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 Программное обеспечение информационных технологий (программирование интернет-изданий)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

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

Тема Программное средство «Ежедневник»

Исполнитель

студентка 2 курса группы 4 Ерчинская Наталия Константиновна

(Ф.И.О.)

Руководитель работы ассистент Северинчик Н. А.

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

Курсовой проект защищен с оценкой

Председатель Северинчик Н. А.

(Подпись)

Минск 2021

**Содержание**

[**ВВЕДЕНИЕ** 2](#_Toc74112612)

[**1.** **Аналитический обзор литературы и постановка задачи** 3](#_Toc74112613)

[**2.** **Анализ требований к программному средству и разработка функциональных требований** 5](#_Toc74112614)

[**2.1. Определение требований к программному средству** 5](#_Toc74112615)

[**2.2. Описание функциональности программного средства** 5](#_Toc74112616)

[**2.3. Описание средств разработки** 6](#_Toc74112617)

[**2.4.** **Спецификация функциональных требований.** 6](#_Toc74112618)

[**3.** **Проектирование программного средства** 8](#_Toc74112619)

[**3.1. Архитектура системы** 8](#_Toc74112620)

[**3.2 Диаграммы UML** 9](#_Toc74112621)

[**3.3. Проектирование логической структуры базы данных** 9](#_Toc74112622)

[**3.4. Взаимодействие окон и представлений** 10](#_Toc74112623)

[**3.5. Структура проекта** 11](#_Toc74112624)

[**4.** **Реализация программного средства** 13](#_Toc74112625)

[**4.1 Реализация MVVM и других шаблонов** 13](#_Toc74112626)

[**4.2. Реализация авторизации и регистрации пользователей** 17](#_Toc74112627)

[**4.3. Добавление записок и целей** 18](#_Toc74112628)

[**5.** **Тестирование, проверка работоспособности и анализ полученных результатов** 19](#_Toc74112629)

[**6.** **Руководство по использованию** 22](#_Toc74112630)

[**6.1. Регистрация и авторизация** 22](#_Toc74112631)

[**6.2. Домашнее представление** 23](#_Toc74112632)

[**6.3. Администрирование** 26](#_Toc74112633)

[**6.3. Смена пользователя** 27](#_Toc74112634)

[**ЗАКЛЮЧЕНИЕ** 28](#_Toc74112635)

[**Список использованной литературы** 29](#_Toc74112636)

[**Приложение А** 30](#_Toc74112637)

[**Приложение Б** 31](#_Toc74112638)

[**Приложение В** 32](#_Toc74112639)

[**Приложение Г** 33](#_Toc74112640)

[**Приложение Д** 34](#_Toc74112641)

[**Приложение Е** 35](#_Toc74112642)

# **ВВЕДЕНИЕ**

В век высокоскоростных технологий успешный бизнес требует высокоскоростных решений. Объёмы задач, информации, контактов и просто обрывочных фактов превосходят все возможные рамки. Когда речь заходит о планировании дел, некоторые люди объясняют, что они привыкли «держать все свои дела в голове». Кто-то действительно обладает великолепной памятью. Однако в подавляющем большинстве случаев человек, утверждающий, что он может обходиться без ежедневника, просто лукавит или заблуждается. Если дел не много, то проблемы нет. Но как только задач становится больше – часть из них забывается или вытесняется.

Note2Self – электронный ежедневник, который подходит для эффективного планирования дел. В данной программе можно записывать как важные дела, так и анализ сделанного, например хронометраж или впечатления от задания. Планируя заранее, можно избежать неприятностей и не забыть о важных мероприятиях или встречах. Программа легка и удобна в использовании. Кроме того, в приложении Note2Self можно каждый день отмечать свое настроение.

В качестве интерфейса прикладного программирования был выбран API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом. Для работы с WPF использовался объектно-ориентированный язык программирования С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Связь между базой данных и приложением на C# осуществлена с помощью Entity Framework. Это технология работы с данными, которая основана на платформе .NET Framework. Она представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Для успешной реализации курсового проекта необходимо:

* провести анализ соответствующей литературы;
* ознакомиться с прототипами программных средств выбранной мной темы;
* определить функциональные требования;
* продумать структуру базы данных;
* продумать структуру проекта;
* реализовать программное средство;
* протестировать программное средство;
* написать руководство пользователя.

Содержание данной пояснительной записки отражает этапы выполнения курсового проекта.

1. **Аналитический обзор литературы и постановка задачи**

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.

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

Google Keep – сервис, созданный компанией Google Inc. в 2013 году, который предназначен для создания и хранения заметок.

Достоинства:

* сервис является бесплатным;
* поддержка разных языков;
* доступен в виде интернет-приложения с доступом через любой совместимый браузер, а также с мобильных устройств;
* современный и интуитивный интерфейс
* имеет только базовые функции: установка напоминания, изменение цвета заметки, вставка изображения и добавление новых листов.

Недостатки:

* нельзя иметь более одного списка в листе;
* не поддерживается форматирование.

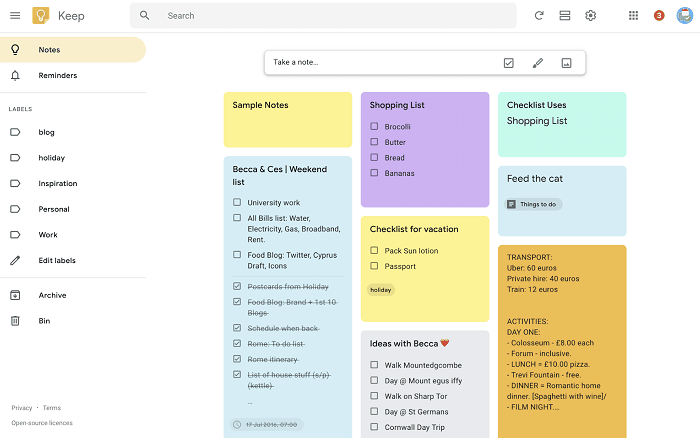


Рисунок 1.1 – Интерфейс веб-приложения Google Keep

Microsoft OneNote – программа для создания быстрых заметок и организации личной информации, разработанная компанией Microsoft.

Достоинства:

* сервис является бесплатным с марта 2014 года;
* поддержка разных языков;
* программа может использоваться на обычных настольных компьютерах с операционными системами Microsoft Windows и macOS, также доступны версии для мобильных ОС;
* OneNote предназначен для ведения коротких заметок — это особенно удобно пользователям планшетных компьютеров, где присутствует возможность рукописного ввода текста и добавления заметок;
* Можно записывать голосовые комментарии;
* информацию в OneNote можно перетаскивать мышью из окна обозревателя;
* имеет только базовые функции: установка напоминания, изменение цвета заметки, вставка изображения и добавление новых листов.

Недостатки:

* нельзя вернуться к старой версии записи.

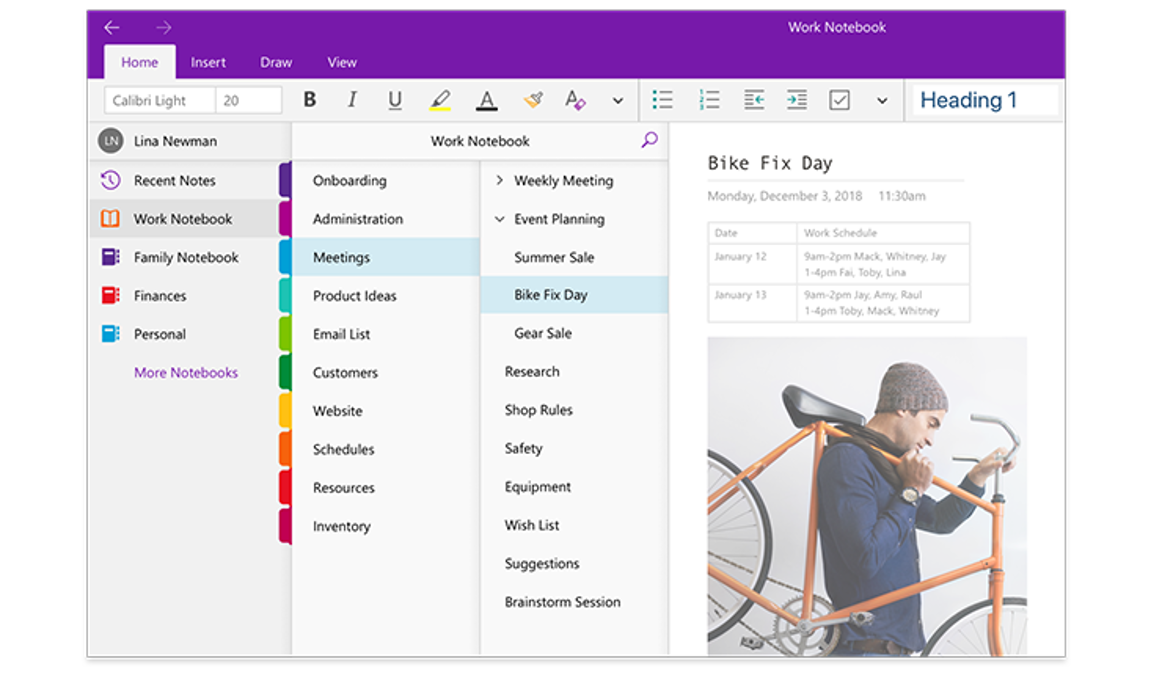


Рисунок 1.2 – Интерфейс приложения Microsoft OneNote

Сформированные на основе прототипов требования к программному средству представлены в разделе 2.

1. **Анализ требований к программному средству и разработка функциональных требований**

В рамках данного курсового проекта было целью разработать программное средство, которое помогло бы рядовому пользователю вести учет своих записок.

Для того, чтобы приступить к этапу разработки проекта, необходимо четко сформулировать функциональные требования к программному средству.

## **2.1. Определение требований к программному средству**

На основе анализа прототипов были выделены основные функции программного средства для администратора:

* просмотр списка пользователей;
* возможность сортировки и поиска пользователей;
* просмотр ограниченной информации о записках конкретного пользователя;
* все функции, доступные пользователю.

Для пользователя:

* создание записки на конкретный день, который можно выбрать путем нажатия на календарь;
* на календаре выделяются дни, которые уже имеют записи;
* указание настроения за день;
* форматирование текста записки: поддерживаются списки, различные шрифты, их цвета и размеры, полужирный, курсивный и подчеркнутый текст, выравнивание текста;
* прикрепление по желанию изображения к выбранному дню;
* привязывание цели к конкретному дню и указание времени выполнения;
* просмотр списка своих записок и целей.

**2.2. Описание функциональности программного средства**

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

Спецификация функциональности программного средства представлена в следующем пункте.

## **2.3. 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 Core 5.0.6;
* 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. Она предоставляет собой объектно-ориентированную технологию доступа к данным, является object-relational mapping (ORM) решением для платформы .NET Core. 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.

* 1. **Спецификация функциональных требований.**

Для функциональности программного средства необходимо создание базы данных для хранения информации приложения. Подробно о базе данных будет рассказано в главе 3.

Для использования программного средства пользователю потребуется иметь аккаунт. При запуске ему будет предложено авторизироваться, если он уже имеет аккаунт, либо пройти регистрацию, если он впервые встречается с Note2Self. Для авторизации входными параметрами являются логин и пароль пользователя, которые содержатся в базе данных. Чтобы зарегистрироваться, необходимо ввести логин и дважды пароль, чтобы пользователь убедился, что его запомнил. Введенные данные, успешно прошедшие валидацию, заносятся в базу данных.

В базе данных уже находится один профиль администратора. Чтобы избежать утечек информации, пользователь не может зарегистрировать себя в качестве администратора: каждый человек, заводящий аккаунт, по умолчанию рядовой пользователь. Администратора же можно завести только непосредственно из базы данных.

При авторизации пользователя его встречают несколько страниц. На странице «Календарь» пользователь может выбрать на календаре нужную дату и заполнить для этого дня записку. На странице «Записки» пользователь видит список всех своих записей, а также может при желании их редактировать: по клику по нужной строке откроется редактор записки. На странице «Цели» пользователь видит список всех своих целей. Также в панели меню присутствует кнопка «Выйти», которая позволяет сменить пользователя.

Администратор при авторизации видит такой же набор страниц за одним исключением: только администратору видна дополнительная кнопка «Администрирование». По нажатию на эту кнопку открывается страница со списком всех пользователей, имеющихся в базе данных. По клику на строку списка администратор может просмотреть ограниченную информацию о записках выбранного пользователя.

Все данные о пользователях, записках и целях должны заноситься в базу данных в соответствующие таблицы.

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. Архитектура системы**

В данном приложении используется архитектурный паттерн MVVM (MODEL-VIEW-VIEWMODEL). Он позволяет отделить логику приложения от визуальной части. MVVM состоит из трех частей:

* Модель описывает используемые в приложении данные;
* Представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением;
* Модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.1. ‒ Структура шаблона MVVM

Также в программном средстве Note2 Self используются шаблоны Repository и Unit of Work.

В приложениях шаблон Repository используется для инкапсуляции логики работы с базой данной. Он удобен, когда приходится оперировать множеством сущностей и моделей. Без этого паттерна пришлось бы даже примитивный метод для добавления строки в базу данных повторять по несколько раз. В то время как Repository позволяет прописать базовый набор методов для всех сущностей, и, где нужно, уже точечно добавить недостающие.

Шаблон Unit of Work помогает упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же DbContext. Объект, реализующий этот паттерн, отвечает за накопление информации о том какие объекты входят в транзакцию и каковы их изменения относительно исходных значений в хранилище.

## **3.2 Диаграммы UML**

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 последовательности взаимодействия с окном авторизации приведена в приложении Б.

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 В, 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.

Диаграммы использования показывает актеров и возможные прецеденты, при работе с приложением. Диаграмма использования приведена в приложении А.

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

Диаграмма классов ViewModel приведена в приложении Г, классов Model – в приложении Д.

## **3.3. Проектирование логической структуры 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 нашей программной подсистемы состоит из 3 таблиц, ее диаграмма изображена в приложении Е.

Таблица Users используется для хранения данных о пользователях. Включает 4 столбца: Id – id пользователя, Username – логин пользователя, Password – пароль пользователя, Role – роль пользователя в приложении, значение берется из перечисления Roles, в котором в данный момент 2 роли: Roles.User = 0, Roles.Admin = 1.

Логика записок основывается на том, что одной запиской считается аналог страницы бумажного ежедневника. Таким образом, именно таблица Notes содержит в себе настроение пользователя, текстовое содержимое, изображение. Информация о целях содержится в отдельной таблице Goals, поскольку в программном средстве Note2Self на один день (то есть записку) может приходиться несколько целей. Таблица Goals соединяется с Notes по внешнему ключу NotesId.

Таблица Notes хранит данные о записках. Включает 6 столбцов: Id – id записки, Text – содержимое записки с форматированием, Day – день, к которому привязана записка, Mood – настроение пользователя (берется из перечисления), UserId – внешний ключ Id из таблицы Users для их связи, Cover – изображение, которое можно прикрепить к записке.

Таблица Goals используется для хранения данных целях пользователя. Включает 4 столбца: Id – id цели, Description – текстовое содержимое цели, NotesId – связь с таблицей Notes по внешнему ключу, CompletionTime – время, на которое записывается цель.

## **3.4. Взаимодействие окон и представлений**

В программном средстве Note2Self все действия происходят внутри одного и того же окна, содержимое же заходится внутри пользовательских элементов управления (UserControl), которые переключаются в зависимости от ситуации.

При запуске приложения в окне запускается представление LoginControl с авторизацией, запрашивающее логин и пароль. Если нужно зарегистрироваться, это представление по клику на кнопку сменится на представление регистрации RegisterControl. После успешной авторизации LoginControl переключается на представление HomeControl, которое вмещает в себе меню и все то, чем может управлять пользователь. То есть, внутри HomeControl пользователь может попасть в представление со списком записок NotesControl, в представление со списком целей GoalsControl и в представление с календарем CalendarControl, внутри которого по клику на день в календаре открывается представление для редактирования записки. А также, если пользователь является администратором, он может открыть внутри HomeControl представление AdminControl со списком пользователей. Это показано на рисунке 3.2.

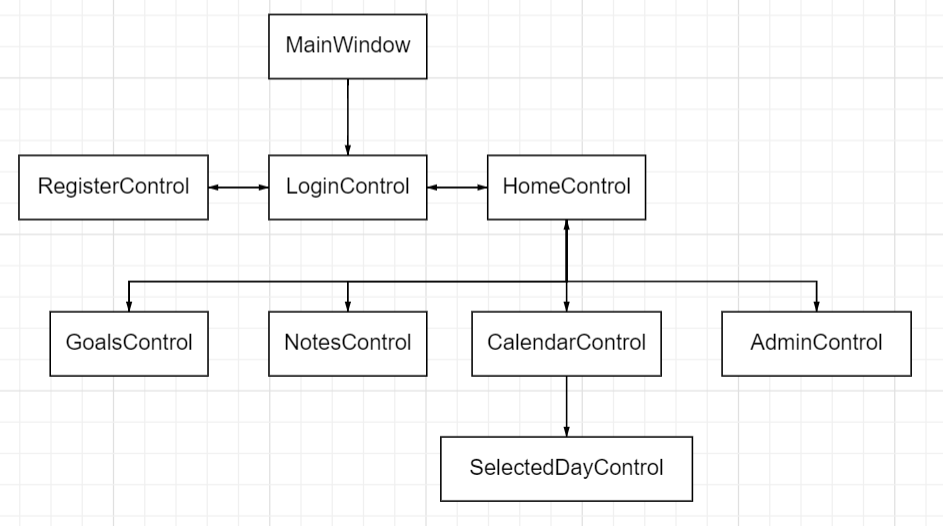


Рисунок 3.2. ‒ Схема взаимодействия окон и представлений

## **3.5. Структура проекта**

Структура проекта представлена на рисунке 3.3.

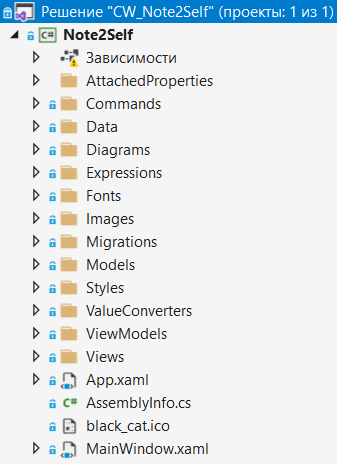


Рисунок 3.2 – структрура проекта

Папка «Data» содержит классы для работы с базой данных.

Папка «Models» содержит классы моделей.

Папка «Styles» содержит стили и шаблоны элементов управления.

Папка «Images» содержит изображения, нужные в приложении.

Папка «ValueConverters» содержит классы преобразования типов данных.

Папка «ViewModels» содержит классы моделей представлений.

Папка «Views» содержит все представления. Связи между этими представлениями описаны в пункте 3.4.

1. **Реализация программного средства**

## **4.1 Реализация MVVM и других шаблонов**

Паттерн MVVM реализуется через базовый класс BaseViewModel (рисунок 4.2) и интерфейс INestedViewModel (рисунок 4.3), наследующий интерфейс INotifyPropertyChanged. На их основе сделаны все. Для реализации паттерна файлы программы были распределены по соответствующим пространствам имен и реализованы следующие функции.

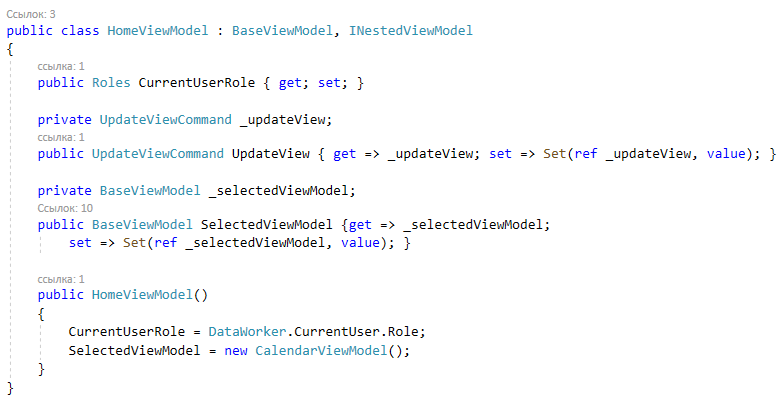


Рисунок 4.1 – пример использования BaseViewModel и INestedViewModel

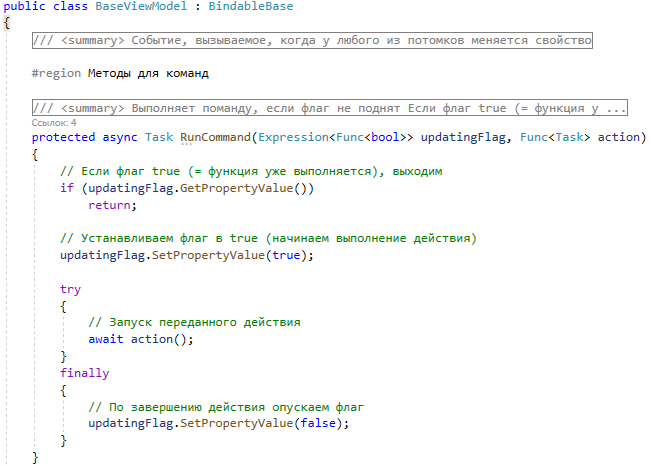


Рисунок 4.2 – структура класса BaseViewModel

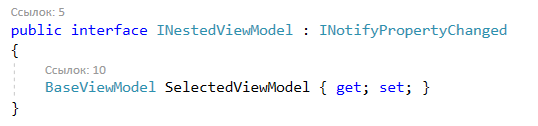


Рисунок 4.3 – структура интерфейса INestedViewModel

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде собственной команды Command код которой приведен в приложении Е. Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться
* Execute: собственно, выполняет логику команды

В программном средстве Note2Self также реализованы шаблоны Repository и Unit Of Work. Рассмотрим их реализацию ниже.

Для начала, рассмотрим шаблон Repository. Для его реализации, прежде всего, нужно было написать интерфейс IRepository, который можно увидеть на рисунке 4.4. В этом интерфейсе описаны методы, которые базовый класс Repository обязывается реализовать.

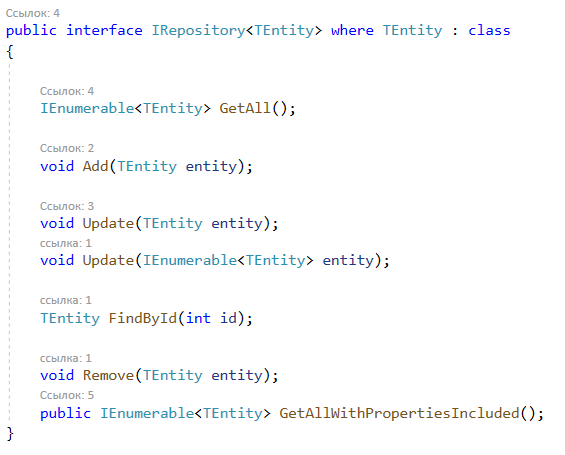


Рисунок 4.4 – структура интерфейса IRepository

Далее, этот интерфейс наследуется базовым классом Repository. В нем реализуются базовые методы, которые будут поддерживаться всеми репозиториями. В конструкторе класса передается неуникальный контекст базы данных, что позволяет методам этого класса быть расширяемыми. Некоторые из этих методов изображены на рисунке 4.5.

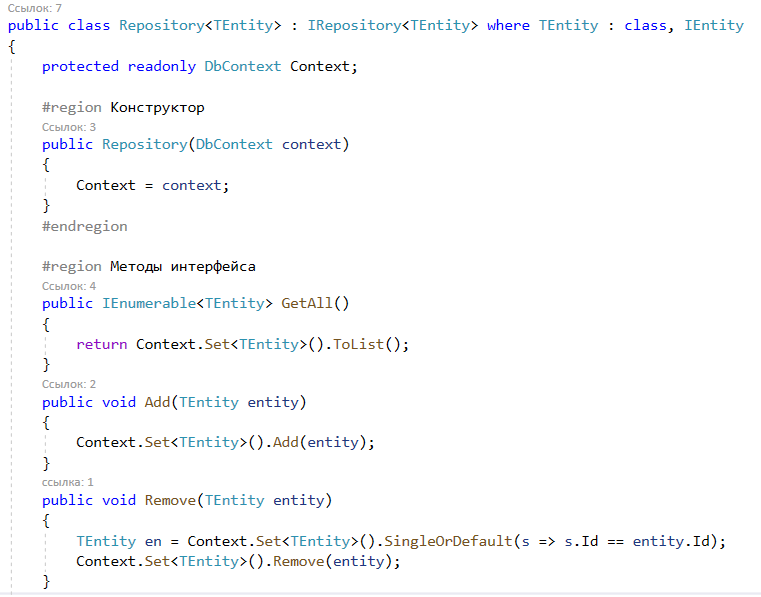


Рисунок 4.5 – фрагмент структуры класса Repository

На примере с сущностью Users продемонстрированно (рисунок 4.6), как связать новый класс UsersRepository с сущностью. В каждой из нужных в этом программном средстве репозиториев будет передаваться контекст базы данных Note2SelfContext.

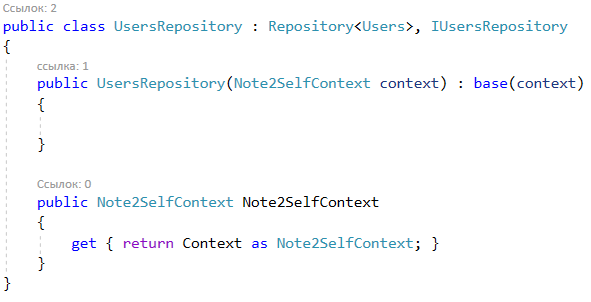
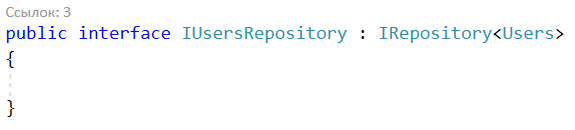


Рисунок 4.6 – структуры интерфейса IUsersRepository и класса UsersRepository

Теперь, когда у нас есть нужные репозитории, перейдем к шаблону Unit Of Work. Интерфейс IUnitOfWork должен наследоваться от IDisposable. Структуру этого интерфейса можно увидеть на рисунке 4.7.

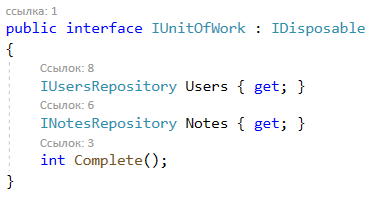


Рисунок 4.7 – структура интерфейса IUnitOfWork

Класс UnitOfWork наследует указанный выше интерфейс и является тем звеном, которое передает репозиториям контекст базы данных, с которым мы работаем, а также он определяет методы SaveChanges() и Dispose() для управления транзакциями, что видно на рисунке 4.8.

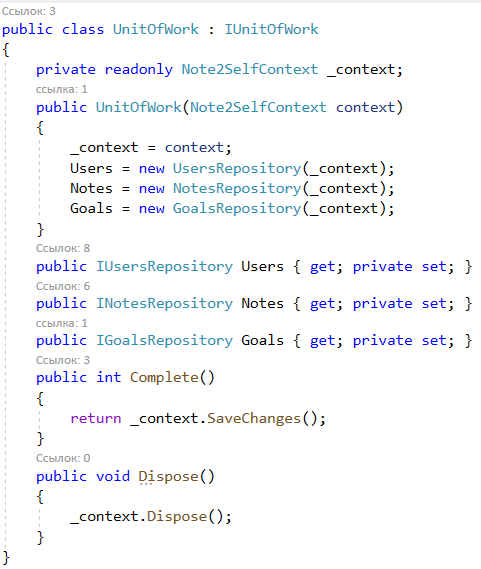


Рисунок 4.8 – структура класса UnitOfWork

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

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

На рисунке 4.9 находится код, реализующий регистрацию.

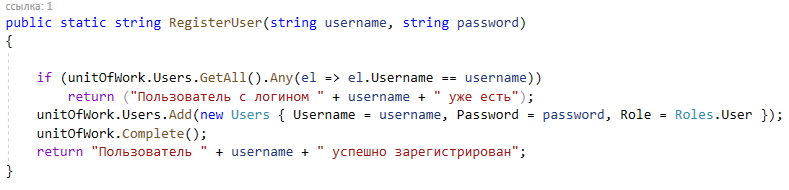
****

Рисунок 4.9 – Команда, реализующая регистрацию и вызываемый ей метод из класса DataWorker

При регистрации пользователь имеет по умолчанию роль Roles.User. Если все поля заполнены, введенный логин не занят и указанные пароли совпадают, то осуществляется регистрация и переход на представление авторизации.

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

На рисунке 4.10 находится код, реализующий авторизацию.

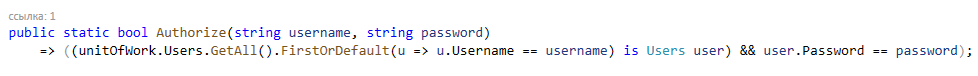
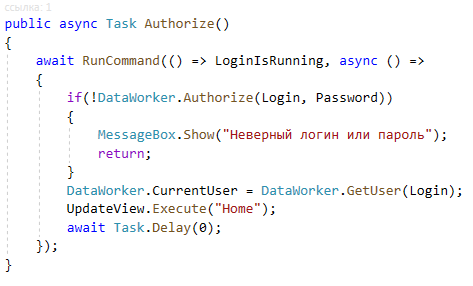
****

Рисунок 4.10 – Команда, реализующая авторизацию и вызываемый ей метод из класса DataWorker

## **4.3. Добавление записок и целей**

Добавление записки работает следующим образом: по клику по дню календаря в CalendarControl открывается редактор записки (SelectedDayControl). Цели добавляются при заполнении строки DagaGrid, который находится в SelectedDayControl. В методе AddNote() происходит проверка, существует ли уже у данного пользователя записка на выбранный день. Если существует, будет вызываться метод из репозитория для обновления, иначе вызывается метод для добавления. Записка обновляется после нажатия по кнопке «Сохранить».

На рисунке 4.11 находится код, реализующий добавление записки.

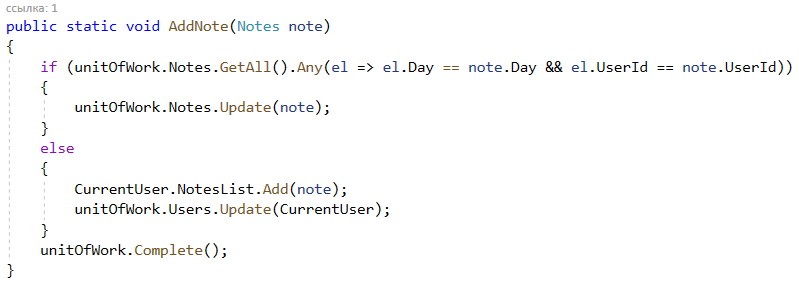
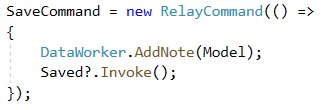
****

Рисунок 4.11 – Команда, реализующая добавление записки и вызываемый ей метод из класса DataWorker

1. **Тестирование, проверка работоспособности и анализ полученных результатов**

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. Если строки не заполнены, логин или пароль введены неверно, то появляется сообщение, говорящее о произошедшей ошибке. Сообщение представлено на рисунке 5.1.

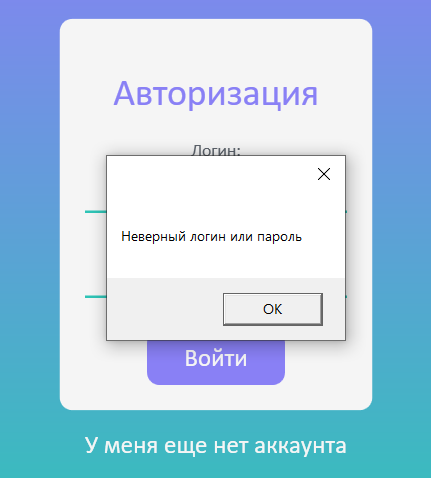


Рисунок 5.1 – При авторизации введены ошибочные данные

Далее рассмотрим представление для регистрации. Оно имеет 3 поля для ввода: логин, пароль и повторный ввод пароля. В приложении предусмотрена обработка следующих ошибок:

Если пользователь оставил хоть одно поле незаполненным, он увидит сообщение с рисунка 5.2.

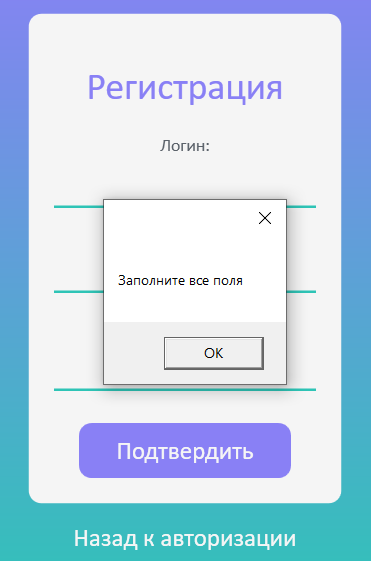


Рисунок 5.2 – При регистрации есть пустые поля

Если пользователь попробует занять логин, который уже существует в базе данных, он увидит сообщение с рисунка 5.3.

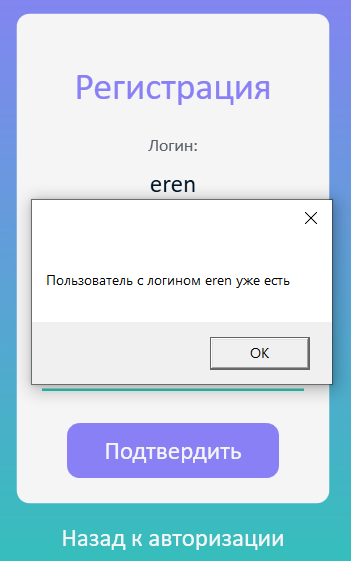


Рисунок 5.3 – При регистрации выбран уже занятый логин

Если пользователь при регистрации введет пароли, которые не совпадают, приложение ему подскажет сообщением с рисунка 5.4.

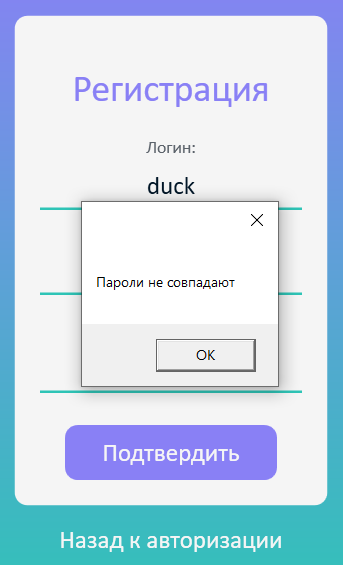


Рисунок 5.4 – При регистрации пользователь ошибся в паролях

1. **Руководство по использованию**

## **6.1. Регистрация и авторизация**

При запуске приложения запускается главное окно, в которое выводится представление с формой для авторизации. Если у пользователя еще нет аккаунта, ему следует нажать на кнопку «У меня еще нет аккаунта», которая его перенаправит на представление с формой для регистрации. Представление авторизации представлено на рисунке 6.1.

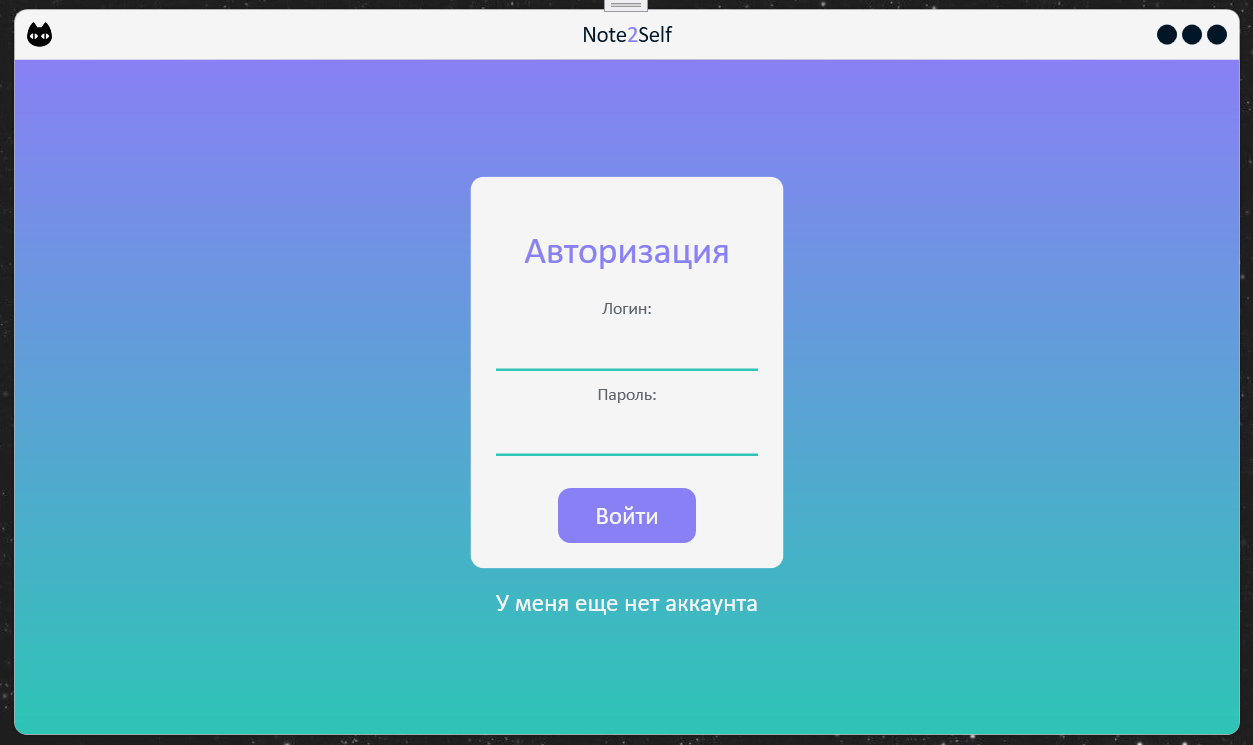


Рисунок 6.1 – Представление авторизации

Если была нажата кнопка «У меня еще нет аккаунта», в открывшемся представлении следует ввести все данные о регистрируемом пользователе. В случае, если кнопка была нажата ошибочно, можно нажать на кнопку «Назад к авторизации» и вернуться к представлению авторизации. Представление регистрации изображено на рисунке 6.2.

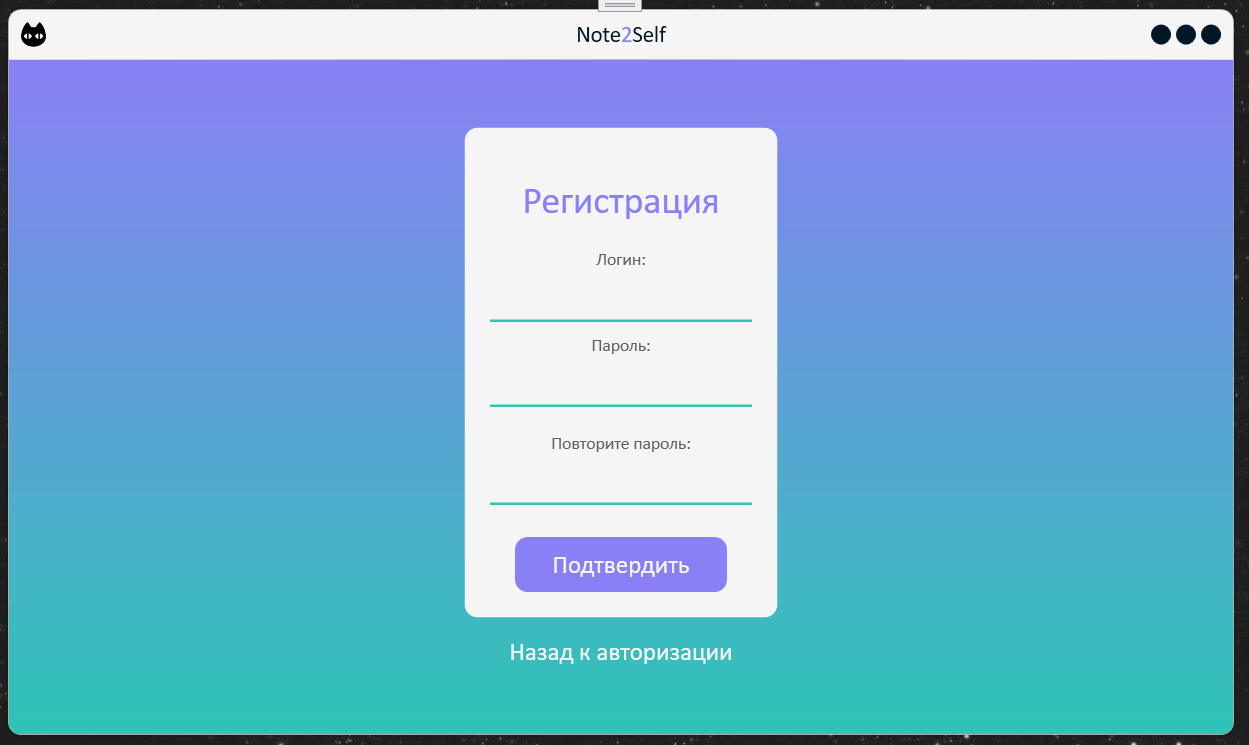


Рисунок 6.2. ‒ Представление регистрации

## **6.2. Домашнее представление**

При успешной авторизации пользователя приложение переключит представление с авторизации на домашнее. На рисунке 6.3. показано то, что новый пользователь увидит после своей первой авторизации.

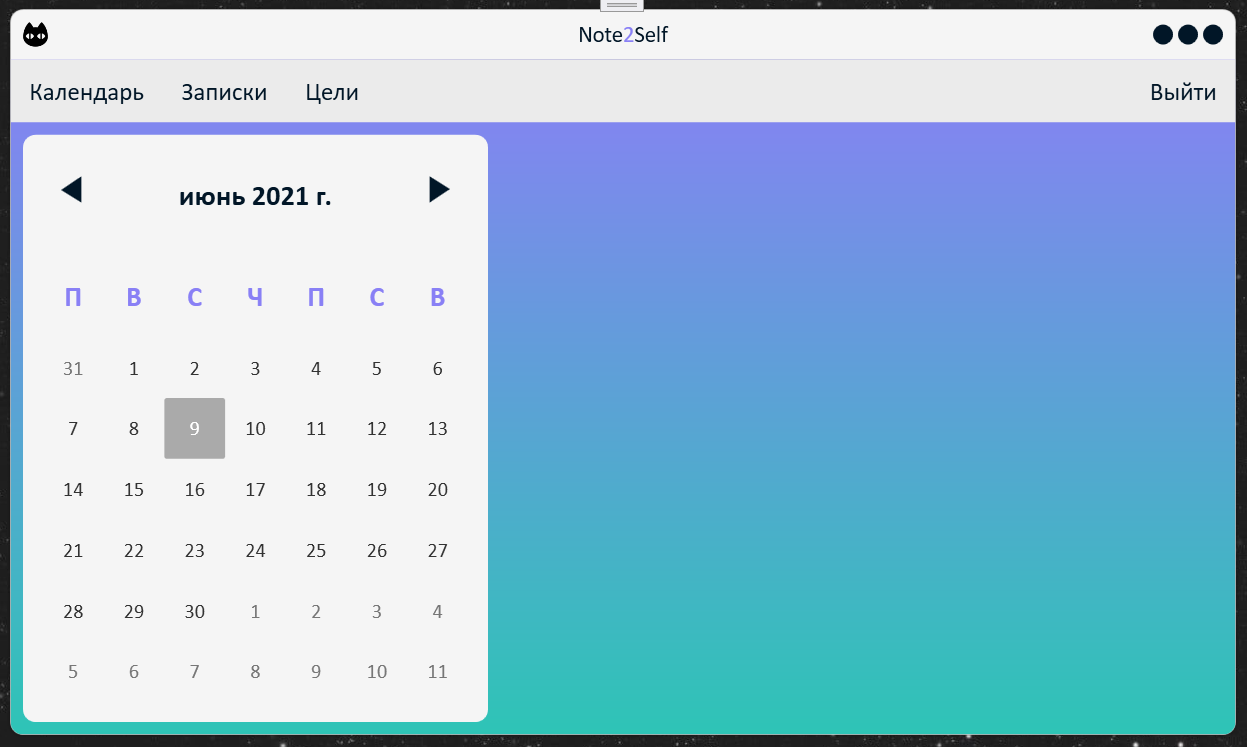


Рисунок 6.3. ‒ Домашнее представление после первой авторизации

Если пользователь желает создать записку, ему следует нажать на нужный день на календаре, впоследствии чего откроется редактор записки, как можно увидеть на рисунке 6.4.

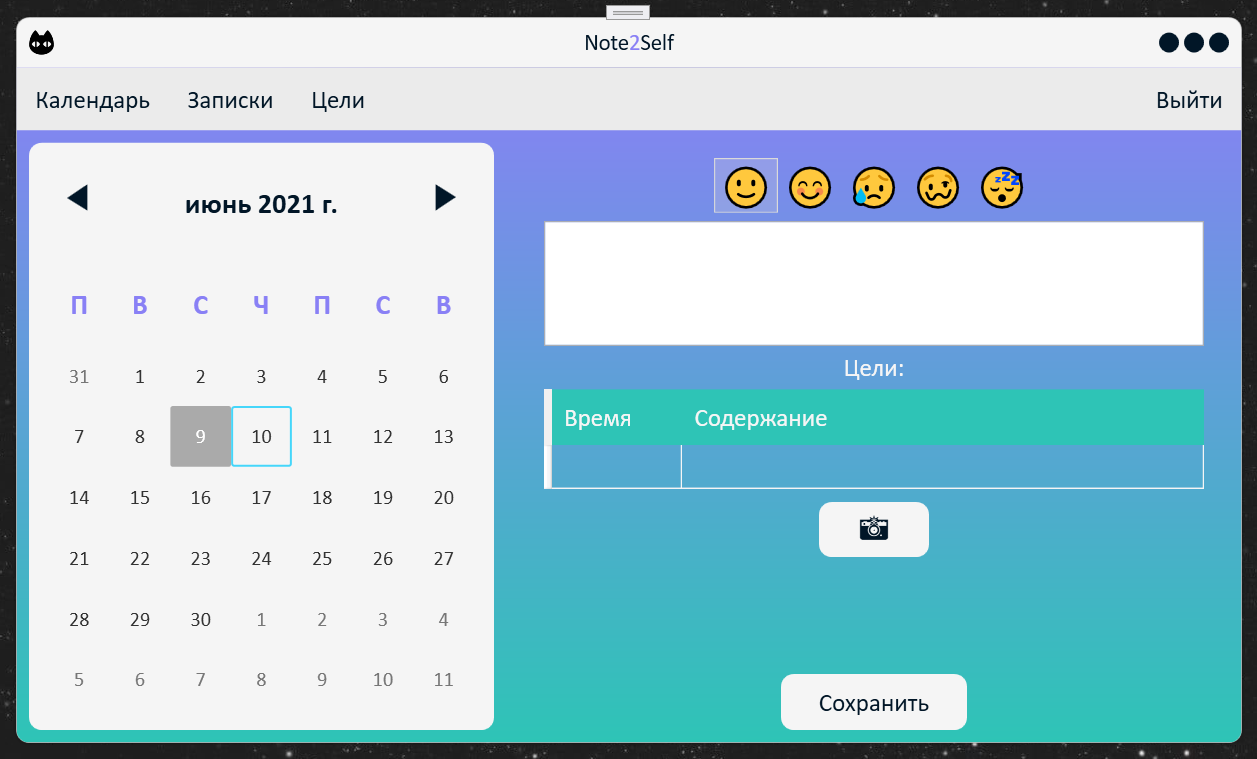


Рисунок 6.4. ‒ Редактор записки

В редакторе записки есть 4 интересующих нас составляющих, которые мы рассмотрим сверху вниз. Во-первых, пользователю дается возможность выбрать настроение для указанного дня из 5 существующих вариантов, это делается кликом мыши по подходящей эмоции. Во-вторых, мы имеем поле для ввода текста, которое поддерживает форматирование. Чтобы получить доступ к редактору текста, пользователю следует выделить нужный фрагмент текста, и тогда всплывет панель инструментов. В-третьих, есть список целей на этот день. По умолчанию он пустой, но, чтобы добавить цель (или несколько), пользователю следует просто построчно занести информацию. Подтвердить строку можно нажатием клавиши Enter. И, наконец, мы имеем кнопку с фотоаппаратом. По нажатию на эту кнопку откроется окно, которое предложит выбрать вам изображение с вашего компьютера, которое вы бы желали прикрепить к данной записке.

Чтобы более информативно проиллюстрировать работу приложения, на следующем рисунке тому же пользователю мы уже создадим 3 записки. Обратите внимание, что дни, на которые уже есть записки, будут окрашены с бирюзовый цвет. Это можно увидеть на рисунке 6.5.

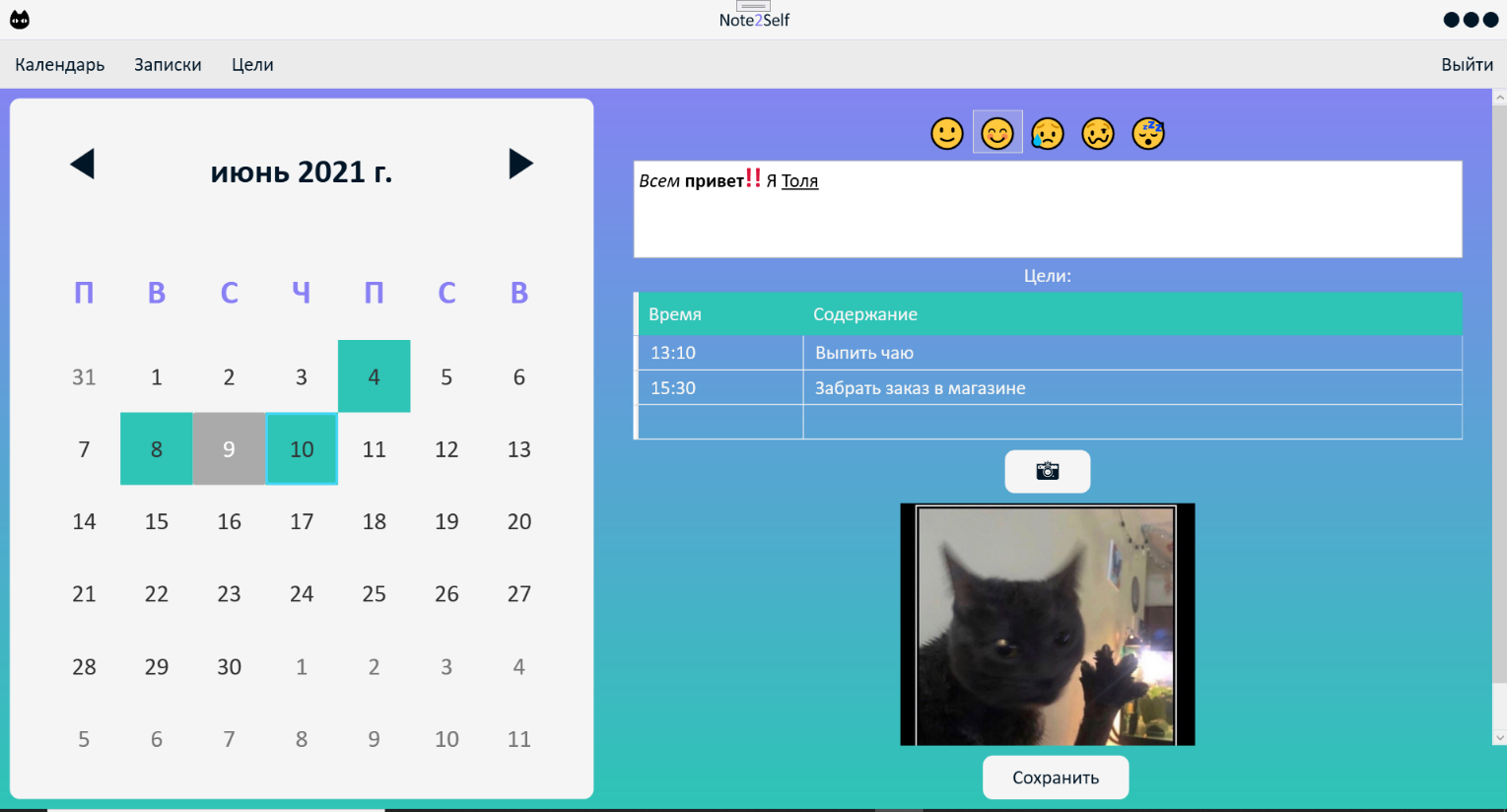


Рисунок 6.5. ‒ Редактор с тремя записками

Чтобы увидеть список всех своих записок, пользователю нужно нажать на кнопку «Записки» в меню, после чего откроется представление со списком. Если нужно, пользователь может нажать на строку списка, из-за чего также откроется редактор записки. Это изображено на рисунке 6.6.

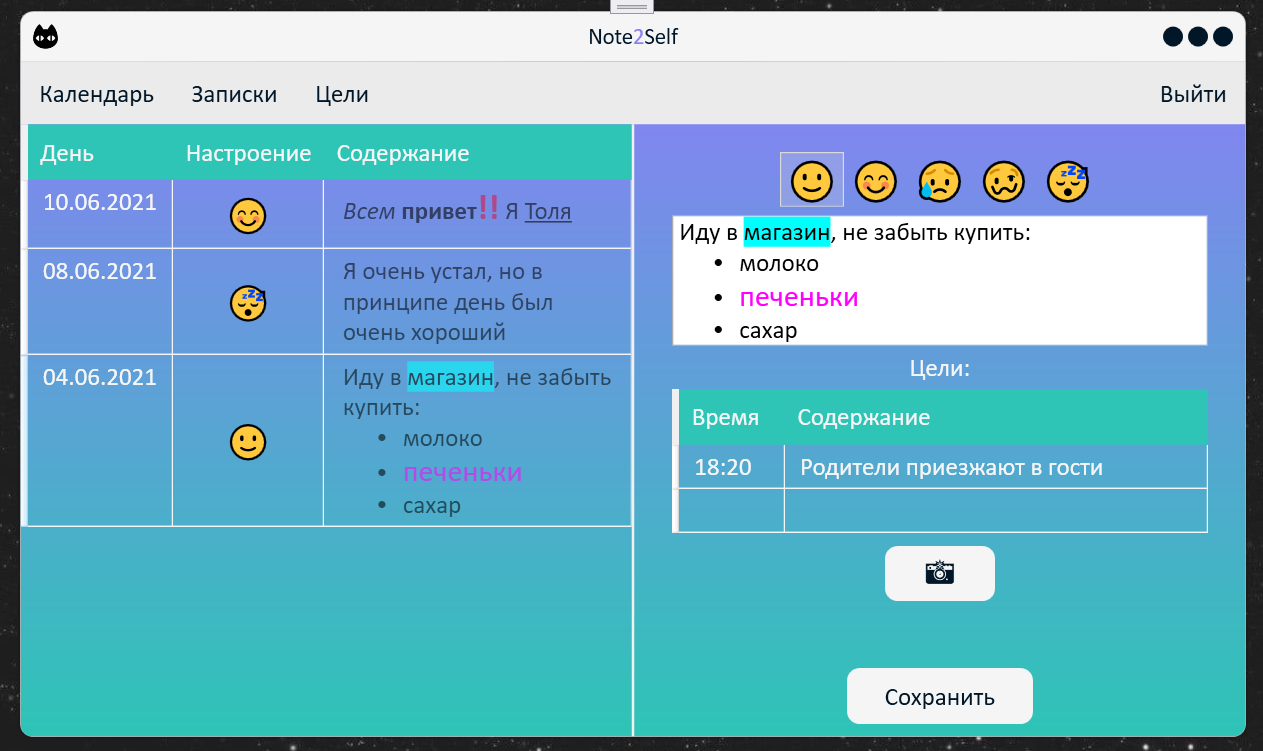


Рисунок 6.6. ‒ Список записок и открытый редактор

Чтобы увидеть список всех своих целей, пользователю следует выбрать кнопку «Цели» и откроется соответствующее представление, что видно на рисунке 6.7.

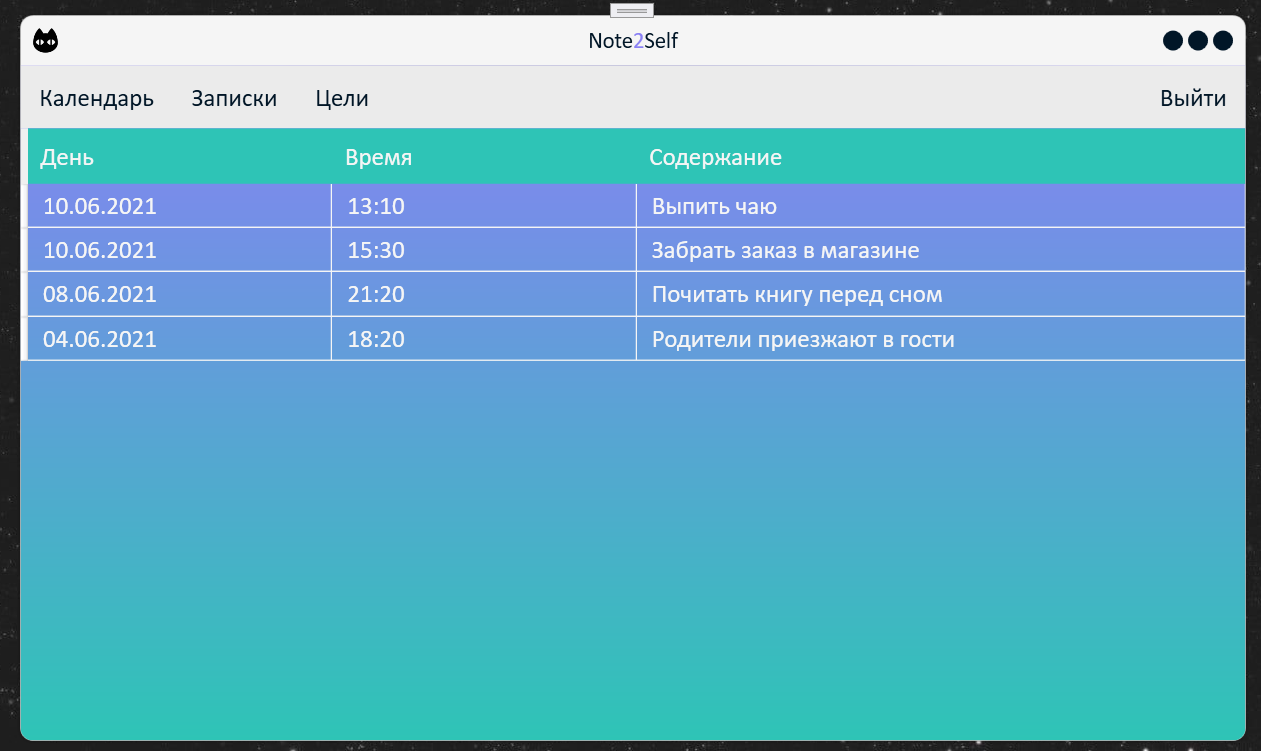


Рисунок 6.7. ‒ Список целей

## **6.3. Администрирование**

При успешной авторизации администратора приложение переключит представление с авторизации на домашнее. В домашнем представлении для администратора доступны все возможности обычного пользователя, а также кнопка «Администрирование» в меню. При нажатии на эту кнопку откроется представление, в котором администратор сможет увидеть список всех пользователей, имеющихся в базе данных. Администратор также может с помощью клика мышкой по строке с нужным пользователем увидеть его записи. Однако в целях конфиденциальности будет видно только дату и настроение. Это проиллюстрировано на рисунке 6.8.

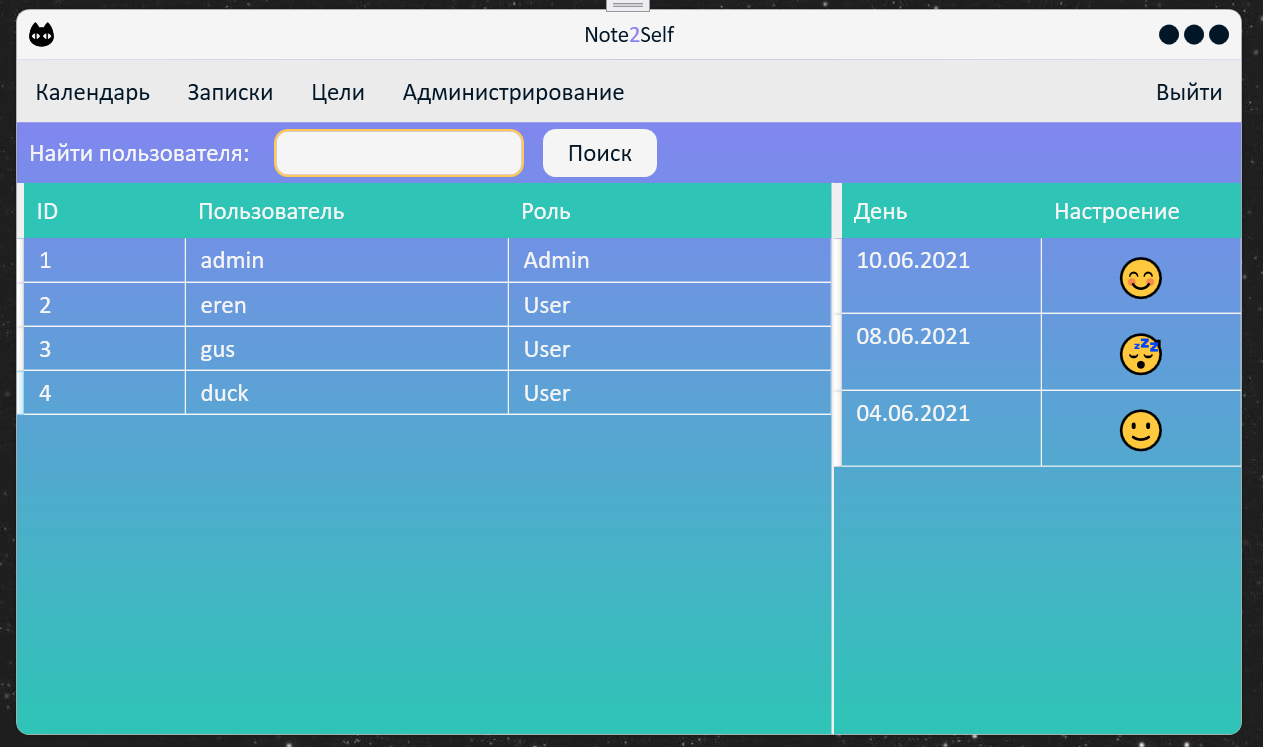


Рисунок 6.8. ‒ Список пользователей, видный только администратору

## **6.3. Смена пользователя**

При желании сменить пользователя или выйти из аккаунта, пользователю стоит нажать на кнопку «Выйти» в правом углу строки меню.

# **ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы было разработано программное средство «Note2Self», которое позволяет рядовому пользователю вести свой виртуальный ежедневник, создавая записки и привязывая к ним свое настроение, форматирование, цели и изображения. работать с базой данных. Основные достоинства – простота использования и оригинальный дизайн.

Разработанное приложение удовлетворяет всем требованиям, предъявленным в задании. В дальнейшем возможно добавление разных видов представления записок на календаре: например, в формате недели.

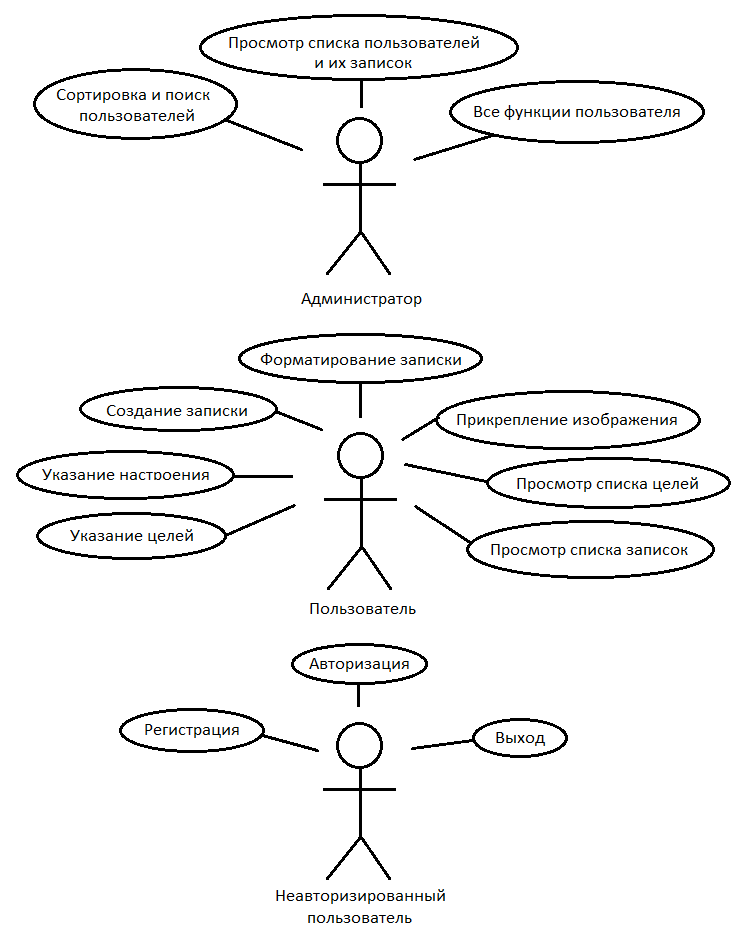
Также в процессе выполнения данного курсового проекта были закреплены навыки в программировании на языке C#, создании приложений на WPF, использование Entity Framework Core 5.0.6, работа с современным паттерном MVVM, проектирование базы данных и реализация их в СУБД MS SQL Server 2019.

Был получен опыт взаимодействия со сторонними библиотеками, изучения их и внедрения в свой проект.

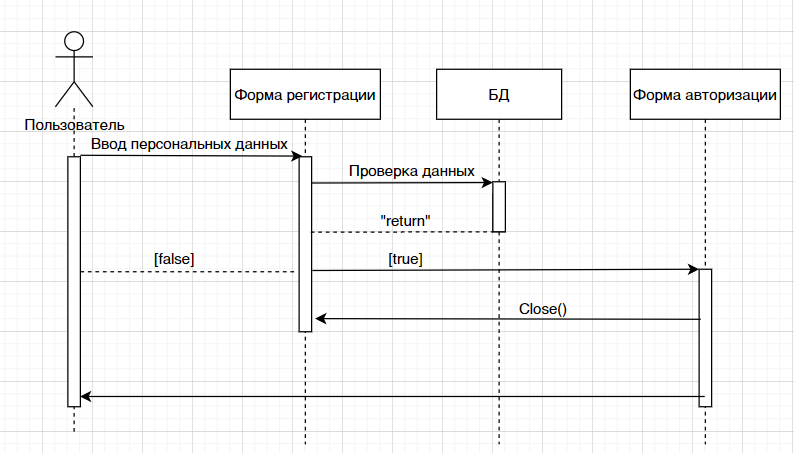
# **Список использованной литературы**

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.

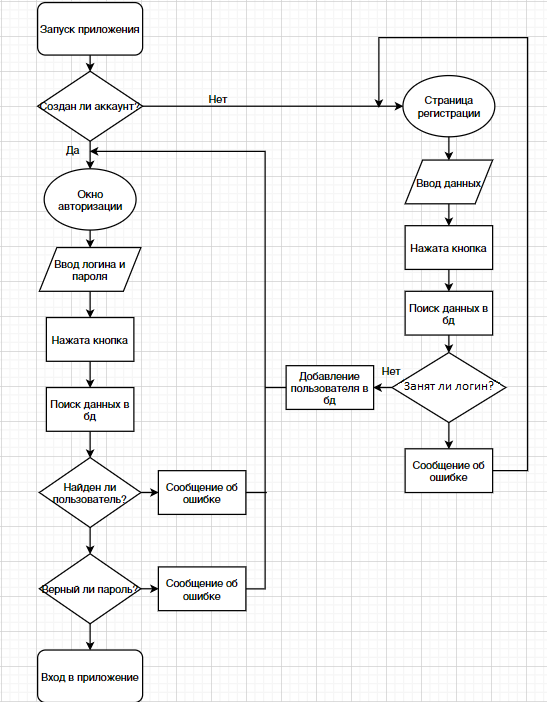
# **Приложение А**



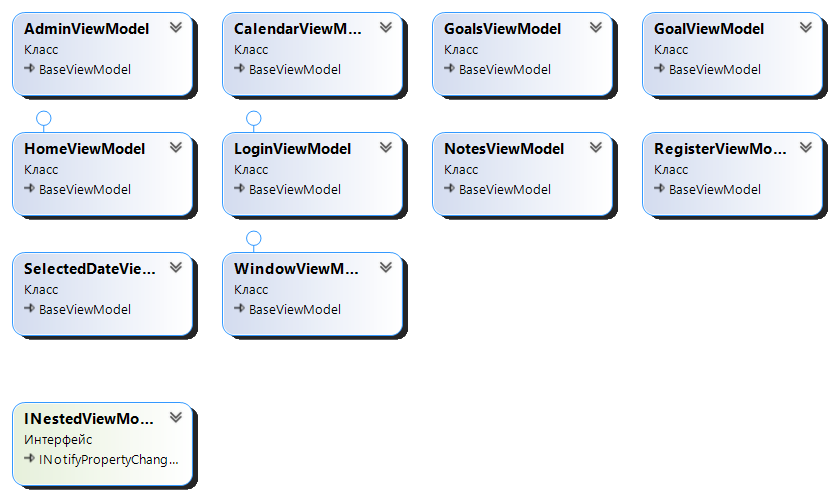
# **Приложение Б**



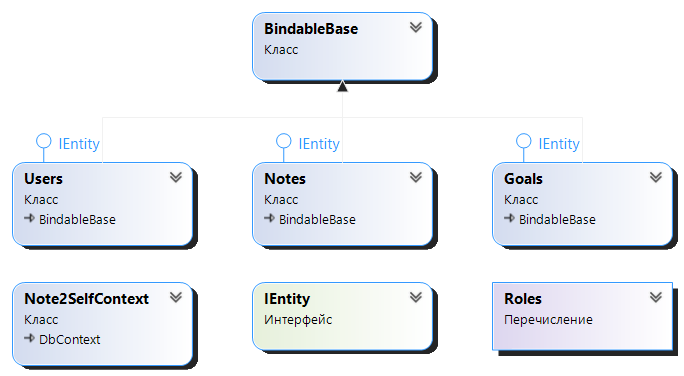
# **Приложение В**



# **Приложение Г**



# **Приложение Д**



# **Приложение Е**

