МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 05 01 Информационные системы и технологии

Направление специальности 1-40 05 01 03 «Информационные системы и технологии (издательско-полиграфический комплекс)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Менеджер задач»

Исполнитель

Студентка 3 курса группы 1 Шишова Т.С.

(Ф. И. О.)

Руководитель работы ассистент Панченко О. Л.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

# **Введение**

В наше быстро меняющееся и ритмичное время, важность эффективного управления задачами и временем становится все более актуальной. Быть продуктивным означает не просто делать много дел, но и делать их вовремя и с учетом их приоритетности. Для этого требуется системный подход, который помогает организовать задачи, установить приоритеты и распределить время максимально эффективно.

Современные технологии и инструменты предоставляют нам возможность значительно улучшить управление нашими задачами и временем. Электронные менеджеры задач стали незаменимыми помощниками в повседневной жизни, облегчая нам процесс организации и планирования. По этим причинам была выбрана тема курсового проекта «Менеджер задач».

Для успешной реализации курсового проекта необходимо:

* провести анализ и обзор аналогичных решений;
* определить функциональные требования;
* продумать архитектуру проекта;
* разработать базу данных;
* реализовать программное средство;
* протестировать программное средство на наличие ошибок;
* написать руководство пользователя.

# **Анализ прототипов и формирование требований к проектируемому программному средству.**

Постановка задачи и обзор аналогов являются важными элементами любой работы. Это позволяет: четко определить цели, избежать дублирования проектов и их ошибок, а также улучшить качество решения и проекта.

## **1.1 Анализ прототипов**

**Trello** — очень известная и популярная в среде иностранных так и русских компаний. Данная программа имеет яркий и настраиваемый интерфейс и включает в себя весь необходимый функционал.

Плюсы:

* понятный, красивый и настраиваемый под себя интерфейс;
* есть возможность устанавливать дату и время дедлайна;
* поддержка и синхронизация с мобильной версией приложения;
* возможность назначать пользователям задачи;
* есть календарь и канбан доска;
* возможность прикреплять к задачам документы и файлы.

Минусы:

* нету возможности отлеживать время, затраченное на задачу;
* нету статистики;
* некоторый функционал присутствует только в платной версии.

Ниже на рисунке 1.1 представлен пример использования Trello для отображения задач фургончика с тако.

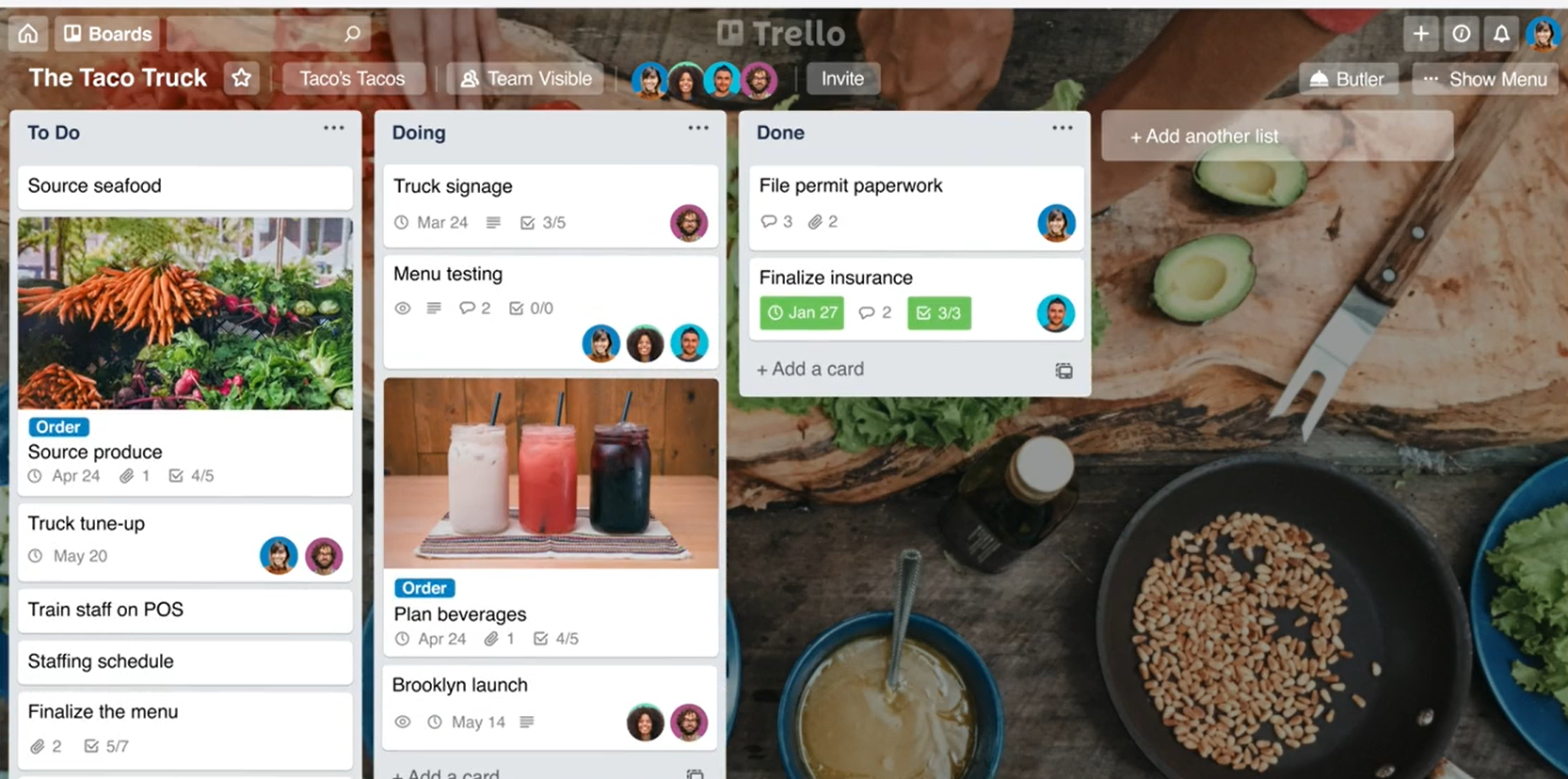


Рисунок 1.1 — Интерфейс приложения «Trello»

**WEEEK** – одна из популярных и качественно сделанных программ, созданная российскими разработчиками. Имеет огромное количество бесплатного функционала. Часто используется в небольших компаниях для отслеживания выполнения задач.

Плюсы:

* возможность устанавливать сроки для задач и следить за их дедлайнами;
* наличие календаря и канбан-доски;
* наличие таймеров для фокуса;
* присутствует таймер для медитации;
* наличие большого количества тем и оформления;
* используется голосовой ввод для создания задач;
* возможность вести коллективную работу над разными проектами;
* распределять задачи по срокам и важности;
* декомпозировать задачи на подзадачи;
* настраивать фильтры для контроля сроков;
* разрабатывать и внедрять собственную базу знаний;
* устанавливать повторяемость для регулярных задач.

Минусы:

* в бесплатной версии можно создавать до 7 задач и 10 папок;
* в бесплатной версии описание в тексте не должно превышать 100 предложений;
* приложение является дорогим.

Данное приложение является одним из популярных для работы в командах. Интерфейс данного приложение представлен на рисунке 1.2.

|  |
| --- |
|  |

Рисунок 1.2 — Интерфейс приложения «WEEEK»

**TickTick** позволяет создавать множество списков задач. Это базовая функция, которая доступна во всех подобных приложениях. TickTick синхронизирует задачи с веб-версией, приложениями для Android, iOS, расширениями для Chrome и Firefox.

Плюсы:

* в бесплатной версии есть весь необходимый функционал (оформление задач или списков/ приоритетов/дедлайнов);
* осуществление интеграции с календарем;
* возможность ставить флажки с приоритетами;
* присутствуют различные параметры сортировки (по порядку, дате, имени, приоритету);
* добавление заметок или комментариев к задачам;
* быстрый поиск задач;
* есть возможность просматривать статистику и выполненные задачи;
* для мобильной версии присутствуют виджеты;
* есть трекер привычек с большим количеством оформления.

Минусы:

* + отсутствие вложенности проектов;
  + календарь работает только в премиум версии;
  + возможность отслеживать изменения задач присутствует только в премиум версии;
  + нету кан-бан доски.

Изображение интерфейса TickTick представлено на рисунке 1.3.

|  |
| --- |
|  |

Рисунок 1.3 — Интерфейс приложения «TickTick»

## **1.2 Функциональные требования**

На основе анализа аналогичных решений был составлен список функций, которые должна выполнять разрабатываемая программа:

Функции администратора сервиса:

* поддерживать работу c базой данных;
* пользователю будут приходить уведомления о приближающихся сроках к выполнению задач.

Функции клиента:

* регистрация и авторизация пользователя;
* редактирование профиля пользователя;
* добавление задач указывая необходимую информацию и привязывание их к дате и времени;
* удаление задач
* распределение задач по приоритету;
* группировка задач;
* добавление к задачам подзадач;
* поиск задач;
* составление отчёта о выполненных задачах в виде графика.

## **1.3 Вывод**

После оценки всех приложений можно заметить, что многие программы имеют схожий интерфейс и концепцию, но каждое приложение реализовало его по-своему. Проанализировав аналоги приложений с тематикой «Менеджер задач» были выделены основные задачи проекта: разработка приложения, позволяющего пользователю осуществлять весь базовый функционал менеджеров задач (добавление и удаление задач, редактирование, приоритезация и т.д.). К задачам должна быть возможность ставить крайний срок выполнения. А подзадачи должны давать возможность пользователю добавлять файлы и фотографии. Приложение также будет показывать статистику выполнения задач за определенный промежуток времени. Интерфейс должен быть удобным, понятным и легко настраиваемым для пользователя.

# **2. Проектирование архитектуры проекта**

Проектирование архитектуры является неотъемлемой частью разработки любого проекта. В данном разделе будут описаны средства, с помощью которых будет осуществляться разработка, определена структура БД, определены окна, из которых будет состоять приложение и разработана диаграмма взаимодействия пользователей – User Case.

## **2.1. Проектирование схемы БД**

Для эффективного хранения и управления данными приложения была разработана логическая схема базы данных. Она определяет структуру данных, их связи и основные таблицы. Ниже представлено краткое описание таблиц базы данных:

Таблица «USER» — хранит информацию о пользователях. Состоит из следующих столбцов:

* ID\_User: уникальный идентификатор пользователя.
* Login: логин пользователя.
* Password: пароль пользователя.
* User\_Name: имя пользователя.
* User\_Surname: фамилия пользователя.
* Email: электронная почта пользователя (необязательное поле).
* Phone: номер телефона пользователя (необязательное поле).
* Contact: контактная информация пользователя (необязательное поле).
* Description: описание о себе для пользователя (необязательное поле).

Таблица «TASK» — отвечает за хранение информации о задачах. Состоит из следующих столбцов:

* ID\_Task: Уникальный идентификатор задачи.
* Task\_UserID: Идентификатор пользователя, связанного с задачей.
* Task\_Name: Текст задачи.
* Time: Время создания задачи.
* Priority: Приоритет задачи.
* IsWork: Флаг, указывающий, является ли задача активной (булево значение).
* isCheck: Флаг, указывающий, была ли задача отмечена как выполненная (булево значение).
* CheckTime: Время, когда задача была отмечена как выполненная.

Таблица «SUBTASKS» — содержит информацию о подзадачах, связанных с задачами. Состоит из следующих столбцов:

* ID\_Subtask: Уникальный идентификатор подзадачи.
* TaskID: Идентификатор задачи, к которой принадлежит данная подзадача.
* Subtask\_Name: Текст подзадачи.
* isCheck: Флаг, указывающий, была ли подзадача отмечена как выполненная (булево значение).

Таблица «TASK\_ATTACHMENTS» — используется для хранения вложений, таких как изображения и файлы, связанных с пользователями, задачами и подзадачами.

* ID\_Task\_Attachments: Уникальный идентификатор вложения задачи.
* TA\_UserID: Идентификатор пользователя, связанного с вложением.
* TA\_TaskID: Идентификатор задачи, связанной с вложением.
* TA\_SubtaskID: Идентификатор подзадачи, связанной с вложением.
* ImagePath: Путь к изображению (необязательное поле).
* Image: Бинарные данные изображения (необязательное поле).
* FilePath: Путь к файлу (необязательное поле).
* File: Бинарные данные файла (необязательное поле).

На основе этих таблиц и их столбцов была разработана логическая схема БД (Рисунок 2.1).

|  |
| --- |
|  |

Рисунок 2.1 — Логическая схема БД

## **2.2 Описание средств разработки**

Для разработки приложения «Менеджер задач» используются:

* среда разработки Microsoft Visual Studio 2022;
* программная платформа .NET;
* язык программирования C#;
* технология WPF;
* SQL Server Manager 2014;

Для разработки приложения был выбран интерфейс прикладного программирования (API) Windows Presentation Foundation (WPF). WPF — это современная система для создания клиентских приложений под операционную систему Windows, которая обладает продвинутыми возможностями взаимодействия с пользователем и интегрирована в платформу .NET. Основой технологии WPF является инфраструктура, построенная на основе DirectX, что делает ее более мощной и гибкой по сравнению с более ранними технологиями, такими как Windows Forms.

Так как разработка будет выполняться на ЯП C#, то для взаимодействия с базой данных требуется промежуточное программное средство. В данном случае в качестве такого средства была выбрана технология Entity Framework. Эта технология представляет собой объектно-ориентированный подход к доступу к данным и представляет собой решение для отображения объектов на реляционные базы данных (ORM) на платформе .NET Framework. Entity Framework позволяет выполнять запросы к данным с использованием LINQ to Entities, что упрощает и ускоряет процесс взаимодействия с базой данных.

Таким образом, выбор использования WPF и Entity Framework позволяет создать мощное и гибкое приложение с современным пользовательским интерфейсом и удобным доступом к данным в БД.

**2.3 Проектирование окон приложения**

В программном средстве «Менеджер задач» при запуске будет отображаться окно входа. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Если пользователь не зарегистрирован, он переходит в другое окно, в котором может ввести необходимые данные для регистрации: новый логин, ФИО, email и пароль. Введенные данные, успешно прошедшие валидацию, будут заносятся в базу данных, а успешно зарегистрированный пользователь сможет зайти в приложение.

На странице «Главная» в самом приложении будет отображаться список задач и подзадач пользователей, а также необходимые инструменты для создания, редактирования, удаления и поиска задач. Задача будет добавляться через текстовое поле для и после нажатия кнопки «Добавить» в соответствующей вкладке окна задача будет создаваться и отображаться. После добавления задачи пользователь может: редактировать ее содержимое, добавить дедлайн, выбрав дату в появившемся календаре, установить приоритет, добавить подзадачу, отметить выполненной задачу или удалить ее. Приоритезация задачи будет влиять на расположение ее в списке задач. Выполненные задачи будут отображаться неделю после их выполнения, затем они не выводятся, но остаются в базе данных.

При добавлении подзадач можно будет: вводить название и текст подзадачи, добавлять картинки и текстовые файлы, отмечать выполненной и удалять подзадачи. Если задача отмечается как выполненная, то все ее подзадачи также отмечаются как выполненными.

На странице «Отчет» будет отображаться статистка о выполненных задачах в виде графика.

В окне «Профиль» можно посмотреть данные пользователя, изменить их, загрузить или обновить фото профиля, а также заполнить более подробную информацию «о себе». Все данные о пользователе будут автоматически обновляться в базе данных. В окне «Профиль» пользователь сможет сменить язык приложения и тему.

В приложении будет присутствовать кнопка для выхода из приложения, которая будет возвращать пользователя к окну авторизации.

**2.4 Проектирование User Case диаграммы**

На основе разработанной логической схемы БД, описанного основного функционала программы и описания будущих окон приложения, был определены основные возможности пользователя. Для наглядного отображения вохможностей пользователя была создана User Case диаграмма (Рисунок 2.2).

|  |
| --- |
|  |

Рисунок 2.2 — User Case диаграмма

**2.5 Вывод**

В данном разделе мы подробно рассмотрели процесс проектирования архитектуры нашего проекта, начиная с выбора интерфейса прикладного программирования (API) и заканчивая разработкой базы данных и определения окон интерфейса приложения. Этот этап разработки является фундаментальным и определяет, как приложение будет взаимодействовать с пользователями и как оно будет управлять данными. Было выбрано Windows Presentation Foundation (WPF) в качестве API для создания клиентской части приложения. Также, для взаимодействия с базой данных будет использоваться технология Entity Framework.

Была разработана логическая схема базы данных, определена структура таблиц и связей между ними. Это обеспечивает эффективное хранение и управление информацией, необходимой для функционирования приложения. Также мы описали функционал будущих окон и страниц приложения, уделяя внимание удобству использования и соответствию потребностям пользователей. Use case диаграмма позволила нам лучше понять, как пользователи будут взаимодействовать с приложением, какие функции им будут доступны и какие задачи они смогут выполнять.