МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий
Кафедра программной инженерии
Специальность 1-40 05 01 Информационные системы и технологии
Направление специальности 1-40 05 01 03 «Информационные системы и технологии (издательско-полиграфический комплекс)»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
КУРСОВОГО ПРОЕКТА:

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»
Тема «Программное средство для цифровой подписи PDF-документов на основе QR-кода»

Исполнитель
студентка 3 курса группы 1 Самсоник Анастасия Ивановна
(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой
Председатель Смелов В.В.
(подпись)

Минск 2023

Содержание

Введение

В современном информационном обществе, где цифровая документация стала неотъемлемой частью нашей повседневной жизни, вопросы обеспечения безопасности, целостности и удобства обмена данными имеют особенно важное значение. В этом контексте технологии создания QR-кодов и электронной цифровой подписи PDF-документов становятся ключевыми инструментами для обеспечения эффективности и надежности документооборота.

Курсовой проект посвящен разработке программного средства, которое объединяет в себе два важных аспекта информационной безопасности и эффективного обмена информацией: генерацию QR-кодов и цифровую подпись PDF-документов на основе QR-кодов.

Актуальность этой темы обусловлена современными требованиями к обеспечению безопасности и целостности данных в информационной среде. Быстрый и удобный доступ к информации, а также гарантированная аутентичность документов имеют важное практическое значение для различных сфер деятельности.

Целью данного курсового проекта является создание программного продукта, способного облегчить процессы обмена информацией и укрепить доверие между участниками этого процесса.

Проектирование программы основано на принципах эффективности, безопасности и простоты использования.

Первый раздел посвящен анализу текущих тенденций в области информационной безопасности и обмена данными с использованием QR-кодов. Здесь мы также рассмотрим алгоритмы решения, проведем обзор прототипов и обоснуем актуальность задачи.

Второй раздел будет посвящен описанию архитектуры и структуры модулей и классов, лежащих в основе разрабатываемого программного средства.

В третьем разделе будут подробно описаны функциональные возможности программы, определены выполняемые функции и рассмотрена модель данных, используемую в проекте.

Четвертый раздел будет посвящен реализации и тестированию программного продукта, его функциональности и безопасности.

Каждый из этих разделов будет нацелен на решение конкретных задач, направленных на достижение поставленной цели проекта.

1. Постановка задачи и анализ аналогов и прототипов

1.1 Анализ аналогов и прототипов

При разработке программного средства для генерации QR-кодов и электронной цифровой подписи PDF-документов, важно рассмотреть существующие аналоги и альтернативные решения, которые также предназначены для решения подобных задач. Этот обзор позволяет выявить преимущества и недостатки конкурирующих продуктов и определить, каким образом наше программное средство может выделиться на фоне аналогов.

Аналог 1: Программное средство генерации QR-кодов "QR Code Generator Pro".

Программное средство "QR Code Generator Pro" предоставляет пользователю возможность быстро и легко создавать QR-коды для различных целей. Интерфейс программного средства представлен на рисунке 1.1.

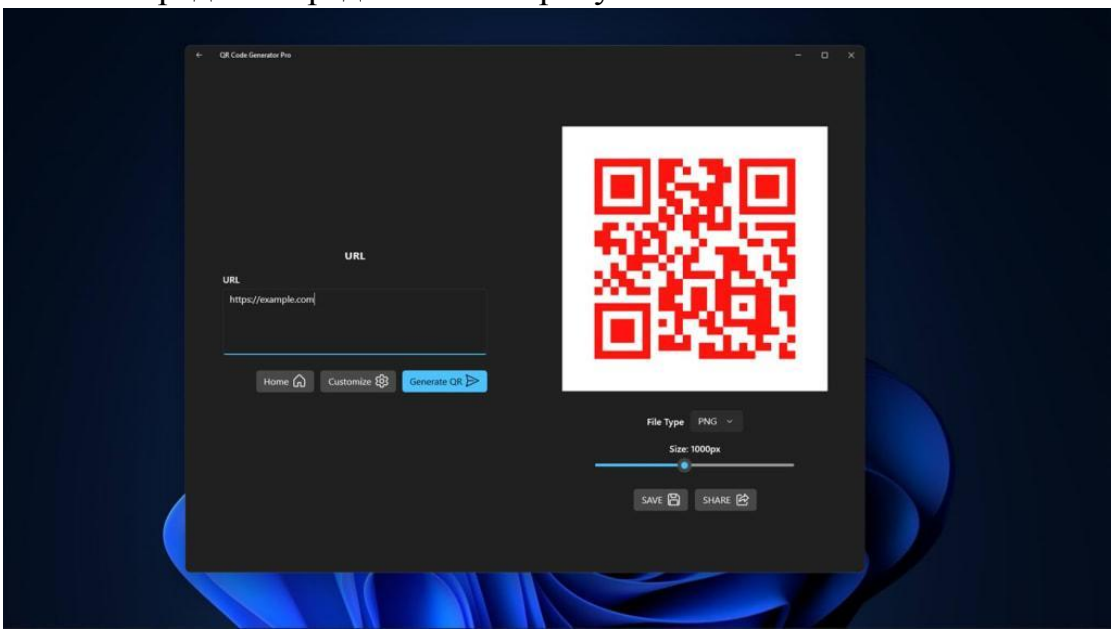


Рисунок 1.1 – ПС «QR Code Generator Pro»

Основные преимущества этой программы: простой и интуитивно понятный интерфейс и широкий спектр настроек. Однако данное программное средство ограничивается только генерацией QR-кодов и не предоставляет функциональности для работы с электронными цифровыми подписями.

Аналог 2: Программное средство для цифровой подписи "PDF Sign&Seal".

Программное средство "PDF Sign & Seal" предназначено для создания электронных цифровых подписей на PDF-документах. Интерфейс программного средства представлен на рисунке 1.2.

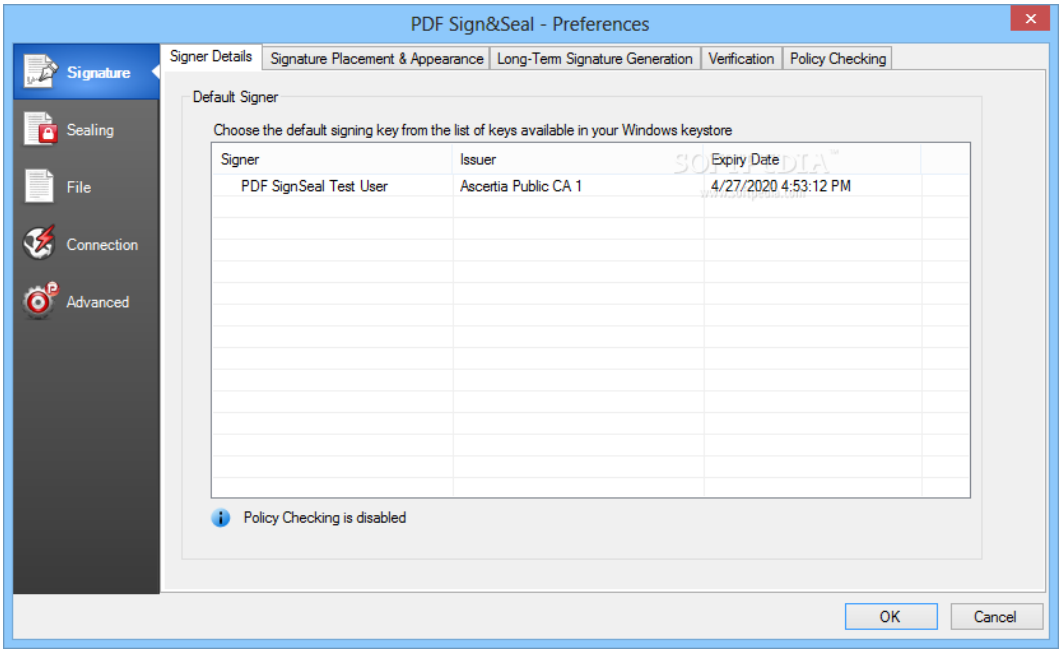


Рисунок 1.2 – ПС «PDF Sign&Seal»

Данное программное средство обладает широкими возможностями по управлению подписями и шифрованием документов, что делает его востребованным инструментом в сфере безопасности PDF-файлов. Тем не менее, данное программное средство ограничивается только работой с PDF и не включает функциональность генерации QR-кодов.

1.2 Постановка задачи

На основе анализа аналогичных решений и с учетом темы проекта был составлен список функций, которые должна выполнять разрабатываемая программа:

- генерация QR-кода на основе введенного пользователем текста;
- загрузка изображения с QR-кодом с устройства для его чтения;
- сохранение полученного изображения на устройство;
- выбор основного цвета получаемого изображения с QR-кодом (по умолчанию цвет чёрный);
- выбор иконки которая будет расположена в центре получаемого изображения с QR-кодом (по умолчанию иконка отсутствует);
- загрузка pdf-документа для цифровой подписи;
- настройка параметров цифровой подписи PDF-документа;
- сохранение полученного подписанного PDF-документа.

1.3 Вывод

После оценки всех приложений можно заметить, что многие программы имеют схожую функциональность, но аналоги проекта практически отсутствуют, что определенно является плюсом. Проанализировав аналоги приложений с тематикой генерации QR-кодов и цифровой подписи PDF-документов были выделены основные задачи проекта: разработка приложения, позволяющего пользователю осуществлять весь базовый функционал приложения для генерации QR-кодов и приложения для цифровой подписи PDF-документов. Должна быть возможность указывать дополнительные параметры получаемого изображения, дополнительные параметры цифровой подписи документа, возможность сохранять изображения QR-кодов и подписанные PDF-документы. Приложение также будет показывать историю сгенерированных QR-кодов и подписанных документов за определенный промежуток времени с возможностью удалять отдельные элементы из истории (БД). Интерфейс должен быть удобным, понятным и легко настраиваемым для пользователя.

2. Проектирование архитектуры проекта

2.1 Проектирование схемы базы данных

Для эффективного хранения и управления данными приложения была разработана логическая схема базы данных. Она определяет структуру данных, их связи и основные таблицы. Логическая схема базы данных представлена на рисунке 2.1.

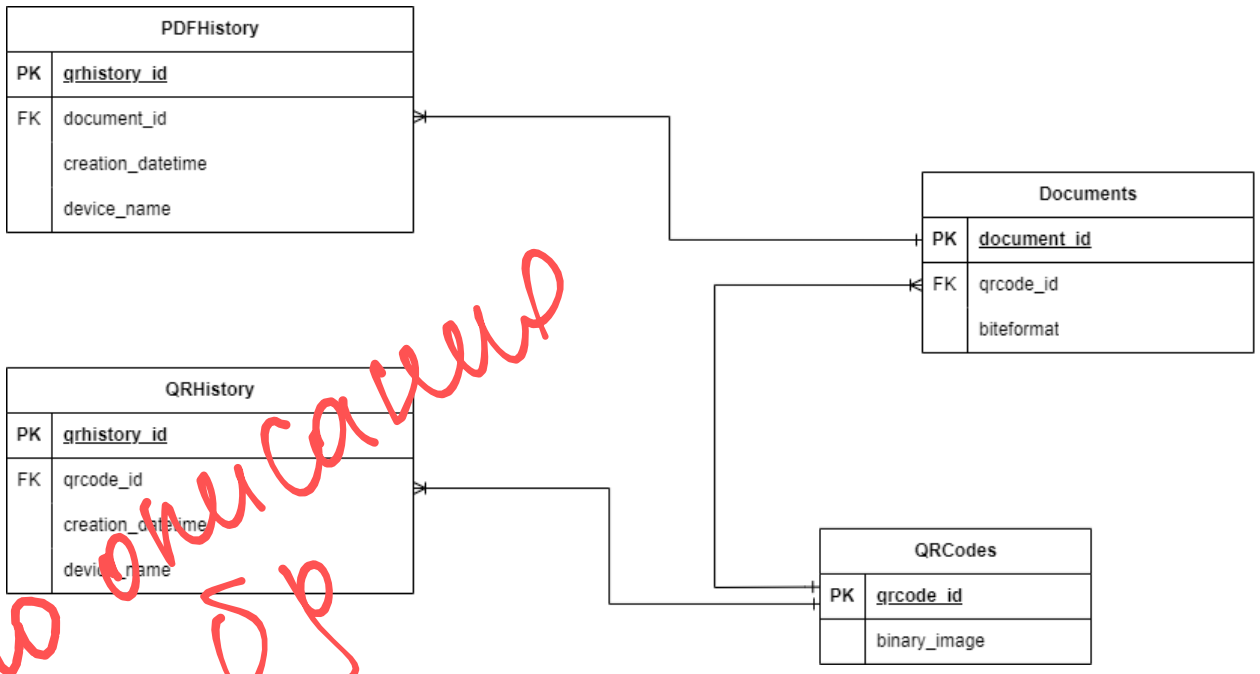


Рисунок 2.1 – Логическая схема БД

Таблица «QRCodes» хранит id QR-кода и его изображение в бинарном виде.

Таблица «QRHistory» хранит id истории создания и информацию о создании QR-кода, такую как дата и время создания и имя устройство, с которого QR-код был создан, ссылается внешним ключом на таблицу «QRCodes». Такое решение было принято с целью оптимизации базы данных (большие данные хранятся в отдельной таблице).

Похожим образом реализована связь таблиц «Documents», которая хранит id подписанного PDF-документа и сам документ в байтовом формате, и «PDFHistory», которая хранит информацию о подписанном документе.

2.2 Описание средств разработки

Для разработки программного средства используются:
– среда разработки Microsoft Visual Studio 2022;
– программная платформа .NET;
– язык программирования C#;
– технология WPF;

– Microsoft SQL Server MS 18;

Для разработки данного приложения была выбрана технология Windows Presentation Foundation (WPF) в качестве интерфейса. WPF представляет собой современный инструмент для создания клиентских приложений, предназначенных для операционной системы Windows. Он обладает передовыми возможностями для взаимодействия с пользователем и интегрирован в платформу .NET.

Основой технологии WPF служит инфраструктура, построенная на основе DirectX, что придает ей значительное преимущество в мощности и гибкости по сравнению с более ранними технологиями, такими как Windows Forms.

2.3 Проектирование окон приложения

В программном средстве при запуске будет отображаться окно для генерации QR-кодов, в котором будет находится обозначенное место под изображение (в котором будет появляться изображение с созданным или загруженным пользователем QR-кодом), рядом с ним будет располагаться поле для ввода текста либо ссылки, под полем будет выпадающий список для выбора иконок, в котором будет отображаться картинка и название иконки (Instagram, Facebook, Telegram и др.), рядом будет раскрывающаяся палитра цветов для выбора основного цвета QR-кода. Также под полем для ввода будет находится кнопка «CREATE» которая преобразует введенные данные в изображение с QR-кодом с заданными параметрами. Под изображением находятся кнопки, позволяющие скачать изображение на устройство, загрузить изображение с устройства и прочитать данные с изображения QR-кода.

В любой момент можно переключиться между окном для создания QR-кода и окном для цифровой подписи PDF-документов. В окне для цифровой подписи PDF-документов будет располагаться кнопка для выбора PDF-документа для подписи, поля и выпадающие списки для выбора параметров цифровой подписи, на каждый вводимый параметр можно будет посмотреть расшифровку по наведению (за что этот параметр отвечает, обязательный ли), будут кнопки для генерации и сохранения документа на устройство.

Также будет окно, в котором пользователь сможет просматривать и при надобности удалять историю своих генераций и подписей (отдельная вкладка для QR-кодов и для PDF-документов). Там будут храниться изображения/документы с датой создания.

2.4 Проектирование User Case диаграммы

На основе разработанной логической схемы БД, описанного основного функционала программы и описания будущих окон приложения, были определены основные возможности пользователя. Для наглядного отображения возможностей пользователя была создана User Case диаграмма (Рисунок 2.2).

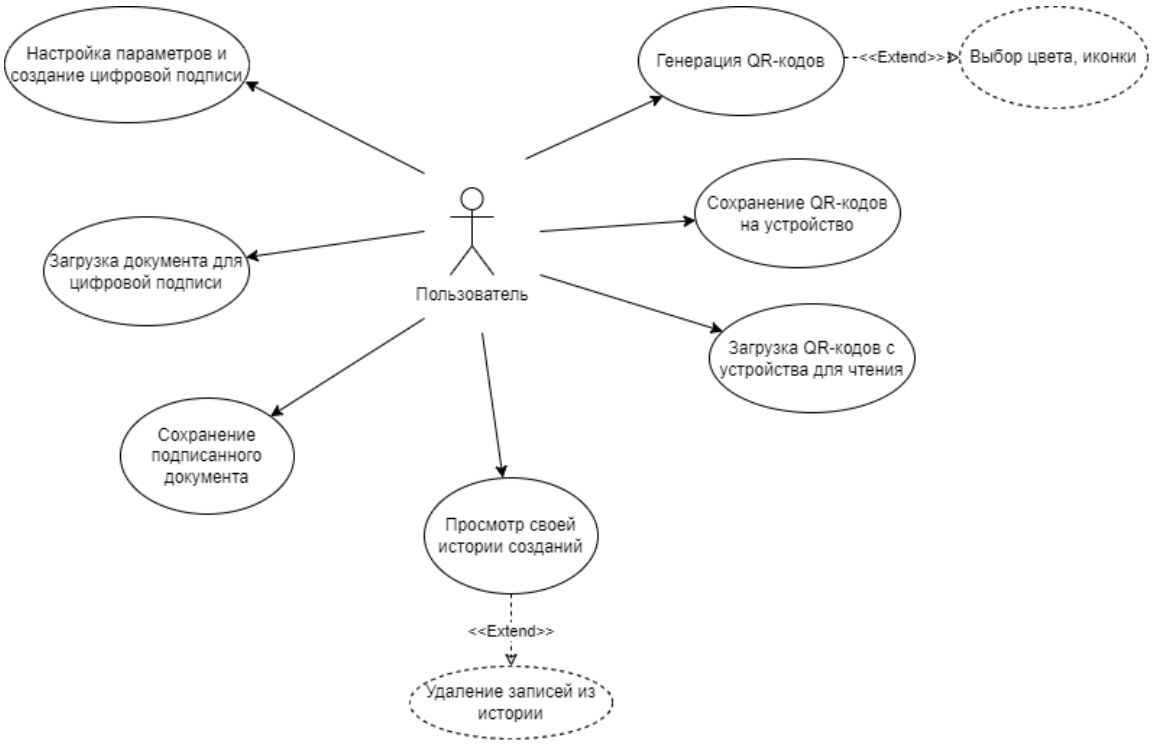


Рисунок 2.2 – User Case диаграмма

В качестве параметров подписи пользователь указывает в первую очередь значение, передаваемое в подпись. Также он может указать отступы сверху и слева документа, размер подписи, цвет (выбором из палитры), прозрачность, стиль границы. Введенные параметры будут проверяться валидацией на случай если пользователь случайно введет слишком большое или отрицательное значение, которое нарушит логику приложения, после чего параметры передаются через binding в программный код и на их основе генерируется подпись (по подтверждению этих параметров пользователей кнопкой “Commit and Create”). Для создания обычного QR-кода (не подписи), пользователь просто указывает то, что хочет закодировать и по желанию выбирает основной цвет и/или иконку. Если пользователь ничего не ввёл и нажал кнопку “Create QRCode” срабатывает исключение, которое оповещает пользователя, что значение не может быть пустым. Если исключения нет и все корректно, то пользователь получает изображение QR-кода со своими параметрами, которое может сохранить на устройство.

2.5 Вывод

В этом разделе было подробно описано проектирование архитектуры проекта, включая схему базы данных, выбор средств разработки, проектирование окон приложения и User Case диаграмму для наглядного отображения возможностей пользователя.

Схема базы данных была разработана для эффективного хранения и управления данными приложения. Она включает в себя таблицы для хранения QR-кодов, истории создания QR-кодов, подписанных PDF-документов и истории подписанных документов.

В качестве средств разработки были выбраны Microsoft Visual Studio 2022, .NET, C#, WPF и Microsoft SQL Server MS 18. Было принято решение использовать WPF для создания интерфейса пользователя, так как эта технология предлагает передовые возможности для взаимодействия с пользователем и интегрирована в платформу .NET.

Были описаны окна приложения, которые будут отображаться при запуске программного средства. Они включают окно для генерации QR-кодов, окно для цифровой подписи PDF-документов и окно для просмотра истории генераций и подписей.

В целом, этот раздел предоставляет подробное описание проектирования архитектуры проекта, что является важным шагом в разработке программного средства. Это помогает обеспечить, что все аспекты проекта были тщательно продуманы и спланированы заранее, что в конечном итоге приведет к созданию качественного и эффективного программного продукта.

3. Проектирование программного средства

3.1. Общая структура

Программное средство «Генератор QR-кодов» имеет следующую структуру, представленную на рисунке 3.1.

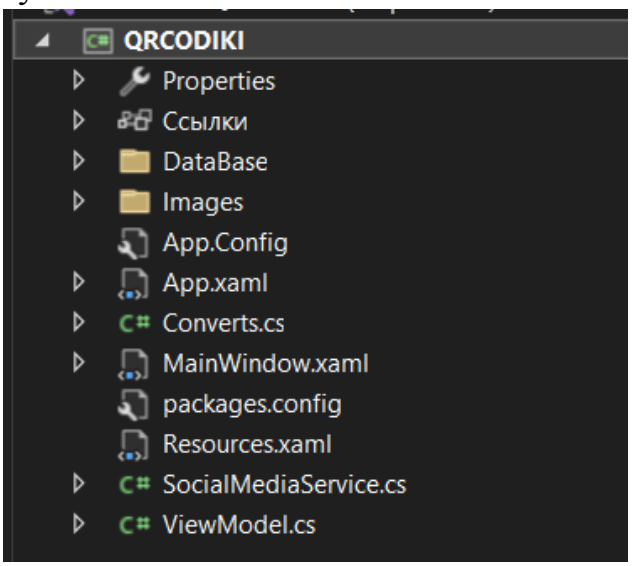


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

Имя файла	Содержание
Ссылки	Подключенные фреймворки и пакеты
Папка DataBase	Папка содержащая файлы для работа с базой данных
Папка Images	Папка содержит изображение которые применяются в программном средстве
App.config	Файл содержит настройки подключения и строку подключения к базе данных
App.xaml	Файл, содержащий словари с ресурсами для всех окон.
MainWindow.xaml	Файл, описывающий разметку окон программного средства, в зависимости имеет MainWindow.xaml.cs
Resources.xaml	Файл, содержащий словарь ресурсов
SocialMediaService.cs	Файл, содержащий классы для хранения и использования иконок для прикрепления к QR-коду
ViewModel.cs	Файл, содержащий модель представления с определением свойств и интерфейсом INotifyPropertyChange
Converts.cs	Содержит интерфейс инициализирующий функции конвертирования файлов или объектов в нужные форматы.

Более подробная структура папки DataBase представлена на Рисунке 3.2.

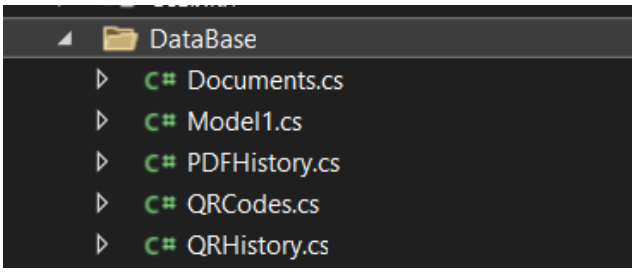


Рисунок 3.2 – Содержимое папки DataBase

В данной папке находятся классы, построенные с помощью EntityFramework подхода CodeFirst из базы данных. Эти классы определяют сущности основных объектов, которые будут взаимодействовать с базой данных через модель данных, которая определена классом Model.

3.2. Взаимоотношения между классами

Для визуализации взаимосвязей между классами используется диаграмма UML – графическое представление набора элементов, изображаемое чаще всего в виде связанного графа с вершинами (сущностями) и ребрами (отношениями).

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других.

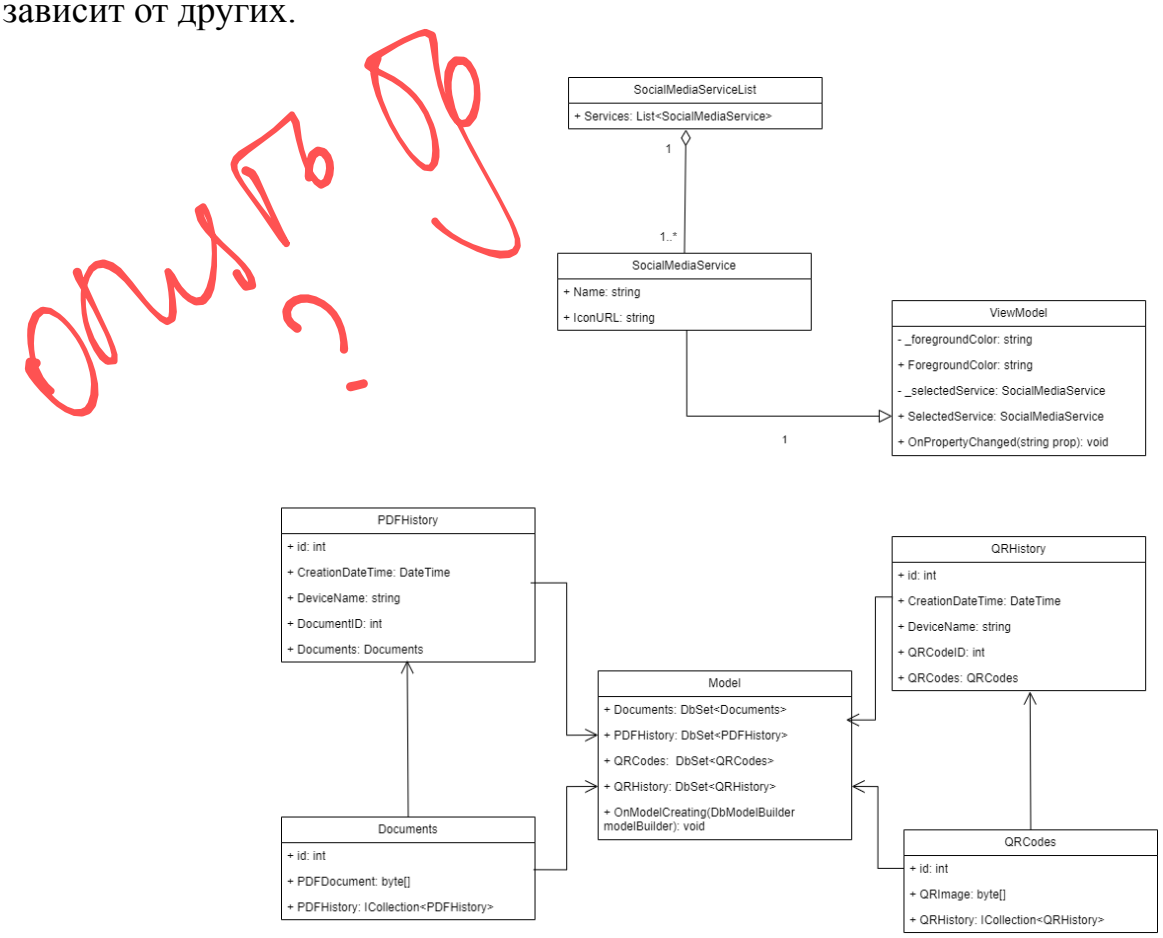


Рисунок 3.2 – Диаграмма классов программного средства

3.3. Модель базы данных

Для реализации поставленной задачи была создана база данных QRCodes. Для ее создания был использован Entity Framework (подход code first). База данных состоит из таблиц, представленных на Рисунке 3.3.

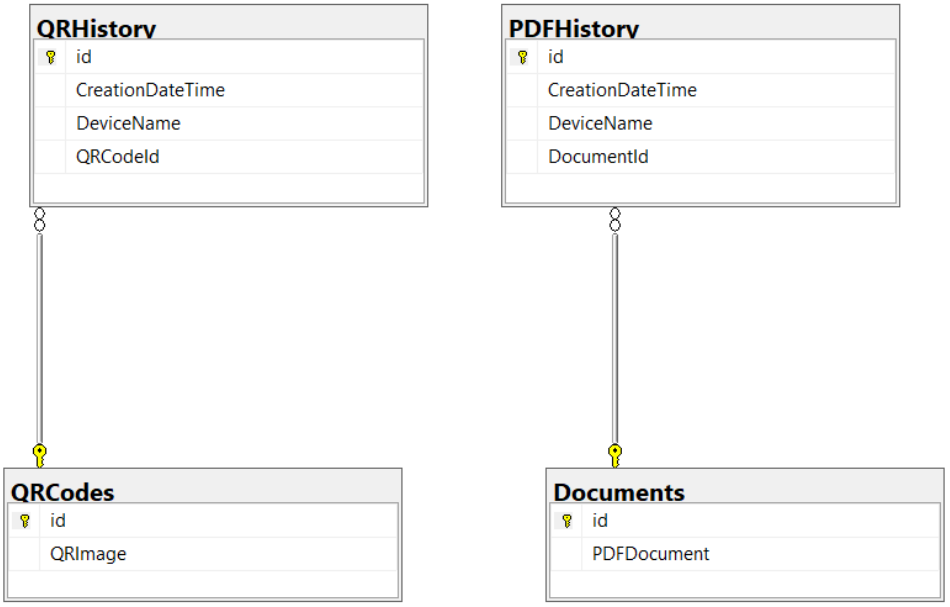


Рисунок 3.3 – Таблицы базы данных

В базе данных находится 4 таблицы. Таблица QRHistory содержит данные о создании qr-кода, такие как время создания, имя устройства с которого qr-код был создан и внешний ключ связанный с таблицей QRCodes ссылающийся на изображение QR-кода в бинарном формате. Проект для QRHistory представлен на Рисунке 3.4.

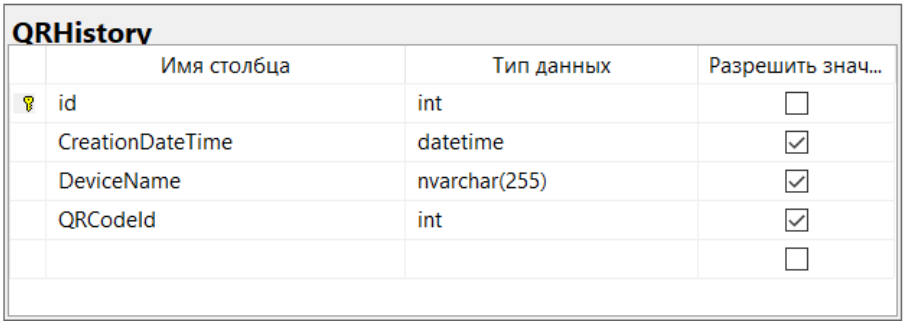
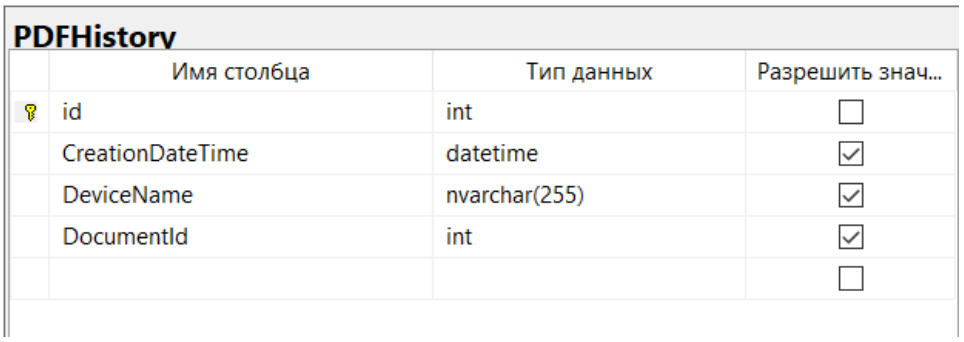


Рисунок 3.4 – Таблица QRHistory

Таблица PDFHistory содержит информацию о подписанных PDF-документах, а именно время создания, устройство с которого, был подписан документ и внешний ключ ссылающийся на. Проект для PDFHistory представлен на Рисунке 3.5.

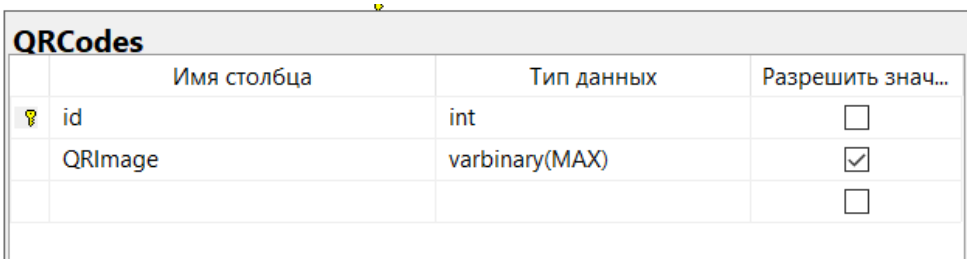


PDFHistory

	Имя столбца	Тип данных	Разрешить знач...
🔑	id	int	☐
	CreationDateTime	datetime	☑
	DeviceName	nvarchar(255)	☑
	DocumentId	int	☑
			☐

Рисунок 3.5 – Таблица PDFHistory

Таблица QRCodes хранит id qr-кода и его бинарное изображение. Проект для QRCodes представлен на Рисунке 3.6.

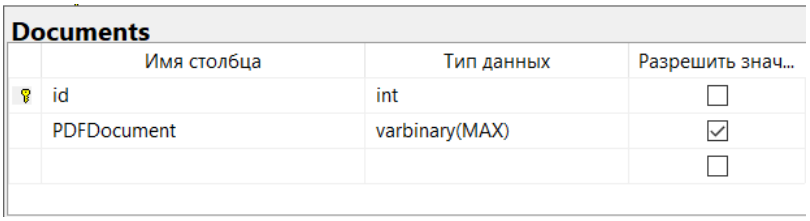


QRCodes

	Имя столбца	Тип данных	Разрешить знач...
🔑	id	int	☐
	QRImage	varbinary(MAX)	☑
			☐

Рисунок 3.6 – Таблица QRCodes

Таблица Documents хранит id подписанного PDF-документа и сам подписанный документ в бинарном формате. Проект для Documents представлен на Рисунке 3.7.



Documents

	Имя столбца	Тип данных	Разрешить знач...
🔑	id	int	☐
	PDFDocument	varbinary(MAX)	☑
			☐

Рисунок 3.7 – Таблица Documents

3.4. Диаграмма последовательностей

В целях визуализации взаимодействия объектов системы между собой во времени в едином сценарии использования используется ещё одна UML-диаграмма – диаграмма последовательностей. Данная диаграмма иллюстрирует, как различные части системы взаимодействуют друг с другом для выполнения функции, а также порядок, в котором происходит взаимодействие при выполнении конкретного случая использования.

Для отображения течения времени используется линия жизни объекта, которая изображается с помощью штриховой линии, которая проводится

вертикально вниз. С помощью линии жизни показывается период, в течение которого объект существует в системе. Сами объекты изображаются в виде прямоугольников, а сообщения, которыми они обмениваются – в виде линий со стрелками.

Диаграмма последовательностей представлена на Рисунке 3.8.

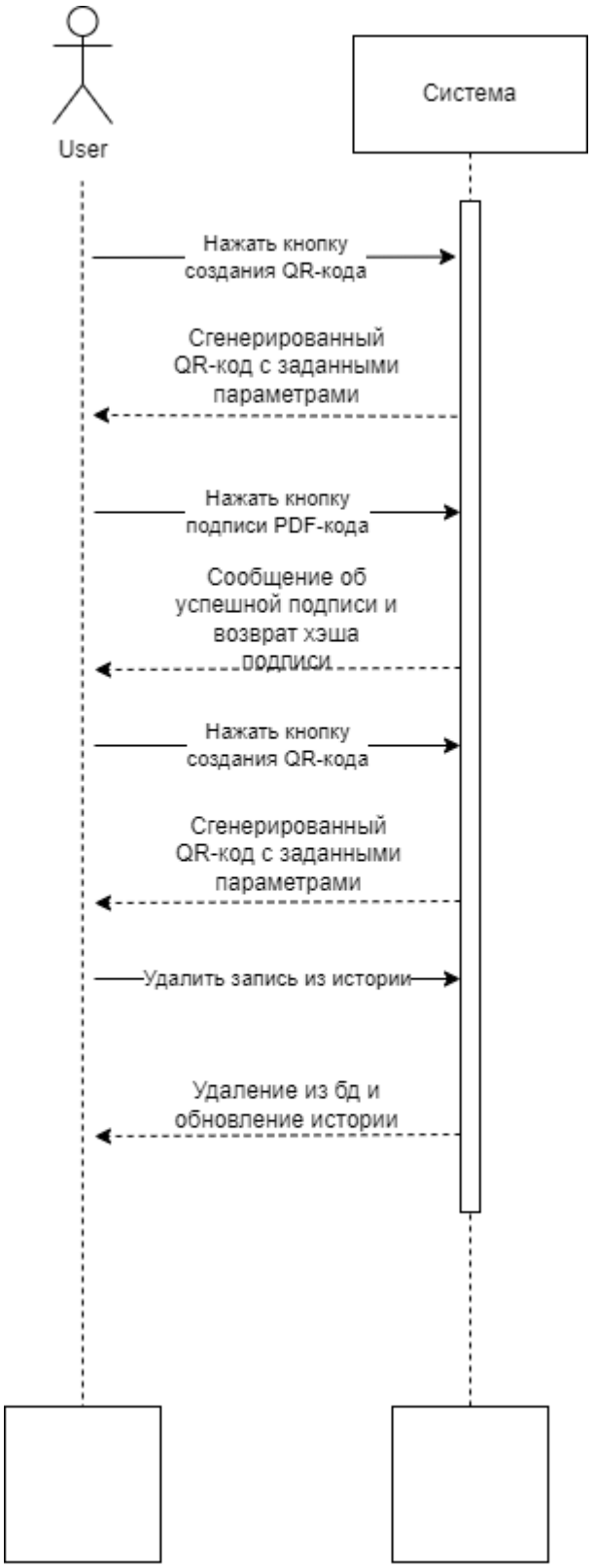


Рисунок 3.8. – Диаграмма последовательностей

3.5. Вывод

В данном разделе была описана структура программного средства, описаны основные элементы структуры, пояснения к ним. Также была разработана диаграмма классов, которая показывается взаимосвязи между сущностями программного средства и описывает модификаторы доступа данных необходимых для функционирования программного средства.

4. Реализация программного средства

4.1. Реализация ViewModel

Класс ViewModel содержит свойства которые связывают интерфейс и модель данных. Листинг класса ViewModel представлен на Листинге 4.1.

```
public class ViewModel : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    //Цвет QR-кода
    private static string _foregroundColor = "#FF000000";
    public string ForegroundColor
    {
        get { return _foregroundColor; }
        set
        {
            if (_foregroundColor != value)
            {
                _foregroundColor = value;
                OnPropertyChanged("ForegroundColor");
            }
        }
    }
    //Выбранная иконка
    private static SocialMediaService _selectedService;

    public SocialMediaService SelectedService
    {
        get { return _selectedService; }
        set
        {
            if (_selectedService != value)
            {
                _selectedService = value;
                OnPropertyChanged(nameof(SelectedService));
            }
        }
    }
    //Отступ от нижнего края документа
    private static int _left = 50;

    public int Left
    {
        get { return _left; }
        set
        {
            if (value != _left && value >= 0 && value <= 100)
            {
                _left = value;
                OnPropertyChanged(nameof(Left));
            }
            else if (value < 0)
            {
                _left = 0;
                OnPropertyChanged(nameof(Left));
```

```
            }
            else if (value > 100)
            {
                _left = 100;
                OnPropertyChanged(nameof(Left));
            }
        }
    }
    //Отступ от верхнего края документа
    private static int _top = 50;

    public int Top
    {
        get { return _top; }
        set
        {
            if (value != _top && value >= 0 && value <= 100)
            {
                _top = value;
                OnPropertyChanged(nameof(Top));
            }
            else if (value < 0)
            {
                _top = 0;
                OnPropertyChanged(nameof(Top));
            }
            else if (value > 100)
            {
                _top = 100;
                OnPropertyChanged(nameof(Top));
            }
        }
    }
    public void OnPropertyChanged([CallerMemberName] string prop = "")
    {
        if (PropertyChanged != null)
            PropertyChanged(this, new PropertyChangedEventArgs(prop));
    }
}
```

Листинг 4.1 – Класс ViewModel

В классе определены такие свойства, как: цвет QR-кода, который передается из палитры в пользовательском интерфейсе, по умолчанию задан черный цвет; выбранная иконка, которую пользователь может выбрать из выпадающего списка; отступ от верхнего и левого края документа, которые пользователь регулирует ползунками для размещения QR-кода на странице документа. Все эти свойства привязаны к соответствующим элементам управления в разметке программного средства режимом двусторонней привязки.

Для отступов реализована валидация, которая не позволяет выбрать значение меньше нуля и больше ста, поскольку регулировать ее можно не только ползунками, но и ручным вводом для удобства пользователя.

4.2. Реализация модели данных

Для определения контекста данных используется подход CodeFirst, который по заданным сущностям сам создает базу данных. Взаимодействие с ней осуществляется через класс Model, который представлен на Листинге 4.2.

```
public partial class Model : DbContext
 {

     public Model()
         : base("name=Model")
     {
     }

     public DbSet<Documents> Documents { get; set; }
     public DbSet<PDFHistory> PDFHistory { get; set; }
     public DbSet<QRCodes> QRCodes { get; set; }
     public DbSet<QRHistory> QRHistory { get; set; }

     protected override void OnModelCreating(DbModelBuilder
modelBuilder)
     {
         modelBuilder.Entity<Documents>()
             .HasMany(e => e.PDFHistory)
             .WithOptional(e => e.Documents)
             .HasForeignKey(e => e.DocumentId);

         modelBuilder.Entity<QRCodes>()
             .HasMany(e => e.QRHistory)
             .WithOptional(e => e.QRCodes)
             .HasForeignKey(e => e.QRCodeId);
     }
 }
```

Листинг 4.2 – Класс Model

В этом классе через DbSet определяются сущности базы данных, а именно: Documents, PDFHistory, QRCodes, QRHistory. Это классы, которые соответствуют таблицам базы данных с соответствующими полями.

4.3. Реализация интерфейса

Было принято решение организовать переключение между логическими компонентами интерфейса с помощью инструмента TabControl. Таким образом, получились следующие вкладки, которое представлены на Рисунке 4.3.

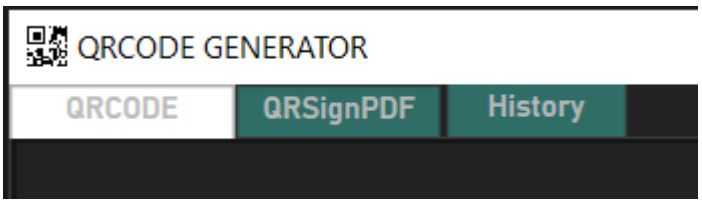


Рисунок 4.3 – Вкладки программного средства

На вкладке QRCODE находится интерфейс для создания QR-кода(Рисунок 4.4). На вкладке QRSignPDF находится интерфейс для подписи PDF-документов (Рисунок 4.5), а последняя вкладка History (Рисунок 4.6) содержит историю

созданий, хранит все созданный QR-коды на устройстве и подписанные PDF-документы.

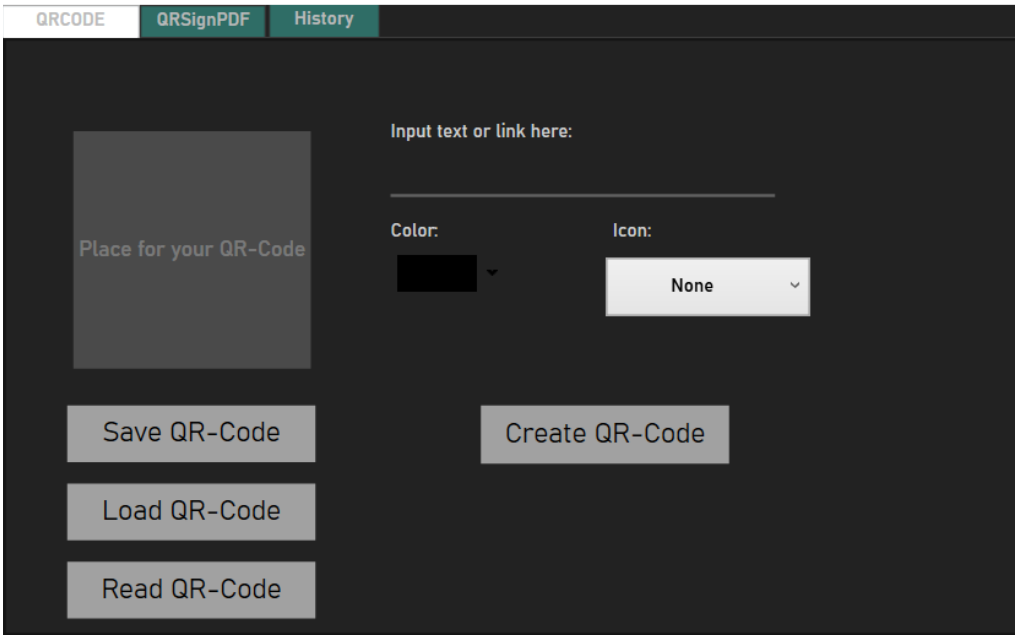


Рисунок 4.4 – Вкладка QRCODE

На этой вкладке находится место для расположения QR-кода, поле для ввода значения или ссылки, выпадающий список для выбора иконки, палитра для выбора цвета и кнопки для взаимодействия. Кнопка Create QR-Code создает QR-код с заданными пользователем свойствами. Если пользователь ничего не ввел, выбрасывается обрабатываемое исключение. MessageBox просит ввести что-то, прежде чем создавать QR-код. Кнопка Save QR-Code отвечает за сохранение полученного изображения QR-кода в png формате на устройство. Кнопка Load QR-Code отвечает за загрузку изображения QR-кода с устройства для чтения с помощью кнопки Read QR-Code.

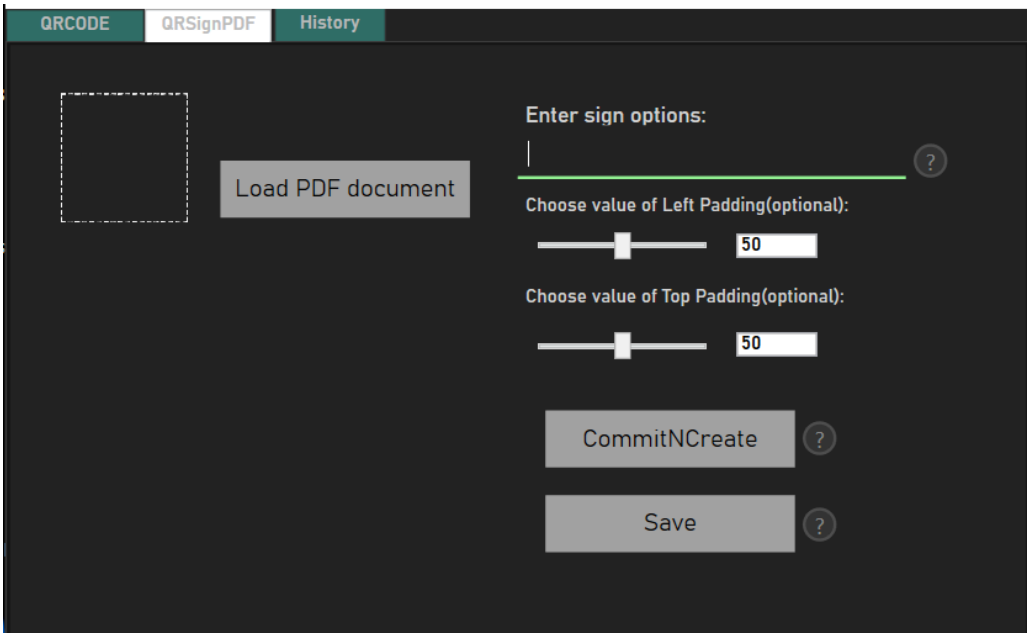


Рисунок 4.5 – Вкладка QRSignPDF

На этой вкладке присутствует поле для иконки документа, кнопка для загрузки документа (Load PDF document), когда документ загружен то появляется его название и иконка на месте границы из белого квадрата. Присутствует поле для ввода текста который будет передан вместе с данными о подписи в QR-код. Ползунками либо вводом определяются отсупы от верхней и нижней границ страницы документа. Кнопка Commit&Create подписывает выбранный документ и сохраняет его в истории созданий, а по кнопке Save можно сохранить подписанный документ.

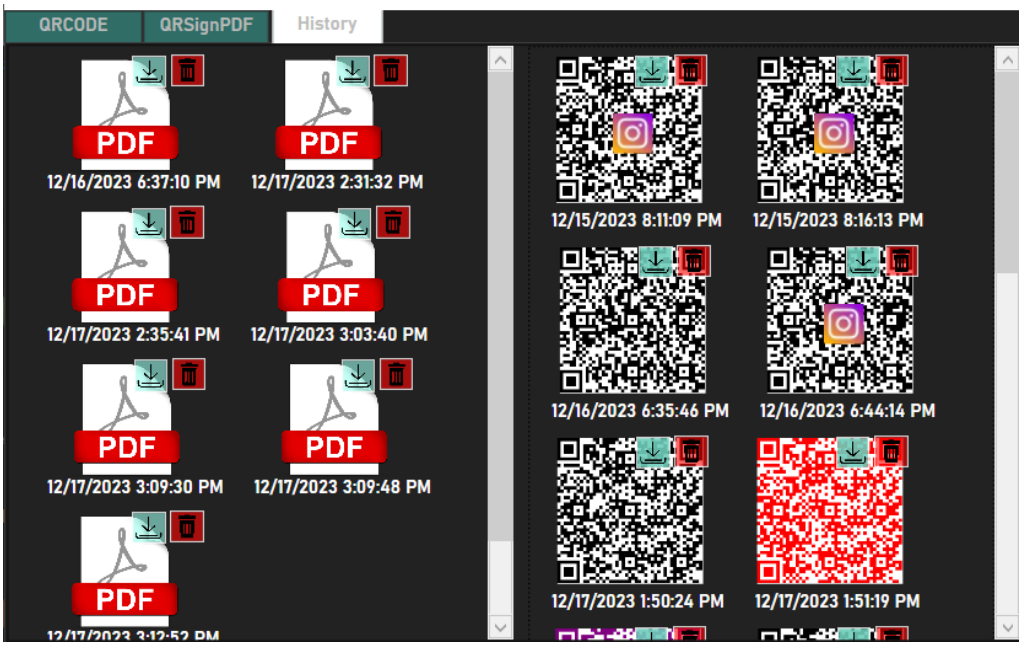


Рисунок 4.6 – Вкладка History

На этой вкладке находятся все созданные QR-коды и подписанный PDF-документы. Два элемента ScrollView связанны с данными из базы данных и подгружают элементы при запуске программного средства, а также при изменении данных (удаление, добавление). Каждый элемент из истории можно удалить из базы данных, либо загрузить на устройство.

4.4. Реализация ЭЦП

Для реализации электронной цифровой подписи необходим сертификат с закрытым ключом, который был найден в базовых сертификатах локального компьютера (Рисунок 4.7).

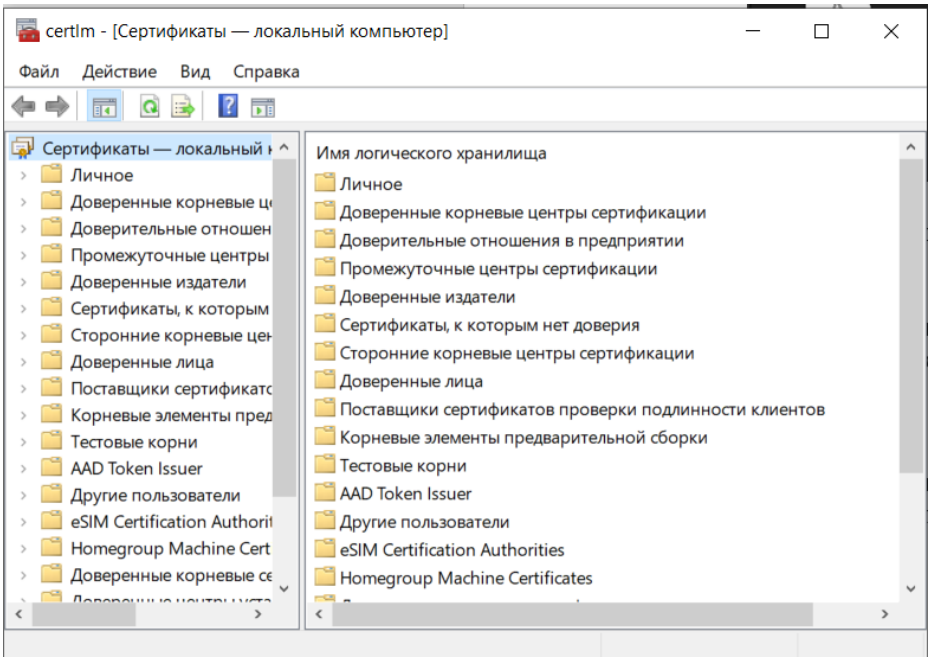


Рисунок 4.7 – Сертификаты локального компьютера

Подходящий сертификат был экспортирован в папку проекта с паролем, который в дальнейшем используется для электронной цифровой подписи PDF-документа.

В программном средстве подпись реализует функция SignPDF, которая представлена на Листинге 4.8.

```
public void SignPdf(string sourcePdfPath, string targetPdfPath,
X509Certificate2 certificate)
 {
     try
     {
         var reader = new PdfReader(sourcePdfPath);
         var stamper = PdfStamper.CreateSignature(reader, new
FileStream(targetPdfPath, FileMode.Create), '\0');

         var signatureAppearance = stamper.SignatureAppearance;
         signatureAppearance.SetVisibleSignature(new
iTextSharp.text.Rectangle(450, 748, 554, 780), 1, "Signature");

         var privateKey =
(RSACryptoServiceProvider)certificate.PrivateKey;
         var chain = new X509Chain();
         chain.Build(certificate);
         var certificateChain = new
List<Org.BouncyCastle.X509.X509Certificate>();

         foreach (var chainElement in chain.ChainElements)
         {
             var cert =
DotNetUtilities.FromX509Certificate(chainElement.Certificate);
             certificateChain.Add(cert);
         }
```

```
        var parameters = new X509Certificate2Signature(certificate,
DigestAlgorithms.SHA1);
        MakeSignature.SignDetached(signatureAppearance, parameters,
certificateChain, null, null, null, 0, CryptoStandard.CMS);

        stamper.Close();

        var newreader = new PdfReader(targetPdfPath);
        var signatures = newreader.AcroFields.GetSignatureNames();
        var qrCodeText = textBox_Sign.Text;

        foreach (var signatureName in signatures)
        {
            var pkcs7 =
newreader.AcroFields.VerifySignature(signatureName);

            if (pkcs7 != null && pkcs7.Verify())
            {
                int pageNumber =
newreader.AcroFields.GetRevision(signatureName);
                var pageContentBytes =
newreader.GetPageContent(pageNumber);

                using (var sha256 =
System.Security.Cryptography.SHA256.Create())
                {
                    var signatureHashBytes =
sha256.ComputeHash(pageContentBytes);
                    var signatureText =
BitConverter.ToString(signatureHashBytes).Replace("-", "");

                    MessageBox.Show("Signature Hash: " +
signatureText);

                    var subjectName = certificate.SubjectName.Name;
                    var issuerName = certificate.IssuerName.Name;
                    var serialNumber = certificate.SerialNumber;

                    qrCodeText += $"\nSignature is valid.\nSubject:
{subjectName}\nIssuer: {issuerName}\nSerial Number: {serialNumber}";
                }
            }
        }

        var qrCodeImage = GenerateQRCode(qrCodeText);

        var image = iTextSharp.text.Image.GetInstance(qrCodeImage);
        image.ScaleToFit(100, 100);
        image.SetAbsolutePosition(_viewModel.Left, 750 -
_viewModel.Top);

        using (var stamper1 = new PdfStamper(new
PdfReader(targetPdfPath), new FileStream(readyfile, FileMode.Create)))
        {
            stamper1.GetOverContent(1).AddImage(image);
            stamper1.Close();
        }
```

```
            Documents _document = new Documents(PdfToByteArray(readyfile));
            PDFHistory _qrHistory = new PDFHistory(DateTime.Now,
Environment.MachineName, _document.id);

            using (Model db = new Model())
            {
                db.Documents.Add(_document);
                db.PDFHistory.Add(_qrHistory);
                db.SaveChanges();
            }

            GetLists();
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
```

Листинг 4.8 – Функция SignPDF

Функция сначала открывает исходный PDF-документ, создает подпись и делает ее видимой на первой странице. Затем он строит цепочку сертификатов и создает подпись с использованием приватного ключа из сертификата. После этого код проверяет подпись, генерирует хеш подписи и добавляет информацию о подписи в текст QR-кода. Затем он генерирует QR-код, добавляет его в PDF и сохраняет изменения. Наконец, он сохраняет информацию о документе и истории в базе данных. Если в процессе возникает исключение, оно выводится в сообщении.

4.5. Вывод

В данном разделе были пояснены основные моменты реализации программного средства: создание модели данных и определение контекста данных, создание ViewModel для обеспечения передачи свойств из пользовательского интерфейса, рассмотрена реализация самого интерфейса и разбор всех вкладок приложения, реализация электронной цифровой подписи документов и разбор главной функции которая осуществляет подпись на основе сертификата подлинности.

5. Тестирование, проверка работоспособности и анализ полученных результатов

5.1. Тестирование обработки исключений и валидации

Первая проверяемая вкладка: QRCodes. Следует убедиться, что никакие действия пользователя не приведут к необрабатываемому исключению и в следствие к нежелаемому прерыванию работы программного средства.

QR-код создается на основе значения введённым пользователем, поэтому важно убедиться что поле для ввода не является пустым. Срабатывание исключение демонстрирует Рисунок 5.1.

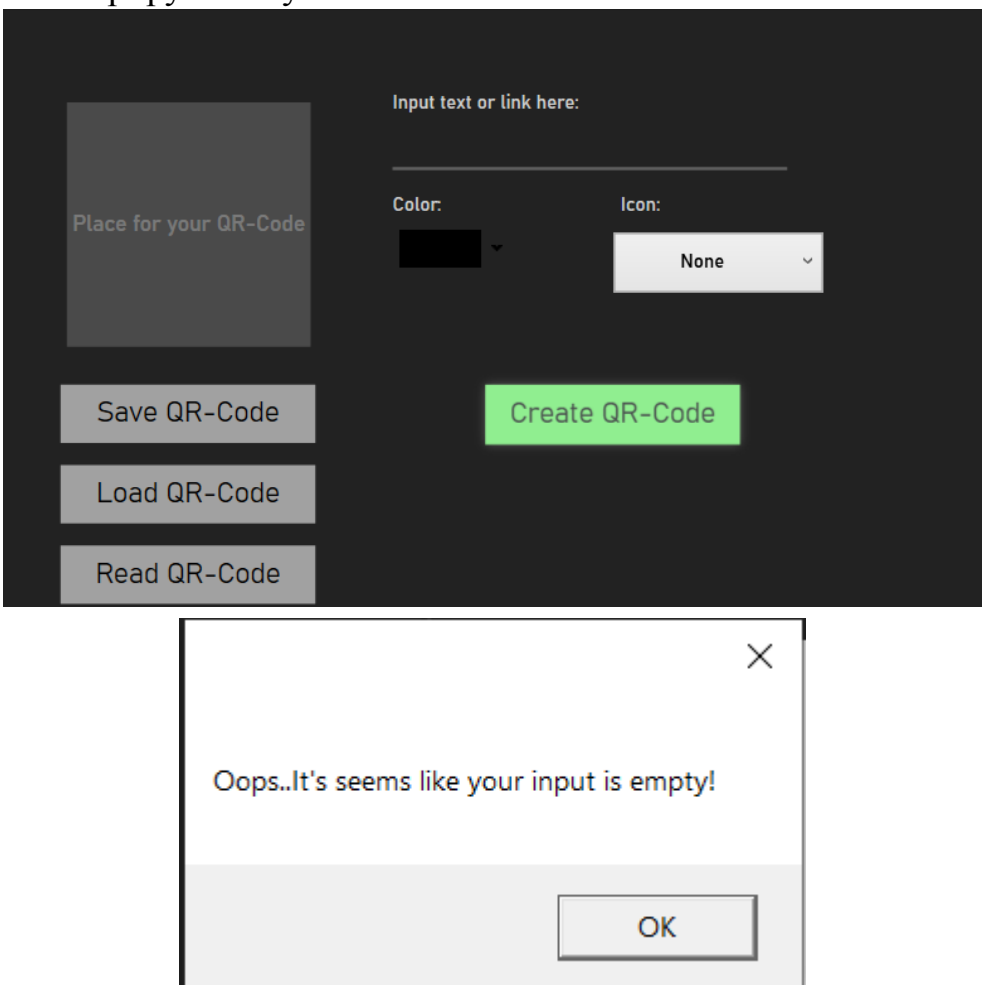


Рисунок 5.1 – Исключение кнопки «Create QR-Code»

В случае если пользователь оставил поле пустым и нажал кнопку создания QR-кода, он увидит сообщение о том, что поле пустое. Пользователю следует заполнить поле, чтобы было что кодировать, иначе сработает обрабатываемое исключение, которое уведомит его о незаполненном поле ввода.

Далее следует убедиться, что при вызове функций сохранения и чтения поле изображение QR-кода не пустое, поскольку невозможно сохранить или прочитать несуществующее изображение. Рисунок 5.2 демонстрирует срабатывание исключения в случае если кнопки Save и Read пытаются использоваться при пустом поле изображения.

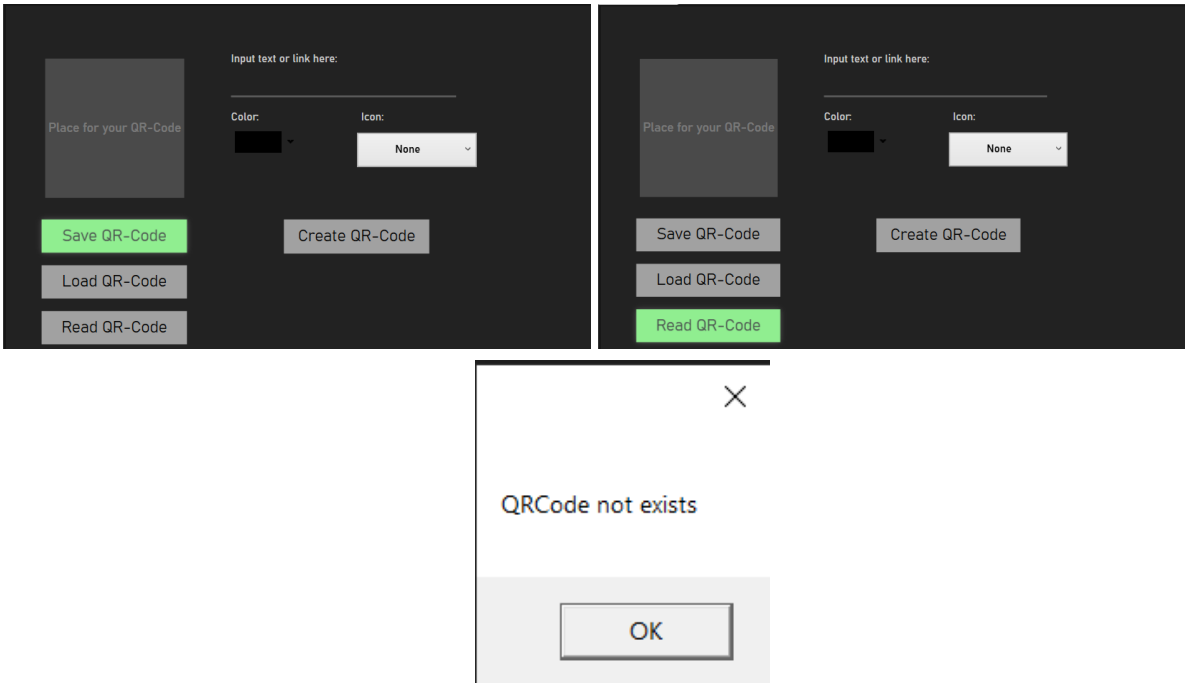
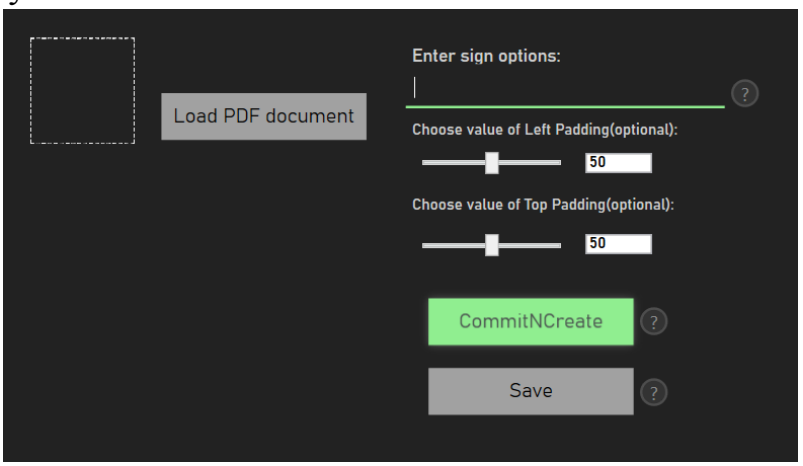


Рисунок 5.2 – Исключение кнопок «Save QR» и «Read QR»

В данном случае обработчик исключений возвращает сообщений «QRCode not exists» пользователю, уведомляя его о том, что прежде чем сохранять или читать QR-код ему необходимо создать его или загрузить.

Далее необходимо протестировать вкладку PDFSign на корректную обработку исключений.

Для подписи документа пользователю необходимо загрузить документ и ввести значение которое будет содержать QR-код, кроме данных о сертификате. Поэтому важно проверить наличие загруженного документа и значения в поле ввода. Демонстрация обработки исключений для заполняемых данных представлена на Рисунке 5.3.



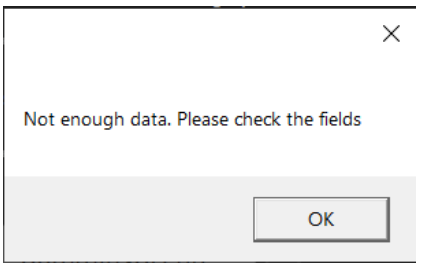


Рисунок 5.3 – Исключение для кнопки «CommitNCreate»

В данном случае пользователю возвращается сообщение о том что данных для подписи недостаточно и просьба ввести данные для возможности подписи документа.

Также для полей для регулировки отступа разработана валидация, которая не позволяет ввести отрицательное значение либо значение больше ста. Демонстрирует это Рисунок 5.4.

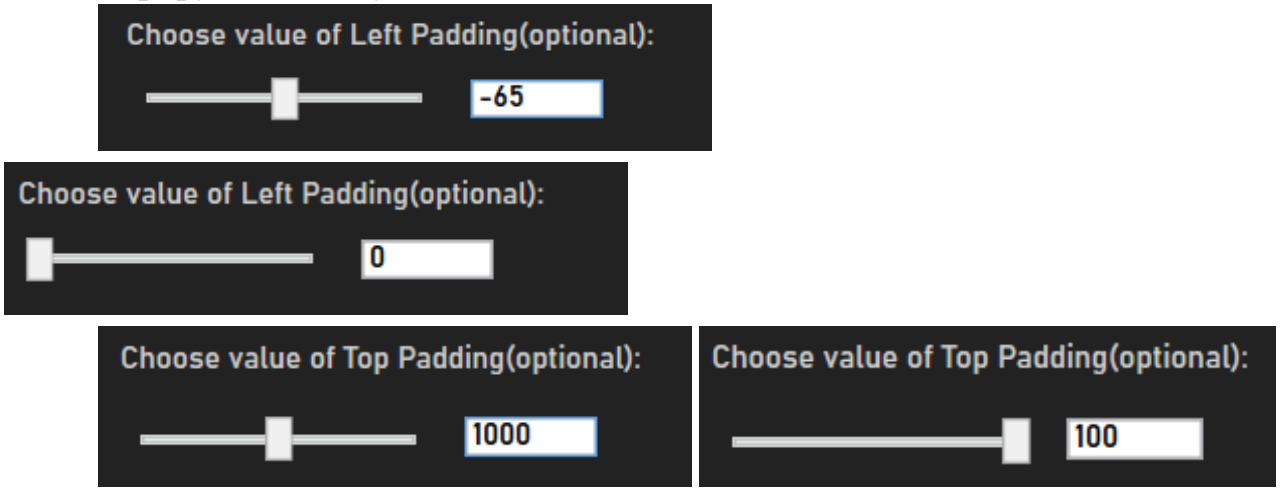


Рисунок 5.4 – Валидация полей регулировки отступа

При введенном значении меньше нуля валидация устанавливает значение поля равным нулю. В случае, когда введенное значение превышает 100, то валидация устанавливает значение поля равным 100.

5.2. Тестирование основных функций

Первая вкладка разработана для создания QR-кода. При выборе параметров она возвращает изображение QR-кода в поле для изображений (Рисунок 5.5).

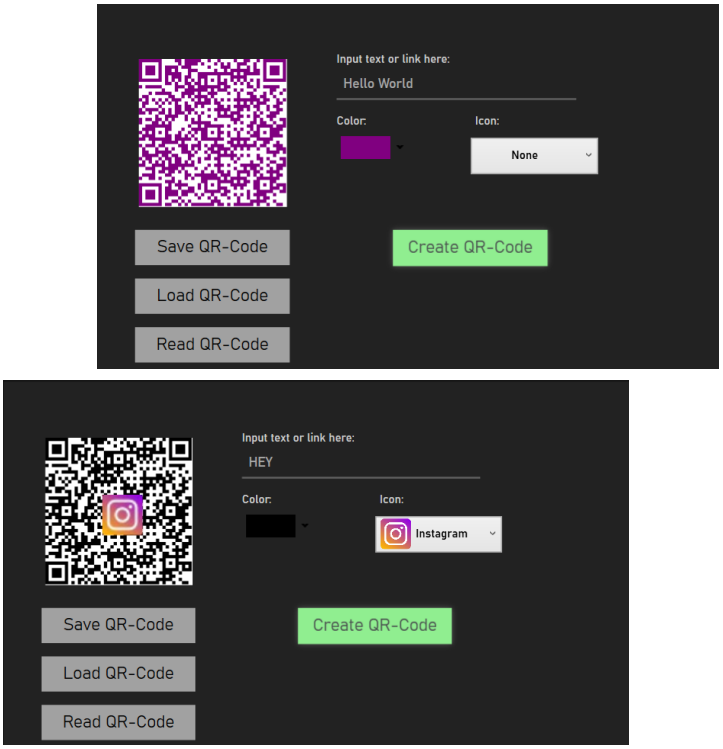


Рисунок 5.5 – Функция создания QR-кода

Также окно содержит функции сохранения (Рисунок 5.6), загрузки (Рисунок 5.7) и чтения QR-кода (Рисунок 5.8).

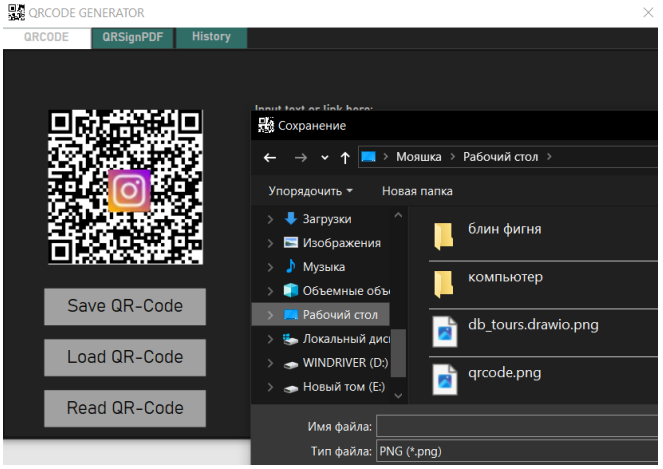


Рисунок 5.6– Функция сохранения QR-кода

Функция открывает проводник для выбора места и названия изображения для сохранения QR-кода. Она сохраняет изображение в формате png или jpg на выбор пользователя.

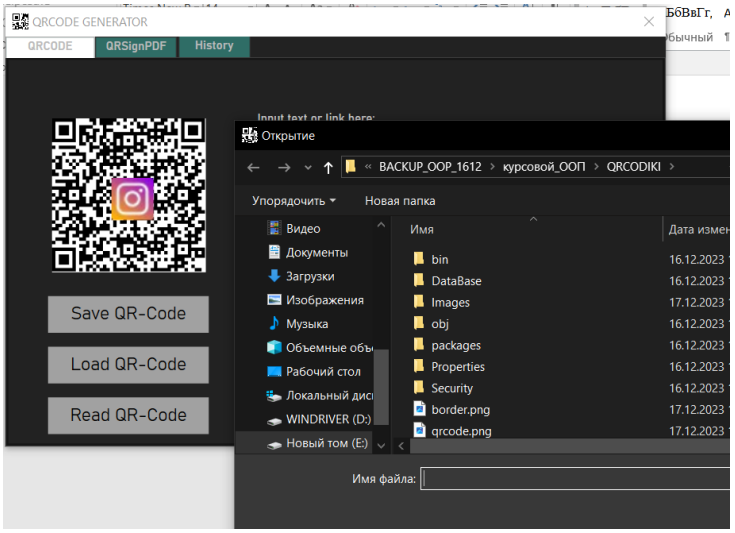


Рисунок 5.7 – Функция загрузки QR-кода

Эта функция также открывает проводник, но уже для выбора изображения в формате png или jpg и передает это изображение в поле для изображения в интерфейс пользователя.

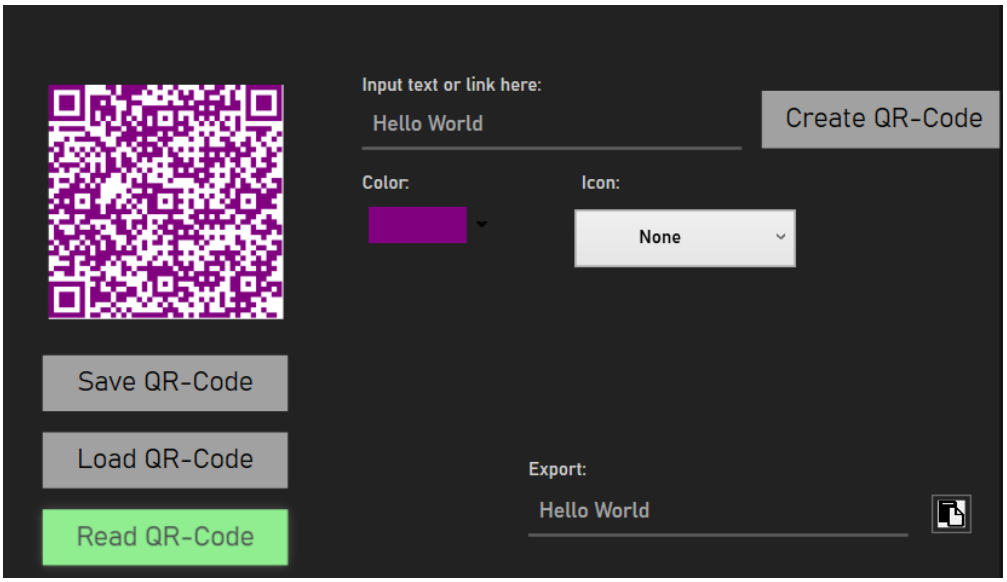


Рисунок 5.8 – Функция чтения QR-кода

Функция раскодирует QR-код и передает полученное значение в появляющееся поле для экспорта, значение которого можно скопировать в буфер обмена по кнопке справа. После копирования дополнительное поле пропадает за ненадобностью.

Далее тестируем работоспособность функции подписи PDF-документа. Для этого со всеми заполненными полями следует нажать кнопку «CommitNCreate». Демонстрация работы кнопки представлена на Рисунке 5.9.

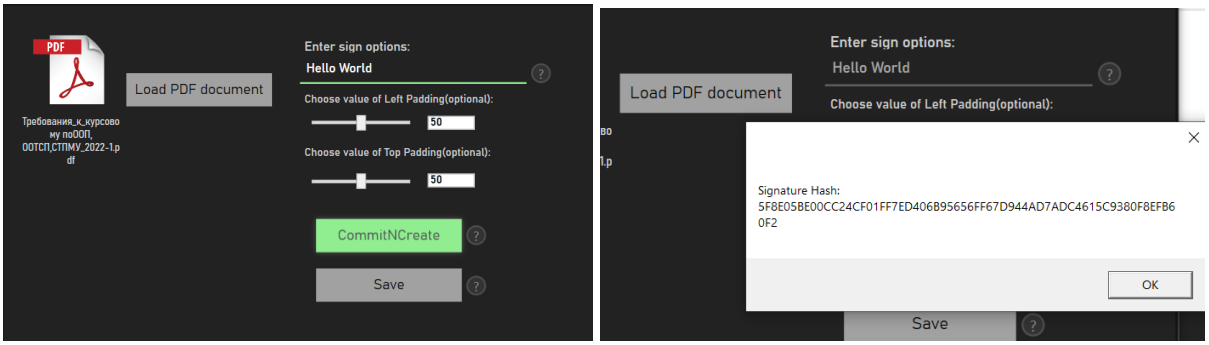


Рисунок 5.9 – Функция подписи PDF-документа.

В случае успешной подписи программа возвращает хэш подписи, что уведомляет пользователя об успешном выполнении функции. Далее следует сохранить подписанный документ через кнопку «Save» (Рисунок 5.10).

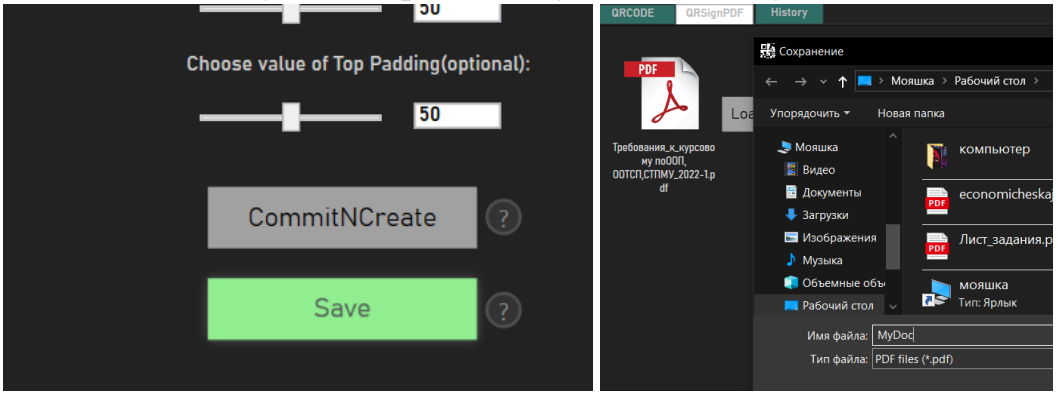


Рисунок 5.10 – Функция сохранения документа

Функция открывает проводник позволяя пользователю выбрать место сохранения документа и присвоить ему имя. Документ сохраняется в формате pdf. Для проверки выполнения подписи открываем подписанный документ (Рисунок 5.11).

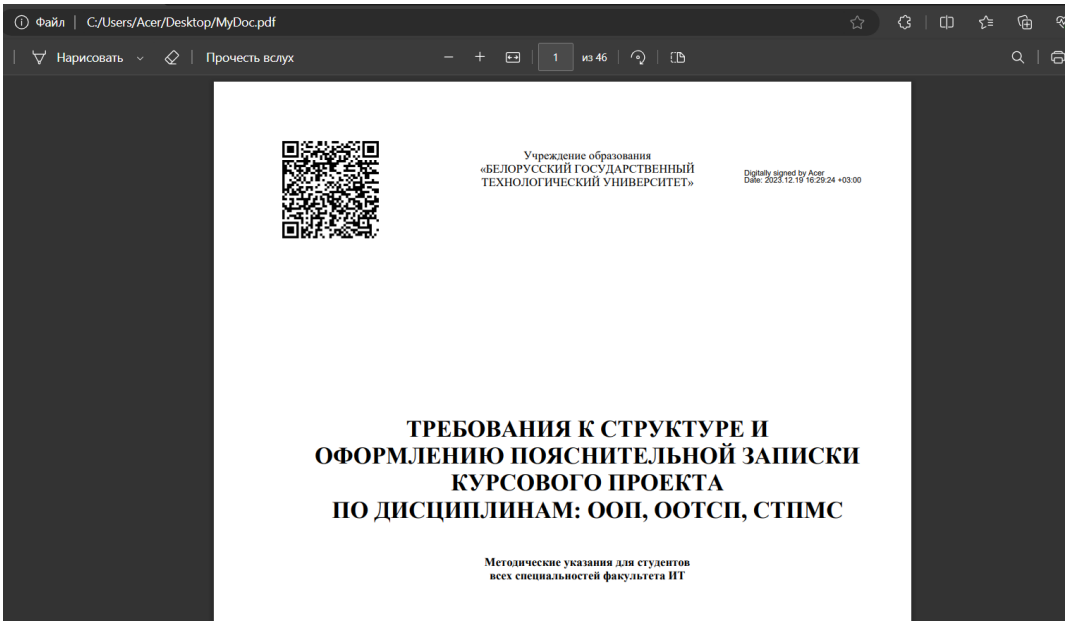


Рисунок 5.11 – Подписанный PDF-документ

История хранит все созданные QR-коды и подписанные документы и связана с базой данных, в любой момент их можно удалить из истории и базы данных соответственно. Также любой элемент можно сохранить на устройство (Рисунок 5.12).

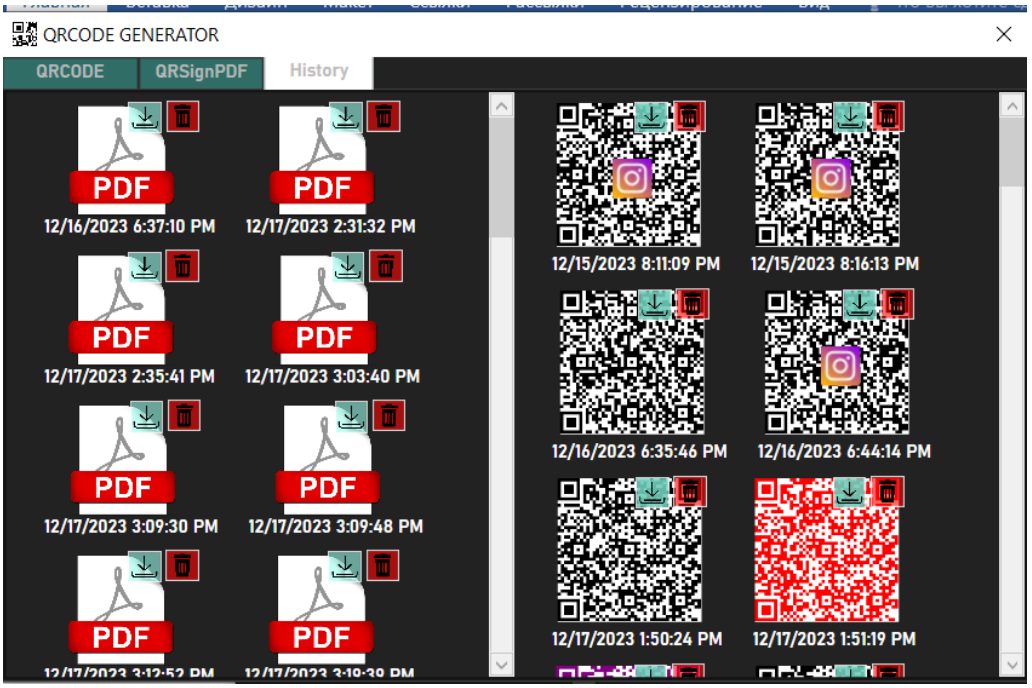


Рисунок 5.12 – История созданий

История действительно содержит все созданный QR-коды и подписанные документы и дает возможность сохранять и/или удалять их.

5.3 Вывод

В ходе тестирования была проверена обработка исключений и было подтверждено, что пользователь не сможет случайно выбить не обрабатывающееся исключение, что приведет к сбою выполнения программы, проверена валидация полей. Также были проверены основные функциональные возможности приложения, такие как создание, загрузка, сохранение, чтение QR-кода, электронная цифровая подпись PDF-документов и их сохранение. Подтвердилось, что история созданий содержит все последние созданные элементы, которые можно сохранить и/или удалить с помощью кнопок, которые располагаются рядом с каждым элементом. Проведенный анализ получаемых данных подтвердил работоспособность программного средства.

Заключение

В заключении данной работы было разработано программное средство для цифровой подписи PDF-документов на основе QR-кода. В ходе работы были выполнены следующие этапы:

- Постановка задачи и обзор литературы, включая анализ алгоритмов решения, обзор прототипов и актуальность задачи.
- Проектирование архитектуры проекта, включая структуру модулей и классов.
- Разработка функциональной модели и модели данных программного средства, включая выполнение функций генерации и чтения QR-кода, загрузки QR-кода с устройства, сохранения QR-кода на устройство и цифровой подписи PDF-документа на основе QR-кода.
- Тестирование разработанного программного средства.

В ходе работы над проектом были разработаны диаграммы, которые описывают структуру программы, возможности пользователя, структуру выполнения программы. Были разработаны такие диаграммы, как: диаграмма классов, диаграмма последовательностей, диаграмма использования, логическая и физическая схемы базы данных.

Основные моменты реализации программного средства были описаны в соответствующем разделе, с приложениями листингов кода.

Результаты тестирования подтвердили корректность работы программного средства и его соответствие заявленным требованиям. Разработанное программное средство может быть использовано для создания QR-кода и для цифровой подписи PDF-документов на основе QR-кода, что позволяет упростить и автоматизировать процесс подписания документов.

Таким образом, цели и задачи проекта были успешно достигнуты, и работа может считаться завершенной.

Список литературных источников

1. 14 лучших программ, чтобы создать QR-код [Электронный ресурс] / Режим доступа: https://www.qr-code-generator.com/blog/best-qr-code-generators/.

2. Бесплатный онлайн-генератор QR-кодов: Создать QR-код для кодирования [Электронный ресурс] / Режим доступа: https://www.qr-code-generator.com/start-create-qr-codes-free/.

3. 7 лучших генераторов QR-кодов в 2023 году — Сервисы на vc.ru [Электронный ресурс] / Режим доступа: https://vc.ru/services/169572-7-luchshih-generatorov-qr-kodov-v-2023-godu.

4. Бесплатный генератор QR-кодов URL - QRCode Online [Электронный ресурс] / Режим доступа: https://www.qrcodeonline.net/.

5. Крючков В.Е. - QR-коды. Их создание и применение. – 2008 г.

6. Подписание документов PDF в Adobe Acrobat [Электронный ресурс] / Режим доступа: https://helpx.adobe.com/ru/acrobat/using/signing-pdfs.html.

7. Как подписать ПДФ (PDF) файл электронной подписью (ЭЦП) [Электронный ресурс] / Режим доступа: https://www.digisigner.com/ru/free-electronic-signature/how-to-sign-a-pdf.

8. Подписать PDF ЭЦП: простые способы добавить цифровую подпись, возможные [Электронный ресурс] / Режим доступа: https://www.digisigner.com/ru/free-electronic-signature/sign-pdf.

9. Как подписать PDF в электронном виде и получить электронные подписи [Электронный ресурс] / Режим доступа: https://acrobat.adobe.com/ru/ru/sign/free-trial-global.html.