**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

**«Сибирский государственный университет науки и технологий  
имени академика М.Ф. Решетнева»**

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

**КУРСОВАЯ РАБОТА**

Объектно-ориентированное программирование

|  |
| --- |
| Разработка программы «Ежедневник» |
|  |

Руководитель В.В.Вдовенко

подпись, дата инициалы, фамилия

Обучающийся БПИ20-02, 201219047 Р.А. Сухачев

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2022 г.

Институт информатики и телекоммуникаций

Кафедра информатики и вычислительной техники

|  |  |  |  |
| --- | --- | --- | --- |
| **ЗАДАНИЕ** | | | |
| на курсовую работу по дисциплине Объектно-ориентированное программирование | | | |
| Обучающемуся *Сухачеву Роману Андреевичу* | | | |
| Группа *БПИ20-02* |  | Форма обучения *очная* |  |
| Тема работы: Разработка программы «Ежедневник» | | | |
| Срок сдачи курсовой работы | | | |
| Перечень вопросов, подлежащих разработке при написании теоретической части: | | | |
| 1. Системный анализ предметной области. | | | |
| 1. Объектно-ориентированное моделирование программы: | | | |
| 2.1.Разработка UML-диаграммы прецедентов | | | |
| * 1. Разработка UML-диаграммы активности | | | |
| 3.Разработка архитектуры программы | | | |
|  | | | |
|  | | | |
| Перечень вопросов, подлежащих разработке при написании практической части (либо указать номер варианта задания) | | | |
| 1. Разработка кода программы | | | |
| \_\_2. Тестирование программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_3. Разработка руководства программиста\_\_\_\_\_\_\_\_\_\_\_\_\_  \_\_4. Разработка руководства пользователя\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
|  | | | |
| Дата выдачи задания: | | | |
| Руководитель *Вдовенко В.В., доцент кафедры ИВТ* | | | |
| (подпись) | | | |
|  | | | |
| Задание принял к исполнению (дата) \_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
| (подпись обучающегося, И.О. Фамилия) | | | |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc6175)

[1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРОГРАММЫ “ЕЖЕДНЕВНИК” 5](#_Toc26030)

[1.1 Функции Ежедневника 5](#_Toc8972)

[1.2 Обзор существующего программного обеспечения 5](#_Toc9788)

[1.3 Выводы по главе 10](#_Toc8830)

[2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА 11](#_Toc13620)

[2.1 Структура программного продукта 11](#_Toc23502)

[2.2 Диаграммы активности и прецедентов 11](#_Toc11320)

[2.3 Руководство программиста 13](#_Toc3331)

[2.4 Краткое руководство пользователя 14](#_Toc22652)

[2.5 Тестирование программного продукта 14](#_Toc16571)

[2.6 Выводы по главе 15](#_Toc11119)

[ЗАКЛЮЧЕНИЕ 16](#_Toc13996)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 17](#_Toc20949)

[ПРИЛОЖЕНИЕ А ЛИСТИНГ ПРОГРАММЫ 18](#_Toc11501)

# ВВЕДЕНИЕ

*Актуальность*. На сегодняшний день у человечества множество мыслей и важных дел, которые трудно держать в голове. Ранее, человеку приходилось использовать записные книжки или «ежедневники», но сейчас, когда под рукой всегда есть мобильный телефон, ноутбук или персональный компьютер появляется возможность реализации мгновенной записи заметок, важных дел или простых мыслей в определённую программу, куда в любой момент можно зайти и увидеть свои записи. Таким образом, разработка и улучшение программ для хранения информации является актуальной задачей.

*Цель и задачи.* Целью курсовой работы является разработка программы «Ежедневник» .

Для достижения поставленной цели необходимо решить следующие задачи:

* изучить программное обеспечение с подобным функционалом;
* изучить средства разработки программного продукта;
* выполнить объектно-ориентированное моделирование программы;
* разработать структуру программного продукта;
* программно реализовать функциональную оболочку разрабатываемого программного обеспечения;
* создать диаграммы активности и прецедентов;
* провести тестирование разработанного программного продукта;
* разработать руководство программиста и пользователя.

*Структура работы.* Пояснительная записка к курсовой работе состоит из введения, двух глав, заключения и списка используемых источников. Изложена на 22 страницах, содержит 1 таблицу и 7 рисунков.

В первой главе курсовой работы приводится теоретическая часть курсовой работой, в ней отражены анализ существующего программного обеспечения с подобным функционалом и анализ существующих программных продуктов для разработки.

Во второй главе приводится структура разработанного обеспечения, показан процесс реализации программного продукта, созданы диаграммы активности и прецедентов.

В заключении описаны результаты теоретического изучения предметной области и практической работы над программным обеспечением.

# 1 ОБЩАЯ ХАРАКТЕРИСТИКА ПРОГРАММЫ “ЕЖЕДНЕВНИК”

## Функции Ежедневника

Ежедневник – это тетрадь для ведения ежедневных записок. В современном понимании – деловой (обыкновенно) блокнот, предназначенный для ежедневного ведения записей о делах, планах, результатах и прочих записях, связанных с ежедневной деловой деятельностью человека или компании/организации. Наиболее популярными видами являются ежедневники в виде книги, в различном переплёте. Несмотря на все разнообразие − самым популярным остаётся ежедневник в виде книги в благородном твёрдом переплёте, выполненным в мягкой коже, как правило снабжённый дополнительной информацией для комфортной работы его владельца (коды телефонной связи, таблицы расстояний между местными областными центрами, конвертация единиц измерения, государственные и международные праздники, и т. п.), а также (только для датированных ежедневников) подготовленный для ведения ежегодного, ежемесячного, еженедельного и ежедневного расписания. Что, впрочем, не мешает существованию недатированных ежедневников, даты и события, в которых проставляет сам владелец.

В настоящее время, на замену физическому ведению заметок и записей появился цифровой вид записи ежедневных дел. Появилось много программ «Ежедневников» на любой выбор, которые доступны на телефоны, планшеты, ноутбуки и компьютеры. Теперь, чтобы записать важное дело или мысль вам лишь нужен ваш гаджет, который всегда под рукой. Также преимущества цифрового «Ежедневника» в том, что вся нумерация, даты и все остальное будет проставляться автоматически.

## 1.2 Обзор существующего программного обеспечения

На сегодняшний день существует ряд разработанных решений задачи автоматизации складского учёта.

***Todoist*** – одно из самых популярных приложений-ежедневников среди индивидуальных предпринимателей, менеджеров команд и крупных компаний [1]. Это помогает вам организовывать свои задачи и управлять ими. *Todoist* интегрируется с множеством различных приложений, что позволяет более эффективно управлять рабочими процессами и задачами. Обладая рядом преимуществ, это программное обеспечение отвечает требованиям хорошо организованной системы управления. Он включает в себя такие функции, как подзадачи, подпроекты, повторяющиеся задачи, предупреждения и многое другое. *Todoist* – отличный способ всегда оставаться в курсе своих задач и быть продуктивным. Это простое в использовании приложение со списком дел, которое позволяет вам управлять своими задачами и делиться ими с коллегами. Главное окно программы представлено на рисунке 1.

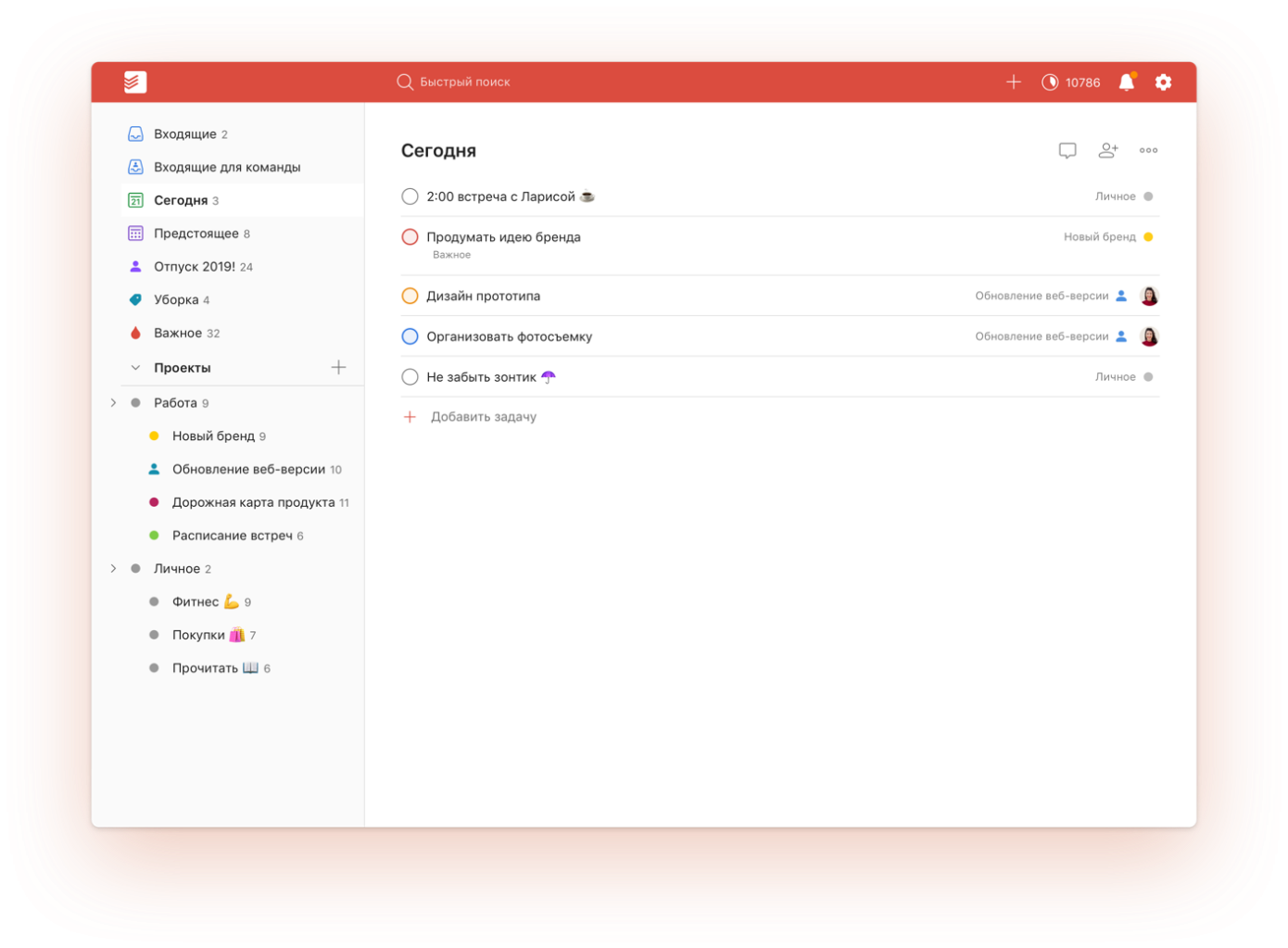


Рисунок 1 − Главное окно программы «*Todoist*»

Основные возможности:

* Приоритизация задач − это поможет вам выполнить самые важные задачи.
* Управление задачами − эта функция помогает вам создавать подзадачи, указывать повторяющиеся задачи, включать сроки выполнения отдельных задач очень просто. Он также помечает задачи в соответствии с порядком их приоритета и устанавливает напоминания, чтобы держать вас под контролем.
* Управление прогрессом − эта функция отслеживает прогресс в использовании функций измерения и отчётности приложения. Это позволяет вам следить за действиями и принимать меры в соответствии с желаемым результатом.
* Управление проектом − он разработан, чтобы помочь вам установить цели проекта и помочь вам не сбиться с пути.

Можно отметить такие недостатки этого программного продукта, как проблемы с организацией подзадач и подпроектов, нет встроенного календаря, технические трудности с переносом задач.

***Serene*** − это программа для macOS, которая даёт вам сверхспособности продуктивности, сочетая три проверенные стратегии для повышения концентрации внимания [2]. Функция блокировки отвлекающих факторов помогает вам лучше сосредоточиться, чтобы добиться большего за меньшее время. Он разработан для того, чтобы вы погрузились в глубокую работу и быстрее достигали целей. Приложение *Serene* не только помогает вам организовать работу, но и помогает вам сосредоточиться на глубокой работе. Это приложение для повышения производительности предназначено для предпринимателей, маркетологов, разработчиков, фрилансеров, дизайнеров, писателей и т. д. Главное окно программы представлено на рисунке 2.



Рисунок 2 − Главное окно программы «Serene»

Основные возможности:

* Таймер сеанса фокусировки − чётко определённые интервалы целенаправленной работы с таймером, который показывает вам время. Знание того, сколько времени у вас осталось на выполнение задач, даёт вам дополнительную мотивацию.
* Блокировщик веб-сайтов и приложений − это помогает блокировать доступ к веб-сайтам, чтобы вы могли сосредоточиться. Знание приложений, которые сильно отвлекают, поможет вам сосредоточиться. Сохранение продуктивности во время фокус-сессий позволяет не отвлекаться от задач без лишних перерывов.
* Трекер активности − чёткое представление о вашем продуктивном времени – отличный способ отслеживать свою активность. Знание того, что вас обычно отвлекает, является ключевым элементом в улучшении привычек.
* Ежедневный план − функция ежедневного планирования помогает вам планировать в соответствии с одной основной целью – отличный способ оставаться организованным. Это помогает делать дела с более ясным намерением и сосредоточенным вниманием.
* Музыка в фокусе − с помощью этой функции вы можете воспроизводить фоновую музыку, чтобы сохранять сосредоточенность и мотивацию.
* Глушитель телефона − он переводит ваш телефон в беззвучный режим, автоматически устраняя ненужные отвлекающие факторы.

***Any.do*** – это приложение для ежедневного планирования, которое поможет вам организовать ваши задачи, списки и напоминания и легко синхронизировать их на всех ваших устройствах. Известный своей высокой эффективностью в офисе, *Any.do*отслеживает ваш прогресс. Главное окно программы представлено на рисунке 3.

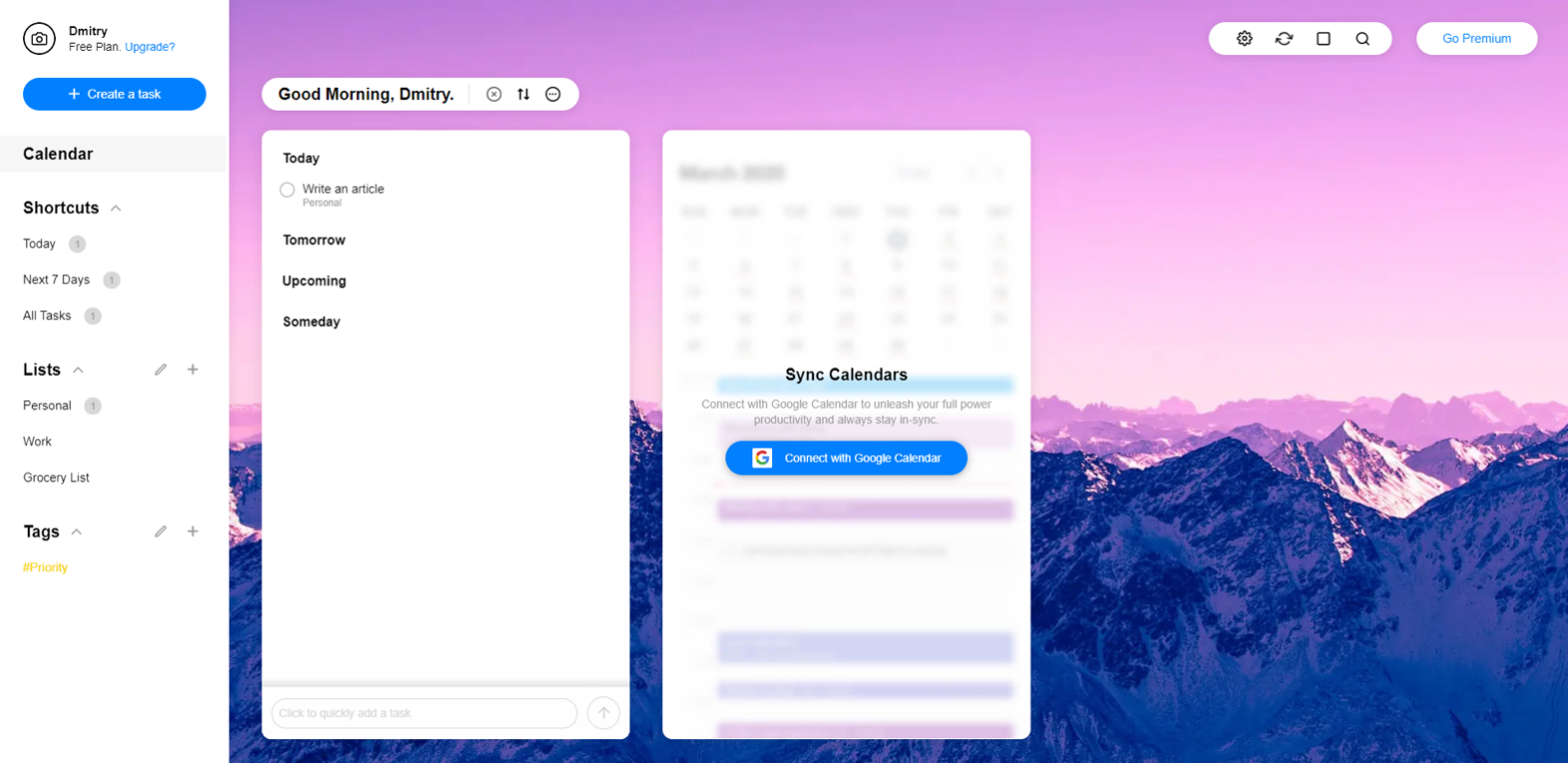


Рисунок 3 − Главное окно программы «Any.do»

Основные возможности:

* Синхронизация данных – Действие или элемент, добавленные на *Any.do*, автоматически синхронизируются на нескольких платформах, таких как Календарь *Google* и напоминания *WhatsApp*, чтобы вы были эффективны.
* Автономный доступ – С *Any.do* вы можете получить доступ к своему инструменту в любом месте и в любое время. Это даёт вам большую свободу, душевное спокойствие и возможность всегда быть в курсе дел.
* Управление задачами – Эта функция объединяет ваш список дел с календарём, чтобы упростить планирование и составление расписания.
* Запись голоса – Этот голосовой захват позволяет отправлять инструкции, даже не нажимая на клавиатуру.

Результаты сравнения рассмотренных программных продуктов представлены в таблице 1.

Таблица 1 – Сравнение программных продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристики | «*Todoist*» | «*Serene*» | **«*Any.do*»** |
| Клиентская ОС | *iOS, Android, Windows*,  *macOS* | *MacOS* | *iOS, Android, Windows*,  *macOS* |
| Встроенный календарь | Нет | Да | Да |
| Синхронизация данных | Нет | Нет | Да |

Продолжение таблицы 1 – Сравнение программных продуктов

|  |  |  |  |
| --- | --- | --- | --- |
| Управление прогрессом | Да | Да | Нет |
| Обязательная платная система | Нет | Да | Нет |

## 1.3 Выводы по главе

В данной главе курсовой работы были рассмотрены наиболее близкие конкуренты по функционалу. Были выявлены основные достоинства и недостатки каждого из них. Также были рассмотрены функции ежедневника.

# 2 РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА

Разрабатываемый программный продукт представляет собой приложение “Ежедневник”.

## 2.1 Структура программного продукта

*UML*-диаграмму классов приложения можно увидеть на рисунке 4.

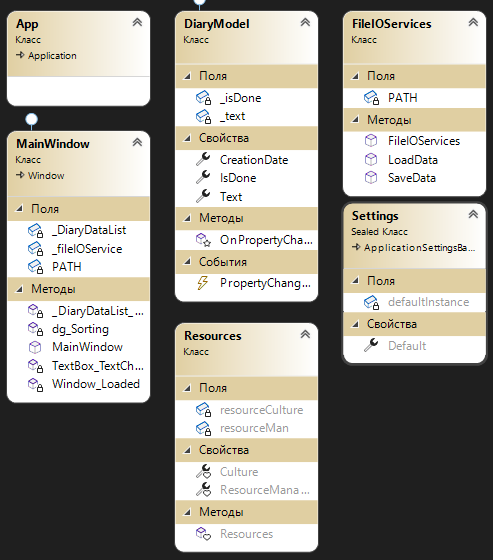


Рисунок 4 – Диаграмма классов

## 2.2 Диаграммы прецедентов и активности

Были разработаны диаграмма прецедентов (Рисунок 5) и диаграмма активностей (Рисунок 6).

Диаграмма прецедентов, позволяют визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой.

Диаграмма активности позволяет более детально визуализировать конкретный случай использования.

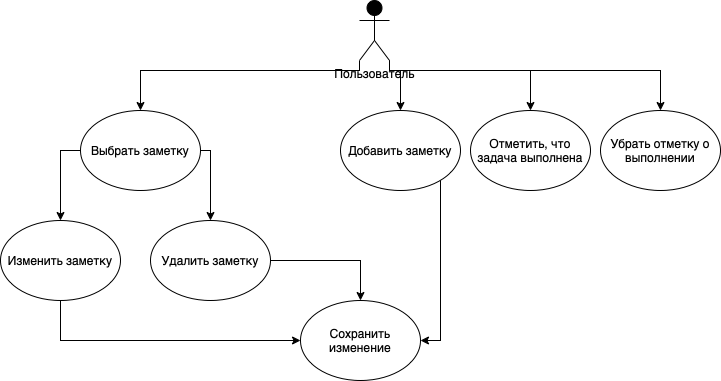


Рисунок 5 – Диаграмма прецедентов

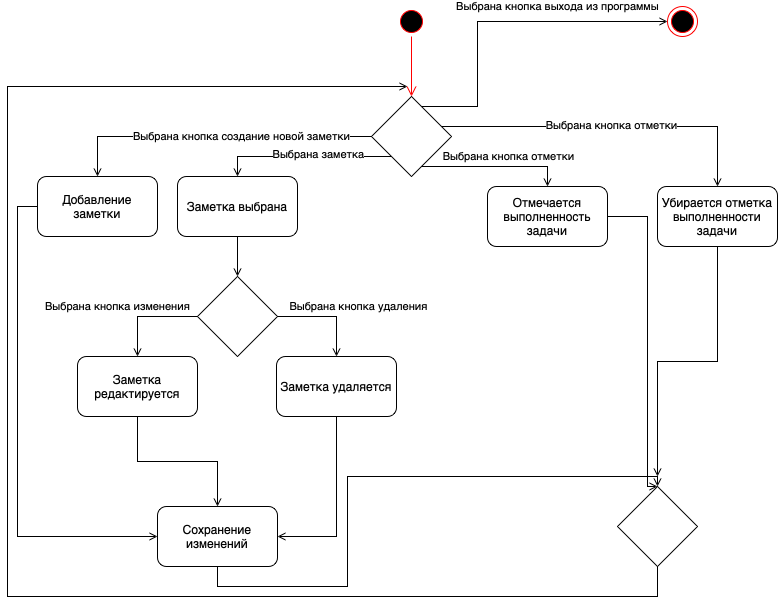


Рисунок 6 – Диаграмма активностей

## 2.3 Руководство программиста

Для запуска проекта необходимо установить:

* *Microsoft Visual Studio* 2022 *v*17.2.3 или выше;

Для работы программы необходима операционная система *Windows.*

Программа состоит из классов

*MainWindow:*

1. Поля класса:

* *\_DiaryDataLis*t – объект класса *BindingList<DiaryModel>*
* *\_fileIOService* – объект класса *FileIOServices*
* *PATH* – строка, содержащая путь файлу *json*

1. Функции:

* *\_DiaryDataList\_ListChanged* – сохраняет данные при их изменении
* *dg\_Sorting* – сортировка
* *TextBox\_TextChanged* – фильтрование данных в реальном времени
* *Window\_Loaded* – при запуске, подключение файла *json* с данными

DiaryModel*:*

1. Поля класса:

* *\_IsDone* – приватное булевое значение
* *\_text* – приватная строка

1. Свойства:

* *CreationDate* – объект класса *DateTime*, содержащий дату создания
* *IsDone –* булевое значение
* *Text –* строка

1. Функции:

* *OnPropertyChanged* – вызывается, при изменении данных

*FileIOServices:*

1. Поля класса:

* *PATH* – строка, содержащая путь файлу *json*

1. Функции:

* *\_FileIOServices* – присваивает путь в *PATH*
* *LoadData* – загрузка данных из *json*
* *SaveData* – сохранение данных

## 2.4 Краткое руководство пользователя

После запуска программы появляется окно программы (Рисунок 7).

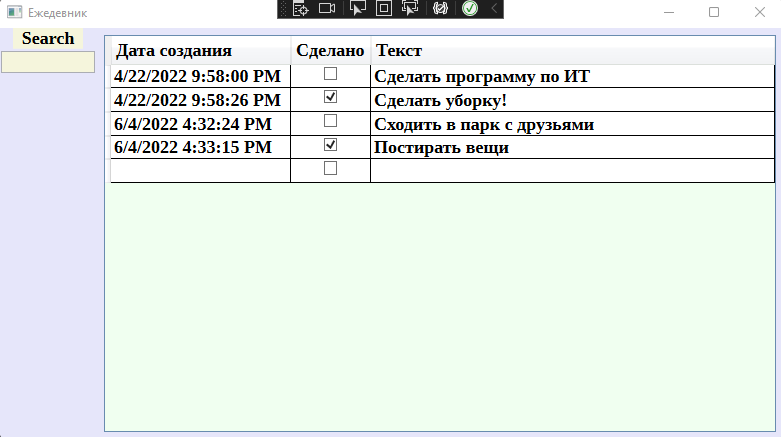


Рисунок 7 – Окно программы

Для того чтобы добавить новую заметку, пользователю необходимо нажать на пустое поле текста, что автоматически создаст новую заметку, которую можно будет заполнить. Чтобы отметить, что задача была выполнена, пользователю необходимо нажать на квадратик рядом с выбранной заметкой и в нём появится отметка о выполнении. Чтобы убрать отметку о выполнении – необходимо повторно нажать на квадратик. Для сортировки, пользователю необходимо нажать на заголовок столбца. Для поиска, пользователю необходимо вписать в строку “*Search*” текст, который содержится в требуемой заметке. Для удаления заметки, пользователю необходимо выбрать заметку и нажать кнопку “*Delete*”.

## 2.5 Тестирование программного продукта

В данном разделе протестированы различные действия пользователя.

Далее будут протестированы все возможные действия пользователя.

**Добавление новой заметки**

При нажатии на пустое поле текста – автоматически создаёт новую пустую заметку с текущим временем.

**Отметить выполненность задачи**

При нажатии на квадратик рядом с выбранной заметкой – в нём появляется галочка, которая показывает, что отметка выполнена.

**Убрать отметку о выполнении**

Повторное нажатие на квадратик – галочка исчезла.

**Сортировка**

При нажатии на заголовок столбца – происходит сортировка по данным этого столбца.

**Поиск**

При написании в строке “*Search*” текста – выводит заметки, в которых содержится этот текст.

**Удаление**

Выбрав заметку и нажав кнопку “*Delete*” на клавиатуре – заметка удаляется.

## 2.6 Выводы по главе

В данной главе были реализованы бизнес-правила функционирования разрабатываемого программного обеспечения, представляющие собой сценарии его использования и чёткое описание происходящего в программе при определённых действиях пользователя.

На следующем этапе был разработан сам программный продукт. Были написаны руководства программиста и пользователя, позволяющие использовать разработанное программное обеспечение в своих целях.

Кроме того, было проведено тестирование разработанного программного обеспечения, в ходе которого было показано корректное функционирование ПО на всех этапах работы.

# ЗАКЛЮЧЕНИЕ

В результате данной курсовой работы была создано приложение “Ежедневник”.

В курсовой работе были решены следующие задачи:

– проделана работа по изучению предметной области;

– найдены сильные и слабые стороны продуктов, похожих по функционалу;

– описана краткая история и функции ежедневников;

– выполнено объектно-ориентированное моделирование программного продукта: разработаны UML-диаграммы классов, прецедентов и активностей;

– выполнено кодирование и тестирование программного продукта “Ежедневник”;

– разработаны руководства программиста и пользователя.

Разработанный “Ежедневник” может использоваться как приложения для ведения заметок пользователем на компьютере. Программа обладает простым и наглядным интерфейсом и не требует дополнительного обучения пользователя.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. World scholarship forum [Электронный ресурс] − URL: <https://worldscholarshipforum.com/ru/работы/лучшее-приложение-для-ежедневного-планирования/> – Текст: электронный (Дата обращения: 12.03.2022)
2. Вдовенко, В. В. Объектно-ориентированное программирование: электронный образовательный ресурс для студентов направления 09.03.02 «Информационные системы и технологии» / В. В. Вдовенко ; Сиб. гос. ун-т науки и технологий. ― Красноярск: СибГУ им. М.Ф. Решетнева, 2022. ― Текст: электронный // Сервер электронно-дистанционного обучения: электронная образовательная среда СибГУ им. М.Ф. Решетнева. – *URL*: <https://dl.sibsau.ru/course/view.php?id=3550>

# ПРИЛОЖЕНИЕ А ЛИСТИНГ ПРОГРАММЫ

**MainWindow.xaml.cs**

using Diary.Models;

using Diary.Services;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Diary

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

private readonly string PATH = $"{Environment.CurrentDirectory}\\DiaryDataList.json";

private BindingList<DiaryModel> \_DiaryDataList;

private FileIOServices \_fileIOService;

public MainWindow()

{

InitializeComponent();

}

private void Window\_Loaded(object sender, RoutedEventArgs e)

{

\_fileIOService = new FileIOServices(PATH);

try

{

\_DiaryDataList = \_fileIOService.LoadData();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

Close();

}

dgDiary.ItemsSource = \_DiaryDataList;

\_DiaryDataList.ListChanged += \_DiaryDataList\_ListChanged;

}

private void \_DiaryDataList\_ListChanged(object sender, ListChangedEventArgs e)

{

if (e.ListChangedType == ListChangedType.ItemAdded || e.ListChangedType == ListChangedType.ItemDeleted || e.ListChangedType == ListChangedType.ItemChanged)

{

try

{

\_fileIOService.SaveData(sender);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

Close();

}

}

}

private void TextBox\_TextChanged(object sender, TextChangedEventArgs e)

{

var tbx = (TextBox)sender;

if (tbx.Text != "")

{

var filteredList = \_DiaryDataList.Where(x => x.Text.ToLower().Contains(tbx.Text.ToLower()));

dgDiary.ItemsSource = null;

dgDiary.ItemsSource = filteredList;

}

else

{

dgDiary.ItemsSource = \_DiaryDataList;

}

}

private void dg\_Sorting(object sender, DataGridColumnEventArgs e)

{

if (e.Column.Header.ToString() == "Дата создания")

{

if (e.Column.SortDirection == null || e.Column.SortDirection == ListSortDirection.Descending)

{

dgDiary.ItemsSource = new ObservableCollection<DiaryModel>(from item in \_DiaryDataList orderby item.CreationDate ascending select item);

e.Column.SortDirection = ListSortDirection.Ascending;

}

else

{

dgDiary.ItemsSource = new ObservableCollection<DiaryModel>(from item in \_DiaryDataList orderby item.CreationDate descending select item);

e.Column.SortDirection = ListSortDirection.Descending;

}

}

else if (e.Column.Header.ToString() == "Сделано")

{

if (e.Column.SortDirection == null || e.Column.SortDirection == ListSortDirection.Descending)

{

dgDiary.ItemsSource = new ObservableCollection<DiaryModel>(from item in \_DiaryDataList orderby item.IsDone ascending select item);

e.Column.SortDirection = ListSortDirection.Ascending;

}

else

{

dgDiary.ItemsSource = new ObservableCollection<DiaryModel>(from item in \_DiaryDataList orderby item.IsDone descending select item);

e.Column.SortDirection = ListSortDirection.Descending;

}

}

else if (e.Column.Header.ToString() == "Текст")

{

if (e.Column.SortDirection == null || e.Column.SortDirection == ListSortDirection.Descending)

{

dgDiary.ItemsSource = new ObservableCollection<DiaryModel>(from item in \_DiaryDataList orderby item.Text ascending select item);

e.Column.SortDirection = ListSortDirection.Ascending;

}

else

{

dgDiary.ItemsSource = new ObservableCollection<DiaryModel>(from item in \_DiaryDataList orderby item.Text descending select item);

e.Column.SortDirection = ListSortDirection.Descending;

}

}

}

}

}

**DiaryModel.cs**

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Diary.Models

{

internal class DiaryModel: INotifyPropertyChanged

{

private bool \_isDone;

private string \_text;

[JsonProperty(PropertyName = "creationDate")]

public DateTime CreationDate { get; set; } = DateTime.Now;

[JsonProperty(PropertyName = "isDone")]

public bool IsDone

{

get { return \_isDone; }

set

{

if (\_isDone == value)

return;

\_isDone = value;

OnPropertyChanged("IsDone");

}

}

[JsonProperty(PropertyName = "text")]

public string Text

{

get { return \_text; }

set

{

if (\_text == value)

return;

\_text = value;

OnPropertyChanged("Text");

}

}

public event PropertyChangedEventHandler PropertyChanged;

protected virtual void OnPropertyChanged(string propertyName = "")

{

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

}

}

**FileIOServices.cs**

using Diary.Models;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace Diary.Services

{

internal class FileIOServices

{

private readonly string PATH;

public FileIOServices(string path)

{

PATH = path;

}

public BindingList<DiaryModel> LoadData()

{

var fileExists = File.Exists(PATH);

if (!fileExists)

{

File.CreateText(PATH).Dispose();

return new BindingList<DiaryModel>();

}

using (var reader = File.OpenText(PATH))

{

var fileText = reader.ReadToEnd();

return JsonConvert.DeserializeObject<BindingList<DiaryModel>>(fileText);

}

}

public void SaveData(object DiaryDataList)

{

using (StreamWriter writer = File.CreateText(PATH))

{

string output = JsonConvert.SerializeObject(DiaryDataList);

writer.Write(output);

}

}

}

}