МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет-приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема Программное средство «Планировщик задач»

Исполнитель

студентка 2 курса группы 5 Городилина Анастасия Сергеевна

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

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

Курсовой проект защищен с оценкой

Председатель Панченко О.Л.

(подпись)

Минск 2024

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc166802904)

[1. Аналитический обзор литературы и постановка задачи 4](#_Toc166802905)

[1.1 Анализ прототипов 4](#_Toc166802906)

[1.2 Постановка задачи 7](#_Toc166802907)

[2. Анализ требований к программному средству и разработка функциональных требований 8](#_Toc166802908)

[2.1 Описание средств разработки 8](#_Toc166802909)

[2.2 Описание разрабатываемой функциональности приложения 9](#_Toc166802910)

[2.3 Спецификация функциональных требований 10](#_Toc166802911)

[3. Проектирование программного средства 11](#_Toc166802912)

[3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства 11](#_Toc166802913)

[3.2. Архитектура системы 12](#_Toc166802914)

[3.3 Схема работы приложения 14](#_Toc166802915)

[3.4 Проектирование логики сценариев использования 14](#_Toc166802916)

[3.5. Проектирование логической структуры базы данных 15](#_Toc166802917)

[4. Создание (реализация) программного средства 17](#_Toc166802918)

[4.1 Выполняемые функции 17](#_Toc166802919)

[4.2 Реализация общей структуры проекта 17](#_Toc166802920)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 21](#_Toc166802921)

[6. Методика установки и использования программного средства 25](#_Toc166802922)

[6.1 Установка программного средства 25](#_Toc166802923)

[6.2 Использование программного средства 27](#_Toc166802924)

[ЗАКЛЮЧЕНИЕ 32](#_Toc166802925)

[СПИСОК ЛИТЕРАТУРЫ 33](#_Toc166802926)

[Приложение А 34](#_Toc166802927)

[Графический материал 36](#_Toc166802928)

[Диаграмма вариантов использования 36](#_Toc166802929)

[Диаграммы классов 37](#_Toc166802930)

[Диаграмма последовательности 38](#_Toc166802931)

# ВВЕДЕНИЕ

Тема разработки курсового проекта "Планировщик задач" представляет значительную актуальность в современном мире, где эффективное управление задачами играет ключевую роль в достижении успеха и выполнении целей. Существующие методы управления задачами имеют свои преимущества, однако с увеличением сложности проектов возникает потребность в более эффективных решениях. Главной целью данного проекта является разработка и реализация программного решения для эффективного планирования и управления задачами.

Программное решение будет ориентировано на универсальность и гибкость, чтобы быть применимым в различных областях и сферах деятельности. Основными принципами разработки являются удобство использования и интуитивно понятный интерфейс для быстрой адаптации пользователей. Оно также предоставит возможность пользователям записывать свои задачи, устанавливать временные рамки и организовывать их по категориям для удобного управления.

Дополнительно, программное решение будет обеспечивать возможность генерации аналитики для оценки эффективности работы. Это поможет пользователям отслеживать прогресс выполнения задач и планировать свою деятельность более эффективно.

В итоге, создание такого планировщика задач позволит пользователям эффективно управлять своим временем и задачами, повышая их продуктивность и достигая поставленных целей.

1. Аналитический обзор литературы и постановка задачи

Для того чтобы окончательно определиться с постановкой задачи моего курсового проекта, необходимо проанализировать прототипы программных средств выбранной темы.

* 1. Анализ прототипов

Сейчас в нашем распоряжении имеются онлайн-сервисы и приложения, которые помогают человеку ответственно планировать свой день, экономить свое время и ресурсы. Мы получаем новые возможности: все наши мысли можно хранить в одном приложении, забыв навсегда о бумажном носителе. Рассмотрим наиболее популярные и удобные планеры.

*ToDoist* – веб-сервис и набор программного обеспечения для управления задачами [1]. Задачи могут также содержать заметки с файлами любого типа. Задачи можно помещать в проекты, сортировать по фильтрам, присваивать им метки, редактировать и экспортировать. Интерфейс приложения показан на рисунке 1.1.1.

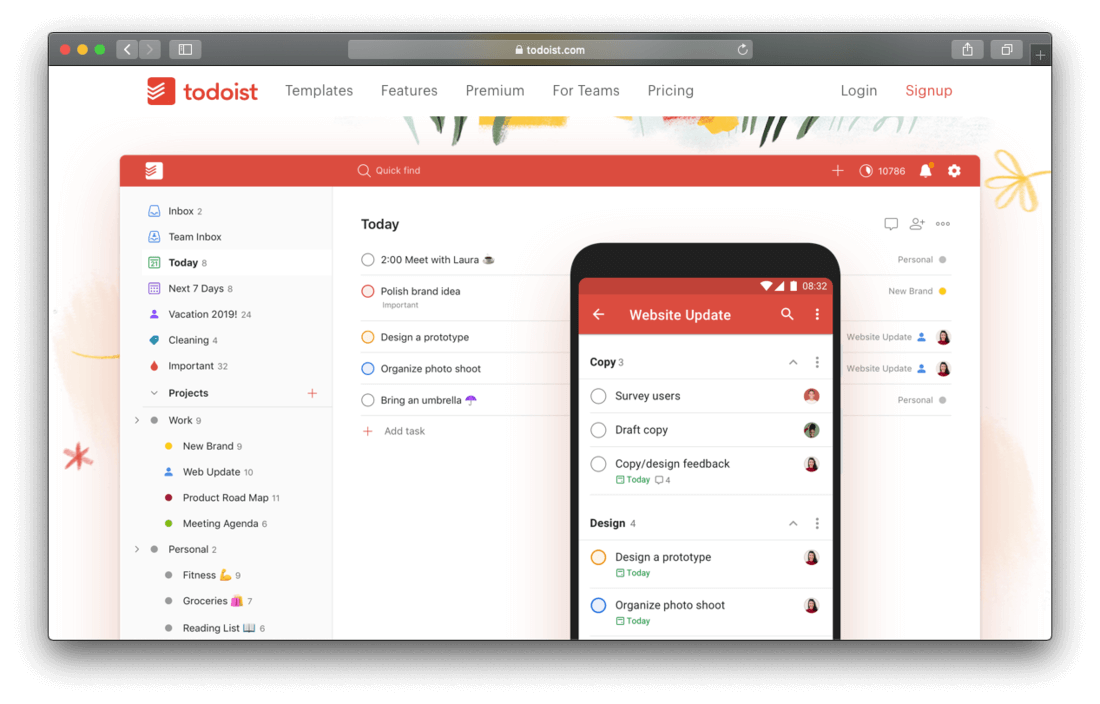


Рисунок 1.1.1 – Интерфейс приложения “ToDoist”

Для каждой из задач виден её проект и приоритет. Отсюда же можно быстро просмотреть комментарии и вложенные файлы. Слева от списка задач можно переключиться на входящие дела или на дела только за сегодня. Также можно быстро перейти в нужный проект или посмотреть задачи по меткам и пользовательским фильтрам.

Сервис отлично работает с горячими клавишами и поэтому управлять задачами можно без отрыва от клавиатуры. Любую задачу можно быстро добавить во входящие или сразу же назначить ей проект, приоритет, теги и напоминание. Также к любой задаче можно добавить неограниченное количество комментариев и файловых вложений.

Фильтрация задач при помощи ключевых слов и меток не только помогает отобразить нужные сейчас дела, но и быстро находить задачи в сложных проектах.

ToDoist поддерживает десятки различных интеграций со сторонними сервисами. Благодаря которым пользователь сможет ещё более продуктивно планировать свой день и выполнять поставленные задачи в срок.

К недостаткам можно отнести то, что синхронизация и многие важные функции доступны только премиум-пользователям, продвинутые «фишки» не очевидны и про них нужно знать, нет гибкой настройки отображения задач.

*Trello* – программа для управления проектами небольших групп, разработанная Fog Creek Software [2]. Веб-приложение напоминает вывеску. Связано это с тем, что с японского языка слово «trello» переводится как «рекламный щит». Трелло позволяет работать над проектами как индивидуально, так и в команде. Также инструмент даёт возможность подключить к работе над задачами и проектом сторонних исполнителей. Trello доступен на компьютерах в виде десктопной версии, в браузерной версии и версии для мобильных устройствах.

Проект в Трелло — отдельная доска, на которой в разных колонках размещаются карточки. Столбцы могут отображать статус задач, находящихся в них, как на канбан-доске или быть определённой сущностью. Задачи в Trello представляют из себя карточки. Карточки могут как содержать тело задачи (описание, требования и т. д.), так и содержать информацию о каком-нибудь объекте. Интерфейс приложения показан на рисунке 1.1.2.

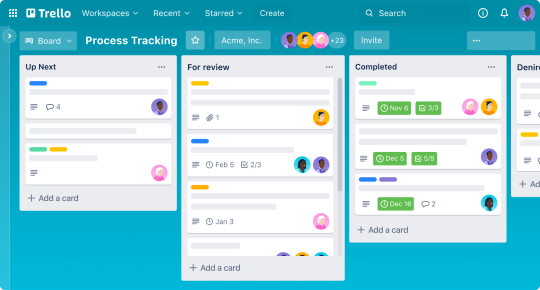


Рисунок 1.1.2 – Интерфейс приложения “Trello”

Карточки имеют множество возможностей. Они предназначены для обсуждений, голосований, загрузки файлов и данных. Есть возможность задавать дедлайны, назначать текстовые и цветовые метки. Важно заметить, что все участники команды видят изменения в реальном времени и могут наблюдать состояния отдельных исполнителей — онлайн или офлайн.

Плюсы:

* удобный и понятный интерфейс;
* многофункциональность в использовании (можно оставить доску заданий приватной, открывать отдельные карточки задач для других пользователей или сделать доску полностью открытой);
* использование с десктопа, данные синхронизируются между разными устройствами;
* есть возможность добавлять абсолютно любые документы в описание к задачам.

Минусы:

* нельзя фиксировать в программе время, затраченное на задачу;
* нет возможности выводить отчеты по задачам.

*TickTick* – это простой и удобный органайзер и планировщик задач с функцией таск-менеджера [3]. Помогает эффективно управлять задачами, своевременно отслеживать действия, требующие регулярного отслеживания. В приложении также имеется опция фильтрации списков. Интерфейс приложения показан на рисунке 1.1.3.

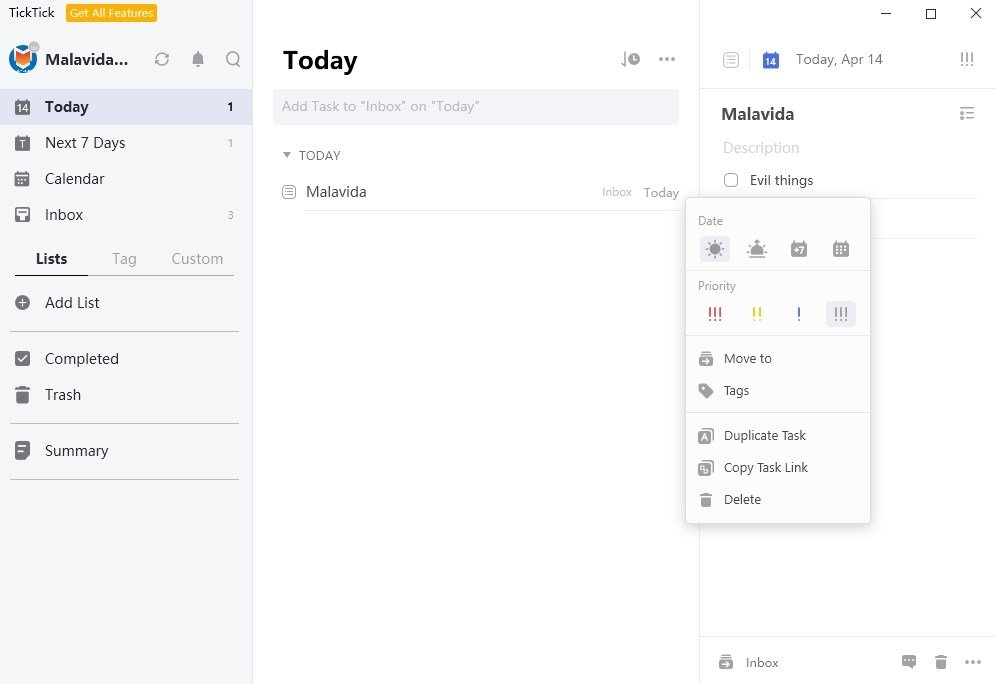


Рисунок 1.1.3 – Интерфейс приложения “TickTick”

Задачи в TickTick, это все ваши дела, которые вам необходимо выполнить. Вы можете присвоить цвета и метки задачам, а также установить дату выполнения. Когда вы создаете задачу, вы вписываете её название, а также добавляете в неё разное содержимое: заметки, чек-листы, файлы.

TickTikc — отличная программа для планирования дня и организации всех своих задач. Также важно отметить, что TickTick способен без потерь в скорости переваривать большие объемы задач, ведь его база данных оптимизирована под высокие нагрузки.

* 1. Постановка задачи

Проанализировав прототипы приложений-планировщиков, выделены основная задача и функциональные требования проекта.

Основной задачей моего курсового проекта является разработка десктопного приложения, позволяющего пользователю добавлять задачи, разбивать их на категории, хранить информацию в централизованной базе данных, а также возможность просмотра статистики выполнения задач. Интерфейс должен быть простым и удобным для использования.

Функциональные требования описаны в главе 2.

1. Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

2.1 Описание средств разработки

Языком разработки курсового проекта является C#. При выполнении курсового проекта будут использованы принципы и приемы ООП. Также будут использоваться технологии Windows Presentation Foundation (WPF).

Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

База данных была разработана в Microsoft SQL Server.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* производительность: SQL Server работает очень быстро;
* надежность и безопасность: SQL Server предоставляет шифрование данных;
* простота: с данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language). Клиент (например, внешняя программа) отправляет запрос на языке SQL посредством специального API. СУБД должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов — Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* code first: разработчик создает класс модели данных, которые будут храниться в бд, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран третий подход.

Приложение необходимо реализовать с использованием шаблона проектирования MVVM и с организацией взаимодействия с базой данных. В качестве клиентской части необходимо реализовать оконное приложение с использованием стандартных и иных библиотек.

* 1. Описание разрабатываемой функциональности приложения

Программное средство предоставляет пользователю следующие функциональные возможности:

* создание, изменение, удаление задач;
* установка сроков выполнения для задач;
* создание периодических задач;
* просмотр статистики за день/неделю/месяц;
* создание категорий и выбор цвета для категории;
* разделение задач по категориям;
* фильтрация задач по категориям;
* установка таймера для задачи;
* просмотр уведомлений о просроченных задачах.

В графической части записки представлена UML-диаграмма возможностей, которая отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

* 1. Спецификация функциональных требований

Для функциональности ПС необходимо создание базы данных для хранения информации приложения. Подробно о разработанной базе данных описано в главе 3.

Основной функцией ПС является создание категорий и добавление задач. Пользователь может просмотреть имеющиеся задачи, а также добавить новые, введя текст в окно для ввода и нажав кнопку «Save» в соответствующем окне. При добавлении задачи пользователь может установить категорию, для которого устанавливается задача, добавить дедлайн, выбрав дату в появившемся календаре, добавить описание, а также указать является ли задача периодической, в таком случае заполнить соответствующие поля сроков периода и частоты появления задачи. После добавления задачи её можно выполнить, изменить или удалить.

Добавленные задачи можно фильтровать по определённой категории. Изначально система сама отфильтровывает список задач по дедлайну и по статусу выполнения. Также возможен отдельный просмотр задач по датам.

Необходимо реализовать возможность установки таймера для задачи и просмотра статистических данных о пользовании приложением. Также появление уведомления о просроченном дедлайне и задачах, чей дедлайн скоро закончится.

Таким образом, в ходе работы над этим разделом были сформулированы основные функциональные треования для проектирования программного средства.

1. Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства

Структура проекта представлена на рисунке 3.1.1.

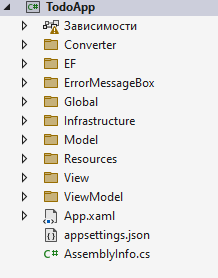


Рисунок 3.1.1 – Структура проекта

Описание приведенной структуры проекта представлено в таблице 3.1.1.

Таблица 3.1.1 – Описание структуры проекта

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Зависимости | Зависимости представляют собой внешние компоненты, библиотеки или пакеты, которые используются вашим проектом для расширения его функциональности или для подключения к сторонним сервисам. |
| EF | Содержит определение контекста базы данных и подключение к ней. |
|  |  |
| Продолжение таблицы 3.1.1 | |
| Model | Подпространство имён, содержащее описание сущностей для работы с БД. |
| Resources | Графические ресурсы приложения, а также словари ресурсов, в которых хранятся стили приложения. |
| Converters | Конвертеры представлений и типов данных для отображения их в TaskList. |
| View | Представления, которые определяют визуальный интерфейс, через который пользователь взаимодействует с приложением. Т.е. это код xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов. |
| ViewModel | Содержит логику по получению данных из модели, которые затем передаются в представление. |
| App.xaml | Основная задача данного файла состоит в определении ресурсов, общих для приложения. |
| Appsettings.json | Файл, который содержит строку подключения к бд |

Таким образом, сформированная таблица помогает понять общую стркутура проекта проектируемого программного средства.

3.2. Архитектура системы

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.2.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.2.1. ‒ Структура шаблона MVVM

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Нередко все сущности (модель) наследует интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged().

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”).

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

В разрабатываемом приложении для работы с базой данных используется фреймворк Entity Framework. Entity Framework предоставляет мощные инструменты для взаимодействия с базой данных, облегчая разработку и управление данными. С его помощью можно создавать модели данных, выполнять запросы, вставлять, изменять и удалять записи в базе данных.

Entity Framework обеспечивает абстракцию от конкретной базы данных, позволяя разработчику сосредоточиться на логике приложения, а не на деталях взаимодействия с базой данных. Это упрощает разработку и обслуживание приложения, а также повышает его гибкость и масштабируемость.

Использование Entity Framework позволяет эффективно работать с базой данных без необходимости прямого взаимодействия с SQL-запросами и управления соединениями с базой данных. Entity Framework также обладает возможностью автоматического создания схемы базы данных на основе моделей данных приложения, что значительно упрощает процесс развертывания и обновления базы данных.

3.3 Схема работы приложения

На рисунке 3.3 представлена схема навигации по страницам приложения.

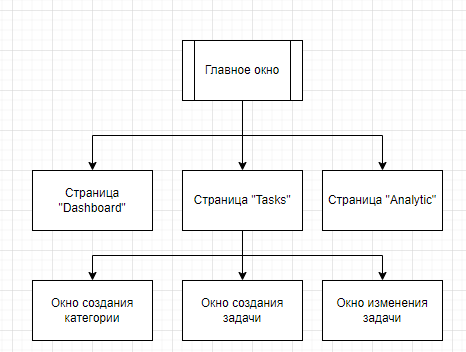


Рисунок 3.3 – Схема навигации по страницам приложения

При запуске приложения появляется главное окно с открытой страницей «Dashboard». С помощью навигационного меню можно переключаться между страницами. Для просмотра страниц добавления задачи или категории нужно нажать на кнопку добавления задачи или категории соответственно.

3.4 Проектирование логики сценариев использования

При проектировании приложения были рассмотрены различные сценарии использования. В графической части представлена диаграмма последовательности добавления задачи.

Из диаграммы видно, что, нажав на кнопку добавления задачи, у пользователя появится возможность добавить задачу. Для выбора категории приложение отправит запрос в базу данных на получение информации обо всех заполненных в расписании предметах. Далее, когда база данных вернет информацию о них, она будет показана пользователю в ComboBox. Заполнив все оставшиеся поля, появится возможность сохранить задачу, нажав на кнопку «Save». После этого, приложение отправит запрос на добавление этой задачи в соответствующую таблицу. Когда задача будет сохранена, она отобразиться в левой части приложения в списке задач.

3.5. Проектирование логической структуры базы данных

Для создания базы данных в приложении была выбрана локальная версия системы управления реляционными базами данных Microsoft SQL Server, известная как LocalDB.

База данных – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ. LocalDB предоставляет широкий набор функциональных возможностей для работы с данными в разрабатываемом приложении. Она поддерживает язык SQL (Structured Query Language), который позволяет выполнять запросы к базе данных, создавать и изменять таблицы, а также управлять данными.

База данных в проекте состоит из 4 таблиц, которые были спроектированы с учетом требований приложения. Каждая таблица имеет свою структуру, определяющую набор полей и их типы данных. Различные таблицы могут быть связаны между собой с помощью ключей, что позволяет устанавливать отношения и связи между данными. Схема базы данных представлена на рисунке 3.5.1.

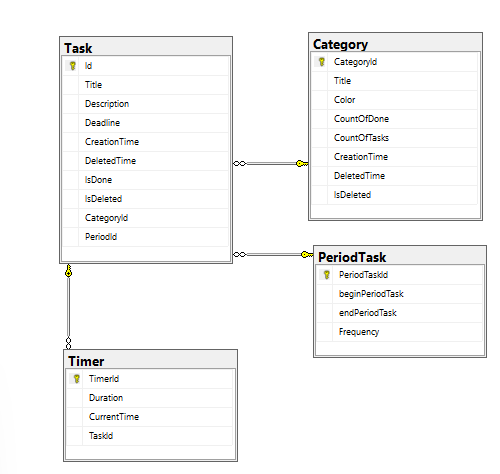


Рисунок 3.5.1 – Схема базы данных

Схема базы данных, представленная на рисунке 3.5.1, визуально представляет структуру и связи между таблицами, что помогает разработчикам и администраторам баз данных лучше понимать организацию данных и облегчает выполнение операций с базой данных, таких как добавление, изменение или удаление данных.

Таблица Task используется для хранения данных о задаче. Включает 10 столбцов: Id – id задачи, Title – название задачи, Description – описание задачи, Deadline – срок выполнения задачи, CreationTime – время создания, DeletedTime – время удаления, IsDone – выполнена ли задача, IsDeleted – удалена ли задача, CategoryId –id категории, PeriodId – id периодической задачи.

Таблица Category используется для хранения данных о категориях. Включает 8 столбцов: CategoryId – id категории, Title – название категории, Color – цвет категории, CountOfDone – количество выполненных задач в категори, CountOfTasks – количество задач в категории, CreationTime – время создания, DeletedTime – время удаления, IsDone – выполнена ли задача, IsDeleted – удалена ли категория.

Таблица PeriodTask используется для хранения данных о периодических задачах. Включает 4 столбца: PeriodId – id задачи, BeginPeriodTask – дата начала периода, EndPeriodTask – дата конца периода, Frequency – частота появления задачи, она может принимать 3 значения («daily», «weekly», «monthly»), которые устанавливаются пользователем в ходе работы с приложением.

Таблица Timer используется для хранения данных о таймере. Включает 5 столбца: TimerId – id таймера, Duration – продолжительность таймера, CurrentTime – текущее время таймера, TaskId – id задачи, с которой связан таймер.

Таблицы Task и Category соединены связью «один ко многим», т. е. объекту одной таблицы можно сопоставить несколько объектов другой таблицы. Остальные задачи связаны связью «один к одному», т.е. объекту одной таблицы можно сопоставить только один объект другой таблицы.

4. Создание (реализация) программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Выполняемые функции

Структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними, широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На диаграмме классов изображены классы программы, методы, поля и свойства классов. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов). Диаграмма классов проекта «Планировщик задач» представлена в графической части.

Рассмотрим реализацию команды для добавления периодических задач, представленную в приложении A.

Добавление периодических происходит следующим образом: пользователь в окне добавления задачи заполняет все поля и также поля для периодических задач (дату окончания периода и частоту), далее нажимает кнопку «Save» для сохранения данных, которые он ввёл. Система создаёт сразу все экземпляры данных задач для выбранного периода и с нужной частотой. В списке задач будет показываться только тот экземпляр периодической задачи, дедлайн которого равен сегодняшнему дню. Остальные экземпляры можно посмотреть на странице «Analytic», выбрав в календаре нужную дату для просмотра.

4.2 Реализация общей структуры проекта

При создании приложения использовался паттерн проектирования MVVM. Он заключается в разделении представления от бизнес-логики напрямую. Это достигается за счёт ввода новой логической конструкции – ViewModel, которая связывает представление и бизнес логику приложения.

В Model находится структура модели базы данных приложения. При проектировании базы данных данного курсового проекта для взаимодействия с базой данных используется Entity Framework.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из базы данных, но и получать объекты, связанные различными ассоциативными связями.

Нередко все сущности (модели) наследуют интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged() как показано ниже:

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName]string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

Листинг 4.2.1 – OnPropertyChanged

Для того, чтобы начать отслеживание свойства, необходимо при его установке вызывать метод OnPropertyChanged(“Имя свойства”) как показано ниже:

public int CountOfDone

{

get { return countOfDone; }

set

{

countOfDone = value;

OnPropertyChanged("CountOfDone");

}

}

Листинг 4.2.2 – Применение OnPropertyChanged

В проекте используется паттерн MVVM, а это значит, что должны присутствовать ViewModels, которые связывают View и Model. Во ViewModel также должна информация обрабатываться и выводится/записываться в нужном виде. Ниже представлена структура пакета ViewModel (рисунок 4.2.1).

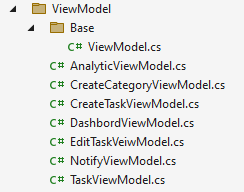


Рисунок 4.2.1 – Структура пакета ViewModel

На следующем рисунке 4.2.2 показана структура пакета View, в котором хранятся файлы представлений на языке разметки xaml, а также соответствующие им файлы на языке C#.

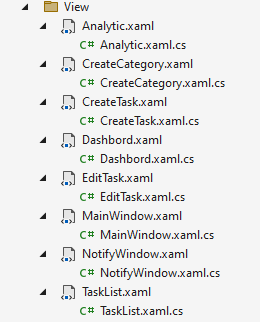


Рисунок 4.2.2 – Структура пакета View

В этом пакете находятся все «окна» и «страницы», которые мы видим, используя приложение.

Назначение каждого файла описано в таблице ниже (таблица 4.2).

Таблица 4.2 – Описание структуры проекта

|  |  |
| --- | --- |
| Название | Краткое описание |
| Analytic.xaml | Окно со статистикой. Позволяет посмотреть статистику за день/неделю/месяц, а также посмотреть задачи на определенную дату. |
| CreateCategory.xaml | Окно создания категории. Имеет поле ввода названия и кнопки выбора цвета категории. |
| CreateTask.xaml | Окно создания задачи. Имеет поля ввода данных о задаче. |
| Dashboard.xaml | Окно которое содержит информационные блоки, а также позволяет установить таймер для задачи. |
| EditTask.xaml | Окно изменения задачи. При открытии этого окна пользователь может изменить данные о задаче или вовсе удалить её. |
| MainWindow.xaml | Главное окно приложения. |
| NotifyWindow.xaml | Окно уведомлений. Данное окно появляется при нажатии на иконку уведомлений и показывает просроченные задачи. |
| TaskList.xaml | Окно со списком задач и категорий, где можно просматривать задачи, отфильтровывать их по категории и переходить на окна создания задач/категорий. |

Таким образом, была рассмотрена структура проекта, а более подробно структура пакета «View».

5. Тестирование, проверка работоспособности и анализ полученных результатов

Для обеспечения корректности работы программы обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

На странице создания задач присутствуют различные способы валидации и защиты от ошибок пользователя. Так, например, пользователь не может задавать дату дедлайна раньше текущей (рисунок 5.1).

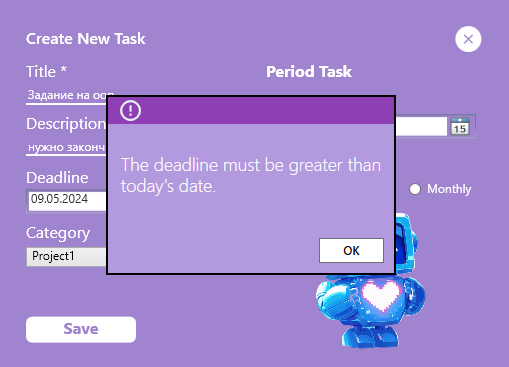


Рисунок 5.1 – Ограничение на выбор даты дедлайна

Точно такая же проверка присутствует и для даты периодических задач. Там также присутствует проверка: пользователь не может ввести дату конца периода раньше текущей (рисунок 5.2).

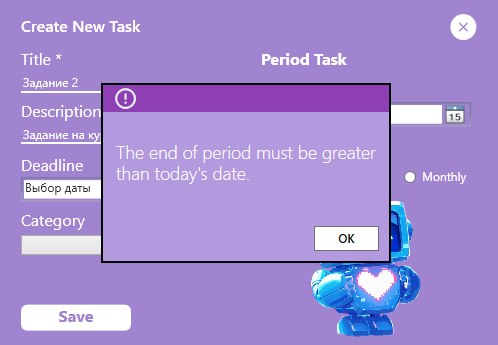


Рисунок 5.2 – Ограничение на выбор даты периодических задач

При попытке создать задачу с незаполненным полем названия также возникает ошибка (рисунок 5.3).

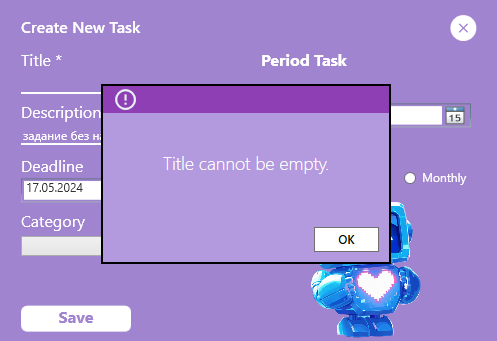


Рисунок 5.3 – Не заполненное поле с названием

Аналогичное исключение появляется при попытки создать задачу с пустым полем описания (рисунок 5.4).

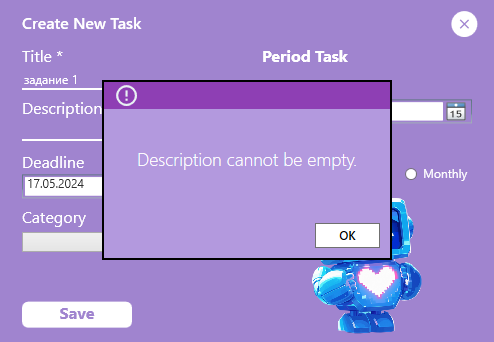


Рисунок 5.4 – Не заполненное поле с описанием

При создании категории пользователь не сможет создать категорию без названия и без выбранного цвета. На рисунке 5.5 продемонстрирована обработка ошибки, если пользователь будет пытаться создать категорию без названия.

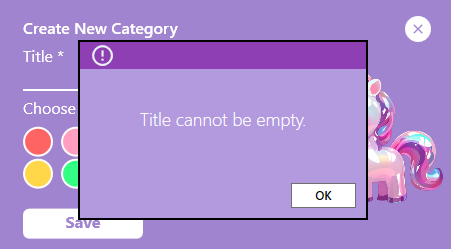


Рисунок 5.5 – Не заполненное поле с названием

Также пользователь не должен обязательно выбрать цвет категории, если он этого не сделает приложение напомнит ему об этом (рисунок 5.6).

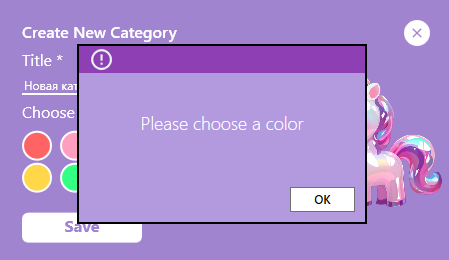


Рисунок 5.6 – Не выбран цвет категории

Чтобы избежать ошибок, связанных с невнимательностью пользователя, когда он случайно нажимает на кнопку удаления задачи или категории, в приложении перед удалением элемента изначально показано окно с вопросом: «Вы точно хотите удалить задачу/категорию?», после чего удаление происходит только при положительном ответе пользователя (рисунок 5.7).

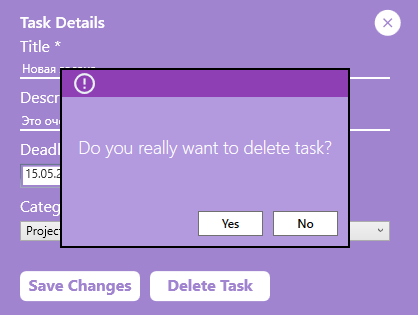


Рисунок 5.7 – Ограничение на случайное удаление

Стоит отметить, что есть такие ошибки, при которых система не будет оповещать о некорректности, а просто не будет совершать действие. Так, например, приложение просто не запустит таймер, у которого не выбрана задача, к которой он привязан. Таким образом, большинство возникающих ошибок в данном приложении были откорректированы.

6. Методика установки и использования программного средства

6.1 Установка программного средства

Для установки программного средства «Планировщик задач», необходимо запустить установочный файл setup.exe. Программное средство работает с локальной базой данных, сервер которой установлен у пользователя на компьютере, поэтому пользователю не нужно устанавливать базу данных вместе с приложением.

При открытии инсталлятора нужно нажать кнопку «Далее» (рисунок 6.1.1).

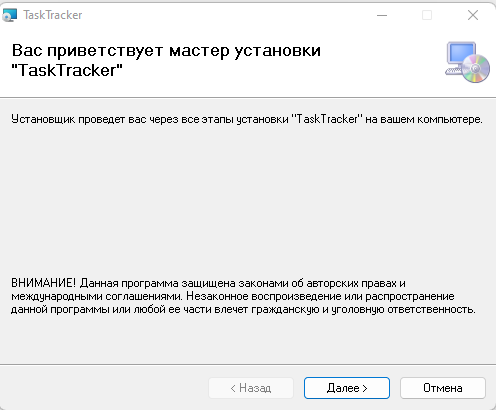


Рисунок 6.1.1 – Первый шаг установки

Следующим шагом нужно выбрать папку, куда инсталлятор будет устанавливать приложение (рисунок 6.1.2). Можно установить по предложенному пути инсталлятора, то есть «C:\Program Files (x86)\TaskTracker». Либо можно выбрать путь установки самим, нажав на кнопку «Обзор». Также можно выбрать будет ли приложение устанавливаться только для вас или для всех пользователей компьютера.

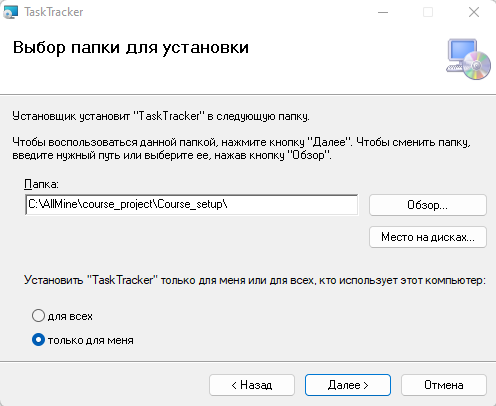


Рисунок 6.1.2 – Второй шаг установки

Следующим шагом нужно нажать кнопку «Далее», после чего начнётся установка приложения (рисунок 6.1.3). Установка может занять какое-то количество времени, нужно подождать её завершения.

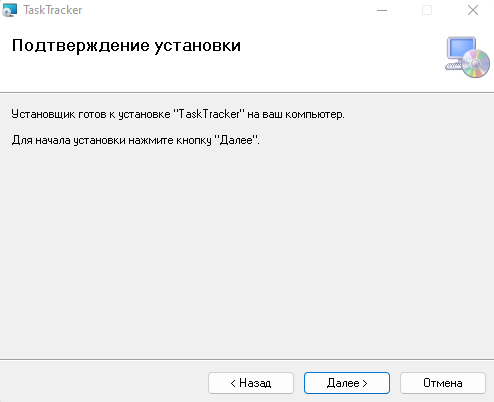


Рисунок 6.1.3 – Третий шаг установки

После этого появится окно, которое уведомляет об успешной установке и возможностью закрыть инсталлятор (рисунок 6.1.4).

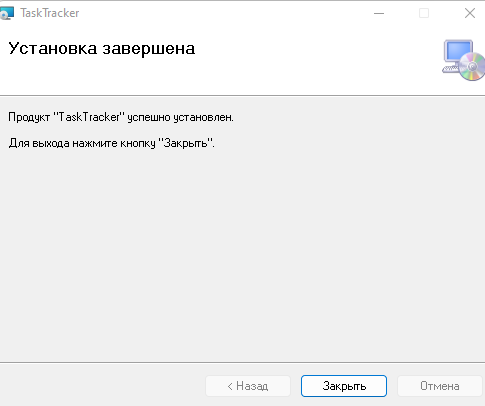


Рисунок 6.1.4 – Четвертый шаг установки

После выполнения этих шагов приложение будет успешно установлено и на рабочем столе компьютера появится ярлык приложения, по нажатию на которое будет запускаться приложение.

6.2 Использование программного средства

При запуске приложения открывается страница «Dashboard», которая представляет собой информационную панель (рисунок 6.2.1). Также слева от страницы находится навигационное меню, с помощью которого можно переходиться по страницам, и вверху программы находится панель с кнопками выхода, информации о программе и просмотра уведомлений, если они сейчас есть.

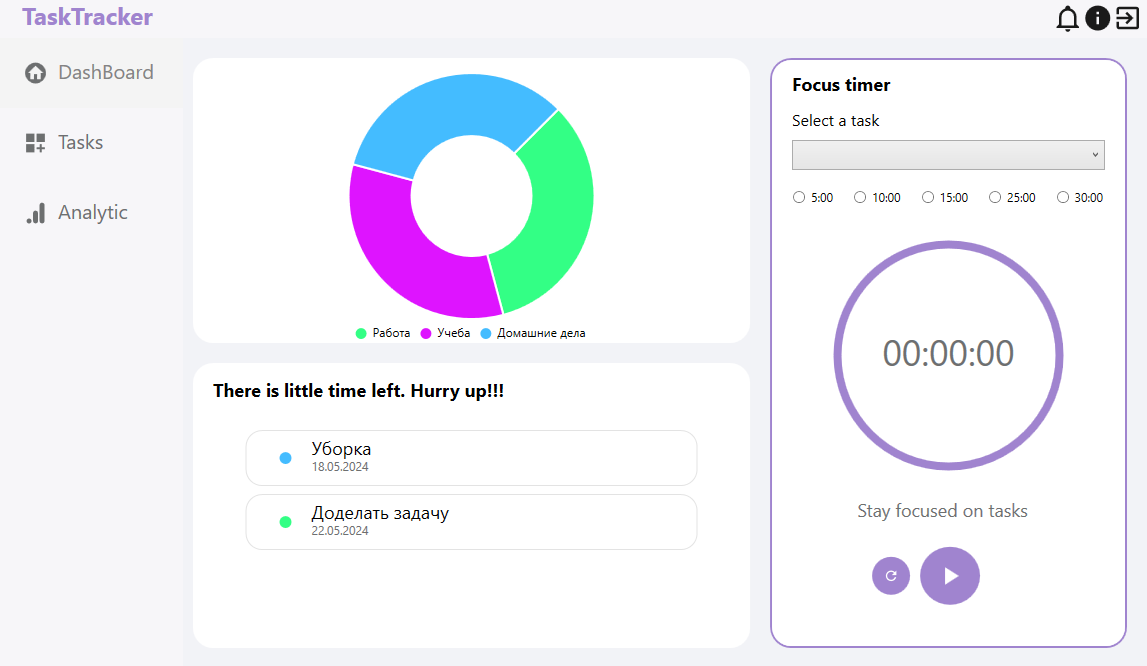


Рисунок 6.2.1 – Страница «Dashboard»

На этой странице можно установить таймер для задачи, чтобы сфокусироваться на её выполнении. Для установки таймера нужно выбрать задачу из выпадающего списка и выбрать время, которое вы хотите потратить на задачу, далее нажать кнопку пуск. После старта таймера вы можете его остановить, нажав на кнопку паузы, или вовсе обнулить, нажав на кнопку обновления.

Также на данной странице представлена круговая диаграмма категорий, где можно увидеть активные категории в данный момент и сколько задач содержит каждая категория.

Внизу страницы находится блок со списком задач, дедлайн которых скоро закончится. Данный блок уведомляет о том, что с определенными задачами лучше поторопиться.

После нажатия на кнопку «Tasks» в навигационном меню, пользователь попадает на страницу списка задач и категорий (рисунок 6.2.2).

На данной странице пользователь может просматривать список доступных задач и категорий. Можно отменить задачу как выполненную, после чего у категории, к которой принадлежит задача, увеличиться заполнение прогресс бара. Можно отфильтровать задачи по категории, с помощью кнопки фильтрации, которая находиться на карточке каждой категории. Также, как и с фильтрацией, на карточке каждой категории есть кнопка удаления категории, после нажатия на которую удалиться выбранная категория и все задачи, связанные с ней.

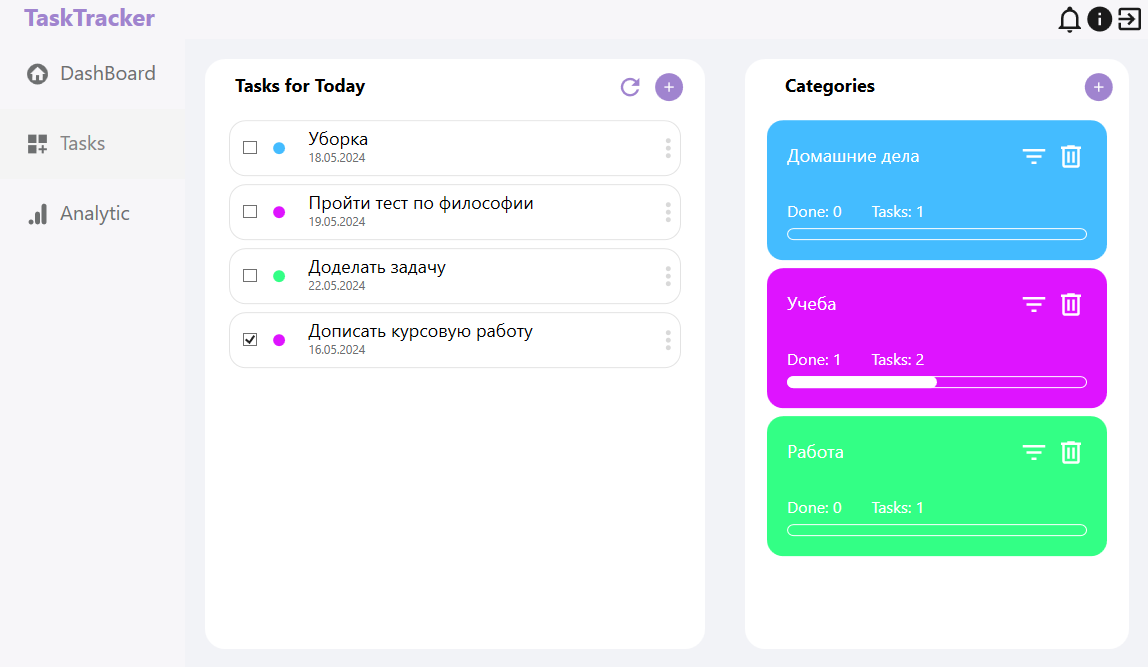


Рисунок 6.2.2 – Страница «Tasks»

Около каждой задачи есть кнопка для более подробной информации, после нажатия на которую открываться окно со всей информацией о задаче (рисунок 6.2.3). На данной странице можно изменить задачу, нажав на кнопку «Save», или удалить задачу, нажав на кнопку «Delete».

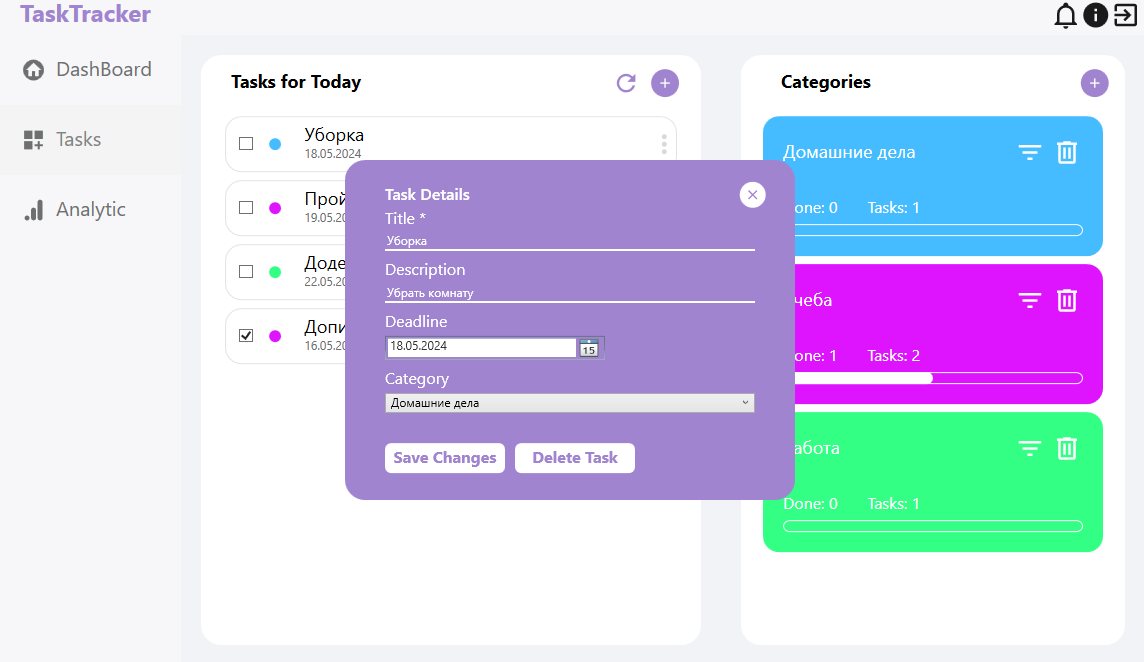


Рисунок 6.2.3 – Окно изменение задачи

После нажатия на кнопки с иконками плюсов на блоке с задачами или с категориями на странице «Tasks», откроется окно для добавления соответственно задачи или категории (рисунки 6.2.4, 6.2.5).

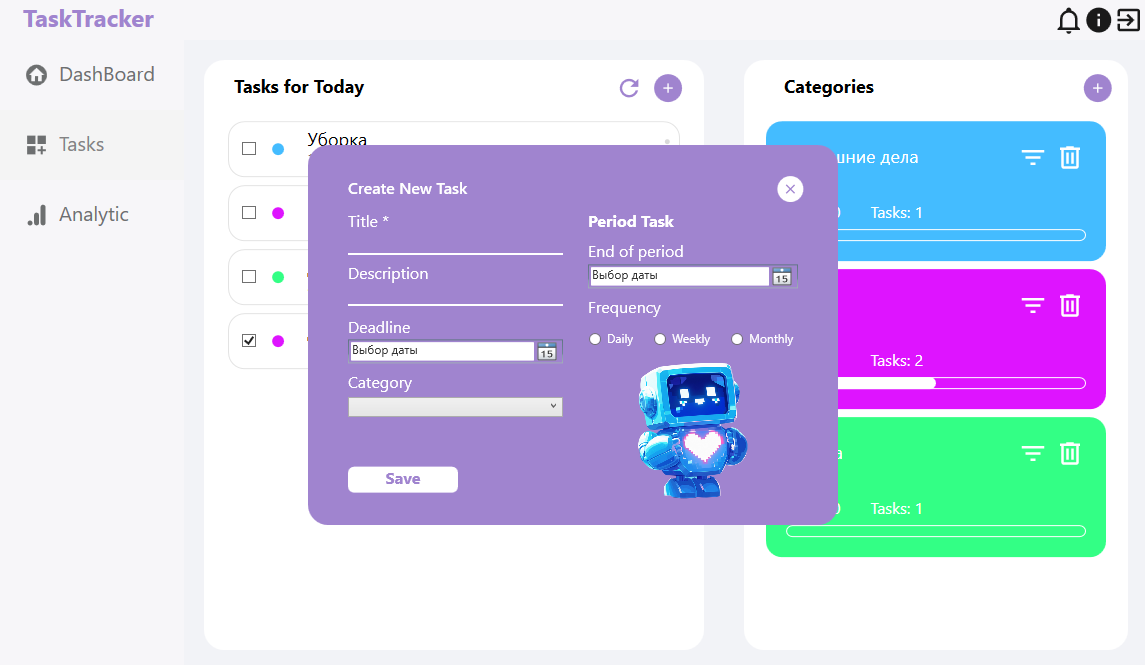


Рисунок 6.2.4 – Окно добавления задачи

Окно добавления задачи имеет поля для ввода данных для создания задач, как обычных, так и периодических. После корректного ввода данных, пользователь должен нажать на кнопку «Save» для сохранения задачи.

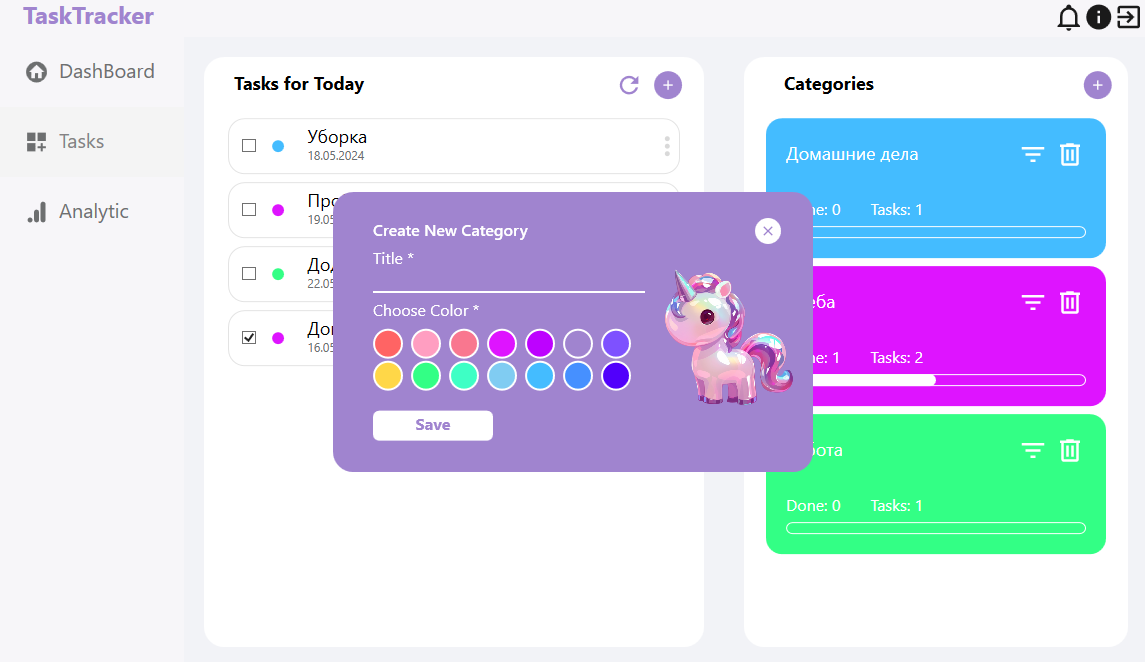


Рисунок 6.2.5 – Окно добавления категории

Окно добавления категории имеет поле для ввода название категории и 14 кнопок с разными цветами. С помощью этих кнопок пользователь может выбрать цвет для категории. И также для сохранение категории нужно нажать на кнопку «Save».

При нажатии на кнопку «Analytic» в навигационном меню пользователь переходит на страницу с аналитикой (рисунок 6.2.6).

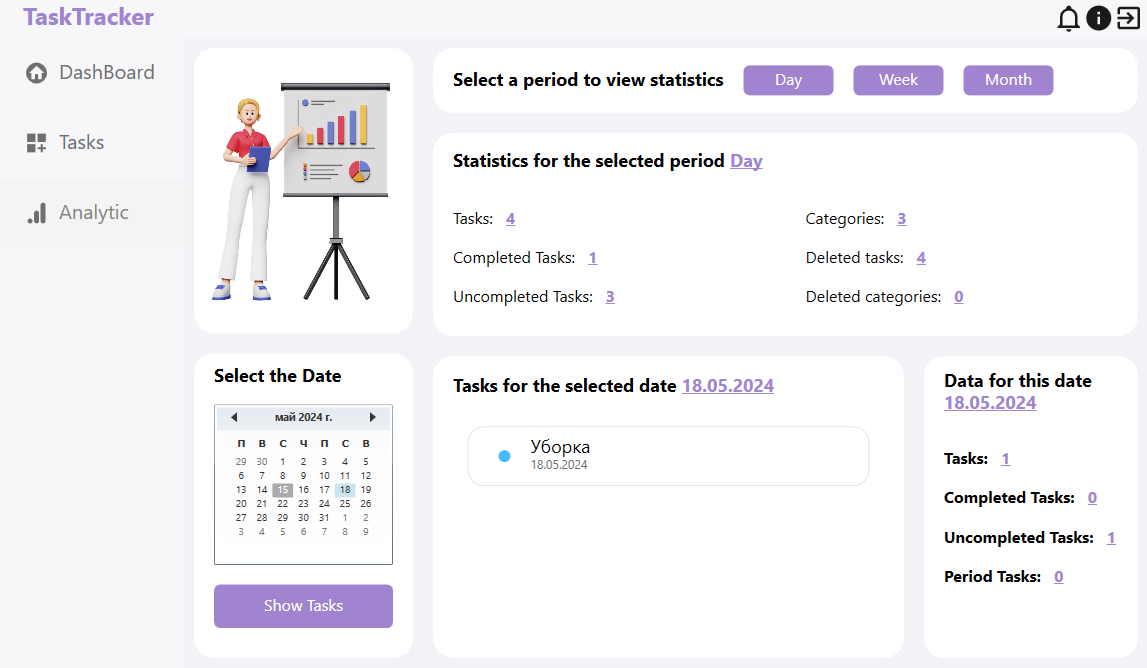


Рисунок 6.2.6 – Окно добавления категории

На данной странице можно посмотреть статистику использования приложения за день/неделю/месяц. Для выбора периода для просмотра статистики нужно воспользоваться кнопками с соответствующими названиями вверху страницы.

Также в левом нижнем углу страницы находиться календарь, где можно выбрать дату и посмотреть задачи за конкретную дату и данные о количестве выполненных/невыполненных задач для текущей даты.

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано программное средство “Планировщик задач”, которое помогает организовывать и планировать свои дела.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* создание, изменение, удаление задач;
* просмотр уже выполненных задач;
* создание категорий;
* фильтрация задач по категориям;
* выбор определенного цвета для категории;
* уведомление о просроченных задачах;
* просмотр статистики за день/неделю/месяц.

Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке. Кроме того, оно имеет удобный и понятный интерфейс.

Проектирование осуществлялось по паттерну проектирования MVVM.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. ToDoist [Электронный ресурс] / Режим доступа: <https://todoist.com/ru>
2. Trello [Электронный ресурс] / Режим доступа: <https://trello.com/ru>
3. TickTick [Электронный ресурс] / Режим доступа: https://ticktick.com/

4) Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2021. – 175 с.

5) Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/>

6) Эндрю Троелсон «Язык программирования C# 6.0 и платформа .NET 4.6»

7) METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com

8) ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru

9) Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>

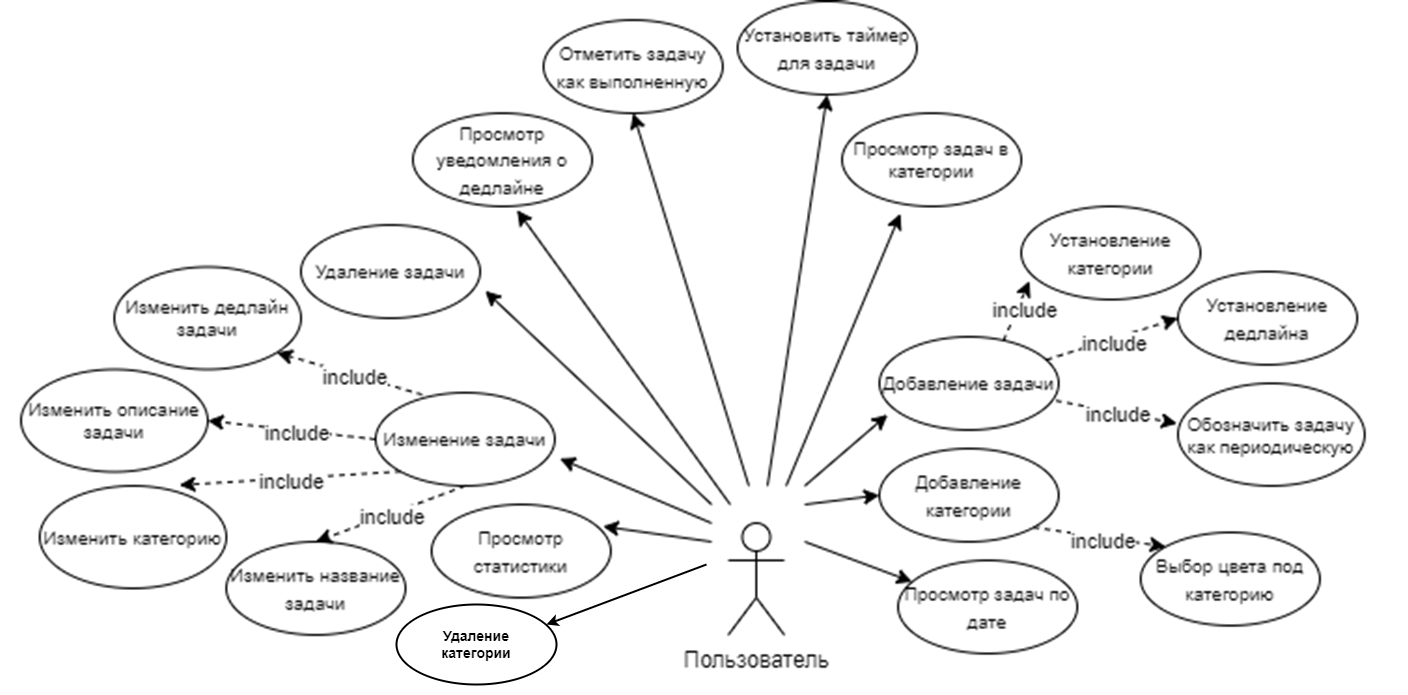
Приложение А

|  |
| --- |
| private void AddPeriodTask()  {  var selectedCategory = db.Categories.FirstOrDefault(c => c.Title == SelectedCategory);  if (SelectedPeriod != null && SelectedFreq != null)  {  if (((DateTime)SelectedPeriod).Date < DateTime.Today)  {  throw new Exception("The end of period must be greater than today's date.");  }  else  {  PeriodTask periodTask = new PeriodTask  {  beginPeriodTask = DateTime.Now,  endPeriodTask = (DateTime)SelectedPeriod,  Frequency = SelectedFreq.ToString()  };  int newPeriodId = periodTask.PeriodTaskId;  db.PeriodTasks.Add(periodTask);  DateTime startPeriod = DateTime.Now;  DateTime endPeriod = (DateTime)SelectedPeriod;  TimeSpan interval;  switch (SelectedFreq)  {  case "daily":  interval = TimeSpan.FromDays(1);  break;  case "weekly":  interval = TimeSpan.FromDays(7);  break;  case "monthly":  interval = TimeSpan.FromDays(30);  break;  default:  throw new Exception("Invalid frequency.");  }  while (startPeriod <= endPeriod)  {  if (startPeriod <= endPeriod)  {  if (selectedCategory != null)  {  Model.Task periodicTask = new Model.Task()  {  Title = Title,  Description = Description,  Deadline = startPeriod,  IsDone = false,  IsDeleted = false,  СreationTime = DateTime.Now,  CategoryId = selectedCategory.CategoryId,  PeriodId = newPeriodId  };  db.Tasks.Add(periodicTask);  }  else  {  Model.Task periodicTask = new Model.Task()  {  Title = Title,  Description = Description,  Deadline = startPeriod,  IsDone = false,  IsDeleted = false,  СreationTime = DateTime.Now,  PeriodId = newPeriodId  };  db.Tasks.Add(periodicTask);  }  }  startPeriod += interval;  }  }  }} |

Листинг – Команда добавления периодических задач

Графический материал

Диаграмма вариантов использования



Диаграммы классов

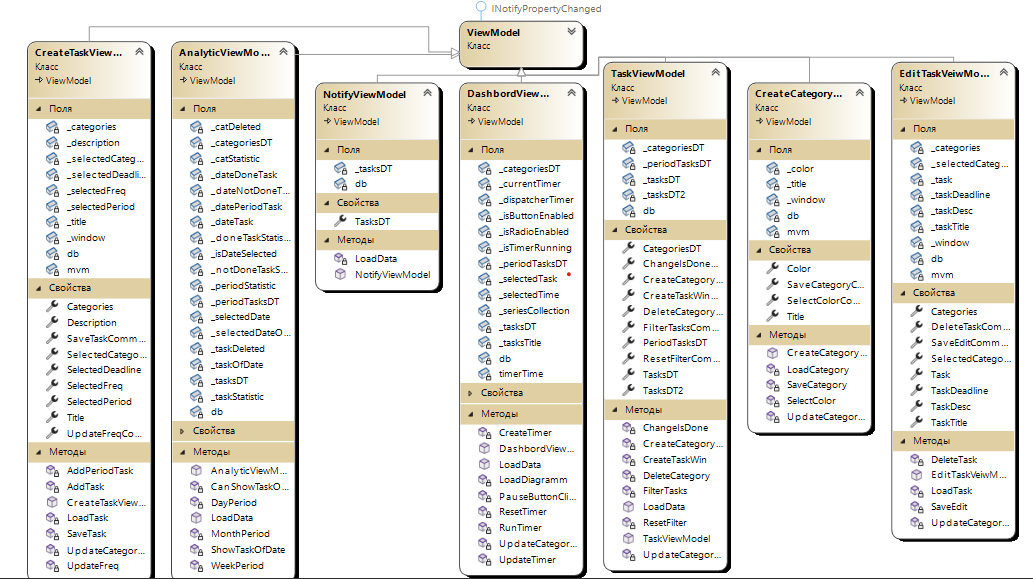


Диаграмма последовательности

