**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**федеральное государственное автономное образовательное   
учреждение высшего образования   
«Санкт-Петербургский политехнический университет Петра Великого»**

**ИНСТИТУТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

|  |  |
| --- | --- |
|  | **Допустить к защите**  Заместитель директора по  учебно-методической работе  \_\_\_\_\_\_\_\_\_\_\_ Е. Г. Конакина .  (Подпись) (И.О.Ф.)  « » 2023 г. |
|  |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Тема Разработка программных модулей для оценки физико-химических процессов

специальность 09.02.07 группа 42919/3

Студент (ка) Горбачев Т. А.

(подпись) (ФИО)

Руководитель Молькова Л. Ю.

(подпись) (ФИО)

Санкт-Петербург

2023

СОДЕРЖАНИЕ

[Введение 3](#_Toc136694485)

[1 Общая часть 4](#_Toc136694486)

[1.1 Постановка задачи 4](#_Toc136694487)

[1.2 Сбор требований 6](#_Toc136694490)

[1.3 Обзор и выбор средств разработки 8](#_Toc136694494)

[2 Специальная часть 14](#_Toc136694501)

[2.1 Проектирование программного обеспечения 14](#_Toc136694502)

[2.2 Разработка программного обеспечения 19](#_Toc136694507)

[3 Экономическая часть 35](#_Toc136694512)

[3.1 Область применения программного продукта и его преимущества перед аналогичным программным продуктом 35](#_Toc136694513)

[3.2 Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад 35](#_Toc136694514)

[3.3 Расчет затрат на разработку 37](#_Toc136694515)

[3.4 Расчет цены и прибыли 41](#_Toc136694521)

[4 Техника безопасности и охрана труда 44](#_Toc136694522)

[4.1 Анализ условий труда разработчика–программиста, работающего в ООО «Оптосенс» 44](#_Toc136694523)

[4.2 Расчёт искусственного освещения в помещении 46](#_Toc136694524)

[4.3 Электробезопасность на предприятии 48](#_Toc136694525)

[4.4 Пожарная безопасность на предприятии 50](#_Toc136694526)

[Заключение 53](#_Toc136694527)

[Список использованных источников 54](#_Toc136694528)

[Приложение А (обязательное) Список используемых сокращений 56](#_Toc136694529)

[Приложение Б (обязательное) Листинг исходного кода приложения 57](#_Toc136694530)

[Приложение В (обязательное) Листинг скрипта базы данных 70](#_Toc136694531)

# ВВЕДЕНИЕ

Современную науку сложно представить без широкого применения моделирования. Суть этого метода состоит в замене исходного объекта его «образом», математической моделью, и последующем ее изучении с помощью реализуемых на компьютерах вычислительно-логических алгоритмов. Работа не с самим объектом (явлением, процессом), а с его моделью дает возможность относительно легко и быстро, а также без существенных затрат исследовать его свойства и поведение в любых мыслимых ситуациях. С другой стороны, вычислительные (компьютерные, симуляционные, имитационные) эксперименты с моделями объектов позволяют, опираясь на мощь современных вычислительных методов и технических инструментов информатики, подробно и глубоко изучать объекты в достаточной полноте, недоступной чисто теоретическим подходам. В результате этого методология математического моделирования бурно развивается, охватывая все новые сферы – от разработки технических систем и управления ими до анализа сложнейших экономических и социальных процессов.

Наиболее впечатляющие успехи достигнуты при применении математического моделирования в инженерии и технологии. Так целью данной работы – создание программных модулей, с помощью которых происходит мультифрактальное моделирование структур полупроводниковых материалов, выявляются и анализируются зависимости между их физическими свойствами и структурой поверхности.

Основные задачи проекта:

* создание алгоритма процесса мультифрактального анализа неоднородных структур для использования в оценке качества полупроводниковых материалов по их изображениям, полученным в оптическом микроскопе;
* проектирование и разработка базы данных для хранения информации о проанализированных структурах;
* оценка экономических показателей проекта;
* оценка безопасности и охраны труда разработчиков приложения.

# Общая часть

## Постановка задачи

### Описание предметной области

Представления о фракталах и мультифракталах широко используются для описания свойств самоподобия, наблюдаемых в самых разных физических ситуациях. К числу фракталов относят геометрические объекты, которые имеют сильно изрезанную форму и демонстрируют некоторую повторяемость в широком диапазоне масштабов. Повторяемость может быть полной (в этом случае говорят о регулярных фракталах), либо может наблюдаться некоторый элемент случайности (такие фракталы называют случайными или стохастическими). Структура случайных фракталов на малых масштабах не является идентичной всему объекту, но их статистические характеристики совпадают, и свойства самоподобия справедливы после усреднения по статистически независимым реализациям объекта.

Сложные фракталы, называемые мультифракталами, важны прежде всего потому, что именно они, как правило, и встречаются в природе, тогда как простые самоподобные объекты являются идеализацией реальных явлений. Фактически, мультифрактальный подход означает, что изучаемый объект каким-то образом можно разделить на части, для каждой из которых наблюдаются свои свойства самоподобия.

Мультифракталы – неоднородные фрактальные объекты, для полного описания которых, в отличие от регулярных фракталов, недостаточно введения всего лишь одной величины его фрактальной размерности, а необходим целый спектр таких размерностей, число которых, вообще говоря, бесконечно. Причина этого заключается в том, что наряду с чисто геометрическими характеристиками, такие фракталы обладают и некоторыми статистическими свойствами.

Мультифрактальный анализ изображения позволяет находить спектр размероностей и в дальнейшем связать его со свойствами исследуемой структуры на изображении. Это позволяет исследовать структуры относительно простыми и быстрыми методами с помощью ЭВМ, не прибегая к использованию сложного научного лабораторного оборудования. Таким образом сокращается время исследований и контроля получаемых структур. Этим обуславливается актуальность приложения для компании ООО «Оптосенс», занимающейся производством газовых оптических датчиков на основе полупроводниковых структур.

Таким образом предметная область состоит из следующих сущностей:

* пользователь (оператор);
* технологический процесс (контроль качества полупроводниковой структуры);
* анализ и статистика (учет и соотнесение мультифрактальных и физико-химических характеристик полупроводниковых структур).

Это позволяет определить минимальные функциональные требования к приложению:

* возможность пользователем проводить мультифрактальный анализ изображений;
* возможность пользователем сохранять данные об анализе и вести статистику.

### Анализ существующих решений

В настоящее время отсутствуют широко распространённые программы для мультифрактального анализа. Существующие программы не являются общедоступными, так как обычно создаются на базе университетов для собственных научно-технических нужд. Также подобное программное обеспечение поставляется вместе с микроскопами, но с функционалом недостаточным для полноценного мультифрактального анализа.

Результат обработки изображения программой «Gwyddion», используемой вместе с микроскопами представлен на рисунке 1.

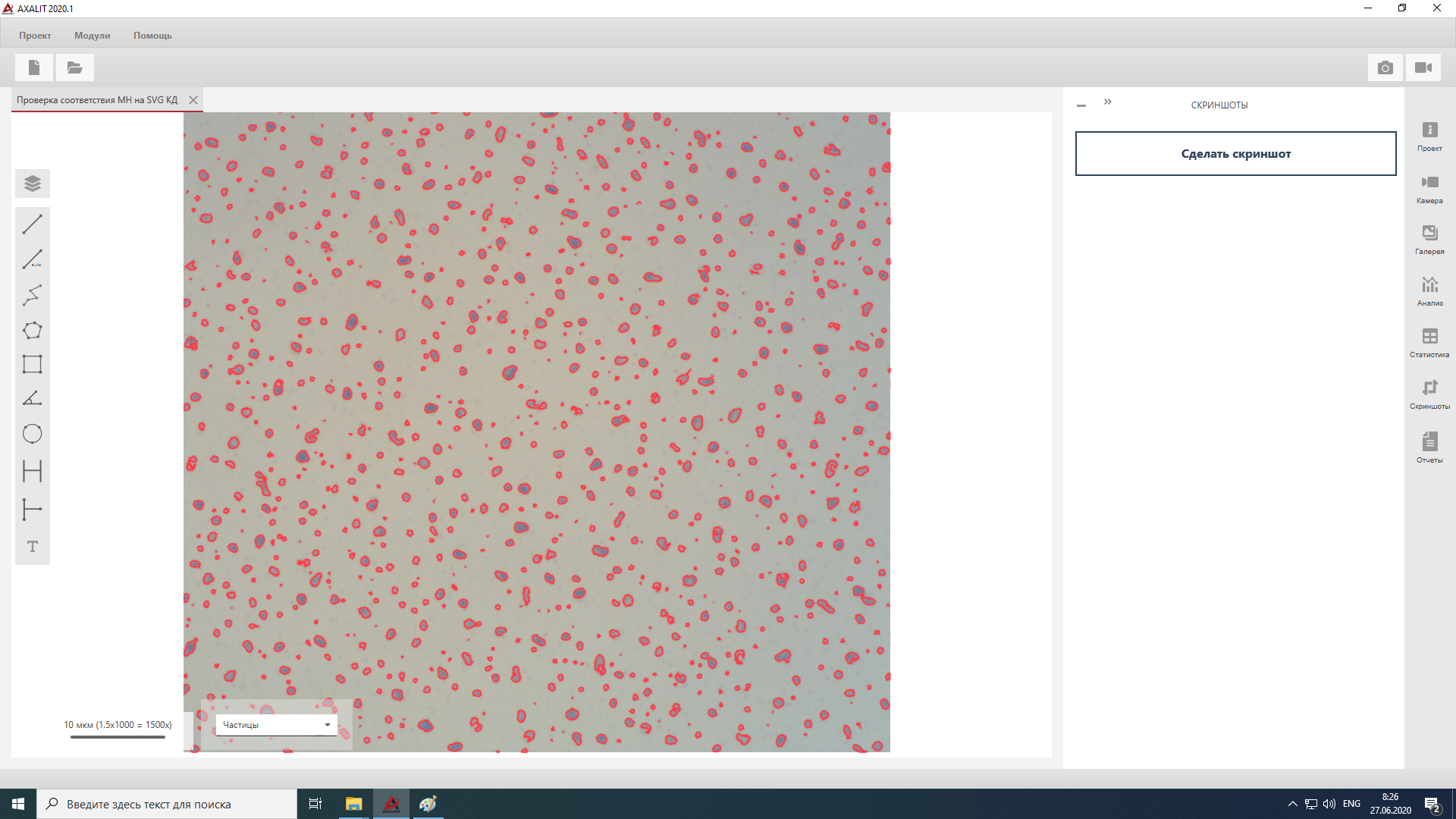


Рисунок 1 – обработанное программой «Gwyddion» изображение неоднородной структуры

## Сбор требований

Чтобы реализовать проект, необходимо корректно сформулировать требования к системе.

На этапе сбора требований основная работа ведется с заказчиком системы и её будущими пользователями.

Цель этапа – точно определить функции продукта и способы его интеграции в существующие процессы.

Сбор требований проводился методом интервьюирования, который заключается в беседе между разработчиком системы и заказчиком.

Вместе с сбором требований был проведен анализ списка заинтересованных лиц и пользователей системы. Было установлено, что пользователями системы будут являться операторы на предприятии, интересантами является руководство организации.

### Пользовательские требования к приложению

Требования, сформированные на основе интервьюирования инженера-исследователя ООО «Оптосенс»:

* В приложении должна быть реализована система регулируемых сохраняемых параметров обработки изображений;
* В приложении должно быть реализовано отображение изображений до и после обработки, а также графиков и прилагаемых к графикам данных необходимых для анализа;
* В приложении должен быть доступен просмотр сохраненных результатов в БД, сохранение данных в БД и в отдельные табличные отчеты Excel.

### Функциональные требования к приложению

Разрабатываемое приложение должно иметь следующие функции:

* Возможность производить мультифрактальный анализ изображений;
* Возможность одновременного выбора нескольких изображений для анализа;
* Возможность регулировки параметров мультифрактального анализа изображений;
* Возможность сохранения и загрузки параметров для обработки изображений;
* Возможность подключения к серверу с БД;
* Возможность сохранения и просмотра данных анализа в БД;
* Возможность сохранения результатов анализа в Excel-файле.

### Визуализация требований

Визуализация требований проводится построением UML диаграмм.

Диаграмма вариантов использования изображена на рисунке 2.

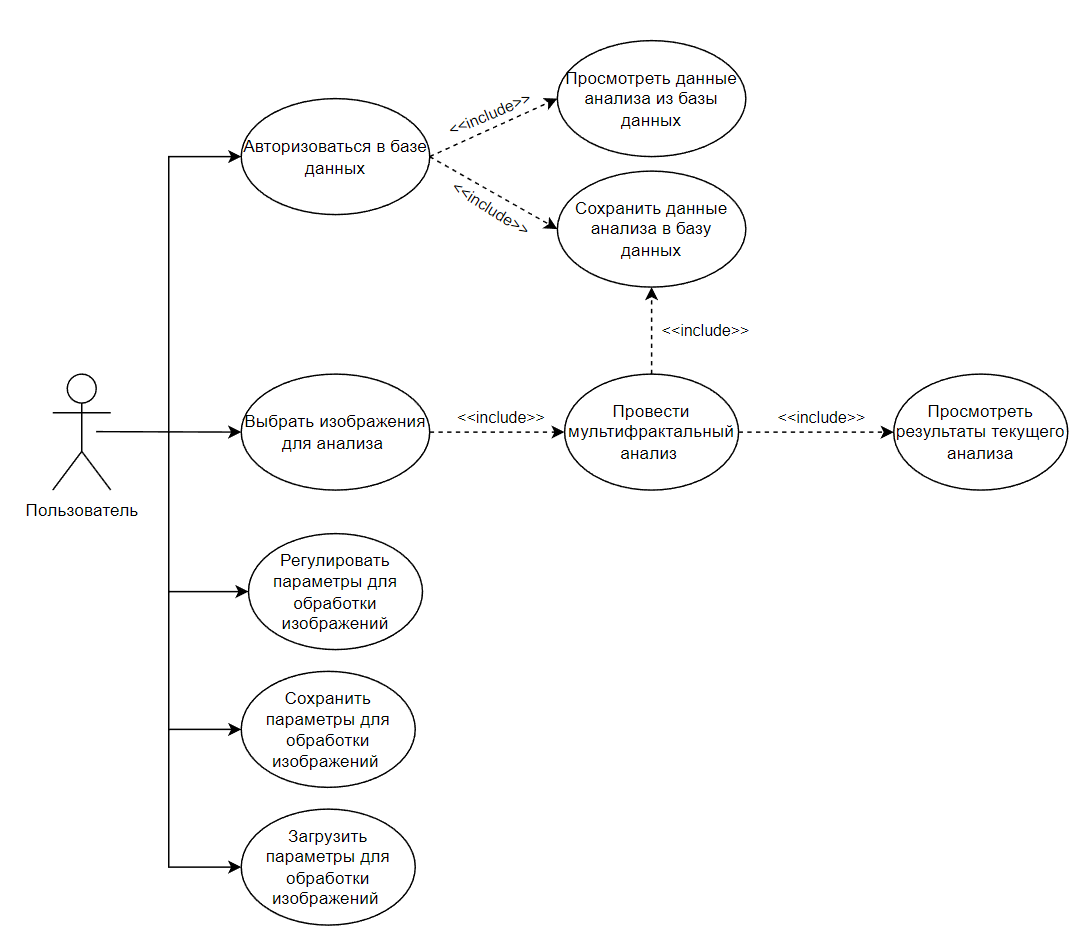


Рисунок 2 – Диаграмма вариантов использования

## Обзор и выбор средств разработки

### Обзор распространенных языков программирования

Разработка десктопных приложений предусматривает использование объектно-ориентированных языков. Среди них наиболее популярными являются Java, Python и C#.

#### Python

Python – простой в освоении язык программирования с активным сообществом разработчиков. Он обладает широким набором библиотек и фреймворков, что делает его подходящим для быстрой разработки. Однако, Python может быть менее эффективным для десктопных приложений, особенно в случае требовательных по производительности проектов.

#### Java

Java – популярный язык программирования, известный своей платформенной независимостью. Он обеспечивает широкие возможности разработки десктопных приложений с использованием богатых библиотек и инструментов. Однако, разработка на Java может быть более громоздкой и требовать больше усилий в сравнении с другими языками.

#### С#

C# – язык программирования, разработанный компанией Microsoft, позволяющий создавать десктопные приложения с использованием WPF. C# обеспечивает удобный синтаксис, широкий набор инструментов и поддержку различных платформных функций. Он особенно хорошо подходит для разработки десктопных приложений под Windows, обеспечивая высокую производительность и интеграцию с другими технологиями Microsoft.

### Выбор языка программирования

После обзора различных языков программирования был избран C#. Он предоставляет удобный и мощный инструментарий для разработки десктопных приложений, особенно с использованием WPF. C# обладает поддержкой платформы Windows и интеграцией с другими технологиями Microsoft, что облегчает создание высокопроизводительных и функциональных приложений для Windows. Кроме того, наличие обширного сообщества разработчиков и ресурсов помогает при освоении и решении возникающих вопросов в процессе разработки [1, 2].

### Обзор распространенных систем управления базами данных

Разработка и проектирование базы данных предусматривает выбор системы управления базами данных.

#### Microsoft Access

СУБД Microsoft Access – простая и удобная среда разработки баз данных. Из недостатков можно отметить скромные технические характеристики, медленную работу с базами данных, отсутствие возможности формирование исполнимого файла. В целом, СУБД Microsoft Access удобна для обучения основам баз данных, но слабо применима в реальных проектах.

#### Microsoft SQL Server

СУБД Microsoft SQL Server – полнофункциональная серверная СУБД, использующая язык запросов Transact-SQL. Оснащена удобными средствами администрирования, архивирования и восстановления баз данных и эффективными средствами защиты. Система обеспечивает надежное и компактное хранение, а также высокопроизводительную обработку баз данных больших объёмов. Microsoft SQL Server поддерживает тиражирование данных, параллельную обработку, отличается простотой управления и использования. Имеются и некоторые технические недостатки, такие как отсутствие кроссплатформенности, ограничения на использование ЦП, ОЗУ, а также ограничения на размер базы данных.

#### MySQL

MySQL имеет схожие характеристики и особенности с Microsoft SQL Server, но по сравнению с ней поддерживает кроссплатформенность, обладает лучшей масштабируемостью и большей производительностью за счет эффективного использования многопоточности и параллельной обработки запросов. Кроме этого, MySQL – свободно распространяемая СУБД с открытым исходным кодом, в отличии от остальных представленных, и имеет большое и активное сообщество.

### Выбор систем управления базами данных

В результате обзора СУБД была избрана MySQL, поскольку она обладает лучшей производительностью, необходимой для работы над проектом. Кроме этого, популярность данного СУБД больше, чем у его аналогов, благодаря этому имеется большое количество учебного и практического материала, помогающего при разработке [3].

### Обзор сред разработки программного обеспечения

Разработка ПО предусматривает выбор среды разработки под конкретные архитектуру, вид приложения и язык программирования.

Для разработки WPF-приложений существуют несколько популярных сред разработки – SharpDevelop, Microsoft Visual Studio Code и Microsoft Visual Studio.

#### SharpDevelop

SharpDevelop – это бесплатная и открытая интегрированная среда разработки для языка программирования C#. Она обладает широким набором функций, включая редактор с подсветкой синтаксиса, отладчик и графический дизайнер форм. SharpDevelop поддерживает разработку консольных приложений, приложений Windows Forms, WPF и ASP.NET, а также имеет активное сообщество разработчиков. Она является бесплатной альтернативой коммерческим IDE, предоставляя разработчикам удобную среду для программирования на C#.

#### Microsoft Visual Studio

Microsoft Visual Studio – это интегрированная среда разработки, разработанная компанией Microsoft, которая предоставляет широкий набор инструментов и функциональности для разработки различных типов приложений, включая приложения для Windows, веб-приложения, мобильные приложения и многое другое. Visual Studio поддерживает несколько языков программирования, включая C#, C++, Visual Basic, JavaScript и другие, и предлагает мощный редактор кода с функциями автодополнения, отладки, профилирования и управления версиями. Она также включает в себя широкий набор инструментов для проектирования пользовательского интерфейса, тестирования приложений и развертывания. Visual Studio обладает обширным экосистемой плагинов и расширений, что позволяет разработчикам настроить среду разработки под свои потребности. Благодаря своей мощности, функциональности и широкому использованию в индустрии, Microsoft Visual Studio является одним из наиболее популярных инструментов разработки программного обеспечения.

#### Microsoft Visual Studio Code

Это бесплатная и расширяемая среда разработки (IDE) с открытым исходным кодом, разработанная Microsoft. Она предоставляет разработчикам широкий спектр функций и возможностей для создания и редактирования кода в различных языках программирования. VS Code обладает интуитивным интерфейсом, поддерживает автодополнение кода, отладку, контроль версий и интеграцию с популярными инструментами и расширениями. Она мощно интегрируется с платформой Microsoft, но также поддерживает разработку для различных операционных систем, включая Windows, macOS и Linux. Благодаря своей гибкости, простоте использования и расширяемости, Microsoft Visual Studio Code стал одним из популярных выборов для многих разработчиков.

### Выбор среды разработки программного обеспечения

После обзора сред разработки программного обеспечения была выбрана среда Microsoft Visual Studio, так как предоставляет разработчикам автоматизацию установки необходимых пакетов и расширений, удобный интерфейс, а также большое количество инструментов для разработки WPF-приложений.

# Специальная часть

## Проектирование программного обеспечения

### Архитектурное проектирование

При разработке данного программного обеспечения необходимо реализовать трехуровневую клиент-серверную архитектуру. В контексте компьютерных технологий клиент-серверная архитектура предполагает наличие нескольких компонентов приложения. Клиентский компонент подключается к серверу, который, в свою очередь, может быть связан с сервером базы данных. В роли сервера может выступать система управления базами данных. Пример клиент-серверной архитектуры показан на рисунке 3.



Рисунок 3 – Визуализация клиент-серверной архитектуры

Клиент – это приложение для конечного пользователя, обычно с графическим интерфейсом.

Сервер приложений – это связующее звено между сервером БД и клиентом, переадресовывающее запрос клиента SQL-Серверу БД и возвращающее ответ обратно клиенту.

Сервер – это обычно стандартная реляционная или объектно-ориентированная СУБД, обеспечивающая хранение и управление данными.

К достоинствам данной архитектуры относятся:

* высокая безопасность;
* высокая надежность;
* низкие требования к производительности и техническим характеристикам терминалов, как следствие снижение их стоимости;
* масштабируемость;
* конфигурируемость – изолированность уровней друг от друга позволяет безопасно и быстро простыми средствами переконфигурировать систему при возникновении сбоев или при плановом обслуживании на любом из уровней.

Данная архитектура обеспечивает большую масштабируемость, безопасность и конфигурируемость по сравнению с файл-серверной архитектурой и двухзвенной клиент-серверной архитектурой [4, 5].

Схема архитектуры разрабатываемого ПО отображена на рисунке 4.



Рисунок 4 – Схема архитектуры разрабатываемого ПО

### Проектирование базы данных

Функционирование клиент-серверных приложений предусматривает создание базы данных. Проектирование БД производилось при использовании СУБД MySQL и веб-приложение для администрирования СУБД phpMyAdmin. Данное веб-приложение позволяет администрировать СУБД непосредственно через веб-обозреватель на компьютере. На сегодняшний день данная система пользуется большим спросом у разработчиков, так как предоставляет широкий функционал для работы с базами данных. Все сущности базы данных приведены к третьей нормальной форме.

Для реализации базы данных были созданы следующие таблицы:

* ProcessingSettings – для хранения параметров;
* UneditedImage – для хранения необработанных изображений;
* EditedImage – для хранения обработанных изображений;
* ProcessingData – для данных о сессии обработки;

Словари данных и схема спроектированной базы данных предоставлены ниже в таблицах 1-4 и на рисунке 5.

Таблица 1 – Словарь данных таблицы ProcessingSettings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **NULL** | **Ключ** | **Описание** |
| ps\_id | int(11) | NOT | PK | Идентификатор параметров |
| ps\_thresholdValue | smallint(6) | NOT | - | Чувствительность преобразования в ЧБ |
| ps\_imageInversion | tinyint(1) | NOT | - | Инвертировано ли изображение |
| ps\_minSignificantArea | tinyint(4) | NOT | - | Фильтрация неоднородностей (в пикселях) |
| ps\_minRectSize | tinyint(4) | NOT | - | Минимальный размер ячейки для анализа |
| ps\_maxRectSize | tinyint(4) | NOT | - | Максимальный размер ячейки для анализа |
| ps\_variableParameter | tinyint(4) | NOT | - | Предельные значения варьируемого параметра |

Таблица 2 – Словарь данных таблицы UneditedImage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **NULL** | **Ключ** | **Описание** |
| ui\_id | int(11) | NOT | PK | Идентификатор необработанного изображения |
| ui\_image | longblob | NOT | - | Изображение, представленное массивом байтов |

Таблица 3 – Словарь данных таблицы EditedImage

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **NULL** | **Ключ** | **Описание** |
| ei\_id | int(11) | NOT | PK | Идентификатор обработанного изображения |
| ei\_image | longblob | NOT | - | Изображение, представленное массивом байтов |

Таблица 4 – Словарь данных таблицы ProcessingData

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Тип** | **NULL** | **Ключ** | **Описание** |
| pd\_id | int(11) | NOT | PK | Идентификатор сессии обработки |
| pd\_time | datetime | NOT | - | Время обработки |
| ps\_id | int(11) | NOT | FK | Внешний ключ, идентификатор параметров |
| ui\_id | int(11) | NOT | FK | Внешний ключ, идентификатор необработанного изображения |
| ei\_id | int(11) | NOT | FK | Внешний ключ, идентификатор обработанного изображения |
| pd\_arrays | mediumtext | NOT | - | XML-файл хранящий данные массивов полученных в процессе обработки изображений |

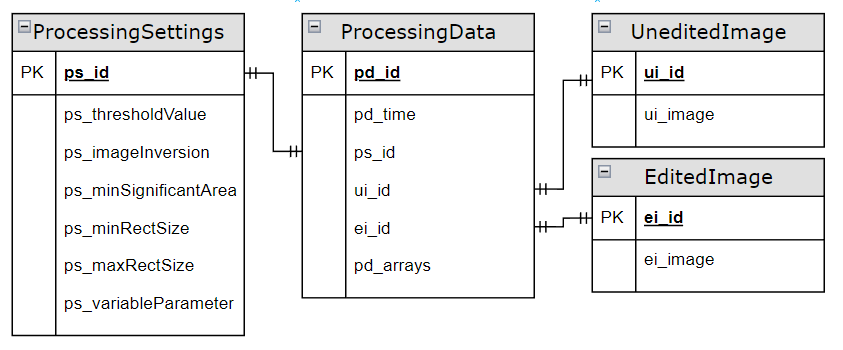


Рисунок 5 – ER-диаграмма спроектированной базы данных

### Проектирование API

API (англ. Application Programming Interface) – описание способов взаимодействия одном программы с другими. API разрабатывалась для упрощения дальнейшего масштабирования ПО и защиты базы данных, так как при использовании API пользователь не имеет прямого доступа к базе данных.

Так и в данной программе создается API для взаимодействия пользователя, использующего разрабатываемое десктопное приложение (клиент), с СУБД, посредством удаленного сервера.

### Проектирование пользовательского интерфейса приложения

Интерфейс пользователя (UI) – это часть программы, находящаяся на виду у пользователя и обеспечивающая отображение данных, управление ими и диалог с пользователем.

Графический пользовательский интерфейс (GUI) – это тип пользовательского интерфейса, который управляет данными, отображает их, а также ведет диалог с пользователем посредством использования графических элементов, такие как кнопки, меню, текстовые поля и другие. GUI обеспечивает удобный способ взаимодействия, позволяя пользователю выполнить операции с помощью мыши и клавиатуры. Также представляет пользователю интуитивный и наглядный способ управления приложением, отображая информацию и реагируя на пользовательские действия. GUI является одним из наиболее распространенных типов интерфейса пользователя и широко применяется в десктопных приложениях.

Пользовательский интерфейс представляет собой совокупность программных и аппаратных компонентов, которые обеспечивают взаимодействие между пользователем и компьютером. Основу этого взаимодействия составляют диалоги, которые представляют собой организованный обмен информацией между человеком и компьютером в реальном времени, направленный на решение конкретной задачи. Диалог включает в себя процессы ввода и вывода информации, которые физически обеспечивают связь между пользователем и компьютером.

На основе пользовательских и функциональных требований к приложению и был спроектирован простой и удобный интерфейс.

## Разработка программного обеспечения

Разработка приложения включает в себя создание алгоритма мультифрактального анализа неоднородных структур на языке C#, интерфейса, базы данных и другое. Для реализации проекта будет использована платформа .NET и система построения клиентский приложений WPF [6, 7].

### Разработка алгоритма мультифрактального анализа неоднородных структур

* + - 1. **Разработка алгоритма обработки изображений**

Для разработки алгоритма обработки изображений была использована спецификация библиотеки алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом OpenCV – OpenCvSharp4.

Разработанный алгоритм для выделения неоднородностей преобразовывает исходное изображение в градации серого, увеличивает контрастность изображения, применяет пороговый фильтр преобразования (чувствительность преобразования) в черно-белое. Затем на контрастном черно-белом варианте изображения производится выделение контуров, которые в последствии проходят фильтрацию, после которой остаются только внешние контуры площадь которых, не менее указанного параметра в пикселях.

Пример изображения, полученного в ходе выполнения данного алгоритма показан на рисунке 6.

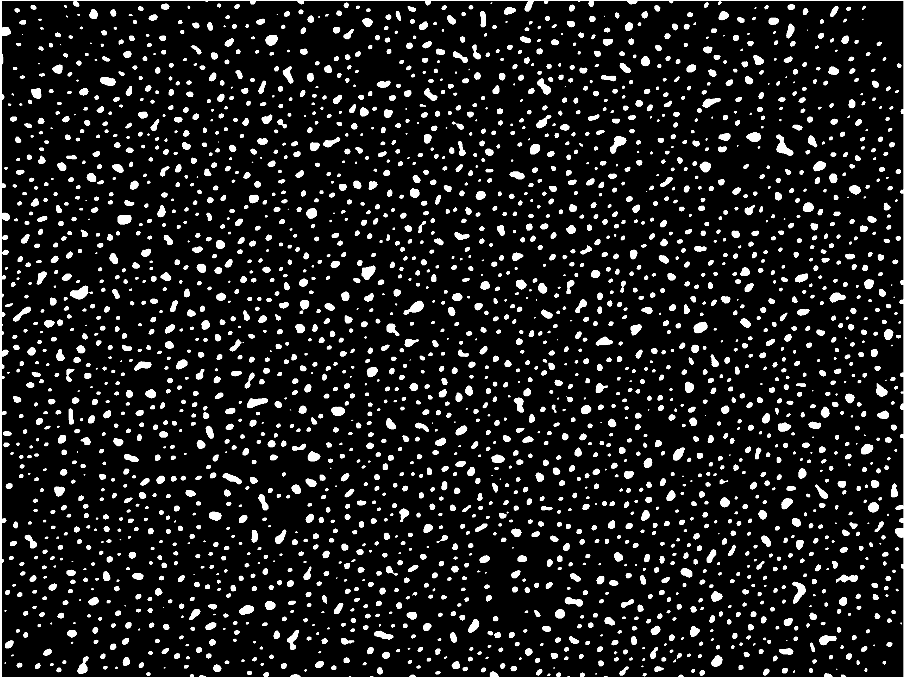


Рисунок 6 – Изображение, обработанное алгоритмом нахождения контуров

* + - 1. **Разработка алгоритма мультифрактального анализа неоднородностей**

После получения обработанного изображения оно разбивается на ячейки с регулируемым размером. Оптимальный размер ячейки выявляется практическим путем для каждого материала, при этом размер должен быть существенно меньше размера неоднородностей, но как можно больше.

Затем для каждой ячейки подсчитывается количество пикселей, принадлежащих неоднородностям. Визуализация этого процесса представлена на рисунке 7.

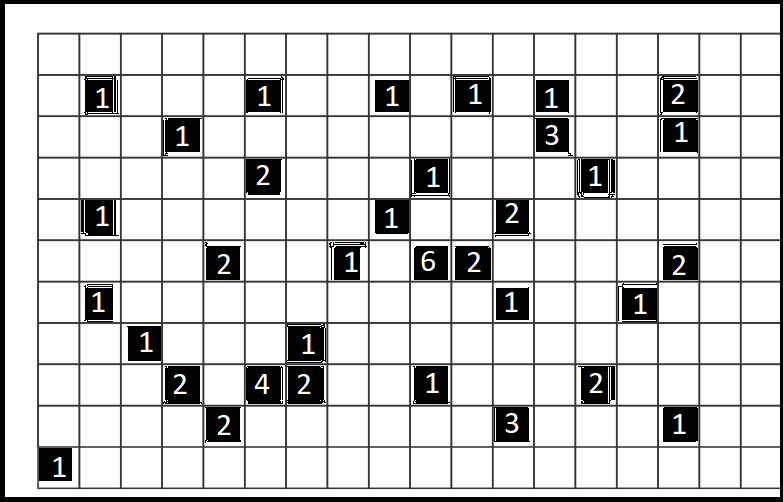


Рисунок 7 – Подсчет числа попавших пикселей неоднородностей в ячейку

В последствии производится расчет вероятности нахождения пикселя в каждой ячейке.

Вероятность (pi) того, что в выбранной ячейке будет закрашенный пиксель:

, (1)

где *ni* – число закрашенных пикселей в i-ой ячейке;

N – общее число закрашенных пикселей.

Таким образом каждой ячейке соответствует своя вероятность.

Следующим действием производится расчет статсумм:

, (2)

где Z – статсумма;

q – варьируемый параметр, «деформирующий» вероятности.

Аналитически q варьируется от - до +. Для расчетов выбирается используемая область q, шаг равняется 1.

После этого производится расчет экспоненты обобщенной корреляционной функции:

, (3)

где – обобщенная корреляционная функция.

Для того, чтобы рассчитать предел, строится график функции в двух логарифмических осях . Методом наименьших квадратов определяется наклон этого графика. Численной значение наклона является оценкой . Этот подход является аналогом правила Лопиталя [8].

Затем производится расчет обобщенных спектров размерности Реньи и – спектров.

Расчет спектра обобщенных размерностей Реньи:

, (4)

где Dq – обобщенный спектр размерностей Реньи.

Расчет – спектров:

(5)

При q = 1:

(6)

* + - 1. **Обработка результатов мультифрактального анализа**

В результате МФ анализа изображения получаются спектр размерностей Реньи и -спектр.

Пример спектра размерностей Реньи представлен на рисунке 8.

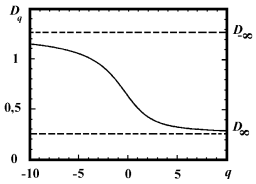


Рисунок 8 – Обобщенный спектр размерностей Реньи

Пример -спектра представлен на рисунке 9.

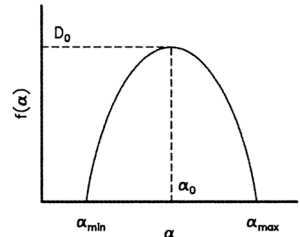


Рисунок 9 – -спектр

Для того, чтобы охарактеризовать мультифрактальность поверхности используются 5 парметров:

* – фрактальная размерность;
* – информационная размерность;
* – корреляционная размерность;
* – отражает плотность больших по размерам элементов;
* – отражает плотность небольших по размерам элементов.

Соотнеся характеристики оптических элементов после напыления с учетом технологических параметров и изображения спектров , и значений , можно сделать выводы об определенных закономерностях этих параметров для оценки качества матрицы неоднородностей.

### Разработка базы данных

Разработка базы данных была произведена с помощью СУБД MySQL, согласно ранее спроектированному словарю данных и ER-диаграмме. Скрипт базы данных находится в Приложении В.

### Разработка десктопного приложения

В качестве основных инструментов для разработки десктопного приложения использовались:

* интегрированная среда разработки Visual Studio;
* СУБД «MySQL».

Основной код программного продукта представлен в Приложении Б [9, 10].

Иллюстрация работы основного функционала программного обеспечения представлена далее.

Вид приложения при открытии отображен на рисунке 10.

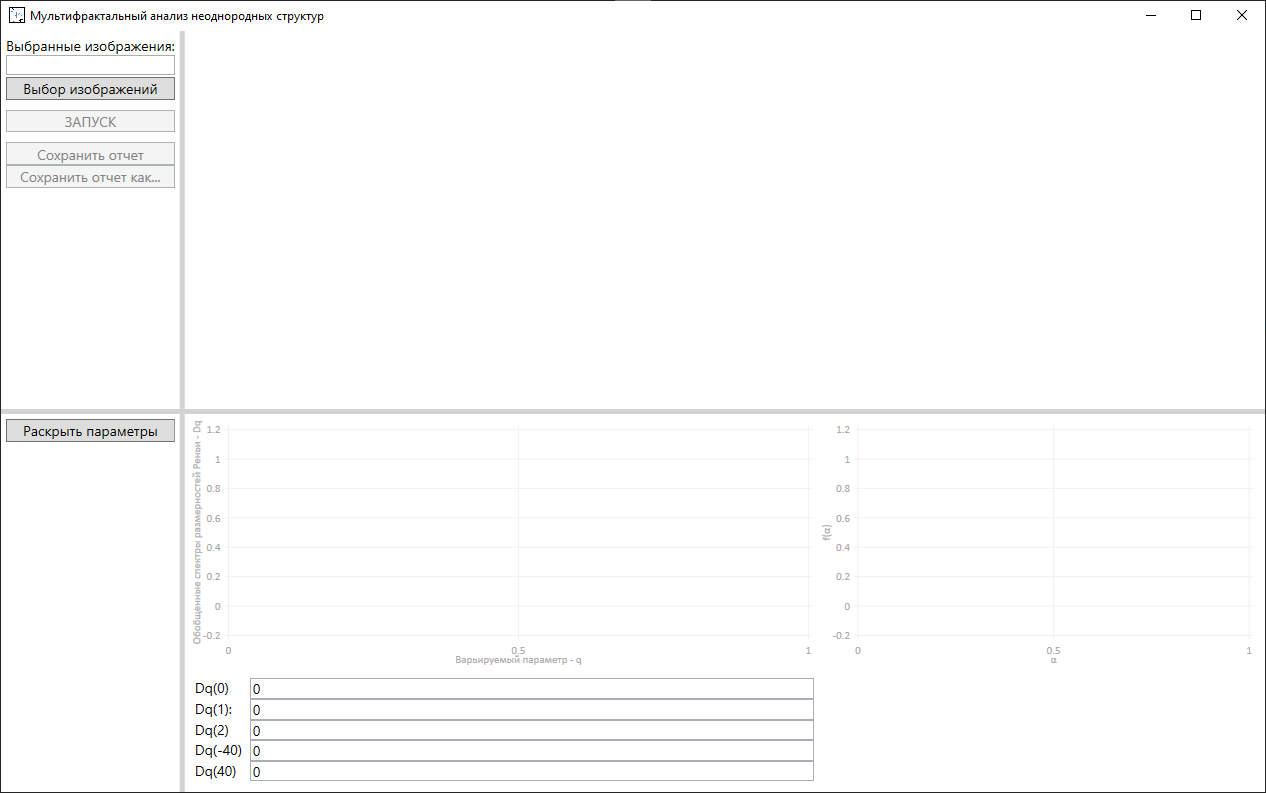


Рисунок 10 – Окно приложения при запуске

Окно выбора изображений и результат выбора изображений представлены на рисунках 11-12.

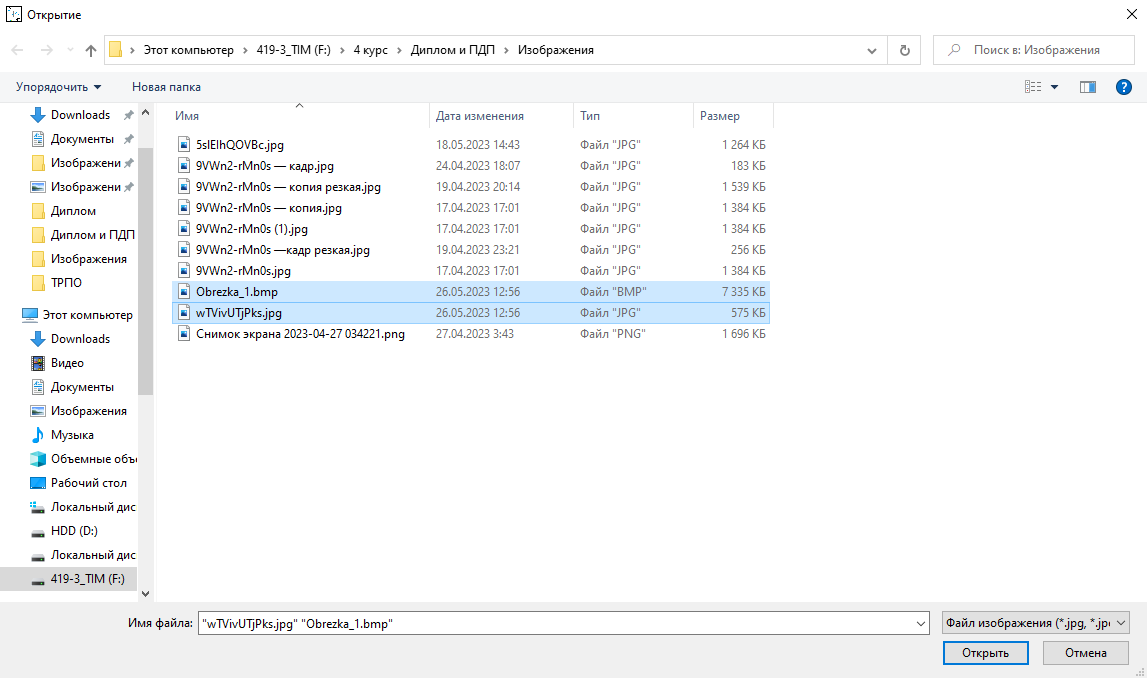


Рисунок 11 – Окно выбора изображений для обработки

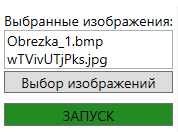


Рисунок 12 – Результат выбора нескольких изображений для обработки

Результат раскрытия области параметров отображен на рисунке13.

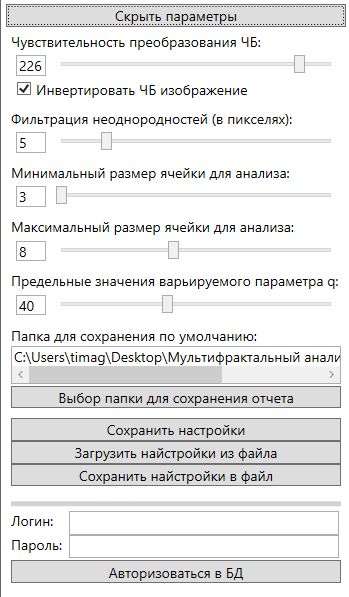


Рисунок 13 – Раскрытая область параметров

Подключение к серверу, авторизация в БД и результат данных действия отображены на рисунках 14-15.



Рисунок 14 – Подключение к серверу, авторизация в БД

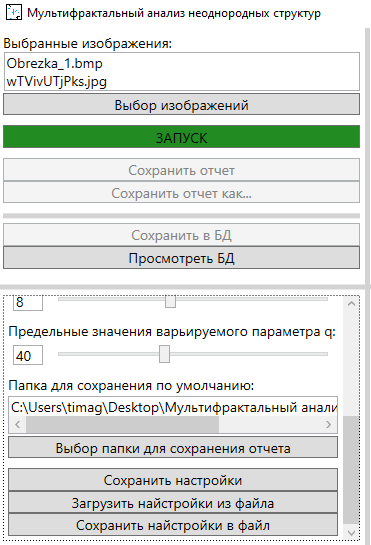


Рисунок 15 – Добавление нового функционала после подключения к серверу и авторизации в БД

Результаты запуска обработки разных изображений с разным параметрами представлены на рисунках 16-17.

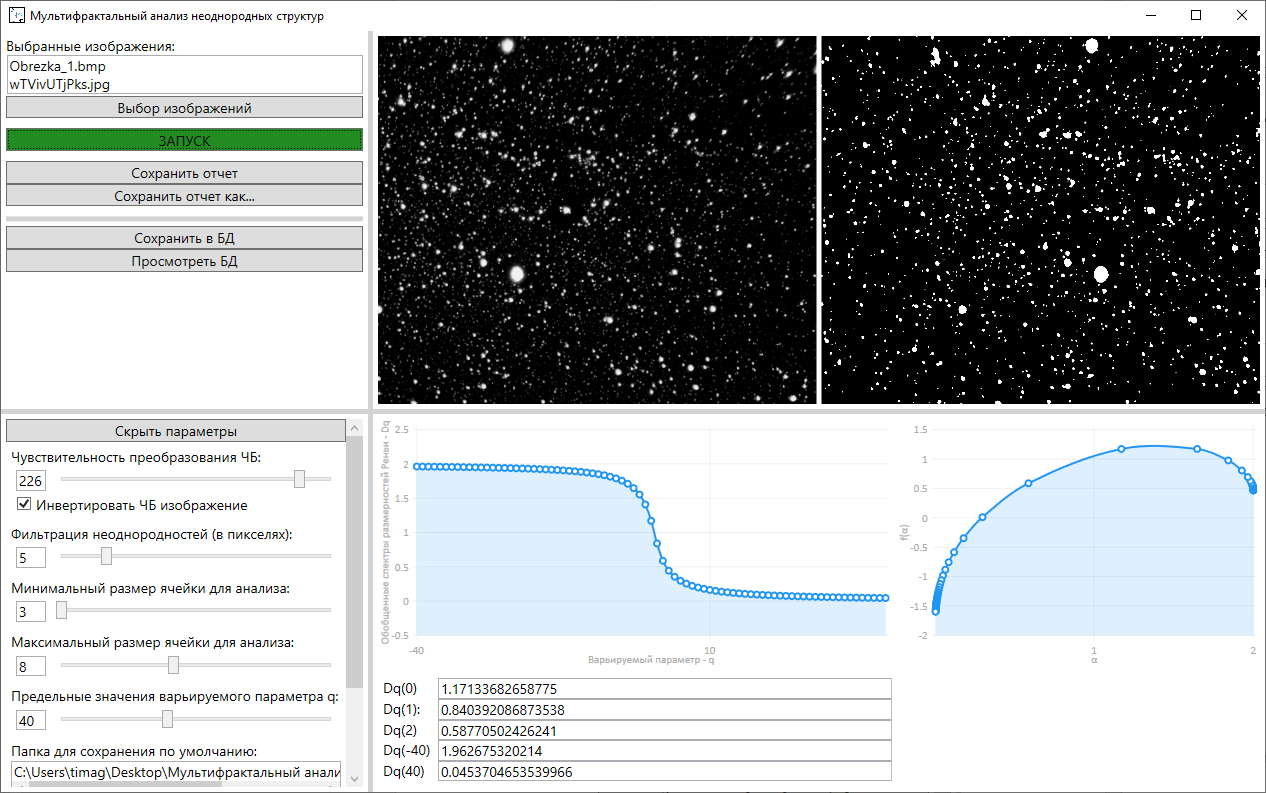


Рисунок 16 – Пример обработки № 1

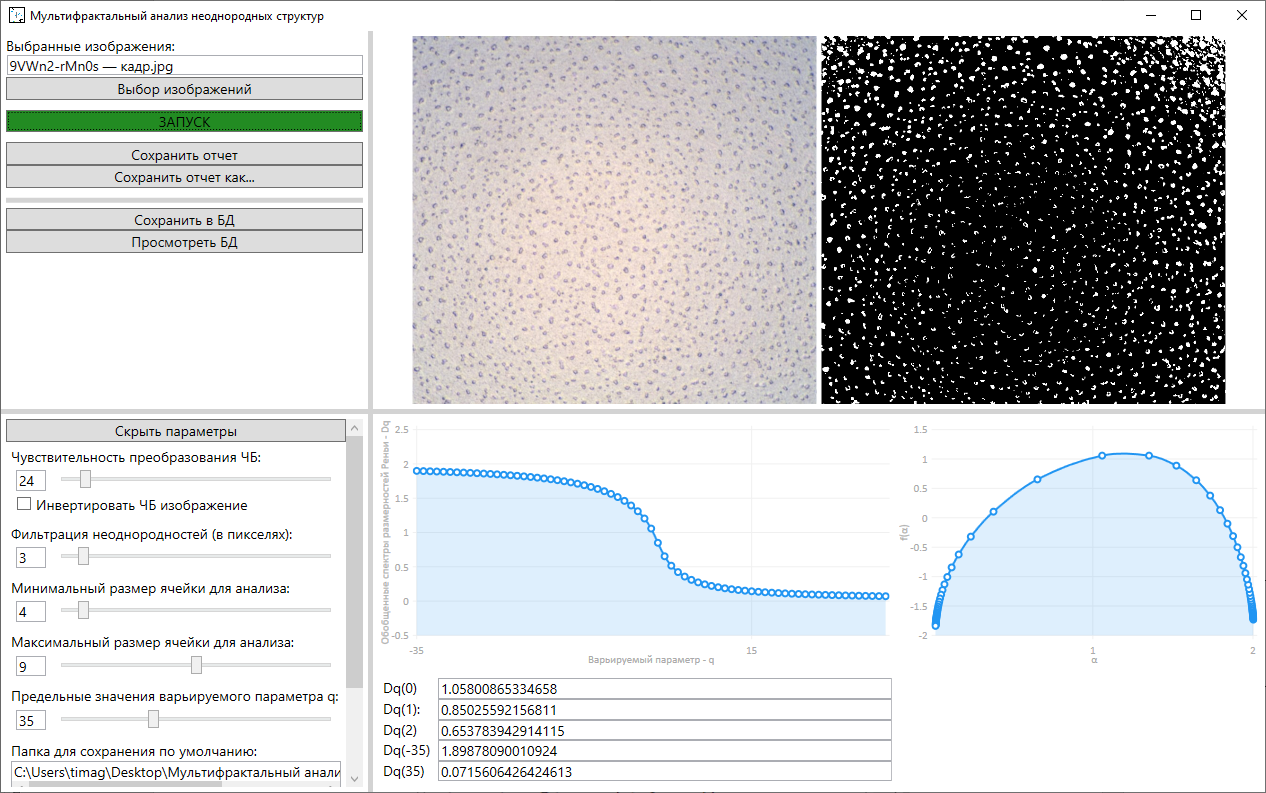


Рисунок 17 – Пример обработки № 2

Результат сохранения отчета по пути по-умолчанию, указываемому в параметрах отображен на рисунке 18.

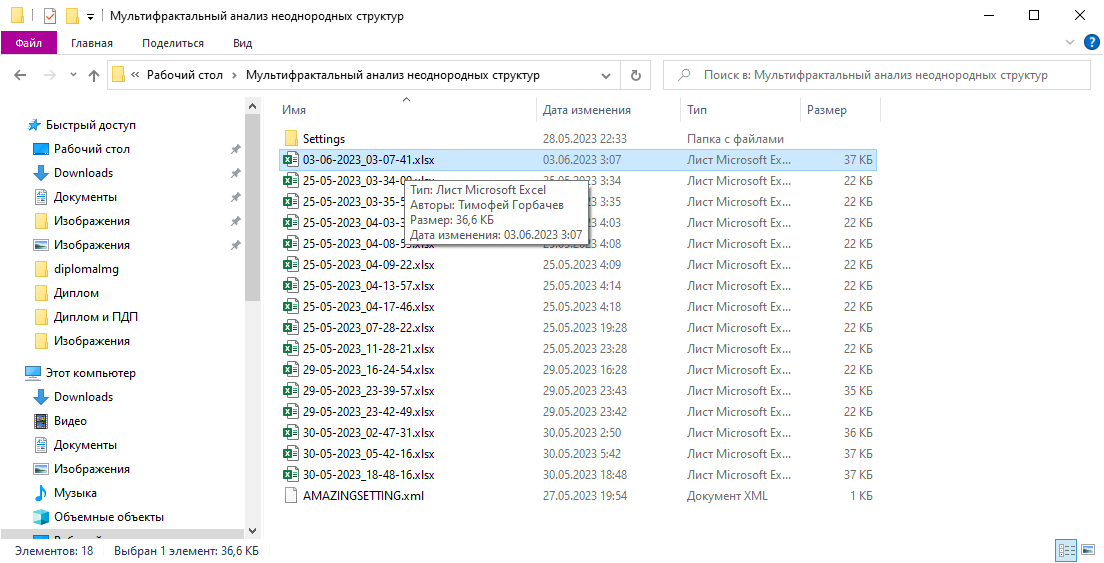


Рисунок 18 – Результат нажатия на «Сохранить отчет»

Окно сохранения «отчета как…», и результат данного сохранения отображены на рисунках 19-20.

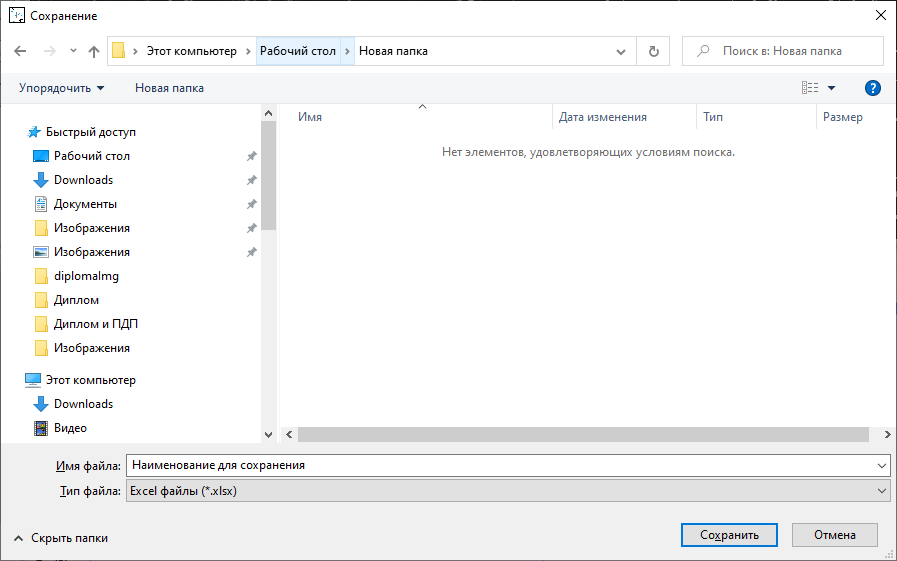


Рисунок 19 – Окно сохранения «Сохранить отчет как…»

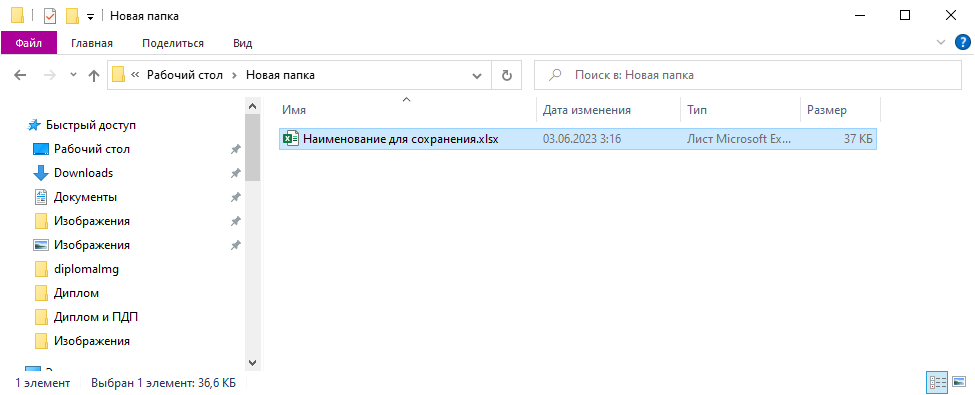


Рисунок 20 – Окно сохранения «Сохранить отчет как…»

Файл отчета представляет из себя Excel-файл, имеющий лист суммирующей информации «Общий» и листы специализированные под каждое изображение. Данный отчет позволяет получить более полную информацию по обработке. Пример отчета и примеры данных листов представлены на изображениях 21-22.

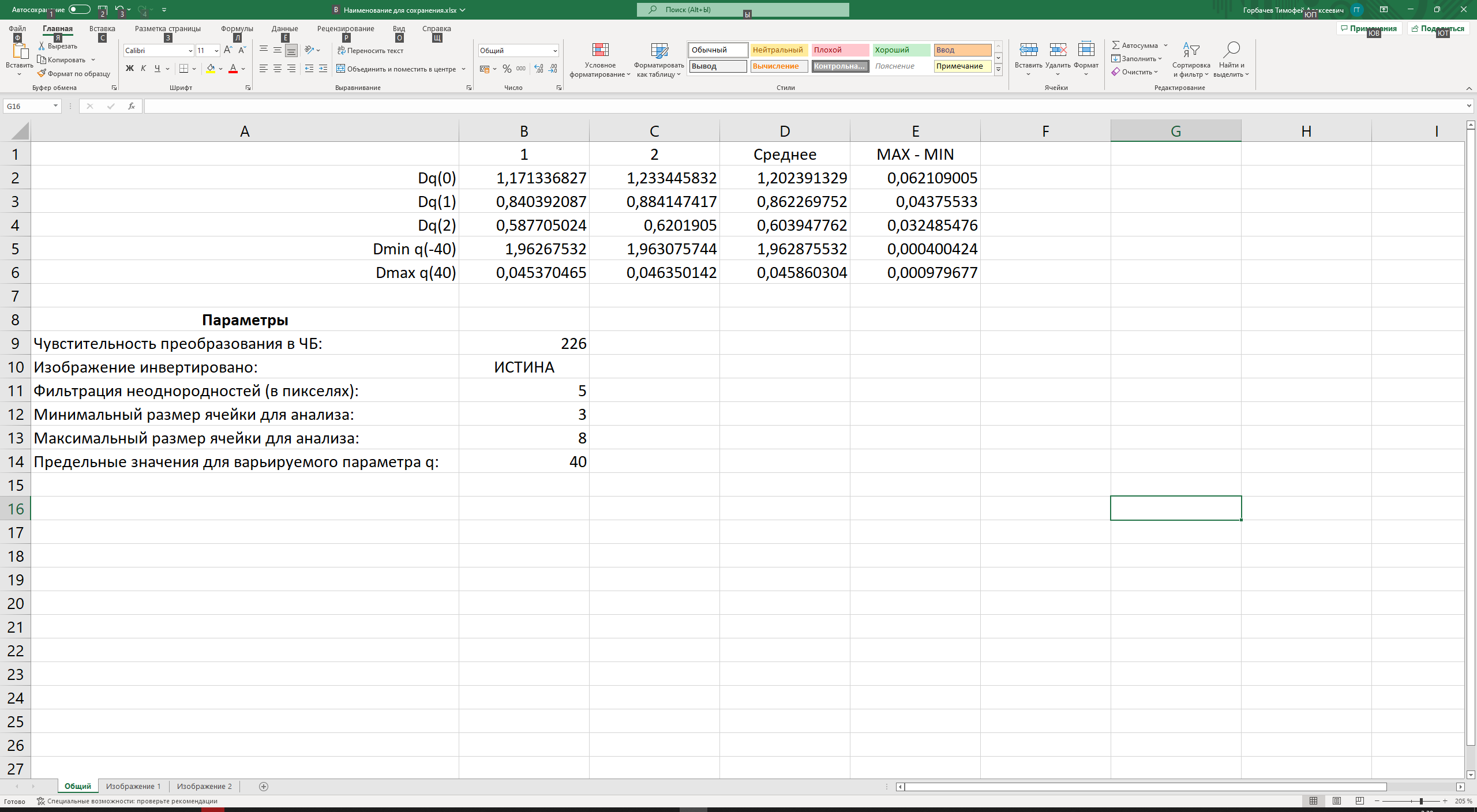


Рисунок 21 – Лист отчета «Общий»

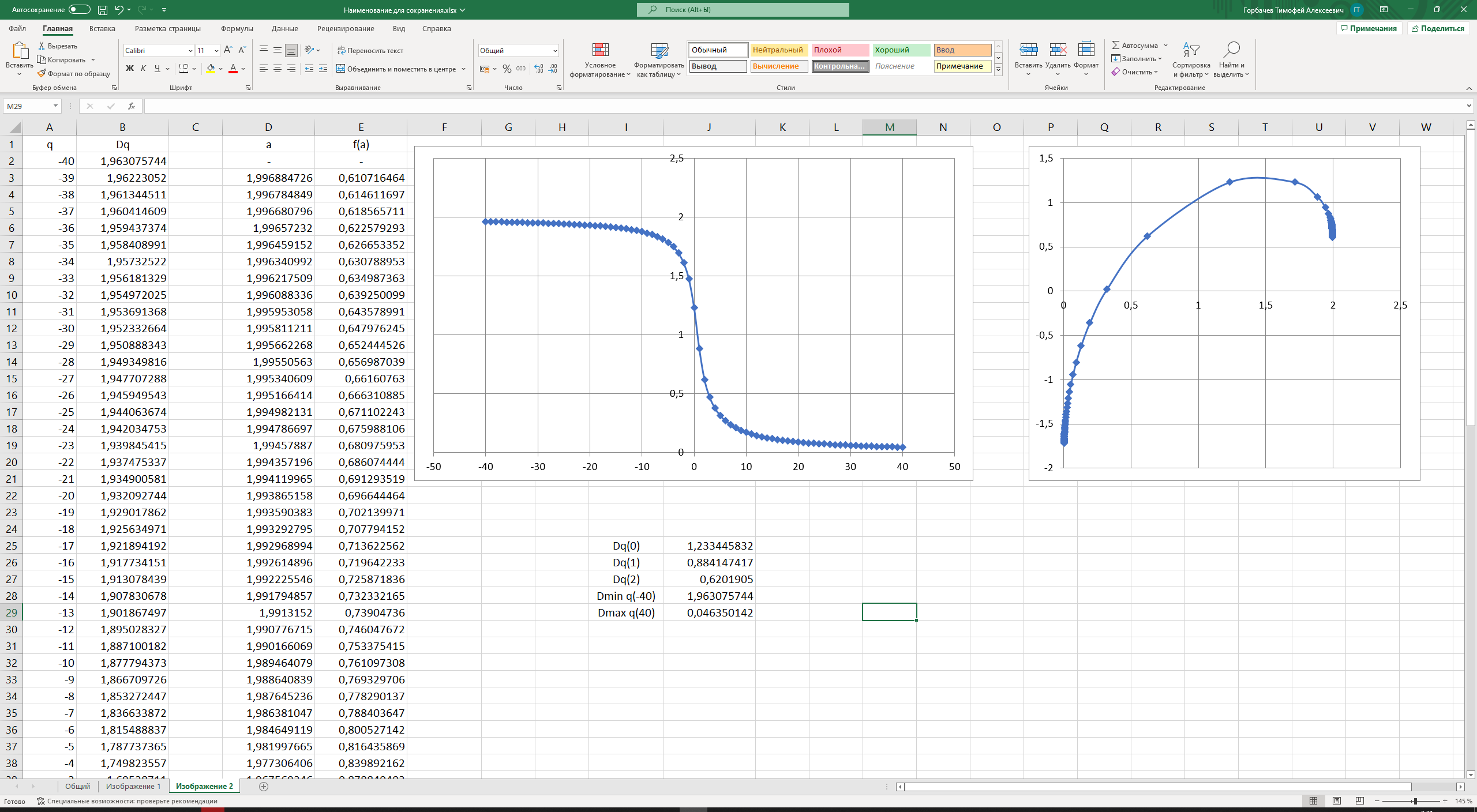


Рисунок 22 – Лист отчета «Изображение 2»

Сохранение данных в БД и просмотр данных из БД отображены на рисунке 23.

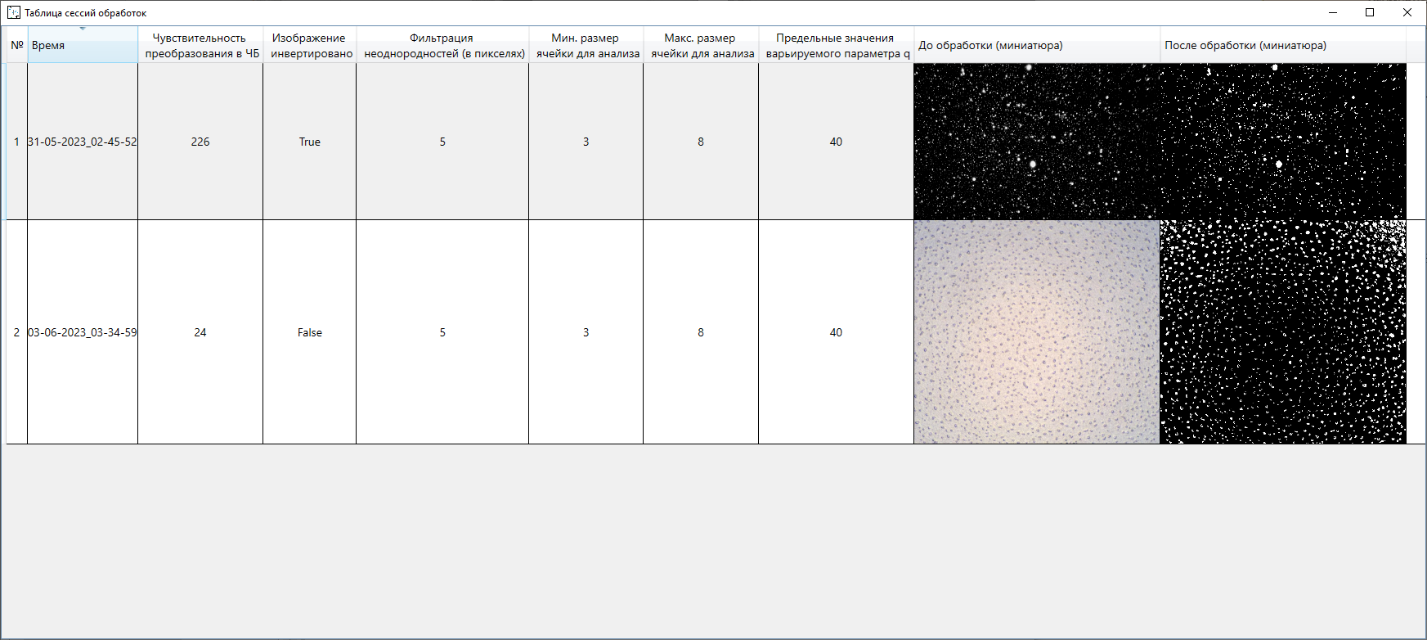


Рисунок 23 – Окно с таблицей сессий обработок, на которой отображена и новая сохраненная сессия

Изменение пути сохранения для сохранения по умолчанию отображено на рисунках 24-25.

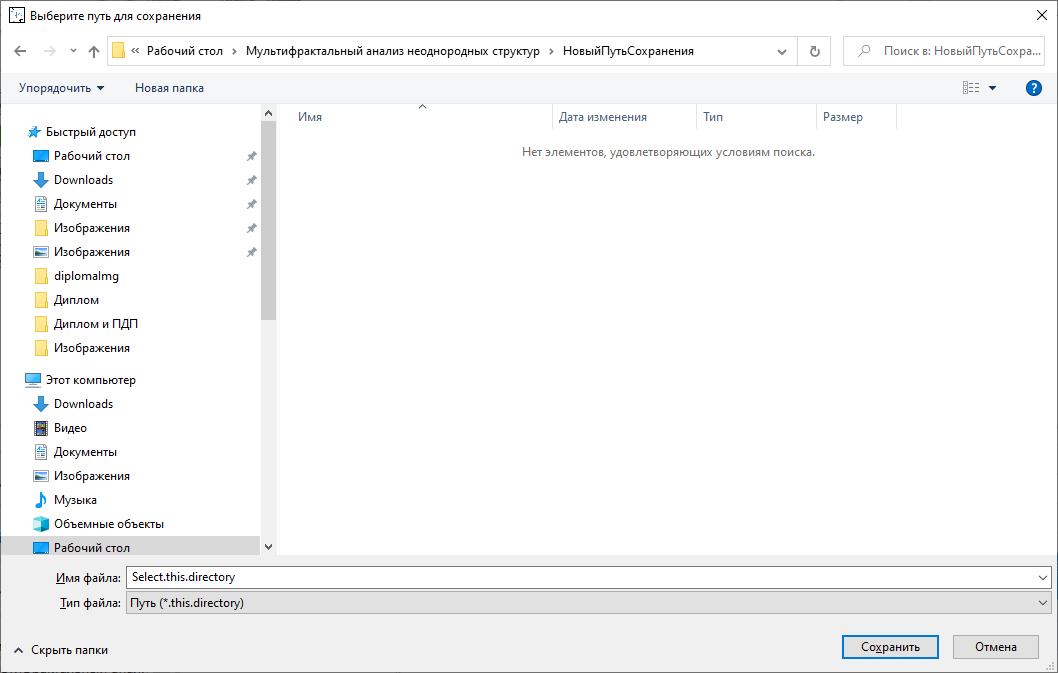


Рисунок 24 – Выбор новой директории для сохранения отчетов по умолчанию

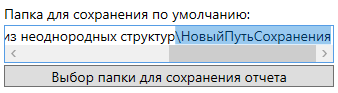


Рисунок 25 – Изменившийся путь для сохранения по умолчанию

Проверка изменения пути для сохранения по умолчанию продемонстрирована на рисунке 26.

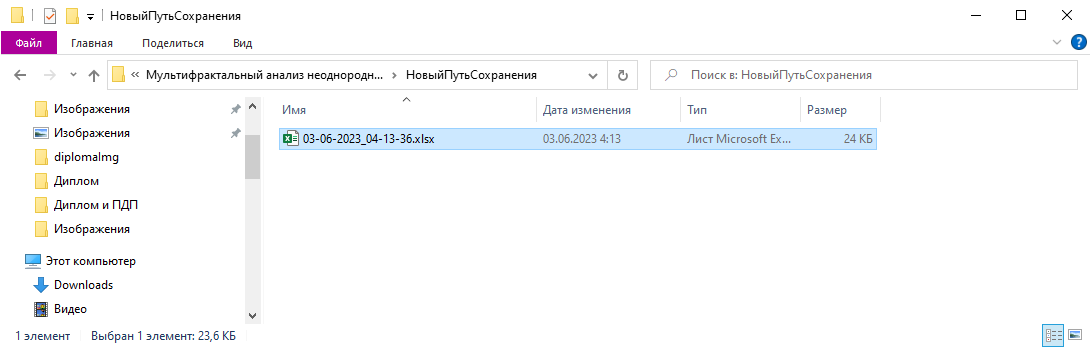


Рисунок 26 – Сохраненный по новому пути файл

Сохранение параметров в файл отображено на рисунках 27-28.

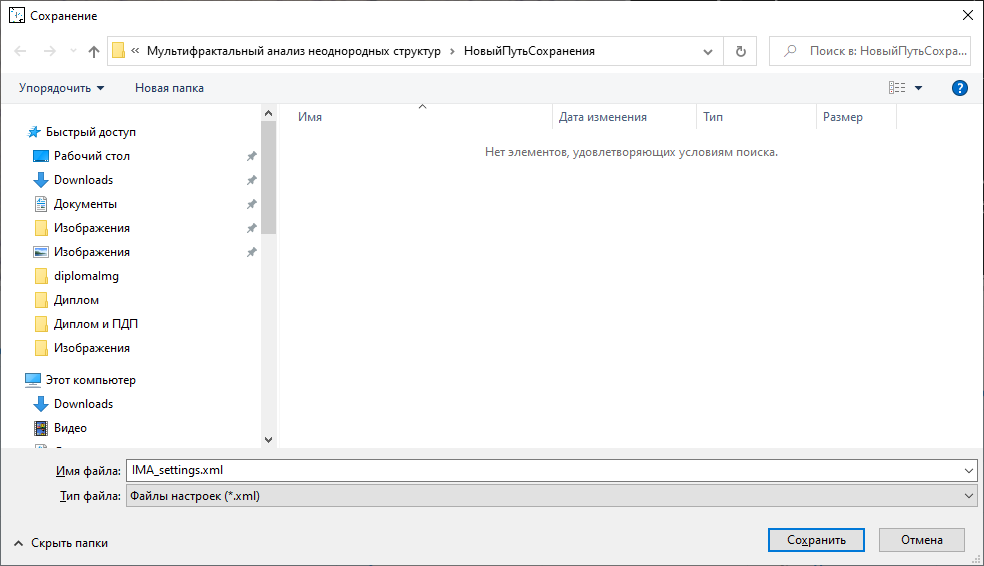


Рисунок 27 – Окно выгрузки файла текущих параметров

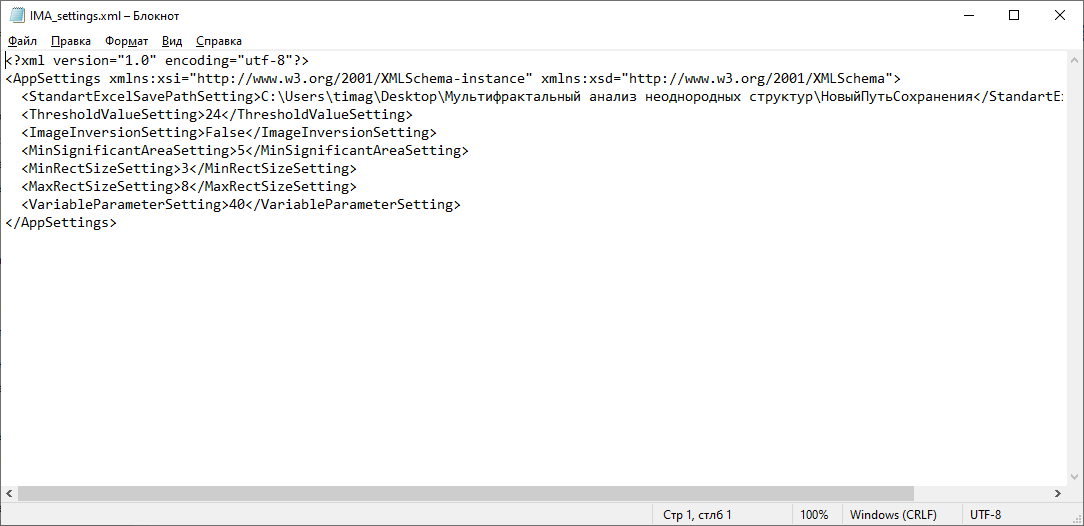


Рисунок 28 – Результат создания файла с параметрами

Загрузка параметров из файла представлена на рисунках 29-30.

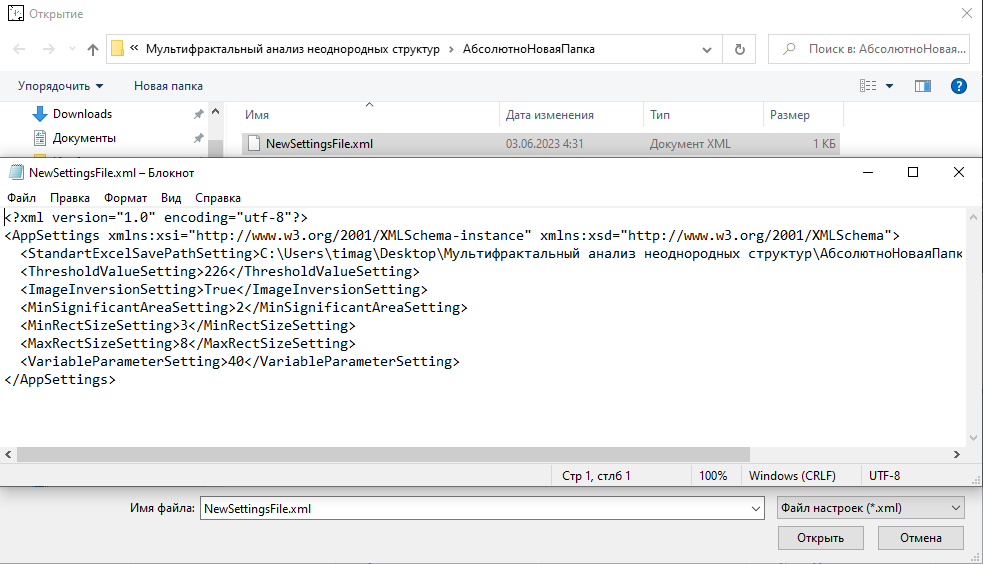


Рисунок 29 – Выбор файла с новыми параметрами



Рисунок 30 – Результат загрузки выбранного файла параметров

### Разработка API

Результатом разработки API выступило создание методов подключения к удаленному серверу, загрузки данных из БД, выгрузки данных из БД при помощи СУБД и посредством удаленного сервера.

# Экономическая часть

## Область применения программного продукта и его преимущества перед аналогичным программным продуктом

Система является модулем, предназначенным для проведения мультифрактального анализа неоднородных структур для ООО «Оптосенс».

Достоинствами данного приложения являются удобство использования в массовом производственном цикле, эргономичность и удобство использования, возможность обработки и характеризации сложных структур, слабо поддающихся классическим методам характеризации, большая чувствительность к изменениям в структуре, простота и скорость анализа для пользователя.

## Трудоемкость разработки программного продукта, квалификация исполнителя и его оклад

Трудоемкость разработки можно определить в таблице 5. Строка «Всего» отображает общую трудоемкость разработки.

Таблица 5 – Трудоемкость разработки программного продукта

|  |  |  |
| --- | --- | --- |
| **Наименование этапа** | **Условное обозначение** | **Трудоемкость выполнения этапа, час** |
| Описание задания | То | 16 |
| Разработка UML - диаграмм | Тд | 16 |
| Разработка Базы данных | Тбд | 32 |
| Проектирование программы | Тпп | 216 |
| Отладка программы | Топ | 24 |
| **Наименование этапа** | **Условное обозначение** | **Трудоемкость выполнения этапа, час** |
| Оценка качества программы | Тоц | 24 |
| Оформление документации | Тд | 24 |
| Всего (Общая трудоемкость разработки) | Тобщ | 352 |

Оклад и тарифная ставка разработчика программного продукта представлены в таблице 6.

Таблица 6 – Трудоемкость разработки программного продукта

|  |  |  |  |
| --- | --- | --- | --- |
| **Исполнитель** | **Оклад, руб.** | **Часовая тарифная ставка, руб/час** | **Количество сотрудников** |
| Разработчик-программист | 50 000 | 284,09 | 1 |

Часовая тарифная ставка ЧТС, руб./час., определяется исходя из месячного оклада, количества рабочих дней в месяце и продолжительности рабочего дня по формуле:

, (7)

где Ом – оклад исполнителя в месяц, руб./мес.;

Д – количество рабочих дней в месяце (для расчета Д = 22 раб. дня);

Тс – продолжительность рабочего дня (для расчета Тс = 8 час.).

В таблице 7 показана стоимость технических средств для разработки системы.

Таблица 7 – Стоимость технических средств разработки ИСУ

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование компонента** | **Цена, руб.** | **Количество, шт.** | **Стоимость, руб.** |
| Моноблок Lenovo IdeaCentre AIO 3 | 65 150 | 1 | 65 150 |
| Brother HL-1110R | 16 827 | 1 | 16 827 |
| Итого: | | | 81 977 |

В таблице 8 представлены затраты на расходные материалы.

Таблица 8 – Планируемые затраты на расходные материалы

|  |  |  |  |
| --- | --- | --- | --- |
| **Затраты** | **Стоимость** | **Количество** | **Сумма, руб.** |
| Интернет | 700 руб/месяц | 2 месяца | 1400 |
| Электричество | 4,88 руб/КВт\*ч | 1170 КВт\*ч | 5709,60 |
| Бумага | 519 руб. | 2 упаковки | 1038 |
| Ручка | 48 руб. | 1 штука | 48 |
| Замена картриджа | 768 руб. | 2 раза | 1536 |
| Итого: | | | 9731,60 |

## Расчет затрат на разработку

Исходные данные, связанные с разработкой программного продукта приведены в таблице 9.

Таблица 9 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Оклад разработчика | Ор | 50000 руб. |
| Время разработки | Трп | 2 месяца |
| Машинное время разработки | Тмч | 1,5 месяца |
| Коэффициент дополнительной заработной платы | Кд | 0,12 |
| Коэффициент страховых взносов | Кст | 0,3 |
| Количество единиц техники | Q | 1 шт. |
| Себестоимость содержания техники | См/ч | 10 руб./час |
| Коэффициент готовности техники | Кгт | 0,95 |
| Число рабочих дней в месяце | ЧРД | 22 дня |
| Продолжительность смены | Тст | 8 часов |
| Коэффициент сменности | Ксм | 1 |
| Коэффициент транспортных расходов | Кт | 0,15 |
| Коэффициент накладных расходов | Кнр | 0,54 |

Произведя вычисления было выявлено, полные затраты на разработку составляют 301756,34 рублей. Промежуточные результаты вычислений представлены в таблице 10.

Таблица 10 – Результаты вычислений

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Оплата труда | Зот | 112 000 руб. |
| Страховые взносы | Зст | 33 600 руб. |
| Содержание ЭВМ | Зэвм | 2508 руб. |
| Затраты на специальные программы | Зсп | 81 977 руб. |
| Расходные материалы | Зрм | 11 191,34 руб. |
| Накладные расходы | Знр | 60 480 руб. |
| Итого затрат на разработку | Зрп | 301 756,34 руб. |

Расчет полных затрат на разработку проектного решения (программного продукта) осуществляется по формуле:

Зрп = Зот + Зст + Зэвм + Зсп + Зрм + Знр, (8)

где Зот – затраты на оплату труда разработчика (разработчиков), руб.;

Зст – страховые взносы по оплате труда во внебюджетные фонды, руб.;

Зэвм – затраты, связанные с содержанием вычислительной техники, руб.;

Зсп – застраты на специальное программное обеспечение, руб.;

Зрм – затраты на расходные материалы, необходимые при разработке программного продукта, руб.;

Знр – затраты по накладным расходам, приходящиеся на разработку программного продукта, руб.;

Зрп = 112000 + 33600 + 2508 + 81977 + 11191,34 + 60480 = 301756,34

### Затраты на оплату труда разработчиков (Зот), руб

Размер фонда оплаты труда разработчиков (Зот) рассчитывается по формуле:

Зот = Ор \* Трп \* (1 + Кд), (9)

где Ор – месячный оклад разработчика проектного решения, руб./мес.;

Трп – время разработки проектного решения разработчиком, мес.(час) включает в себя машинное время работы над проектом (Тмрп);

Кд – Коэффициент дополнительной заработной платы разработчика;

Если время разработки программного продукта менее месяца, т.е. определено в часах то в формулу подставляется часовая тарифная ставка определяемая по формуле 1:

3от = 50000 \* 2 \* (1 + 0,12) = 112000

### Затраты по страховым взносам (Зст), руб

Сумма страховых взносов определяется по формуле:

Зст = Кст \* Зот, (10)

где Кст – коэффициент страховых взносов для расчета отчислений во внебюджетные фонды.

Зст = 0,3 \* 112000 = 33600

### Затраты по содержанию ЭВМ (Зэвм), руб

Затраты, связанные с эксплуатацией и содержанием ЭВМ, определяются по формуле:

Зэвм = Тмрп \* Кгт \*Q \* См/ч, (11)

где Тмрп – машинное время на разработку проектного решения, час;

Кгт – коэффициент готовности техники;

Q – Количество условных единиц, используемой техники;

См/ч – стоимость машино-часа, эксплуатации оборудования, руб. в час.

Зэвм = 264 \* 0,95 \* 1 \* 10 = 2508

Так как машинное время может измеряться в месяцах, а себестоимость машино-часа за один час, то машинное время необходимо перевести в часы.

Перевод рабочего времени в часы осуществляется по формуле:

Тмрп = Тмч \* Чрд \* Тсм \* Ксм, (12)

где Тмч – рабочее время в месяцах;

Чрд – число рабочих дней в месяце;

Тсм – продолжительность рабочей смены;

Ксм – Количество рабочих смен.

Тмрп = 1,5 \* 22 \* 8 \* 1 = 264

### Затраты на расходные материалы (Зрм), руб

Затраты на расходные материалы необходимые для разработки проектного решения определяются по формуле:

Зрм =, (13)

где Qi – количество i-го вида материала, шт.;

Pi – цена i-го вида материала, руб.;

n – количество видов материалов;

KT – коэффициент транспортных расходов.

Зрм = 9731,60 \* (1 + 0,15) = 11 191,34

### Затраты по накладным расходам (Знр)

Затраты по накладным расходам определяются по формуле:

Знр = Кнр \* Зот, (14)

где Кнр – коэффициент накладных расходов (или процент) принимается для расчета по данным предприятия;

Зот – затраты по оплате труда, руб.

Знр = 0,54 \* 112000 = 60480

## Расчет цены и прибыли

Исходные данные, связанные с разработкой программного продукта приведены в таблице 11.

Таблица 11 – Исходные данные

| **Наименование** | **Обозначение** | **Значение** |
| --- | --- | --- |
| Коэффициент рентабельности | Кр | 0,25 |
| **Наименование** | **Обозначение** | **Значение** |
| Коэффициент налога надбавочную стоимость | Кндс | 0,2 |
| Ставка налога | Кнп | 0,2 |
| Стоимость затрат на разработку и внедрение | Квпр | 301756,34 руб. |

Произведя вычисления было выявлено, что чистая прибыль будет составлять 60351,268 рублей. Промежуточные результаты вычислений представлены в таблице 13.

Таблица 12 – Результаты вычислений

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение** | **Значение** |
| Плановая прибыль | Ппл | 75 439,085 руб. |
| Налог на добавочную стоимость | НДС | 75 439,085 руб. |
| Цена программного продукта | Цпп | 452 634,51 руб. |
| Налог на прибыль | НП | 15 087,817 руб. |
| Чистая прибыль | ЧП | 60 351,268 руб. |
| Поступление в бюджет | ПБ | 90 526,902 руб. |

Цена программного продукта, который разработан одной организацией по заказу другой и не предназначен для тиражирования, определяется по формуле:

Цпп = Квпр + Ппл + НДС, руб. (15)

Цпп = 301 756,34 + 75 439,085 + 75 439,085 = 452 634,51

где Ппл – планируемая прибыль рассчитывается по формуле: Ппл = Квпр \* Кр, руб. исходя из нормативной рентабельности (коэффициента) Кр = 0,25.

Ппл = 301 756,34 \* 0,25 = 75 439,085

НДС – Налог на добавленную стоимость определяется исходя из Кндс = 0,2 (ставка налога 20%) по формуле:

НДС = (Квпр + Ппл) \* Кндс, руб (16)

НДС = (301756,34 + 75 439,085) \* 0,2 = 75 439,085

Каждое предприятие с полученной прибыли перечисляет государству налог на прибыль. На сегодня ставка налога 20% (Кнп – 0,2) от полученной прибыли, и определяется по формуле:

НП = Ппл \* Кнп, руб (17)

НП = 75 439,085 \* 0,2 = 15 087,817

Чистая прибыль, остающаяся в распоряжении предприятия:

ЧП = Ппл – НП, руб. (18)

ЧП = 75 439,085 – 15 087,817 = 60 351,268

Поступление в бюджет складываются из налога на прибыль и НДС.

ПБ = НП + НДС, руб. (19)

ПБ = 15 087,817 + 75 439,085 = 90 526,902

Исходя из расчетов экономической части затраты на разработку составили 301756,34 рублей, цена программного продукта составляет 452634,51 рублей, а чистая прибыль составляет 60 351,268 рублей. ООО «Оптосенс» готов приобрести продукт за данную цену и реализовать его в своей системе для проведения мультифрактального анализа неоднородных структур в своей коммерческой деятельности.

# Техника безопасности и охрана труда

## Анализ условий труда разработчика–программиста, работающего в ООО «Оптосенс»

Рабочее место разработчика-программиста – это его рабочий кабинет, либо часть помещения, в котором он проводит большую часть работы. Совершенно очевидно, что чем лучше приспособлено рабочее место для выполнения функций работника, тем производительнее и эффективнее будет его трудовая деятельность.

Программист имеет собственное рабочее место, оборудованное ПЭВМ. Рабочий стул (кресло) должен быть подъёмно-поворотным и регулируемым по высоте и углам наклона сиденья и спинки, а также – расстоянию спинки до переднего края сиденья.

Рабочая поверхность стола сделана из дерева. На столе размещены ЭВМ, монитор и принтер. Площадь поверхности стола предполагает размещение монитора, устройств ввода/вывода и рабочей зоны с местом расположения оперативной документации.

Кабинет, в котором расположено рабочее место, представляет собой помещение площадью 15 м2, длиной 5 м, шириной 3 м и высотой 2,85 м.

Для создания благоприятных условий для зрительного восприятия кабинет оформлен в светло-бежевом цвете.

Для создания и поддержания независимо от наружных условий оптимальных значений температуры, влажности, чистоты воздуха, в холодное время используется водяное отопление, в тёплое время года применяется кондиционирование воздуха.

Режим работы разработчика-программиста – односменный с пятидневной рабочей неделей и ненормированным рабочим днём. Продолжительность ежегодного отпуска – 28 календарных дней.

При идентификации вредных производственных факторов было установлено, что опасными могут являться:

* шум и вибрация, источником которого является оргтехника;
* электромагнитные поля и излучения от дисплея ПК;
* статическое электричество, накапливаемое на клавиатуре, дисплее, корпусе системного блока;
* электрический ток в электрических сетях;
* запылённость рабочей зоны, так влажная уборка рабочей поверхности проводиться не ежедневно, а только по мере загрязнения;
* статические перегрузки, так как работа разработчика-программиста связана с неизменной статической позой, иногда неудобной позе (работа за ПК), что может привести к искривлению позвоночника, остеохондрозу, застою в органах малого таза;
* перенапряжение анализаторов, в большей степени зрительного, что может привести к различным нарушениям зрения (снижение остроты, спазм и уменьшение запаса аккомодации), помимо этого повышение зрительной нагрузки может приводить к общему утомлению, возникновению головной боли, ухудшению самочувствия;
* умственное перенапряжение, так как в процессе деятельности программист анализирует и перерабатывает огромное количество административных сетей, разрабатывает программные продукты, поддерживает технику в рабочем состоянии.

В целях выявления вредных и (или) опасных производственных факторов и осуществления мероприятий по приведению условий труда в соответствие государственными нормативными требованиями охраны трудаежегодно проводится аттестация рабочих мест по условиям труда – оценка условий труда на рабочих местах. Аттестации подлежат все имеющиеся в организации рабочие места.

При аттестации рабочих мест проводят оценку условий труда, гигиеническую оценку и оценку травмоопасности рабочих мест. При этом учитывают наличие средств коллективной защиты, обеспеченность работников средствами индивидуальной защиты и определяют эффективность этих средств.

Согласно санитарным правилам СП 2.2.3670-20 «Санитарно-эпидемиологические требования к условиям труда», дисплей должен располагаться на расстоянии 60-70 см, но не менее 50 см от глаз. Между боковыми поверхностями мониторов должно быть не менее 1,2 м. При использовании жидкокристаллических дисплеев на каждый компьютер должно приходиться не менее 4,5 м2 площади. На дисплей ПК не должен попадать прямой солнечный свет во избежание бликов и повышенной нагрузки на зрение.

Поэтому в кабинете, где работает программист, стоит только один компьютер, а комплектующая оргтехника расположена достаточно далеко непосредственно от самого специалиста. Это обусловлено тем, что все копировальные аппараты, принтеры и факс в периоды своей работы выделяют не только вредные токсичные газы, но и шумы и вибрацию. Данные факторы оказывают гораздо меньшее негативное влияние, нежели излучение от компьютера, но также являются вредными и могут нанести вред здоровью сотрудника.

Для снижения напряжения с глаз, а также для снятия мышечного напряжения во время рабочего дня проводится проветривание в течение 10 – 15 минут каждые 2 часа.

Анализ рабочего места разработчика-программиста ООО «Оптосенс» показал, что оно полностью соответствует СП 2.2.3670–20 «Санитарно-эпидемиологические требования к условиям труда».

## Расчёт искусственного освещения в помещении

Через зрительный анализатор человек получает около 80% из общего объема информации. Качество поступающей, в основном, зависит от освещения в помещении. Если оно неудовлетворительное, это приводит к утомлению организма в целом. Кроме того, нерациональное освещение может явиться причиной травматизма: плохо освещенные опасные зоны, слепящие источники света и блики от них, резкие тени ухудшают видимость настолько, что вызывают полную потерю ориентировки работающих, снижают производительность труда и увеличивают брак продукции. Поэтому необходимо достаточное искусственное освещение для рабочего места.

Условия работы для расчета показателя освещенности рабочего места разработчика-программиста являются:

* помещение размером 15 квадратных метров;
* высота помещения 2,85 метров, длина 5 метров, ширина 3 метра;
* светильники с люминесцентными лампами дневного света с потребляемой мощностью 65 Ватт в количестве трех штук, новые;
* крупногабаритная мебель отсутствует.

Расчёт общего равномерного искусственного освещения горизонтальной рабочей поверхности выполняется методом коэффициента использования светового потока. Световой поток (лм) одной лампы рассчитывается по формуле 20:

 (20)

где Ен – нормируемая минимальная допустимая освещённость по СНиП 23-05-95, для помещения 300 Лк;

*S* – площадь освещаемого помещения, м2;

z – коэффициент неравномерности освещения, *z* = 1,1;

K – коэффициент запаса, учитывающий снижение освещённости в процессе эксплуатации вследствие загрязнения и старения ламп и светильников, а также снижения отражающих свойств поверхностей помещения для кабинета будет равен 1,4;

Nc – число светильников в помещении;

– коэффициент затенения, обычно 1;

=i – коэффициент использования светового потока.

Коэффициент использования светового потока, давший название методу расчёта, определяется по индексу помещения i в зависимости от типа светильника и коэффициентов отражения света от потолка, стен и пола по формулам 21-23:

, (21)

где А, В – длина и ширина помещения в плане, м.;

Нп – высота помещения, м.;

Нс – высота рабочей поверхности над полом, м.

(22)

(23)

В помещении используются светильники с люминесцентными лампами дневного света с потребляемой мощностью 65 Ватт, световой поток которых 4000 Лм, который в свою очередь превышает минимально необходимый полученный в ходе расчетов показатель в 2655,17 Лм для одной лампы по заданному нормативу в 300 Лк, поэтому освещение на рабочем месте разработчика-программиста в ООО «Оптосенс» соответствует СНиП 23-05-95.

## Электробезопасность на предприятии

При поступлении на работу сотрудник должен пройти инструктаж по технике безопасности и электробезопасности. Работника знакомят с основными правилами по технике безопасности, предлагают внимательно прочитать действующие на предприятии инструкции, поясняя при этом отдельные правила и требования.

Инструктаж по технике безопасности при выполнении конкретной работы проводит руководитель соответствующего производственного участка, показывая инструктируемому правильные безопасные приемы работы.

Повторный инструктаж проводится для рабочих независимо от их квалификации, стажа и опыта работы, не реже 1 раза в 6 месяцев по программе инструктажа на рабочем месте.

При нарушении работающими правил и инструкций по технике безопасности, технологической и производственной дисциплины, а также в случаях изменения технологического процесса или вида работы проводят дополнительные инструктажи.

Все инструктажи оформляются записями в специальном журнале с указанием номеров или шифров инструкций. Журнал о проведении инструктажа хранится у руководителя подразделения.

Работник на рабочем месте должен соблюдать общие меры электробезопасности. При этом запрещается:

* прикасаться к задней панели системного блока при включенном питании;
* допускать попадание влаги на поверхность системного блока (процессора), монитора, рабочую поверхность клавиатуры, дисководов, принтеров и др. устройств;
* производить самостоятельно вскрытие и ремонт оборудования;
* пользоваться неисправными электроприборами и электропроводкой;
* ремонтировать электроприборы самостоятельно;
* подвешивать электропровода на гвоздях, металлических и деревянных предметах, перекручивать провод, закладывать провод и шнуры на водопроводные трубы и батареи отопления, вешать что-либо на провода, вытягивать за шнур вилку из розетки;
* прикасаться одновременно к персональному компьютеру и к устройствам, имеющим соединение с землей (радиаторы отопления, водопроводные краны, трубы и т.п.);
* пользоваться самодельными электронагревательными приборами и электроприборами с открытой спиралью;
* наступать на переносимые электрические провода, лежащие на полу.
* При работе с электроприборами и оргтехникой (персональные компьютеры, принтеры, сканеры, копировальные аппараты, факсы, бытовые электроприборы, приборы освещения):
* Автоматические выключатели и электрические предохранители должны быть всегда исправны.
* Изоляция электропроводки, электроприборов, выключателей, штепсельных розеток, ламповых патронов и светильников, а также шнуров, с помощью которых включаются в электросеть электроприборы, должны быть в исправном состоянии.
* Электроприборы необходимо хранить в сухом месте, избегать резких колебаний температуры, вибрации, сотрясений.
* Для подогрева воды пользоваться сертифицированными электроприборами с закрытой спиралью и устройством автоматического отключения, с применением несгораемых подставок.

В ООО «Оптосенс» соблюдаются все правила электробезопасности и своевременно проводятся инструктажи.

## Пожарная безопасность на предприятии

Инструктаж по пожарной безопасности проводится по программе, разработанной ответственным по охране труда организации, с учетом требований стандартов, правил, норм и инструкций о мерах пожарной безопасности. Продолжительность инструктажа устанавливается в соответствии с утвержденной программой. Инструктаж по пожарной безопасности, как правило, проводится совместно с инструктажем по технике безопасности и в те же сроки.

Инструктаж по пожарной безопасности проходят все работники организации, независимо от их образования, стажа работы по данной профессии или должности, временные работники, командированные, обучающиеся и студенты, прибывшие на практику.

О проведении инструктажа по пожарной безопасности работник, проводивший инструктаж, делает запись в журнале учета проведения инструктажей по пожарной безопасности, с обязательной подписью инструктируемого и инструктирующего. При регистрации внепланового инструктажа по пожарной безопасности указывают причину его проведения.

Все работники организаций должны допускаться к работе только после прохождения противопожарного инструктажа, а при изменении специфики работы проходить дополнительное обучение предупреждению и тушению возможных пожаров в порядке, установленном руководителем.

На проведение первичного противопожарного инструктажа необходимо отводить не менее 1 ч. Инструктируемые должны ознакомиться:

* С действующими на объекте правилами пожарной безопасности и инструкциями;
* С производственными участками, наиболее опасными в пожарном отношении, где запрещается курить, применять открытый огонь;
* С возможными причинами возникновения пожара и мерами его предупреждения;
* С практическими действиями в случае возникновения пожара – вызов пожарной помощи, использование первичных средств пожаротушения, место расположения ближайшего телефона и ознакомление с правилами поведения в случае возникновения пожара, эвакуации людей и материальных ценностей.

При первичном инструктаже инструктирующий обязан рассказать о производственных установках с повышенной пожарной опасностью, мерах предотвращения пожаров и загораний, указать место курения, ознакомить вновь поступившего с имеющимися на объекте средствами пожаротушения, показать ближайший телефон и объяснить правила поведения в случае возникновения пожара.

Проведение противопожарного инструктажа в обязательном порядке должно сопровождаться практическим показом способов использования имеющихся на объекте средств пожаротушения (противогазы, респираторы, огнетушители и т.д.).

Соблюдение рассмотренных в данном разделе мероприятий позволяет снизить утомляемость и травматизм, повысить производительность труда, обеспечить комфортные условия трудовой деятельности.

В ООО «Оптосенс» своевременно проводятся инструктажи по пожарной безопасности. Все обязательные правила пожарной безопасности соблюдены.

Вывод: в данной главе была успешно описана техника безопасности и охраны труда сотрудника разработчика-программиста ООО «Оптосенс». Все нормы техники безопасности и охраны труда соблюдены.

# ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта были созданы программные модули, с помощью которых происходит мультифрактальный анализ структур полупроводниковых материалов, выявляются и исследуются зависимости между их физическими свойствами и структурой поверхности. Программа предназначена для сотрудников ООО «Оптосенс». Разработанное приложение позволяет сократить время оценки качества полупроводниковых материалов по их изображениям, полученным в оптическом микроскопе.

Результаты работы показали, что создание программных модулей для оценки физико-химических процессов и свойств имеет большой потенциал для развития. Благодаря использованию передовых технологий и знаний в области программирования, удалось создать программу, которая обладает высокой функциональностью и позволяет пользователям производить количественный анализ полупроводниковых материалов.

В заключение можно отметить, что приложение имеет практическую ценность, поскольку оно может быть использовано на предприятиях для оценки качества физико-химических параметров неоднородных структур. Также создание такого проекта открывает новые возможности для развития десктопных приложений в области моделирования и анализа физико-химических процессов и материалов с развитой структурой поверхности.

Таким образом, в проекте последовательно выполнены все задачи и была достигнута поставленная цель по разработке программных модулей для оценки физико-химических процессов и свойств, которые помогают пользователю проводить мультифрактальный анализ неоднородных структур для использования в оценке качества полупроводниковых материалов по их изображениям, полученным в оптическом микроскопе.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Рихтер, Д. CLR via C# Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Д. Рихтер. – СПб.: ПИТЕР, 2022. –895 с.
2. Хейлсберг, А. Язык программирования C#, 4-е издание / А. Хейлсберг, М. Торгерсен, С. Вилтамут, П. Голд. – СПб.: ПИТЕР, 2011. –773 с.
3. Плещев, В. В. Сравнительный анализ программных средств разработки приложений и баз данных и индивидуализация учебного процесса их изучения / В. В. Плещев. – Текст: непосредственный // Известия УрГЭУ. – 2003. – № 7 – С. 69.
4. Мартин, Р. Чистый Код / Роберт Мартин. – СПб.: ПИТЕР, 2019. – 457 с.
5. Мартин, Р. Чистая Архитектура / Роберт Мартин. – СПб.: ПИТЕР, 2019. – 350 с.
6. Фленов, М. Библия C#, 3-е издание / М. Фленов. – СПб.: БХВ, 2016. – 544 с.
7. Культин Н., Microsoft Visual C# в задачах и примерах, 2-е издание / Н. Культин. – СПб.: БХВ, 2015. – 265 с.
8. Албахари, Д. Карманный справочник: скорая помощь для программистов на C# 8.0 / Д. Албахари, Б. Албахари. – СПб.: Диалектика, 2020. – 240 с.
9. Албахари, Д. Справочник. Полное описание языка / Д. Албахари, Б. Албахари. – СПб.: Диалектика, 2018. – 1024 с.
10. Мартин, Р. Идеальный программист / Роберт Мартин. – СПб.: ПИТЕР, 2019. – 214 с.
11. ГОСТ 19.105-78 Общие требования к программным документам
12. ГОСТ 19.106-78 Требования к программным документам, выполненным печатным способом
13. ГОСТ 19.201-78 Техническое задание. Требования к содержанию и оформлению
14. ГОСТ 19.301-78 Программа и методика испытаний. Требования к содержанию и оформлению
15. ГОСТ 19.401-78 Текст программы. Требования к содержанию и оформлению
16. ГОСТ 19.402-78 Описание программы. Требования к содержанию и оформлению
17. ГОСТ 19.505-79 Руководство оператора. Требования к содержанию и оформлению
18. ГОСТ 19.701-90 Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
19. «C# programming guide» [Электронный ресурс] – Режим доступа : https://learn.microsoft.com/ru-ru/dotnet/csharp/programming-guide/, свободный.
20. «Awesome .NET» [Электронный ресурс] – Режим доступа : https://github.com/quozd/awesome-dotnet/blob/master/README.md, свободный.
21. «Полное руководство по языку программирования С# 11 и платформе .NET 7» [Электронный ресурс] – Режим доступа : [https://metanit.com/sharp/tutorial/](https://metanit.com/sharp/tutorial/,%20), свободный.
22. «Документация библиотеки компьютерного зрения с открытым кодом» [Электронный ресурс] – Режим доступа : https://docs.opencv.org/4.x/, свободный.

**ПРИЛОЖЕНИЕ А**(обязательное) **Список используемых сокращений**

**А.1 Список используемых сокращений**

1. ПО (от англ. «software») – программное обеспечение: одна или несколько программ (комплекс), которые используются для нормального функционирования компьютера;
2. База Данных (или «БД», от англ. «Database») – 1) программный или программно-аппаратный комплекс, разработанный для хранения большого объема различной информации; 2) совокупность данных, хранимых в соответствии со схемой данных, операции с которыми выполняют в соответствии с правилами средств моделирования данных;
3. API – Application program interface – программный интерфейс;
4. UML –Unified Modeling Language – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур;
5. WPF – Windows Presentation Foundation – система для построения клиентских приложений Windows, графическая подсистема в составе .NET Framework, использующая язык XAML;
6. IDE – Integrated development environment – комплекс программных средств, используемый программистами для разработки программного обеспечения;
7. XML – Extensible Markup Language – расширяемый язык разметки;
8. СУБД – Система управления базами данных;
9. ПК – персональный компьютер;
10. ЭВМ – электронно-вычислительная машина;
11. МФ – мультифрактальный.

**ПРИЛОЖЕНИЕ Б**(обязательное) **Листинг исходного кода приложения**

**Б.1 Листинг классов приложения**

using Microsoft.Win32;

using OpenCvSharp;

using System.Windows;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Media.Imaging;

using MultifractalAnalysis.ViewModel.Chart;

using MultifractalAnalysis.Model.Data.DataBase;

using MultifractalAnalysis.View;

using MultifractalAnalysis.Model.Functionality;

using MultifractalAnalysis.ViewModel.Support;

using static MultifractalAnalysis.Model.Functionality.ExcelFileCreation;

using static MultifractalAnalysis.Model.Functionality.ImageProcessing;

namespace MultifractalAnalysis.ViewModel

{

public class ImageProcessingWindowVM : BaseViewModel

{

public ImageProcessingWindowVM()

{

// ViewModels для графиков

RenieChartVM = new RenieChartVM();

SpectraChartVM = new SpectraChartVM();

// Значения по умолчанию

\_uneditedImage = null;

\_editedImage = null;

\_imagesFullNames = new List<string>();

\_imagesShortNames = String.Empty;

\_maxTextBoxSize = 205;

\_trueRenie = null;

\_spectraIntermediateDiffs = null;

\_spectra = null;

\_renieOn0 = 0;

\_renieOn1 = 0;

\_renieOn2 = 0;

\_renieOnMin = 0;

\_renieOnMax = 0;

\_timeStamp = "";

\_processingBtnIsEnabled = false;

\_saveBtnIsEnabled= false;

\_settingsVisibility = false;

\_settingsVisibilityBtnText = "Раскрыть параметры";

\_dataBaseWorker = null;

\_DBLogin = "";

\_DBpassword = "";

\_loginVisibility = true;

\_dataBaseButtonsVisibility = false;

var path = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) +

"\\Мультифрактальный анализ неоднородных структур\\Settings\\Default\\";

if (!Directory.Exists(path))

{

Directory.CreateDirectory(path);

}

\_defaultSettingsFilePath = path + \_settingsFilename;

Settings = new AppSettings();

//Проверка наличия файла настроек

if (File.Exists(\_defaultSettingsFilePath))

{

// Если существует - чтение файла

GetSettingsValuesFromFile(\_defaultSettingsFilePath);

}

else

{

// Иначе - создание файла со стандартными параметрами

SetDefaultSettings();

SetSettingsValues(Settings);

Settings.Save(DefaultSettingsFilePath);

}

}

protected override void OnPropertyChanged(string propertyName, object oldValue, object newValue)

{

base.OnPropertyChanged(propertyName, oldValue, newValue);

if (propertyName == nameof(ImagesFullNames))

{

if (ImagesFullNames != null && ImagesFullNames.Count > 0)

{

ImagesShortNames = String.Empty;

for (int i = 0; i < ImagesFullNames.Count; i++)

{

ImagesShortNames += System.IO.Path.GetFileName(ImagesFullNames[i]);

if (i < ImagesFullNames.Count - 1)

{

ImagesShortNames += "\n";

}

}

ProcessingBtnIsEnabled = true;

}

else

{

ImagesShortNames = "";

ProcessingBtnIsEnabled = false;

}

}

if (propertyName == nameof(SettingsVisibility))

{

if (\_settingsVisibility == true)

{

MaxTextBoxSize = 356;

}

else

{

MaxTextBoxSize = 205;

}

}

}

// Установка стандартных параметров динамических настроек приложения

private void SetDefaultSettings()

{

// Значения по умолчанию

\_standartExcelSavePath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + "\\Мультифрактальный анализ неоднородных структур";

\_thresholdValue = 24;

\_isGoingToBeInverted = false;

\_minSignificantArea = 5;

\_minRectSize = 3;

\_maxRectSize = 8;

\_variableParameter = 40;

}

// Переменная для сохранения времени обработки изображения

private string \_timeStamp;

// Постоянное имя файла настоек

private const string \_settingsFilename = "IMA\_settings.xml"; // inhomogeneities multifractal analysis

// Путь до файла настроек

private string \_defaultSettingsFilePath;

public string DefaultSettingsFilePath

{

get => \_defaultSettingsFilePath;

set => \_defaultSettingsFilePath = value;

}

// Класс настроек

public AppSettings Settings { get; private set; }

private void GetSettingsValuesFromFile(String path)

{

Settings = Settings.Read(path);

StandartExcelSavePath = Settings.StandartExcelSavePathSetting;

ThresholdValue = Convert.ToInt32(Settings.ThresholdValueSetting);

IsGoingToBeInverted = Convert.ToBoolean(Settings.ImageInversionSetting);

MinSignificantArea = Convert.ToInt32(Settings.MinSignificantAreaSetting);

MaxRectSize = Convert.ToInt32(Settings.MaxRectSizeSetting);

MinRectSize = Convert.ToInt32(Settings.MinRectSizeSetting);

VariableParameter = Convert.ToInt32(Settings.VariableParameterSetting);

}

private void SetSettingsValues(AppSettings appSettings)

{

appSettings.StandartExcelSavePathSetting = \_standartExcelSavePath;

appSettings.ThresholdValueSetting = \_thresholdValue.ToString();

appSettings.ImageInversionSetting = \_isGoingToBeInverted.ToString();

appSettings.MinSignificantAreaSetting = \_minSignificantArea.ToString();

appSettings.MinRectSizeSetting = \_minRectSize.ToString();

appSettings.MaxRectSizeSetting = \_maxRectSize.ToString();

appSettings.VariableParameterSetting = \_variableParameter.ToString();

}

// ViewModels графиков (отдельные user-control)

private RenieChartVM \_renieChartVM;

public RenieChartVM RenieChartVM

{

get => \_renieChartVM;

set => Set(ref \_renieChartVM, value);

}

private SpectraChartVM \_spectraChartVM;

public SpectraChartVM SpectraChartVM

{

get => \_spectraChartVM;

set => Set(ref \_spectraChartVM, value);

}

// Изображения

private BitmapImage \_uneditedImage;

public BitmapImage UneditedImage

{

get => \_uneditedImage;

set => Set(ref \_uneditedImage, value);

}

private BitmapImage \_editedImage;

public BitmapImage EditedImage

{

get => \_editedImage;

set => Set(ref \_editedImage, value);

}

// Изображение (путь/имя) для обработки

private List<string> \_imagesFullNames;

public List<string> ImagesFullNames

{

get => \_imagesFullNames;

set => Set(ref \_imagesFullNames, value);

}

private string \_imagesShortNames;

public string ImagesShortNames

{

get => \_imagesShortNames;

set => Set(ref \_imagesShortNames, value);

}

// Переменная - максимальный размер полей наименования изображения и пути сохранения отчетов

private int \_maxTextBoxSize;

public int MaxTextBoxSize

{

get => \_maxTextBoxSize;

set => Set(ref \_maxTextBoxSize, value);

}

// Стандартный путь для сохранения

private string \_standartExcelSavePath;

public string StandartExcelSavePath

{

get => \_standartExcelSavePath;

set => Set(ref \_standartExcelSavePath, value);

}

// Пороговое значение для преобразования изображения в черно-белый формат

private int \_thresholdValue;

public int ThresholdValue

{

get => \_thresholdValue;

set

{

if (value < 1)

{

Set(ref \_thresholdValue, 1); return;

}

if (value > 255)

{

Set(ref \_thresholdValue, 255); return;

}

Set(ref \_thresholdValue, value);

}

}

// Параметр для инвертирования ЧБ изображения

private bool \_isGoingToBeInverted;

public bool IsGoingToBeInverted

{

get => \_isGoingToBeInverted;

set => Set(ref \_isGoingToBeInverted, value);

}

// Минимальная площадь для распознавания неоднородности

private int \_minSignificantArea;

public int MinSignificantArea

{

get => \_minSignificantArea;

set

{

if (value < 1)

{

Set(ref \_minSignificantArea, 1); return;

}

if (value > 25)

{

Set(ref \_minSignificantArea, 25); return;

}

Set(ref \_minSignificantArea, value);

}

}

// Минимальный размер ячеек на которые разбивается изображение для расчета

private int \_minRectSize;

public int MinRectSize

{

get => \_minRectSize;

set

{

if (value >= MaxRectSize)

{

Set(ref \_minRectSize, MaxRectSize - 1); return;

}

if (value < 3)

{

Set(ref \_minRectSize, 3); return;

}

if (value > 14)

{

Set(ref \_minRectSize, 14); return;

}

Set(ref \_minRectSize, value);

}

}

// Максимальный размер ячеек на которые разбивается изображение для расчета

private int \_maxRectSize;

public int MaxRectSize

{

get => \_maxRectSize;

set

{

if (value <= MinRectSize)

{

Set(ref \_maxRectSize, MinRectSize + 1); return;

}

if (value < 4)

{

Set(ref \_maxRectSize, 4); return;

}

if (value > 15)

{

Set(ref \_maxRectSize, 15); return;

}

Set(ref \_maxRectSize, value);

}

}

// Варьируемый параметр для рассчетов

private int \_variableParameter;

public int VariableParameter

{

get => \_variableParameter;

set

{

if (value < 1)

{

Set(ref \_variableParameter, 1); return;

}

if (value > 100)

{

Set(ref \_variableParameter, 100); return;

}

Set(ref \_variableParameter, value);

}

}

// Массивы рассчитанных данных для сохранения

private double[] \_trueRenie;

public double[] TrueRenie

{

get => \_trueRenie;

set => \_trueRenie = value;

}

private double[] \_spectraIntermediateDiffs;

public double[] SpectraIntermediateDiffs

{

get => \_spectraIntermediateDiffs;

set => \_spectraIntermediateDiffs = value;

}

private double[] \_spectra;

public double[] Spectra

{

get => \_spectra;

set => \_spectra = value;

}

// Cписки массивов данных для сохранения, для загрузки в excelFile

private List<double[]> \_trueRenieArrays;

private List<double[]> \_spectraIntermediateDiffsArrays;

private List<double[]> \_spectraArrays;

// Данные расчета обобщенных спектров размерностей Реньи

// Значения массива для определенных значений варьируемого параметра

private double \_renieOn0;

public double RenieOn0

{

get => \_renieOn0;

set => Set(ref \_renieOn0, value);

}

private double \_renieOn1;

public double RenieOn1

{

get => \_renieOn1;

set => Set(ref \_renieOn1, value);

}

private double \_renieOn2;

public double RenieOn2

{

get => \_renieOn2;

set => Set(ref \_renieOn2, value);

}

private double \_renieOnMin;

public double RenieOnMin

{

get => \_renieOnMin;

set => Set(ref \_renieOnMin, value);

}

private double \_renieOnMax;

public double RenieOnMax

{

get => \_renieOnMax;

set => Set(ref \_renieOnMax, value);

}

// Переменная отвечающая за активность кнопки запуска

private bool \_processingBtnIsEnabled;

public bool ProcessingBtnIsEnabled

{

get => \_processingBtnIsEnabled;

set => Set(ref \_processingBtnIsEnabled, value);

}

// Переменная отвечающая за активность кнопок сохранения данных

private bool \_saveBtnIsEnabled;

public bool SaveBtnIsEnabled

{

get => \_saveBtnIsEnabled;

set => Set(ref \_saveBtnIsEnabled, value);

}

// Переменная отвечающая за отображение параметров

private bool \_settingsVisibility;

public bool SettingsVisibility

{

get => \_settingsVisibility;

set => Set(ref \_settingsVisibility, value);

}

private string \_settingsVisibilityBtnText;

public string SettingsVisibilityBtnText

{

get => \_settingsVisibilityBtnText;

set => Set(ref \_settingsVisibilityBtnText, value);

}

// Экземпляр класса отвечающего за взаимодействие с БД на сервере

DataBaseWorker \_dataBaseWorker;

// Логин и пароль

private string \_DBLogin;

public string DBLogin

{

get => \_DBLogin;

set => Set(ref \_DBLogin, value);

}

private string \_DBpassword;

public string DBPassword

{

get => \_DBpassword;

set => Set(ref \_DBpassword, value);

}

// Переменная отвечающая за отображение авторизации для БД

private bool \_loginVisibility;

public bool LoginVisibility

{

get => \_loginVisibility;

set => Set(ref \_loginVisibility, value);

}

// Переменная отвечающая за отображение кнопок взаимодействия с БД

private bool \_dataBaseButtonsVisibility;

public bool DataBaseButtonsVisibility

{

get => \_dataBaseButtonsVisibility;

set => Set(ref \_dataBaseButtonsVisibility, value);

}

// Команда выбора изображения

private RelayCommand \_pickImage;

public RelayCommand PickImage

{

get

{

return \_pickImage ?? new RelayCommand(obj =>

{

PickImageFile();

}

);

}

}

private void PickImageFile()

{

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Файл изображения (\*.jpg, \*.jpeg, \*.png, \*.bmp) | \*.jpg; \*.jpeg; \*.png; \*.bmp";

openFileDialog.FilterIndex = 1;

openFileDialog.Multiselect = true;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() != true)

{

return;

}

ImagesFullNames = openFileDialog.FileNames.ToList();

}

// Команда обработки изображения и расчетов

private RelayCommand \_runImageAnalysis;

public RelayCommand RunImageAnalysis

{

get

{

return \_runImageAnalysis ?? new RelayCommand(obj =>

{

RunAnalysis();

}

);

}

}

// Запуск цикла обработки

private void RunAnalysis()

{

\_trueRenieArrays = new List<double[]>();

\_spectraIntermediateDiffsArrays = new List<double[]>();

\_spectraArrays = new List<double[]>();

bool isFirst = true;

for (int i = 0; i < ImagesFullNames.Count; i++)

{

ImageProcessing(i, isFirst);

if (i == 0)

{

// Отображение данных первого изображения

DisplayData();

isFirst = false;

}

// Наполнение данными для создания Excel-файла

\_trueRenieArrays.Add(\_trueRenie);

\_spectraIntermediateDiffsArrays.Add(\_spectraIntermediateDiffs);

\_spectraArrays.Add(\_spectra);

}

// Штамп времени обработки (по окончанию обработки)

\_timeStamp = GetCurrentTime();

}

// Анализ одного изображения

private void ImageProcessing(int index, bool isFirst)

{

if (String.IsNullOrWhiteSpace(ImagesFullNames[index])) { return; }

// Создание матрицы изображения

Mat image = Cv2.ImRead(ImagesFullNames[index]);

// Получение данных о внешних контурах

List<List<OpenCvSharp.Point>> contours = GetOuterContours(image, ThresholdValue, minContourArea: MinSignificantArea);

// Создание матричного изображения для статистических расчетов

Mat imageNew = new Mat(image.Height, image.Width, MatType.CV\_8UC3, new Scalar(255, 255, 255));

Cv2.FillPoly(imageNew, contours, new Scalar(0, 0, 0));

if (IsGoingToBeInverted == true)

{

imageNew = ~imageNew;

}

// Статистические расчеты

double[,] statSumArray = RectsStatisticSumInRange(imageNew, MinRectSize, MaxRectSize, VariableParameter);

double[] tauArray = GetTauArray(statSumArray, VariableParameter, MinRectSize, MaxRectSize);

double[] reversedRenie = GetReversedRenie(tauArray, VariableParameter);

\_trueRenie = GetTrueRenie(reversedRenie);

SpectrumData spectraData = GetSpectraData(\_trueRenie, VariableParameter);

\_spectraIntermediateDiffs = spectraData.IntermediateDifferences;

\_spectra = spectraData.Spectrums;

// Отображение вариантов изображения до и после обработки (для первого изображения)

if (isFirst == true)

{

UneditedImage = MatToBitmapImage(image);

EditedImage = MatToBitmapImage(imageNew);

}

// Освобождение ресурсов

image.Dispose();

imageNew.Dispose();

}

// Отображения данных по текущему изображению

private void DisplayData()

{

// Отображение основных значений обобщенных спектров размерностей Реньи

RenieOn0 = \_trueRenie[(\_trueRenie.Length - 1) / 2];

RenieOn1 = \_trueRenie[((\_trueRenie.Length - 1) / 2) + 1];

RenieOn2 = \_trueRenie[((\_trueRenie.Length - 1) / 2) + 2];

RenieOnMin = \_trueRenie[0];

RenieOnMax = \_trueRenie[\_trueRenie.Length - 1];

// Передача значений для графиков

RenieChartVM.SetDataForChart(VariableParameter, \_trueRenie);

SpectraChartVM.SetDataForChart(\_spectraIntermediateDiffs, \_spectra);

// Активация кнопок сохранения отчета

SaveBtnIsEnabled = true;

}

// Команда сохранения данных в excel-файле

private RelayCommand \_saveToExcelFileStandartWay;

public RelayCommand SaveToExcelFileStandartWay

{

get

{

return \_saveToExcelFileStandartWay ?? new RelayCommand(obj =>

{

SaveToExcelStandartWay();

}

);

}

}

private void SaveToExcelStandartWay()

{

if (\_trueRenie == null || \_spectraIntermediateDiffs == null || \_spectra == null) { return; }

if (!Directory.Exists(\_standartExcelSavePath))

{

Directory.CreateDirectory(\_standartExcelSavePath);

}

string path = \_standartExcelSavePath + "\\" + \_timeStamp;

AppSettings settings = new AppSettings();

SetSettingsValues(settings);

CreateExcelFileWithExactWay(\_trueRenieArrays, \_spectraIntermediateDiffsArrays, \_spectraArrays, settings, path);

}

private RelayCommand \_saveToExcelFilePickWay;

public RelayCommand SaveToExcelFilePickWay{

get

{

return \_saveToExcelFilePickWay ?? new RelayCommand(obj =>

{

SaveToExcelPickWay();

}

);}}

private void SaveToExcelPickWay()

{

if (\_trueRenieArrays.Count == 0 || \_spectraIntermediateDiffsArrays.Count == 0 || \_spectraArrays.Count == 0) { return; }

string fileName = \_timeStamp;

AppSettings settings = new AppSettings();

SetSettingsValues(settings);

CreateExcelFile(\_trueRenieArrays, \_spectraIntermediateDiffsArrays, \_spectraArrays, settings, fileName);

}

// Команда изменения директории для сохранения файлов по умолчанию

private RelayCommand \_changeExcelFileStandartWay;

public RelayCommand ChangeExcelFileStandartWay{

get{

return \_changeExcelFileStandartWay ?? new RelayCommand(obj =>

{

PickExcelFileStandartWay();

}

);}}

private void PickExcelFileStandartWay()

{

var dialog = new SaveFileDialog();

dialog.InitialDirectory = \_standartExcelSavePath;

dialog.RestoreDirectory = true;

dialog.Title = "Выберите путь для сохранения";

dialog.Filter = "Путь|\*.this.directory"; // Для избежания отображения файлов

dialog.FileName = "Select"; // "select.this.directory"

if (dialog.ShowDialog() == true)

{

string path = dialog.FileName;

// Удаление "ненастоящей" части пути

path = path.Replace("\\Select.this.directory", "");

path = path.Replace(".this.directory", "");

// Если пользователь изменил имя файла, создание новой директории

if (Directory.Exists(path))

{

Directory.CreateDirectory(path);

}

StandartExcelSavePath = path;

}

}

// Команда открытия/скрытия параметров

private RelayCommand \_changeSettingsVisibility;

public RelayCommand ChangeSettingsVisibility

{

get

{

return \_changeSettingsVisibility ?? new RelayCommand(obj =>

{

ReverseSettingsVisibility();

}

);

}

}

private void ReverseSettingsVisibility()

{

SettingsVisibility = !SettingsVisibility;

if( \_settingsVisibility == true)

{

SettingsVisibilityBtnText = "Скрыть параметры";

}

if (\_settingsVisibility == false)

{

SettingsVisibilityBtnText = "Раскрыть параметры";

}

}

// Команда сохранения параметров приложения

private RelayCommand \_saveApplicationSettings;

public RelayCommand SaveApplicationSettings

{

get

{

return \_saveApplicationSettings ?? new RelayCommand(obj =>

{

SaveAppSettings();

}

);

}

}

private void SaveAppSettings(){

SetSettingsValues(Settings);

Settings.Save(\_defaultSettingsFilePath);

}

// Команда загрузки параметров из файла

private RelayCommand \_loadApplicationSettingsFromFile;

public RelayCommand LoadApplicationSettingsFromFile

{

get

{

return \_loadApplicationSettingsFromFile ?? new RelayCommand(obj =>

{

LoadAppSettingsFromFile();

}

);

}

}

private void LoadAppSettingsFromFile(){

OpenFileDialog openFileDialog = new OpenFileDialog();

openFileDialog.Filter = "Файл настроек (\*.xml) | \*.xml";

openFileDialog.FilterIndex = 1;

openFileDialog.RestoreDirectory = true;

if (openFileDialog.ShowDialog() != true)

{

ImagesFullNames = null;

return;

}

GetSettingsValuesFromFile(openFileDialog.FileName);

}

// Команда сохранения параметров в файл

private RelayCommand \_saveApplicationSettingsToFile;

public RelayCommand SaveApplicationSettingsToFile{

get{

return \_saveApplicationSettingsToFile ?? new RelayCommand(obj =>

{

SaveAppSettingsToFile();

}

);}

}

private void SaveAppSettingsToFile(){

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "Файлы настроек (\*.xml)|\*.xml|Все файлы (\*.\*)|\*.\*";

saveFileDialog.FilterIndex = 1;

saveFileDialog.RestoreDirectory = true;

saveFileDialog.FileName = \_settingsFilename;

if (saveFileDialog.ShowDialog() != true)

{

return;

}

SetSettingsValues(Settings);

Settings.Save(saveFileDialog.FileName);

}

// Команда подключения к БД

private RelayCommand \_authorizeToServerDB;

public RelayCommand AuthorizeToServerDB{

get{

return \_authorizeToServerDB ?? new RelayCommand(obj =>

{

InitializeDataBaseWorker();

}

);}}

private void InitializeDataBaseWorker()

{

DataBaseWorker dbw = new DataBaseWorker(DBLogin, DBPassword);

if (dbw.TestConnection() != true)

{

DBLogin = "Ошибка подключения";

DBPassword = "Попробуйте вновь";

return;

}

else

{

DataBaseButtonsVisibility = true;

LoginVisibility = false;

\_dataBaseWorker = dbw;

}

}

// Команда сохранения отображаемого анализа в БД

private RelayCommand \_saveCurrentDataToDB;

public RelayCommand SaveCurrentDataToDB

{

get{

return \_saveCurrentDataToDB ?? new RelayCommand(obj =>

{

SaveCurrentAnalysisToDB();

}

);

}}

private void SaveCurrentAnalysisToDB(){

\_dataBaseWorker.InsertUneditedImage(UneditedImage);

\_dataBaseWorker.InsertEditedImage(EditedImage);

AppSettings settings = new AppSettings();

SetSettingsValues(settings);

\_dataBaseWorker.InsertSettings(settings);

SerializableArrays sa = new SerializableArrays(\_trueRenieArrays[0], \_spectraIntermediateDiffsArrays[0], \_spectraArrays[0]);

string serializedArrays = sa.GetArraysSerialized();

\_dataBaseWorker.InsertMainData(\_timeStamp, serializedArrays);

}

// Команда открытия окна для просмотра сессий в БД

private RelayCommand \_openSessionTableView;

public RelayCommand OpenSessionTableView

{

get{

return \_openSessionTableView ?? new RelayCommand(obj =>

{

OpenSessionTable();

}

);}

}

private void OpenSessionTable()

{

var sessionTableVM = new SessionTableVM(\_dataBaseWorker.LoadSessionDataFromDataBase());

var sessionTableWindow = new SessionTableView(sessionTableVM);

SetWindowPostionAndOpen(sessionTableWindow);

}

// Вспомогательные методы //

private string GetCurrentTime(){

return DateTime.Now.ToString("dd-MM-yyyy\_HH-mm-ss");

}

// Метод открытия окон

private void SetWindowPostionAndOpen(System.Windows.Window window){

window.Owner = System.Windows.Application.Current.MainWindow;

window.WindowStartupLocation = WindowStartupLocation.CenterOwner;

window.ShowDialog();

}

}

}

**ПРИЛОЖЕНИЕ В**(обязательное) **Листинг скрипта базы данных**

**В.1 Листинг скрипта базы данных**

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

-- База данных: `sql7622483`

-- Создание таблиц

CREATE TABLE `EditedImage` (

`ei\_id` int(11) NOT NULL,

`ei\_image` longblob NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `ProcessingData` (

`pd\_id` int(11) NOT NULL,

`pd\_time` datetime NOT NULL,

`ps\_id` int(11) NOT NULL,

`ui\_id` int(11) NOT NULL,

`ei\_id` int(11) NOT NULL,

`pd\_arrays` mediumtext NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `ProcessingSettings` (

`ps\_id` int(11) NOT NULL,

`ps\_thresholdValue` smallint(6) NOT NULL,

`ps\_imageInversion` tinyint(1) NOT NULL,

`ps\_minSignificantArea` tinyint(4) NOT NULL,

`ps\_minRectSize` tinyint(4) NOT NULL,

`ps\_maxRectSize` tinyint(4) NOT NULL,

`ps\_variableParameter` tinyint(4) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE `UneditedImage` (

`ui\_id` int(11) NOT NULL,

`ui\_image` longblob NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

-- Индексы сохранённых таблиц

ALTER TABLE `EditedImage`

ADD PRIMARY KEY (`ei\_id`);

ALTER TABLE `ProcessingData`

ADD PRIMARY KEY (`pd\_id`);

ALTER TABLE `ProcessingSettings`

ADD PRIMARY KEY (`ps\_id`);

ALTER TABLE `UneditedImage`

ADD PRIMARY KEY (`ui\_id`);

-- AUTO\_INCREMENT для сохранённых таблиц

ALTER TABLE `EditedImage`

MODIFY `ei\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `ProcessingData`

MODIFY `pd\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `ProcessingSettings`

MODIFY `ps\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1;

ALTER TABLE `UneditedImage`

MODIFY `ui\_id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=1; COMMIT;