МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПЕУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Объектно-ориентированное программирование"

|  |  |
| --- | --- |
| Специальность: 1-40 05 01-03 Информационные системы и технологии  Студент: Шимчёнок Е.К. | Группа:\_\_1\_\_ |
| **Тема: Программное средство «Ежедневник»** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2022 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции пользователя:
  + Выполнять регистрацию и авторизацию;
  + Добавлять, удалять и редактировать задачи в любой день;
  + Добавлять, удалять и редактировать подзадачи;
  + Отмечать выполненные задачи и подзадачи;
  + Сгруппировывать задачи в группы и просматривать.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение. Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных программного средства (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
* Листинги программы представляются частично в приложении.
* Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2022 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2022 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2022 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2022 |  |
| 5 | Кодирование программного средства | 23.04.2022 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2022 |  |
| 7 | Оформление пояснительной записки | 7.05.2022 |  |
| 8 | Сдача проекта | 20.05.2022 |  |

**5. Дата выдачи задания \_\_\_\_**14.02.2022**\_\_\_\_**

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *О.Л. Панченко*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Е.К. Шимчёнок*

(дата и подпись студента)

**Оглавление**

[**1.** **Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования** 4](#_Toc104890158)

[**1.1** **Обзор и анализ аналогов** 4](#_Toc104890159)

[**2.** **Анализ и проектирование архитектуры приложения** 7](#_Toc104890160)

[**2.1** **Анализ требований к программному средству** 7](#_Toc104890161)

[**2.2** **Разработка функциональных требований** 8](#_Toc104890162)

[**2.3** **Описание средств разработки** 8](#_Toc104890163)

[**3.** **Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса** 10](#_Toc104890164)

[**3.1** **Общая структура проекта** 10](#_Toc104890165)

[**3.2. Взаимодействие окон и представлений** 11](#_Toc104890166)

[**3.3.** **Проектирование базы данных** 12](#_Toc104890167)

[**3.4. Диаграмма последовательности** 13](#_Toc104890168)

[**4.** **Реализация программного средства** 15](#_Toc104890169)

[**4.1. Реализация авторизации и регистрации пользователей** 15](#_Toc104890170)

[**4.2 Работа с базой данных в проекте** 15](#_Toc104890171)

[**5.** **Тестирование программного средства** 17](#_Toc104890172)

[**5.1. Тестирование** 17](#_Toc104890173)

[**ЗАКЛЮЧЕНИЕ** 21](#_Toc104890174)

[**Список использованной литературы** 22](#_Toc104890175)

[**Приложение А** 23](#_Toc104890176)

[**Приложение Б** 24](#_Toc104890177)

[**Приложение В** 27](#_Toc104890178)

[**Приложение Г** 29](#_Toc104890179)

# **Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования**

## **Обзор и анализ аналогов**

Сейчас в нашем распоряжении имеются онлайн-ежедневники и приложения, которые помогают человеку распланировать свой день, экономя свое время и ресурсы. Мы получаем новые возможности: всю нашу информацию о делах можно хранить в одном приложении, забыв навсегда о бумажном носителе.

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

Так как на сегодняшний день существует множество различных электронных ежедневников, планеров и органайзеров, предназначенных для записи дел и планов на будущее, рассмотрим два аналога по интересующему нам местоположению - г. Минск. Первым будет ежедневник «Diaro», доменный адрес которого – https://diaroapp.com/.

Первый пример – приложение для планирования дел под названием «Diaro» (рис. 1.1) [2].

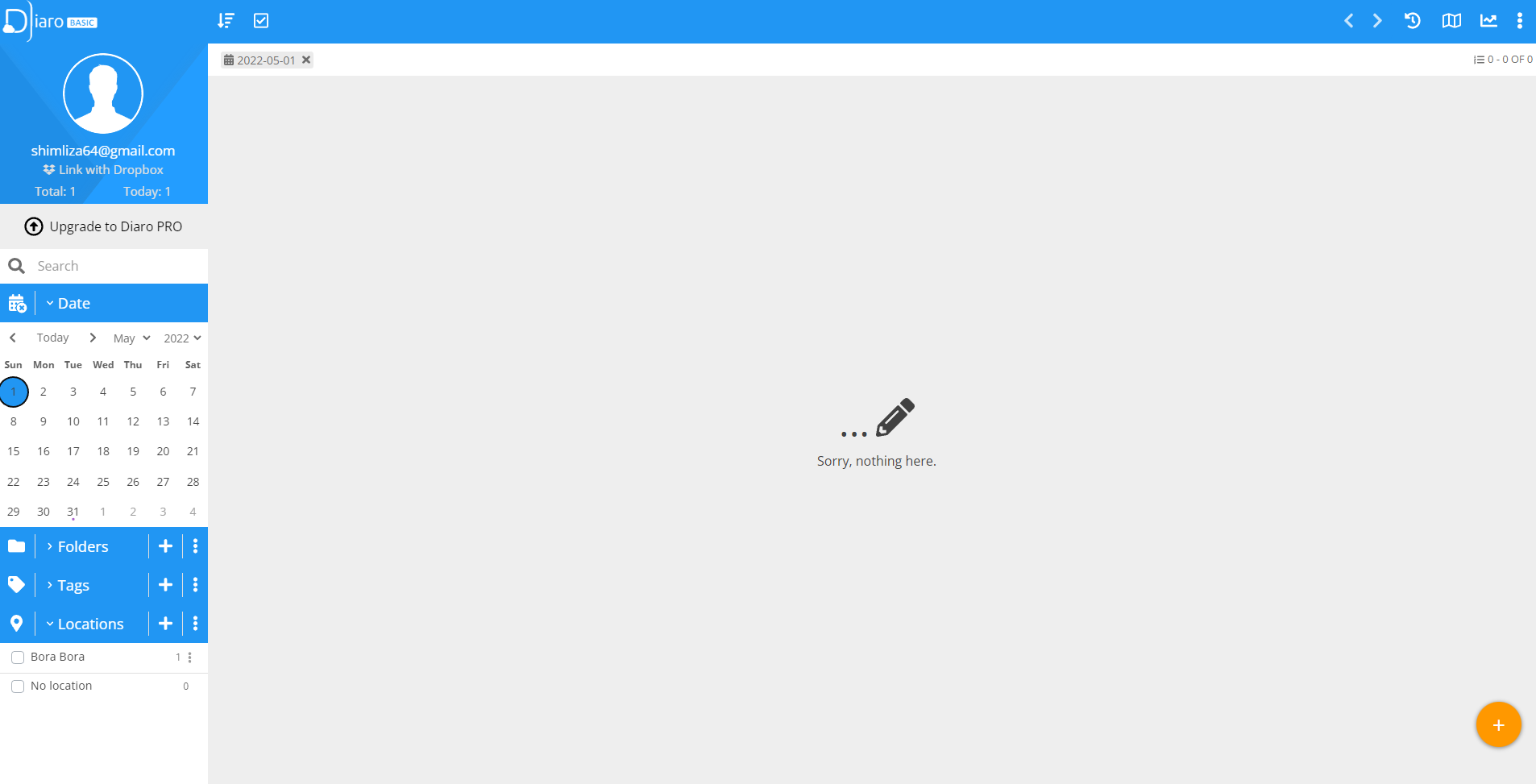


Рисунок 1.1 –Приложение для планирования дел «Diaro»

Интерфейс этого приложения с виду не такой простой и интуитивно понятный, как хотелось бы: отсутствуют подсказки при наведении на объекты: значки, кнопки, различные иконки и т.д. Проект выглядит немного устаревшим – обновление дизайна интерфейса пошло бы ему на пользу.

Данное программное средство имеет обширный функционал. Присутствует возможность сортировать задачи и фильтровать их по местоположению, сфере деятельности и т.д. Не имеется функционал для установки версии для слабовидящих, а также для смены языка.

Самым важным плюсом этого приложения является возможность указать желаемый интервал дат для вывода списка задач. Также можно задать местоположение, необходимую папку (группу) и тэги. Имеются кнопки для возвращения к текущему дню недели, перелистывания дней, выбора места, генерирования диаграммы (рис. 1.2).



Рисунок 1.2 –Кнопки «Текущий день», «Следующий день», «Предыдущий день», «Места» и «Диаграмма»

По итогу можно сказать, что данный ежедневник – достаточно хорошее программное средство для планирования дел, однако имеет изначально непонятный интерфейс и немного тусклый дизайн, который было бы неплохо модернизировать. Функционал обширный и в корректировке не нуждается.

Также отсутствуют подсказки при наведении на объекты: значки, кнопки, различные иконки и т.д.

Второй пример –  программное средство «Tweek Календарь» (рис. 1.3) [3]. Доменный адрес – https://tweek.so/.

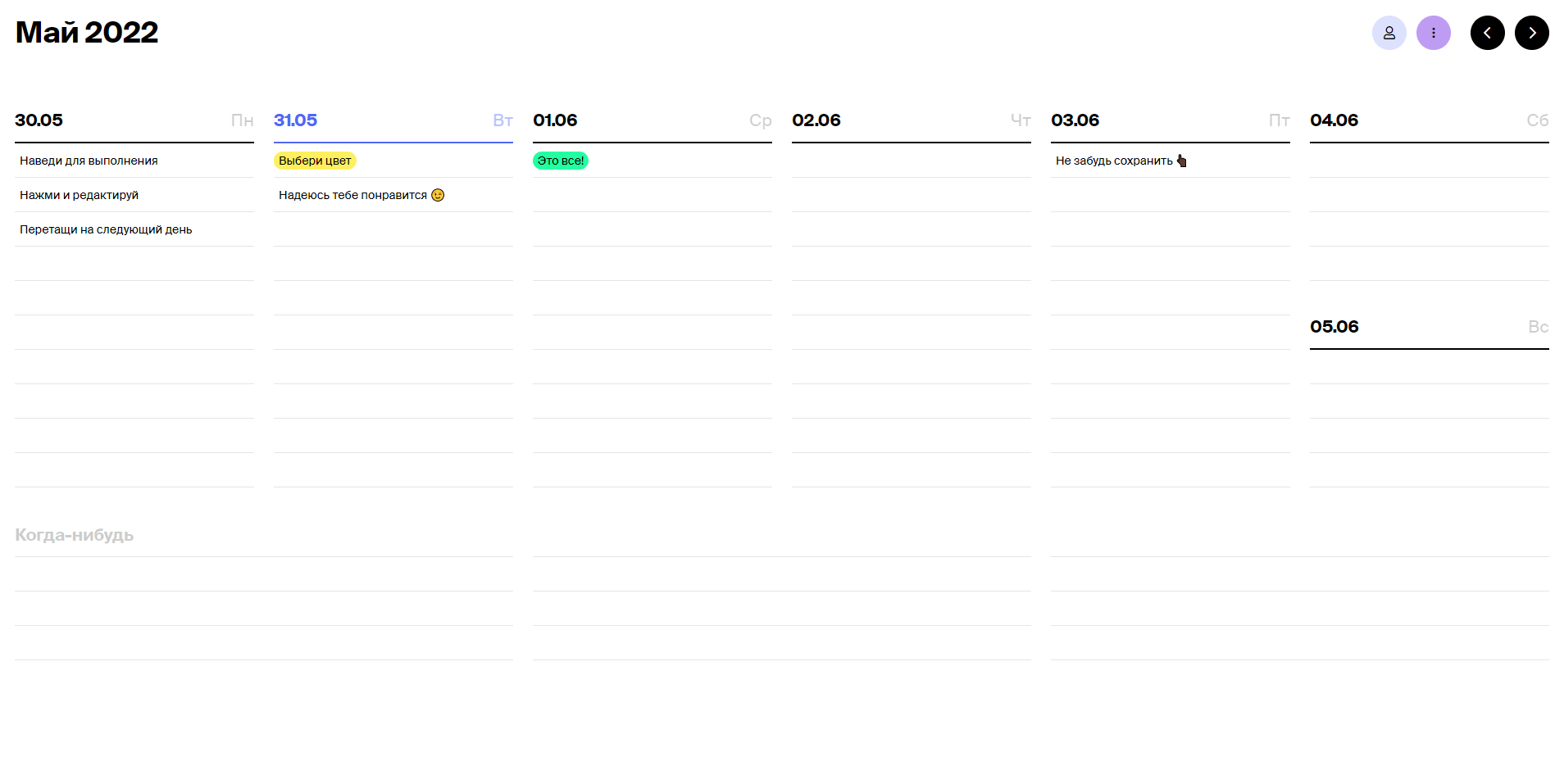


Рисунок 1.3 –Программное средство «Tweek»

Интерфейс этого программного средства максимально прост. Существует только главная страница с полями. Есть кнопки для перехода между неделями (рис. 1.4). Все, что можно сделать пользователю, это записать дела в определенный день недели.

Очень непродуктивно используется пространство на главное странице. Нет никакого поиска или сортировки. Группировка дел невозможна. Очень заметно неравномерное распределение блоков на главном экране. Заметно, что содержимое страницы не отцентрировано. Есть возможность выделять записи раазными цветами. Также можно использовать приложение как гость, однако при выходе все записи исчезнут.

Система не имеет обширный функционал, который позволяет осуществлять все необходимые операции. В целом, его можно использовать только для каких-то повседневных дел.



Рисунок 1.4 –Переход между неделями

Также есть возможность выбора языка (рис. 1.5).

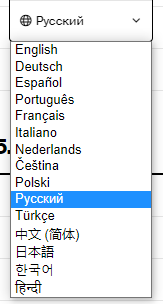


Рисунок 1.5 –Выбор языка

По итогу можно сказать, что данное программное средство является очень удобным приложением для повседневных записей. Приложение не позволяет отображать большое количество дел, не предоставляет никакой группировки, а также не хватает наличие фильтрации и сортировки.

# **Анализ и проектирование архитектуры приложения**

## **Анализ требований к программному средству**

На предыдущем этапе курсового проекта были рассмотрены несколько аналогов программных средств электронного ежедневника. Функционал моего приложения будет включать в себя совокупность основных возможностей аналогов с вариативной доработкой и расширением. Стоит учесть все излишки и проблемы рассмотренных аналогов, а также добавить некоторые дополнительные возможности.

UML — унифицированный язык моделирования (Unified Modeling Language) — это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

Диаграмма — это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования.

Диаграмма деятельности, как и диаграмма состояний, отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

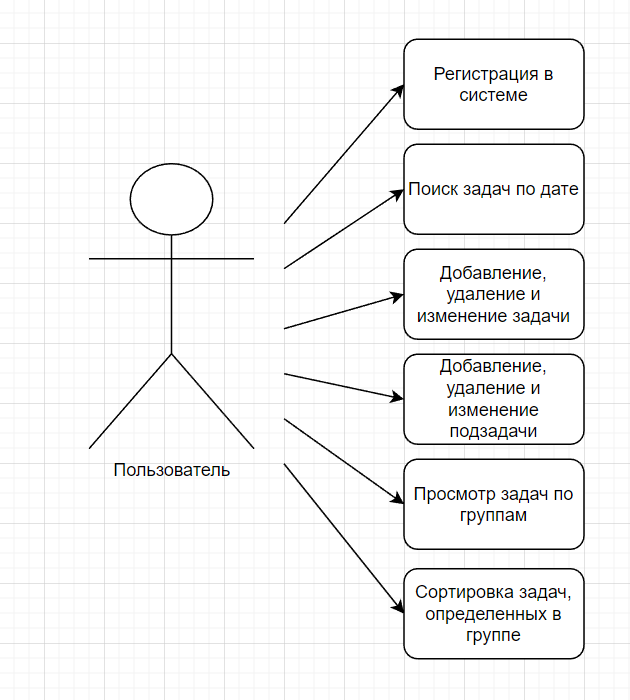
Часть диаграммы деятельности данного приложения приведена в приложении А, данная диаграмма приведена для части программы, связанной со входом пользователя в приложение. Она разбита на 2 части: авторизация и регистрация, для каждой из этих частей приведен свой алгоритм, осуществляющий все проверки и этапы обработки информации необходимые для корректной работы данной части приложения.

Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования.

Благодаря анализу похожих по функционалу приложений, было принято решение проектировки функционала для одной основной категории пользователей: пользователи. Представление общего функционала для группы можно увидеть на Use Case диаграмме (рис. 2.1).

Описание функциональности программного средства представлено с помощью UML-диаграммы вариантов использования, которая представлена в приложении А.

Диаграмма классов представлена в приложении Д.



Use Case диаграмма (рис. 2.1).

## **Разработка функциональных требований**

Разработаем примерный список функций, которые сможет выполнять пользователь.

В основной функционал пользователя должны входить регистрация с заполнением некоторых полей, функции поиска задач по дате, добавление задач, подзадач, возможность из редактирования и удаления, а также группировка по группам.

## **Описание средств разработки**

При разработке приложения были использованы:

* интегрированная среда разработки Microsoft Visual Studio 2019;
* программная платформа .NET 5.0;
* язык программирования C#;
* расширяемый язык разметки XAML;
* технология WPF;
* технология Entity Framework 6.4.4;
* Microsoft SQL Server 2019.

В качестве интерфейса прикладного программирования был выбран обширный API-интерфейс – Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. В основе графической технологии WPF лежит мощная инфраструктура, основанная на DirectX. Это является одним из основных отличий WPF от более ранней технологии создания пользовательских интерфейсов – Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом – С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология Entity Framework. Entity Framework предоставляет возможность взаимодействия с объектами посредством LINQ to Entities. Центральной концепцией Entity Framework является понятие сущности или entity. Сущность представляет набор данных, ассоциированных с определенным объектом. Поэтому данная технология предполагает работу не с таблицами, а с объектами и их наборами.

# **Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса**

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.

## **Общая структура проекта**

В данной главе будут описаны все шаги проектирования и разработки приложения, в результате которых, получилась определенная структура решения (рис. 3.1).

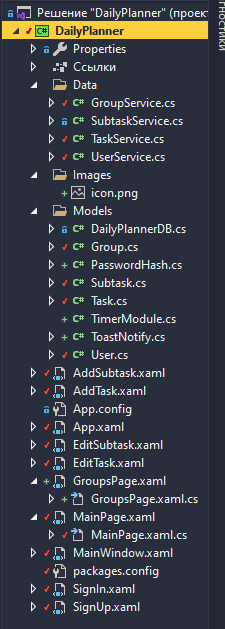


Рисунок 3.1 –Структура проекта

В решении присутствуют разные виды классов. Опишем их более подробно:

* файлы MainWindow.xaml, MainPage.xaml, SignIn.xaml, SignUp.xaml, AddSubtask.xaml, AddTask.xaml, EditTask.xaml и др. являются основными классами, реализующими окна WPF приложения, и содержат xaml разметку. Их аналоги с расширением xaml.cs содержат всю необходимую логику взаимодействия, описанную на языке C#. MainWindow представляет собой страницу входа, перенаправляющий нас на SignIn – страницу входа пользователя. Пользователь может войти в аккаунт или зарегистрироваться, после чего попадает на главную страницу, где может добавить, удалить или изменить задачу, подзадачу.
* со страницы MainPage.xaml пользователь может перейти на страницу GroupsPage.xaml, чтобы просмотреть группы и задачи или воспользоваться поиском.
* папка Images содержит всю графическую информацию, необходимую для приложения, такую как изображения и иконки;
* классы ToastNotify.cs, PasswordHash.cs необходимы для того, чтобы мы могли реализовывать какие-либо дополнительные функции, которые не смогли бы реализовать с имеющимися классами.

Разработка приложения выполнялась в следующей последовательности:

* проектирование и реализация базы данных;
* подключение базы данных к проекту;
* проектирование и реализация xaml разметки окон приложения;
* создание дополнительных классов для обеспечения работы функционала;
* улучшение внешнего вида программы.

## **3.2. Взаимодействие окон и представлений**

При запуске приложения в окне запускается представление SignIn с авторизацией, запрашивающее email и пароль. Если нужно зарегистрироваться, это представление по клику на кнопку сменится на представление регистрации SignUp. После успешной авторизации SignIn переключается на представление MainPage, которое вмещает в себе меню и все то, чем может управлять пользователь. То есть, внутри MainPage пользователь может попасть в представления AddTask, AddSubtask, EditTask, EditSubtask, а также в представление GroupsPage. Это продемонстрировано на рисунке (рис. 3.2).

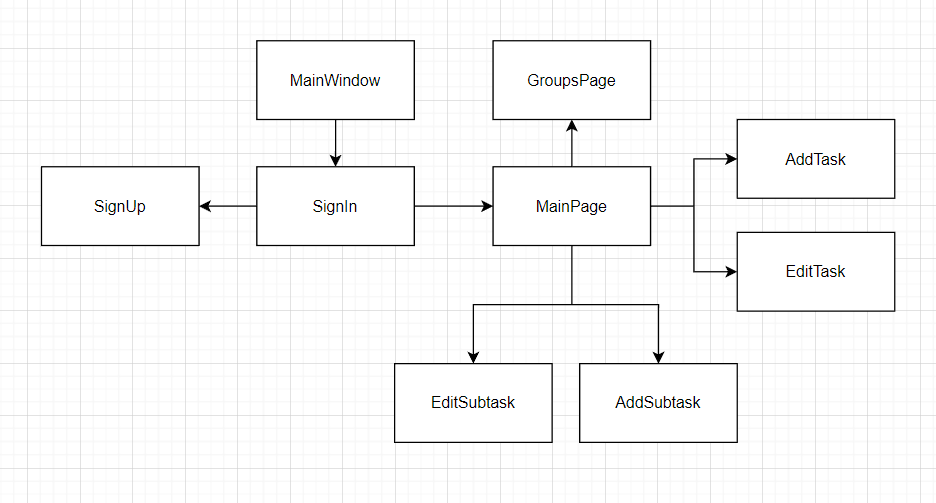


Рисунок 3.2 – взаимодействие окон и представлений

## **Проектирование базы данных**

Для реализации поставленной задачи была создана база данных. Для её создания использовались система управления реляционными базами данных Microsoft SQL Server 2019.

База данных – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимо от прикладных программ.

Система управления базами данных – совокупность программ и языковых средств, предназначенных для управления данными в базе данных, ведения базы данных и обеспечения взаимодействия ее с прикладными программами.

База данных программной подсистемы ежедневник состоит из 4 таблиц, ее диаграмма изображена на рис 3.3.

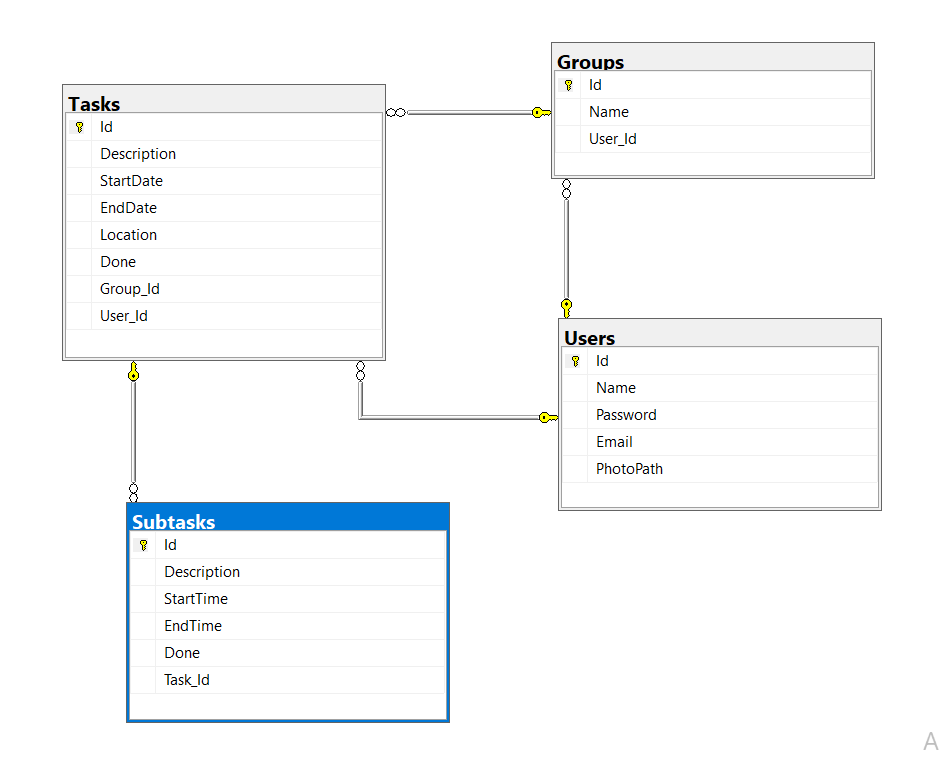


Рисунок 3.3 –Диаграмма базы данных электронного ежедневника

Таблица Users используется для хранения данных о пользователях. Включает 5 столбцов: email – мейл, password – пароль, name – имя, id – идентификатор, PhotoPath – строка пути к фотографии.

Таблица Tasks используется для хранения данных о задачах. Включает 8 столбцов: id – идентификатор, description – действие/описание задачи, startDate – начало задачи/события, endDate – конец задачи/события, location – местоположение события, done – выполнено/не выполнено, done – выполнено/не выполнено, group\_id – связь с группой из таблицы Groups, User\_Id – связь с пользователем из таблицы Users.

Таблица Subtasks используется для хранения данных о подзадачах. Столбцы: id, description, starttime, endTime, Done, task\_id.

Таблица Groups отображает информацию о группах. Столбцы: id, name, user\_id.

## **3.4. Диаграмма последовательности**

Диаграмма последовательности входа в приложение представлена на рис. 3.4. Как можно заметить на схеме, процесс состоит из следующих этапов: ввод логина и пароля, нажатие кнопки «Войти», валидация введенных данных, проверка соответствия логина и пароля в базе данных, и соответственно ответные реакции системы.

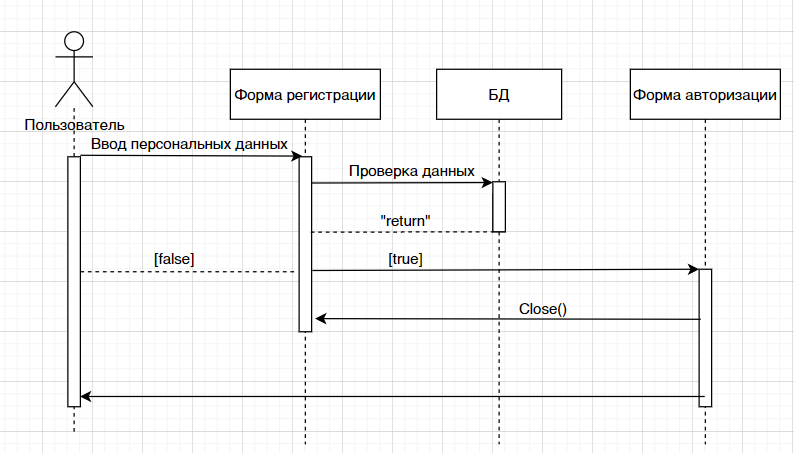


Рисунок 3.4 –Диаграмма последовательности входа в приложение

1. **Реализация программного средства**

## **4.1. Реализация авторизации и регистрации пользователей**

Чтобы пользоваться приложением, каждый пользователь должен войти в систему, предварительно зарегистрировавшись.

На рисунке 4.1 находится код, реализующий регистрацию.

|  |
| --- |
| public static void AddUser(User user)  {  user.Password = PasswordHash.GetHash(user.Password);  using (DailyPlannerDB plannerDB = new DailyPlannerDB())  {  plannerDB.Users.Add(user);  plannerDB.SaveChanges();  }  } |

Рисунок 4.1 – Регистрация пользователя

При авторизации проверяется, существует ли пользователь с такими данными, а затем проверяется, совпадает ли пароль с паролем в базе данных.

На рисунке 4.2 находится код, реализующий авторизацию.

|  |
| --- |
| public static User GetUserbyLoginPassword(string login, string password)  {  using (DailyPlannerDB plannerDB = new DailyPlannerDB())  {  DailyPlannerDB.CurrentUser = plannerDB.Users.First(u => u.Email == login && u.Password == password);  return DailyPlannerDB.CurrentUser;  }  } |

Рисунок 4.2 – Авторизация пользователя

## **4.2 Работа с базой данных в проекте**

База данных для приложения создается с помощью Database CodeFirst. Запрос создания таблиц изображен на рис. 4.3.

|  |
| --- |
| public class DailyPlannerDB : DbContext  {  public DailyPlannerDB() : base("DefaultConnection")  {  }  public virtual DbSet<User> Users { get; set; }  public virtual DbSet<Task> Tasks { get; set; }  public virtual DbSet<Group> Groups { get; set; }  public virtual DbSet<Subtask> Subtasks { get; set; }  public static User CurrentUser;  } |

Рисунок 4.3 – Код создания таблиц в БД

В следующих классах хранятся функции, с помощью которых реализуется работа с базой данных (рис. 4.6).

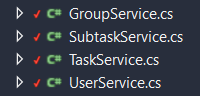


Рисунок 4.6 – Добавление, удаление, изменение и вывод данных в/из БД

1. **Тестирование программного средства**

## **5.1. Тестирование**

В курсовом проекте задействуется обработка ошибок, таким образом, что пользователь будет уведомлен о неудачном выполнении запроса к базе данных, или недоступности данных в формах.

При запуске приложения открывается представление с формой для авторизации, куда следует ввести свой логин и пароль (рис.5.1). Если строки не заполнены, логин или пароль введены неверно, то появляется сообщение, говорящее о произошедшей ошибке. Сообщение представлено на рисунке 5.2.

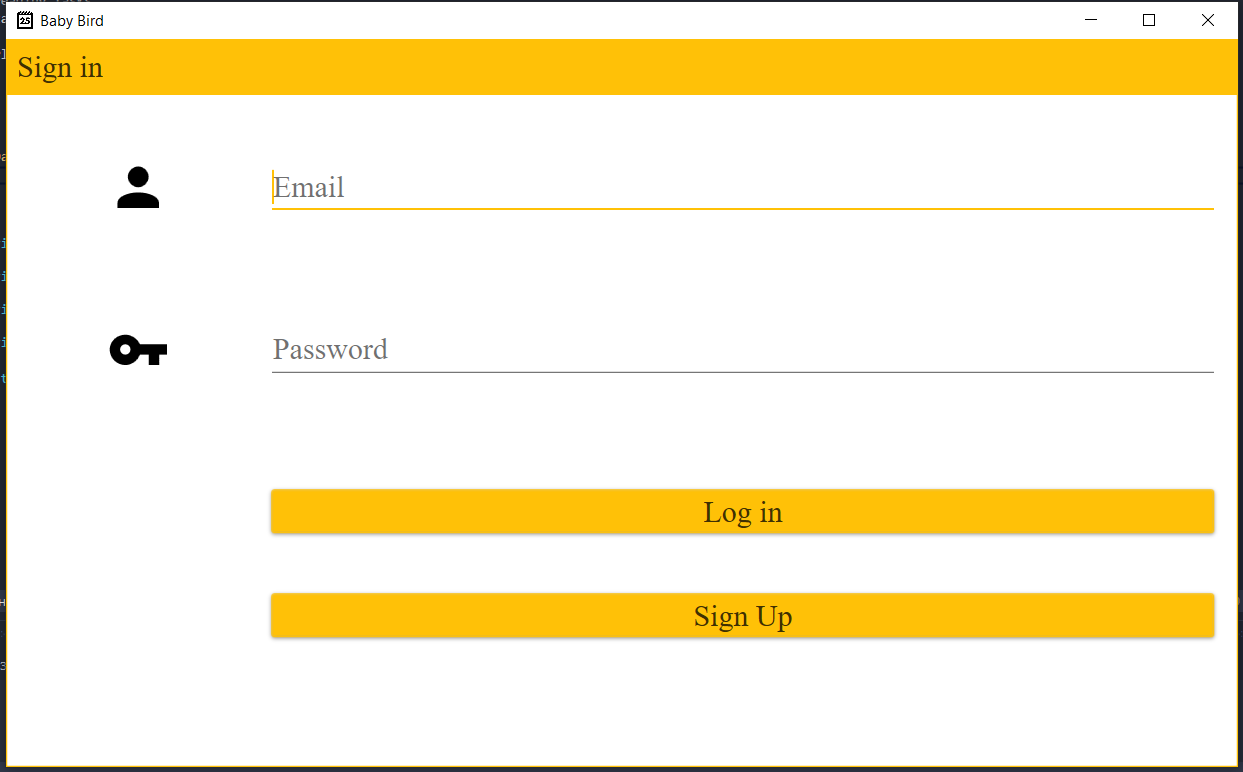


Рисунок 5.1 – Форма входа

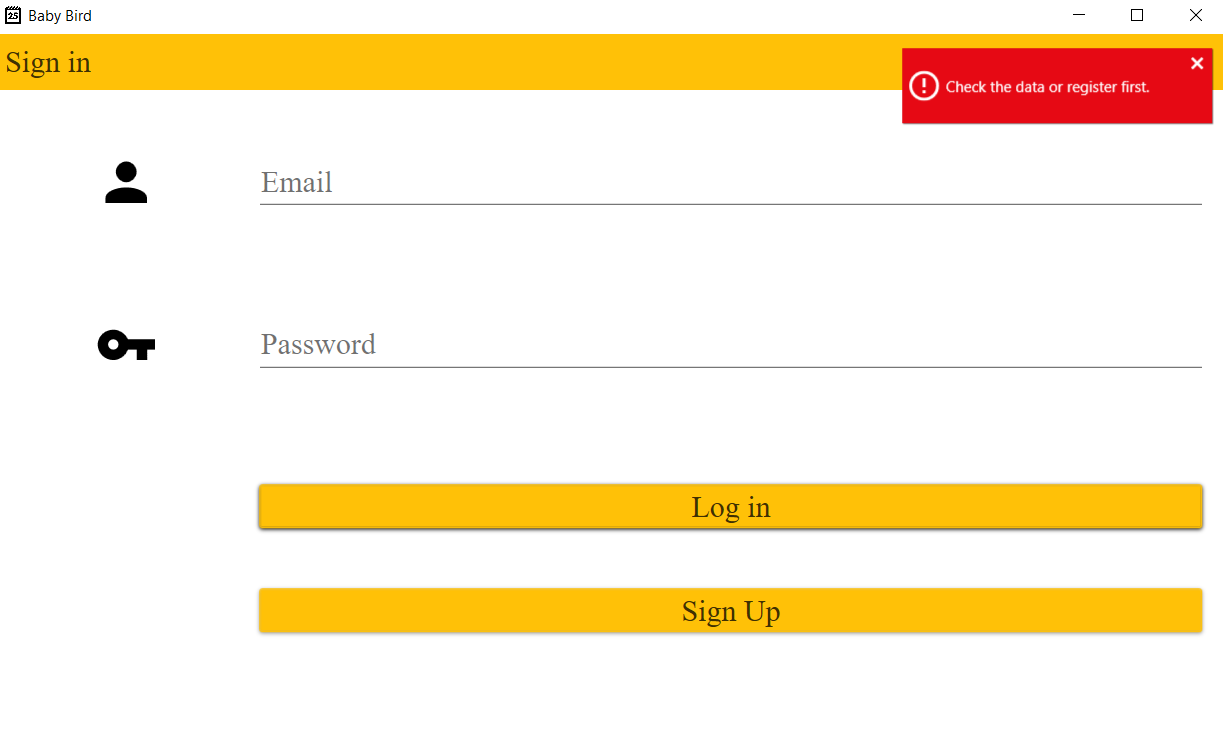


Рисунок 5.2 – Сообщение об ошибке

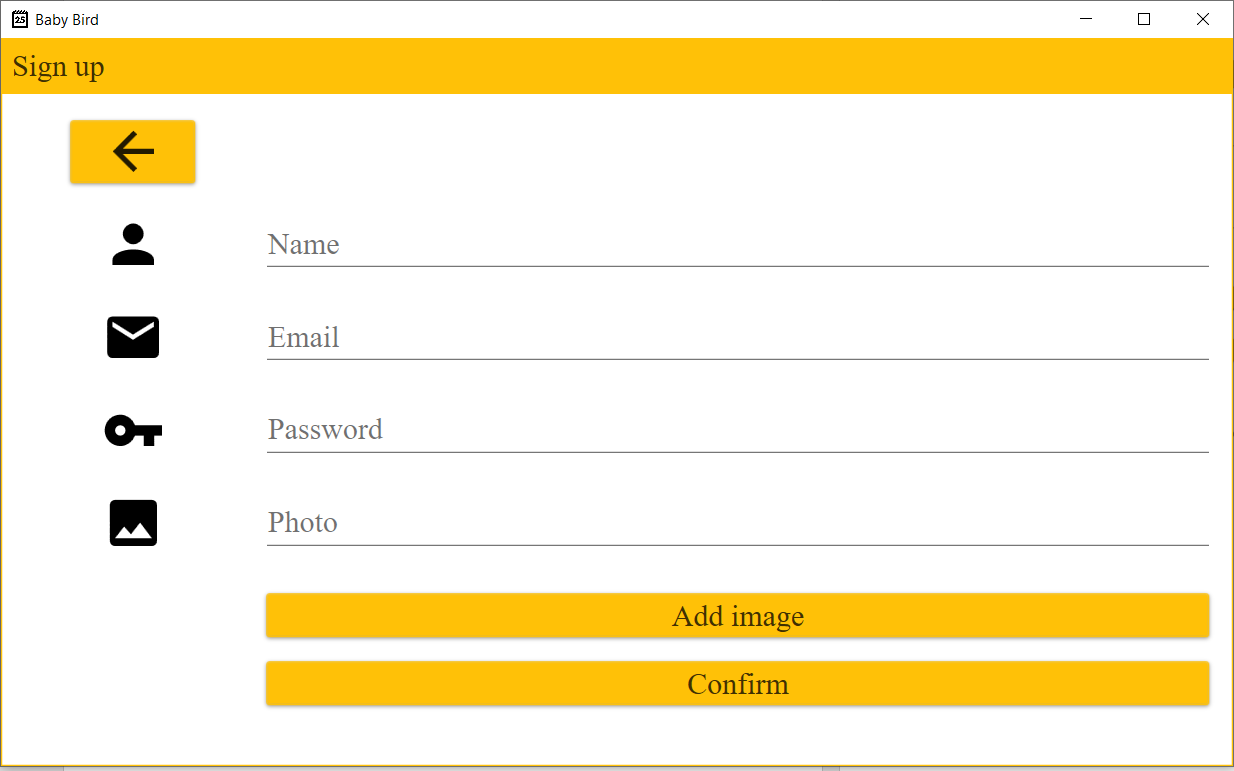


Рисунок 5.3 – Форма регистрации

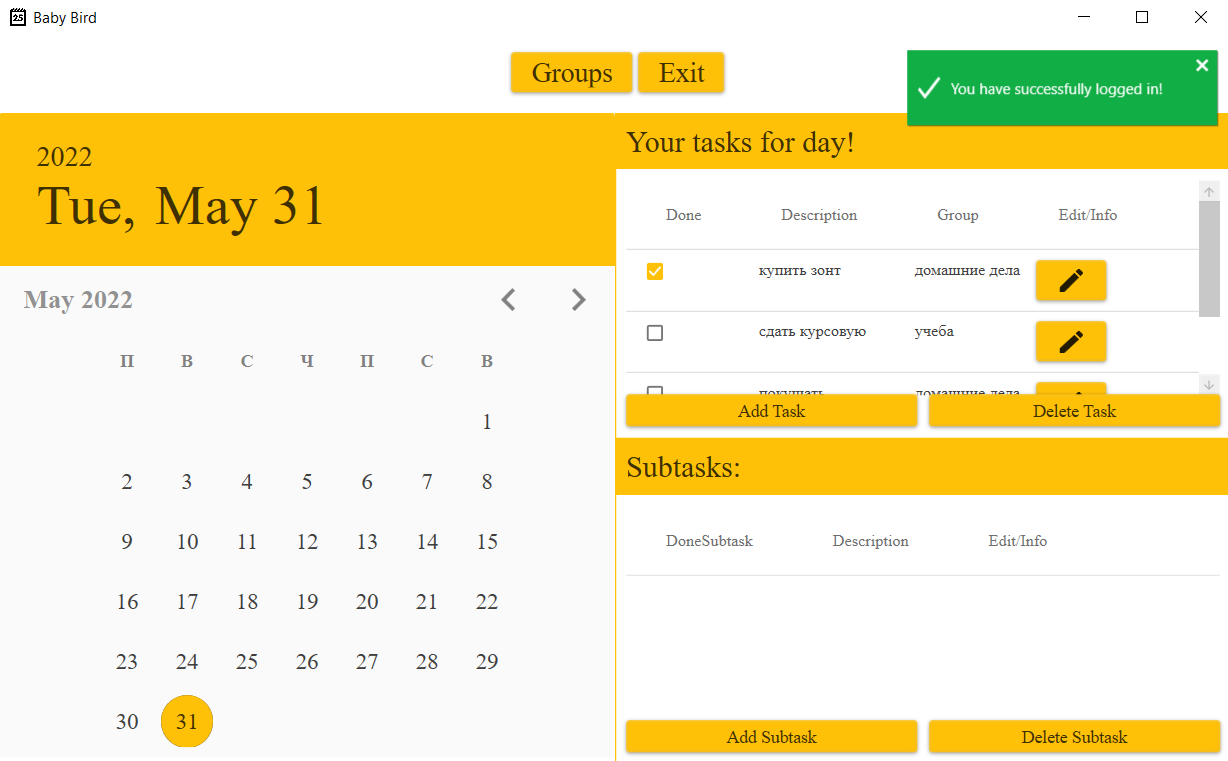


Рисунок 5.4 – Главная страница



Рисунок 5.5 – Страница добавления и редактирования задачи

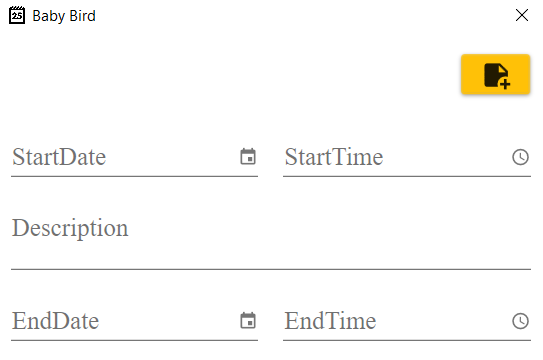


Рисунок 5.6 – Страница добавления и редактирования подзадачи

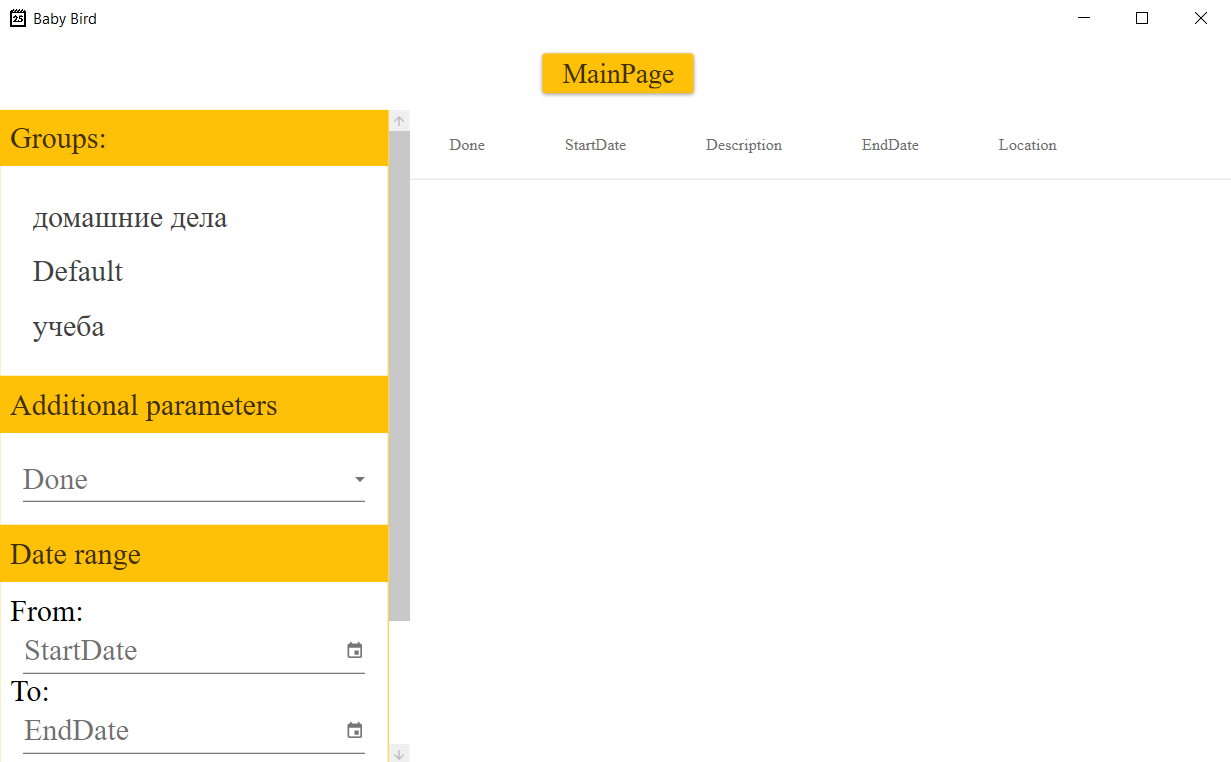


Рисунок 5.6 – Страница просмотра задач по группам

**ЗАКЛЮЧЕНИЕ**

Итогом курсовой работы стало программное средство, предназначенное для записи задач в электронном ежедневнике. В ходе выполнения был проведен тщательный анализ, который включал в себя обзор аналогов, выявление положительных и отрицательных сторон конкурентов, проверка актуальности данного ПО, постановка целей и задач, формирование технического задания.

Было проведено проектирование, включающее разработку структуры базы данных, разработку архитектуры приложения, разработка пользовательского интерфейса, разработку классов и других функциональных частей.

Благодаря подготовительным этапам получилась стабильная реализация приложения. Тщательно спроектированная база данных позволит в будущем без значительных её изменений добавить огромную функциональную составляющую, позволяющую превратить программное средство для организации жизни в полноценное приложение.

На последних этапах разработки программное средство электронный ежедневник было протестировано для выявления неточностей в работе, а также для всех категорий пользователей было предложено руководство по пользованию.

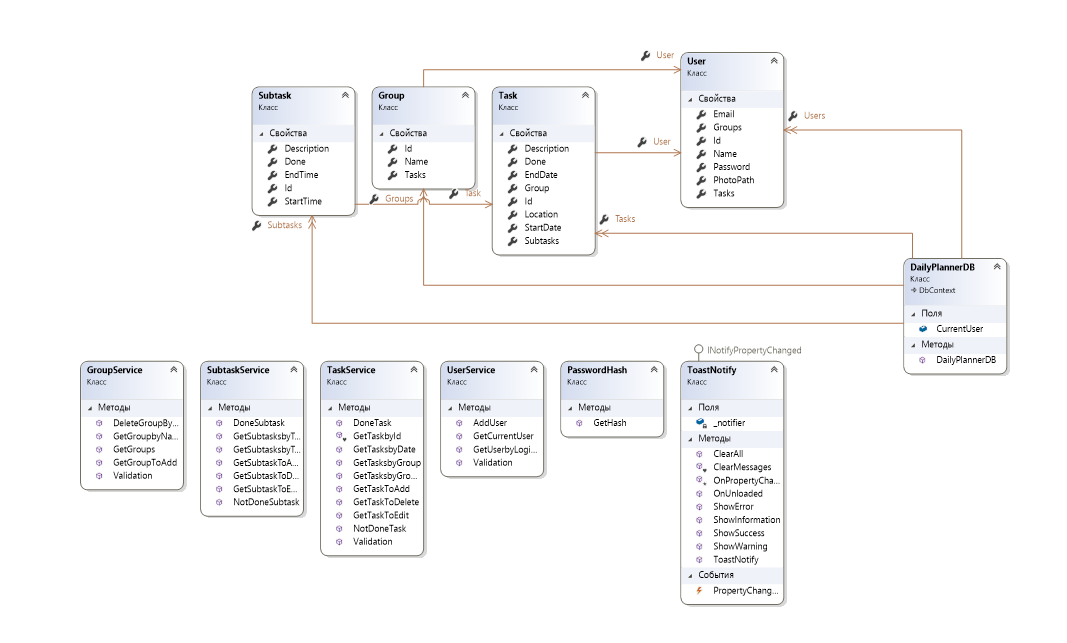
Для выполнения курсовой работы одним из важнейших этапов стало изучение материала по таким направлениям, как «Объектно-ориентированное программирование на языке С#» и «Реляционные базы данных SQL на основе SQL Server».

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

# **Список использованной литературы**

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com. Дата доступа: 23.04.2021
2. eduneo.ru [Электронный ресурс] / Режим доступа: <https://www.eduneo.ru/7test/>. Дата доступа 18.04.2021
3. Блинова Е.А. Курс лекций по базам данных / Е.А. Блинова
4. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> – Дата доступа: 10.04.2021.

# **Приложение А**



# **Приложение Б**

MainPage.xaml

|  |
| --- |
| <Page x:Class="DailyPlanner.MainPage"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"  xmlns:local="clr-namespace:DailyPlanner"  mc:Ignorable="d"  d:DesignHeight="620" d:DesignWidth="900"  Height="auto"  Width="auto"  FontFamily="Times New Roman"  FontSize="24"  Title="MainPage">  <Page.Resources>  <DataTemplate x:Key="FourthCellTasks">  <StackPanel Orientation="Horizontal">  <Button x:Name="EditTask" Click="EditTask\_Click">  <materialDesign:PackIcon Kind="Edit"  Width="auto" Height="auto"  Grid.Column="0" Grid.Row="0"  HorizontalAlignment="Center"  VerticalAlignment="Center"/>  </Button>  </StackPanel>  </DataTemplate>  <DataTemplate x:Key="ThirdthCellSubtasks">  <StackPanel Orientation="Horizontal">  <Button x:Name="EditSubtask" Click="EditSubtask\_Click">  <materialDesign:PackIcon Kind="Edit"  Width="auto" Height="auto"  Grid.Column="0" Grid.Row="0"  HorizontalAlignment="Center"  VerticalAlignment="Center"/>  </Button>  </StackPanel>  </DataTemplate>  </Page.Resources>  <Grid>  <Grid.RowDefinitions>  <RowDefinition Height="1\*" />  <RowDefinition Height="9\*" />  </Grid.RowDefinitions>  <StackPanel Grid.Row="0" Orientation="Horizontal"  HorizontalAlignment="Center" VerticalAlignment="Center">  <Button x:Name="bt\_GroupsPage" Content="Groups" Margin="5,0,0,0" FontSize="22" Click="bt\_GroupsPage\_Click"/>  <Button x:Name="bt\_SignInPage" Content="Exit" Margin="5,0,0,0" FontSize="22" Click="bt\_SignInPage\_Click"/>  </StackPanel>  <Grid Background="white" Margin="0" Grid.Row="1">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="1\*" />  <ColumnDefinition Width="1\*" />  </Grid.ColumnDefinitions>  <Calendar x:Name="calendar" Grid.Column="0"  Margin="0"  Width="auto"  Height="auto"  IsTodayHighlighted="True" SelectionMode="SingleDate"  SelectedDatesChanged="calendar\_SelectedDatesChanged">  <Calendar.LayoutTransform>  <TransformGroup>  <ScaleTransform ScaleX="1.5" ScaleY="1.5"/>  <SkewTransform/>  <RotateTransform/>  <TranslateTransform/>  </TransformGroup>  </Calendar.LayoutTransform>  </Calendar>  <Grid Grid.Column="1" Margin="0,3.5,0,0">  <Grid.RowDefinitions>  <RowDefinition Height="1\*" />  <RowDefinition Height="1\*" />  </Grid.RowDefinitions>  <GroupBox Header="Your tasks for day!"  Grid.Row="0"  FontSize="24">  <Grid Margin="0">  <Grid.RowDefinitions>  <RowDefinition Height="8.7\*" />  <RowDefinition Height="1.3\*" />  </Grid.RowDefinitions>  <ListView x:Name="lv\_ListTasks"  Margin="0" Grid.Row="0"  Height="auto" SelectionChanged="lv\_ListTasks\_SelectionChanged">  <ListView.View>  <GridView>  <GridViewColumn Header="Done" Width="auto">  <GridViewColumn.CellTemplate>  <DataTemplate>  <CheckBox IsChecked="{Binding Done, Mode=TwoWay}" Checked="cb\_Done\_Checked" Unchecked="cb\_NotDone\_Unchecked"/>  </DataTemplate>  </GridViewColumn.CellTemplate>  </GridViewColumn>  <GridViewColumn DisplayMemberBinding="{Binding Description}" Width="auto" Header="Description" />  <GridViewColumn DisplayMemberBinding="{Binding Group.Name}" Width="auto" Header="Group" />  <GridViewColumn Header="Edit/Info" Width="auto" CellTemplate="{StaticResource FourthCellTasks}" />  </GridView>  </ListView.View>  </ListView>  <Grid Grid.Row="1">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="1\*" />  <ColumnDefinition Width="1\*" />  </Grid.ColumnDefinitions>  <Button x:Name="bt\_AddTask" Height="auto" Width="auto" Grid.Column="0"  Content="Add Task"  Click="bt\_AddTask\_Click" Margin="0,0,5,0"/>  <Button x:Name="bt\_DeleteTask" Height="auto" Width="auto" Grid.Column="1"  Content="Delete Task"  Click="bt\_DeleteTask\_Click" Margin="5,0,0,0"/>  </Grid>  </Grid>  </GroupBox>  <GroupBox Header="Subtasks:"  Grid.Row="1"  FontSize="24">  <Grid Margin="0">  <Grid.RowDefinitions>  <RowDefinition Height="8.7\*" />  <RowDefinition Height="1.3\*" />  </Grid.RowDefinitions>  <ListView x:Name="lv\_ListSubtasks"  Margin="0" Grid.Row="0"  Height="auto">  <ListView.View>  <GridView>  <GridViewColumn Header="DoneSubtask" Width="auto">  <GridViewColumn.CellTemplate>  <DataTemplate>  <CheckBox IsChecked="{Binding Done, Mode=TwoWay}" Checked="cb\_DoneSubtask\_Checked" Unchecked="cb\_NotDoneSubtask\_Unchecked"/>  </DataTemplate>  </GridViewColumn.CellTemplate>  </GridViewColumn>  <GridViewColumn DisplayMemberBinding="{Binding Description}" Header="Description"/>  <GridViewColumn Header="Edit/Info" Width="auto" CellTemplate="{StaticResource ThirdthCellSubtasks}" />  </GridView>  </ListView.View>  </ListView>  <Grid Grid.Row="1">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="1\*" />  <ColumnDefinition Width="1\*" />  </Grid.ColumnDefinitions>  <Button x:Name="bt\_AddSubtaskask" Height="auto" Width="auto" Grid.Column="0"  Content="Add Subtask"  Click="bt\_AddSubtask\_Click" Margin="0,0,5,0"/>  <Button x:Name="bt\_DeleteSubtask" Height="auto" Width="auto" Grid.Column="1"  Content="Delete Subtask"  Click="bt\_DeleteSubtask\_Click" Margin="5,0,0,0"/>  </Grid>  </Grid>  </GroupBox>  </Grid>  </Grid>  </Grid>  </Page> |

# **Приложение В**

AddTask.xaml

|  |
| --- |
| <Window x:Class="DailyPlanner.AddTask"  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"  xmlns:d="http://schemas.microsoft.com/expression/blend/2008"  xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"  xmlns:local="clr-namespace:DailyPlanner"  xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"  mc:Ignorable="d"  Title="Baby Bird"  Height="340" Width="470"  FontFamily="Times New Roman"  FontSize="18"  Icon="Images/icon.png"  ResizeMode="NoResize">  <Grid Background="white">  <Grid.ColumnDefinitions>  <ColumnDefinition Width="1\*" />  <ColumnDefinition Width="1\*" />  </Grid.ColumnDefinitions>  <Grid.RowDefinitions>  <RowDefinition Height="1\*" />  <RowDefinition Height="1\*" />  <RowDefinition Height="1\*" />  <RowDefinition Height="1\*" />  <RowDefinition Height="1\*" />  <RowDefinition Height="1\*" />  </Grid.RowDefinitions>  <DatePicker Grid.Row="1" Grid.Column="0"  Margin="10,8" x:Name="dp\_StartDate"  materialDesign:HintAssist.Hint="StartDate" />  <materialDesign:TimePicker x:Name="tp\_StartTime" Grid.Row="1" Grid.Column="1"  materialDesign:HintAssist.Hint="StartTime" Margin="10,8" />  <Button x:Name="bt\_AddTask" Grid.Row="0" Width="55" Grid.ColumnSpan="2"  HorizontalAlignment="Right" Margin="0,0,10,0"  Click="bt\_AddTask\_Click">  <materialDesign:PackIcon Kind="NotePlus" Width="25" Height="25"  HorizontalAlignment="Center" VerticalAlignment="Center"/>  </Button>  <TextBox Grid.Row="2" Margin="10,8" x:Name="tb\_Description"  Grid.ColumnSpan="2"  materialDesign:HintAssist.Hint="Description" />  <ComboBox Grid.Row="3" Margin="10,8"  Grid.ColumnSpan="2"  Name="cb\_Group"  StaysOpenOnEdit="True"  materialDesign:HintAssist.Hint="Group"  IsEditable="True"  TextSearch.TextPath="Name">  <ComboBox.ItemTemplate>  <DataTemplate>  <StackPanel>  <TextBlock x:Name="tb\_Group" Text="{Binding Name}" />  </StackPanel>  </DataTemplate>  </ComboBox.ItemTemplate>  </ComboBox>  <TextBox Grid.Row="4" Margin="10,8" x:Name="tb\_Location" Grid.ColumnSpan="2"  materialDesign:HintAssist.Hint="Location" />  <DatePicker Grid.Row="5" Margin="10,8" x:Name="dp\_EndDate" Grid.Column="0"  materialDesign:HintAssist.Hint="EndDate" />  <materialDesign:TimePicker x:Name="tp\_EndTime" Grid.Row="5" Grid.Column="1"  materialDesign:HintAssist.Hint="EndTime" Margin="10,8" />  </Grid>  </Window> |