



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 проектами»

Студент группы Дьяченко В. А., БСБО-05-22

(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 20 23 г.

Допущена

к защите

« 7 » 06 20 23 г.

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.)
« 17 » 02 20 23 г.

ЗАДАНИЕ

на выполнение курсовой работы по дисциплине

«Технологии программирования»

Тема курсовой работы «Разработка приложения для управления проектами»

Студент Дьяченко Владимир Андреевич Группа БСБО-05-22

Исходные данные

Перечень вопросов, подлежащих разработке, и обязательного графического материала:

Титульный лист; Содержание; Введение; Глава 1. Исследование методов управления проектами; Глава 2. Моделирование логики работы 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)

Дьяченко В. А.

(Ф.И.О. обучающегося)

Москва 2023 г.

Оглавление

Введение	6
Глава 1. Исследование методов управления проектами	8
1.1. Анализ существующих решений	8
1.1.1. «YouGile»	9
1.1.2. «Habitica»	11
1.1.3. «Asana»	12
1.1.4. «ToDoist»	13
1.1.5. Сравнительный анализ	13
1.2. Инструментальный аппарат	15
1.3. Постановка задач	15
Выводы	16
Глава 2. Моделирование логики работы программы	17
2.1. Общее представления программы.	17
2.2. Описание модулей программы	18
2.2.1. Модуль ввода логина и пароля	18
2.2.2. Модуль регистрации	19
2.2.3. Модуль меню	20
2.2.4. Модуль добавления проекта	21
2.2.5. Модуль проекта	22
2.2.6. Модуль добавления, изменения задачи	23
2.2.7. Модуль команды	24
2.2.8. Модуль добавление сотрудника	25
2.2.9. Модуль «Мои задачи»	26
Выводы	27
Глава 3. Разработка программы	28
3.1. Начальный экран	28
3.2. Окно регистрации	29
3.3. Класс «Tasks»	30
3.4. Окно меню	31

3.4.1. Диаграмма Ганта	36
3.5. Окно добавления проекта	38
3.6. Окно Добавления сотрудника	40
3.7. Окно изменения, удаления сотрудника	40
3.8. Окно добавления задачи	41
3.9. Окно просмотра задачи.....	43
Выводы	43
Заключение.....	44
Список использованной литературы.....	46
Приложения	48
Приложение А. Листинг кода	48
Приложение А.1. Программный код окна входа в программу.....	48
Приложение А.2. Программный код окна регистрации.....	49
Приложение А.3. Программный код окна меню.....	50
Приложение А.4. Программный код класса «Tasks»	65
Приложение А.5. Программный код окна добавления проекта.....	66
Приложение А.6. Программный код окна добавления задачи	67
Приложение А.7. Программный код окна изменения задачи.....	71
Приложение А.8. Программный код окна добавления сотрудника.....	72
Приложение А.9. Программный код окна изменения сотрудника	73
Приложение Б. Интерфейс приложения	75
Приложение Б.1. Окно входа в программу.....	75
Приложение Б.2. Окно регистрации.....	75
Приложение Б.3. Окно меню при первом запуске.....	76
Приложение Б.4. Окно меню пример с задачами	76
Приложение Б.5. Диаграмма Ганта	77
Приложение Б.6. Окно добавления задачи	77
Приложение Б.7. Окно изменения задачи	78
Приложение Б.8. Окно добавления сотрудника.....	78
Приложение Б.9. Окно изменения сотрудника	79

Приложение Б.10. Окно добавления проекта.....	79
---	----

Введение

В современном мире для каждой компании важно оставаться конкурентоспособной на мировом рынке для увеличения своей прибыли и дальнейшего развития. В этом помогает грамотное управление своими проектами. Вот несколько причин, почему данная процедура крайне необходима в любой организации: целесообразное распределение и экономия ресурсов, таких как время, бюджет, трудовые ресурсы и т.д., оптимизация процесса работы, грамотная постановка и разделение задач, отслеживание прогресса и качества работы, устранение рисков на этапе их образования.

Так как управление проектами непростой процесс, на мировом рынке появилось немало компаний, разработавших свое программное обеспечение, основанное на различных по степени эффективности, сложности и структуре алгоритмах, для помощи организациям в оптимизации своей работы. Также для удобного контроля и оценивания дальнейших действий важной частью этих программ является визуализация поставленных задач в виде различных схем и графиков, обеспечение коммуникации между участниками команды и возможность интеграции с другими приложениями, такими как электронная почта, календарь, CRM-система и др.

Целью этой курсовой работой станет разработка программы для управления проектами.

Задачи курсовой работы:

- 1) поиск, исследование, выделение преимуществ и недостатков существующих решений для управления проектами;
- 2) моделирование логики и структуры программы;
- 3) разработка программного обеспечения с возможностью составления плана проекта разделения крупных задач на конкретные подзадачи

разного уровня сложности реализации для дальнейшего делегирования между специалистами и установлением сроков сдачи работ.

Глава 1. Исследование методов управления проектами

В данной главе будут рассмотрены 4 различных программ. Каждое из них имеет как бесплатную, так и платную версию, однако анализу будет подвергнуты лишь бесплатные версии в связи с их общедоступностью.

В данной работе каждая программа будет оценена по ряду следующих критериев:

- 1) количество возможных пользователей;
- 2) виды представления задач;
- 3) количество проектов, задач;
- 4) возможность коммуникации между сотрудниками внутри приложения;
- 5) возможность интеграций с другими приложениями;
- 6) возможность создания отчетов.

Для продвижения платной версии разработчики ограничивают в бесплатной версии важные аспекты программы. Будет оценено количество доступных пользователей, проектов и задач в бесплатной версии.

В работе будет представлено количество возможных видов представления задач в каждой программе.

Каждое приложение будет рассмотрено на наличие возможности интеграций других приложений. Также будет указано наличие возможности коммуникации между сотрудниками.

Для более удобной работы важен такой параметр, как возможность создавать отчеты, диаграммы, графики. В работе будут рассмотрены наличие данного параметра и способы его реализации в программах.

В конце разбора всех программ по критериям будет представлена сравнительная таблица для быстрого анализа каждой программы.

1.1. Анализ существующих решений

На сегодняшний день множество компаний предлагают свое программное обеспечение для помощи в управлении проектами. Базовый функционал у большинства решений одинаков. Каждая программа имеет

возможность создания проектов, досок, задач и подзадач; организации сотрудников в системе: возможность указывать им должности, в зависимости от которых выдаются определенные права доступа к каждому проекту – сортировать свои задачи в личном планировщике. Также некоторые программы могут отображать текущие состояние проекта в виде различных отчетов и диаграмм. Возможна интеграция с другими приложениями разной направленности и передача или хранение файлов во внутреннем хранилище. Наличие чата как к каждой задаче, так и с определенным человеком, группой, совершение видео и аудио звонков.

1.1.1. «YouGile»

Первым приложением для анализа будет «YouGile». В программе каждый проект разбивается на несколько уровней: доски, колонки, задачи. Пользователь может в неограниченном количестве менять и добавлять каждый уровень проекта. По умолчанию в доске находятся три колонки: «Новые», «В работе», «Готовые». Каждая доска может быть представлена как в виде доски, так и в виде диаграммы Ганта с указанием связей между задачами и их дедлайном. К каждой задаче можно добавить метку, встроенную к таким, относятся: назначение конкретного исполнителя и установление дедлайна, приоритета задачи – или создаваемую. Приложение может само создавать задачу с заданной периодичностью. При нажатии на задачу открывается панель со всей информацией о ней. Здесь можно добавить к задаче описание и также разбить её на подзадачи, при этом автоматически появляется шкала прогресса выполнения задачи.

В программе есть возможность формирования отделов с добавлением в них сотрудников и назначением руководителей. Каждому подчиненному можно назначить индивидуальную настраиваемую роль в зависимости от прав доступа к определенному проекту.

Перейдем к блоку коммуникаций. В каждой задаче есть отдельная вкладка чата для сотрудников, имеющих доступ к этой задаче. Также можно создавать групповые и личные чаты. Уведомления о пропущенных

сообщениях от каждого чата могут приходить на почту или в виде мобильных уведомлений.

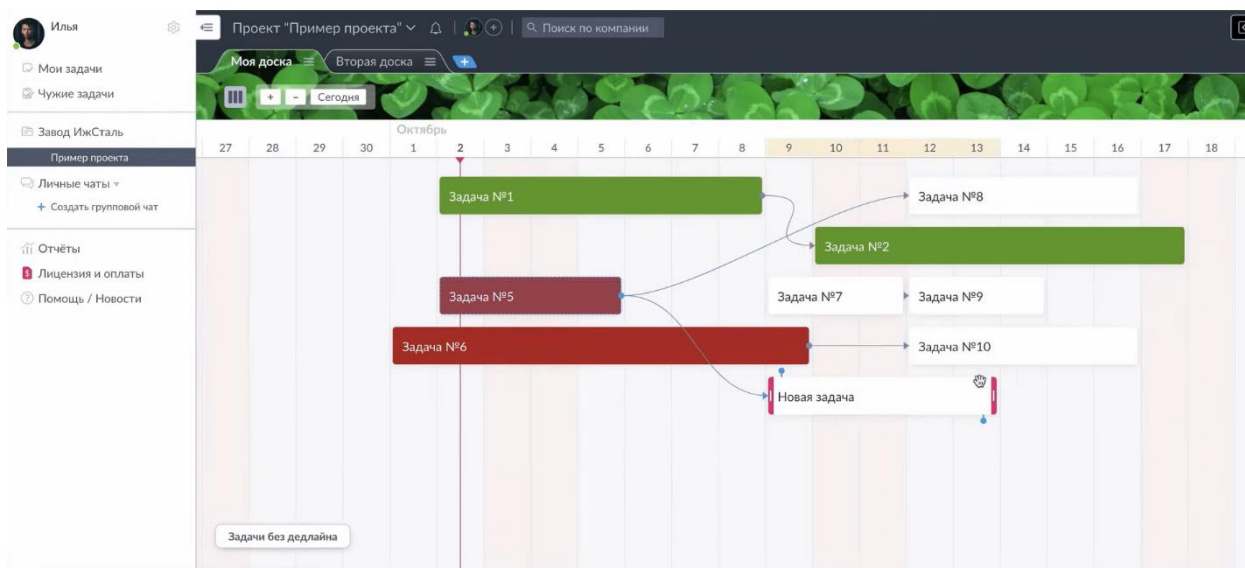


Рисунок 1.1. Пример диаграммы Ганта в приложении «YouGile»

На панели «Отчёты» представлена статистика компании в нескольких видах. Первый – «Общий» разделен на вкладки: по проектам, по отделам, по людям – в каждой видно количество созданных и выполненных задач, также можно вывести собственный запрос. При нажатии на число появляется список определенных задач с информацией о них. Второй – «Табличный отчет» полностью настраивается вами: количество колонок, рядов, их наполнение. Полученный отчет можно скачать в виде «Excel» таблицы или представить в виде диаграммы Ганта (рис. 1.1.). Третий – «Лента событий» - здесь собраны все действия каждого сотрудника в приложении, в заголовке у кадра представлена статистика его активности. Есть график событий-визуальное представления прогресса на заданном участке времени.

В левой части приложения находится меню, в котором представлены все вышеописанные разделы для быстрого доступа к ним.

В бесплатной версии существует ограничение по количеству пользователей, доступно до 10. Количество задач и объём загружаемых файлов не ограничен.

1.1.2. «Habitica»

«Habitica» – необычное приложение, позволяющее формировать список актуальных задач, трекер привычек. Планер разработан на основе компьютерной ролевой игры, мотивирующей выполнять с каждым днем все больше рутинных задач ради достижения внутриигровых целей.

При регистрации программа просит создать собственного персонажа (аватара), прокачка которого основная цель пользователя. Аватар имеет несколько характеристик: сила, интеллект, телосложение, восприятие, здоровье, опыт, мана. При достижении 10 уровня программа предлагает выбрать класс своего персонажа, в зависимости от мотивирующих для пользователя факторов. За выполнение каждой задачи игрок получает внутриигровую валюту «золотые монеты» и опыт для прокачки своего аватара.

Задачи подразделяются на три типа. Первый «Привычки» подразделяются на вредные и полезные, если пользователь выполнил полезную привычку он получает награду, если выполнил вредную персонаж получает урон по здоровью, данная активность имеет счетчик прогресса. Второй «Ежедневные дела» – обновляются в зависимости от периода, который устанавливает пользователь, при пропуске данной активности внутриигровой персонаж также получает урон по здоровью. Третий «Задачи» - здесь представлены задачи, которые нужно выполнять один раз. Каждая задача подразделяется по уровню сложности: пустяк, легко, нормально, сложно. Более высокая сложность приводит к большей внутриигровой награде при завершении задачи, но также к большому внутриигровому ущербу. Внутриигровая валюта нужна для внесения дизайнерских изменений своего аватара и повышения его характеристик. Также существуют испытания, которые пользователь может сам себе придумать или взять из предложенных другими.

В приложении есть командные соревнования, испытания между командами, гильдиями. Группы могут устраивать охоту на монстров, где

каждое выполненное задание наносит последнему урон, а пропущенное – группе.

Программа полностью бесплатна для одного пользователя, для создания группы с распределением внутренних обязанностей придется платить. Ограничений на количество задач нет. Данное приложение больше подходит для индивидуального пользования из-за своего комичного стиля.

1.1.3. «Asana»

Третьим приложением для разбора будет «Asana». В верхней левой панели находится поиск для нахождения любой задачи, подзадачи, сотрудника, из всех проектов. На главной странице для удобства отображены избранные пользователем проекты и задачи или последние с истекающим сроком сдачи, также здесь находятся задачи, в которые пользователь недавно заходил. Есть отдельная вкладка с задачами, которые находятся в работе конкретного пользователя во всех проектах с возможностью сортировки по приоритету.

В «Asana» есть возможность отображения проектов в виде списка, доски Канбана, календаря и диаграммы Ганта с возможностью указания зависимостей. Одной из особенностей этой программы являются гиперссылки на задачи. В каждой задаче в описании или в подзадачах можно ссылаться на другую задачу в любом проекте, выстраивая последовательное исполнение. Можно создавать и настраивать иерархию компании: отделы, роли сотрудников. В разделе «Цель» указана глобальная или годовая цель компании с динамичной шкалой прогресса.

Бесплатная версия данной программы сильно урезана, возможно подключить до 15 пользователей, создать бесконечное количество задач. Вырезана возможность создавать отчеты, графики, диаграммы по проектам, отделам. Также вырезана работа с готовыми формами шаблонами проектов, что не удобно для работы в крупной компании.

1.1.4. «ToDoist»

Последняя программа на рассмотрение «ToDoist». Функционал данного приложения минималистичен по сравнению с предыдущими. На левой панели представлено меню с перечнем функций. Каждый проект можно отобразить в зависимости от удобства для пользователя в виде доски или списка. Также есть отдельная панель с заданиями на текущий день и на предстоящий. Каждая задача рассортирована как по дедлайну, так и по личному приоритету, существует всего 4 уровня встроенных уровня приоритета. Есть возможность добавить описание и комментарий к задаче, разделить её на подзадачи или назначить определенному исполнителю.

Коммуникация в «ToDoist» происходит только через комментарии к задачам.

В программе представлен один вид отчетов. Показано сколько задач пользователь выполнил за определенный период времени.

Бесплатная версия довольно урезана по сравнению с основной. Можно создавать всего до 5 проектов и прикреплять до 5 сотрудников. Приложение сохраняет только 1 неделю истории активности пользователя. Из-за своих ограничений в бесплатной версии «ToDoist» больше подходит для индивидуального планирования.

1.1.5. Сравнительный анализ

В данном пункте, опираясь на описание, будет проведен сравнительный анализ и представлена таблица (см. таблица 1.1). Для всех критериев будут определены шкалы оценок. Их всего три:

- 1) абсолютная;
- 2) неограниченная;
- 3) бинарная.

Абсолютная шкала оценок будет относиться к количеству возможных пользователей, видам представления задач. Абсолютная шкала оценок показывает какое-то целое число. Для всех критериев это будет количество определенных параметров в каждой программе.

Неограниченная шкала оценок будет относиться к возможному количеству создания проектов и задач. Диапазон данной шкалы от единицы до бесконечности.

Бинарная шкала оценок принимает только два значения: это да, либо нет. Данная шкала будет относиться к критериям: возможность коммуникации между сотрудниками, возможность интеграций с другими приложениями и возможность создания отчетов.

Таблица 1.1. Сравнительный анализ

Программа Критерии	YouGile	Habitica	Asana	ToDoist
Количество возможных пользователей	10	1	15	5
Виды представления задач	3	1	4	2
Количество задач.	Сколько угодно	Сколько угодно	Сколько угодно	150
Количество проектов	Сколько угодно	Сколько угодно	Сколько угодно	80
Возможность коммуникации между сотрудниками	Да	Да	Да	Да
Возможность интеграций с другими приложениями	Да	Нет	Да	Нет
Возможность создания отчетов	Да	Нет	Нет	Нет

Все программы, которые были описаны выше, по-своему уникальны. Но в любой программе существуют как свои плюсы, так и минусы.

Почти все приложения кроме «Todoist» имеют большой плюс в виде отсутствия ограничения на количество создаваемых проектов или задач.

Также почти в каждой программе существуют от 2 видов представления задач, что показывает важность в этой функции.

Не особой популярностью пользуется наличие возможности создания отчетов, так как присутствует лишь в одном приложении. Во всех рассматриваемых программах присутствует возможность коммуникации между сотрудниками, следовательно это необходимая функция в приложениях для управления проектами.

Минусом каждой программы является ограничение по количеству пользователей, разрабатываемая программа постарается убрать этот недостаток, ограничиваясь лишь памятью доступной на устройстве.

Проанализировав все это, стоит выделить основные пункты, которые будут учитываться при разработке собственной программы:

- 1) возможность создания отчетов;
- 2) несколько видов представления задач;
- 3) добавление чата в каждую задачу.

1.2. Инструментальный аппарат

Программа будет реализована на языке программирования C#, так как C# – это современный, инновационный, кроссплатформенный объектно-ориентированный язык программирования с открытым кодом [7]. Программа будет реализована только для компьютера. Основным инструментом для разработки выбрана платформа «Windows Forms». Данный выбор обусловлен тем, что данная платформа проста в использовании. Инструмент очень эффективен в реализации простых компьютерных приложений, что и является целью данной работы [8].

1.3. Постановка задач

На главном экране программы будет представлен список задач, назначенных пользователю, отсортированных по приближению к сроку сдачи.

Будет реализовано меню, содержащее следующие позиции:

- 1) список проектов;
- 2) команда;
- 3) мои задачи.

Каждый проект будет иметь вид списка и диаграммы Ганта. Их смена будет производиться в верхней панели проекта.

В каждой задаче будет внутренний чат. К любой задаче пользователь может указать следующие пункты:

- 1) описание;
- 2) срок выполнения;
- 3) фамилия, имя сотрудника, обязанного её выполнить.

Пользователь сможет добавлять новых пользователей в свои проекты и устанавливать им роли, отображающие уровень доступа к различным задачам.

Выводы

В данной главе был произведен анализ 4 программ для управления проектами: «YouGile», «Habitica», «Asana», «Todoist». Каждое приложение имеет краткое описание функционала с выявлением преимуществ и недостатков. Также были проанализированы в соответствии с разобранными критериями: количество возможных пользователей; виды представления задач; количество проектов, задач; возможность коммуникации между сотрудниками внутри приложения; возможность интеграций с другими приложениями; возможность создания отчетов. Для удобства была представлена сравнительная таблица. Был указан инструментальный аппарат. Итогом стала постановка задач для программы, учитывая все плюсы и минусы разобранных программ. В следующей главе, опираясь на поставленные условия, программа будет структурно разобрана и описана на основе связанных между собой модулей, каждый отвечающий за отдельную часть интерфейса или алгоритма программы.

Глава 2. Моделирование логики работы программы

В этой главе будет смоделирована работа программы, опираясь на поставленные в предыдущей главе задачи.

Приложение будет структурно разобрано и описано на основе связанных между собой модулей, каждый из которых отвечает за отдельную часть интерфейса или алгоритма программы.

Для анализа было выявлено 10 модулей:

- 1) модуль ввода логина и пароля;
- 2) модуль регистрации;
- 3) модуль меню;
- 4) модуль добавления проекта;
- 5) модуль проекта;
- 6) модуль добавления, изменения задачи;
- 7) модуль команды;
- 8) модуль добавления сотрудника;
- 9) модуль «Мои задачи».

В каждой подглаве после детального описания модуля будет представлена его работа в виде блок-схемы.

2.1. Общее представление программы.

При запуске приложения пользователь попадает в модуль ввода логина и пароля, далее если пользователь нажмёт на кнопку «Регистрация» он попадает в модуль регистрации, если нажмет на кнопку «Войти» будет осуществлен переход в модуль меню.

Модуль меню представлен в левой части интерфейса программы. В нём представлены кнопки для перехода между несколькими модулями, также здесь представлен список всех проектов, доступных пользователю. В нижней части присутствует кнопка «Добавить проект», открывающая модуль добавления проекта.

В модуле проекта представлены задачи в виде списка, в шапке есть две кнопки «Список» и «Диаграмма», меняющие вид представления задач. Снизу

представлена кнопка «Добавить задачу», открывающая модуль добавления задачи

В модуле команды представлен список сотрудников, снизу находится кнопка «Добавить сотрудника», открывающая модуль добавления сотрудника.

Модуль «Мои задачи» показывает определенные задачи вошедшего в систему пользователя отсортированные по приближению к сроку сдачи.

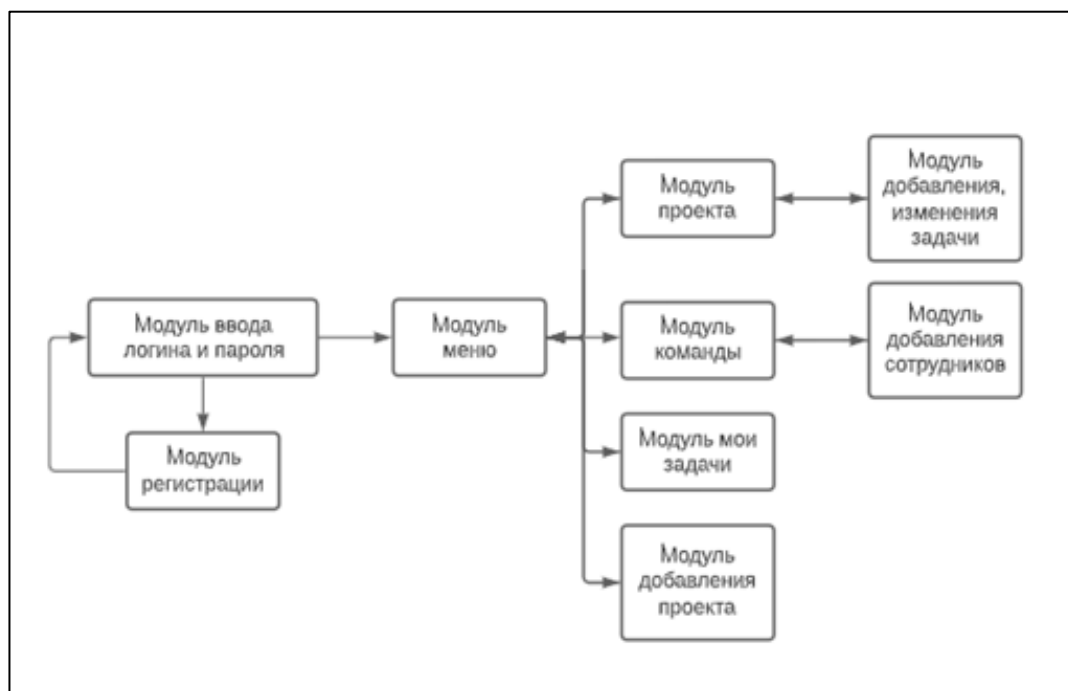


Рисунок 2.1 – функциональная схема программы

2.2. Описание модулей программы

В данной подглаве будет реализовано проектирование всех модулей приложения с описанием их принципов работы и рассмотрены алгоритмы, которые составляют основу этих модулей.

2.2.1. Модуль ввода логина и пароля

При открытии программы пользователь попадает в этот модуль. Здесь представлено два поля для ввода данных, в первом требуется ввести имя пользователя, а во втором пароль от учетной записи. Под этими полями расположена кнопка «Войти», после нажатия которой алгоритм сравнивает введенные данные с существующими в файлах программы и, если найдется совпадения, то открывается модуль меню, а если пользователь вводит некорректные, несуществующие данные в поля, то над ними появляется

надпись «Неверный логин или пароль». Когда у пользователя отсутствует логин и пароль, он должен нажать на кнопку «Регистрация», открывающая модуль регистрации для внесения данных его учетной записи в систему.

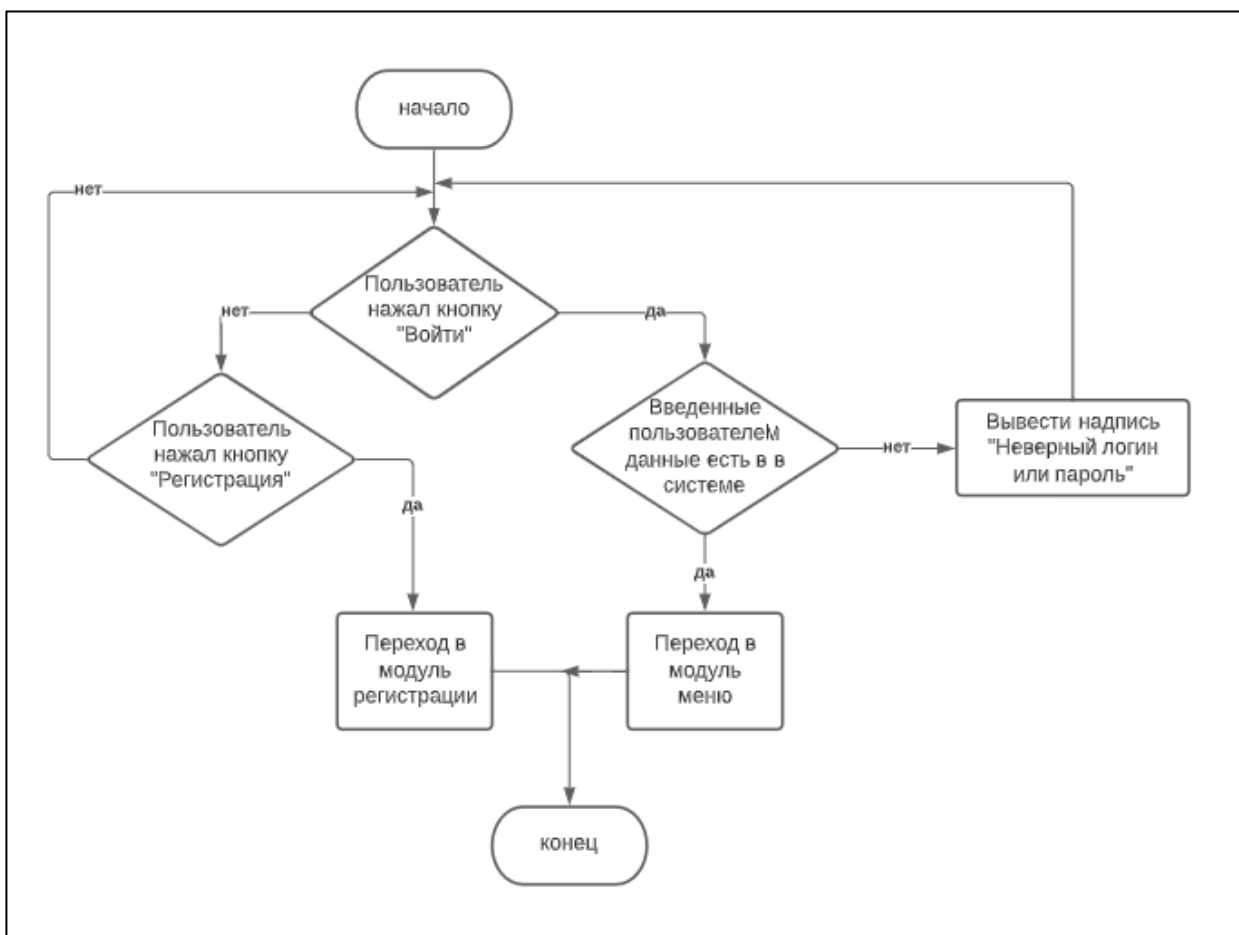


Рисунок 2.2 – блок-схема модуля ввода логина и пароля

2.2.2. Модуль регистрации

В данном модуле представлено два поля для ввода данных: «Имя пользователя» и «Пароль». Под ними находится кнопка «Добавить». В начале данная кнопка сравнивает введенное пользователем имя с существующими в файлах программы и если обнаружит совпадения, то над первым полем появится надпись «Такое имя уже существует», иначе алгоритм добавляет введенный логин и пароль в файлы программы для дальнейшего использования и выводит надпись «Вы успешно зарегистрированы». Также в модуле находится кнопка «Назад», которая возвращает пользователя в блок ввода логина и пароля.

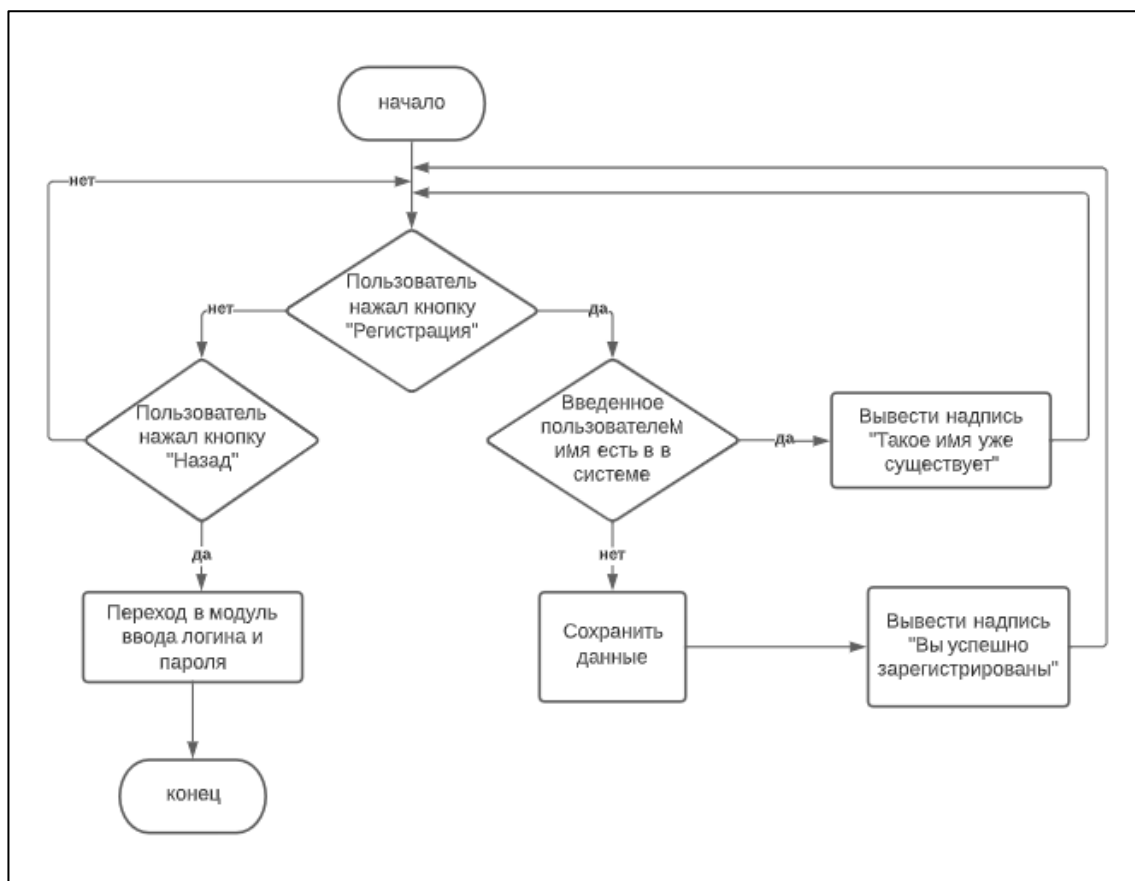


Рисунок 2.3 – блок-схема модуля регистрации

2.2.3. Модуль меню

Данный модуль расположен в левой части интерфейса. В его верхней части отображается имя пользователя, вводимое при входе в приложение. Модуль меню создан для удобного перемещения по программе, в нём представлены функциональные кнопки со следующим функционалом:

- 1) кнопка с именем проекта, созданным пользователем, перемещает пользователя в модуль с конкретным проектом;
- 2) кнопка «Моя команда» перемещает пользователя в модуль команды;
- 3) кнопка «Мои задачи» перемещает пользователя в модуль мои задачи;
- 4) кнопка «Добавить проект» перемещает пользователя в модуль добавления проекта.

После взаимодействия с модулем добавления проекта введенное пользователем имя проекта отобразится в модуле меню в виде кнопки.

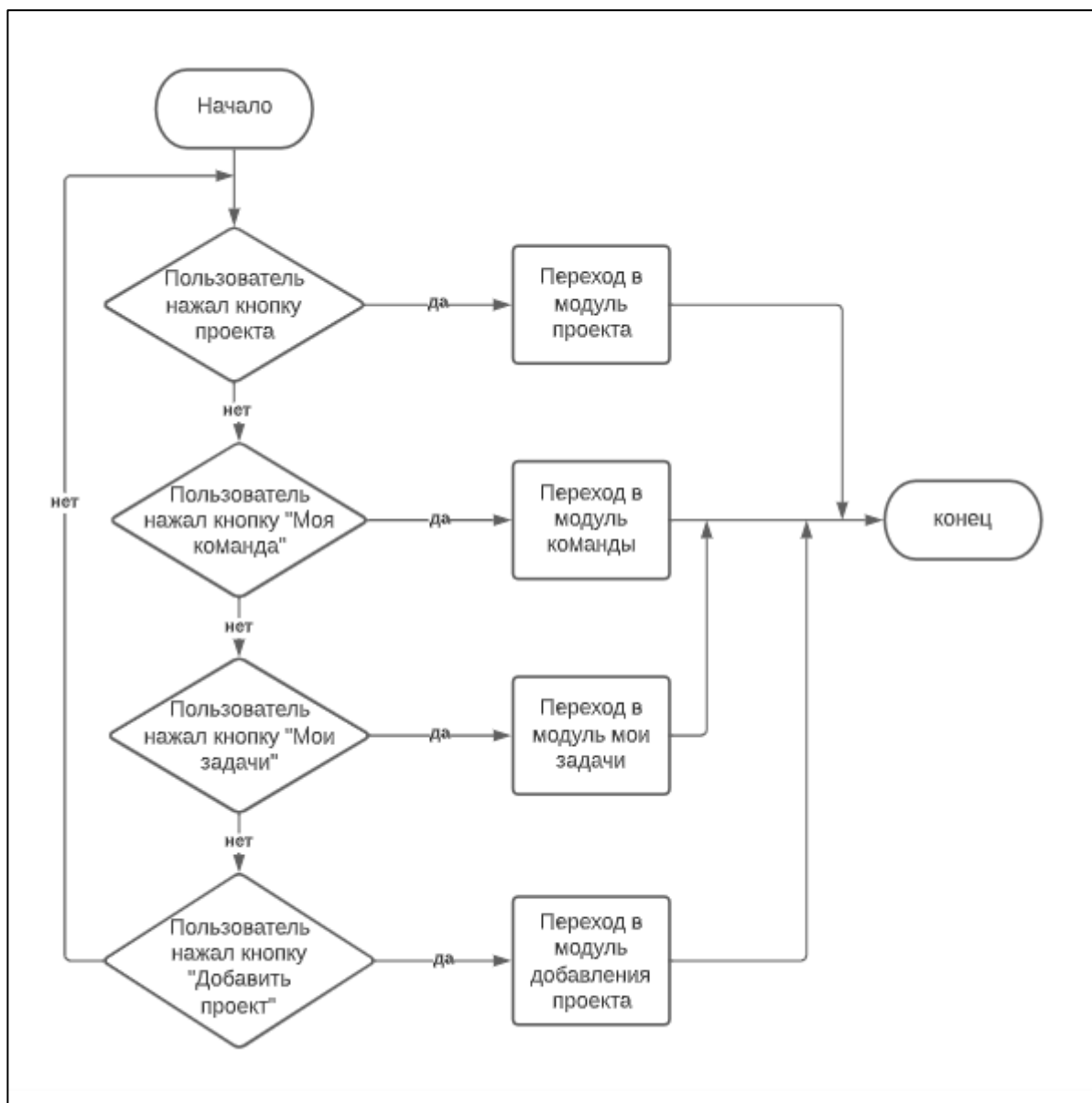


Рисунок 2.4 – блок-схема модуля меню

2.2.4. Модуль добавления проекта

В этом модуле находится поле для ввода данных и две кнопки. Пользователь должен ввести имя для своего нового проекта в данное поле. Также есть возможность выбрать сотрудников, которые будут отображаться в модуле добавления, изменения задач, после этого нажать кнопку «Добавить», и данный модуль закроется, а введенное пользователем имя проекта отобразится в модуле меню в виде кнопки. Если пользователь не ввел значение в поле, то кнопка не работает, а над полем появится надпись «Это поле обязательно для заполнения». Также присутствует кнопка «Назад» закрывающая данный модуль.

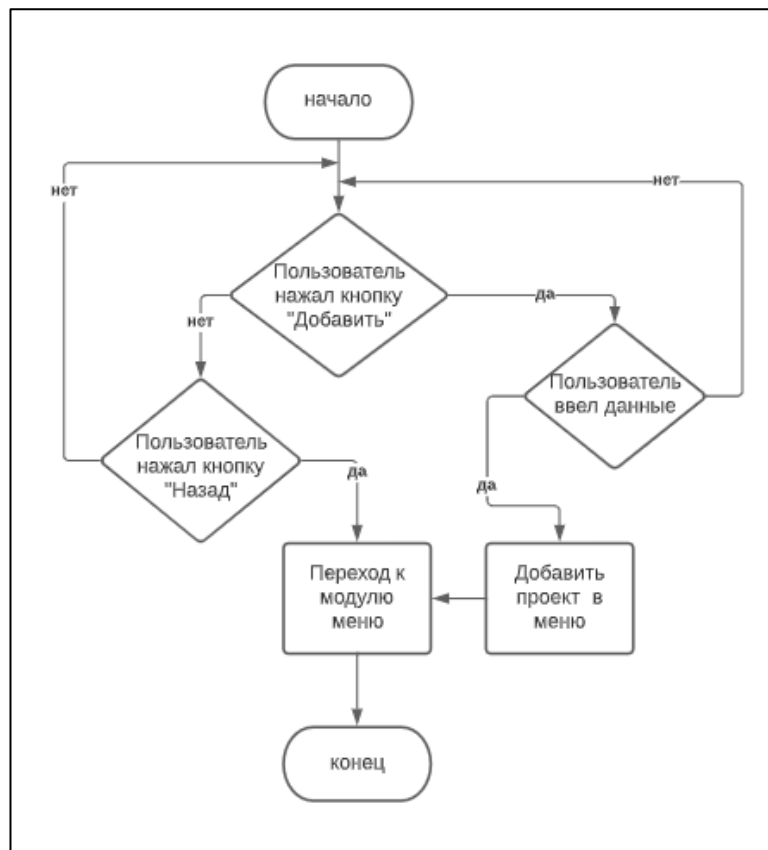


Рисунок 2.5 – блок-схема модуля добавления проекта

2.2.5. Модуль проекта

В данном модуле представлены задачи, относящиеся к конкретному проекту в виде списка. Справа от каждой задачи присутствует кнопка в виде кружка нажатие на которую расценивается программой как выполнение этой задачи и удаляет задачу из списка. Снизу присутствует кнопка «Добавить задачу» открывающая модуль добавления, изменения задачи. При нажатии на любую задачу открывается модуль добавления, изменения задачи, с данными этой задачи.

Также в верхней части модуля представлены две кнопки «Список» и «Диаграмма», меняющие вид представления задач на табличный или в виде диаграммы Ганта соответственно. Вид диаграммы представляет из себя таблицу, в которой сверху представлены календарные дни, а в крайней левой колонке представлены задачи по проекту. Прямоугольники на их пересечение закрашены, показывая дедлайн к конкретной задаче.

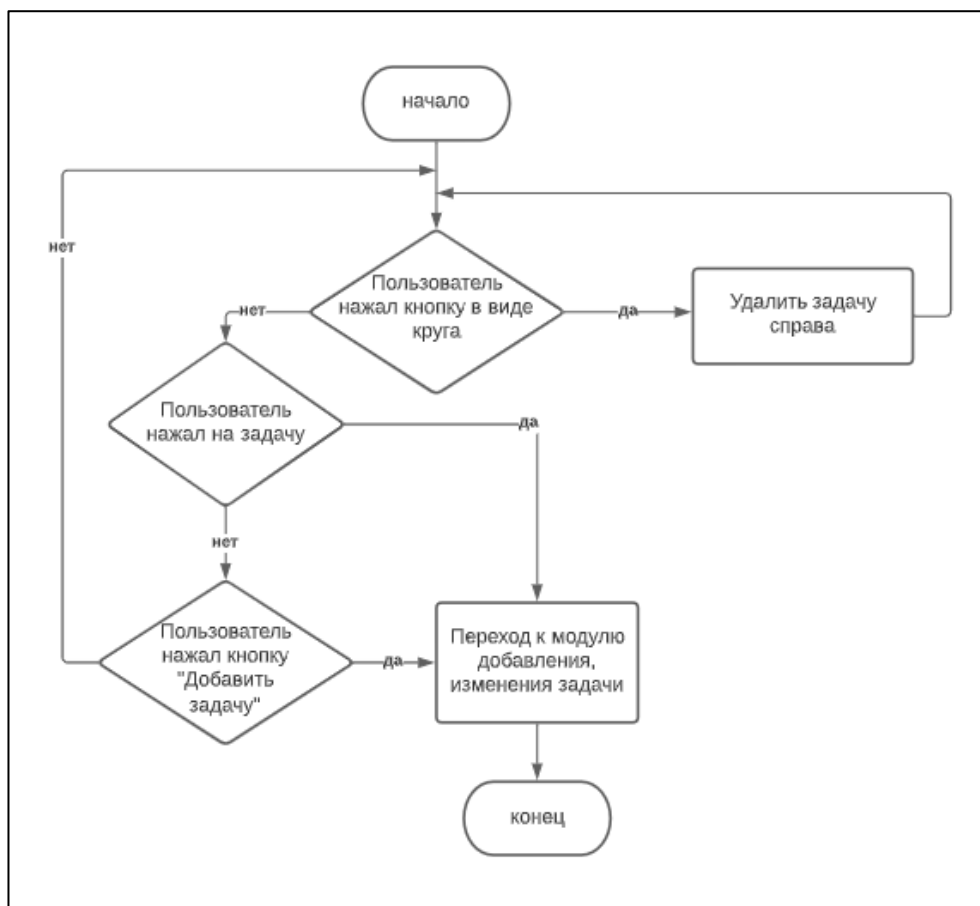


Рисунок 2.5– блок-схема модуля проекта

2.2.6. Модуль добавления, изменения задачи

У каждой задачи есть несколько атрибутов:

- 1) название задачи;
- 2) срок выполнения задачи;
- 3) описание задачи;
- 4) исполнитель;
- 5) сложность задачи.

При создании задачи открывается модуль добавления задач, через который осуществляется добавления этих атрибутов к задаче. Также в модуле представлено две кнопки, первая «Добавить» закрывающая данный модуль и добавляя задачу в список задач в модуле проектов. Если пользователь не ввел название задачи, то кнопка не сработает и появится надпись «это поле обязательно для заполнения» над этим полями. Добавления сложности у задачи будет реализовано в виде списка, где представлено три уровня

сложности: сложно, нормально, легко – в зависимости от выбранной пользователем сложности задача будет окрашена в конкретный цвет: красный, желтый, зеленый соответственно. Если пользователь не выбрал сложность, то добавленная задача будет окрашена в чёрный. Поле исполнителя будет реализовано в виде выпадающего списка, в котором будут отображены сотрудники, показанные в модуле команды. Вторая кнопка «Назад» закрывает данный модуль.

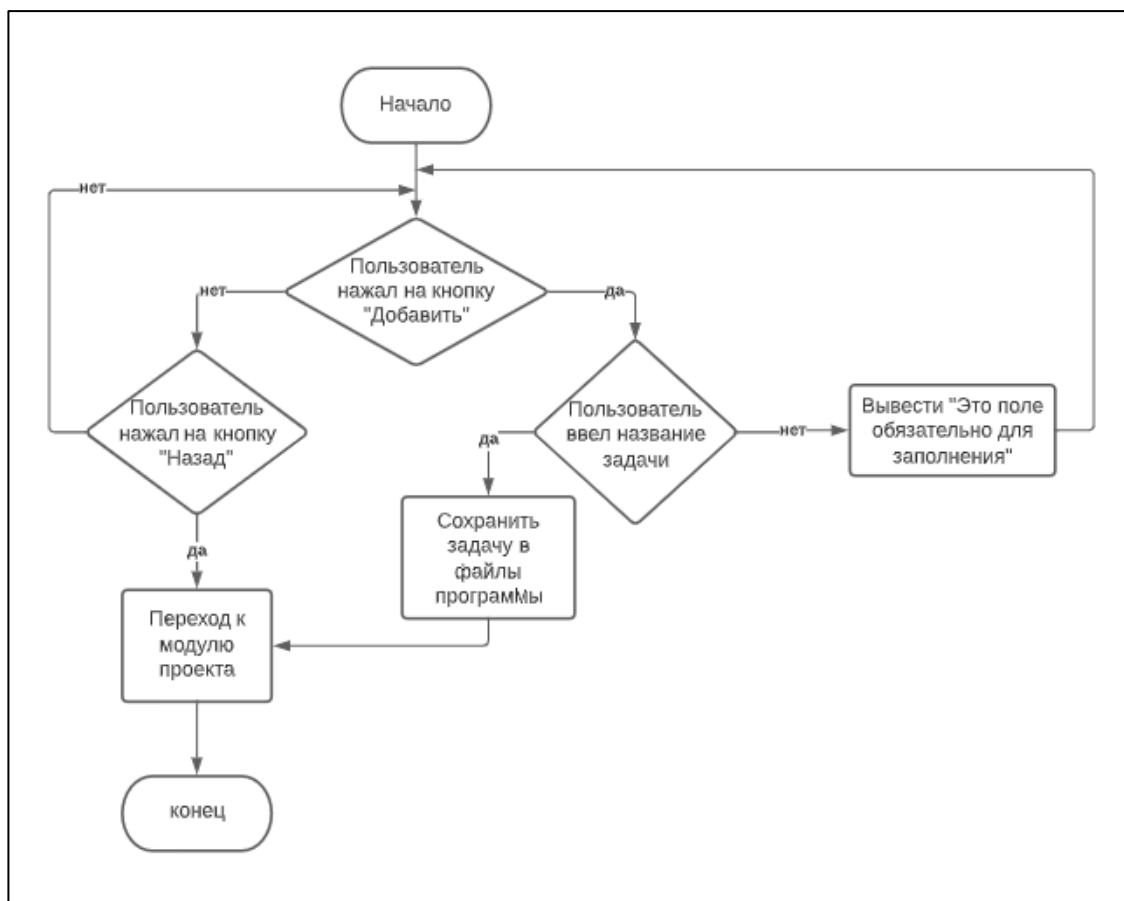


Рисунок 2.6 – блок-схема модуля добавления, изменения задачи

2.2.7. Модуль команды

В данном модуле алгоритм считывает из внутренних файлов программы имена добавленных пользователем сотрудников и отображает прочитанный список имен на экран. Снизу данного модуля представлена кнопка «Добавить сотрудника», открывающая модуль добавления сотрудника.



Рисунок 2.7 – блок-схема модуля команды

2.2.8. Модуль добавление сотрудника

У каждого сотрудника имеется несколько атрибутов, назначаемые через данный модуль:

- 1) имя сотрудника;
- 2) доступ сотрудника к проектам.

Второй пункт реализован в виде списка с существующими в программе проектами, где справа от проекта будет находится кнопка, при нажатии на которую появится галочка, символизирующая системе о присоединение этого сотрудника к этому проекту. В данном модуле находятся две кнопки: «Добавить» и «Назад». При нажатии пользователем на первую программа считывает данные из полей и записывает во внутренние файлы, после этого закрывает модуль. Если пользователь не ввел данные в поле «имя сотрудника» над этим полем появится надпись «Это поле обязательно для заполнения» и кнопка не сработает. При нажатии на кнопку «Назад» модуль закрывается.

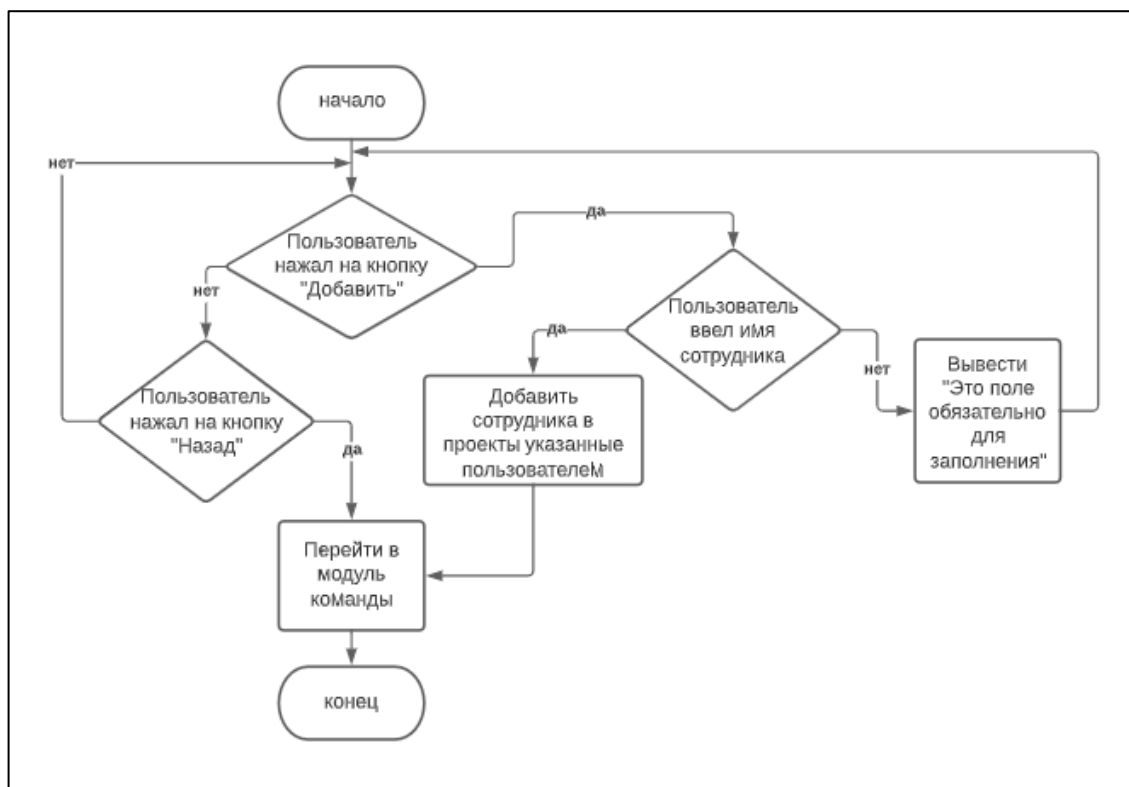


Рисунок 2.8 – блок-схема модуля ввода логина и пароля

2.2.9. Модуль «Мои задачи»



Рисунок 2.9 – блок-схема модуля мои задачи

В данном модуле в виде списка представлены задачи, которые назначены на выполнение пользователю, вошедшему в систему. Справа от

каждой задачи будет отображаться кнопка в виде круга. Нажатие на неё символизирует выполнение задачи, и система убирает её из списка.

Выводы

В данной главе была описана функциональная работа программы. Приложение было поделено на 10 различных модулей: модуль ввода логина и пароля, модуль регистрации, модуль меню, модуль добавления, изменения проекта, модуль проекта, модуль добавления задачи, модуль команды, модуль добавления сотрудника, модуль «Мои задачи», каждый из которых отвечал за свою часть приложения. После было представлено краткое описание работы программы с указанием на связанные между собой модули. Также в главе представлено подробное описание особенностей и алгоритмов каждого модуля с приложением подробной блок-схемы для упрощенного планирования действий при разработке программы. В следующей главе опираясь на разобранные модули будет разработана программа и описана её работа исходя из инструментального аппарата, описанного в первой главе.

Глава 3. Разработка программы

В данной главе будут описаны методы и способы реализации некоторых аспектов программы. Также будут представлены скриншоты приложения и листинги кода, иллюстрирующие работу программы.

3.1. Начальный экран

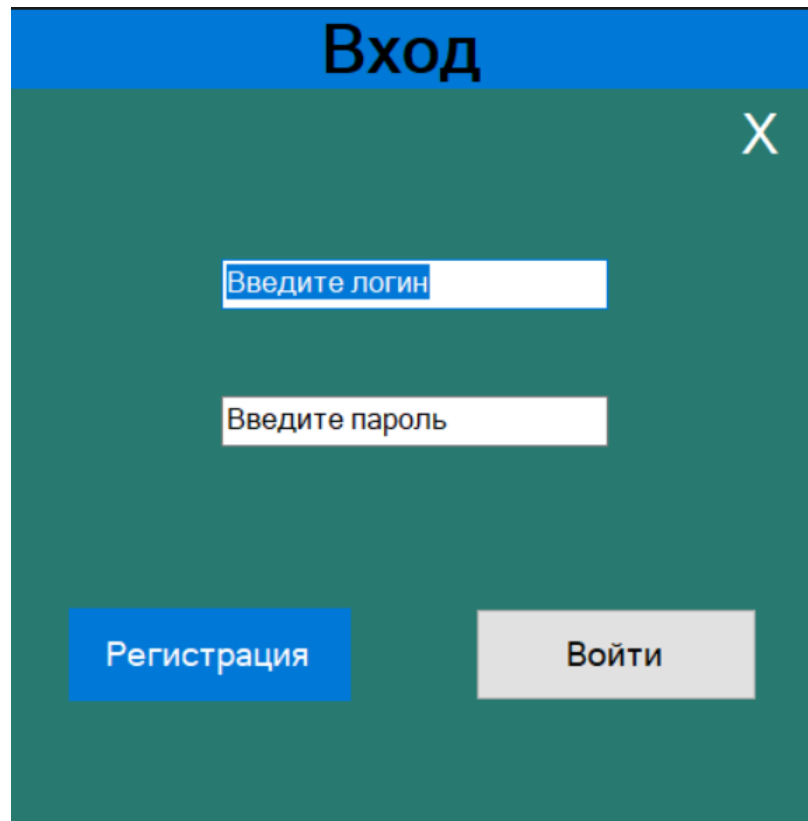


Рисунок 3.1 – окно входа в программу

При запуске программы появляется форма входа в основное меню программы (см. рис. 3.1). На экране данного окна приложения представлены два поля для ввода данных. Первое принимает логин пользователя, второе пароль от аккаунта пользователя. Также на экране есть две кнопки: «Регистрация» и «Войти». Первая открывает форму регистрации (см. листинг 3.1). Вторая считывает данные из полей и проверяет их с помощью функции на соответствие в файлах программы. Если данные введены верно, то открывается форма «Меню» приложения. Если данные введены не верно, то над полями для ввода данных появляется текст «Неверный логин или пароль», который изначально скрыт от пользователя (см. листинг 3.1). В левом верхнем

углу находится кнопка «Заккрыть» в виде крестика, которая полностью закрывает программу (см. листинг 3.1).

```
private void buttonClose_MouseClick(object sender, MouseEventArgs e)
{
    Application.Exit();
}
private void button1_MouseClick(object sender, MouseEventArgs e)
{
    this.Hide();
    registrationForm registrationForm = new registrationForm();
    registrationForm.Show();
}
string check(string login,string passwond)
{
    string filepath = @"users.txt";
    StreamReader sr = new StreamReader(filepath);
    string line;
    string[] mas = new string[2];
    while ((line = sr.ReadLine()) != null)
    {
        mas = line.Split(',');
        if (mas[0] == login)
        {
            sr.Close();
            return mas[1];
        }
    }
    sr.Close();
    return "0";
}
private async void button_input_MouseClick(object sender, MouseEventArgs e)
{
    string login = Login.Text;
    string password = Password.Text;
    string pas = check(login, password);
    if (String.Equals(password,pas))
    {
        this.Hide();
        MenuForm Menu = new MenuForm(login);
        Menu.Show();
    }
    else
    {
        check2.Show();
        await Task.Delay(1000);
        check2.Hide();
    }
}
```

Листинг 3.1 – описание действий при нажатии на кнопки «Регистрация», «Заккрыть», «Войти»

3.2. Окно регистрации

В данном окне представлены два поля для ввода данных: логина и пароля – и две кнопки «Назад», «Регистрация». Первая закрывает окно и возвращает пользователя в форму входа. Вторая считывает данные из полей и сравнивает есть ли уже такой логин в файлах программы. Если есть, то

программа отображает изначально скрытый текст «Такой логин уже существует». Если совпадений в файле не найдено, то новый логин и пароль записывается в файл, между ними ставится разделитель в виде знака «,» (см. листинг 3.2).

```
string login = Login.Text;
string password = Password.Text;
string filepath = @"users.txt"; //путь к файлу csv
StreamReader sr = new StreamReader(filepath);
string line;
string[] mas = new string[2];
bool flag = false;
while ((line = sr.ReadLine()) != null)
{
    mas = line.Split(',');
    if (mas[0] == login) {flag = true;}
}
sr.Close();
if (!flag)
{
    Directory.CreateDirectory($"users\{login}");
    Directory.CreateDirectory($"users\{login}\project");
    File.Create($"users\{login}\worker.txt").Close();
    using (StreamWriter sw = new StreamWriter(filepath, true,
        Encoding.Default))
    {
        sw.WriteLine($"{login},{password}");
    }
    MessageBox.Show("успешно");
}
else
{
    check.Show();
    await Task.Delay(1000);
    check.Hide();
}
```

Листинг 3.2 – метод проверки и записи логина и пароля в файл

3.3. Класс «Tasks»

Данный класс был создан для удобного хранения задач. У класса есть несколько свойств:

- 1) название задачи;
- 2) сотрудник, обязанный её выполнить;
- 3) уровень её сложности;
- 4) описание к задаче;
- 5) дата начала выполнения задачи;
- 6) дата конца выполнения задачи.

Уровень задачи измеряется числовым значением от 0 до 3, где 0 – без уровня сложности, 1-легкая, 2-средняя, 3-сложная. Даты реализованы с

помощью структуры «DateTime». Данный класс имеет атрибут [Serializable] для дальнейшей сериализации данного объекта. В классе представлен метод для вывода объекта в виде строки.

Также существует класс «MasTasks», содержащий в себе коллекцию объектов класса «Tasks». Этот класс также имеет атрибут [Serializable].

3.4. Окно меню

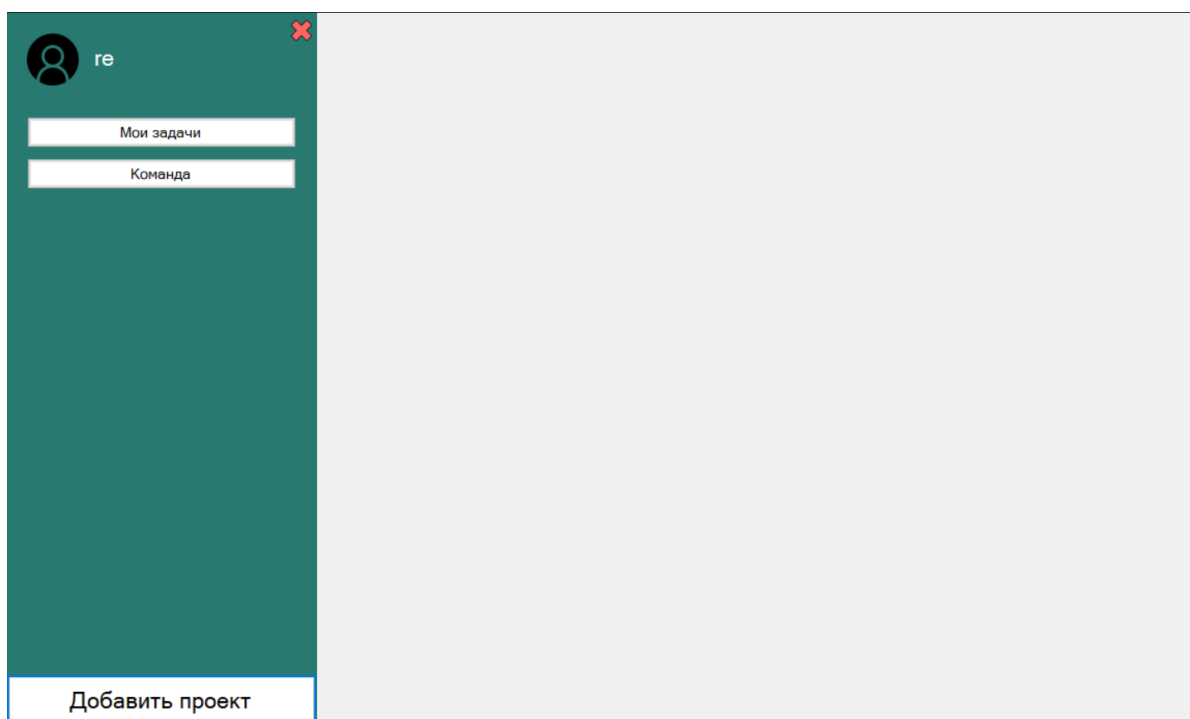


Рисунок 3.2 – окно меню при первом запуске

Данное окно является основным во всей программе. В нем представлены две панели «menu» и «desk». При вызове окна в него передается параметр логин пользователя, под которым был осуществлен вход. Этот логин отображается в верху панели «menu». В первой панели изначально представлены 4 кнопки «Команда», «Мои задачи», «Добавить проект» и «Заккрыть» (см. рис. 3.2). Кнопка «Заккрыть» идентична такой же кнопке в окне входа. Кнопка «Добавить проект» открывает форму для добавления проекта. Кнопка «Мои задачи» отображает все задачи в виде списка из всех проектов, у которых в свойстве сотрудник указано значение «Я». Данный список представлен в виде объекта «radio button» из платформы «Windows Form» (см. листинг 3.3).

```

private void MyTask_MouseClick(object sender, MouseEventArgs e)
{
    desk.Controls.Clear();
    MasTasks tasks= new MasTasks();
    foreach (string file in Directory.EnumerateFiles($"users/{Login}/project", "*",
        SearchOption.AllDirectories))
    {
        foreach(Tasks t in Desserialized(file, 0).ListTasks)
        {
            if (t.worker=="Я")
            {
                tasks.ListTasks.Add(t);
            }
        }
    }
    AddRadioButton(tasks,desk);
}

```

Листинг 3.3 – описание действия при нажатии на кнопку «Мои задачи»

При нажатии пользователем на кнопку «Команда» панель «desk» очищается, после происходит чтение сотрудников из файлов программы с дальнейшим выводом их имени в панель «desk» в виде списка. Данный список представлен в виде объекта «label» из платформы «Windows Form» (см. листинг 3.4). При наведении курсора мыши на имя сотрудника текст «label» становится синего цвета, что подсказывает пользователю, что на него можно нажать. При нажатии открывается окно для изменения или удаления сотрудника. В правом нижнем углу отображается кнопка «Добавить сотрудника», при нажатии на которую открывается форма для добавления сотрудника.

```

desk.Controls.Clear();
Button addTask = new Button();
addTask.Location = new Point(500, 620);
addTask.Size = new Size(300, 50);
addTask.Text = "Добавить сотрудника";
addTask.BackColor = Color.White;
addTask.Font = new Font("Srbija Sans", 16);
addTask.MouseClick += addWorker_MouseClick;
desk.Controls.Add(addTask);
StreamReader sr = new StreamReader($"users/{Login}/worker.txt");
string line;
string[] masname;
int y = 20;
while ((line=sr.ReadLine()) != null)
{
    if (string.IsNullOrEmpty(line)) { continue; }
    masname = line.Split('*');
    Label name= new Label();
    name.Location = new Point(50, y);
    name.Size = new Size(800, 20);
    if (y < 500) {y += 25;}
    else {y = 515;}
    name.Text = masname[0];
    name.Font = new Font("Srbija Sans", 14);
}

```



```

name.MouseClick += DeleteWorker_MouseClick;
name.Cursor = Cursors.Hand;
name.MouseHover += name_input_MouseHover;
name.MouseLeave += name_MouseLeave;
desk.Controls.Add(name);
}
sr.Close();

```

Листинг 3.4 – функция добавления сотрудников в панель «desk»

В данном окне может быть ещё множество кнопок. Это зависит от добавленных пользователем проектов. При загрузке окна происходит чтение имен всех файлов из папки имя пользователя из папки «project», далее создается кнопка с текстом названия файла и добавляется в панель меню (см. листинг 3.5).

```

int y=180;
foreach (string file in Directory.EnumerateFiles($"users/{Login}/project", "*",
SearchOption.AllDirectories))
{
string name = file.Split('#')[1];
Button test = new Button();
test.Location = new Point(20, y);
if (y < 500) {y += 40;}
else {y = 515;}
test.Size = new Size(260, 30);
test.Text = $"{name.Substring(0, name.Length - 4)}";
test.BackColor = Color.White;
test.Cursor = Cursors.Hand;
test.Font = new Font("Srbija Sans", 10);
test.MouseClick += project_MouseClick;
menu.Controls.Add(test);
}

```

Листинг 3.5 – функция добавления кнопок в панель «menu»

При нажатии на каждую новую кнопку в панель «desk» панель очищается. Далее, исходя из текста в кнопке, из файла с таким же названием десериализируется коллекция объектов класса «Tasks», после чего создается объект «radio button» из платформы «Windows Form» (см. листинг 3.6). У этой кнопки в тексте содержатся некоторые свойства объектов: название, сотрудник, дата окончания. Также в зависимости от параметра сложности объекта «Tasks» текст кнопки будет окрашен в определённый цвет:

- 1) 0 – черный;
- 2) 1 – зеленый;
- 3) 2 – жёлтый;
- 4) 3 – красный.

```

private void AddRadioButton(MasTasks Listtask, Panel panel)
{

```

```

masplus = new List<PictureBox>();
int y = 40;
foreach(Tasks t in Listtask.ListTasks)
{
    RadioButton rb= new RadioButton();
    rb.Name = $"{y}";
    rb.Location = new Point(50,y);
    rb.Size = new Size(500,40);
    rb.Text= t.Writer();
    switch (t.level)
    {
        case 1:
            rb.ForeColor = Color.Green;break;
        case 2:
            rb.ForeColor = Color.FromArgb(176, 171, 16);break;
        case 3:
            rb.ForeColor = Color.Red; break;
        default:break;
    }
    rb.Font= new Font("Times New Roman", 14);
    rb.MouseClick += rb_MouseClick;
    panel.Controls.Add(rb);
    PictureBox more= new PictureBox();
    more.Size = new Size(20, 20);
    more.Name= $"{y}";
    more.Tag = t;
    more.Location = new Point(810, y);
    more.SizeMode = PictureBoxSizeMode.StretchImage;
    more.ImageLocation = @"images\plus.png";
    more.Cursor = Cursors.Hand;
    more.MouseClick += More_MouseClick;
    more.MouseHover += More_MouseHover;
    more.MouseLeave += More_MouseLeave;
    masplus.Add(more);
    panel.Controls.Add(more);
    if (y < 500) { y += 32;}
    else {y = 515;}
}
}

```

Листинг 3.6 – функция добавления кнопок задач

При нажатии на задачу, представленную в виде «radio button» происходит удаление задачи из файла. В начале осуществляется десериализация коллекции, содержащей все задачи, после чего, забирая текст из нажатой кнопки мы сравним три параметра с каждым объекта класса «Tasks» в этой коллекции, после этого удалим как задачу, так и кнопку. В конце лист будет заново сериализован (см. листинг 3.7).

```

private void rb_MouseClick(object sender, MouseEventArgs e)
{
    RadioButton Buf = (RadioButton)sender;
    MasTasks ListTask;
    ListTask = Desserialized(NameProject);
    string[] masstr = { "; " };
    string[] mas = Buf.Text.Split(masstr, StringSplitOptions.RemoveEmptyEntries);
    Tasks buffer= new Tasks();
    foreach (Tasks t in ListTask.ListTasks)
    {

```

```

        if ((t.Name==mas[0]) && (t.worker== mas[1]) &&
(t.date.ToString("dd/MM/yyyy")== mas[2]))
        {
            buffer= t;
            foreach(PictureBox p in masplus)
            {
                if (p.Name==Buf.Name)
                {
                    p.Dispose();
                    masplus.Remove(p);
                    break;
                }
            }
            Buf.Dispose();
        }
        ListTask.ListTasks.Remove(buffer);
        Serealize(ListTask);
    }
}

```

Листинг 3.7 – функция удаления задачи

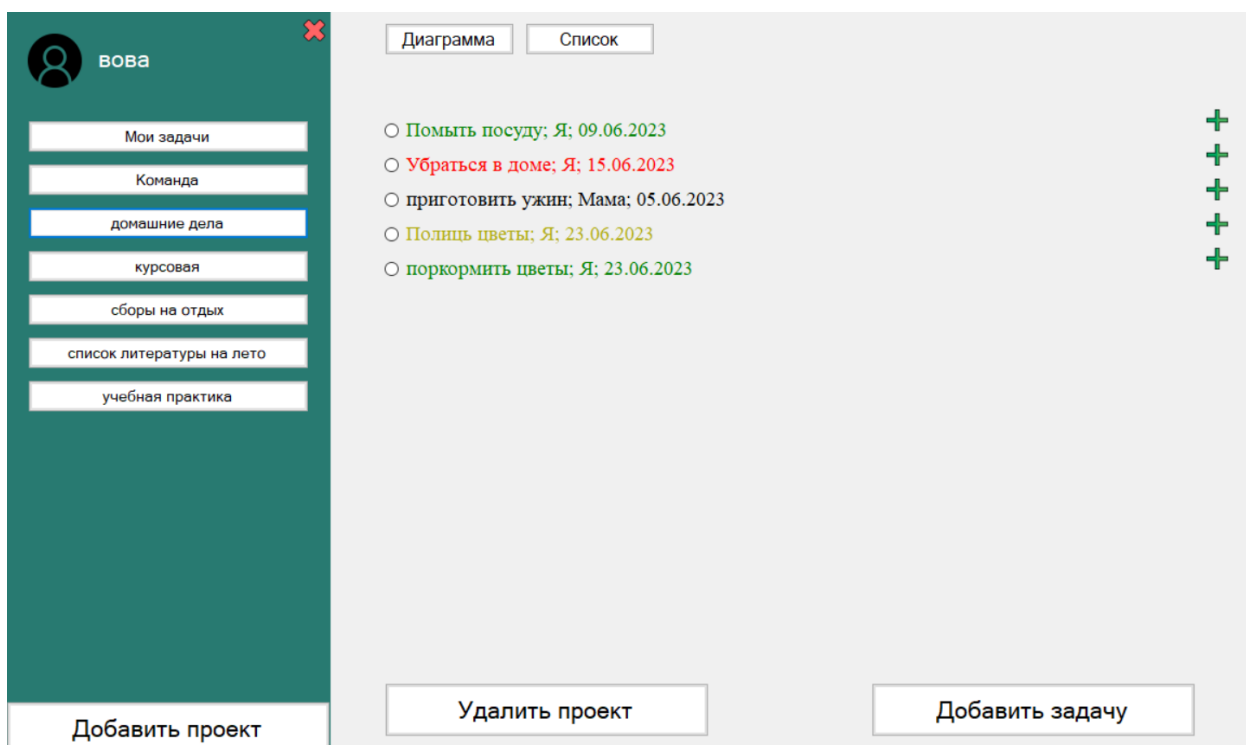


Рисунок 3.3 – окно меню с отображением проекта

Также при нажатии на кнопку проекта по мимо добавления списка добавляются 4 кнопки:

- 1) «Добавить задачу»;
- 2) «Удалить проект»;
- 3) «Диаграмма»;
- 4) «Список».

Первая кнопка открывает окно для добавления задач. Вторая кнопка удаляет проект, файл с проектом и обновляет форму с меню. Две последних кнопки меняют вид отображения задач третья отображает в виде диаграммы Ганта, четвертая отображает в виде списка описанного ранее (см. рис. 3.3).

3.4.1. Диаграмма Ганта

При нажатии на кнопку проекта появляется кнопка «Диаграмма», которая отображает задачи в данном проекте в виде диаграммы Ганта (см. рис. 3.4). На панели «desk» появляется объект «datagridview» из платформы «Windows Form» представляющий из себя таблицу. При инициализации этого объекта происходит десериализация коллекции объектов «Tasks», откуда название каждого объекта становится названием отдельной строки. Также для добавления столбцов создаются два массива, в первом даты начала задач, а во втором даты конца задач. После этого массивы сортируются с помощью пирамидальной сортировки.

```
private void addrows(string NameProject)
{
    ganta.ClearSelection();
    int i = 0;
    MasTasks mt = Desserialized(NameProject);
    addColumn(mt);
    ganta.DefaultCellStyle.Font = new Font("Srbija Sans", 32);
    foreach (Tasks t in mt.ListTasks)
    {
        ganta.Rows.Add();
        ganta.Rows[i].HeaderCell.Value = string.Format(t.Name, "0");
        i++;
    }
    paintingCells(mt);
}

private void addColumn(MasTasks mt)
{
    int i = 0;
    DateTime[] mas = new DateTime[mt.ListTasks.Count()];
    DateTime[] masSt = new DateTime[mt.ListTasks.Count()];
    foreach (Tasks t in mt.ListTasks)
    {
        mas[i] = t.date.Date;
        masSt[i] = t.dateSt.Date;
        i++;
    }
    sorting(mas, mas.Length);
    sorting(masSt, masSt.Length);
    List<DateTime> allDates = new List<DateTime>();
    for (DateTime date = masSt[0].Date; date <= mas[masSt.Length - 1].Date;
        date = date.AddDays(1)) { allDates.Add(date); }
    foreach (DateTime t in allDates)
    {
        DataGridViewColumn column = new DataGridViewColumn();
```

```

column.HeaderText = t.ToString("dd/MM");
column.CellTemplate = new DataGridViewTextBoxCell();
ganta.Columns.Add(column);
}
}

```

Листинг 3.8 – функции добавления строчек и столбцов в таблицу

Столбцы добавляются диапазоном дат от первого элемента первого массива до последнего элемента второго массива. Для покраски ячеек создается объект класса «Dictionary<TKey,TValue>», где ключом является название десериализованной задачи, а значением является коллекция объектов «DateTime», куда входят диапазон дат между началом данной задачи и её окончанием. Далее происходит сравнение между ключом и названием строки, при соответствии коллекция дат, приведённых к строке, сравнивается с названием строки. Если совпало, то ячейка закрашивается, если нет, то ячейка остается белой (см. рис. 3.4), (см. листинг 3.8, 3.9).

```

private void paintingCells(MasTasks mt)
{
Dictionary<string, List<string>> dict = new Dictionary<string, List<string>>();
foreach (Tasks t in mt.ListTasks)
{
    List<string> list = new List<string>();
    for (DateTime date = t.dateSt.Date; date <= t.date.Date; date =
date.AddDays(1))
    { list.Add(date.ToString("dd/MM")); }
    dict.Add(t.Name, list);
}
Random random = new Random();
for (int i = 0; i < ganta.Rows.Count; i++)
{
    int k = 0;
    List<string>list = dict[ganta.Rows[i].HeaderCell.Value.ToString()];
    Color color = Color.FromArgb(random.Next(0, 255), random.Next(0,
255), random.Next(0, 255));
    for (int j = 0; j < ganta.Columns.Count; j++)
    {
        if (list[k] == ganta.Columns[j].HeaderCell.Value.ToString())
        {
            ganta.Rows[i].Cells[j].Style.BackColor = color;
            k++;
            if(k==dict[$"{ganta.Rows[i].HeaderCell.Value}"].Count())
            { k--; }
        }
    }
}
}
}

```

Листинг 3.9 – функции окрашивания ячеек таблицы

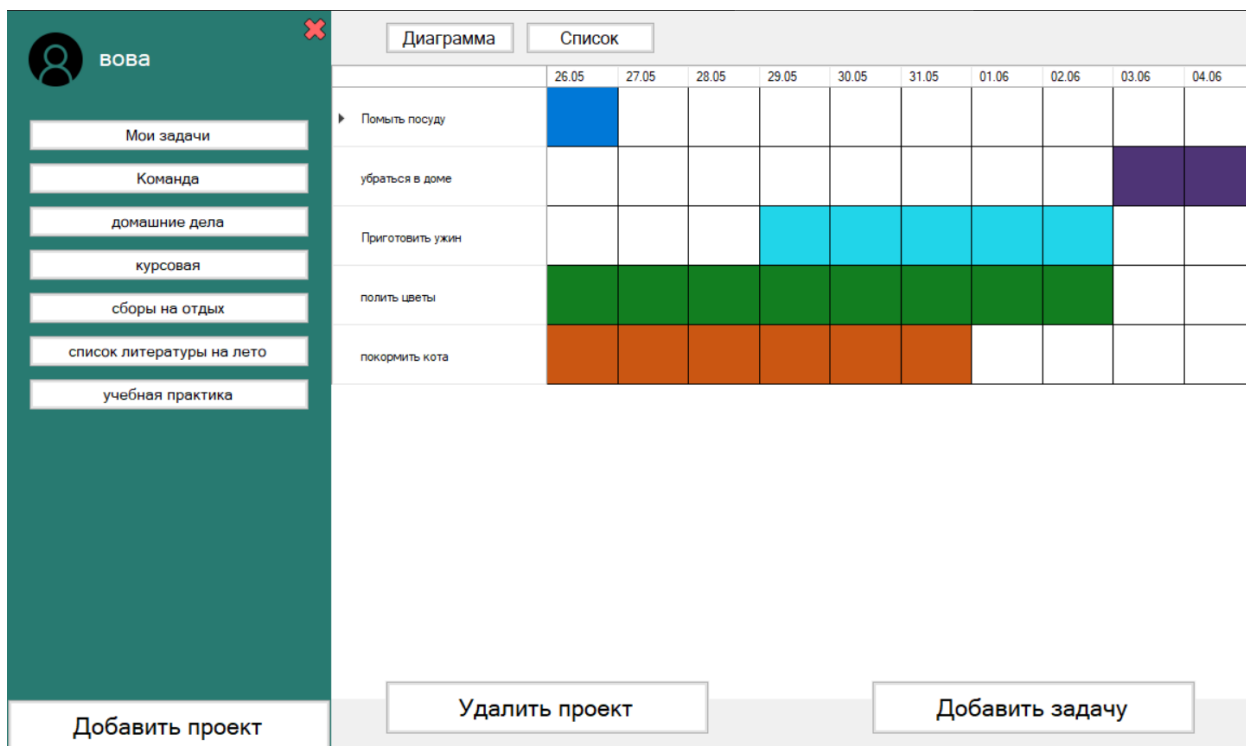


Рисунок 3.4 – окно меню с диаграммой Ганта

3.5. Окно добавления проекта

В данном окне представлено поле для ввода данных, принимающее название для нового проекта. Также здесь находится объект «CheckBox» из платформы «Windows Form», в котором представлен список со множественным выбором имен всех сотрудников, сохранённых в файлах программы. На форме находятся две кнопки: «Назад» и «Добавить» (см. рис. 3.5). При нажатии на первую данная форма закрывается и открывается окно меню. При нажатии на вторую создается файл с расширением «.txt» где названием является текст, введённый в поле для ввода данных. После этого собираются имена всех отмеченных сотрудников. Далее программа считывает индексы строк этих рабочих в массив, который передается в функцию для перезаписи файла, добавляя к отмеченным рабочим новый проект. После чего форма закрывается и открывается окно меню (см. листинг 3.10, 3.11).

```
private void ButAdd_MouseClick(object sender, MouseEventArgs e)
{
    File.Create($"@users\{login}\project\#{nameproject.Text}.txt").Close();
    int[] mas = new int[index];
    int indexnew = 0;
    int j=0;
```

```

string line;
string[] masline;
foreach(string t in workercheck.CheckedItems)
{
    StreamReader sr = new StreamReader($"users\\{login}\\worker.txt");
    while ((line = sr.ReadLine()) != null)
    {
        masline = line.Split('*');
        if (t == masline[0])
        {
            mas[j++] = indexnew;
        }
        indexnew++;
    }
    sr.Close();
}
rewrite(mas);
MenuForm menu = new MenuForm(login);
menu.Show();
this.Close();
}

```

Листинг 3.10 – описание действий при нажатии на кнопки «Добавить»

Рисунок 3.5 – интерфейс модуля добавления проекта

```

private void rewrite(int[] mas)
{
    int i = 0;
    int j = 0;
    string path = $"users/{login}/worker.txt";
    string tempPath = path + ".txt";
    using (StreamReader sr = new StreamReader(path))
    using (StreamWriter sw = new StreamWriter(tempPath)) {
        while (!sr.EndOfStream)
        {
            string line = sr.ReadLine();
            if (mas[j] == i)
            {
                sw.WriteLine(line+"*"+nameproject.Text);
                j++;
            }
            else { sw.WriteLine(line); }
        }
    }
}

```

```

        i++;
    }
}
File.Delete(path);
File.Move(tempPath, path);
}

```

Листинг 3.11 – функция перезаписи файла

3.6. Окно Добавления сотрудника

Данное окно довольно похоже на окно добавления проекта, но здесь представлен список с множественным выбором из всех проектов (см. листинг 3.12). Кнопка «Назад» работает идентично, как в окне добавления проекта. Кнопка «Добавить» записывает сотрудника в файл программы следующим образом, в начале идет имя из поля для ввода данных, после через разделитель в виде знака «*» записываются выбранные из списка проекты (см. листинг 3.13).

```

private void addProject(string Login)
{
    foreach (string file in Directory.EnumerateFiles($"users/{Login}/project", "*",
        SearchOption.AllDirectories))
    {
        string name = file.Split('#')[1];
        ChooseProject.Items.Add($"{name.Substring(0, name.Length - 4)}");
    }
}

```

Листинг 3.12 – функция добавления проекта в список

```

virtual internal void AddBut_MouseClick(object sender, MouseEventArgs e)
{
    string work = NameText.Text;
    foreach(string it in ChooseProject.CheckedItems)
    {
        work+= $"{it}";
    }
    using (StreamWriter sw = new StreamWriter($"users/{Login}/worker.txt",
        true, Encoding.Default))
    {
        sw.WriteLine(work);
    }
    MessageBox.Show("успешно");
}

```

Листинг 3.13 – функция записи сотрудника в файл

3.7. Окно изменения, удаления сотрудника

Данная форма наследуется от формы добавления сотрудника. Это окно вызывается при нажатии на имя конкретного сотрудника в форме меню, поэтому в конструкторе данная форма принимает имя сотрудника, на которого нажал пользователь для занесения в поле для ввода данных. Также по имени

программа ищет сотрудника в файлах, после чего считывает к каким проектам он относится. Эти проекты будут отмечены в списке со множественным выбором изначально. В данном окне изменён функционал и текст кнопок. Первая «Изменить» собирает данные из поля для ввода данных и списка после чего ищет сотрудника в файле и забирает индекс строки, на которой он записан, после этого срабатывает функция перезаписи файла, в которой эта строчка изменяется (см. листинг 3.14). Вторая кнопка «Удалить» работает идентично, но для перезаписи передает пустую строку.

```
override internal void AddBut_MouseClick(object sender, MouseEventArgs e)
{
    StreamReader sr = new StreamReader($"users/{Login}/worker.txt");
    string line;
    string[] masproject;
    int index=0;
    while ((line = sr.ReadLine()) != null)
    {
        masproject = line.Split('*');
        if (masproject[0] == Name)
        {
            break;
        }
        index++;
    }
    sr.Close();
    string work = NameText.Text;
    foreach (string it in ChooseProject.CheckedItems)
    {
        work += $"{it}";
    }
    RewriteLine(index,work);
    MenuForm menu = new MenuForm(Login);
    menu.Show();
    MessageBox.Show("Сотрудник изменён");
    this.Close();
}
```

Листинг 3.14 – функция изменения сотрудника в файле

3.8. Окно добавления задачи

В интерфейсе данного окна представлены различные поля для ввода данных для конкретного свойства объекта класса «Tasks». Для названия и описания отображаются обычные текстовые поля, для выбора сотрудника есть выпадающий список. Этот список пополняется с помощью метода, в котором считывается файл с сотрудниками и если у сотрудника есть доступ к проекту, в котором создается задача, то он добавится в список (см. листинг 3.15). Для выбора дат на интерфейсе представлены два календаря. Также в форме

расположены две кнопки: «Назад» и «Добавить» (см. рис. 3.6). Первая закрывает данное окно. Вторая собирает введенные пользователем в поля данные, после этого создает объект класса «Tasks». Далее происходит десериализация из файла проекта всех задач в коллекцию проектов и добавляет в эту коллекцию новый объект, после чего коллекция обратно сериализуется в файл (см. листинг 3.15).

Рисунок 3.6 – интерфейс окна добавления задачи

```
private void WorkerAdd(ComboBox Worker)
{
    StreamReader sr = new StreamReader($"users\\{Login}\\worker.txt");
    string line; string[] mas;
    while ((line = sr.ReadLine()) != null)
    {
        mas = line.Split('*');
        for(int i = 1; i < mas.Length; i++)
        {
            if (mas[i] == NameProject)Worker.Items.Add(mas[0]);}
        }
    }
    sr.Close();
}
private void ButAdd_MouseClick(object sender, MouseEventArgs e)
{
    Tasks          Newtask          = new
        Tasks(NameTask.Text,Worker.Text,level,Description.Text,date.Value,
            dateSt.Value);
    MasTasks Listtas = new MasTasks();
    Listtas= Desserialized(Listtas);
```

```
Listtas.ListTasks.Add(Newtask);  
Serealize(Listtas);  
MessageBox.Show("Задача добавлена");  
}
```

Листинг 3.15 – функции добавления сотрудников и сохранения задач

3.9. Окно просмотра задачи

Данное окно по интерфейсу наследуется от окна для добавления задач. Единственное отличие в том, что в форме одна кнопка вместо двух. В форме меню справа от задачи находится кнопка в виде плюса, при нажатии на которую осуществляется переход к данному окну. В данной форме в полях для ввода данных представлены данные определенной задачи, выбранной пользователем. Здесь представлена кнопка «Назад», закрывающая эту форму.

Выводы

В данной главе разработана программа на основании модулей, представленных в предыдущей главе. Были описаны 8 окон: окно входа, окно регистрации, окно меню, окно добавления сотрудников, окно изменения сотрудника, окно добавления задач, окно просмотра задач, окно добавления проектов. Также были представлен класс «Tasks» созданный для удобного хранения задач при разработке и использования приложения. В данной главе представлен метод создания диаграммы Ганта в программе. Для каждой формы далее описаны принципы работы, методы, функционирующие внутри программы и атрибуты интерфейса для взаимодействия с пользователем. На протяжении всей главы расположены листинги кода программы, с детальным описанием принципа работы каждой функции, представленной в них. В главе можно увидеть скриншоты пользовательского интерфейса получившегося приложения почти для каждой формы.

Заключение

Поставленные задачи выполнены, поставленная цель достигнута – была разработана программа:

- 1) было проведено исследование, выделение преимуществ и недостатков существующих решений для управления проектами;
- 2) было осуществлено моделирование логики и структуры программы;
- 3) была разработана программа с возможностью составления плана проекта разделения крупных задач на конкретные подзадачи разного уровня сложности реализации для дальнейшего делегирования между специалистами и установлением сроков сдачи работ.

В первой главе был произведен анализ 4 программ для управления проектами: «YouGile», «Habitica», «Asana», «Todoist». Каждое приложение имеет краткое описание функционала с выявлением преимуществ и недостатков. Также были проанализированы в соответствии с разобранными критериями: количество возможных пользователей; виды представления задач; количество проектов, задач; возможность коммуникации между сотрудниками внутри приложения; возможность интеграций с другими приложениями; возможность создания отчетов. Для удобства была представлена сравнительная таблица. Был указан инструментальный аппарат. Итогом стала постановка задач для программы, учитывая все плюсы и минусы разобранных программ.

Во второй главе была описана функциональная работа программы. Приложение было поделено на 10 различных модулей: модуль ввода логина и пароля, модуль регистрации, модуль меню, модуль добавления, изменения проекта, модуль проекта, модуль добавления задачи, модуль команды, модуль добавления сотрудника, модуль «Мои задачи», каждый из которых отвечал за свою часть приложения. После было представлено краткое описание работы программы с указанием на связанные между собой модули. Также в главе представлено подробное описание особенностей и алгоритмов каждого

модуля с приложением подробной блок-схемы для упрощенного планирования действий при разработке программы.

В третьей главе разработана программа на основании модулей, представленных в предыдущей главе. Были описаны 8 окон: окно входа, окно регистрации, окно меню, окно добавления сотрудников, окно изменения сотрудника, окно добавления задач, окно просмотра задач, окно добавления проектов. Также были представлен класс «Tasks» созданный для удобного хранения задач при разработке и использования приложения. В данной главе представлен метод создания диаграммы Ганта в программе. Для каждой формы далее описаны принципы работы, методы, функционирующие внутри программы и атрибуты интерфейса для взаимодействия с пользователем. На протяжении всей главы расположены листинги кода программы, с детальным описанием принципа работы каждой функции, представленной в них. В главе можно увидеть скриншоты пользовательского интерфейса получившегося приложения почти для каждой формы.

Список использованной литературы

1. Менеджмент ИТ-проекта / Хабр [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/169693/>. Дата обращения: 18.03.23.
2. Управление инновационными проектами и программами / Батраева И.М., Сумина Е.В. // КиберЛенинка [Электронный ресурс]. Режим доступа: <https://cyberleninka.ru/article/n/upravlenie-innovatsionnymi-proektami-i-programmami>. Дата обращения: 18.03.23.
3. Управление проектами и приоритетными программами / Юрьева Т.В. // КиберЛенинка [Электронный ресурс]. Режим доступа: <https://cyberleninka.ru/article/n/upravlenie-proektami-i-prioritetnymi-programmami/viewer>. Дата обращения: 18.03.23.
4. Современные методы управления проектами / Сербская О.В. // КиберЛенинка [Электронный ресурс]. Режим доступа: <https://cyberleninka.ru/article/n/sovremennye-metody-upravleniya-proektami/viewer>. Дата обращения: 18.03.23.
5. Обзор 10 бесплатных систем управления проектами. Что даром, а за что придется платить / Хабр [Электронный ресурс]. Режим доступа: <https://habr.com/ru/companies/yougile/articles/537920/>. Дата обращения: 20.03.23.
6. 52 системы управления проектами для командной работы в разных сферах / Хабр [Электронный ресурс]. Режим доступа: <https://habr.com/ru/companies/yougile/articles/545614/>. Дата обращения: 20.03.23.
7. Краткий обзор языка C# / Microsoft [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>. Дата обращения: 05.04.2023.
8. Руководство по классическим приложениям (Windows Forms.NET) / Microsoft [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ruru/dotnet/desktop/winforms/overview/?view=netdesktop-6.0>. Дата обращения: 05.04.2023.

9. DateTime Структура (System) / Microsoft Learn [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/system.datetime?view=net-8.0>. Дата обращения: 4.05.2023.

10. Бесплатное обучение системе управления проектами / YouGile [Электронный ресурс]. Режим доступа: <https://ru.yougile.com/training>. Дата обращения: 20.03.23.

11. Control.Tag Свойство (System.Windows.Forms) / Microsoft Learn [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.forms.control.tag?view=windowsdesktop-6.0>. Дата обращения: 04.05.2023.

12. Dictionary<TKey,TValue> Class (System.Collections.Generic) / Microsoft Learn [Электронный ресурс]. Режим доступа: <https://learn.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=net-7.0>. Дата обращения: 06.05.2023.

Приложения

Приложение А. Листинг кода

Приложение А.1. Программный код окна входа в программу

```
public partial class input : Form
{
    public input()
    {
        InitializeComponent();
        check2.Hide();
        StartPosition= FormStartPosition.CenterScreen;
    }
    private void textBox2_Click(object sender, EventArgs e)
    {
        Password.Clear();
        Password.UseSystemPasswordChar = true;
    }
    private void textBox1_Click(object sender, EventArgs e)
    {
        Login.Clear();
    }

    private void buttonClose_MouseClick(object sender, MouseEventArgs e)
    {
        Application.Exit();
    }
    private void button1_MouseClick(object sender, MouseEventArgs e)
    {
        this.Hide();
        registrationForm registrationForm = new registrationForm();
        registrationForm.Show();
    }

    string check(string login,string password)
    {
        string filepath = @"users.txt";
        StreamReader sr = new StreamReader(filepath);
        string line;
        string[] mas = new string[2];
        while ((line = sr.ReadLine()) != null)
        {
            mas = line.Split(',');
            if (mas[0] == login)
            {
                sr.Close();
                return mas[1];
            }
        }
        sr.Close();
        return "0";
    }
    private async void button_input_MouseClick(object sender, MouseEventArgs e)
    {
        string login = Login.Text;
        string password = Password.Text;
        string pas = check(login, password);
        if (String.Equals(password,pas))
        {
            this.Hide();
            MenuForm Menu = new MenuForm(login);
            Menu.Show();
        }
    }
}
```



```

        else
        {
            check2.Show();
            await Task.Delay(1000);
            check2.Hide();
        }
    }
    Point lastPoint;
    private void panel1_MouseMove(object sender, MouseEventArgs e)
    {
        if (e.Button == MouseButtons.Left)
        {
            this.Left += e.X - lastPoint.X;
            this.Top += e.Y - lastPoint.Y;
        }
    }
    private void panel1_MouseDown(object sender, MouseEventArgs e)
    {
        lastPoint = new Point(e.X, e.Y);
    }
}

```

Приложение А.2. Программный код окна регистрации

```

public partial class registrationForm : Form
{
    public registrationForm()
    {
        InitializeComponent();
        check.Hide();
        StartPosition = FormStartPosition.CenterScreen;
    }
    private void Password_MouseClick(object sender, MouseEventArgs e)
    {
        Password.Clear();
    }

    private void Login_MouseClick(object sender, MouseEventArgs e)
    {
        Login.Clear();
    }

    private void OutBut_MouseClick(object sender, MouseEventArgs e)
    {
        this.Close();
        input input = new input();
        input.Show();
    }

    private async void RegistrationBut_MouseClick(object sender, MouseEventArgs e)
    {
        if ((Login.Text == "") || (Login.Text == "Введите логин"))
        {
            MessageBox.Show("Данный логин недопустим");
            return;
        }
        string login = Login.Text;
        string password = Password.Text;
        string filepath = @"users.txt"; //путь к файлу csv
        StreamReader sr = new StreamReader(filepath);
        string line;
        string[] mas = new string[2];
        bool flag = false;
        while ((line = sr.ReadLine()) != null)

```

```

    {
        mas = line.Split(',');
        if (mas[0] == login)
        {
            flag = true;
        }
    }
    sr.Close();
    if (!flag)
    {
        Directory.CreateDirectory($"users\\{login}");
        Directory.CreateDirectory($"users\\{login}\\project");
        File.Create($"users\\{login}\\worker.txt").Close();
        using (StreamWriter sw = new StreamWriter(filepath, true,
            Encoding.UTF8))
        {
            sw.WriteLine($"{login},{password}");
        }
        MessageBox.Show("успешно");
    }
    else
    {
        check.Show();
        await Task.Delay(1000);
        check.Hide();
    }
}
Point lastPoint;
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}
private void panel1_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
}

```

Приложение А.3. Программный код окна меню

```

public partial class MenuForm : Form
{
    public string Login;
    Panel menu;
    Panel desk;
    Panel hat;
    DataGridView ganta;

    public string NameProject;
    public MenuForm(string login)
    {
        InitializeComponent();
        Login = login;
    }
}

```

```

        this.StartPosition = FormStartPosition.CenterScreen;
        this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.None;
    }

    private void MenuForm_Load(object sender, EventArgs e)
    {
        //menu
        menu = new Panel();
        menu.Dock = DockStyle.Left;
        menu.Size = new Size(300, 574);
        menu.BackColor = Color.FromArgb(40, 122, 113);
        menu.AutoScroll = true;
        menu.MouseMove += panel_MouseMove;
        menu.MouseDown += panel_MouseDown;
        Controls.Add(menu);

        //desk
        desk = new Panel();
        desk.Dock = DockStyle.Right;
        desk.Size = new Size(ClientSize.Width - menu.Width, ClientSize.Height);
        desk.AutoScroll = false;
        desk.MouseMove += panel_MouseMove;
        desk.MouseDown += panel_MouseDown;
        Controls.Add(desk);

        PictureBox close=new PictureBox();
        close.Location=new Point(menu.Width-25, 7);
        close.Size = new Size(20,20);
        close.ImageLocation = @"images\close.png";
        close.SizeMode = PictureBoxSizeMode.StretchImage;
        close.MouseClick += Close_MouseClick;
        close.Cursor= Cursors.Hand;
        menu.Controls.Add(close);

        //login
        PictureBox pictureBox = new PictureBox();
        pictureBox.Location = new Point(20, 20);
        pictureBox.Size = new Size(50, 50);
        pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
    }

```

```

pictureBox.ImageLocation = @"images\user.png";
menu.Controls.Add(pictureBox);

//ник нейм пользователя
Label Login = new Label();
Login.Location = new Point(80, 30);
Login.Size = new Size(150, 30);
Login.Text = this.Login;
Login.Font = new Font("Srbija Sans", 16);
Login.ForeColor = Color.White;
menu.Controls.Add(Login);

//добавить проект
Button add = new Button();
add.Dock = DockStyle.Bottom;
add.Size = new Size(150, 50);
add.BackColor = Color.White;
add.Text = "Добавить проект";
add.Font = new Font("Srbija Sans", 16);
add.TabIndex = 1;
add.Cursor = Cursors.Hand;
menu.Controls.Add(add);
add.MouseClick += add_MouseClick;

// мои задачи
Button task = new Button();
task.Location = new Point(20, 100);
task.Size = new Size(260, 30);
task.Text = "Мои задачи";
task.BackColor = Color.White;
task.Font = new Font("Srbija Sans", 10);
task.Cursor = Cursors.Hand;
task.MouseClick += MyTask_MouseClick;
menu.Controls.Add(task);

// команда
Button team = new Button();
team.Location = new Point(20, 140);
team.Size = new Size(260, 30);

```

```

        team.Text = "Команда";
        team.Font = new Font("Srbija Sans", 10);
        team.BackColor = Color.White;
        team.Cursor = Cursors.Hand;
        team.MouseClick += TeamAdd_MouseClick;
        menu.Controls.Add(team);

        addButton(this.Login);
    }
    private void Close_MouseClick(object sender, MouseEventArgs e)
    {
        System.Windows.Forms.Application.Exit();
    }
    private void MyTask_MouseClick(object sender, MouseEventArgs e)
    {
        desk.Controls.Clear();
        MasTasks tasks= new MasTasks();
        foreach (string file in Directory.EnumerateFiles($"users/{Login}/project",
        "*", SearchOption.AllDirectories))
        {
            foreach(Tasks t in Desserialized(file, 0).ListTasks)
            {
                if (t.worker=="Я")
                {
                    tasks.ListTasks.Add(t);
                }
            }
        }
        AddRadioButton(tasks,desk);
    }
    private void TeamAdd_MouseClick(object sender, MouseEventArgs e)
    {
        desk.Controls.Clear();
        Button addTask = new Button();
        addTask.Location = new Point(500, 620);
        addTask.Size = new Size(300, 50);
        addTask.Text = "Добавить сотрудника";
        addTask.BackColor = Color.White;
        addTask.Font = new Font("Srbija Sans", 16);
    }

```

```

addTask.MouseClick += addWorker_MouseClick;
desk.Controls.Add(addTask);

StreamReader sr = new StreamReader($"users/{Login}/worker.txt");
string line;
string[] masname;
int y = 20;
while ((line=sr.ReadLine()) != null)
{
    if (string.IsNullOrEmpty(line)) { continue; }
    masname = line.Split('*');
    Label name= new Label();
    name.Location = new Point(50, y);
    name.Size = new Size(800, 25);
    if (y < 500)
    {
        y += 30;
    }
    else
    {
        y = 515;
    }
    name.Text = masname[0];
    name.Font = new Font("Srbija Sans", 14);
    name.MouseClick += DeleteWorker_MouseClick;
    name.Cursor = Cursors.Hand;
    name.MouseHover += name_input_MouseHover;
    name.MouseLeave += name_MouseLeave;
    desk.Controls.Add(name);
}
sr.Close();
}

private void name_input_MouseHover(object sender, EventArgs e)
{
    Label name = (Label)sender;
    name.ForeColor = Color.Blue;
}

private void name_MouseLeave(object sender, EventArgs e)
{

```

```

        Label name = (Label)sender;
        name.ForeColor = Color.Black; // ставим черный цвет текста
    }
    private void DeleteWorker_MouseClick(object sender, MouseEventArgs e)
    {
        Label buf=(Label)sender;
        DelWorkerForm form = new DelWorkerForm(buf.Text, Login);
        this.Close();
        form.Show();
    }
    private void addWorker_MouseClick(object sender, MouseEventArgs e)
    {
        AddWorkerForm AddWorker= new AddWorkerForm(Login);
        AddWorker.Show();
    }

    private void addButton(string Login)
    {
        int y=180;
        foreach (string file in Directory.EnumerateFiles($"users/{Login}/project",
        "*", SearchOption.AllDirectories))
        {
            string name = file.Split('#')[1];
            Button test = new Button();
            test.Location = new Point(20, y);
            if (y < 500) {y += 40;}
            else {y = 515;}
            test.Size = new Size(260, 30);
            test.Text = $"{name.Substring(0, name.Length - 4)}";
            test.BackColor = Color.White;
            test.Cursor = Cursors.Hand;
            test.Font = new Font("Srbija Sans", 10);
            test.MouseClick += project_MouseClick;
            menu.Controls.Add(test);
        }
    }
    private void add_MouseClick(object sender, MouseEventArgs e)
    {
        this.Close();
    }

```

```

        AddProjectForm AddProject = new AddProjectForm(Login);
        AddProject.Show();
    }

    private void project_MouseClick(object sender, MouseEventArgs e)
    {
        desk.Controls.Clear();
        Button Buf = (Button)sender;
        NameProject = Buf.Text;

        Button addTask = new Button();
        addTask.Location = new Point(500, 620);
        addTask.Size = new Size(300, 50);
        addTask.Text = "Добавить задачу";
        addTask.Font = new Font("Srbija Sans", 16);
        addTask.BackColor = Color.White;
        addTask.MouseClick += addTask_MouseClick;
        desk.Controls.Add(addTask);

        Button saveTask = new Button();
        saveTask.Location = new Point(50, 620);
        saveTask.Size = new Size(300, 50);
        saveTask.Text = "Удалить проект";
        saveTask.Font = new Font("Srbija Sans", 16);
        saveTask.BackColor = Color.White;
        saveTask.MouseClick += DelProject_MouseClick;
        desk.Controls.Add(saveTask);

        hat = new Panel();
        hat.Location = new Point(0, 50);
        hat.Size = new Size(ClientSize.Width - menu.Width, ClientSize.Height - 100);
        hat.MouseMove += panel_MouseMove;
        hat.MouseDown += panel_MouseDown;
        desk.Controls.Add(hat);

        Button diagramma = new Button();
        diagramma.Location = new Point(50, 10);
        diagramma.Size = new Size(120, 30);
        diagramma.Text = "Диаграмма";
    }

```



```

diagramma.BackColor = Color.White;
diagramma.Font = new Font("Srbija Sans", 12);
diagramma.MouseClick += Diagramma_MouseClick;
desk.Controls.Add(diagramma);

Button spisok= new Button();
spisok.Location = new Point(180, 10);
spisok.Size = new Size(120, 30);
spisok.Text = "Список";
spisok.BackColor = Color.White;
spisok.Font = new Font("Srbija Sans", 12);
spisok.MouseClick += spisok_MouseClick;
desk.Controls.Add(spisok);

AddRadioButton(Desserialized(NameProject),hat);
}
private void spisok_MouseClick(object sender, MouseEventArgs e)
{
    hat.Controls.Clear();
    AddRadioButton(Desserialized(NameProject),hat);
}
private void Diagramma_MouseClick(object sender, MouseEventArgs e)
{
    hat.Controls.Clear();
    ganta= new DataGridView();
    ganta.AllowUserToAddRows = false;
    ganta.AllowUserToDeleteRows = false;
    ganta.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
    ganta.AutoSizeRowsMode =
DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    ganta.BackgroundColor = SystemColors.ActiveCaption;
    ganta.ColumnHeadersHeightSizeMode =
DataGridViewColumnHeadersHeightSizeMode.AutoSize;
    ganta.Dock = DockStyle.Fill;
    ganta.GridColor = Color.Black;
    ganta.BackgroundColor = Color.White; ganta.BorderStyle = BorderStyle.None;
    ganta.Location = new Point(0, 0);
    ganta.Name = "dataGridView1";
    ganta.ReadOnly = true;

```

```

        ganta.RowHeadersWidth = 200;
        ganta.RowTemplate.Height = 24;
        ganta.Size = new Size(hat.Width, hat.Height-70);
        hat.Controls.Add(ganta);
        ganta.ColumnHeadersHeight = 200;
        ganta.SelectionChanged += ganta_SelectionChanged;
        addrows(NameProject);
    }
    private void addrows(string NameProject)
    {
        ganta.ClearSelection();
        int i = 0;
        if (new FileInfo($"users\{Login}\project\#{NameProject}.txt").Length == 0)
        {
            MessageBox.Show("Сначала добавьте задачу",
                "Ошибка",
                MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            return;
        }
        MasTasks mt = Desserialized(NameProject);
        addColumn(mt);
        ganta.DefaultCellStyle.Font = new Font("Srbija Sans", 32);
        foreach (Tasks t in mt.ListTasks)
        {
            ganta.Rows.Add();
            ganta.Rows[i].HeaderCell.Value = string.Format(t.Name, "0");
            i++;
        }
        paintingCells(mt);
    }
    private void addColumn(MasTasks mt)
    {
        int i = 0;
        DateTime[] mas = new DateTime[mt.ListTasks.Count()];
        DateTime[] masSt = new DateTime[mt.ListTasks.Count()];
        foreach (Tasks t in mt.ListTasks)
        {

```

```

        mas[i] = t.date.Date;
        masSt[i] = t.dateSt.Date;
        i++;
    }
    sorting(mas, mas.Length);
    sorting(masSt, masSt.Length);
    List<DateTime> allDates = new List<DateTime>();
    for (DateTime date = masSt[0].Date; date <= mas[masSt.Length - 1].Date;
date = date.AddDays(1))
    { allDates.Add(date);}
    foreach (DateTime t in allDates)
    {
        DataGridViewColumn column = new DataGridViewColumn();
        column.HeaderText = t.ToString("dd/MM");
        column.CellTemplate = new DataGridViewTextBoxCell();
        ganta.Columns.Add(column);
    }
}

private void paintingCells(MasTasks mt)
{
    Dictionary<string, List<string>> dict = new Dictionary<string,
List<string>>();
    foreach (Tasks t in mt.ListTasks)
    {
        List<string> list = new List<string>();
        for (DateTime date = t.dateSt.Date; date <= t.date.Date; date =
date.AddDays(1))
        { list.Add(date.ToString("dd/MM")); }
        dict.Add(t.Name, list);
    }
    Random random = new Random();
    for (int i = 0; i < ganta.Rows.Count; i++)
    {
        int k = 0;
        List<string> list = dict[ganta.Rows[i].HeaderCell.Value.ToString()];
        Color color = Color.FromArgb(random.Next(0, 255), random.Next(0, 255),
random.Next(0, 255));
        for (int j = 0; j < ganta.Columns.Count; j++)
        {

```

```

        if (list[k] == ganta.Columns[j].HeaderCell.Value.ToString())
        {
            ganta.Rows[i].Cells[j].Style.BackColor = color;
            k++;
            if (k == dict["{ganta.Rows[i].HeaderCell.Value}"].Count())
            {
                k--;
            }
        }
    }
}

private int AddPyramid(DateTime[] mas, int i, int N)
{
    int imax;
    DateTime buf;
    if ((2 * i + 2) < N)
    {
        if (mas[2 * i + 1] < mas[2 * i + 2]) imax = 2 * i + 2;
        else imax = 2 * i + 1;
    }
    else imax = 2 * i + 1;
    if (imax >= N) return i;
    if (mas[i] < mas[imax])
    {
        buf = mas[i];
        mas[i] = mas[imax];
        mas[imax] = buf;
        if (imax < N / 2) i = imax;
    }
    return i;
}

private void sorting(DateTime[] mas, int len)
{
    for (int i = len / 2 - 1; i >= 0; --i)
    {
        long prev_i = i;
        i = AddPyramid(mas, i, len);
        if (prev_i != i) ++i;
    }
}

```

```

    }
    DateTime buf;
    for (int k = len - 1; k > 0; --k)
    {
        buf = mas[0];
        mas[0] = mas[k];
        mas[k] = buf;
        int i = 0, prev_i = -1;
        while (i != prev_i)
        {
            prev_i = i;
            i = AddPyramid(mas, i, k);
        }
    }
}

private void ganta_SelectionChanged(object sender, EventArgs e)
{
    if (MouseButtons != System.Windows.Forms.MouseButtons.None)
        ((DataGridView)sender).CurrentCell = null;
}

List<PictureBox> masplus;
private void AddRadioButton(MasTasks Listtast, Panel panel)
{
    masplus = new List<PictureBox>();
    int y = 40;
    foreach(Tasks t in Listtast.ListTasks)
    {
        RadioButton rb= new RadioButton();
        rb.Name = $"{y}";
        rb.Location = new Point(50,y);
        rb.Size = new Size(500,40);
        rb.Text= t.Writer();
        switch (t.level)
        {
            case 1:
                rb.ForeColor = Color.Green;break;
            case 2:
                rb.ForeColor = Color.FromArgb(176, 171, 16);break;
        }
    }
}

```

```

        case 3:
            rb.ForeColor = Color.Red; break;
            default:break;
    }
    rb.Font= new Font("Times New Roman", 14);
    rb.MouseClick += rb_MouseClick;
    panel.Controls.Add(rb);
    PictureBox more= new PictureBox();
    more.Size = new Size(20, 20);
    more.Name= $"{y}";
    more.Tag = t;
    more.Location = new Point(810, y);
    more.SizeMode = PictureBoxSizeMode.StretchImage;
    more.ImageLocation = @"images\plus.png";
    more.Cursor = Cursors.Hand;
    more.MouseClick += More_MouseClick;
    more.MouseHover += More_MouseHover;
    more.MouseLeave += More_MouseLeave;
    masplus.Add(more);
    panel.Controls.Add(more);

    if (y < 500) { y += 32;}
    else {y = 515;}
}
}

private void More_MouseLeave(object sender, EventArgs e)
{
    (sender as PictureBox).Size = new Size(20, 20);
    (sender as PictureBox).Refresh();
}

private void More_MouseHover(object sender, EventArgs e)
{
    (sender as PictureBox).Size = new Size(25,25);
    (sender as PictureBox).Refresh();
}
}

```

```

private void More_MouseClick(object sender, MouseEventArgs e)
{
    PictureBox buf = (PictureBox)sender;
    WatchTask watchTask = new WatchTask(NameProject, Login, (Tasks)buf.Tag);
    watchTask.Show();
}

private void rb_MouseClick(object sender, MouseEventArgs e)
{
    RadioButton Buf = (RadioButton)sender;
    MasTasks ListTask;
    ListTask = Desserialized(NameProject);
    string[] masstr = { "; " };
    string[] mas = Buf.Text.Split(masstr,
StringSplitOptions.RemoveEmptyEntries);
    Tasks buffer= new Tasks();
    foreach (Tasks t in ListTask.ListTasks)
    {
        if ((t.Name==mas[0]) && (t.worker== mas[1]) &&
(t.date.ToString("dd/MM/yyyy")== mas[2]))
        {
            buffer= t;
            foreach(PictureBox p in masplus)
            {
                if (p.Name==Buf.Name)
                {
                    p.Dispose();
                    masplus.Remove(p);
                    break;
                }
            }
            Buf.Dispose();
        }
    }
    ListTask.ListTasks.Remove(buffer);
    Serealize(ListTask);
}

private void DelProject_MouseClick(object sender, MouseEventArgs e)
{

```

```

        if (MessageBox.Show(
            "Вы точно хотите удалить проект",
            "Подтверждение",
            MessageBoxButtons.YesNo,
            MessageBoxIcon.Information,
            MessageBoxDefaultButton.Button1) == DialogResult.Yes)
        {
            File.Delete($"users\\{Login}\\project\\#{NameProject}.txt");
            MenuForm menu=new MenuForm(Login);
            this.Close();
            menu.Show();
        }
    }

    private void Serealize(MasTasks Listtas)
    {
        XmlSerializer formatter = new XmlSerializer(typeof(MasTasks));
        File.Delete($"users\\{Login}\\project\\#{NameProject}.txt");
        using (FileStream fs = new
        FileStream($"users\\{Login}\\project\\#{NameProject}.txt",
        FileMode.OpenOrCreate,FileAccess.Write))
        {
            formatter.Serialize(fs, Listtas);
        }
    }

    private MasTasks Desserialized(string NameProject)
    {
        MasTasks Listtast=new MasTasks();
        XmlSerializer formatter = new XmlSerializer(typeof(MasTasks));
        using (FileStream fs = new
        FileStream($"users\\{Login}\\project\\#{NameProject}.txt", FileMode.Open))
        {
            if (new FileInfo($"users\\{Login}\\project\\#{NameProject}.txt").Length
            != 0)
            {
                Listtast = (MasTasks)formatter.Deserialize(fs);
            }
        }
        return Listtast;
    }
}

```



```

private MasTasks Desserialized(string path,int i)
{
    MasTasks Listtast = new MasTasks();
    XmlSerializer formatter = new XmlSerializer(typeof(MasTasks));
    using (FileStream fs = new FileStream($"@{path}", FileMode.Open))
    {
        if (new FileInfo($"@{path}").Length != 0)
        {
            Listtast = (MasTasks)formatter.Deserialize(fs);
        }
    }
    return Listtast;
}

private void addTask_MouseClick(object sender, MouseEventArgs e)
{
    AddTaskForm AddTask = new AddTaskForm(NameProject,this.Login);
    AddTask.Show();
}

Point lastPoint;
private void panel_MouseMove(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left)
    {
        this.Left += e.X - lastPoint.X;
        this.Top += e.Y - lastPoint.Y;
    }
}

private void panel_MouseDown(object sender, MouseEventArgs e)
{
    lastPoint = new Point(e.X, e.Y);
}
}

```

Приложение А.4. Программный код класса «Tasks»

```

[Serializable] public class Tasks
{
    public string Name;
    public string worker;
    public int level;
}

```

```

    public string Description;
    public DateTime date;
    public DateTime dateSt;

    public Tasks() { }
    public Tasks(string name, string worker, int level, string description,
DateTime date, DateTime dateSt)
    {
        Name = name;
        this.worker = worker;
        this.level = level;
        Description = description;
        this.date = date;
        this.dateSt = dateSt;
    }
    public string Writer()
    {
        return $"{this.Name}; {this.worker};
{this.date.ToString("dd/MM/yyyy")}";
    }
}

[Serializable]
public class MasTasks
{
    public List<Tasks> ListTasks = new List<Tasks>();
}

```

Приложение А.5. Программный код окна добавления проекта

```

public partial class AddProjectForm : Form
{
    string login;
    int index=0;
    public AddProjectForm(string login)
    {
        InitializeComponent();
        this.login = login;
        this.FormBorderStyle = FormBorderStyle.None;

        AddWorker();
    }
    private void ButOut_MouseClick(object sender, MouseEventArgs e)
    {
        MenuForm menu = new MenuForm(login);
        menu.Show();
        this.Close();
    }
    private void AddWorker()
    {
        StreamReader sr = new StreamReader($"@users\{login}\worker.txt");
        string line;
        string[] mas;
        while ((line = sr.ReadLine()) != null)
        {
            if (string.IsNullOrEmpty(line)) {continue;}
            mas = line.Split('*');
            workercheck.Items.Add(mas[0]);
            index++;
        }
        sr.Close();
    }
    private void ButAdd_MouseClick(object sender, MouseEventArgs e)
    {
        if ((nameproject.Text == "")||(nameproject.Text == "Введите название"))

```

```

    {
        MessageBox.Show("Введите название проекта");
        return;
    }
    File.Create($"users\\{login}\\project\\#{nameproject.Text}.txt").Close();
    int[] mas = new int[index];
    int indexnew = 0;
    int j=0;
    string line;
    string[] masline;
    foreach(string t in workercheck.CheckedItems)
    {
        StreamReader sr = new StreamReader($"users\\{login}\\worker.txt");
        while ((line = sr.ReadLine()) != null)
        {
            masline = line.Split('*');
            if (t == masline[0])
            {
                mas[j++] = indexnew;
            }
            indexnew++;
        }
        sr.Close();
    }
    rewrite(mas);

    MenuForm menu = new MenuForm(login);
    menu.Show();
    this.Close();
}
private void rewrite(int[] mas)
{
    int i = 0;
    int j = 0;
    string path = $"users/{login}/worker.txt";
    string tempPath = path + ".txt";
    using (StreamReader sr = new StreamReader(path)) // читаем
    using (StreamWriter sw = new StreamWriter(tempPath)) // и сразу же пишем
    во временный файл
    {
        while (!sr.EndOfStream)
        {
            string line = sr.ReadLine();
            if (mas[j] == i)
            {
                sw.WriteLine(line+"*"+nameproject.Text);
                j++;
            }
            else { sw.WriteLine(line); }
            i++;
        }
    }
    File.Delete(path); // удаляем старый файл
    File.Move(tempPath, path); // переименовываем временный файл
}
private void nameproject_MouseClick(object sender, MouseEventArgs e)
{
    nameproject.Clear();
}
}

```

Приложение А.6. Программный код окна добавления задачи

```

public partial class AddTaskForm : Form
{

```

```

Panel head;
protected TextBox NameTask;
protected TextBox Description;
protected ComboBox Worker;
protected DateTimePicker date;
protected DateTimePicker dateSt;
protected RadioButton Easy;
protected RadioButton Medium;
protected RadioButton Hard;
protected Button ButAdd;
Button ButOut;
protected string Login;
protected string NameProject;
protected int level=0;
public AddTaskForm(string nameProject, string login)
{
    InitializeComponent();
    NameProject = nameProject;
    Login = login;
}

private void AddTaskForm_Load(object sender, EventArgs e)
{
    //head
    head= new Panel();
    head.Dock= DockStyle.Top;
    head.Size = new Size(1, 50);
    head.BackColor= Color.FromArgb(40, 122, 113);
    Controls.Add(head);

    //label
    Label hatter = new Label();
    hatter.Text = "Добавление задачи";
    hatter.Dock = DockStyle.Bottom;
    hatter.Location = new Point(1, 10);
    hatter.Size = new Size(410, 50);
    hatter.TextAlign = ContentAlignment.TopCenter;
    hatter.Font = new Font("Srbija Sans", 24);
    hatter.ForeColor = Color.White;
    head.Controls.Add(hatter);

    //name
    NameTask = new TextBox();
    NameTask.Text = "Введите название";
    NameTask.Location= new Point(50, 60);
    NameTask.Size = new Size(250,30);
    NameTask.MouseClick += Name_MouseClick;
    Controls.Add(NameTask);

    Label dat1= new Label();
    dat1.Text = "Введите дату начала";
    dat1.Location= new Point(50, 115);
    dat1.Size = new Size(250,20);
    dat1.Font = new Font("Srbija Sans", 10);
    Controls.Add(dat1);

    Label dat2 = new Label();
    dat2.Text = "Введите конечную дату";
    dat2.Location = new Point(50, 170);
    dat2.Size = new Size(250, 20);
    dat2.Font = new Font("Srbija Sans", 10);
    Controls.Add(dat2);

    // календарь
    dateSt = new DateTimePicker();

```

```

dateSt.Location = new Point(50,140);
dateSt.Size = new Size(250, 30);
Controls.Add(dateSt);

date = new DateTimePicker();
date.Location = new Point(50, 195);
date.Size = new Size(250, 30);
Controls.Add(date);

//Сотрудник
Worker = new ComboBox();
Worker.Text = "Выбор сотрудника";
Worker.Location = new Point(50, 90);
Worker.Size = new Size(250, 30);
WorkerAdd(Worker);
Controls.Add(Worker);

//сложность
Easy=new RadioButton();
Easy.Location = new Point(100, 225);
Easy.Size = new Size(200, 30);
Easy.Text = "Легкая";
Easy.Font = new Font("Srbija Sans", 15);
Easy.MouseClick += Easy_CheckedChanged;
Controls.Add(Easy);

Medium = new RadioButton();
Medium.Location = new Point(100, 260);
Medium.Size = new Size(200, 30);
Medium.Text = "Средняя";
Medium.Font = new Font("Srbija Sans", 15);
Medium.MouseClick += Medium_CheckedChanged;
Controls.Add(Medium);

Hard = new RadioButton();
Hard.Location = new Point(100, 295);
Hard.Size = new Size(200, 30);
Hard.Text = "Сложная";
Hard.Font = new Font("Srbija Sans", 16);
Hard.MouseClick += Hard_CheckedChanged;
Controls.Add(Hard);

//описание
Description= new TextBox();
Description.Text = "Описание";
Description.Multiline= true;
Description.Location = new Point(50,330);
Description.Size= new Size(250, 120);
Controls.Add(Description);

//кнопка назад
ButOut = new Button();
ButOut.Text = "Назад";
ButOut.Location = new Point(40, ClientSize.Height - 60);
ButOut.Size = new Size(100,30);
ButOut.Font = new Font("Srbija Sans", 12);
Controls.Add(ButOut);
ButOut.MouseClick += ButOut_MouseClick;

//кнопка добавить
ButAdd = new Button();
ButAdd.Text = "Добавить";
ButAdd.Location = new Point(200, ClientSize.Height - 60);
ButAdd.Size = new Size(100, 30);
ButAdd.Font = new Font("Srbija Sans", 12);

```

```

        Controls.Add(ButAdd);
        ButAdd.MouseClick += ButAdd_MouseClick;
    }
    private void WorkerAdd(ComboBox Worker)
    {
        StreamReader sr = new StreamReader($"users\\{Login}\\worker.txt");
        string line;
        string[] mas;
        while ((line = sr.ReadLine()) != null)
        {
            mas = line.Split('*');
            for(int i = 1; i < mas.Length; i++)
            {
                if (mas[i] == NameProject)
                {
                    Worker.Items.Add(mas[0]);
                }
            }
        }
        sr.Close();
    }
    private void Easy_CheckedChanged(object sender, MouseEventArgs e)
    {
        this.level = 1;
    }
    private void Medium_CheckedChanged(object sender, MouseEventArgs e)
    {
        this.level = 2;
    }
    private void Hard_CheckedChanged(object sender, MouseEventArgs e)
    {
        this.level = 3;
    }
    protected virtual void ButOut_MouseClick(object sender, MouseEventArgs e)
    {
        this.Close();
    }
    private void Name_MouseClick(object sender, MouseEventArgs e)
    {
        NameTask.Clear();
    }

    private void ButAdd_MouseClick(object sender, MouseEventArgs e)
    {
        if (NameTask.Text=="")
        {
            MessageBox.Show("Введите название задачи");
            return;
        }
        if (date.Value >= dateSt.Value)
        {
            Tasks Newtask = new
Tasks(NameTask.Text, Worker.Text, level, Description.Text, date.Value, dateSt.Value);
            MasTasks Listtas = new MasTasks();
            Listtas= Desserialized(Listtas);
            Listtas.ListTasks.Add(Newtask);
            Serealize(Listtas);
            MessageBox.Show("Задача добавлена");
        }
        else
        {
            MessageBox.Show("Начальная дата не может быть позже конечной",
"Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

```

```

    }
    private void Serealize(MasTasks Listtas)
    {
        MasTasks newList = Listtas;
        XmlSerializer formatter = new XmlSerializer(typeof(MasTasks));
        using (FileStream fs = new
        FileStream($"@users\{Login}\project\#{NameProject}.txt", FileMode.Open))
        {
            formatter.Serialize(fs, newList);
        }
    }
    private MasTasks Desserialized(MasTasks Listtast)
    {
        XmlSerializer formatter = new XmlSerializer(typeof(MasTasks));
        using (FileStream fs = new
        FileStream($"@users\{Login}\project\#{NameProject}.txt", FileMode.Open))
        {
            if (new FileInfo($"@users\{Login}\project\#{NameProject}.txt").Length
            != 0)
            {
                Listtast = (MasTasks)formatter.Deserialize(fs);
            }
        }
        return Listtast;
    }
}

```

Приложение А.7. Программный код окна изменения задачи

```

public partial class WatchTask : AddTaskForm
{
    Tasks task;
    public WatchTask(string nameProject, string login, Tasks task) :
    base(nameProject, login)
    {
        InitializeComponent();
        this.task = task;
    }

    private void WatchTask_Load(object sender, EventArgs e)
    {
        ButAdd.Dispose();
        MasTasks mt = Desserialized();

        foreach(Tasks t in mt.ListTasks)
        {
            if ((t.Name==task.Name)&&(t.worker==task.worker) && (t.level ==
            task.level) && (t.date == task.date) && (t.dateSt == task.dateSt) &&
            (t.Description == task.Description))
            {
                Console.WriteLine("dsdsd");
                NameTask.Text = t.Name;
                Description.Text = t.Description;
                Worker.Text = t.worker;
                date.Value=t.date;
                dateSt.Value=t.dateSt;
                switch (t.level)
                {
                    case 1:Easy.Select();break;
                    case 2:Medium.Select(); break;
                    case 3: Hard.Select(); break;
                    default: break;
                }
                break;
            }
        }
    }
}

```

```

    }
}

protected override void ButOut_MouseClick(object sender, MouseEventArgs e)
{
    this.Close();
}

private MasTasks Desserialized()
{
    MasTasks Listtast = new MasTasks();
    XmlSerializer formatter = new XmlSerializer(typeof(MasTasks));
    using (FileStream fs = new
FileStream($"@users\{Login}\project\#{NameProject}.txt", FileMode.Open))
    {
        if (new FileInfo($"@users\{Login}\project\#{NameProject}.txt").Length
!= 0)
        {
            Listtast = (MasTasks)formatter.Deserialize(fs);
        }
    }
    return Listtast;
}
}

```

Приложение А.8. Программный код окна добавления сотрудника

```

public partial class AddWorkerForm : Form
{
    public string Login;
    public AddWorkerForm()
    {
        this.FormBorderStyle = FormBorderStyle.None;
        InitializeComponent();
    }
    public AddWorkerForm(string login)
    {
        InitializeComponent();
        Login= login;
        addProject(login);
    }
    private void addProject(string Login)
    {
        foreach (string file in Directory.EnumerateFiles($"users/{Login}/project",
"*.txt", SearchOption.AllDirectories))
        {
            string name = file.Split('#')[1];
            ChooseProject.Items.Add($"{name.Substring(0, name.Length - 4)}");
        }
    }

    private void Name_Click(object sender, EventArgs e)
    {
        NameText.Clear();
    }

    virtual internal void OutBut_MouseClick(object sender, MouseEventArgs e)
    {
        this.Close();
    }

    virtual internal void AddBut_MouseClick(object sender, MouseEventArgs e)
    {
        if ((NameText.Text == "") || (NameText.Text == "Введите ФИО сотрудника"))
        {

```



```

        MessageBox.Show("Введите ФИО сотрудника");
        return;
    }
    string work = NameText.Text;
    foreach(string it in ChooseProject.CheckedItems)
    {
        work+= $"{it}";
    }
    using (StreamWriter sw = new StreamWriter($"users/{Login}/worker.txt",
true, Encoding.UTF8))
    {
        sw.WriteLine(work);
    }
    MessageBox.Show("успешно");
}
}

```

Приложение А.9. Программный код окна изменения сотрудника

```

public partial class DelWorkerForm : AddWorkerForm
{
    string NameWor;
    public DelWorkerForm(string name, string login) : base(login)
    {
        InitializeComponent();
        NameWor = name;
        DataEntry();
        AddBut.Text = "Изменить";
        OutBut.Text = "Удалить";
        this.StartPosition = FormStartPosition.CenterScreen;
    }
    private void DataEntry()
    {
        NameText.Text = NameWor;
        StreamReader sr = new StreamReader($"users/{Login}/worker.txt");
        string line;
        string[] masproject;
        while ((line = sr.ReadLine()) != null)
        {
            masproject = line.Split('*');
            if (masproject[0] == NameWor)
            {
                for (int i = 0; i < ChooseProject.Items.Count; i++)
                {
                    for (int j = 1; j < masproject.Length; j++)
                    {
                        string buf = ChooseProject.Items[i].ToString();
                        if (buf == masproject[j])
                        {
                            ChooseProject.SetItemChecked(i, true);
                        }
                    }
                }
                break;
            }
        }
        sr.Close();
    }

    override internal void AddBut_MouseClick(object sender, MouseEventArgs e)
    {
        StreamReader sr = new StreamReader($"users/{Login}/worker.txt");
        string line;
    }
}

```

```

string[] masproject;
int index=0;
while ((line = sr.ReadLine()) != null)
{
    masproject = line.Split('*');
    if (masproject[0] == NameWor)
    {
        break;
    }
    index++;
}
sr.Close();
string work = NameText.Text;
foreach (string it in ChooseProject.CheckedItems)
{
    work += $"{it}";
}
RewriteLine(index,work);
MenuForm menu = new MenuForm(Login);
menu.Show();
MessageBox.Show("Сотрудник изменён");
this.Close();
}
private void RewriteLine(int lineIndex, string newValue)
{
    int i = 0;
    string path = $"users/{Login}/worker.txt";
    string tempPath = path + ".txt";
    using (StreamReader sr = new StreamReader(path)) // читаем
    using (StreamWriter sw = new StreamWriter(tempPath, true, Encoding.UTF8))
    // и сразу же пишем во временный файл
    {
        while (!sr.EndOfStream)
        {
            string line = sr.ReadLine();
            if (lineIndex == i)
            {
                sw.WriteLine(newValue);
            }
            else
            {
                sw.WriteLine(line);
            }
            i++;
        }
    }
    File.Delete(path); // удаляем старый файл
    File.Move(tempPath, path); // переименовываем временный файл
}
override internal void OutBut_MouseClick(object sender, MouseEventArgs e)
{
    StreamReader sr = new StreamReader($"users/{Login}/worker.txt");
    string line;
    string[] masproject;
    int index = 0;
    while ((line = sr.ReadLine()) != null)
    {
        masproject = line.Split('*');
        if (masproject[0] == NameWor)
        {
            break;
        }
        index++;
    }
    sr.Close();
}

```

```

string test = null;
RewriteLine(index, test);
MenuForm menu= new MenuForm(Login);
menu.Show();
MessageBox.Show("Сотрудник удалён");
this.Close();
    }
}

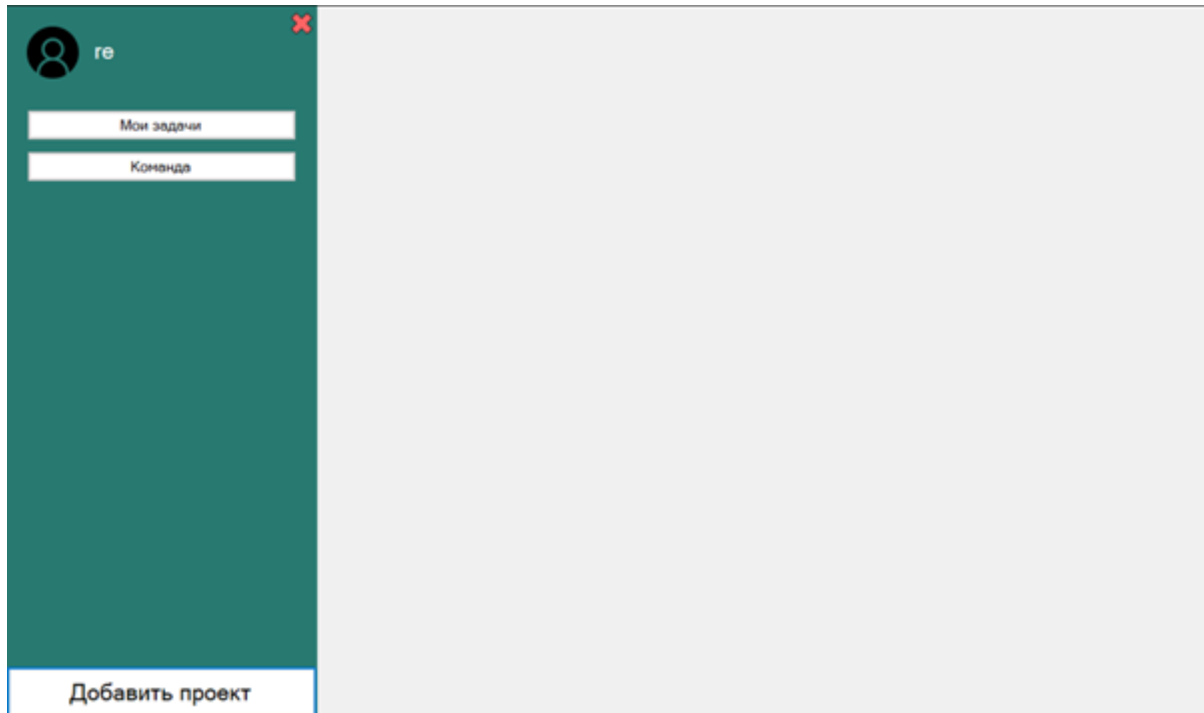
```

Приложение Б. Интерфейс приложения

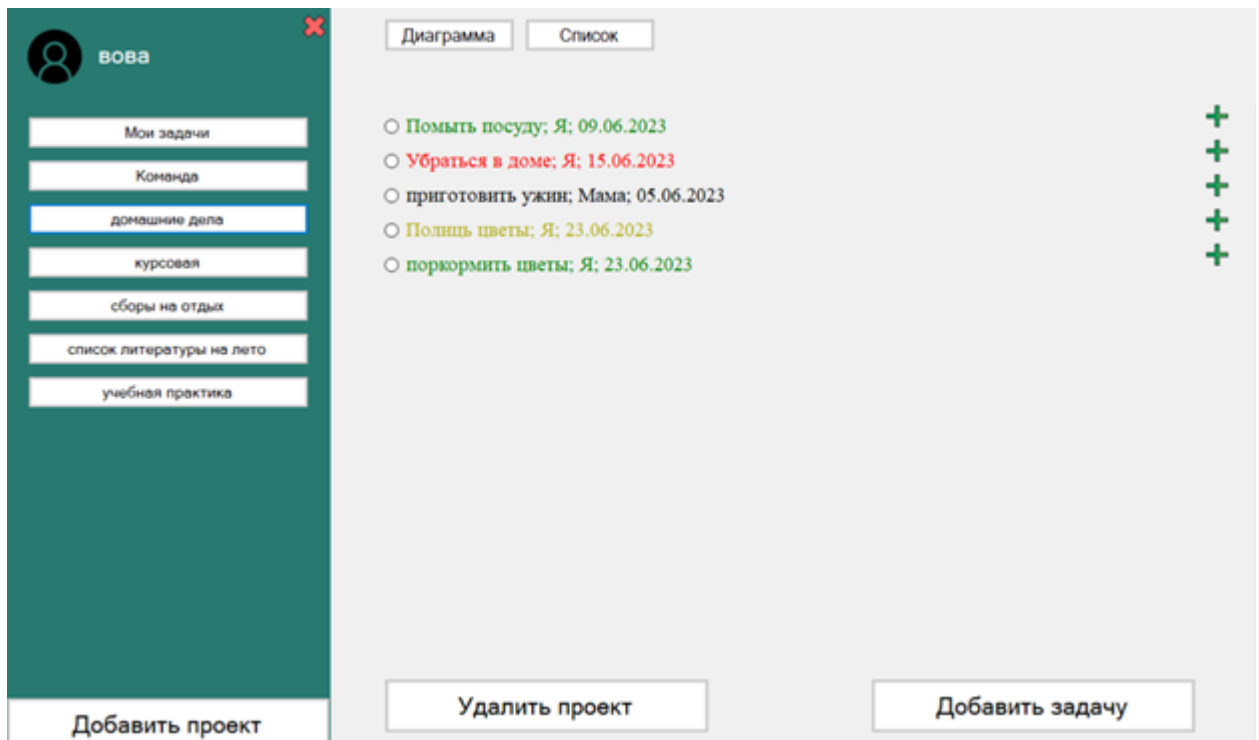
Приложение Б.1. Окно входа в программу

Приложение Б.2. Окно регистрации

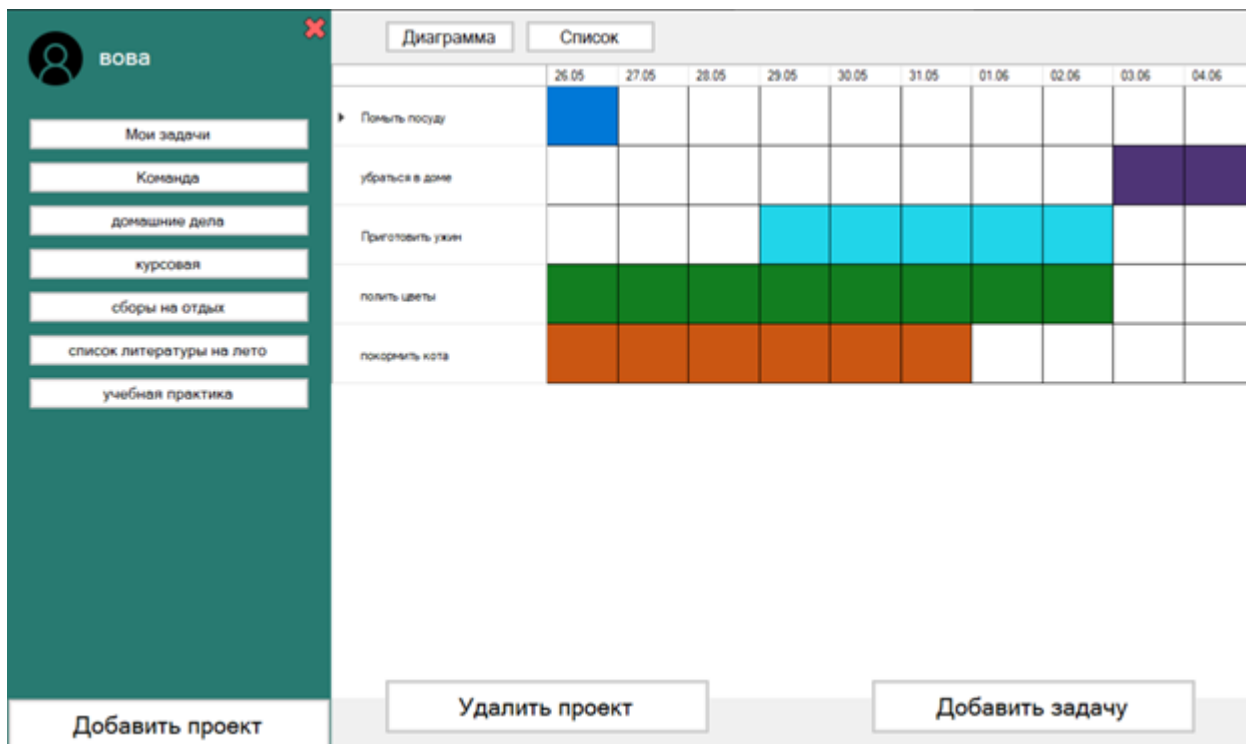
Приложение Б.3. Окно меню при первом запуске



Приложение Б.4. Окно меню пример с задачами



Приложение Б.5. Диаграмма Ганта



Приложение Б.6. Окно добавления задачи

The screenshot shows a 'Добавление задачи' (Add Task) window. It contains the following fields and controls:

- Text input: 'Введите название' (Enter name)
- Dropdown menu: 'Выбор сотрудника' (Employee selection)
- Text input: 'Введите дату начала' (Enter start date), with a date picker showing '26 мая 2023 г.'
- Text input: 'Введите конечную дату' (Enter end date), with a date picker showing '26 мая 2023 г.'
- Radio buttons for task difficulty: 'Легкая' (Easy), 'Средняя' (Medium - selected), 'Сложная' (Difficult)
- Text area: 'Описание' (Description)
- Buttons: 'Назад' (Back) and 'Добавить' (Add)

Приложение Б.7. Окно изменения задачи

Добавление задачи

Я

▼

Введите дату начала

15 июня 2023 г.

▼

Введите конечную дату

23 июня 2023 г.

▼

☐ Легкая

☒ Средняя

☐ Сложная

Полиция Герань на подоконнике, Полиция Алоэ в коридоре

Назад

Приложение Б.8. Окно добавления сотрудника

Добавление сотрудника

☐ домашние дела

☐ курсовая

☐ сборы на отдых

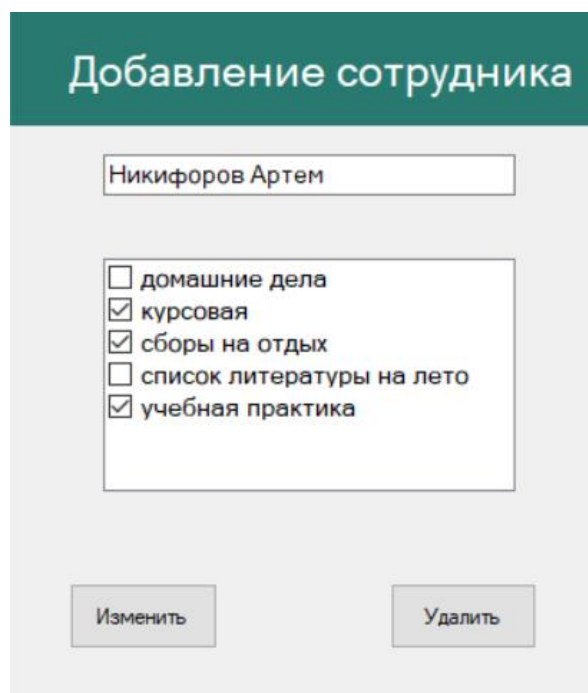
☐ список литературы на лето

☐ учебная практика

Добавить

Назад

Приложение Б.9. Окно изменения сотрудника



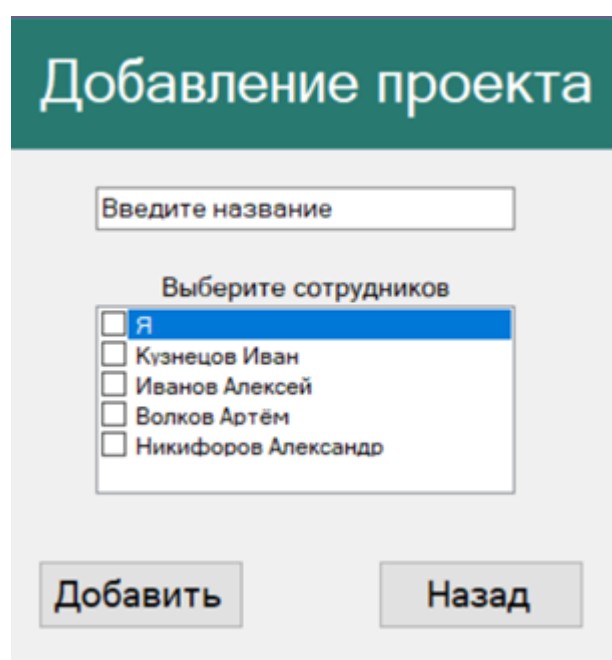
Добавление сотрудника

Никифоров Артем

- ☐ домашние дела
- ☒ курсовая
- ☒ сборы на отдых
- ☐ список литературы на лето
- ☒ учебная практика

Изменить Удалить

Приложение Б.10. Окно добавления проекта



Добавление проекта

Введите название

Выберите сотрудников

- ☒ Я
- ☐ Кузнецов Иван
- ☐ Иванов Алексей
- ☐ Волков Артём
- ☐ Никифоров Александр

Добавить Назад