValue) label12.Text =
task.startDate.ToString().Split()[0];
            if (task.finishDate != DateTime.MinValue) label13.Text =
task.finishDate.ToString().Split()[0];
            stageNameLabel.Text = stage.name;
            projectNameLabel.Text = project.name;
            if (task.description != null) descriptionBox.Text = task.description;
            DataTable tags = task.GetTasksTags();
            foreach (DataRow tagRow in tags.Rows)
            {
                Tag tag = new Tag(tagRow["Label"].ToString());
                Panel tagPanel = tag.GetBigLabelPreview();
                tagsFlowLayout.Controls.Add(tagPanel);
            }
        }
    }
}

```

```

        UpdateAttachedFiles();
        labelForm = new LabelForm(task);
    }
    public void UpdateResponsible()
    {
        label7.Text = task.responsible;
    }
    public void UpdateDates()
    {
        label12.Text = task.startDate.ToString().Split()[0];
        label13.Text = task.finishDate.ToString().Split()[0];
    }
    public void UpdateTags()
    {
        tagsFlowLayout.Controls.Clear();
        tagsFlowLayout.Controls.Add(addLabel);
        DataTable tags = task.GetTasksTags();
        foreach (DataRow tagRow in tags.Rows)
        {
            Tag tag = new Tag(tagRow["label"].ToString());
            Panel tagPanel = tag.GetBigLabelPreview();
            tagsFlowLayout.Controls.Add(tagPanel);
        }
    }
    private void closeButton_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void editNameButton_MouseEnter(object sender, EventArgs e)
    {
        editNameButton.Image = Resources.editDark32;
    }
    private void editNameButton_MouseLeave(object sender, EventArgs e)
    {
        editNameButton.Image = Resources.editLight32;
    }
    private void deleteButton_MouseEnter(object sender, EventArgs e)
    {
        deleteButton.Image = Resources.deleteDark321;
    }
    private void deleteButton_MouseLeave(object sender, EventArgs e)
    {
        deleteButton.Image = Resources.deleteLight321;
    }
    private void closeButton_MouseEnter(object sender, EventArgs e)
    {
        closeButton.Image = Resources.closeDark32;
    }
    private void closeButton_MouseLeave(object sender, EventArgs e)
    {
        closeButton.Image = Resources.closeLight32;
    }
    private void addResponsButton_MouseEnter(object sender, EventArgs e)
    {
        addResponsButton.Image = Resources.add_user_dark;
    }
    private void addResponsButton_MouseLeave(object sender, EventArgs e)
    {
        addResponsButton.Image = Resources.add_user_light;
    }
    private void editFinishDateButton_MouseEnter(object sender, EventArgs e)
    {
        editFinishDateButton.Image = Resources.editDark32;
    }
}

```

```

private void editFinishDateButton_MouseLeave(object sender, EventArgs e)
{
    editFinishDateButton.Image = Resources.editLight32;
}
private void editStageButton_MouseEnter(object sender, EventArgs e)
{
    editStageButton.Image = Resources.editDark32;
}
private void editStageButton_MouseLeave(object sender, EventArgs e)
{
    editStageButton.Image = Resources.editLight32;
}
private void addAttachmentButton_MouseEnter(object sender, EventArgs e)
{
    addAttachmentButton.Image = Resources.addDark32;
}
private void addAttachmentButton_MouseLeave(object sender, EventArgs e)
{
    addAttachmentButton.Image = Resources.addLight32;
}

private void addResponsButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<AddRespForm>().Count() > 0) return;
    AddRespForm form = new AddRespForm(task, this);
    form.Show();
}

private void editFinishDateButton_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<AddDateForm>().Count() > 0) return;
    AddDateForm form = new AddDateForm(task, this);
    form.Show();
}
ComboBox stages;
private void editStageButton_Click(object sender, EventArgs e)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("SELECT `project` FROM `stages`
WHERE `id` = @sI", db.getConnection());
    command.Parameters.Add("@sI", MySqlDbType.Int32).Value = task.stage;
    db.openConnection();
    object projId = command.ExecuteScalar();
    db.closeConnection();
    Project project = new Project(projId.ToString());
    DataTable stagesTable = project.GetProjectsStages();
    List<TagBoxElem> stagesList = new List<TagBoxElem>();
    foreach (DataRow row in stagesTable.Rows)
    {
        stagesList.Add(new TagBoxElem(row));
    }
    stages = new ComboBox();
    stages.ValueMember = "id";
    stages.Location = new Point(313, 143);
    stages.Size = new Size(117, 24);
    stages.DataSource = stagesList;
    stages.Text = stageNameLabel.Text;
    stages.Font = new Font("Century Gothic", 10);
    Controls.Add(stages);
    stages.BringToFront();
    stages.KeyPress += (sender2, e2) =>
    {
        if (e2.KeyChar == (char)Keys.Enter)
        {

```



```

        task.UpdateStage(((TagBoxElem)stages.SelectedValue).id);
        stageNameLabel.Text = stages.SelectedItem.ToString();
        ProjectForm projectForm =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
        projectForm.UpdateProjectView();
        Controls.Remove(stages);
    }
};

}

private void editDescription_Click(object sender, EventArgs e)
{
    descriptionBox.ReadOnly = false;
    descriptionBox.Enabled = true;
}

private void saveDescription_Click(object sender, EventArgs e)
{
    descriptionBox.ReadOnly = true;
    descriptionBox.Enabled = false;
    task.description = descriptionBox.Text;
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET
`description` = @tD WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("@tD", MySqlDbType.LongText).Value =
descriptionBox.Text;
    command.Parameters.Add("@tI", MySqlDbType.LongText).Value = task.id;
    db.openConnection();
    command.ExecuteNonQuery();
    db.closeConnection();
}

private void editDescription_MouseEnter(object sender, EventArgs e)
{
    editDescription.Image = Resources.editDark32;
}

private void editDescription_MouseLeave(object sender, EventArgs e)
{
    editDescription.Image = Resources.editLight32;
}

private void saveDescription_MouseEnter(object sender, EventArgs e)
{
    saveDescription.Image = Resources.saveDark;
}

private void saveDescription_MouseLeave(object sender, EventArgs e)
{
    saveDescription.Image = Resources.saveLight32;
}

private void addLabel_Click(object sender, EventArgs e)
{
    if (Application.OpenForms.OfType<LabelForm>().Count() > 0)
Application.OpenForms.OfType<LabelForm>().FirstOrDefault().Show();
    labelForm = new LabelForm(task);
    labelForm.Show();
}

private void deleteButton_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Вы уверены, что хотите удалить
задачу?", "Подтверждение удаления", MessageBoxButtons.OKCancel);

    if (result == DialogResult.OK)
    {
        task.Delete();
        ProjectForm openedPF =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();

```

```

        openedPF.UpdateProjectView();
        this.Close();
    }
    else if (result == DialogResult.Cancel)
    {
        return;
    }
}

private void editNameButton_Click(object sender, EventArgs e)
{
    Controls.Remove(taskNameLabel);
    TextBox taskNameField;
    taskNameField = new TextBox();
    taskNameField.Font = new Font("Century Gothic", 16);
    taskNameField.Location = new Point(58, 5);
    taskNameField.Size = new Size(286, 37);
    taskNameField.Text = task.name;
    taskNameField.KeyPress += (sender2, e2) =>
    {
        if (e2.KeyChar == (char)Keys.Enter)
        {
            task.UpdateName(taskNameField.Text);
            taskNameLabel.Text = task.name;
            Controls.Add(taskNameLabel);
            Controls.Remove(taskNameField);
            if (Application.OpenForms.OfType<ProjectForm>().Count() != 0)
                Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
        }
    };
    this.Controls.Add(taskNameField);
}

private void addAttachmentButton_Click(object sender, EventArgs e)
{
    openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Multiselect = true;

    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        foreach (string filePath in openFileDialog1.FileNames)
        {
            byte[] fileData = File.ReadAllBytes(filePath);
            SaveFileToDatabase(filePath.Split('\\').Last(), fileData);
        }
    }
    UpdateAttachedFiles();
}

private void SaveFileToDatabase(string fileName, byte[] fileData)
{
    DB db = new DB();
    MySqlCommand command = new MySqlCommand("INSERT INTO `attachments`
(`name`, `data`, `task`) VALUES (@aN, @aD, @aT)", db.getConnection());
    command.Parameters.Add("@aN", MySqlDbType.VarChar).Value = fileName;
    command.Parameters.Add("@aD", MySqlDbType.LongBlob).Value = fileData;
    command.Parameters.Add("@aT", MySqlDbType.Int32).Value = task.id;
    db.openConnection();
    try { command.ExecuteNonQuery(); }
    catch
    {
        MessageBox.Show("Превышен допустимый размер файла");
    }
    db.closeConnection();
}

private void UpdateAttachedFiles()

```

```

    {
        attachmentsFlowLayout.Controls.Clear();
        foreach (DataRow attachmentRow in task.GetAttachments().Rows)
        {
            Attachment file = new Attachment(attachmentRow);
            attachmentsFlowLayout.Controls.Add(file.GetAttachmentPreview());
        }
    }
    private void addLabel_MouseEnter(object sender, EventArgs e)
    {
        addLabel.Image = Resources.addDark32;
    }

    private void addLabel_MouseLeave(object sender, EventArgs e)
    {
        addLabel.Image = Resources.addLight;
    }
}
}

```

Приложение А.20. Программный код формы LabelForm

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public partial class LabelForm : Form
    {
        Task task;
        public LabelForm(Task task)
        {
            this.task = task;
            this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
            InitializeComponent();
            List<TagBoxElem> labelsBoxSource = new List<TagBoxElem>();
            foreach (DataRow labelRow in task.GetProject().GetProjectLabels().Rows)
            {
                labelsBoxSource.Add(new TagBoxElem(labelRow));
            }
            labelsBox.ValueMember = "id";
            labelsBox.DataSource = labelsBoxSource;
        }

        private void newLabelButton_Click(object sender, EventArgs e)
        {
            this.Hide();
            NewLabelForm nLF = new NewLabelForm();
            nLF.Location = this.Location;
            nLF.Show();
        }

        private void setLabelButton_Click(object sender, EventArgs e)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("INSERT INTO `task-label`
(`task`, `label`) VALUES (@t, @l)", db.getConnection());
            command.Parameters.Add("@t", MySqlDbType.VarChar).Value = task.id;
            command.Parameters.Add("@l", MySqlDbType.Int32).Value =
((TagBoxElem)labelsBox.SelectedValue).id;
        }
    }
}

```

```

        db.openConnection();
        command.ExecuteNonQuery();
        db.closeConnection();
        this.Close();
        TaskForm openedTF =
Application.OpenForms.OfType<TaskForm>().FirstOrDefault();
        openedTF.UpdateTags();
        if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
    }

    private void closeButton_Click(object sender, EventArgs e)
    {
        this.Close();
    }
}
}

```

Приложение A.21. Программный код формы NewLabelForm

```

using MySql.Data.MySqlClient;
using System;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public partial class NewLabelForm : Form
    {
        Color color;
        public NewLabelForm()
        {
            InitializeComponent();
        }
        private void closeButton_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void setLabelButton_Click(object sender, EventArgs e)
        {
            DB db = new DB();
            MySqlCommand command = new MySqlCommand("INSERT INTO `labels` (`name`,
`project`, `color`) VALUES (@LN, @LP, @LC)", db.getConnection());
            command.Parameters.Add("@LN", MySqlDbType.VarChar).Value =
newTagNameTextBox.Text;
            ProjectForm pForm =
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault();
            command.Parameters.Add("@LP", MySqlDbType.Int32).Value =
pForm.curProject.id;
            command.Parameters.Add("@LC", MySqlDbType.VarChar).Value =
color.ToArgb().ToString();
            db.openConnection();
            command.ExecuteNonQuery();
            db.closeConnection();
            this.Close();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            if (colorDialog1.ShowDialog() == DialogResult.OK)
            {
                color = colorDialog1.Color;
            }
        }
    }
}

```

```

        colorPreview.BackColor = color;
    }
}
}
}

```

Приложение А.22. Программный код формы AddRespForm

```

using MySql.Data.MySqlClient;
using System;
using System.Collections.Generic;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public partial class AddRespForm : Form
    {
        public Task task;
        public TaskForm taskForm;
        public AddRespForm(Task task)
        {
            InitializeComponent();
            this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
            this.task = task;
            responsibleBox.DataSource = GetProjectMembers();
        }
        public AddRespForm(Task task, TaskForm taskForm)
        {
            InitializeComponent();
            this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
            this.task = task;
            this.taskForm = taskForm;
            responsibleBox.DataSource = GetProjectMembers();
        }
        private List<string> GetProjectMembers()
        {
            List<string> members = new List<string>();
            string stage, project, mail;
            DB db = new DB();

            db.openConnection();

            MySqlCommand command = new MySqlCommand("SELECT `stage` FROM `tasks`
WHERE `id` = @tI", db.getConnection());
            command.Parameters.Add("@tI", MySqlDbType.Int32).Value = task.id;
            object result = command.ExecuteScalar();
            stage = result.ToString();

            command = new MySqlCommand("SELECT `project` FROM `stages` WHERE `id` =
@sI", db.getConnection());
            command.Parameters.Add("@sI", MySqlDbType.Int32).Value = stage;
            project = command.ExecuteScalar().ToString();

            command = new MySqlCommand("SELECT * FROM `involved` WHERE `project` =
@pI", db.getConnection());
            command.Parameters.Add("@pI", MySqlDbType.Int32).Value = project;
            DataTable table = new DataTable();
            MySqlDataAdapter adapter = new MySqlDataAdapter();
            adapter.SelectCommand = command;
            adapter.Fill(table);
        }
    }
}

```

```

        foreach (DataRow row in table.Rows)
        {
            command = new MySqlCommand("SELECT `mail` FROM `users` WHERE `id` =
@uI", db.getConnection());
            command.Parameters.Add("@uI", MySqlDbType.Int32).Value =
row["user"].ToString();
            mail = command.ExecuteScalar().ToString();
            members.Add(mail);
        }
        members.Add("");
        db.closeConnection();
        return members;
    }
    private void closeButton_Click(object sender, EventArgs e)
    {
        this.Close();
    }
    private void button1_Click(object sender, EventArgs e)
    {
        string mail = responsibleBox.Text;
        DB db = new DB();
        db.openConnection();
        if (mail == "")
        {
            MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET
`responsible` = NULL WHERE `id` = @tI", db.getConnection());
            command.Parameters.Add("@tI", MySqlDbType.Int32).Value = task.id;
            command.ExecuteNonQuery();
            task.responsible = null;
        }
        else
        {
            MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET
`responsible` = @tR WHERE `id` = @tI", db.getConnection());
            command.Parameters.Add("@tR", MySqlDbType.VarChar).Value = mail;
            command.Parameters.Add("@tI", MySqlDbType.Int32).Value = task.id;
            command.ExecuteNonQuery();
            task.responsible = mail;
        }

        db.closeConnection();
        if (taskForm != null) taskForm.UpdateResponsible();
        task.UpdateResponsible();
        if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
        Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView();
        this.Close();
    }
}
}
}

```

Приложение A.23. Программный код формы AddDateForm

```

using MySql.Data.MySqlClient;
using System;
using System.Drawing;
using System.Linq;
using System.Windows.Forms;

namespace TikTask
{
    public partial class AddDateForm : Form
    {
        public Task task;
        public TaskForm taskForm;
    }
}

```

```

public AddDateForm(Task task)
{
    InitializeComponent();
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    this.task = task;
    if (task.startDate != DateTime.MinValue) startTimePicker.Value =
task.startDate;
    if (task.finishDate != DateTime.MinValue) finishTimePicker.Value =
task.finishDate;
}
public AddDateForm(Task task, TaskForm taskForm)
{
    InitializeComponent();
    this.Location = new Point(Cursor.Position.X, Cursor.Position.Y);
    this.task = task;
    this.taskForm = taskForm;
    if (task.startDate != DateTime.MinValue) startTimePicker.Value =
task.startDate;
    if (task.finishDate != DateTime.MinValue) finishTimePicker.Value =
task.finishDate;
}
private void closeButton_Click(object sender, EventArgs e)
{
    this.Close();
}

private void button1_Click(object sender, EventArgs e)
{
    task.startDate = startTimePicker.Value;
    task.finishDate = finishTimePicker.Value;

    DB db = new DB();
    db.openConnection();

    MySqlCommand command = new MySqlCommand("UPDATE `tasks` SET `startDate`
= @tSD WHERE `id` = @tI", db.getConnection());
    command.Parameters.Add("tSD", MySqlDbType.Date).Value = task.startDate;
    command.Parameters.Add("tI", MySqlDbType.Int32).Value = task.id;
    command.ExecuteNonQuery();

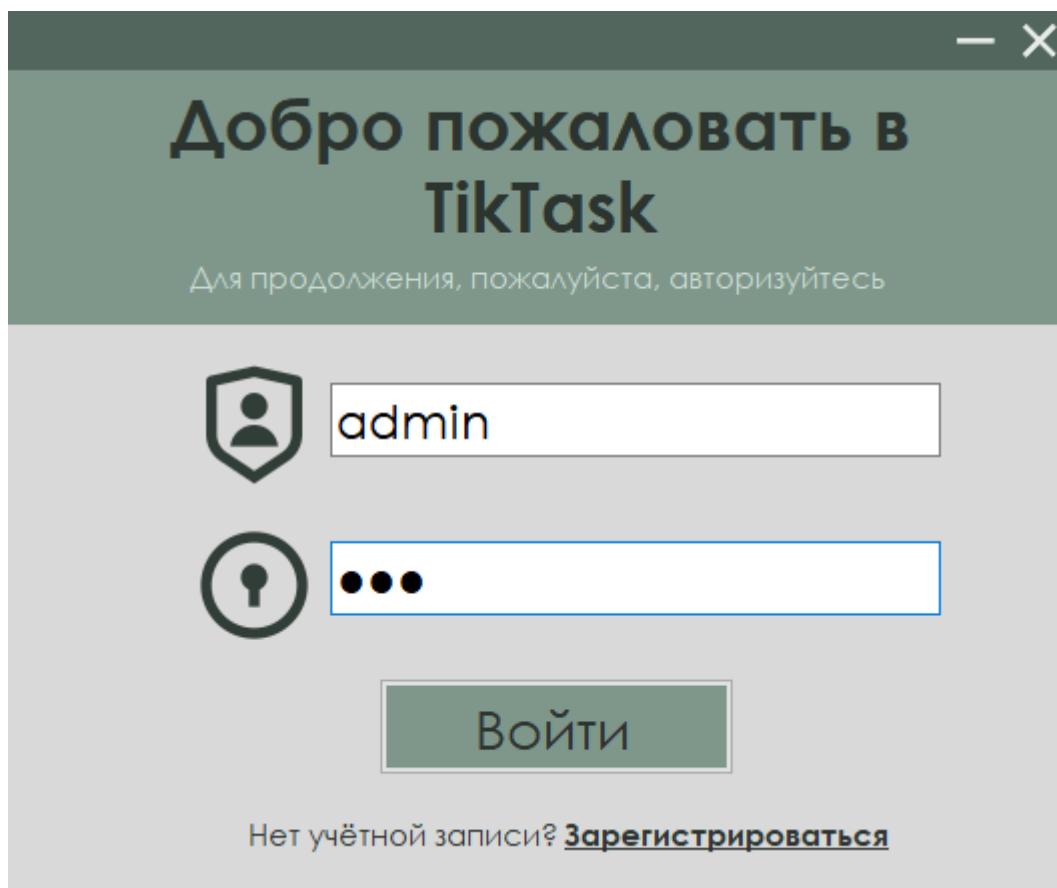
    command = new MySqlCommand("UPDATE `tasks` SET `finishDate` = @tFD WHERE
`id` = @tI", db.getConnection());
    command.Parameters.Add("tFD", MySqlDbType.Date).Value = task.finishDate;
    command.Parameters.Add("tI", MySqlDbType.Int32).Value = task.id;
    command.ExecuteNonQuery();

    db.closeConnection();

    task.UpdateFinishDate();
    if (taskForm != null) taskForm.UpdateDates();
    if (Application.OpenForms.OfType<MainForm>().FirstOrDefault() != null)
Application.OpenForms.OfType<MainForm>().FirstOrDefault().UpdateTasks();
    if (Application.OpenForms.OfType<ProjectForm>().FirstOrDefault() !=
null)
Application.OpenForms.OfType<ProjectForm>().FirstOrDefault().UpdateProjectView(); ;
    this.Close();
}
}
}

```


Приложение Б. Интерфейс приложения
Приложение Б.1. Окно авторизации




The image shows a login window for an application named 'TikTask'. The window has a dark green header bar with a title bar containing a minus sign and a close button (X). Below the header, the text 'Добро пожаловать в TikTask' is displayed in a large, bold, dark font. Underneath this, a smaller line of text reads 'Для продолжения, пожалуйста, авторизуйтесь'. The main area of the window is light gray and contains two input fields. The first field is preceded by a shield icon with a person silhouette and contains the text 'admin'. The second field is preceded by a circular key icon and contains three black dots, indicating a password. Below these fields is a green button with the text 'Войти'. At the bottom of the window, there is a link that says 'Нет учётной записи? Зарегистрироваться'.

Добро пожаловать в
TikTask

Для продолжения, пожалуйста, авторизуйтесь

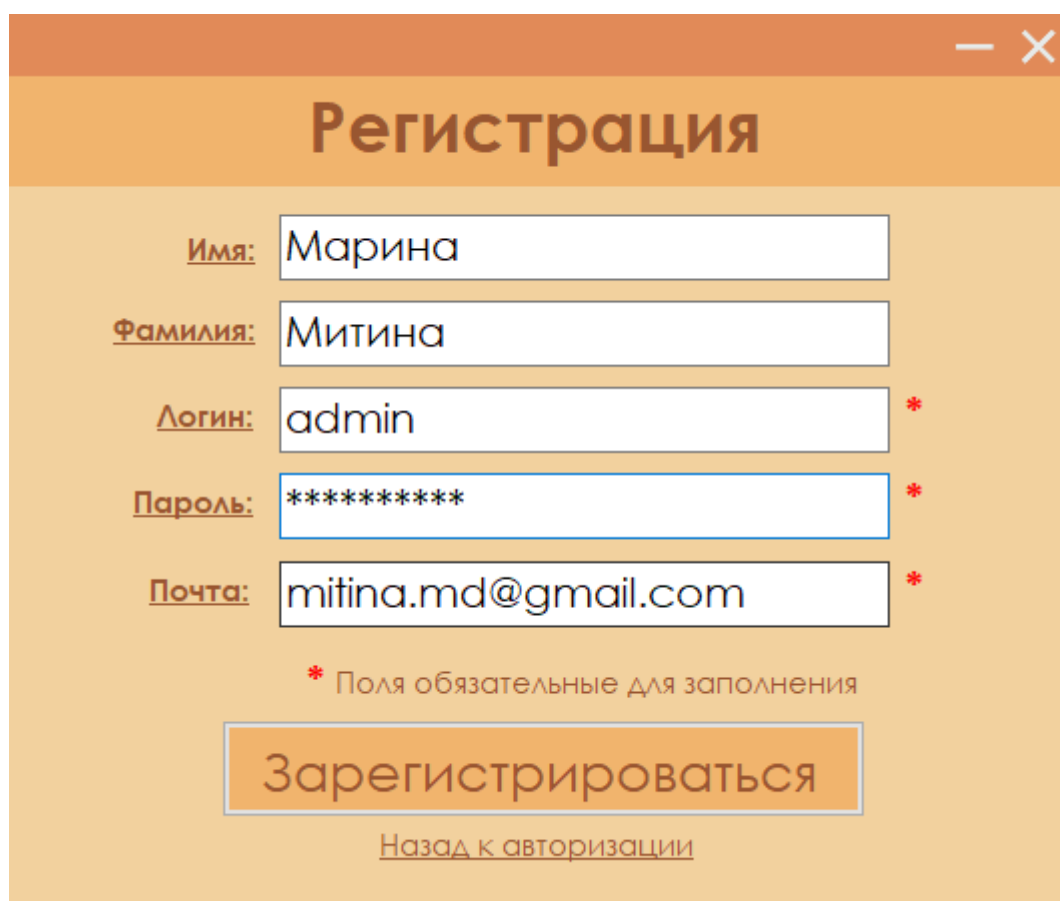
 admin

 ●●●

Войти

Нет учётной записи? [Зарегистрироваться](#)

Приложение Б.2. Окно регистрации



Registration window with an orange header and background. The title "Регистрация" is centered in the header. Below it are five input fields with labels: "Имя:" (Marina), "Фамилия:" (Митина), "Логин:" (admin), "Пароль:" (masked with asterisks), and "Почта:" (mitina.md@gmail.com). Each field has a red asterisk to its right. Below the fields is a red asterisk followed by the text "Поля обязательные для заполнения". At the bottom is a large orange button labeled "Зарегистрироваться" and a link "Назад к авторизации".

Регистрация

Имя: Марина

Фамилия: Митина

Логин: admin *

Пароль: ***** *

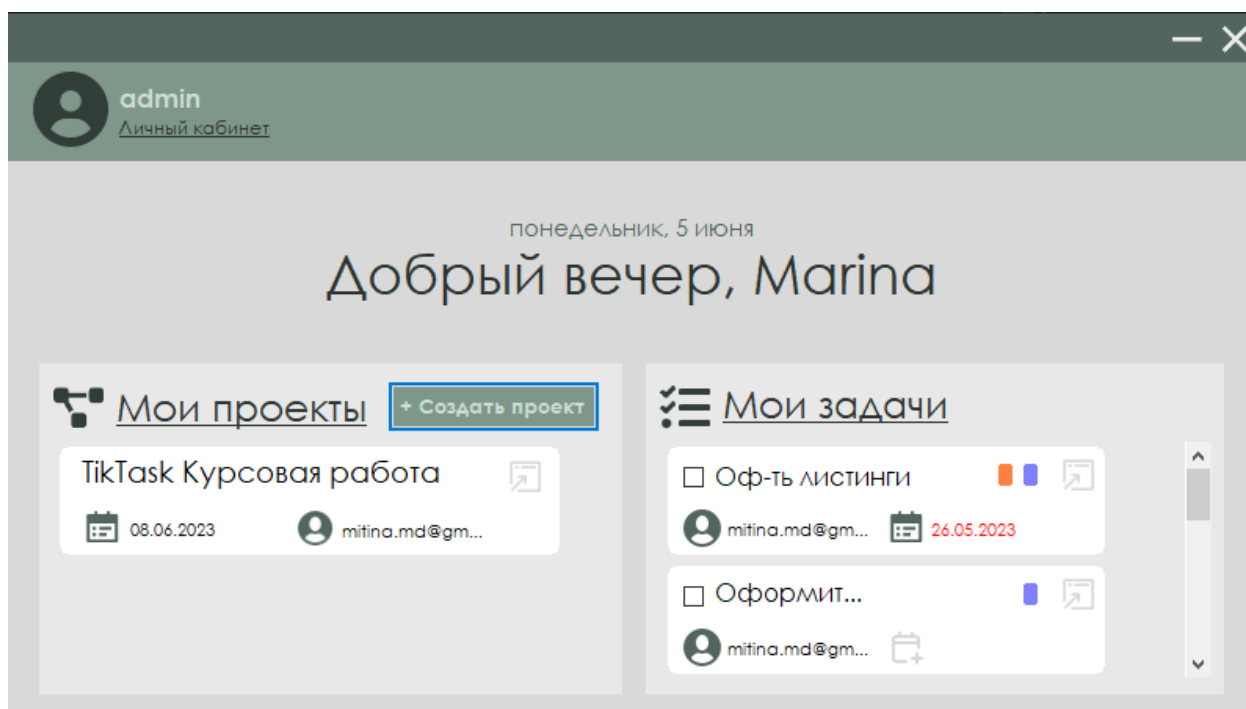
Почта: mitina.md@gmail.com *

* Поля обязательные для заполнения

Зарегистрироваться

[Назад к авторизации](#)

Приложение Б.3. Окно главной страницы



Main page screenshot. The top header shows a user profile for "admin" with a link to "Личный кабинет". Below the header, the date "понедельник, 5 июня" is displayed, followed by a large greeting "Добрый вечер, Marina". The main content area is divided into two columns. The left column, titled "Мои проекты", contains a project card for "TikTask Курсовая работа" with a date of "08.06.2023" and a user icon. The right column, titled "Мои задачи", contains two task cards: "Оф-ть листинги" with a date of "26.05.2023" and "Оформит..." with a user icon. A vertical scrollbar is visible on the right side of the task list.

admin
Личный кабинет

понедельник, 5 июня

Добрый вечер, Marina

Мои проекты + Создать проект

TikTask Курсовая работа
08.06.2023 mitina.md@gm...

Мои задачи

Оф-ть листинги
mitina.md@gm... 26.05.2023

Оформит...
mitina.md@gm...

Приложение Б.4. Окно личного кабинета

Личный кабинет

Уведомления

ivanov@mail.ru приглашает вас присоед... [Принять](#)

Имя Marina

Фамилия Mitina

Логин admin

Почта mitina.md@gmail.com

[Изменить пароль](#)

Выйти

Приложение Б.5. Окно изменения пароля

Старый пароль

Новый пароль

Повторите новый пароль

[Изменить](#)

Приложение Б.6. Окно нового проекта

Новый проект

Название:

Сроки: 2 июня 2023 г. 7 июня 2023 г.

Создать

Июнь 2023

Пн	Вт	Ср	Чт	Пт	Сб	Вс
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	1	2
3	4	5	6	7	8	9

Сегодня: 05.06.2023

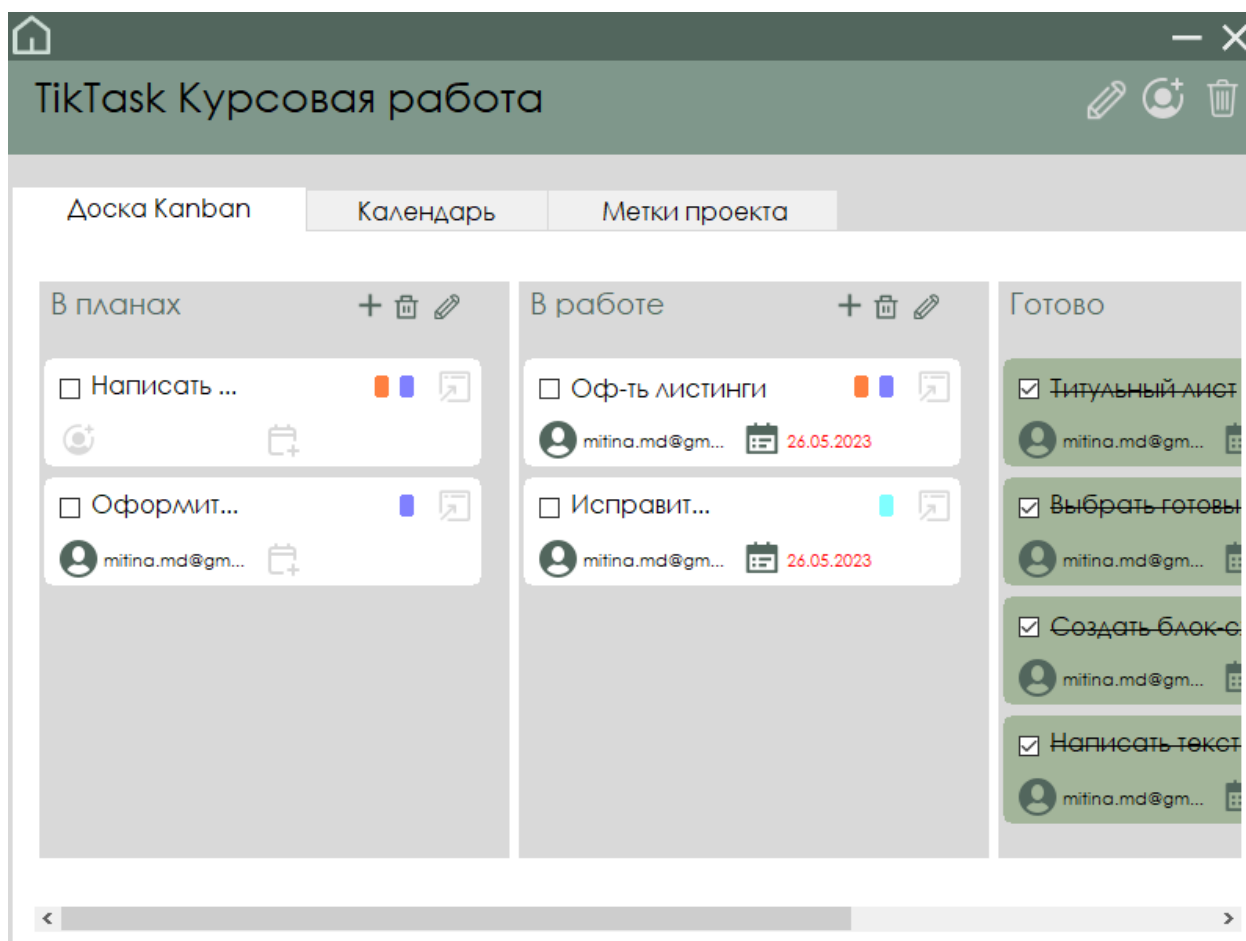
Приложение Б.7. Окно пустого проекта

Пустой проект

Доска Kanban Календарь Метки проекта

+ Добавить колонку

Приложение Б.8. Окно проекта. Вкладка «Доска Kanban»



Приложение Б.9. Окно проекта. Вкладка «Календарь»

ТikTask Курсовая работа

Доска Kanban

Календарь

Метки проекта

Задачи на 26.05.2023

Оф-ть листинги

mitina.md@gm...

26.05.2023

Исправит...

mitina.md@gm...

26.05.2023

Задачи с неустановленным сроком

Оформит...

mitina.md@gm...

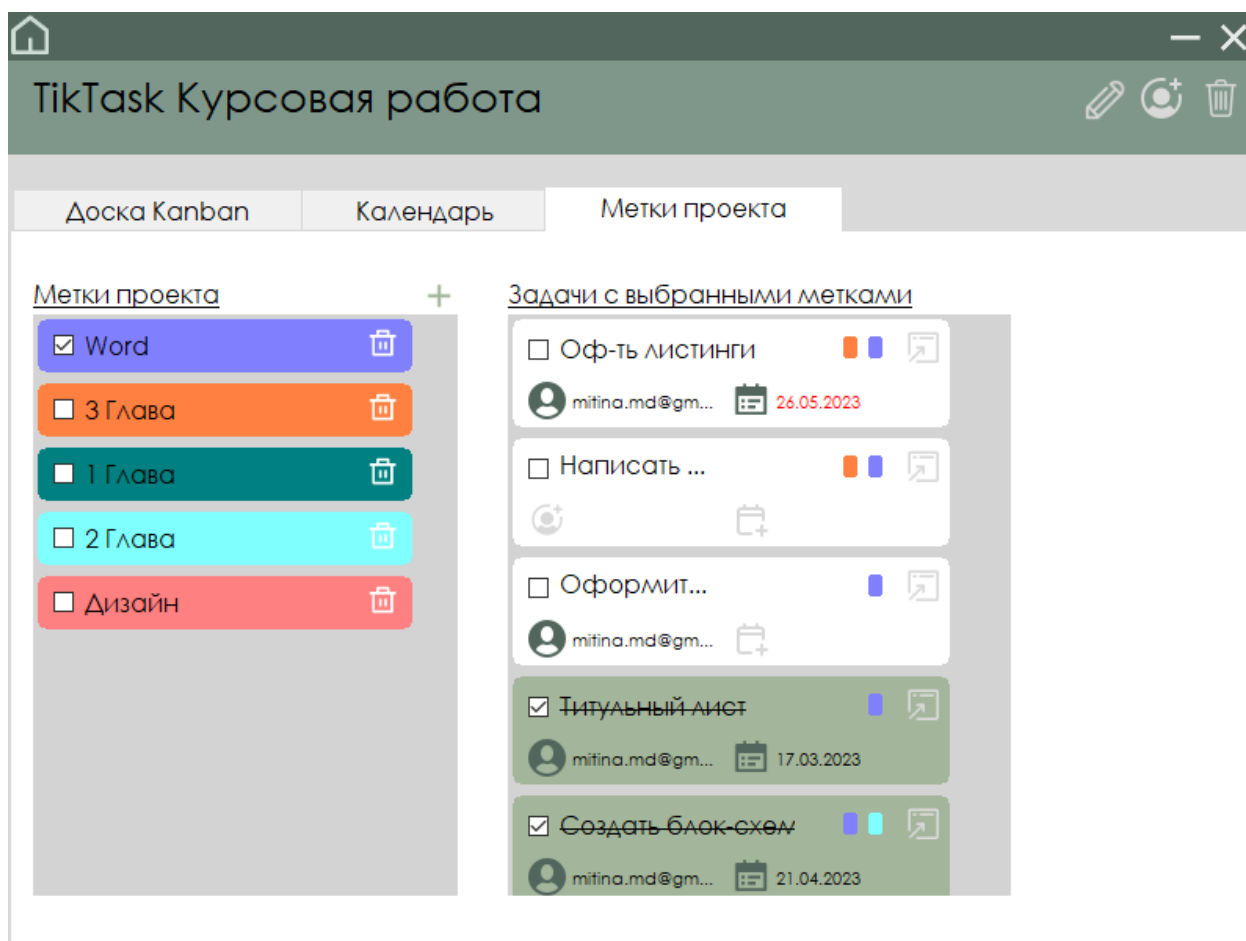
Май 2023

Пн	Вт	Ср	Чт	Пт	Сб	Вс
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

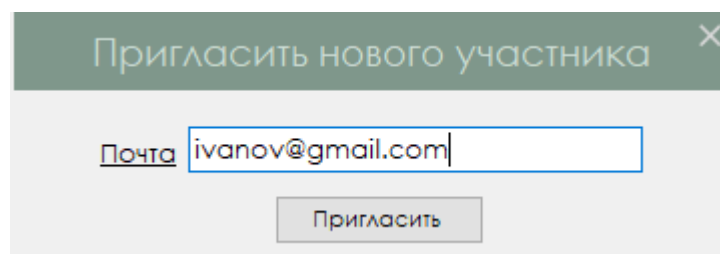
Сегодня: 05.06.2023

141

Приложение Б.10. Окно проекта. Вкладка «Метки проекта»



Приложение Б.11. Окно приглашения нового участника



Приложение Б.12. Окно задачи

The screenshot shows a task management window titled "Оф-ть ЛИСТИНГИ". It contains several fields and sections:

- Исполнитель** (Executor): A text field containing "mitina.md@gmail.com".
- Сроки** (Deadlines): A date range from "19.05.2023" to "26.05.2023".
- Проект** (Project): A text field containing "TikTask Курсовая работа".
- Этап** (Stage): A text field containing "В работе".
- Описание** (Description): A text area containing "Листинг может быть цветной".
- Метки** (Tags): A section with two tags: "Word" and "3 Глава".
- Вложения** (Attachments): A section with one attachment: "Пример оформления третьей главы.pdf".

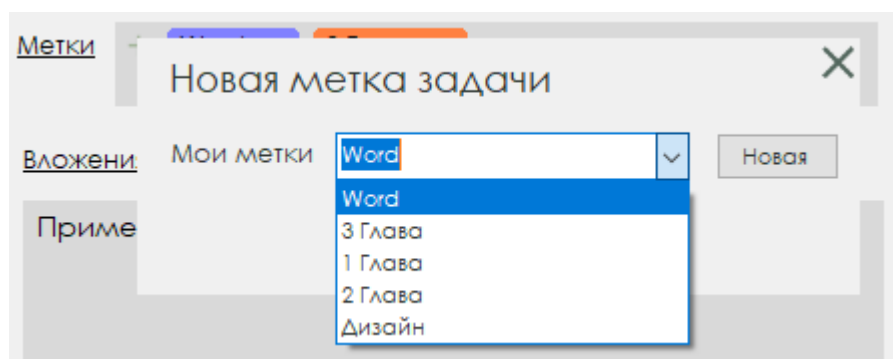
Приложение Б.12. Окно делегирования задачи

The screenshot shows a small window titled "Исполнитель" (Executor) with a close button (X). It contains a text field with "mitina.md@gmail.com" and a "Назначить" (Assign) button.

Приложение Б.13. Окно присвоения даты

The screenshot shows a small window titled "Сроки" (Deadlines) with a close button (X). It contains two date pickers: "19 мая 2023 г." and "26 мая 2023 г.". Below them is a "Применить" (Apply) button.

Приложение Б.14. Окно присвоения метки



Приложение Б.15. Окно создания новой метки

