МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
**«Южно-Уральский государственный университет
(национальный исследовательский университет)»**
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка приложения для визуализации данных геоинженерных изысканий с использованием OpenStreetMap

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-312.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ И.И. Клебанов

Автор работы,
студент группы КЭ-401
_____ В.Е. Севастьянов

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___»_____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-401
Севастьянову Валентину Евгеньевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)
Разработка приложения для визуализации данных геоинженерных изысканий с использованием OpenStreetMap.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Разработка на языке C# Visual Studio. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/?view=vs-2022 (дата обращения: 03.02.2024 г.).

3.2. OpenStreetMap [Электронный ресурс] URL: https://www.openstreetmap.org (дата обращения: 06.03.2024 г.).

3.3. Документация PostgreSQL и Postgres Pro. [Электронный ресурс] URL: https://postgrespro.ru/docs (дата обращения: 25.03.2024 г.).

3.4. GMap.NET. Страница разработчика. [Электронный ресурс] URL: https://github.com/radioman/greatmaps (дата обращения: 10.03.2024 г.).

4. **Перечень подлежащих разработке вопросов**

4.1. Провести анализ предметной области и существующих решений.

4.2. Спроектировать архитектуру интерфейс приложения.

4.3. Спроектировать и внедрить базу данных в приложение.

4.4. Реализовать использование OpenStreetMaps.

4.5. Протестировать систему

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. И.И. Клебанов

Задание принял к исполнению В.Е. Севастьянов

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

Современный мир стремительно развивается в направлении цифровизации и автоматизации процессов в различных сферах деятельности. В контексте геоинженерных исследований, важным является доступ к актуальным географическим данным. Развитие технологий и программных средств позволяет создавать мощные инструменты для обработки и визуализации геологических данных, что позволит пользователям повысить эффективность своей работы. Десктопные приложения, становятся все более востребованными среди специалистов в области геологии и строительства.

Современные технологии предоставляют геологическим предприятиям удобные возможности для хранения и управления большим объемом заказов на проведение исследований. С учетом того, что предприятия часто имеют сотни заказов, каждый из которых требует хранения значительного объема данных, важно иметь эффективную систему хранения и управления информацией.

Десктопные приложения, предоставляют удобные инструменты для организации хранения заказов и связанных с ними данных. Геологические предприятия могут использовать такие приложения для создания базы данных заказов, где каждый заказ может быть легко идентифицирован, отслежен и управляем.

На сегодняшний день отсутствует широко распространенное решение, предоставляющее удобный интерфейс для визуализации геологических данных с возможностью отмечать на карте географические точки и прикреплять к ним соответствующие файлы. Создание такого приложения представляет собой актуальную задачу, которая может быть востребована в различных областях, таких как геология, инженерное строительство, геодезия и другие.

Постановка задачи

Целью выпускной квалификационной работы является разработка десктопного приложения, с помощью которого можно будет делать отметки на карте, а также к этим отметкам прикреплять файлы некоторых типов.

Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ предметной области и существующих решений;
2) спроектировать архитектуру и интерфейс приложения;
3) спроектировать и внедрить базу данных;
4) реализовать использование OpenStreetMaps;
5) протестировать систему.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, и списка литературы. Объем работы составляет 35 страницы, объем списка литературы – 15 источников.

В первой главе рассматриваются похожие проекты, приводится обзор средства разработки приложения – WPF (Windows Presentation Foundation), а также обзор OpenStreetMaps и его сравнение с аналогами.

Вторая глава посвящена анализу функциональных и нефункциональных требований к системе, также приводится проектирование архитектуры разрабатываемой системы.

В третьей главе приводятся детали реализации различных компонентов системы, предоставляются листинги программы и пояснения к ним.

В четвертой главе представлены результаты функционального тестирования разработанной системы.

В заключении были приведены основные результаты, полученные при выполнении выпускной квалификационной работы.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Обзор существующих аналогов

На сегодняшний день существует не так много сервисов или приложений, которые позволили бы оставлять отметки на картах, и прикреплять к этим отметкам файлы.

Один из самых популярных в мире сервисов – это Google Карты. Этот набор приложений был создан в 2005 году компанией Google. Сервис Google Карт представляет из себя интерактивную карту, на которой можно ставить метки, метки можно разделять на представленные категории, а также добавлять свои. Добавление файлов реализовано тоже, однако с ограничениями. Добавлять можно фотографии или же видео, одним из минусов такой системы является то, что фото и видео остаются в общем доступе, следовательно, никаких конфиденциальных данных загрузить не получится. Скриншот интерфейса Google карт представлен на рисунке 1.

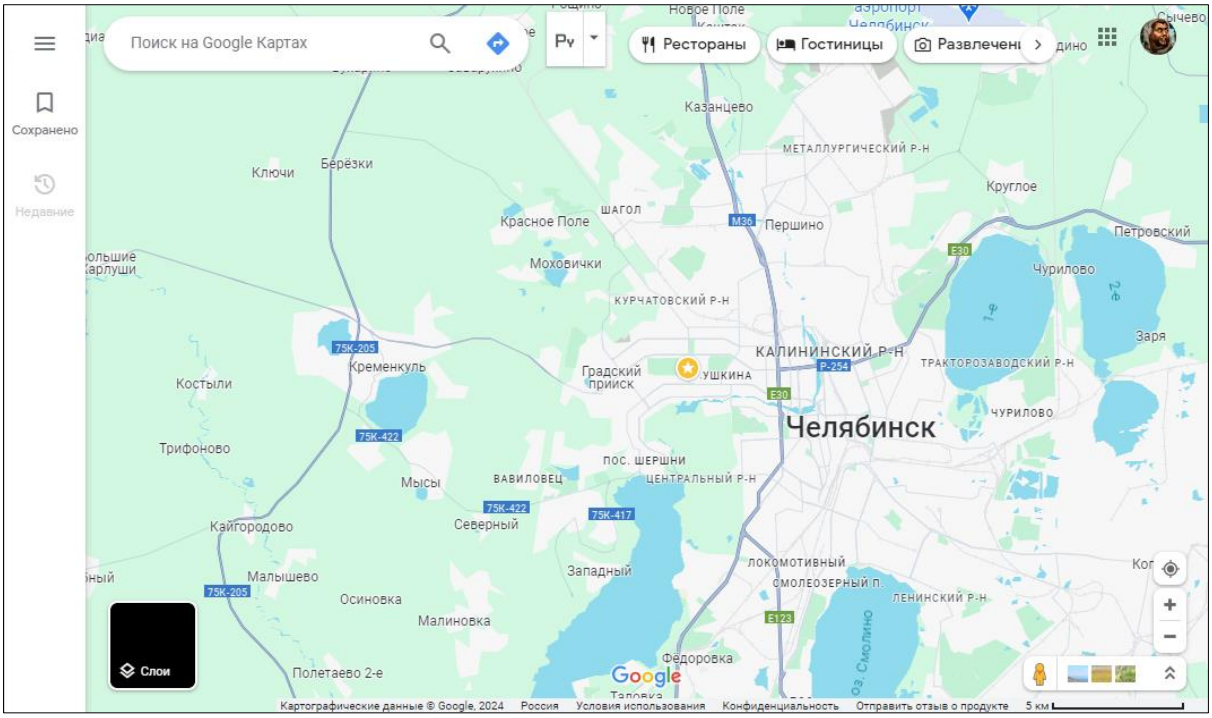

Рисунок 1 – Интерфейс Google карт

На территории России одним из популярных вариантов является 2ГИС. Это электронный справочник с картами, который выпускается с

1999 года компанией «ДубльГИС». Данное приложение позволяет просматривать карты России и еще 11 стран [2].

К особенностям приложения можно отнести то, что карты каждой области загружаются отдельно, значит их можно просматривать и без интернета. Касательно меток, которые можно ставить на карте, в данном приложении функционал не так велик, как в тех же Goggle картах. Можно написать текст заметки, однако прикрепить какие-либо файлы нельзя. Пример интерфейса изображен на рисунке 2.

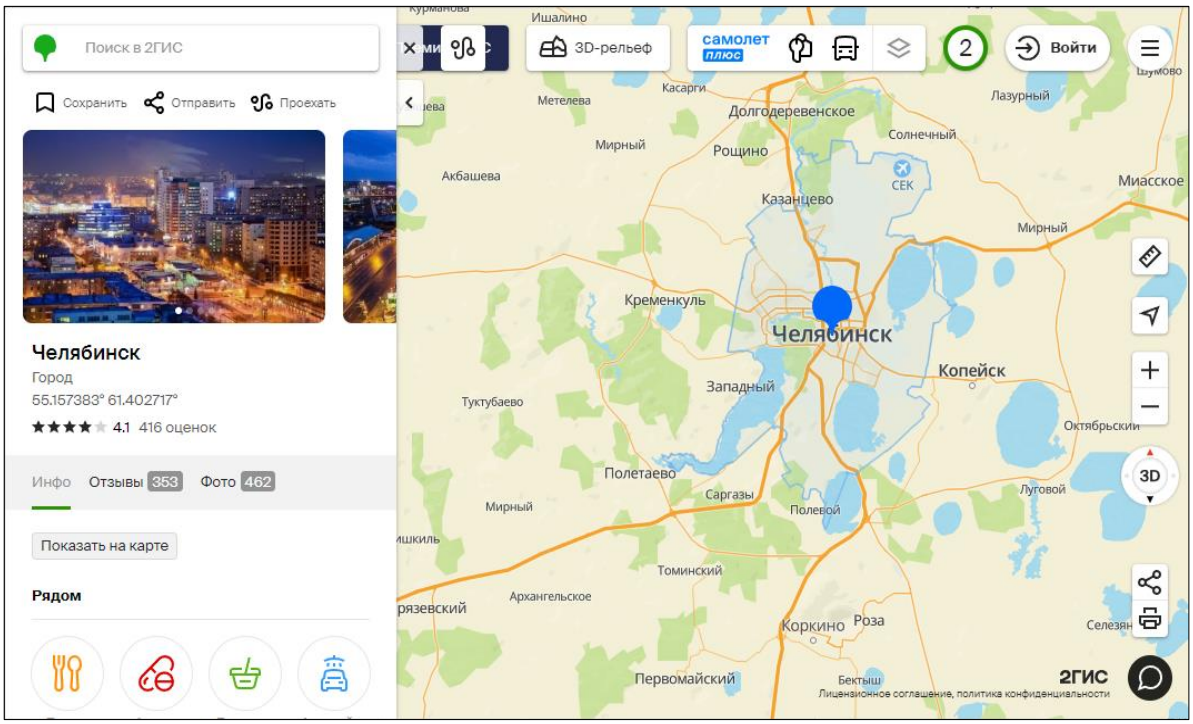

Рисунок 2 – Интерфейс 2ГИС

1.2. Обзор существующих сервисов с интеграцией карт

Сегодня существует ряд сервисов, предоставляющих возможность интеграции интерактивных карт в десктопные приложения. Однако, несмотря на разнообразие предложений, актуальность данных, удобство использования и гибкость настроек остаются ключевыми критериями при выборе подходящего сервиса. В данном аспекте важно провести анализ текущего состояния сервисов и выявить их сильные и слабые стороны.

Одним из активно развивающихся сервисов является Google Maps Platform. Он предоставляет широкие возможности для работы с географическими данными и обладает высокой степенью надежности благодаря поддержке крупной корпорации Google. Данный сервис позволяет внедрить карты в десктопное приложение, однако, чтобы пользоваться полным функционалом данного сервиса необходимо платить. Главная страница сайта изображена на рисунке 3.

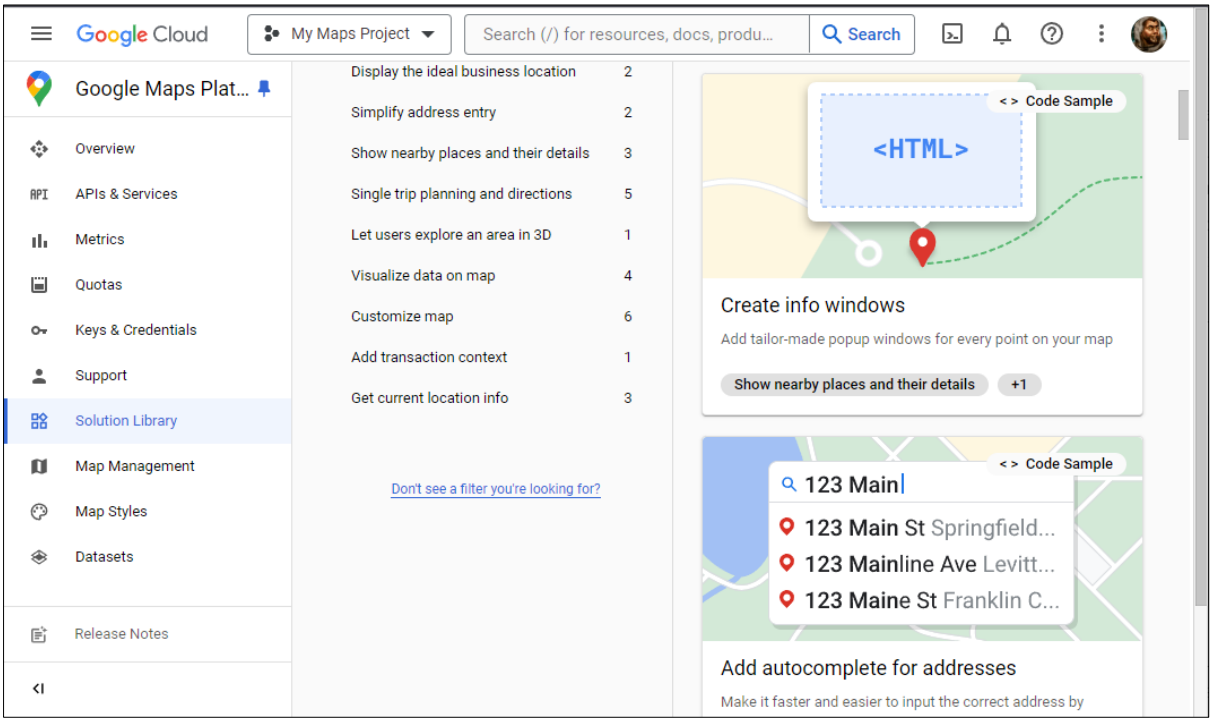

Рисунок 3 – Главная страница сервиса Google maps platform

Конкурентом Google в сфере интеграции карт в приложения и веб-сервисы выступает MapBox. Пожалуй, самым приятным плюсом данного сервиса является обширная и информативная документация, которая облегчает работу с сервисом и использование его функционала. Также к плюсам относится удобный и приятный интерфейс. Однако все это доступно лишь после оплаты услуг.

Достаточно популярный вариант для внедрения актуальных карт в десктопные приложения и веб-сервисы – OpenStreetMap. Данный сервис предоставляет работу с открытыми данными. Картографические данные

OSM доступны для свободного использования, изменения и распространения, что делает их идеальным выбором для создания приложения. Актуальность карт ничем не хуже вышеперечисленных аналогов [5]. Ключевой причиной выбора данного сервиса стало то, что он полностью бесплатен в использовании, а также свободно доступен на территории России. Главная страница сервиса OpenStreetMaps изображена на рисунке 4.

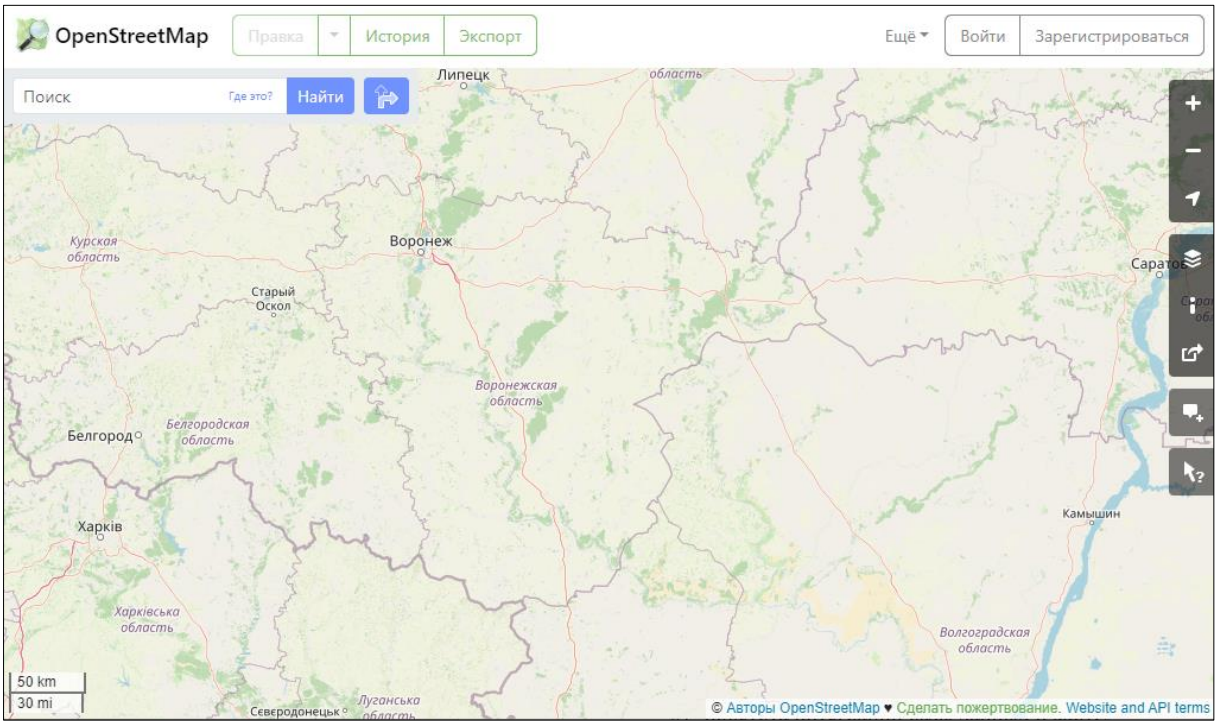

Рисунок 4 – Главная страница сервиса OpenStreetMaps

1.3. Обзор WPF и сравнение с аналогами

Windows Presentation Foundation (WPF) – это фреймворк для создания графических пользовательских интерфейсов в десктопных приложениях под операционные системы Windows. Разработанный корпорацией Microsoft, WPF предоставляет разработчикам широкие возможности для создания современных и стильных пользовательских интерфейсов с использованием XML-подобного языка разметки (XAML) и языка программирования C#. WPF имеет следующие преимущества:

1) информативная и обширная документация;
2) векторная графика и анимации;

3) простота интеграции базы данных и карт;

4) интуитивный интерфейс.

Аналогом WPF является WinForms от тех же Microsoft, однако этот фреймворк является более старой и простой версией WPF. WinForms тесно интегрируется с .NET Framework, что позволяет использовать все возможности этой платформы, включая доступ к обширным библиотекам классов .NET и поддержку популярных языков программирования, таких как C# и VB.NET. WinForms пригоден для создания небольших приложений, однако функционал WPF более обширный.

Для мультиплатформенной разработки подойдет Electron. Это платформа, которая позволяет создавать приложения как для Windows, так и для MacOS и Linux. Она использует веб-технологии (HTML, CSS, JavaScript). К плюсам можно отнести:

1) кроссплатформенность;

2) онлайн обновления приложения.

Вывод по первой главе

В данной главе были рассмотрены сервисы, функционал которых схож с целями выпускной квалификационной работы. Были определены преимущества и недостатки выбранного для разработки приложения программного обеспечения. Также рассмотрены некоторые существующие сегодня сервисы для интеграции карт в десктопное приложение и обосновано, почему был выбран именно OpenStreetMaps. В конце было проведено сравнение WPF с аналогами.

2. ПРОЕКТИРОВАНИЕ

2.1. Требования к системе

Функциональные требования определяют, каким должно быть поведение продукта в тех или иных условиях. Они определяют, что разработчики должны создать, чтобы пользователи смогли выполнить свои задачи (пользовательские требования). Функциональные требования описываются в форме традиционных утверждений со словами «должен» или «должна»: «У пассажира должна быть возможность распечатать посадочные талоны на все рейсы, на которые он зарегистрировался» или «Если в профиле пассажира не указаны предпочтения по выбору места, система резервирования должна сама назначить ему место».

Благодаря анализу предметной области были выделены следующие функциональные требования к приложению.

1. При использовании приложения пользователь должен иметь возможность ставить метки на карте.

2. Пользователь должен иметь возможность прикреплять файлы некоторых разрешений к метке.

Нефункциональные требования – это описание свойства или особенности, которым должна обладать система, или ограничение, которое должна соблюдать система. В ходе анализа были выявлены следующие нефункциональные требования.

1. Приложения должно быть реализовано при помощи фреймворка WPF.

2. Интеграция карт должна быть реализована при помощи OpenStreetMaps.

3. Хранение и использование прикрепленных файлов должно быть реализовано при помощи PostgreSQL.

2.2. Диаграмма вариантов использования

Для проектирования системы был применен язык UML (язык графического описания для объектного моделирования). На основе требований к проектируемой системе была разработана диаграмма вариантов использования, которая отражает взаимодействие внешнего актера «пользователь» с программной системой. Данная диаграмма представлена на рисунке 5.

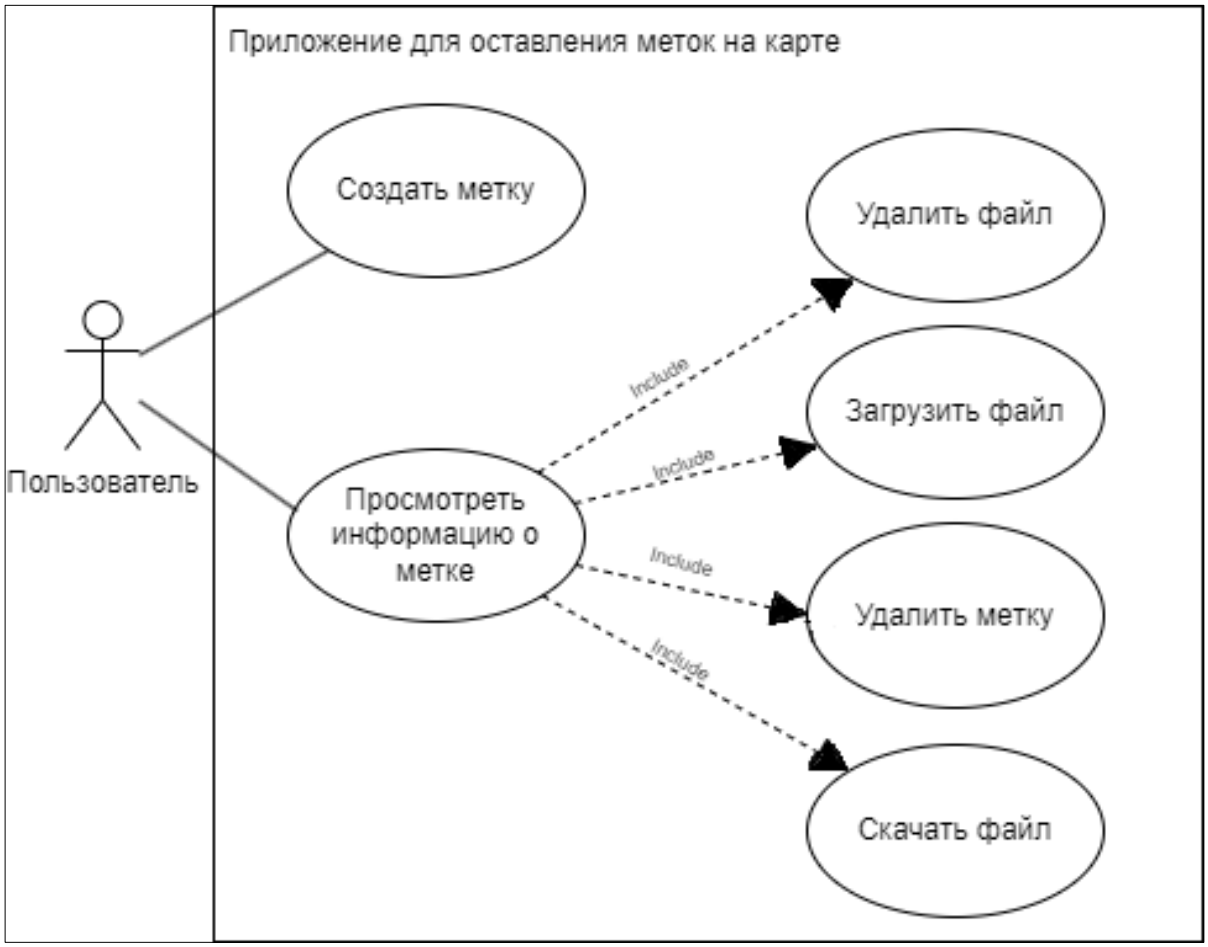

Рисунок 5 – Диаграмма вариантов использования

Пользователь может использовать следующие функции приложения.

1. Создать метку – пользователь может оставить метку в конкретном месте, чтобы в дальнейшем иметь возможность прикрепить к этой метке файлы, на карте появляется красная точка.

2. Посмотреть информацию о точке – пользователь может открыть дополнительное окно с информацией о метке, оно содержит название метки и список файлов, прикрепленных к этой метке.

3. Удалить файл – пользователь может удалить файл в окне дополнительной информации.

4. Загрузить файл – пользователь может прикрепить к метке дополнительные файлы в окне информации о метке, для этого нужно указать путь к файлу.

5. Скачать файл – пользователь может скачать выбранный файл в окне информации о метке.

6. Удалить метку – пользователь имеет возможность удалить метку и все прикрепленные к ней файлы.

2.3. Проектирование схемы базы данных

Схема базы данных приложения будет состоять из двух таблиц – `Marks` и `Files`. На рисунке 6 изображена схема базы данных.

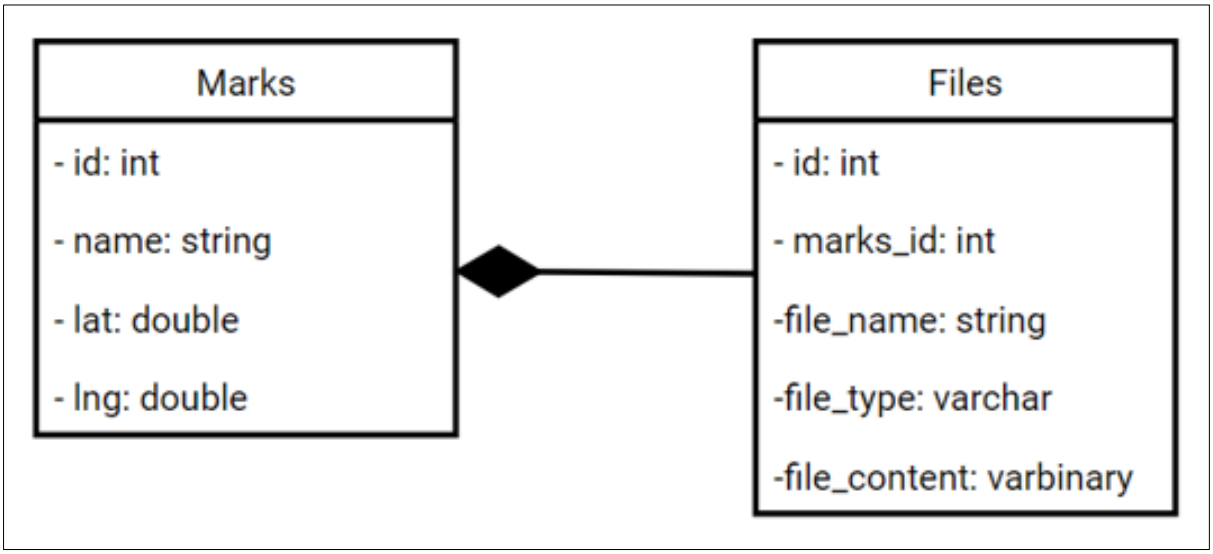

Рисунок 6 – Схема базы данных приложения

Таблица Files хранит в себе данные о файлах, которые прикрепили к меткам и имеет следующие поля:

1) id – уникальный идентификатор файла;
2) marks_id – уникальный идентификатор метки;
3) file_name – название загруженного файла;
4) file_content – содержимое файла представленное в бинарном виде.

Таблица Marks хранит координаты поставленной метки и ее название и имеет следующие поля:

1) id – уникальный идентификатор метки;
2) name – название метки;
3) lat – значение долготы;
4) lng – значение широты.

2.4. Проектирование архитектуры

Для обеспечения эффективной реализации функциональных требований к системе разрабатывается следующая архитектура приложения.

Интерфейс пользователя

Включает в себя графический пользовательский интерфейс, разработанный с использованием Windows Presentation Foundation (WPF), который предоставляет удобный и интуитивно понятный способ взаимодействия пользователя с приложением.

Компонент для работы с картами

Данный компонент отвечает за отображение интерактивной карты и реализацию функций добавления меток и поиска адресов на карте. Для этого может использоваться встроенный компонент WPF для работы с картами или сторонняя библиотека, например, OpenStreetMap.

Модуль управления данными

Включает в себя компоненты и сервисы, отвечающие за хранение, обработку и доступ к данным приложения. В качестве базы данных для хранения информации о метках и прикрепленных файлах используется PostgreSQL.

Модуль обработки файлов

Данный модуль предназначен для загрузки, хранения и обработки прикрепленных файлов типа XML и DOCX. Он обеспечивает возможность связи файлов с соответствующими метками на карте и их отображения при необходимости.

В приложении будет применен такой архитектурный подход как Model-View-Controller [9].

MVC – это архитектурный шаблон, который используется для создания структурированных и организованных программных приложений. Он разделяет приложение на три взаимосвязанных компонента: модель (Model), представление (View) и контроллер (Controller).

Рассмотрим основные компоненты MVC.

1. Модель (Model). Модель представляет собой центральный компонент архитектуры MVC. Она отвечает за представление данных и бизнес-логики приложения. Модель содержит данные и методы для работы с ними, а также реализует бизнес-правила и логику приложения.

2. Представление (View). Данный компонент представляет собой графический интерфейс пользователя (GUI) или другой способ представления информации. Отвечает за отображение данных пользователю и взаимодействие с ним. Оно принимает данные от модели и представляет их в удобном для пользователя виде.

3. Контроллер (Controller). Контроллер выступает в роли посредника и отвечает за управление взаимодействием между моделью и представлением. Обрабатывает пользовательский ввод, получает данные из модели и обновляет представление в соответствии с этими данными.

Схема архитектурного подхода MVC изображена на рисунке 7. На ней продемонстрированы связи между основными компонентами контроллера, модели и представления. Показано как они взаимодействуют друг с другом и с помощью чего именно.

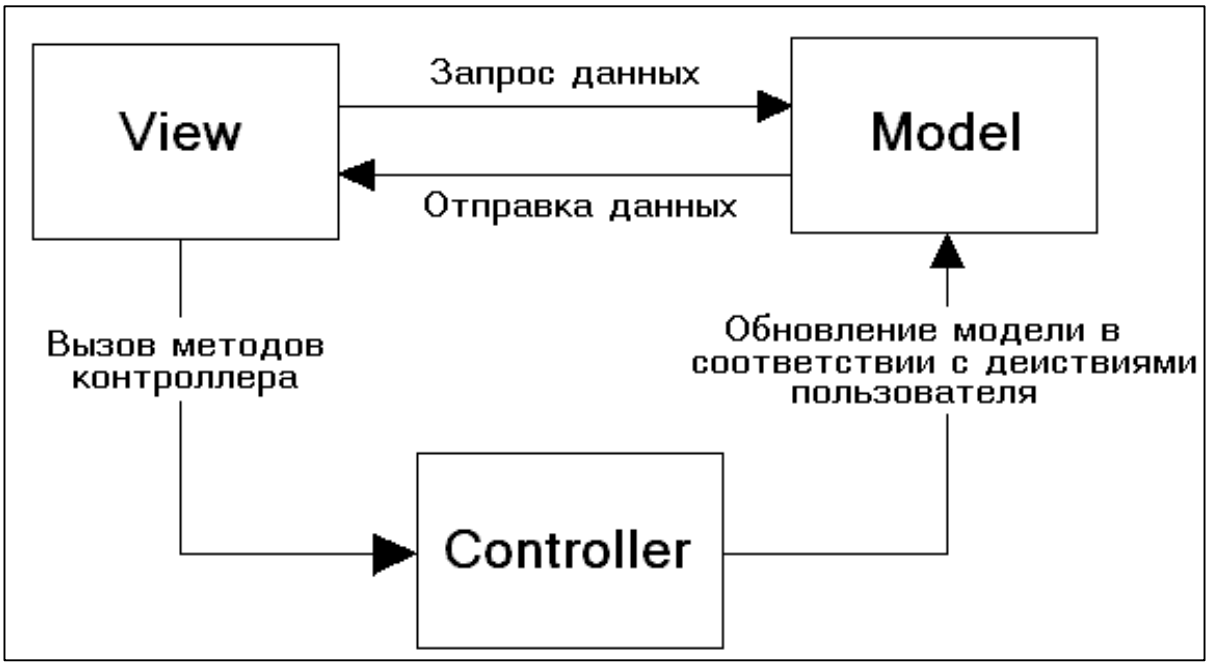

Рисунок 7 – Схема подхода MVC

Вывод по второй главе

В данной главе были выделены и рассмотрены функциональные и нефункциональные требования, выбрана архитектура приложения MVC, описано как она устроена и в чем ее преимущества в рамках данного приложения. Была спроектирована база данных, созданы две основные таблицы и показаны связи между ними. Также была представлена диаграмма вариантов использования, которая демонстрирует взаимодействия пользователя и системы.

3. РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

3.1. Программные средства

В процессе разработки приложения для постановки меток и загрузки файлов были использованы следующие средства.

1. Windows Presentation Foundation (WPF) – это фреймворк для создания десктопных приложений под операционную систему Windows, который обеспечивает разработку современных и интерактивных пользовательских интерфейсов с использованием языка разметки XAML (eXtensible Application Markup Language) и языка программирования C#.

2. Язык C# – это универсальный объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он был создан в начале 2000-х годов и стал основным языком разработки для платформы .NET.

3. XAML (eXtensible Application Markup Language) – это декларативный язык разметки, используемый в технологиях, связанных с разработкой пользовательских интерфейсов, в основном на платформе Microsoft. Он предоставляет удобный способ описания пользовательского интерфейса приложения, его структуры, визуального представления и взаимодействия с элементами управления.

4. PostgreSQL – это мощная и расширяемая объектно-реляционная система управления базами данных (СУБД), которая предоставляет надежное хранение и обработку данных для широкого спектра приложений.

5. OpenStreetMap (OSM) – это коллаборативный проект по созданию бесплатной и открытой картографической базы данных мира. Он был запущен в 2004 году и предоставляет пользователям возможность создавать, редактировать и обновлять картографические данные по всему миру.

6. GMap.NET – это бесплатная и открытая библиотека для .NET, которая предоставляет инструменты для интеграции интерактивных карт Google Maps, Bing Maps и OpenStreetMap в приложения на платформе .NET.

3.2. Интерфейс приложения

UI дизайн, или дизайн пользовательского интерфейса, включает в себя разработку визуальных элементов, с помощью которых пользователь взаимодействует с продуктом. Это включает в себя цветовую гамму, типографику, иконки, кнопки и прочие элементы интерфейса [4].

Интерфейс приложения состоит из одного главного окна и двух вызываемых окон. Главная страница содержит карту OpenStreetMap, которая была внедрена в приложение при помощи библиотеки GMap.NET. На карте есть красные метки к которым прикреплены файлы. Справа от карты находится таблица, которая хранит в себе список добавленных на карту меток. Над таблицей находится кнопка «Обновить». Расположение описанных выше элементов интерфейса можно увидеть на рисунке 8.

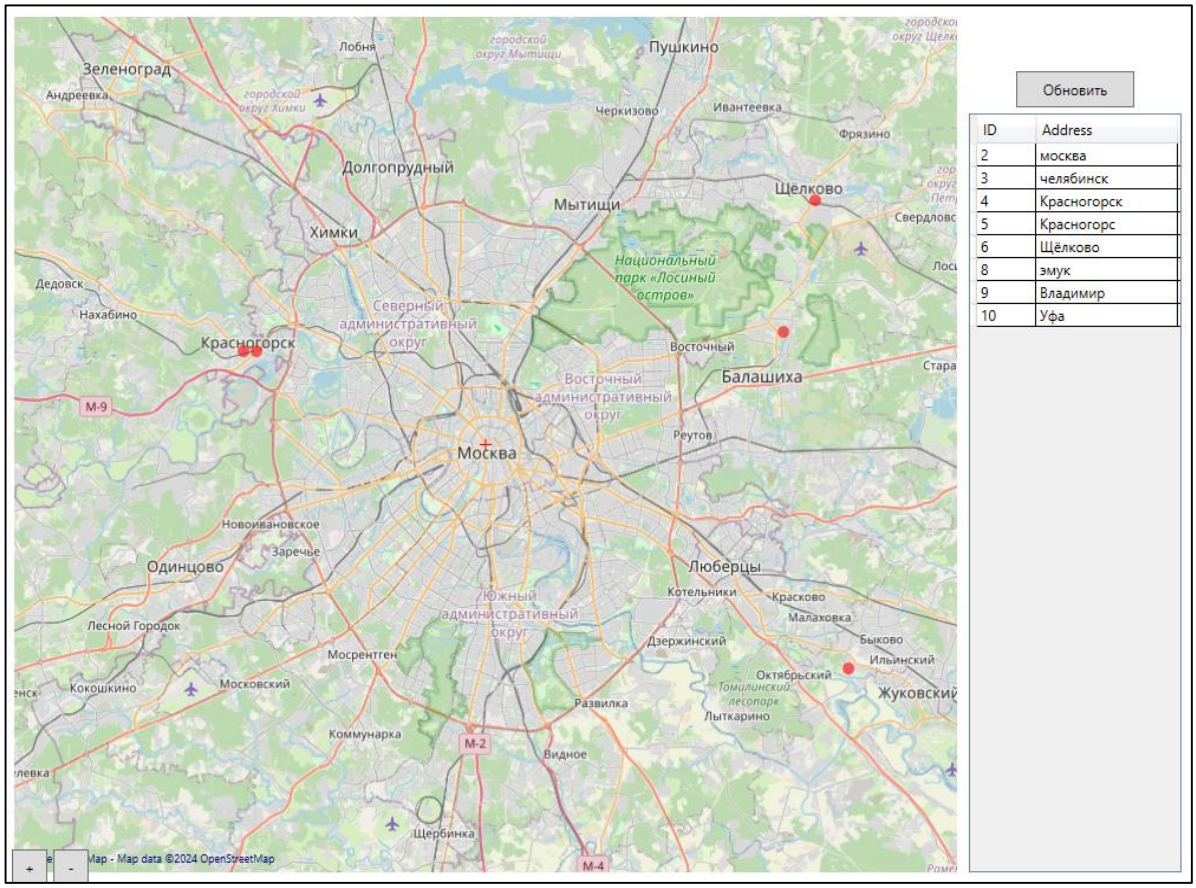

Рисунок 8 – Главная страница приложения

При нажатии левой кнопкой мыши на карте, появляется окно создания метки. Окно состоит из поля для ввода текста, в него нужно вводить название метки, кнопки «загрузить файл» и поля для названия файла, оно заполняется автоматически. Также есть две кнопки взаимодействия «Сохранить» и «Отмена» (рисунок 9).

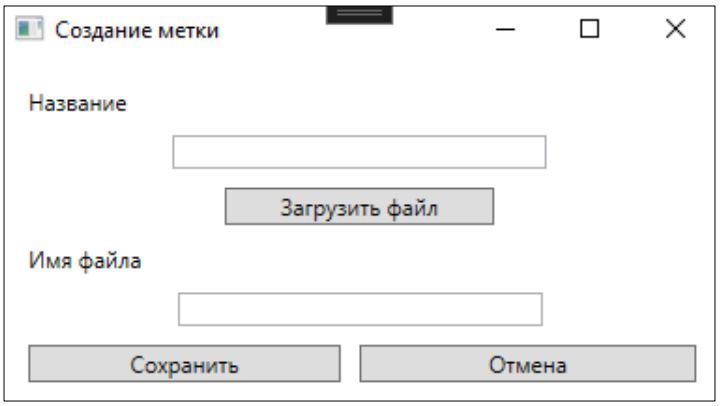

Рисунок 9 – Окно создания метки

При нажатии на кнопку «Загрузить файл» вызывается метод `UploadFileButton_Click` (листинг 1), который в переменную `FileName` записывает название метки, в переменную `FileData` записывает данные загруженного файла в бинарном виде.

Листинг 1 – Метод для загрузки файлов

```
private void UploadFileButton_Click(object sender, RoutedEventArgs e)
{
   OpenFileDialog openFileDialog = new OpenFileDialog();
   if (openFileDialog.ShowDialog() == true)
   {
      FileName = openFileDialog.FileName;
      FileNameTextBox.Text = System.IO.Path.GetFileName(FileName);
      FileData = File.ReadAllBytes(FileName);
   }
}
```

При нажатии на кнопку «Сохранить» вызывается метод `SaveButton_Click` (листинг 2), который в переменную Address записывает название метки и присваивает переменной DialogResult значение true, а затем закрывает окно.

Листинг 2 – Метод для сохранения результатов

```
private void SaveButton_Click(object sender, RoutedEventArgs e)
{
    Address = AddressTextBox.Text;
    DialogResult = true;
    Close();
}
```

Также если дважды нажать по любой метке из таблицы откроется окно информации о метке (рисунок 10). Оно состоит из двух текстовых строк, которые содержат ID и название метки, табличной части, которая содержит названия файлов и их ID и кнопки «Загрузить новые файлы» и «Удалить метку».

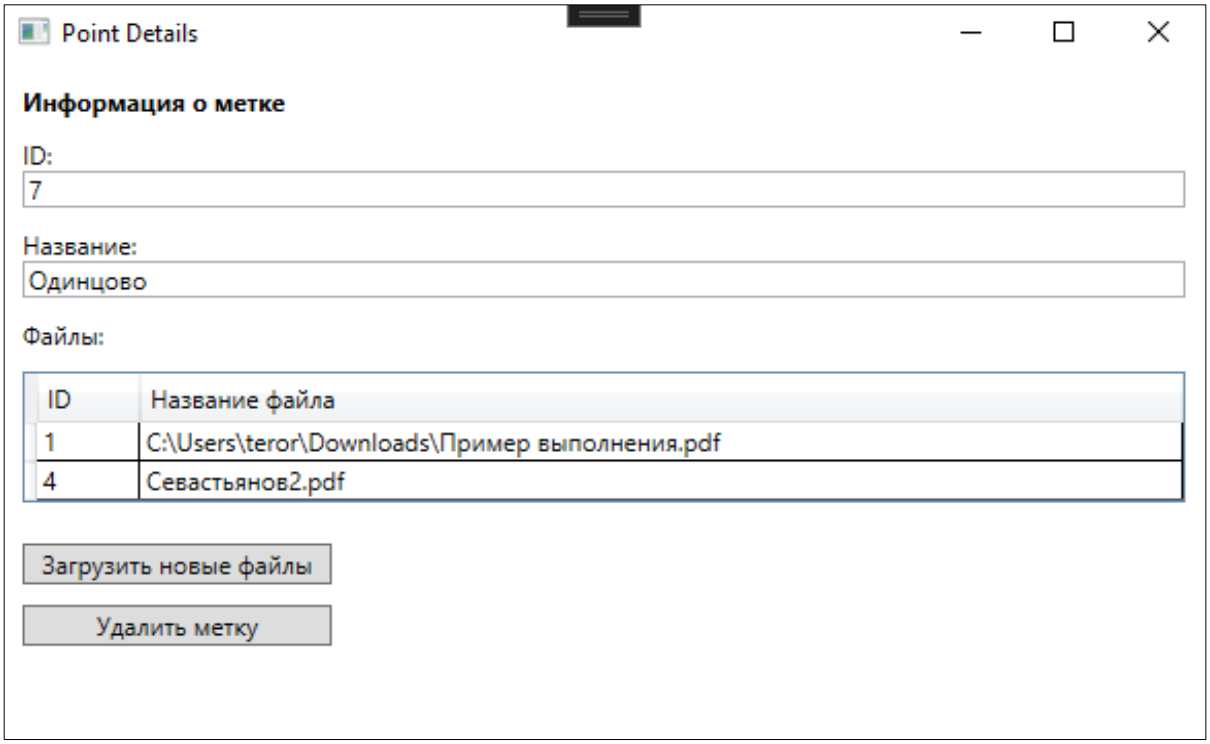

Рисунок 10 – Окно информации о метке

Если правой кнопкой мыши нажать по элементу таблицы появится меню с двумя вариантами взаимодействия «скачать файл» и «удалить файл».

«Скачать файл» вызывает метод `DownloadFile_Click` (листинг 3), который получает данные файла из базы и далее открывает Проводник, для того чтобы пользователь указал путь для сохранения файла, после этого файл скачивается по указанному пути.

Листинг 3 – Метод для скачивания файла

```
private void DownloadFile_Click(object sender, RoutedEventArgs e)
{
    var selectedFile = (Tuple<int, int, string, byte[]>)FilesDataGrid.Se-
lectedItem;
    if (selectedFile != null)
    {
        int fileId = selectedFile.Item1;
        string fileName = selectedFile.Item3;

        byte[] fileData = files.GetFileById(fileId).fileData;

        if (fileData != null)
        {
            SaveFileDialog saveFileDialog = new SaveFileDialog();
            saveFileDialog.FileName = fileName;
            if (saveFileDialog.ShowDialog() == true)
            {
                try
                {
                    File.WriteAllBytes(saveFileDialog.FileName, fileData);
                    MessageBox.Show("Файл успешно скачан и сохранен.");
                }
                catch (Exception ex)
                {
                    MessageBox.Show("Ошибка при сохранении файла: " +
ex.Message);
                }
            }
        }
    }
}
```

«Удалить файл» вызывает метод `DeleteFile_Click` (листинг 4), который уточняет, нужно ли удалять метку, и при положительном выборе удаляет ее из базы данных.

Листинг 4 – Метод для удаления файлов конкретной метки

```
private void DeleteFile_Click(object sender, RoutedEventArgs e)
{
    var selectedFile = (Tuple<int, int, string, byte[]>)FilesDataGrid.Se-
lectedItem;
    if (selectedFile != null)
    {
        int fileId = selectedFile.Item1;

        MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите
удалить этот файл?", "Подтверждение удаления", MessageBoxButton.YesNo, Mes-
sageBoxImage.Question);
        if (result == MessageBoxResult.Yes)
        {
            files.DeleteFile(fileId);
            LoadFiles();
        }
     }
}
```

При нажатии на кнопку «Загрузить новые файлы» вызывается метод `UploadFileButton_Click`.

При нажатии на кнопку «Удалить метку» вызывается метод `DeleteButton_Click`, который уточняет, нужно ли удалять метку, при положительном ответе метка удаляется из базы данных и окно закрывается (листинг 5).

Листинг 5 – Метод для удаления метки и всех ее файлов

```
private void DeleteButton_Click(object sender, RoutedEventArgs e)
{
    int selectedPoint = PointId;
    if (selectedPoint >= 0)
    {
        MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите
удалить эту метку и все связанные с ней файлы?", "Подтверждение удаления",
MessageBoxButton.YesNo, MessageBoxImage.Question);
        if (result == MessageBoxResult.Yes)
        {
            marks.DeleteMark(selectedPoint);
            Close();
        }
    }
}
```

Разработанный интерфейс обеспечивает удобное взаимодействие с метками на карте и файлами, прикрепленными к этим меткам. Пользователь может легко добавлять, удалять и просматривать метки и связанные с ними файлы.

3.3. Работа с базой данных

Работа с базой данных является важной частью данного приложения, так как хранение информации занимает ключевую роль в этой работе. Для хранения информации в данной работе был использован PostgreSQL. Также для взаимодействия с базой данных из приложения на основе WPF была использована библиотека .NET Npgsql. Именно благодаря данной библиотеке появляется возможность подключения к базе данных, отправления sql-запросов из приложения, а также их обработка.

База данных приложения имеет всего две таблицы.

Таблица Points

Данная таблица отвечает за хранение информации о созданных ранее метках. Таблица имеет следующие столбцы:

1) point_id, тип данных int, хранит индивидуальные номера каждой метки;
2) latitude, тип данных double, хранит координату долготы;
3) longitude, тип данных double, хранит координату широты;
4) address, тип данных string, хранит название созданной метки.

Таблица Files

Данная таблица отвечает за хранение информации о файлах, которые прикрепляются к меткам в ходе работы. Таблица имеет следующие столбцы:

1) file_id, тип данных int, хранит индивидуальные номера каждого файла;
2) point_id, тип данных int, хранит индивидуальный номер метки, к которой прикреплен данный файл;
3) filename, тип данных string, хранит название файла или полный путь к нему;
4) file_data, тип данных бинарный, хранит данные загруженного файла в бинарном виде.

Данная таблица связана с таблицей Points с помощью столбца point_id, поэтому у таблицы Files включено каскадное удаление. Это облегчит работу с данными, а также упростит удаление файлов одновременно с меткой, к которой они привязаны.

Одним из ключевых аспектов работы с базой данных являются операции создания (Create), чтения (Read), обновления (Update) и удаления (Delete) данных, то есть CRUD-операции. Для реализации этого аспекта были созданы два класса Marks и Files. Каждый класс отвечает за взаимодействие с соответствующей ему таблицей.

Класс Marks

Данный класс отвечает за взаимодействие с таблицей Points, а также CRUD-операции для этой таблицы. Данный класс содержит несколько методов.

`CreateMark` создает новую метку в базе данных с заданными параметрами. Метод получает на вход координаты широты и долготы, а также название метки, введенное пользователем. Как видно по листингу 6, в начале идет подключение к базе данных, далее при помощи функционала библиотеки Npgsql базе данных передается простой sql-запрос, который добавляет в таблицу новую метку с заданными параметрами.

Данный метод используется при нажатии левой кнопкой мыши по карте на главном экране приложения.

Листинг 6 – Метод для создания новой метке в базе данных

```
public void CreateMark(double latitude, double longitude, string address)
{
    using (var connection = OpenConnection())
    {
        using (var cmd = new NpgsqlCommand("INSERT INTO Points (latitude,
longitude, address) VALUES (@latitude, @longitude, @address)", connection))
        {
            cmd.Parameters.AddWithValue("latitude", latitude);
            cmd.Parameters.AddWithValue("longitude", longitude);
            cmd.Parameters.AddWithValue("address", address);
            cmd.ExecuteNonQuery();
        }
    }
}
```

`GetMarkById` возвращает информацию (значения долготы, широты и название метки) о метке по ее индивидуальному номеру. На вход получает id метки, и возвращает соответствующие ей значения (листинг 7). Также использует sql-запрос для взаимодействия с базой данных.

Данный метод используется, когда на главном экране пользователь открывает подробную информацию о метке, метод подгружает id и название.

Листинг 7 – Метод для получения данных по id метки

```
public (int pointId, double latitude, double longitude, string address) GetMarkById(int pointId)
{
    using (var connection = OpenConnection())
    {
        using (var cmd = new NpgsqlCommand("SELECT point_id, latitude, longitude, address FROM Points WHERE point_id = @point_id", connection))
        {
            cmd.Parameters.AddWithValue("point_id", pointId);
            using (var reader = cmd.ExecuteReader())
            {
                if (reader.Read())
                {
                    return (
                        reader.GetInt32(0),
                        reader.GetDouble(1),
                        reader.GetDouble(2),
                        reader.GetString(3)
                    );
                }
                else
                {
                    throw new Exception("Mark not found");
                }
            }
        }
    }
}
```

`DeleteMark` удаляет метку и все привязанные к ней файлы. На вход получает только id удаляемой метки (листинг 8). Использует sql-запрос удаления метки, так как файлы связаны с меткой по id они также каскадно удалятся.

Данный метод используется, когда пользователь нажимает на кнопку «Удалить метку» в окне подробной информации.

Листинг 8 – Метод для удаления метки и ее файлов

```
public void DeleteMark(int pointId)
{
    using (var connection = OpenConnection())
    {
        using (var cmd = new NpgsqlCommand("DELETE FROM Points WHERE point_id = @point_id", connection))
        {
            cmd.Parameters.AddWithValue("point_id", pointId);
            cmd.ExecuteNonQuery();
        }
    }
}
```

Метод `UpdateMark` реализован в коде программы, однако нигде не используется. Метод получает на вход все данные точки и меняет их в таблице при помощи sql-запроса.

Также в классе реализован метод `GetAllMarks`, который возвращает список всех меток и их параметров. Реализован идентично методу `GetMarkById`.

Класс Files

Данный класс отвечает за взаимодействие с таблицей Files, а также CRUD-операции для этой таблицы.

`CreateFile` создает новый файл в базе данных со значениями, которые были переданы на вход. Метод реализован при помощи библиотеки Npgsql, sql-запросы также сделаны благодаря функционалу данной библиотеки (листинг 9).

Данный метод вызывается, когда создается новая точка, при левом нажатии на карту на главной странице приложения. Также метод вызывается, когда добавляются новые файлы в окне информации о метке.

Листинг 9 – Метод для создания файла

```
public void CreateFile(int pointId, string fileName, byte[] fileData)
{
    using (var connection = OpenConnection())
    {
        using (var cmd = new NpgsqlCommand("INSERT INTO Files (point_id, 
filename, file_data) VALUES (@point_id, @filename, @file_data)", connection))
        {
            cmd.Parameters.AddWithValue("point_id", pointId);
            cmd.Parameters.AddWithValue("filename", fileName);
            cmd.Parameters.AddWithValue("file_data", fileData);
            cmd.ExecuteNonQuery();
        }
    }
}
```

`GetFileById` возвращает информацию (название и бинарный вид данных файла) о файле по его индивидуальному номеру. На вход получает id файла, и возвращает соответствующие ей значения (листинг 10). Так же, как и ранее использует sql-запрос для взаимодействия с базой данных.

Данный метод используется при заполнении таблицы файлов в окне информации о метке.

Листинг 10 – Метод для получения значений параметров таблицы Files

```
public (int fileId, int pointId, string fileName, byte[] fileData) GetFile-
ById(int fileId)
 {
     using (var connection = OpenConnection())
     {
         using (var cmd = new NpgsqlCommand("SELECT file_id, point_id, file-
name, file_data FROM Files WHERE file_id = @file_id", connection))
         {
             cmd.Parameters.AddWithValue("file_id", fileId);
             using (var reader = cmd.ExecuteReader())
             {
                 if (reader.Read())
                 {
                     return (
                         reader.GetInt32(0),
                         reader.GetInt32(1),
                         reader.GetString(2),
                         (byte[])reader["file_data"]
                     );
                 }
                 else
                 {
                     throw new Exception("File not found");
                 }
              }
          }
     }
}
```

`DeleteFile` реализован аналогично одноименному методу из класса Marks. Метод используется при выборе опции «Удалить файл» в окне информации о метке.

3.4. Дополнительные классы

Для более удобного подключения к базе данных были также разработаны два дополнительных класса. Эти классы позволяют настроить параметры подключения так, как это необходимо пользователю.

Конфигурация приложения осуществляется с помощью класса Configuration, который отвечает за чтение и сохранение параметров подключения к базе данных PostgreSQL. При создании объекта этого класса происходит попытка прочитать конфигурационный файл в формате JSON. Если

файл не найден или его формат неверен, создается конфигурация с параметрами по умолчанию. Эти параметры включают в себя адрес сервера, порт, имя пользователя, пароль и имя базы данных (листинг 11). Класс также предоставляет метод для сохранения текущей конфигурации в файл и метод для получения строки подключения, необходимой для подключения к базе данных (листинг 13).

Листинг 11 – Поля класса Configuration

```
public string Host { get; set; }
public string Port { get; set; }
public string User { get; set; }
public string Pass { get; set; }
public string Base { get; set; }
```

Json файл содержит значения для подключения к базе данных. Эти значения может ввести пользователь или же если файл некорректный или пустой, то вводятся данные по умолчанию (листинг 12).

Листинг 12 – Данные по умолчанию для конфигуратора

```
public void Default()
{
    Host = "127.0.0.1";
    Port = "5432";
    User = "postgres";
    Pass = "13500420";
    Base = "MarksAndFiles";
}
```

Листинг 13 – Метод для сохранения конфигурации

```
public void Save()
{
    try
    {
        string jsonObject = JsonConvert.SerializeObject(this);
        byte[] bytes = Encoding.Default.GetBytes(jsonObject);
        FileStream file = File.OpenWrite("./app.Json");
        file.Write(Encoding.Default.GetBytes(jsonObject),0, bytes.Length);
        file.Close();
    }
    catch(Exception ex) {
        MessageBox.Show(ex.Message, "Внимание!", MessageBoxButton.OK, Mes-
sageBoxImage.Error);
    }
}
```

Класс Database управляет подключением к базе данных. Он использует объект Configuration для получения строки подключения и обеспечивает методы для открытия и закрытия соединения. При вызове метода Connect устанавливается новое подключение с использованием переданной конфигурации. Метод Reconnect позволяет обновить или повторно установить подключение, закрыв старое, если оно активно. Все взаимодействия с базой данных происходят через статическое свойство Connection, которое предоставляет текущее подключение. Из листинга 14 видно, что в начале задаются значения переменной configuration, далее проверяется что конфигурация существует и в конце при помощи функционала библиотеки Npgsql открывается подключение к базе данных с заданными параметрами.

Листинг 14 – Код для подключения к базе данных

```
public static void Reconnect(Configuration config = null) {
    if(config != null)
    {
        configuration = config;
    }

    if(configuration == null)
    {
        throw new Exception("Объект не был инициализирован");
    }

    if(Connection != null) {
        if (Connection.State != System.Data.ConnectionState.Closed)
        {
            Connection.Close();
        }
    }

    Connection = new NpgsqlConnection(configuration.GetConnectionString());
    Connection.Open();
}
```

Вывод по третьей главе

В данной главе был приведен список всех использованных программных средств, рассмотрены все классы разработанного приложения, а также их методы. Также был продемонстрирован интерфейс приложения и то, как окна взаимодействуют друг с другом и с базой данных. Также было продемонстрировано, как приложение подключается к базе данных.

4. ТЕСТИРОВАНИЕ СИСТЕМЫ

Для проверки работоспособности приложения было проведено функциональное тестирование. Функциональное тестирование – это вид тестирования, проверяющий функциональные возможности программы, те функции, которые были реализованы согласно постановке задачи [10].

В ходе данного тестирования проверялось соответствие приложения предъявленным функциональным требованиям. В таблице 1 представлены результаты тестирования.

Таблица 1 – Таблица протокола тестирования

№	Название теста	Действия	Результат	Тест пройден?
1	Подключение к базе данных	Задать параметры подключения в app.json и запустить приложение	Приложение подключено к базе данных	Да
2	Добавление метки на карту	Нажать ЛКМ по любому месту на карте.	На карте появилась красная точка запись с координатами.	Да
3	Добавление метки с названием	Нажать ЛКМ по любому месту на карте и в новом окне ввести название	На карте появилась точка, в базе данных появилась запись с названием и координатами.	Да
4	Добавление метки с названием и файлом	Нажать ЛКМ по любому месту на карте и в новом окне ввести название и выбрать файл для загрузки	На карте появилась точка, в базе данных появилась запись с названием, координатами и данными файла.	Да
5	Просмотр карты	Зажатой ПКМ передвигать карту в разные места.	Карта свободно двигается.	Да
6	Приближение/отдаление карты	Колесиком мыши приблизить и отдалить карту.	Масштаб карты меняется соответственно движению колесика мыши.	Да
7	Обновление списка меток справа от карты	Нажать на кнопку «обновить».	База данных обновляется, метки на карте обновляются.	Да

Окончание таблицы 1

№	Название теста	Действия	Результат	Тест пройден?
8	Открыть окно информации о метке	Сделать двойное нажатие кнопки мыши по любому элементу из списка меток.	Открывается окно информации о метке.	Да
9	Удалить файл из таблицы в окне информации о метке	Нажать ПКМ по любому элементу таблицы и выбрать пункт «Удалить файл», нажать да.	Таблица обновлена, файл удалён из базы данных.	Да
11	Скачать файл в окне информации о метке	Нажать ПКМ по любому элементу таблицы и выбрать пункт «Скачать файл», выбрать путь	Файл скачан по установленному пути	Да
12	Удалить метку в окне информации о метке	Нажать на кнопку «Удалить метку», нажать да.	Окно закрывается, метка удаляется из базы данных.	Да
13	Попытаться удалить метку в окне информации о метке, но в конце отменить решение	Нажать на кнопку «Удалить метку», нажать нет.	Окно остается открытым, метка не удаляется.	Да
14	Добавить файл в окне информации о метке	Нажать на кнопку «Загрузить новый файл», выбрать файл для загрузки.	Список файлов обновлен, в нем появился новый только что добавленный файл.	Да

Вывод по четвертой главе

В четвертой главе было проведено функциональное тестирование приложения по сформированному набору тестов. Все тесты из набора были выполнены успешно. Можно сделать вывод, что разработанное приложение соответствует требованиям.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было реализовано десктопное приложение для визуализации данных с использованием OpenStreetMap.

Основные результаты:

1) выполнен анализ предметной области;
2) спроектировано приложение;
3) приложение было реализовано;
4) проведено тестирование приложения.

В ходе работы над проектом были достигнуты поставленные цели: создано удобное и интуитивно понятное приложение, позволяющее пользователям эффективно работать с географическими данными и сопутствующими файлами. Разработанный интерфейс обеспечивает легкость навигации и взаимодействия с картой, а интеграция с базой данных гарантирует надежное хранение и доступ к данным. В ходе выполнения выпускной квалификационной работы была изучена платформа WPF для создания современного и удобного пользовательского интерфейса. Также были освоены две ключевые библиотеки GMap.NET, которая позволила интегрировать карты в приложение и обеспечить функционал для добавления и управления метками, и Npgsql, который позволил реализовать надежное и эффективное управление данными, хранящимися в базе данных.

ЛИТЕРАТУРА

1. Разработка на языке C# Visual Studio. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/?view=vs-2022 (дата обращения: 30.01.2024 г.).

2. Все о компании 2ГИС. [Электронный ресурс] URL: http://info.2gis.ru/novosibirsk/ (дата обращения: 12.02.2024 г.).

3. Bucanek J. Model-view-controller pattern. // Learn Objective-C for Java Developers., 2009. – С. 353–402.

4. Mooney P., Minghini M. A review of OpenStreetMap data. // Mapping and the citizen sensor., 2017. – С. 37–59.

5. OpenStreetMap [Электронный ресурс] URL: https://www.openstreetmap.org (дата обращения: 06.03.2024 г.).

6. GMap.NET. Страница разработчика. [Электронный ресурс] URL: https://github.com/radioman/greatmaps (дата обращения: 10.03.2024 г.).

7. Документация PostgreSQL и Postgres Pro. [Электронный ресурс] URL: https://postgrespro.ru/docs (дата обращения: 25.03.2024 г.).

8. Хафизов Д.Д., Лебедева А.В., Латкина К.А. Дизайн интерфейсов. // Научный Лидер., 2023. – С. 11.

9. Sells C., Griffiths I. Programming WPF: Building Windows UI with Windows Presentation Foundation. – O'Reilly Media, Inc., 2007.

10. Schwichtenberg H., Schwichtenberg H. Installing Entity Framework Core // Modern Data Access with Entity Framework Core: Database Programming Techniques for .NET,. NET Core, UWP, and Xamarin with C#., 2018. – С. 15–29.

11. Documentation|Npgsql documentation. [Электронный ресурс] URL: https://www.npgsql.org/doc/ (дата обращения: 03.04.2024 г.).

12. Gansner E.R., Hu Y., Kobourov S. GMap: Visualizing graphs and clusters as maps // 2010 IEEE Pacific Visualization Symposium (PacificVis). – IEEE, 2010. – С. 201–208.

13. Drake J.D., Worsley J. C. Practical PostgreSQL. – O'Reilly Media, Inc., 2002. – 107–148.

14. Дунаев В.В. Базы данных. Язык SQL для студента, 2 изд. – БХВ-Петербург, 2012. – 73–90.

15. Старолетов С.М. Основы тестирования и верификации программного обеспечения: учебное пособие / С.М. Старолетов. – 2-е изд.,стер. – Санкт–Петербург: Лань, 2020. – 344 с.