Министерство науки и высшего ОБРАЗОВАНИЯ РОССИЙСКОЙ федерации

федеральное государственное бюджетное образование учреждение

высшего образования

«Курганский государственный университет»

Кафедра «Программное обеспечение автоматизированных систем»

**Агрегатор онлайн-публикаций**

**ОТЧЕТ**

**по учебной практике**

Разработал студент гр. ИТ-0930222 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / С.Ю. Чернов /

подпись, дата

Направление 09.03.03 – «Прикладная информатика»

Руководитель, к.ф. – м.-н., доцент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / О.С. Черепанов /

подпись, дата

Курган, 2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc168441987)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc168441988)

[1.1 Характеристика предметной области 4](#_Toc168441989)

[1.2 Спецификация требований к программному обеспечению 4](#_Toc168441990)

[2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 6](#_Toc168441991)

[2.1 MVVM - архитектура 6](#_Toc168441992)

[2.2 Диаграмма пакетов классов 7](#_Toc168441993)

[2.3 Диаграмма классов 8](#_Toc168441994)

[3 ОБОСНОВАНИЕ ВЫБОРА СРЕДСТВ РЕАЛИЗАЦИИ 24](#_Toc168441995)

[3.1 Выбор средств реализации 24](#_Toc168441996)

[ЗАКЛЮЧЕНИЕ 27](#_Toc168441997)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28](#_Toc168441998)

# ВВЕДЕНИЕ

В современном мире, насыщенном информацией, поиск актуальных и интересных материалов по различным темам становится всё более сложным. Разнообразие источников, от новостных сайтов и блогов до социальных сетей и онлайн-журналов, создаёт информационный шум, мешающий пользователям найти действительно ценный контент.

Реализация программного приложения обеспечит реализацию следующих возможностей:

1. Авторизация пользователей;
2. Добавление источников (каналов) статей/новостей;
3. Добавление публикаций в «Избранное»;
4. Добавление публикаций в «Отложенное»;
5. Рассылка уведомлений о выходе новых публикаций;
6. Чтение публикации внутри приложения;
7. Открытие публикации в браузере;
8. Копирование ссылки на публикацию в буфер обмена;
9. Изменение темы приложения (на светлую или темную);
10. Сохранение настроек приложения.

Целью данной курсовой работы является создание графического пользовательского приложения (RSS-ридера), которое позволяет пользователям легко находить и отслеживать интересующие их материалы, собранные из различных источников (RSS-лент).

Для достижения поставленной цели были поставлены следующие задачи:

1. Изучить предметную область RSS-ридера и требования к его программной реализации;
2. Определить функциональные требования к системе;
3. Разработать архитектуру приложения;
4. Программно реализовать систему.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 Характеристика предметной области

Аббревиатуры RSS расшифровывается как Really Simple Syndicate — это расширение стандарта передачи данных XML, используемое для организации заголовков новостей.

Если на сайте публикуют новости, статьи, посты или другой контент единого формата, то с помощью технологии RSS из этих материалов автоматически формируется структурированная лента. В основном в нем хранится заголовок источника, ссылка на него, его описание и список из последних выпущенных публикаций, в котором у каждой публикации написан заголовок и ссылка, так же может быть описание, дата выхода, ссылка на изображение. Для чтения таких лент обычно используют RSS-ридеры.

RSS-ридер (или агрегатор) — это приложение, которое позволяет подписываться на RSS-ленты разных ресурсов. Благодаря этому пользователь может в одном месте получать анонсы обновлений из нескольких источников.

Благодаря агрегатору пользователь подписывается в приложении на разные ресурсы (сайты) и получает вышедшие публикации в нем, что позволяет добавлять публикации в отложенное или избранное, присылать уведомления о новых вышедших публикациях и хранить их в том порядке, в котором они выходили.

## 1.2 Спецификация требований к программному обеспечению

Основные функции автоматизированной системы:

1. Авторизация пользователей;
2. Добавление источников (каналов) статей/новостей;
3. Добавление публикаций в «Избранное»;
4. Добавление публикаций в «Отложенное»;
5. Рассылка уведомлений о выходе новых публикаций;
6. Чтение публикации внутри приложения;
7. Открытие публикации в браузере;
8. Копирование ссылки на публикацию в буфер обмена;
9. Изменение темы приложения (на светлую или темную);
10. Сохранение настроек приложения.

# 2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## 2.1 MVVM - архитектура

Для организации работы приложения использовалась MVVM-архитектура (Рисунок 2.1). MVVM — это паттерн архитектуры, который разделяет приложение на три ключевых компонента: Model, View и ViewModel.

* Model: это бизнес-логика и данные. Здесь обрабатываются данные, выполняются операции и происходит взаимодействие с источниками данных.
* View: это пользовательский интерфейс (UI) приложения. View ответственен за отображение данных и реагирование на взаимодействие пользователя.
* ViewModel: ViewModel является посредником между Model и View. Он содержит логику, связанную с UI, и управляет состоянием приложения. ViewModel предоставляет данные, которые View отображает, и обрабатывает действия пользователя.

Преимущества выбора данной архитектуры:

1. Разделение логики и представления. MVVM четко разделяет представление (View), бизнес-логику (ViewModel) и данные (Model). Это позволяет поддерживать код в чистоте и лучше организовать проект.
2. Уменьшение кода. Большая часть логики обработки данных выносится в ViewModel, что уменьшает объем кода, который нужно писать в коде представления.
3. Поддержка повторного использования компонентов. ViewModel можно легко переиспользовать в разных частях приложения или даже в разных проектах, поскольку они не зависят от конкретного представления.
4. Гибкость. Отделение ViewModel от View позволяет легко изменять или обновлять представление без изменения логики приложения.
5. Модульность. Каждый компонент (Model, ViewModel, View) разрабатывается отдельно, что делает их более модульными и менее взаимозависимыми [[1](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)].

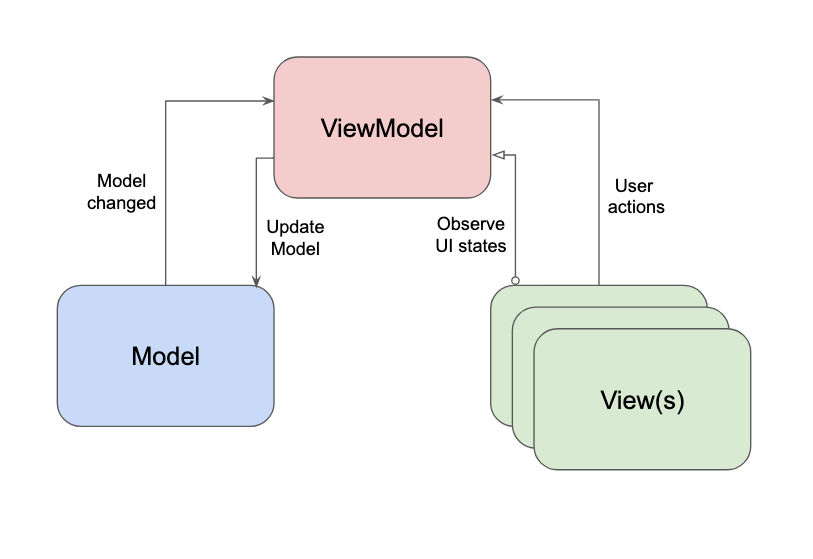


Рисунок 2.1 – Диаграмма архитектуры – MVVM

## 2.2 Диаграмма пакетов классов

На рисунке 2.2 представлена диаграмма пакетов:

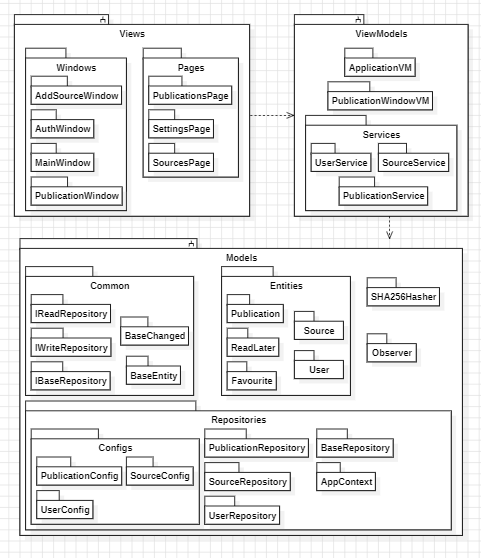


Рисунок 2.2 – Диаграмма пакетов

Диаграмма пакетов включает в себя:

1. Models – классы, содержащие данные и методы по их использованию;
2. ViewModels – классы связи Model с представлением View;
3. Views – классы, предоставляющие классы и страницы главного окна для отображения данных пользователю.

## 2.3 Диаграмма классов

На рисунке 2.3 представлена диаграмма классов.

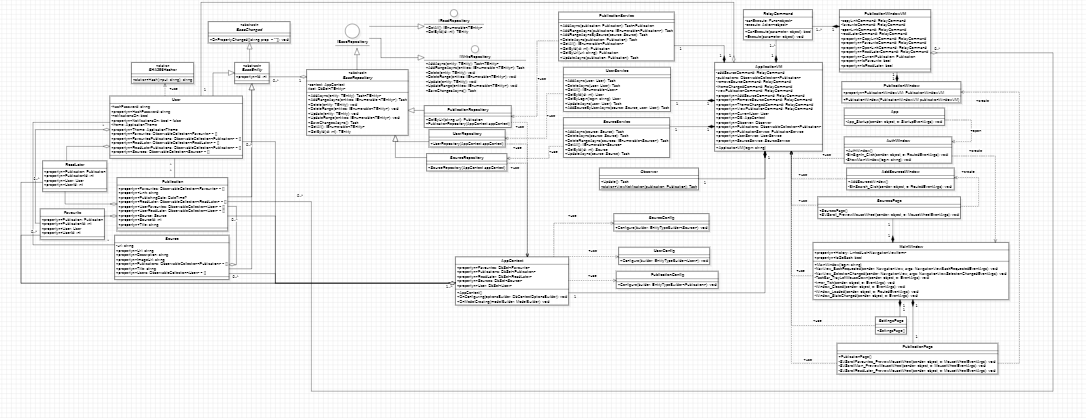


Рисунок 2.3 – Диаграмма классов

Класс App – точка входа программы. Изначально открывает окно авторизации AuthWindow.

Класс AuthWindow – начальное окно. Предназначено для регистрации или входа пользователя в свой профиль. После успешной авторизации, открывает окно MainWindow.

Класс MainWindow – главное окно приложения. В нем отображаются страницы: SettingsPage, SourcesPage, PublicationsPage.

Класс SettingsPage – страница настроек приложения. Через нее осуществляет изменение темы приложения и включение или отключение отправки уведомлений.

Класс SourcesPage – страница, отображающие добавленные источники пользователем. При нажатии на кнопку «Добавить», открывает окно AddSourcesWindow.

Класс AddSourcesWindow – окно добавления источника.

Класс PublicationsPage – страница, отображающие последние вышедшие онлайн-публикации источников. При нажатии на одну из публикаций, открывает окно PublicationWindow. Отображает вкладки «Основное», «Отложенное» и «Избранное».

Класс PublicationWindow – окно отображения публикации внутри приложения. Позволяет добавить в избранное или отложенное, скопировать ссылку на эту публикацию в буфер обмена и открыть ее через браузер.

Класс ApplicationVM – модель представления, к которой могут обращаться представления для работы с моделями. В таблице 2.1 представлены поля класса. В таблице 2.2 представлены свойства класса. В таблице 2.3 представлены методы класса.

Таблица 2.1 – поля класса ApplicationVM

|  |  |
| --- | --- |
| Поле | Описание |
| addSourceCommand: RelayCommand | Команда для обработки добавления нового источника. |
| publications: ObservableCollection<Publication> | Коллекция для хранения публикаций всех источников. |
| removeSourceCommand: RelayCommand | Команда для обработки удаления выбранного источника (со всеми его публикациями). |
| themeChangedCommand: RelayCommand | Команда для обработки изменения темы приложения. |
| viewPublicationCommand: RelayCommand | Статическая команда для обработки нажатия на публикацию. Открывает новое окно PublicationWindow, показывающая выбранную публикацию. |

Таблица 2.2 – свойства класса ApplicationVM

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>AddSourceCommand: RelayCommand | Свойство для работы с командой addSourceCommand. |
| <<property>>RemoveSourceCommand: RelayCommand | Свойство для работы с командой removeSourceCommand. |
| <<property>>ThemeChangedCommand: RelayCommand | Свойство для работы с командой themeChangedCommand. |
| <<property>>ViewPublicationCommand: RelayCommand | Свойство для работы с командой viewPublicationCommand. |
| <<property>>CurrentUser: User | Статическое свойство для хранения текущего пользователя. |
| <<property>>DB: AppContext | Статическое свойство для хранения контекста базы данных. |
| <<property>>Observer: Observer | Статическое свойство для хранения наблюдателя, обновляющего список публикаций источников. |
| <<property>>Publications: ObservableCollection<Publication> | Свойство для отображения публикаций источников данного пользователя. |
| <<property>>PublicationService: PublicationService | Статическое свойство для работы с таблицей публикаций в базе данных. |
| <<property>>UserService: UserService | Статическое свойство для работы с таблицей пользователей в базе данных. |
| <<property>>SourceService: SourceService | Статическое свойство для работы с таблицей источников в базе данных. |

*Окончание таблицы 2.2*

Таблица 2.3 – методы класса ApplicationVM

|  |  |
| --- | --- |
| Метод | Описание |
| ApplicationVM(in login:string) | Конструктор класса ApplicationVM, в котором приложение получает текущего пользователя по его логину. |

Класс PublicationVM – модель представления, к которой может обращаться PublicationWindow для работы с данными. В таблице 2.4 представлены поля класса. В таблице 2.5 представлены свойства класса.

Таблица 2.4 – поля класса PublicationVM

|  |  |
| --- | --- |
| Поле | Описание |
| copyLinkCommand: RelayCommand | Команда для обработки копирования ссылки данной публикации в буфер обмена. |
| favouriteCommand: RelayCommand | Команда для обработки добавления данной публикации в список «Избранное». |
| openLinkCommand: RelayCommand | Команда для обработки открытия данной публикации в браузере. |
| readLaterCommand: RelayCommand | Команда для обработки добавления данной публикации в список «Отложенное». |

Таблица 2.5 – свойства класса PublicationVM

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>CopyLinkCommand: RelayCommand | Свойство для работы с командой copyLinkCommand. |
| <<property>>FavouriteCommand: RelayCommand | Свойство для работы с командой favouriteCommand. |
| <<property>>OpenLinkCommand: RelayCommand | Свойство для работы с командой openLinkCommand. |
| <<property>>ReadLaterCommand: RelayCommand | Свойство для работы с командой readLaterCommand. |
| <<property>>CurrentPublication: Publication = publication | Свойство, хранящее текущую публикацию. |
| <<property>>IsFavourite: bool | Свойство для определения того, была ли данная публикация добавлена в список «Избранное». |
| <<property>>IsReadLater: bool | Свойство для определения того, была ли данная публикация добавлена в список «Отложенное». |

Класс Observer – класс, для обновления последних публикаций источников и отправке уведомлений пользователю о их выходе. В таблице 2.6 представлены методы класса.

Таблица 2.6 – методы класса Observer

|  |  |
| --- | --- |
| Метод | Описание |
| Update(): Task | Метод, предназначенный для обновления последних вышедших публикаций у всех источников текущего пользователя. |
| ViewNotification(in publication:Publication): Task | Статический метод, предназначенный для создания уведомления о выходе переданной в метод публикации. |

Класс PublicationService – класс, для работы с базой данных через PublicationRepository. В таблице 2.7 представлены методы класса.

Таблица 2.7 – методы класса PublicationService

|  |  |
| --- | --- |
| метод | Описание |
| AddAsync(in publication:Publication): Task<Publication | Асинхронный метод, предназначенный для добавления публикации в базу данных. |
| AddRangeAsync(in publications:IEnumerable<Publication>): Task | Асинхронный метод, предназначенный для добавления нескольких публикаций в базу данных. |
| AddRangeAsyncBySource(in source:Source): Task | Асинхронный метод, предназначенный для добавления всех публикаций переданного источника в базу данных. |
| DeleteAsync(in publication:Publication): Task | Асинхронный метод, предназначенный для удаления публикаций из базы данных. |
| GetAll(): IEnumerable<Publication> | Метод, предназначенный для получения всех публикаций из базы данных. |
| GetById(in id:int): Publication | Метод, предназначенный для получения публикации по первичному ключу из базы данных. |
| GetByUrl(in url:string): Publication | Метод, предназначенный для получения публикации по ссылке из базы данных. |
| UpdateAsync(in publication:Publication): Task | Асинхронный метод, предназначенный для обновления публикации в базе данных. |

*Окончание таблицы 2.7*

Класс UserService – класс, для работы с базой данных через UserRepository. В таблице 2.8 представлены методы класса.

Таблица 2.8 – методы класса UserService

|  |  |
| --- | --- |
| Метод | Описание |
| AddAsync(in user:User): Task | Асинхронный метод, предназначенный для добавления пользователя в базу данных. |
| DeleteAsync(in user:User): Task | Асинхронный метод, предназначенный для удаления пользователя из базы данных. |
| GetAll(): IEnumerable<User> | Метод, предназначенный для получения всех пользователей из базы данных. |
| GetById(in id:int): User | Метод, предназначенный для получения пользователя по первичному ключу из базы данных. |
| GetByLogin(in login:string): User | Метод, предназначенный для получения публикации по логину из базы данных. |
| UpdateAsync(in user:User): Task | Асинхронный метод, предназначенный для обновления пользователя в базе данных. |
| AddSourceByUserAsync(in source:Source, in user:User): Task | Асинхронный метод, предназначенный для добавления источника в список источников пользователя в базу данных. |

Класс SourceService – класс, для работы с базой данных через SourceRepository. В таблице 2.9 представлены методы класса.

Таблица 2.9 – методы класса SourceService

|  |  |
| --- | --- |
| Метод | Описание |
| AddAsync(in source:Source): Task | Асинхронный метод, предназначенный для добавления источника в базу данных. |
| DeleteAsync(in source:Source): Task | Асинхронный метод, предназначенный для удаления источника из базы данных. |
| DeleteRangeAsync(in sources:IEnumerable<Source>): Task | Асинхронный метод, предназначенный для удаления нескольких источников из базы данных. |
| GetAll(): IEnumerable<Source> | Метод, предназначенный для получения всех источников из базы данных. |
| GetById(in id:int): Source | Метод, предназначенный для получения источника по первичному ключу из базы данных. |
| UpdateAsync(in source:Source): Task | Асинхронный метод, предназначенный для обновления источника в базе данных. |

Класс PublicationRepository – класс, для работы с таблицей «Publications» в базе данных. В таблице 2.10 представлены методы класса. Наследуется от BaseRepository.

Таблица 2.10 – методы класса PublicationRepository

|  |  |
| --- | --- |
| Метод | Описание |
| GetByUrl(in string url): Publication | Метод, предназначенный для получения публикации по ссылке из базы данных. Работает на прямую с базой данных. |
| PublicationRepository(in AppContext appContext) | Конструктор класса PublicationRepository, в котором хранится контекст базы данных. |

Класс UserRepository – класс, для работы с таблицей «Users» в базе данных. В таблице 2.11 представлены методы класса. Наследуется от BaseRepository.

Таблица 2.11 – методы класса UserRepository

|  |  |
| --- | --- |
| Метод | Описание |
| UserRepository(in AppContext appContext) | Конструктор класса UserRepository, в котором хранится контекст базы данных. |

Класс SourceRepository – класс, для работы с таблицей «Sources» в базе данных. В таблице 2.12 представлены методы класса. Наследуется от BaseRepository.

Таблица 2.12 – методы класса SourceRepository

|  |  |
| --- | --- |
| Метод | Описание |
| SourceRepository(in AppContext appContext) | Конструктор класса SourceRepository, в котором хранится контекст базы данных. |

Класс BaseRepository – абстрактный класс, от которого наследуются все репозитории. В таблице 2.13 представлены поля класса. В таблице 2.14 представлены методы класса. Наследуется от IBaseRepository.

Таблица 2.13 – поля класса BaseRepository

|  |  |
| --- | --- |
| Поле | Описание |
| context: AppContext | Поле, хранящее контекст базы данных. |
| set: DbSet<TEntity> | Поле, хранящее таблицу сущностей. |

Таблица 2.14 – методы класса BaseRepository

|  |  |
| --- | --- |
| Метод | Описание |
| AddAsync(in entity:TEntity): Task<TEntity> | Асинхронный метод, предназначенный для добавления сущности в базу данных. |
| AddRangeAsync(in entities:IEnumerable <TEntity>): Task | Асинхронный метод, предназначенный для добавления нескольких сущностей в базу данных. |
| Delete(in entity:TEntity): void | Метод, предназначенный для удаления сущности из базы данных. |
| DeleteRange(in entities:IEnumerable<TEntity>): void | Метод, предназначенный для удаления нескольких сущностей из базы данных. |
| Update(in entity:TEntity): void | Метод, предназначенный для обновления сущности в базе данных. |
| UpdateRange(in entities:IEnumerable<TEntity>): void | Метод, предназначенный для обновления нескольких сущностей в базе данных. |
| SaveChangesAsync(): Task | Асинхронный метод, предназначенный для сохранения изменений в базе данных. |
| GetAll(): IEnumerable<TEntity> | Метод, предназначенный для получения всех сущностей из базы данных. |
| GetById(in id:int): TEntity | Метод, предназначенный для получения сущности по первичному ключу из базы данных. |

*Окончание таблицы 2.14*

Класс IBaseRepository – интерфейс, который должен реализовать BaseRepository. Наследуется от IWriteRepository и IReadRepository.

Класс IWriteRepository – интерфейс, который обобщает методы для записи в базу данных. В таблице 2.15 представлены методы класса.

Таблица 2.15 – методы класса IWriteRepository

|  |  |
| --- | --- |
| Метод | Описание |
| AddAsync(in entity:TEntity): Task<TEntity> | Асинхронный метод, предназначенный для добавления сущности в базу данных. |
| AddRangeAsync(in entities:IEnumerable <TEntity>): Task | Асинхронный метод, предназначенный для добавления нескольких сущностей в базу данных. |
| Delete(in entity:TEntity): void | Метод, предназначенный для удаления сущности из базы данных. |
| DeleteRange(in entities:IEnumerable<TEntity>): void | Метод, предназначенный для удаления нескольких сущностей из базы данных. |
| Update(in entity:TEntity): void | Метод, предназначенный для обновления сущности в базе данных. |
| UpdateRange(in entities:IEnumerable<TEntity>): void | Метод, предназначенный для обновления нескольких сущностей в базе данных. |
| SaveChangesAsync(): Task | Асинхронный метод, предназначенный для сохранения изменений в базе данных. |

*Окончание таблицы 2.15*

Класс IReadRepository – интерфейс, который обобщает методы для чтения из базы данных. В таблице 2.16 представлены методы класса.

Таблица 2.16 – методы класса IReadRepository

|  |  |
| --- | --- |
| Метод | Описание |
| GetAll(): IEnumerable<TEntity> | Метод, предназначенный для получения всех сущностей из базы данных. |
| GetById(in id:int): TEntity | Метод, предназначенный для получения сущности по первичному ключу из базы данных. |

Класс AppContext – контекст базы данных в приложении. Для конфигурации таблиц использует SourceConfig, UserConfig и PublicationConfig. В таблице 2.17 представлены свойства класс. В таблице 2.18 представлены методы класс.

Таблица 2.17 – свойства класса AppContext

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>Favourites: DbSet<Favourite> | Свойство, содержащее таблицу «Избранное» базы данных. |
| <<property>>Publications: DbSet<Publication> | Свойство, содержащее таблицу «Публикации» базы данных. |
| <<property>>ReadLater: DbSet<ReadLater> | Свойство, содержащее таблицу «Отложенное» базы данных. |
| <<property>>Sources: DbSet<Source> | Свойство, содержащее таблицу «Источники» базы данных. |
| <<property>>User: DbSet<User> | Свойство, содержащее таблицу «Пользователи» базы данных. |

*Окончание таблицы 2.17*

Таблица 2.18 – методы класса AppContext

|  |  |
| --- | --- |
| Метод | Описание |
| AppContext() | Конструктор класса AppContext. |
| OnConfiguring(in optionsBuilder:DbContextOptionsBuilder): void | Метод, предназначенный для конфигурации параметров базы данных, например, расположение и название. |
| OnModelCreating(in modelBuilder:ModelBuilder): void | Метод, в котором происходит конфигурация таблиц базы данных. |

Класс SourceConfig – класс, предназначенный для конфигурации таблицы «Sources». В таблице 2.19 представлены методы класса.

Таблица 2.19 – методы класса SourceConfig

|  |  |
| --- | --- |
| Метод | Описание |
| Configure(in builder:EntityTypeBuilder<Source>): void | Метод, предназначенный для конфигурации таблицы «Источники». |

Класс UserConfig – класс, предназначенный для конфигурации таблицы «Users». В таблице 2.20 представлены методы класса.

Таблица 2.20 – методы класса UserConfig

|  |  |
| --- | --- |
| Метод | Описание |
| Configure(in builder:EntityTypeBuilder<User>): void | Метод, предназначенный для конфигурации таблицы «Пользователи». |

Класс PublicationConfig – класс, предназначенный для конфигурации таблицы «Publications». В таблице 2.21 представлены методы класса.

Таблица 2.21 – методы класса PublicationConfig

|  |  |
| --- | --- |
| Метод | Описание |
| Configure(in builder:EntityTypeBuilder<Publication>): void | Метод, предназначенный для конфигурации таблицы «Публикации». |

Класс User – модель данных пользователя. В таблице 2.22 представлены поля класс. В таблице 2.23 представлены свойства класса.

Таблица 2.22 – поля класса User

|  |  |
| --- | --- |
| Поле | Описание |
| hashPassword: string | Поле, хранящее хешированный пароль пользователя в базе данных. |
| notificationsOn: bool | Поле для хранения параметра отправки уведомлений, который установил пользователь. |
| theme: ApplicationTheme | Поле для хранения выбранной темы приложения пользователем. |

Таблица 2.23 – свойства класса User

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>HashPassword: string | Свойство для работы с полем hashPassword. |
| <<property>>NotificationsOn: bool = false | Свойство для работы с полем notificationsOn. |
| <<property>>Theme: ApplicationTheme | Свойство для работы с полем theme. |
| <<property>>Favourites: ObservableCollection<Favourite> = [] | Свойство, хранящее коллекцию данных Favourite. |
| <<property>>FavouritesPublications: ObservableCollection<Publication> = [] | Свойство, хранящее коллекцию публикаций, добавленных в список «Избранное». |
| <<property>>ReadLater: ObservableCollection<ReadLater> = [] | Свойство, хранящее коллекцию данных ReadLater. |
| <<property>>ReadLaterPublications: ObservableCollection<Publication> = [] | Свойство, хранящее коллекцию публикаций, добавленных в список «Отложенное». |
| <<property>>Sources: ObservableCollection<Source> = [] | Свойство, хранящее коллекцию добавленных пользователем источников. |

*Окончание таблицы 2.23*

Класс SHA256Hasher – класс, предназначенный для хеширования пароля пользователя. В таблице 2.24 представлены методы класса.

Таблица 2.24 – методы класса SHA256Hasher

|  |  |
| --- | --- |
| Метод | Описание |
| <<static>>Hash(in input:string): string | Статический метод для хеширования строки. |

Класс Publication – модель данных онлайн-публикаций. В таблице 2.25 представлены свойства класс.

Таблица 2.25 – свойства класса Publication

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>Favourites: ObservableCollection<Favourite> = [] | Свойство, хранящее коллекцию данных Favourite. |
| <<property>>Link: string | Свойство для хранения ссылки на публикацию. |
| <<property>>PublishingDate: DateTime? | Свойство для хранения даты выхода публикации. |
| <<property>>ReadLater: ObservableCollection<ReadLater> = [] | Свойство, хранящее коллекцию данных ReadLater. |
| <<property>>UserFavourites: ObservableCollection<User> = [] | Свойство для хранения пользователей, у которых публикация добавлена в «Избранное». |
| <<property>>UserReadLater: ObservableCollection<User> = [] | Свойство для хранения пользователей, у которых публикация добавлена в «Отложенно». |
| <<property>>Source: Source | Свойство для хранения источника, к которому относится публикация. |
| <<property>>SourceId: int | Свойство для хранения первичного ключа источника, к которому относится публикация. |
| <<property>>Title: string | Свойство для заголовка публикации. |

*Окончание таблицы 2.25*

Класс Source – модель данных источников. В таблице 2.26 представлены поля класс. В таблице 2.27 представлены свойства класса.

Таблица 2.26 – поля класса Source

|  |  |
| --- | --- |
| Поле | Описание |
| url: string | Поле для хранения ссылки на источник. |

Таблица 2.27 – свойства класса Source

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>Url: string | Свойство для работы с полем url. |
| <<property>>Description: string | Свойство для хранения описания источника. |
| <<property>>ImageUrl: string | Свойство для хранения ссылки на изображения источника. |
| <<property>>Publications: ObservableCollection<Publication> = [] | Свойство для хранения всех публикаций источника. |
| <<property>>Title: string | Свойство для хранения названия источника. |
| <<property>>Users: ObservableCollection<User> = [] | Свойство для хранения пользователей, у которых есть данный источник. |

Класс Favourite – модель избранной публикации пользователя. В таблице 2.28 представлены свойства класса.

Таблица 2.28 – свойства класса Favourite

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>Publication: Publication | Свойство для хранения публикации. |
| <<property>>PublicationId: int | Свойство для первичного ключа публикации. |
| <<property>>User: User | Свойство для хранения пользователя. |
| <<property>>UserId: int | Свойство для хранения первичного ключа пользовалея. |

Класс ReadLater – модель отложенной публикации пользователя. В таблице 2.29 представлены свойства класса.

Таблица 2.29 – свойства класса ReadLater

|  |  |
| --- | --- |
| Свойство | Описание |
| <<property>>Publication: Publication | Свойство для хранения публикации. |
| <<property>>PublicationId: int | Свойство для первичного ключа публикации. |
| <<property>>User: User | Свойство для хранения пользователя. |
| <<property>>UserId: int | Свойство для хранения первичного ключа пользовалея. |

# 3 ОБОСНОВАНИЕ ВЫБОРА СРЕДСТВ РЕАЛИЗАЦИИ

## 3.1 Выбор средств реализации

В качестве средств реализации клиент-серверного приложения были выбраны следующие инструменты:

* Язык программирования С# 12, .NET 8.0 и WPF.
* SQLite 3 для хранения данных в базе данных.
* Пакеты: CodeHollow.FeedReader для работы с RSS, FluentWPF для создания приложений в стиле системы Windows 10, Hardcodet.NotifyIcon.Wpf для сворачивания приложения в панель задач, MahApps.Metro для применения стилей, Microsoft.EntityFrameworkCore для работы с базой данных, Microsoft.EntityFrameworkCore.Sqlite для работы с базой данных SQLite, Microsoft.Toolkit.Uwp.Notifications для создания уведомлений, Microsoft.Web.WebView2 для отображения страницы онлайн-публикации, ModernWpfUI для созданий окон и страниц приложения, ModernWpfUI.MahApps для связывания ModernWpfUI с MahApps.Metro.

C# — это кроссплатформенный язык общего назначения, который делает разработчиков продуктивным при написании высокопроизводительного кода. С миллионами разработчиков C# является самым популярным языком .NET. C# имеет широкую поддержку в экосистеме и всех рабочих нагрузок .NET. На основе объектно-ориентированных принципов он включает множество функций из других парадигм, а не наименее функционального программирования. Низкоуровневые функции поддерживают сценарии высокой эффективности без написания небезопасного кода. Большая часть среды выполнения и библиотек .NET написана на C#, и прогресс в C# часто используется для всех разработчиков .NET [[2](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)].

WPF (Windows Presentation Foundation) – это технология в платформе .NET, позволяющая создавать графические интерфейсы. В отличие от традиционных приложений на базе WinForms, которые используют User32 и GDI+ для рендеринга элементов управления и графики, приложения WPF используют DirectX. Это означает, что большая часть работы по отрисовке графики, от простых кнопок до сложных 3D-моделей, выполняется графическим процессором на видеокарте, что позволяет использовать аппаратное ускорение. Одной из примечательных особенностей WPF является использование XAML, декларативного языка разметки интерфейса на основе XML. Это позволяет разработчикам создавать богатые интерфейсы, используя либо декларативное объявление интерфейса, либо код на C#, VB.NET, F#, либо их комбинацию. WPF был впервые выпущен в 2006 году вместе с .NET Framework 3.0 и Windows Vista. С тех пор он стал частью экосистемы .NET и развивался вместе с ней. Последняя версия фреймворка .NET, .NET 8, полностью поддерживает WPF [[3](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)].

SQLite – это широко распространенный движок реляционных баз данных, который можно найти на различных устройствах и платформах, включая Android, iOS, Mac и Windows. Он также используется в популярных браузерах, таких как Firefox, Chrome и Safari. В отличие от других систем баз данных, SQLite не требует отдельного сервера и может напрямую обращаться к файлу базы данных на диске. Это устраняет необходимость в установке или настройке. SQLite поддерживает основные функции, присутствующие в других реляционных СУБД, такие как таблицы, индексы, триггеры и представления. В нем используется язык SQL, который похож на другие реализации и диалекты, используемые в других реляционных СУБД. Формат файлов базы данных является кроссплатформенным, что позволяет легко переносить файлы между различными операционными системами. Кроме того, SQLite поддерживается такими распространенными языками программирования, как Python, C# и Java, что делает его универсальным для различных сценариев разработки приложений [[4](#_СПИСОК_ИСПОЛЬЗОВАННЫХ_ИСТОЧНИКОВ)].

# ЗАКЛЮЧЕНИЕ

Во время прохождения практики были выполнены следующие задачи:

* 1. Проведен анализ предметной области;
  2. Разработана спецификация требований к системе;
  3. Создана архитектура системы;
  4. Спроектирована структура таблицы для сохранения данных;
  5. Описан выбор средств реализации;
  6. Реализованы классы программы.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Простая архитектура с использованием MVVM [Электронный ресурс] / Хабр : сайт. Электрон. текстовые дан. – Режим доступа: <https://habr.com/ru/articles/776344/>, свободный - Загл. с экрана.
2. Краткий обзор языка C# [Электронный ресурс] / Microsoft : сайт. Электрон. текстовые дан. – Режим доступа: <https://learn.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/overview>, свободный – Загл. с экрана.
3. C# и WPF | Введение [Электронный ресурс] / Metanit : сайт. Электрон. текстовые дан. – Режим доступа: <https://metanit.com/sharp/wpf/1.php>, свободный - Загл. с экрана.
4. SQLite | Введение [Электронный ресурс] / Metanit : сайт. Электрон. текстовые дан. – Режим доступа: <https://metanit.com/sql/sqlite/1.1.php>, свободный – Загл. с экрана.