**Содержание**

[Введение 4](#_Toc191651163)

[1. Предварительное планирование и проверка осуществимости 5](#_Toc191651164)

[1.1 Требования к системе в целом 5](#_Toc191651165)

[1.2 Требования к функциям, выполняемым системой 7](#_Toc191651166)

[1.3 Требования к видам обеспечения 8](#_Toc191651167)

[1.4 Сравнение и анализ аналогов системы 9](#_Toc191651168)

[1.5 Инфологическая модель предметной области 13](#_Toc191651169)

[1.6 Прототипирование интерфейса 14](#_Toc191651170)

[2. Практическая часть 19](#_Toc191651171)

[2.1 Назначение и цели создания системы 19](#_Toc191651172)

[2.2 Характеристики объекта автоматизации 19](#_Toc191651173)

[2.3 Обзор инструментальных средств создания продукта 20](#_Toc191651174)

[2.4 Обоснование выбора средств реализации ИС 23](#_Toc191651175)

[2.5 Описание логики функционирования ПО 23](#_Toc191651176)

[2.6 Реализация интерфейса ПО 27](#_Toc191651177)

[2.7 Написание программного кода 31](#_Toc191651178)

[Заключение 34](#_Toc191651179)

[Список использованных источников 35](#_Toc191651180)

[Приложение 36](#_Toc191651181)

# Введение

В современном мире информационные технологии играют важную роль в оптимизации процессов управления и повышения эффективности работы государственных учреждений. Администрации, как важные элементы местного самоуправления, сталкиваются с необходимостью внедрения современных решений для автоматизации документооборота, управления данными и взаимодействия с населением.

Разработка модуля информационной системы направлена на улучшение внутренних процессов администрации, автоматизация процессов учёта документов и введения журнала.

Актуальность темы – потребность автоматизации учёта поступающих документов и введение журналов.

Цель – разработка модуля информационной системы для администрации сельского поселения.

Объект – деятельность архивного отдела администрации сельского поселения.

Предмет – модуль информационной системы для администрации сельского поселения.

Для достижения цели, требуется выполнить следующие задачи:

1. Провести предварительное планирование и проверку осуществимости.
2. Определить требования к программному продукту.
3. Разработать прототип приложения.
4. Реализовать приложение в выбранной среде IDE.
5. Разместить проект на веб-сервисе GitHub или аналогичном.

# Предварительное планирование и проверка осуществимости

# 1.1 Требования к системе в целом

Требования к структуре и функционированию системе

Система должна состоять из сервера баз данных, приложение.

Сервер БД отвечает за хранение базы данных, а также организует работу приложения и обмен данными с сервером баз данных. Приложение – основная часть системы – обрабатывает данные полученные от сервера БД и выводит их в понятном для конечного пользователя виде, а также формирует запросы от пользователя к серверу БД.

Требования к численности и квалификации персонала

Пользователями приложения являются глава администрации сельского поселения, заместитель главы администрации, а также секретарь. Пользователи используют систему через приложение, поэтому количество людей желающих просмотреть приложение или воспользоваться услугами ограниченно.

Обслуживающий персонал:

• IT-специалист (системный администратор)– 1 человек;

Пользователи должны обладать определенными навыками при работе с приложение, а именно работа с приложением Visual Studio (С#, WPF).

Обслуживающий персонал должен иметь навыки работы с операционными системами (Windows, Linux, и др.), работа с приложением (Visual Studio 2022).

Системный администратор должен иметь опыт работы с SQL-server.

Требования к надежности

Программное обеспечение системы должно функционировать и не приводить к зависанию системы при любых ошибочных действиях операторов в среде данного программного обеспечения.

Требования к эргономике и технической эстетике

Программное обеспечение и технические средства должны иметь эстетически привлекательный вид и быть удобными в использовании.

Требования к эксплуатации и техническому обслуживанию

Для нормальной работы системы необходимо произвести ряд действий

На компьютере:

• установить Microsoft SQL Server;

• установить программу Visual Studio;

На сервере базы данных:

• запустить программу установки SQL-server.

• импортировать базу данных;

• запустить программу установки сервера БД.

Требования к допустимым площадям размещения ИС, к параметрам сети энергоснабжения и т. д. определяются требованиями использующегося аппаратного обеспечения.

Требования к защите информации от несанкционированного доступа

Защита ИС и информации, с которой она работает, от несанкционированного доступа возлагается операционную систему, частично на систему аутентификации. В процессе аутентификации имя пользователя и его пароль передаются через базу данных SQL server. Пароли хранятся в базе данных в зашифрованном виде.

Требования по сохранности информации при авариях

Сохранность данных должна быть обеспеченна:

• при сбоях в работе оборудования сети в любой момент времени;

• при сбоях электропитания или программного обеспечения не ведущих к физическому уничтожению или повреждению информации.

Требования к защите от влияния внешних воздействий

Автоматизированная система должна предоставлять пользователю привычный, общепринятый в среде Microsoft Windows интерфейс. Программная документация, поставляемая с программой, должна быть оформлена в соответствии со стандартом ЕСПД.

Требования к стандартизации и унификации

Автоматизированная система должна предоставлять пользователю привычный, общепринятый в среде Microsoft Windows интерфейс. Программная документация, поставляемая с программой, должна быть оформлена в соответствии со стандартом ЕСПД.

# 1.2 Требования к функциям, выполняемым системой

Функции подсистемы «Сервер базы данных»:

• получение запросов от «клиента БД», расположенного на SQL Server;

• проверка легальности выполнения запроса на основе идентификационной информации, вложенной в запрос;

• выполнение SQL-запросов к БД;

• передача результата SQL-запроса «клиенту БД».

Функции подсистемы «Приложение»:

• предоставление информации об определенных документах;

• предоставление возможности просмотра архива;

• возможность добавления нового документа;

# 1.3 Требования к видам обеспечения

Требования к математическому обеспечению

В основе разрабатываемого ПО должна лежать реляционная модель базы данных. Все таблицы должны находиться в третьей или четвертой нормальной форме. Связи между таблицами должны быть вида: «один к одному» или «один ко многим». При проектировании используется метод «сущность – связь».

Требования к информационному обеспечению

Данные в системе должны быть организованны в виде связанных таблиц. Хранение и доступ к данным осуществляется с применением реляционные систем управления базами данных. Информационный обмен между компонентами системы осуществляется по технологии клиент-сервер. Желательно обеспечить возможность резервного копирования и восстановления данных.

Требования к лингвистическому обеспечению

В качестве языка программирования используются – С#, SQL Server, взаимодействие пользователя и ПО осуществляется через оконный интерфейс.

Требования к программному обеспечению

Система должна иметь следующее программное обеспечение: ОС Windows 10 или выше, SQL Server, библиотека Net Framework 4.8.

Требования к техническому обеспечению

Для работоспособности приложения необходимы следующие программные и технические средства:

• клавиатура;

• манипулятор мышь;

• процессор 10th Gen Intel(R) Core(TM)

• оперативная память 8 ГБ и больше;

• версия Windows 7 и выше;

# 1.4 Сравнение и анализ аналогов системы

Мною были рассмотрены аналоги приложений для работы с документами и выделила преимущества и недостатки.

* 1С: Документооборот

<https://edo.1c.ru/>

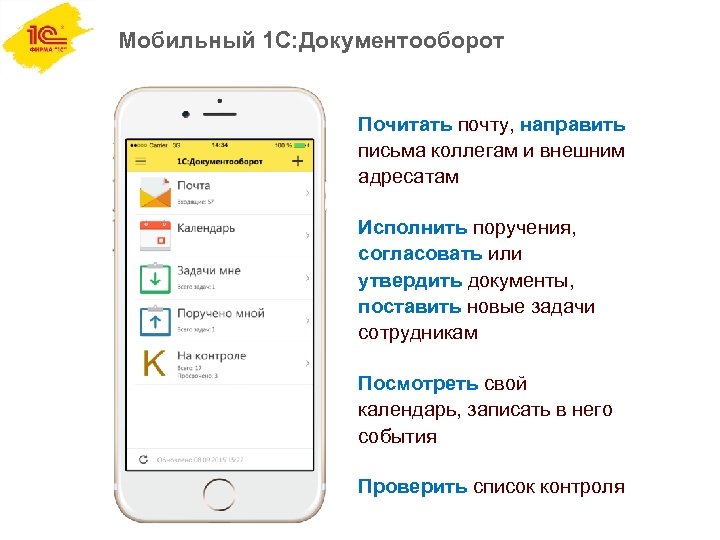


Рисунок 1.1 - Интерфейс системы «1С: Документооборот»

1С: Документооборот — это программное решение, разработанное компанией 1С, предназначенное для автоматизации процессов документооборота в организациях. Оно позволяет эффективно управлять созданием, обработкой, хранением и передачей документов, а также упрощает согласование и контроль за выполнением задач.

Преимущества:

* Автоматизация процессов: система позволяет автоматизировать рутинные задачи, связанные с документооборотом, что значительно ускоряет рабочие процессы.
* Упрощение согласования: встроенные механизмы согласования документов позволяют быстро и удобно организовать процесс согласования между различными подразделениями.
* Интеграция с другими системами: 1С: Документооборот может интегрироваться с другими продуктами 1С, а также с внешними системами, что позволяет создать единое информационное пространство.

Недостатки:

* Сложности с настройкой: настройка системы под специфические бизнес-процессы может потребовать значительных усилий и времени.
* Ограниченный функционал: в некоторых случаях функционал может быть сложной, особенно если используются разные версии программного обеспечения.
* Необходимость обучения: несмотря на интуитивный интерфейс, пользователям может потребоваться время для освоения всех возможностей системы.
* WPF Office

<https://www.wps.com/>

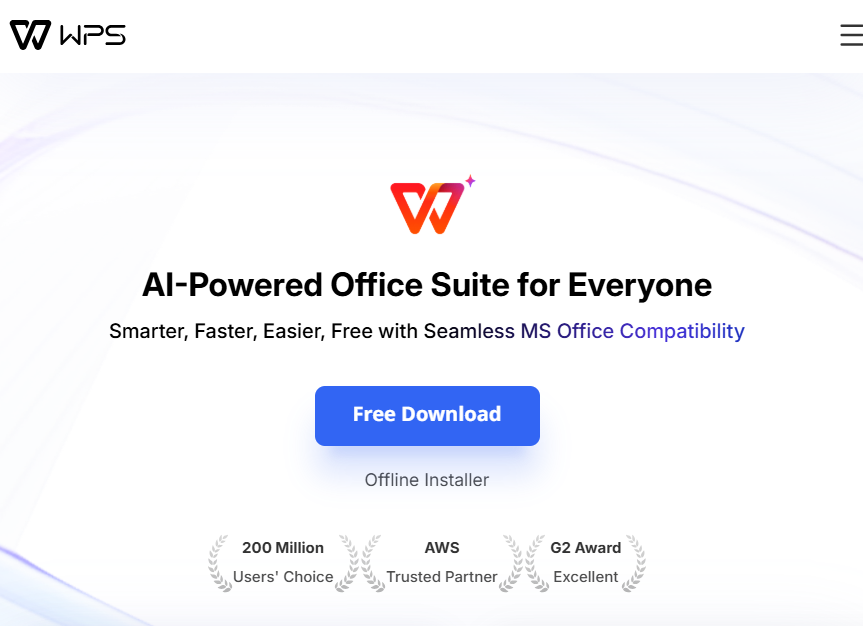


Рисунок 1.2 - Интерфейс системы «WPF Office»

WPF Office относится к использованию Windows Presentation Foundation (WPF) для разработки пользовательских интерфейсов в приложениях Microsoft Office. Это позволяет создавать более современные и интерактивные интерфейсы, используя возможности WPF, такие как привязка данных и стилизация. WPF Office предоставляет разработчикам инструменты для создания приложений, которые могут интегрироваться с Microsoft Office, улучшая пользовательский опыт.

Преимущества:

* Интерактивность: позволяет создавать более динамичные и отзывчивые интерфейсы, что улучшает пользовательский опыт.
* Привязка данных: упрощает работу с данными, позволяя автоматически обновлять интерфейс при изменении данных.
* Стилизация и темизация: возможность применения различных стилей и тем, что позволяет создавать уникальные интерфейсы.
* Поддержка графики: возможность использования векторной графики и анимаций, что делает интерфейсы более привлекательными.

Недостатки:

* Сложность в обучении: требует значительных знаний и опыта в WPF и .NET, что может быть препятствием для новичков.
* Производительность: в некоторых случаях может быть менее производительным по сравнению с традиционными WinForms приложениями, особенно при сложных интерфейсах.
* Совместимость: возможны проблемы с совместимостью на старых версиях Windows или Office, что может ограничить аудиторию пользователей.
* Сложность разработки: разработка может быть более сложной и требовать больше времени на реализацию по сравнению с другими технологиями.
* LibreOffice

<https://ru.libreoffice.org/>



Рисунок 1.3 - Интерфейс системы «LibreOffice»

LibreOffice — это бесплатный и открытый офисный пакет, который предоставляет пользователям инструменты для работы с текстами, таблицами, презентациями и другими документами.

Преимущества:

* Бесплатность: LibreOffice является полностью бесплатным, что делает его доступным для всех пользователей.
* Функциональность: обеспечивает широкий набор инструментов для работы с документами, что делает его подходящим для большинства офисных задач.
* Сообщество: активное сообщество разработчиков и пользователей, которое поддерживает и развивает проект.

Недостатки:

* Совместимость: иногда возникают проблемы с совместимостью форматов, особенно с документами Microsoft Office, что может привести к потере форматирования.
* Интерфейс: Интерфейс может показаться менее интуитивным для новых пользователей по сравнению с Microsoft Office.
* Ограниченные функции: Некоторые специализированные функции, доступные в Microsoft Office, могут отсутствовать или быть менее развитыми в LibreOffice.

В ходе анализа и сравнений схожих информационных систем я пришла к выводу, что для администрации сельского поселения целесообразно разработать собственную информационную систему. Она будет отличаться простотой использования и высокой доступностью, что особенно важно для работников с разным уровнем компьютерной грамотности.

# 1.5 Инфологическая модель предметной области

Мною была разработана инфологическая модель в которой администратор имеет полный контроль над всеми аспектами системы, он отвечает за общую конфигурацию, настройку параметров, установку обновлений и обеспечение бесперебойной работы.

Инфологическая модель изображена на рисунке 1.4.

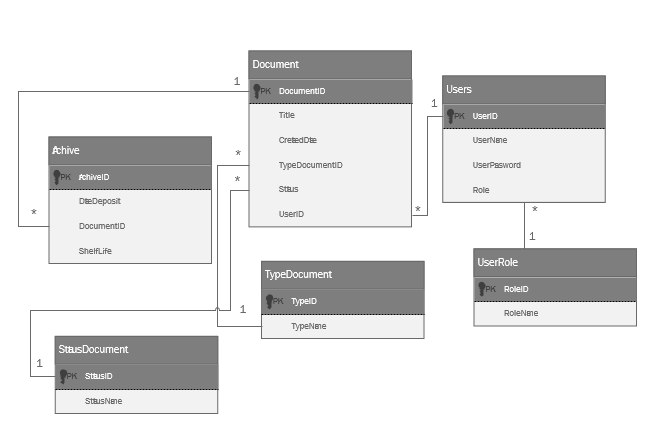


Рисунок 1.4 - Инфологическая модель

# 1.6 Прототипирование интерфейса

Модуль информационной системы будет состоять из следующих компонентов и страниц:

• окно авторизации;

• главная страница;

• страница со списком документов;

• страница с добавлением документа;

• страница с редактирование документа;

• страница с журналом документов (архивом);

Страница авторизации (рисунок 1.5) является прототипом для будущего приложения. На ней содержатся поля для ввода логина и пароля, фон, а также кнопка входа.

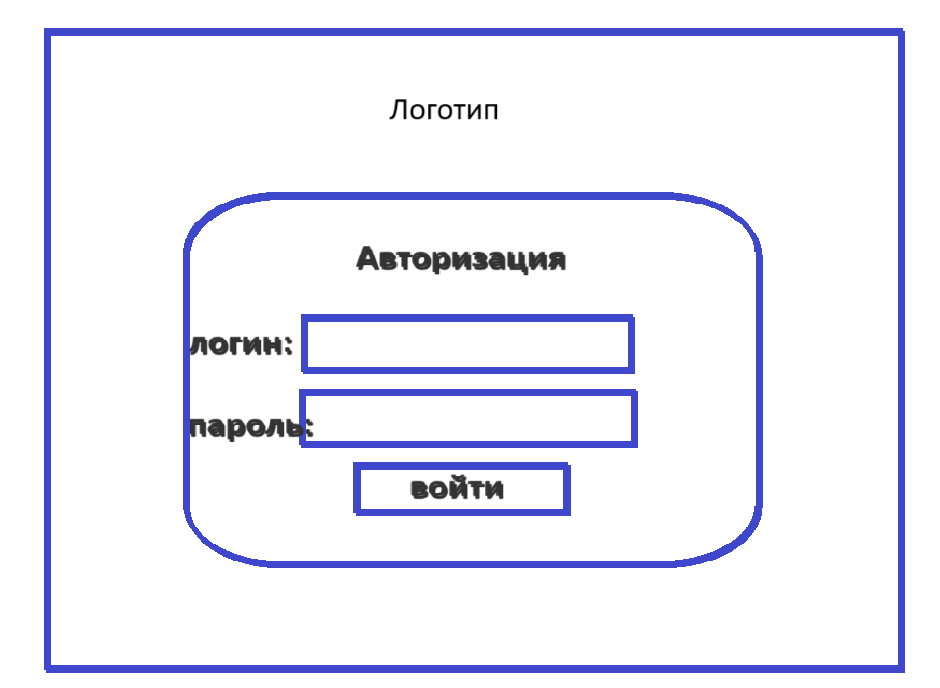


Рисунок 1.5 - Прототип страницы «Авторизация»

Главная страница на рисунке 1.6 содержит название, также будет отображаться горизонтальная панель для перехода к списку документов и архиву.

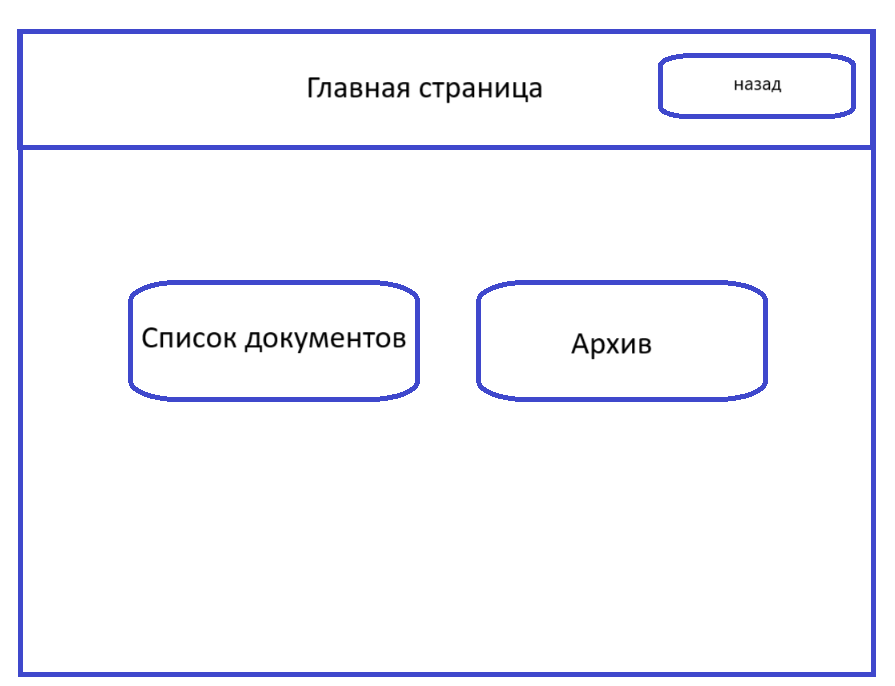


Рисунок 1.6 - Прототип страницы «Главная страница»

Прототип страницы со списком документов на рисунке 1.7. состоит из названия (заголовка) и таблицей с информацией о документе, а также кнопка создания и редактирования.

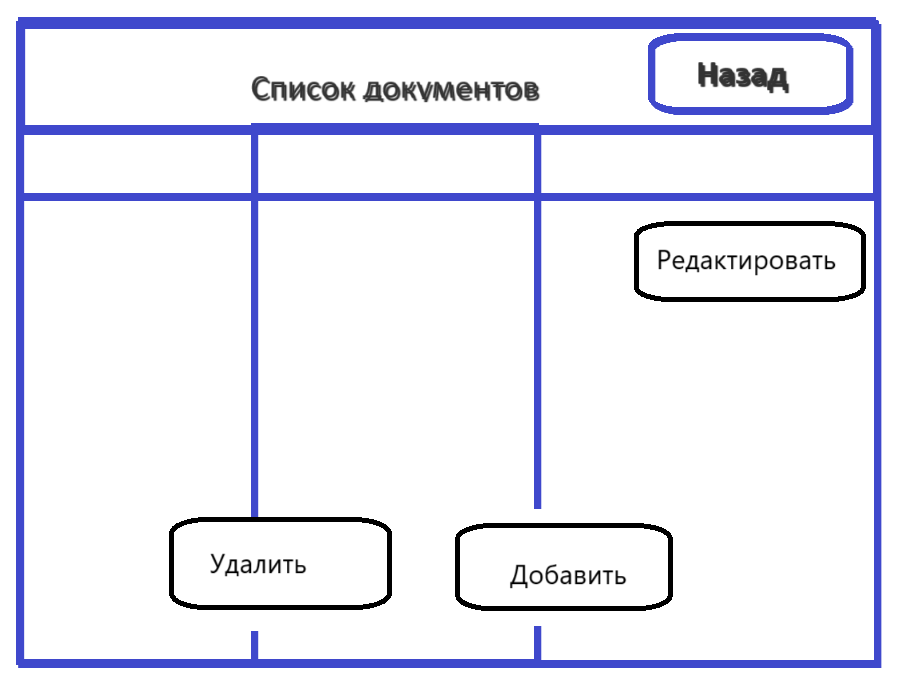


Рисунок 1.7 - Прототип страницы «Список документов»

Прототип страницы с добавление документа на рисунке 1.8. состоит из названия документа, дата создания, дата хранения, тип документа, статус документа.

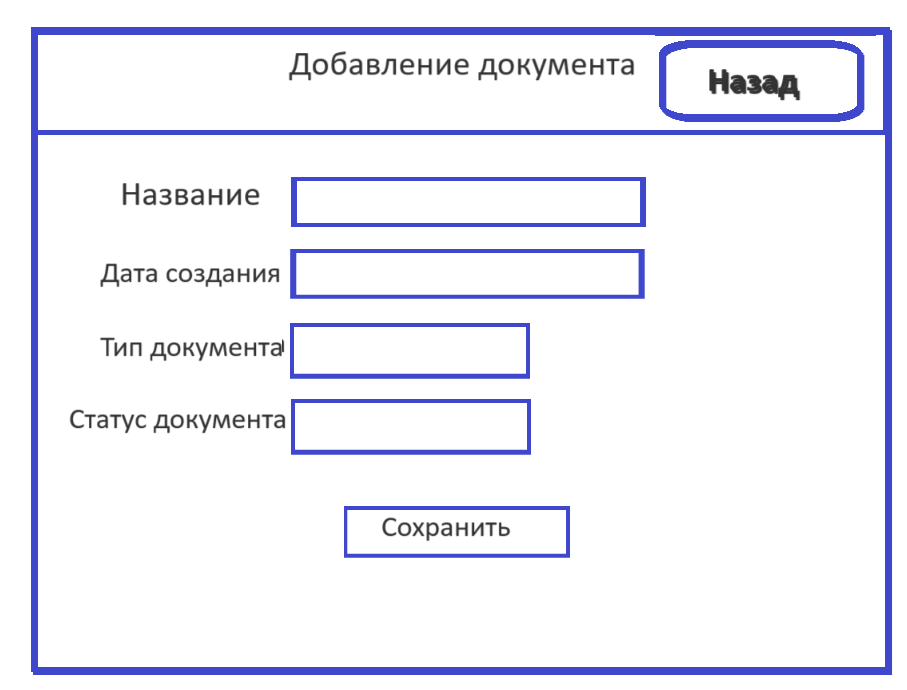


Рисунок 1.8. - Прототип страницы «Добавление документа»

Прототип страницы редактирование документа на рисунке 1.9. состоит из названия документа, дата создания, дата хранения, тип документа, статус документа.

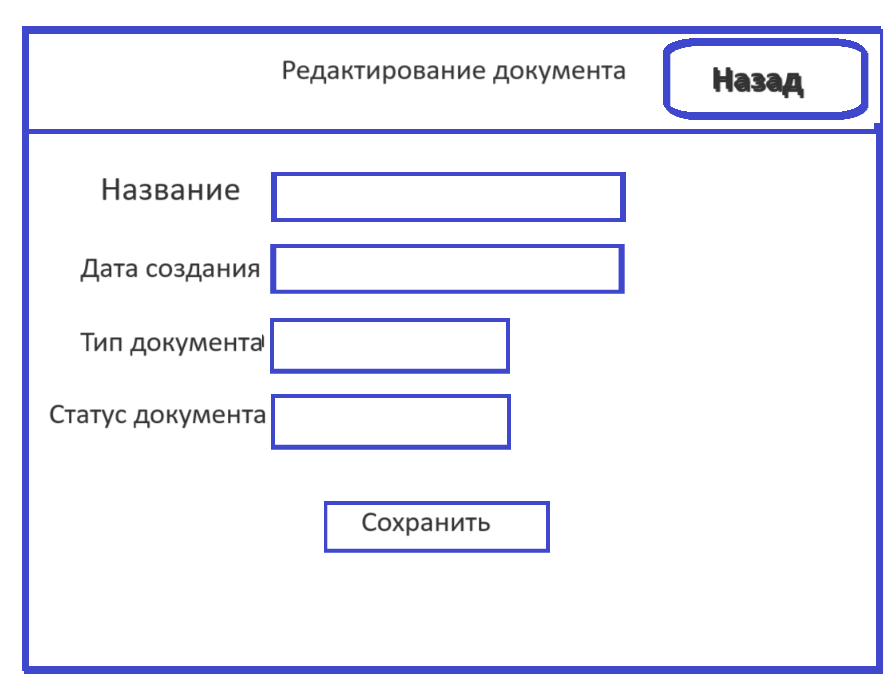


Рисунок 1.9. - Прототип страницы «Редактирование документа»

Прототип страницы архива на рисунке 1.10. состоит из таблицы где будут отображаться данные о документах которые будут хранится.

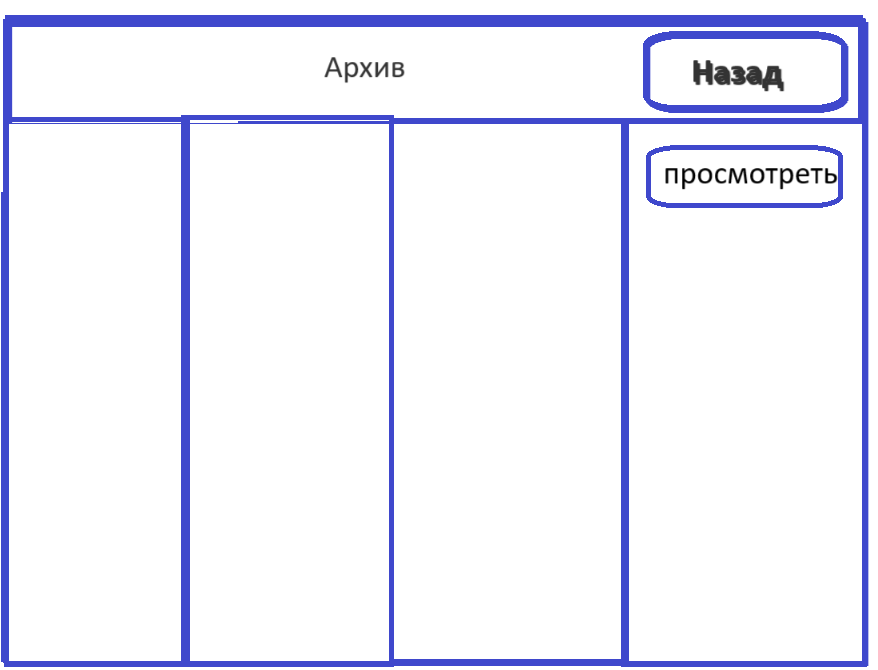


Рисунок 1.10 - Прототип страницы «Архив»

Прототип страницы просмотр документа архива на рисунке 1.11 состоит из названия документа, дата внесения , срок хранения.

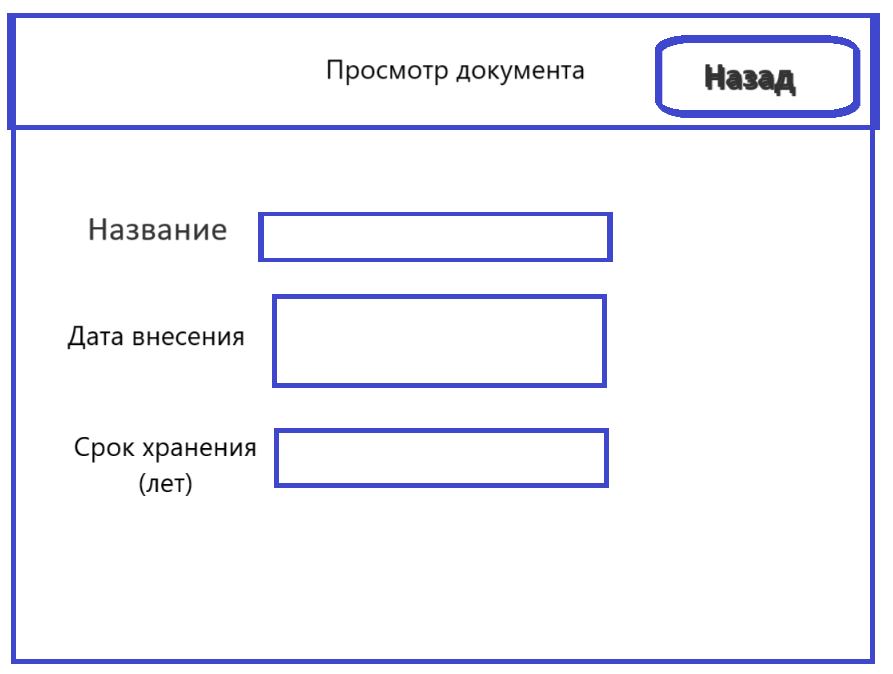


Рисунок 1.11- Прототип страницы «Просмотр документа»

# 2. Практическая часть

# 2.1 Назначение и цели создания системы

Полное наименование – «Разработка модуля информационной системы для администрации сельского поселения»

Назначение системы

Назначение создания модуля информационной системы для администрации сельского поселения заключается в автоматизации процесса учёта документов, введение журнала.

Цели создания приложения:

• автоматизация введения журнала документов;

• автоматизация учёта документов;

• повышение доступности предоставленной информации.

# 2.2 Характеристики объекта автоматизации

Краткие сведения об объекте автоматизации

Объекты автоматизации:

• процесс получения информации о документах;

• процесс редактирования и удаление записей;

• процесс обеспечения доступа к архивированным документам;

В процессе выдачи информации о документах должны отображаться следующие данные:

• название документа (дела, акта);

• дата создания.

• тип документа.

• статус документа.

В процессе создания и редактирования информации о документах должны отображаться следующие данные:

• название документа (дела, акта);

• дата создания.

• тип документа.

• статус документа.

В процессе выдачи информации об архивированных документах должно отображаться следующее:

• дата внесения.

• документ.

• срок хранения.

Сведения об условиях эксплуатации объекта автоматизации и характеристик окружающей среды

Условия эксплуатации объекта автоматизации (на стороне БД):

• операционная система MS Windows 10/11;

• SQL Server;

Условия эксплуатации объекта автоматизации (на стороне клиента)

• клавиатура;

• манипулятор мышь;

• процессор 10th Gen Intel(R) Core(TM)

• оперативная память 8 ГБ и больше;

• версия Windows 7 и выше;

Физические условия эксплуатации и характеристики окружающей среды определяются соответствующими характеристиками окружающей среды используемого аппаратного обеспечения, указанными в документации на него.

# 2.3 Обзор инструментальных средств создания продукта

Мною были рассмотрены несколько инструментальных средства для создания приложения.

Visual Studio- это интегрированная среда разработки (IDE) от Microsoft, предназначенная для разработки приложений на различных языках программирования, включая C#. Она предоставляет мощные инструменты для написания, отладки и тестирования кода.

Преимущества:

* Интуитивно понятный интерфейс: Удобный интерфейс с возможностью настройки под нужды разработчика.
* Инструменты отладки: Мощные инструменты для отладки, включая точки останова, просмотр переменных и анализ стека вызовов.
* Поддержка различных типов проектов: Возможность создания веб-приложений, настольных приложений, мобильных приложений и игр.

Недостатки:

* Требовательность к ресурсам: может требовать много системных ресурсов, особенно на старых компьютерах.
* Сложность для новичков: большое количество функций и настроек может быть сложным для начинающих разработчиков

Visual Studio Code (VCode)- это легковесный редактор кода от Microsoft, который поддерживает множество языков программирования и предоставляет мощные инструменты для разработки.

Преимущества:

* Легковесность: быстрый запуск и низкие системные требования, что делает его идеальным для небольших проектов и быстрого редактирования кода.
* Расширяемость: поддержка множества расширений, которые позволяют добавлять новые функции, такие как поддержка языков, инструменты отладки и интеграция с системами контроля версий.

Недостатки:

* Ограничение возможности: не предоставляет всех функций полноценной IDE, таких как мощные инструменты отладки и визуальный дизайнер.
* Необходимость настройки: для полноценной работы может потребоваться установка и настройка множества расширений.

Windows Forms (WF) - это фреймворк для создания настольных приложений на платформе .NET. Он предоставляет набор компонентов для создания графического интерфейса пользователя (GUI).

Преимущества:

* Простота разработки: WF предлагает визуальный дизайнер, который позволяет быстро создавать интерфейсы, перетаскивая элементы на форму.
* Быстрая разработка: Идеален для создания простых и средних приложений, где не требуется сложная графика и анимация.

Недостатки:

* Ограничение возможности интерфейса: не поддерживает современные графические элементы и анимацию, что может ограничивать дизайн приложения.
* Устаревание: С развитием технологий и появлением WPF и других фреймворков WF становится менее актуальным для новых проектов.

Windows Presentation Foundation (WPF) — это более современный фреймворк для создания настольных приложений на платформе .NET, который использует XAML (расширяемый язык разметки) для определения интерфейса.

Преимущества:

* Современный интерфейс: позволяет создавать привлекательные и интерактивные пользовательские интерфейсы с использованием стилей, шаблонов и анимации.
* Поддержка MVVM (Model View ViewModel): WPF поддерживает архитектурный паттерн.

Недостатки:

* Сложность разработки: может потребовать больше времени на изучение и разработку по сравнению с WF, особенно для новичков.
* Требования к ресурсам: приложения на WPF могут потреблять больше ресурсов, чем приложения на WF, особенно при использовании сложных графических элементов.

# 2.4 Обоснование выбора средств реализации ИС

Я решила использовать Visual Studio для разработки, так как она представляет мощные инструменты для создания, отладки и тестирования приложений, а также поддерживает различные технологии и платформы, что позволяет эффективно разрабатывать как настольные, так и веб-приложения. Кроме того, интеграция с системами контроля версий и наличие визуального дизайнера упрощает процесс разработки и командную работу.

Для разработки можно использовать любой язык, совместимый с .NET: C#, Python и другие. Я выбрала C#, так как он прост в освоении и безопасен, кроме того, данный язык облегчает разработку и поддержку приложений на платформе WPF, а также обеспечивает быструю работу приложений благодаря оптимизированному коду и поддержке JIT-компиляции. Также я выбрала сервер базы данных SQL Server так, как он известен своей стабильностью и способностью обрабатывать большие объемы данных, что особенно важно для приложений, работающих с интенсивными нагрузками.

# 2.5 Описание логики функционирования ПО

В ходе разработки курсового проекта для приложения была разработана модель базы данных, которая представлена на рисунке 2.1.

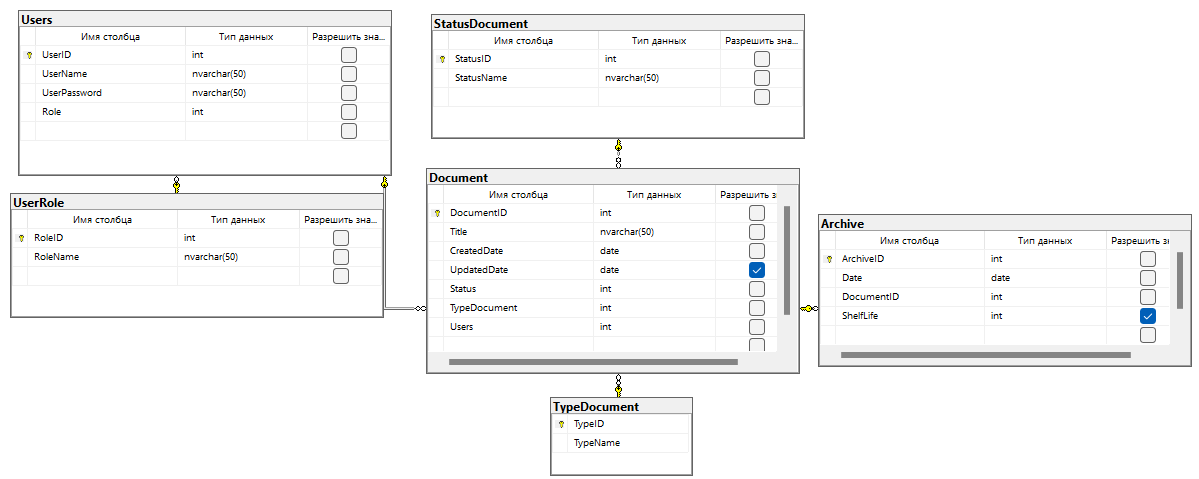


Рисунок 2.1 – Модель базы данных

Таблица «Users» хранит информацию о логинах и паролях пользователях администрации. Данные таблицы «Users» изображены на рисунке 2.2.

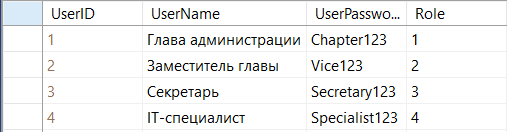


Рисунок 2.2 - Данные таблицы «Users»

Таблица 1 – Описание структуры «Users»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| UserID | int | Идентификатор пользователей |
| UserName | nvarchar(50) | Логин пользователя |
| UserPassword | nvarchar(50) | Пароль пользователя |
| Role | int | Идентификатор роли пользователя |

Таблица «UserRole» хранит информацию о роли пользователя в системе. Данные таблицы «UserRole» изображены на рисунке 2.3.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.3 - Данные таблицы «UserRole»

Таблица 2 – Описание структуры «UserRole»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| RoleID | int | Идентификатор роли пользователя |
| RoleName | nvarchar(50) | Название роли пользователя |

Таблица «StatusDocument» хранит информацию о статусе документов. Данные таблицы «StatusDocument» изображены на рисунке 2.4.

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.4 - Данные таблицы «StatusDocument»

Таблица 3 – Описание структуры «StatusDocument»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| StatusID | int | Идентификатор статуса документа |
| StatusName | nvarchar(50) | Название статуса документа |

Таблица «TypeDocument» хранит информацию о типах документов. Данные таблицы «TypeDocument» изображены на рисунке 2.5.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.5 - Данные таблицы «TypeDocument»

Таблица 4 – Описание структуры «TypeDocument»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| TypeID | int | Идентификатор типа документа |
| TypeName | nvarchar(50) | Название типа |

Таблица «Document» хранит информацию о документах, например название, дата создания, дата хранения, статус документа, тип документа, пользователь. Данные таблицы «Document» изображены на рисунке 2.6.

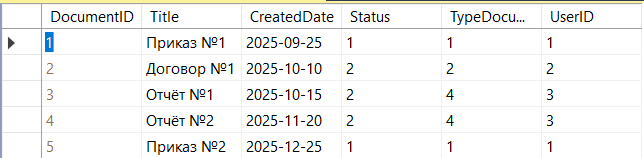


Рисунок 2.6 - Данные таблицы «Document»

Таблица 5 – Описание структуры «Document»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| DocumentID | int | Идентификатор документа |
| Title | nvarchar(50) | Название документа |
| CreatedDate | date | Дата создания документа |
| Status | int | Идентификатор статуса документа |
| TypeDocumentID | int | Идентификатор типа документа |
| UserID | int | Идентификатор пользователя |

Таблица «Archive» хранит информацию об архиве, где записываются данные о хранения документов. Данные таблицы «Archive» изображены на рисунке 2.7.

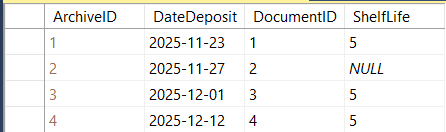


Рисунок 2.7 - Данные таблицы «Archive»

Таблица 6 – Описание структуры таблицы «Archive»

|  |  |  |
| --- | --- | --- |
| Название | Тип данных | Описание |
| ArchiveID | int | Идентификатор архива |
| DateDeposit | Date | Дата внесения |
| DocumentID | int | Идентификатор документов |
| ShelfLife | int | Срок хранения |

# 2.6 Реализация интерфейса ПО

В ходе разработки курсового проекта была выбрана цветовая модель, показанная на рисунке 2.8. В качестве основного фона используется светло серый цвет, а в качестве дополнительного: RGB (245,245,245).

Изображение выглядит как Красочность, Графика, снимок экрана, графический дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.8 – Цветовая модель

Для акцентирования внимания пользователя на интерфейс использован цвет RGB (96,116,173).

Также для был разработан логотип для моей информационной системы. Итоговый логотип, использованный в системе показан на рисунке 2.9.

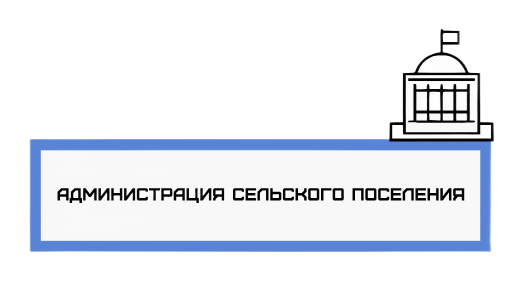


Рисунок 2.9 - Логотип приложения «Администрация сельского поселения»

При запуске приложения открывается главное окно со страницей авторизации показанная на рисунке 2.10. В верхней части расположен логотип, ниже имеются поля ввода логина и пароля, а также кнопка для входа в систему.

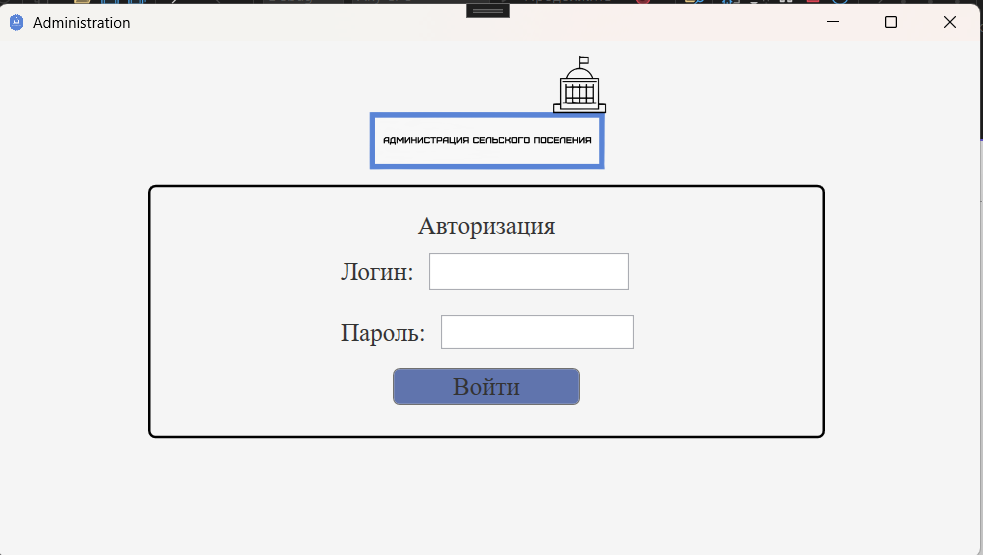


Рисунок 2.10 - Стартовое окно и страница «Авторизации»

При входе в систему, пользователя встречает главная страница изображенное на рисунке 2.11, на этой странице реализованы кнопки для перехода к списку документов и архиву, а также кнопка назад.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.11 - Страница «Главная страница»

После нажатия на кнопку «Список документов» открывается список документов, изображенный на рисунке 2.12, с нее можно перейти к добавлению или редактированию документа, а также удалить существующий из списка.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.12 - Страница «Список документов»

На странице «Список документов» изображенная на рисунке 2.13 при нажатии на кнопку добавить открывается страница для добавления документа.

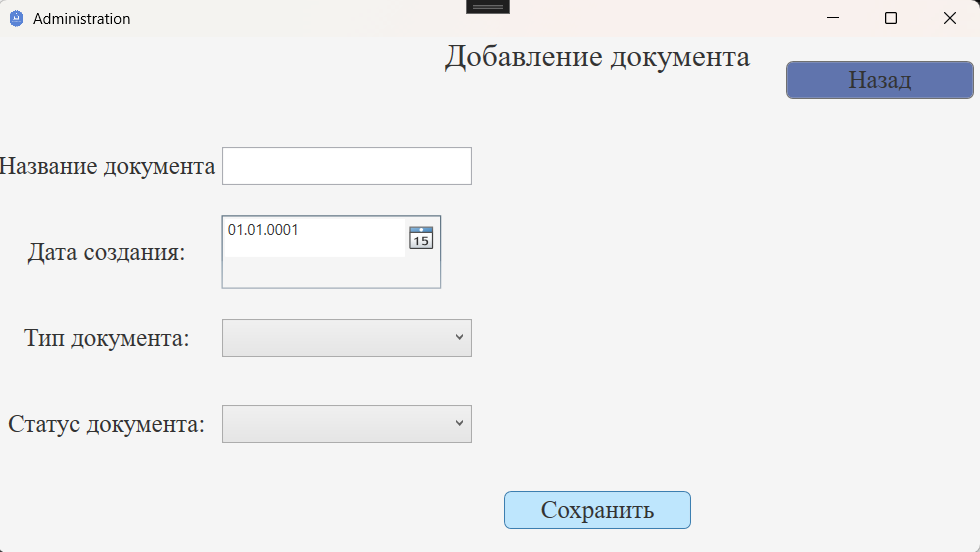


Рисунок 2.13 - Страница «Добавление документа»

На странице «Список документов» изображенная на рисунке 2.14 при нажатии на кнопку редактировать открывается страница для редактирования документа.

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.14 - Страница «Редактирование документа»

При нажатии кнопки архив на главной странице открывается список документов, изображенный на рисунке 2.15 который будет храниться в архиве.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.15 - Страница «Архив»

При нажатии на кнопку просмотреть открывается страница для просмотра документа изображенная на рисунке 2.16.

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2.16 - Страница «Просмотр документа»

# 2.7 Написание программного кода

Мною был разработан код для работы клиентской части программы.

Соединение с базой данных происходит с помощью метода GetContext(), представленного в листинге 1.

Листинг 1 – «DataBase.Context.cs»

private static DataBaseEntities \_context; public static DataBaseEntities GetContext() { if (\_context == null) \_context = new DataBaseEntities(); return \_context; }

Метод, представленный в листинге 1.2 обеспечивает базовую аутентификацию пользователя и обработку возможных ошибок ввода данны.

Листинг 1.2 AuthPage.xaml.cs

|  |
| --- |
| Try{if(string.IsNullOrWhiteSpace(txtBoxLogin.Text)||string.IsNullOrWhiteSpace(passBoxPassoPassword)){MessageBox.Show("Введите логин и пароль.", "Ошибка!", MessageBoxButton.OK,MessageBoxImage.Warning);return; } Users userObj = DataBaseEntities.GetContext().Users .FirstOrDefault(x => x.UserName == txtBoxLogin.Text && x.UserPassword == passBoxPassword.Password); if (userObj != null){ AppFrame.AuthUser = userObj; AppFrame.MainFrame.Navigate(new MainPage());} else{ MessageBox.Show("Неверное имя логина или пароль.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning);}} catch (Exception ex){ MessageBox.Show($"{ex.Message}", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning);}} |

Также будет проверятся корректность при сохранении нового документа, представленного в листинге 1.3.

Листинг 1.3 AddDocumentPage.xaml.cs

|  |
| --- |
| StringBuilder errors = new StringBuilder();if (string.IsNullOrWhiteSpace(\_currentDocument.Title)) errors.AppendLine("Укажите название документа"); if (\_currentDocument.TypeDocument == null) errors.AppendLine("Выберите тип документа"); if (errors.Length > 0){ MessageBox.Show(errors.ToString());return;} \_currentDocument.UserID = AppFrame.AuthUser.UserID; if (\_currentDocument.DocumentID == 0) DataBaseEntities.GetContext().Document.Add(\_currentDocument); try{ DataBaseEntities.GetContext().SaveChanges();MessageBox.Show("Информация сохранена"); AppFrame.MainFrame.GoBack();}catch (Exception ex){ MessageBox.Show(ex.Message.ToString());}} |

Метод, представленный на листинге 1.4 обеспечивает возможность удаления выбранных элементов из «DtGridDocument», с запросом подтверждения и обработкой возможных ошибок в процессе.

Листинг 1.4 ListDocumentPage.xaml.cs

|  |
| --- |
| vardocumentsForRemovingDtGridDocument.SelectedItems.Cast<Document>().ToList();if (documentsForRemoving.Count == 0) { MessageBox.Show("Пожалуйста, выберите документы для удаления.","Ошибка",MessageBoxButton.OK, MessageBoxImage.Warning); return;}if(MessageBox.Show($"Выточнохотитеудалитьследующие{documentsForRemoving.Count}документов?","Внимание",MessageBoxButton.YesNo,MessageBoxImage.Question) == MessageBoxResult.Yes) { try{ var context = DataBaseEntities.GetContext();foreach (vardocumentindocumentsForRemoving){ context.Document.Remove(document);}context.SaveChanges(); MessageBox.Show("Данные удалены"); DtGridDocument.ItemsSource = context.Document.ToList();}catch (Exception ex) {MessageBox.Show($"Ошибка при удалении: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } |

# Заключение

В ходе выполнения курсового проекта был разработан модуль информационной системы для администрации сельского поселения, который автоматизирует процесс учёта документов и введения журнала (архив).

Также закреплены теоретические знания и практические навыки работы с SQL Server, C# и XAML, умение работать с литературой, анализировать источники, делать обоснованные выводы.

Полученные теоретические и практические навыки будут использованы в дальнейшем при написании дипломной работы в будущем. Провела предварительное планирование и проверку осуществимости.

В рамках курсового проекта были выполнены следующие цели:

• Автоматизировала введение журнала документов (архива);

• Автоматизировала учёт документов;

• Повысила доступность предоставленной информации.

А также поставленные задачи:

1. Определила требования к программному продукту.
2. Разработала прототип приложения.
3. Реализовала приложение в выбранной среде IDE.
4. Создала сборку для установки.

Разместила проект на веб-сервисе GitHub

# Список использованных источников

**Интернет-источники:**

1. Сысолетин, Е. Г.  Разработка интернет-приложений: учебник для вузов / Е. Г. Сысолетин, С. Д. Ростунцев ; под научной редакцией Л. Г. Доросинского. — Москва: Издательство Юрайт, 2025. — 80 с. — (Высшее образование). — ISBN 978-5-534-17124-2. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: <https://urait.ru/bcode/562916> (дата обращения: 28.02.2025).В. В. Соколова Разработка мобильных приложений учебник для СПО Москва.Юрайт.2025 [сайт]. — URL: <https://urait.ru/viewer/razrabotka-mobilnyh-prilozheniy-566082#page/1> Режим доступа: для авторизированных пользователей.
2. Соколова, В. В. Вычислительная техника и информационные технологии. Разработка мобильных приложений: учебник для вузов / В. В. Соколова. — Москва: Издательство Юрайт, 2025. — 160 с. — (Высшее образование). — ISBN 978-5-534-16302-5. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/561336 (дата обращения: 28.02.2025).
3. Тузовский, А. Ф. Проектирование и разработка web-приложений: учебник для вузов / А. Ф. Тузовский. — Москва: Издательство Юрайт, 2025. — 219 с. — (Высшее образование). — ISBN 978-5-534-16300-1. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/561176 (дата обращения: 28.02.2025).
4. Тузовский, А. Ф. Проектирование и разработка web-приложений: учебник для среднего профессионального образования / А. Ф. Тузовский. — Москва: Издательство Юрайт, 2025. — 219 с. — (Профессиональное образование). — ISBN 978-5-534-16767-2. — Текст: электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/565693 (дата обращения: 28.02.2025).

# Приложение

**Листинги программного кода**

App.xaml

|  |
| --- |
| **<**Application.Resources> <Style TargetType="TextBlock"> <Setter Property="FontSize" Value="20"/> <Setter Property="FontFamily" Value="Times New Roman"/> <Setter Property="Foreground" Value="#333333"/> <Setter Property="TextAlignment" Value="Center"/> </Style> <Style TargetType="Image"> <Setter Property="Height" Value="120"/> <Setter Property="Width" Value="220"/> </Style> <Style TargetType="Button"> <Setter Property="Height" Value="30"/> <Setter Property="Width" Value="150"/> <Setter Property="Margin" Value="5"/> <Setter Property="Background" Value="#6074ad"/> </Style> <Style TargetType="TextBox"> <Setter Property="Height" Value="30"/> <Setter Property="Width" Value="200"/> <Setter Property="Margin" Value="5"/> </Style> <Style TargetType="ComboBox"> <Setter Property="HorizontalAlignment" Value="Left"/> <Setter Property="Height" Value="30"/> <Setter Property="Width" Value="200"/> <Setter Property="Margin" Value="5"/> </Style> </Application.Resources> |

AuthPage.xaml

|  |
| --- |
| <Grid Background="#F5F5F5"> <Image Source="/Resources/Image2.png" HorizontalAlignment="Center" VerticalAlignment="Top"/> <Border BorderBrush="Black" BorderThickness="2" CornerRadius="5" Margin="120,115,124,95"> <StackPanel Orientation="Vertical" Grid.Row="1" VerticalAlignment="Center" HorizontalAlignment="Center" Margin="20"> <TextBlock Text="Авторизация" TextAlignment="Center"/> <StackPanel Orientation="Horizontal" Margin="5"> <TextBlock Text="Логин:" Margin="7" TextAlignment="Right" VerticalAlignment="Center"/> <TextBox x:Name="txtBoxLogin" Margin="5" Width="160"/> </StackPanel> <StackPanel Orientation="Horizontal" Margin="5"> <TextBlock Text="Пароль:" Margin="7" TextAlignment="Right" VerticalAlignment="Center"/> <PasswordBox x:Name="passBoxPassword" Margin="5" Width="155"/> </StackPanel> <StackPanel Orientation="Vertical"> <Button x:Name="Loginbutton" Content="Войти" Click="Loginbutton\_Click" Background="#6074ad"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </StackPanel> </StackPanel> </Border> </Grid> |

AuthPage.xaml.cs

|  |
| --- |
| public partial class AuthPage : Page { public AuthPage() { InitializeComponent(); } private void Loginbutton\_Click(object sender, RoutedEventArgs e) { try { if (string.IsNullOrWhiteSpace(txtBoxLogin.Text) || string.IsNullOrWhiteSpace(passBoxPassword.Password)) { MessageBox.Show("Введите логин и пароль.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning); return; } Users userObj = DataBaseEntities.GetContext().Users .FirstOrDefault(x => x.UserName == txtBoxLogin.Text && x.UserPassword == passBoxPassword.Password); if (userObj != null) { AppFrame.AuthUser = userObj; AppFrame.MainFrame.Navigate(new MainPage()); } else { MessageBox.Show("Неверное имя логина или пароль.", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning); } } catch (Exception ex) { MessageBox.Show($"{ex.Message}", "Ошибка!", MessageBoxButton.OK, MessageBoxImage.Warning); } } } |

MainPage.xaml

|  |
| --- |
| <Grid Background="#F5F5F5"> <Grid.RowDefinitions> <RowDefinition Height="70"/> <RowDefinition Height="\*"/> </Grid.RowDefinitions> <TextBlock Text="Главная страница" TextAlignment="Center" FontSize="30"/> <Button x:Name="btnBack" Content="Назад" Click="btnBack\_Click" Background="#6074ad" Height="30" Width="100" HorizontalAlignment="Right"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> <StackPanel Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Center" Grid.Row="1"> <Button x:Name="Documentbtn" Content="Список документов" Click="Documentbtn\_Click" Background="#6074ad" Height="80" Width="200"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> <Button x:Name="Archivebtn" Content="Архив" Click="Archivebtn\_Click" Background="#6074ad" Height="80" Width="200"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </StackPanel> </Grid> |

MainPage.xaml.cs

|  |
| --- |
| public partial class MainPage : Page { public MainPage() { InitializeComponent(); CheckerRole(); } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.GoBack(); } private void Documentbtn\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.Navigate(new ListDocumentPage()); } private void Archivebtn\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.Navigate(new ArchivePage()); } private void CheckerRole() { switch (AppFrame.AuthUser .Role) { case 1: case 2: AppFrame.MainFrame.Navigate(new ListDocument\_GPage()); break; case 3: case 4: AppFrame.MainFrame.Navigate(new ListDocumentPage()); break; } } } |

ListDocument.xaml

|  |
| --- |
| <Grid Background ="#F5F5F5"> <Grid.RowDefinitions> <RowDefinition Height="70"/> <RowDefinition Height="\*"/> </Grid.RowDefinitions> <TextBlock Text="Список документов" TextAlignment="Center" FontSize="32"/> <Button x:Name="btnBack" Content="Назад" Click="btnBack\_Click" Background="#6074ad" HorizontalAlignment="Right" Margin="5"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> <DataGrid x:Name="DtGridDocument" AutoGenerateColumns="False" IsReadOnly="True" Grid.Row="1"> <DataGrid.Columns> <DataGridTextColumn Header="Название" Binding="{Binding Title}" Width="13\*"/> <DataGridTextColumn Header="Дата создания" Binding="{Binding CreatedDate, StringFormat='{}{0:dd.MM.yyyy}'}" Width="13\*"/> <DataGridTextColumn Header="Статус документа" Binding="{Binding StatusDocument.StatusName}" Width="17\*"/> <DataGridTextColumn Header="Тип документа" Binding="{Binding TypeDocument.TypeName}" Width="17\*"/> <DataGridTemplateColumn Width="auto"> <DataGridTemplateColumn.CellTemplate> <DataTemplate> <Button Content="Редактировать" Name="Editbtn" Click="Editbtn\_Click" Background="#6074ad"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </DataTemplate> </DataGridTemplateColumn.CellTemplate> </DataGridTemplateColumn> </DataGrid.Columns> </DataGrid> <StackPanel Grid.Row="2" Orientation="Horizontal" HorizontalAlignment="Center" VerticalAlignment="Bottom"> <Button Content="Удалить" x:Name="Deletebtn" Click="Deletebtn\_Click" Margin="20" Background="#F44336"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> <Button Content="Добавить" x:Name="Addbtn" Click="Addbtn\_Click" Background="#4CAF50"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </StackPanel> </Grid> |

ListDocument.xaml.cs

|  |
| --- |
| public partial class ListDocumentPage : Page { public ListDocumentPage() { InitializeComponent(); DtGridDocument.ItemsSource = DataBaseEntities.GetContext().Document.ToList(); } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.GoBack(); } private void Editbtn\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.Navigate(new EditDocumentPage()); } private void Addbtn\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.Navigate(new AddDocumentPage()); } private void Deletebtn\_Click(object sender, RoutedEventArgs e) { var documentsForRemoving = DtGridDocument.SelectedItems.Cast<Document>().ToList(); if (documentsForRemoving.Count == 0) { MessageBox.Show("Пожалуйста, выберите документы для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning); return; } if (MessageBox.Show($"Вы точно хотите удалить следующие {documentsForRemoving.Count} документов?", "Внимание", MessageBoxButton.YesNo, MessageBoxImage.Question) == MessageBoxResult.Yes) { try { var context = DataBaseEntities.GetContext(); foreach (var document in documentsForRemoving) { context.Document.Remove(document); } context.SaveChanges(); MessageBox.Show("Данные удалены"); DtGridDocument.ItemsSource = context.Document.ToList(); } catch (Exception ex) { MessageBox.Show($"Ошибка при удалении: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); } } } private void Page\_IsVisibleChanged(object sender, DependencyPropertyChangedEventArgs e) { if (Visibility == Visibility.Visible) { DataBaseEntities.GetContext().ChangeTracker.Entries().ToList().ForEach(p => p.Reload()); DtGridDocument.ItemsSource = DataBaseEntities.GetContext().Document.ToList(); } } } |

AddDocument.xaml

|  |
| --- |
| <Grid Background="#F5F5F5"> <Grid.ColumnDefinitions> <ColumnDefinition Width="177\*"/> <ColumnDefinition Width="623\*"/> </Grid.ColumnDefinitions> <Grid.RowDefinitions> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> </Grid.RowDefinitions> <TextBlock Text="Добавление документа" Grid.Column="1" FontSize="25"/> <Button x:Name="btnBack" Content="Назад" Click="btnBack\_Click" Grid.Column="1" HorizontalAlignment="Right"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> <TextBlock Text="Название документа:" Grid.Row="1" VerticalAlignment="Center"/> <TextBox x:Name="txtTitle" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left" Text="{Binding Title}"/> <TextBlock Text="Дата создания:" Grid.Row="2" Grid.Column="0" VerticalAlignment="Center"/> <DatePicker x:Name="datePickerCreatedDate" Grid.Row="2" Grid.Column="1" Margin="5" HorizontalAlignment="Left" SelectedDate="{Binding CreatedDate}"/> <TextBlock Text="Тип документа:" Grid.Row="3" Grid.Column="0" VerticalAlignment="Center"/> <ComboBox Name="cmbType" Grid.Row="3" Grid.Column="1" HorizontalAlignment="Left" SelectedItem="{Binding TypeDocument}" DisplayMemberPath="TypeName"/> <TextBlock Text="Статус документа:" Grid.Row="4" Grid.Column="0" VerticalAlignment="Center"/> <ComboBox Name="cmbStatus" Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left" SelectedItem="{Binding StatusDocument}" DisplayMemberPath="StatusName"/> <Button x:Name="btnSave" Content="Сохранить" Click="btnSave\_Click" Grid.Row="5" Grid.Column="1"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </Grid> |

AddDocument.xaml.cs

|  |
| --- |
| public partial class AddDocumentPage : Page { private Document \_currentDocument = new Document(); public AddDocumentPage() { InitializeComponent(); DataContext = \_currentDocument; cmbType.ItemsSource = DataBaseEntities.GetContext().TypeDocument.ToList(); cmbStatus.ItemsSource = DataBaseEntities.GetContext().StatusDocument.ToList(); } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.GoBack(); } private void btnSave\_Click(object sender, RoutedEventArgs e) { StringBuilder errors = new StringBuilder(); if (string.IsNullOrWhiteSpace(\_currentDocument.Title)) errors.AppendLine("Укажите название документа"); if (\_currentDocument.TypeDocument == null) errors.AppendLine("Выберите тип документа"); if (errors.Length > 0) { MessageBox.Show(errors.ToString()); return; } \_currentDocument.UserID = AppFrame.AuthUser .UserID; if (\_currentDocument.DocumentID == 0) DataBaseEntities.GetContext().Document.Add(\_currentDocument); try { DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Информация сохранена"); AppFrame.MainFrame.GoBack(); } catch (Exception ex) { MessageBox.Show(ex.Message.ToString()); } } } |

EditDocument.xaml

|  |
| --- |
| <Grid Background="#F5F5F5"> <Grid.ColumnDefinitions> <ColumnDefinition Width="177\*"/> <ColumnDefinition Width="623\*"/> </Grid.ColumnDefinitions> <Grid.RowDefinitions> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> </Grid.RowDefinitions> <TextBlock Text="Редактирование документа" Grid.Column="1" FontSize="25"/> <Button x:Name="btnBack" Content="Назад" Click="btnBack\_Click" Grid.Column="1" HorizontalAlignment="Right"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> <TextBlock Text="Название документа:" Grid.Row="1" VerticalAlignment="Center"/> <TextBox x:Name="txtTitle" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left" Text="{Binding Title}"/> <TextBlock Text="Дата создания:" Grid.Row="2" Grid.Column="0" VerticalAlignment="Center"/> <DatePicker x:Name="datePickerCreatedDate" Grid.Row="2" Grid.Column="1" Margin="5" HorizontalAlignment="Left" SelectedDate="{Binding CreatedDate}"/> <TextBlock Text="Тип документа:" Grid.Row="3" Grid.Column="0" VerticalAlignment="Center"/> <ComboBox Name="cmbType" Grid.Row="3" Grid.Column="1" HorizontalAlignment="Left" SelectedItem="{Binding TypeDocument}" DisplayMemberPath="TypeName"/> <TextBlock Text="Статус документа:" Grid.Row="4" Grid.Column="0" VerticalAlignment="Center"/> <ComboBox Name="cmbStatus" Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left" SelectedItem="{Binding StatusDocument}" DisplayMemberPath="StatusName"/> <Button x:Name="btnSave" Content="Сохранить" Click="btnSave\_Click" Grid.Row="5" Grid.Column="1"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </Grid> |

EditDocument.xaml.cs

|  |
| --- |
| public partial class EditDocumentPage : Page { private Document \_currentDocument = new Document(); public EditDocumentPage() { InitializeComponent(); DataContext = \_currentDocument; cmbType.ItemsSource = DataBaseEntities.GetContext().TypeDocument.ToList(); cmbStatus.ItemsSource = DataBaseEntities.GetContext().StatusDocument.ToList(); } private void btnBack\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.GoBack(); } private void btnSave\_Click(object sender, RoutedEventArgs e) { StringBuilder errors = new StringBuilder(); if (string.IsNullOrWhiteSpace(\_currentDocument.Title)) errors.AppendLine("Укажите название документа"); if (\_currentDocument.TypeDocument == null) errors.AppendLine("Выберите тип документа"); if (errors.Length > 0) { MessageBox.Show(errors.ToString()); return; } \_currentDocument.UserID = AppFrame.AuthUser .UserID; if (\_currentDocument.DocumentID == 0) DataBaseEntities.GetContext().Document.Add(\_currentDocument); try { DataBaseEntities.GetContext().SaveChanges(); MessageBox.Show("Информация сохранена"); AppFrame.MainFrame.GoBack(); } catch (Exception ex) { MessageBox.Show(ex.Message.ToString()); } } } |

ViewingDocumentPage.xaml

|  |
| --- |
| <Grid Background="#F5F5F5"> <Grid.ColumnDefinitions> <ColumnDefinition Width="177\*"/> <ColumnDefinition Width="623\*"/> </Grid.ColumnDefinitions> <Grid.RowDefinitions> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> <RowDefinition/> </Grid.RowDefinitions> <TextBlock Text="Просмотр документа" FontSize="32" TextAlignment="Center" Grid.Column="1"/> <TextBlock Text="Название:" FontWeight="Bold" Grid.Row="1" VerticalAlignment="Center"/> <TextBox x:Name="txtDocument" Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left"/> <TextBlock Text="Дата внесения:" FontWeight="Bold" Grid.Row="2" VerticalAlignment="Center"/> <TextBox x:Name="txtDateDeposit" Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left"/> <TextBlock Text="Срок хранения (лет):" FontWeight="Bold" Grid.Row="3" VerticalAlignment="Center"/> <TextBox x:Name="txtShelfLife" Grid.Row="3" Grid.Column="1" HorizontalAlignment="Left"/> <Button Content="Назад" Click="Button\_Click" Background="#6074ad" HorizontalAlignment="Right" Grid.Column="1"> <Button.Resources> <Style TargetType="Border"> <Setter Property="CornerRadius" Value="5"/> </Style> </Button.Resources> </Button> </Grid> |

ViewingDocumentPage.xaml.cs

|  |
| --- |
| public partial class ViewingDocumentPage : Page { private Archive \_document; public ViewingDocumentPage(Archive document) { InitializeComponent(); \_document = document; txtDocument.Text = \_document.Document.Title; txtDateDeposit.Text = \_document.DateDeposit.ToString("dd.MM.yyyy"); txtShelfLife.Text = \_document.ShelfLife.ToString(); } private void Button\_Click(object sender, RoutedEventArgs e) { AppFrame.MainFrame.GoBack(); } } |