МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г.ЧЕРНЫШЕВСОКГО»**

Кафедра Математического и компьютерного моделирования

# Автоматизация постобработки данных в CAD/CAE системах

МАГИСТЕРСКАЯ РАБОТА

студента 2 курса 247 группы

направления 09.04.03 — Прикладная информатика

механико-математического факультета

Иванова Дмитрия Алексеевича

Научный руководитель
доцент, к.т.н., доцент И. А. Панкратов

Зав. кафедрой
зав.каф., д.ф.-м.н., доцент Ю. А. Блинков

Саратов 2021

# СОДЕРЖАНИЕ

Стр.

# ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

CAD (от англ. Computer-aided design) – автоматизированная система, реализующая информационную технологию выполнения функций проектирования;

CAE (от англ. Computer-aided engineering) – общее название для программ и программных пакетов, предназначенных для решения различных инженерных задач;

CFD (от англ. Computational fluid dynamics) – вычислительная гидродинамика;

БД – база данных;

СУБД – система управления базами данных;

виджет – примитив графического интерфейса пользователя;

ИС – информационная система.

# ВВЕДЕНИЕ

При обработке экспериментальных данных и расчетов, полученных в результате математического моделирования физических процессов в CAD/CAE системах, особенно, когда проводится, например, серия экспериментов, в которых входные данные незначительно изменяются часто порождается большой объем результатов, подлежащих анализу или иными словами – постобработке. Причем, зачастую для анализа необходимо провести однотипные манипуляции с полученными мало различающимися данным. Учитывая все вышесказанное, становится ясна необходимость автоматизации такого процесса постобработки данных.

В магистерской работе была спроектирована и разработана ИС позволяющая упростить процесс анализа полученных экспериментальных данных. Для автоматизации был использован язык программирования Python, для хранения результатов – NoSQL подход, а конкретно СУБД MongoDB.

Таким образом целью данной магистерской работы является проектирование и разработка информационной системы, автоматизирующей рутинные операции анализа экспериментальных данных. Задачи:

- Кратко рассмотреть платформу для численного моделирования OpenFOAM и пакет для визуализации ParaView.
- Рассмотреть существующие решения по данной тематике.
- Спроектировать информационную систему и создать UML-диаграммы для ее описания.
- Провести обзор средств разработки.
- Разработать помогающую в анализе экспериментальных данных информационную систему.

# 1 Об OpenFOAM и ParaView

## 1.1 OpenFOAM

OpenFOAM (англ. Open Source Field Operation And Manipulation CFD ToolBox) — открытая интегрируемая платформа для численного моделирования задач механики сплошных сред [1].

Это пакет программ распространяемых свободно под лицензией GNU GPL, позволяющей решать задачи механики сплошных сред, в частности:

- Прочностные расчеты;
- Гидродинамика ньютоновских и неньютоновских вязких жидкостей как в несжимаемом,
- так и сжимаемом приближении с учётом конвективного теплообмена и действием сил гравитации. Для моделирования турбулентных течений возможно использование RANS-моделей, LES- и DNS-методов. Возможно решение дозвуковых, околозвуковых и сверхзвуковых задач; Задачи теплопроводности в твёрдом теле;
- Многофазные задачи, в том числе с описанием химических реакций компонент потока;
- Задачи, связанные с деформацией расчётной сетки;
- Сопряжённые задачи;
- Некоторые другие задачи, при математической постановке которых требуется решение дифференциальных уравнений в частных производных в условиях сложной геометрии среды;

В основе кода лежит набор библиотек, предоставляющих инструменты для решения систем дифференциальных уравнений в частных производных как в пространстве, так и во времени. Рабочим языком кода является C++. OpenFOAM состоит из приблизительно 250 программ основанных на более чем 100 библиотеках. Каждое приложения выполняет свою конкретную задачу в рамках процесса расчета. Этапы работы представленные в соответствии с рисунком 1.1.

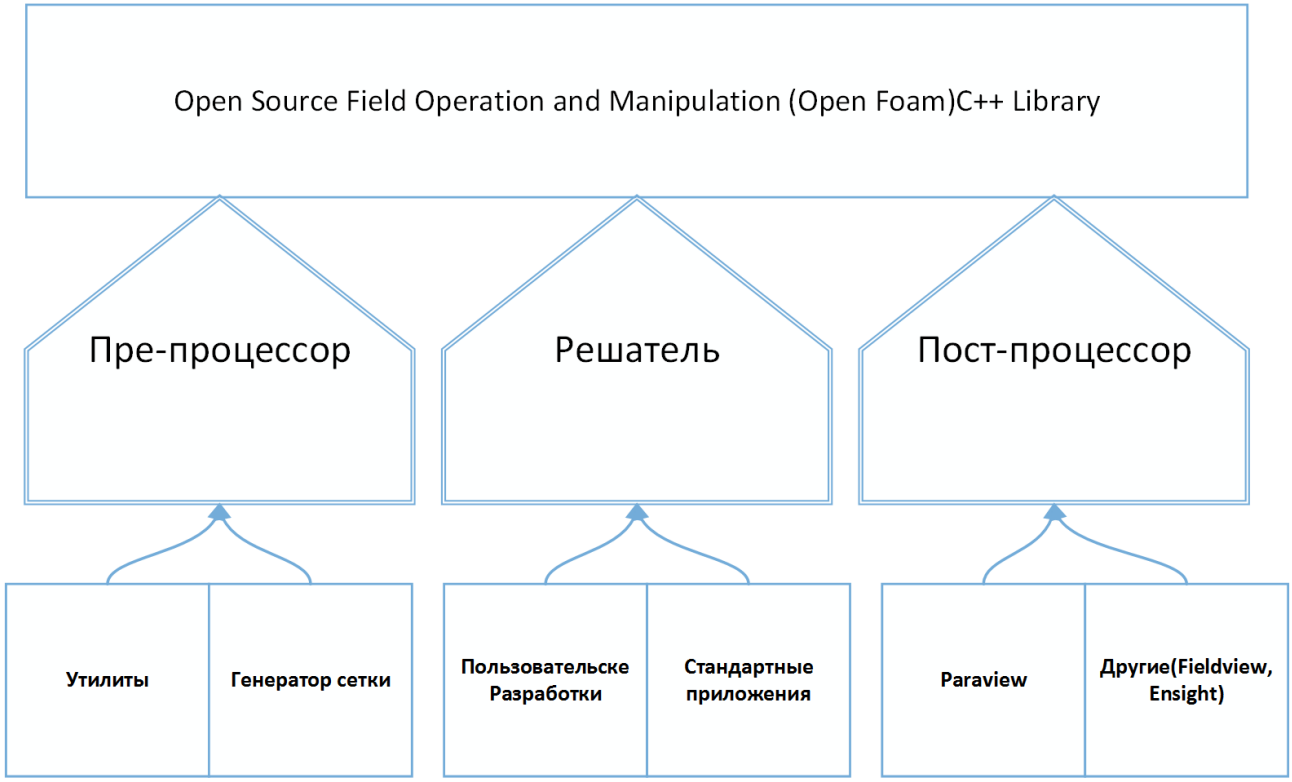


Рисунок 1.1 — Утилиты и программы входящие в пакет OpenFOAM, сгруппированные по этапам работы с расчетом

Работа с программой делится на три этапа:

1. Препроцессинг;
2. Решение;
3. Постпроцессинг.

На этапе препроцессинга в специальных файлах задаются входные данные для рассчета примера, такие как: начальное время, конечное, шаг и так далее. Также параметры для хранения решения: время, формат, тип сжатия. Также в препроцессинг включены настройки выбора различных схем расчета, которые влияют на точность и стабильность решения. После этого отдельно генерируется расчетная область (сетка), которая впоследствии может быть обработана различными утилитами [2]. Затем запускается решатель, который производит расчет. На этапе постпроцессинга полученные данные представляются в виде графиков. Также используются некоторые утилиты, например для конвертации из внутреннего формата OpenFOAM в широко используемый формат vtk.

## 1.2 ParaView

ParaView – открытый графический кросс-платформенный пакет для интерактивной визуализации в исследовательских целях, разрабатываемый Национальной Лабораторией Сандиа, компанией Kitware и Национальной Лабораторией Лос-Аламоса [3].

Пакет ParaView предоставляет пользователю возможности интерактивной визуализации и исследования больших массивов данных для качественного и количественного анализа.

Пакет может быть использован на компьютерах с операционными системами Windows, Linux, Mac OS X.

При разработке авторы придерживаются следующих целей:

- Открытость, кросс-платформенность — в пакете используются только открытые, мульти-платформенные технологии для визуализации данных.
- Поддержка различных, в том числе, гетерогенных вычислительных систем.
- Создание гибкого, интуитивного пользовательского интерфейса.

Таким образом, пакет ParaView во многом является скорее технологией обработки, чем всего лишь программным средством [4].

Некоторые возможности пакета:

- Визуализация расчетных областей.
- Визуализация полей (давление, скорость, температура, смещения и прочее).
- Построение срезов областей как плоскостью, так и заданной функцией.
- Построение изо-поверхностей.
- Построение векторных полей и линий тока.
- Позволяет показывать динамику развития протекающего процесса, отображая анимацию.

Основной формат данных ParaView – VTK, но пакет также содержит драйверы для работы с форматом OpenFOAM и поставляется вместе с дистрибутивом пакета.

Работа с ParaView может осуществляться как в интерактивном, так и пакетном режиме.

ParaView также предлагает богатый и мощный програмный интерфейс на языке Python. Это позволяет пользователям автоматизировать обработку своих данных и использовать возможности, так называемого, набора инструментов визуализации – Visualization Tool Kit (VTK) [5].

## 2 Обзор существующих решений

Информационная система должна выполнять постобработку данных, полученных в результате численно эксперимента в пакете OpenFOAM, делая упор на автоматизацию функций для работы с серией данных. Рассмотрим доступные приложения осуществляющие автоматизацию рутинных функций в рамках пакета OpenFOAM.

### 2.1 HELYX OS

HELYX-OS - это графический пользовательский интерфейс с открытым исходным кодом, разработанный компанией ENGYS для работы со стандартными библиотеками OpenFOAM, предоставляемыми OpenFOAM Foundation и ESI-OpenCFD. Приложение предназначено для академического использования и работы с CFD начального уровня. Распространяется в соответствии с GNU General Public License [6].

HELYX-OS предоставляет полностью интерактивную, простую в использовании среду для выполнения всех задач предварительной обработки в процессе CFD, включая создание сетки, определение случая и выполнение решателя.

Существует также версия для корпоративного использования – CFD HELYX.

Преимущества:

- Встроенная поддержка как OpenFOAM, так и OpenFOAM+: возможность загружать существующие примеры, читая настройки непосредственно из доступных текстовых файлов проекта.
- Программа доступна на платформах Linux и Windows. Однако версия для Windows платна.
- Управление утилитой построения сеток snappyHexMesh, включая такие возможности как отображение геометрии и непосредственное построение прямо в окне приложения.
- Отдельный мониторинг решателя с отслеживанием остатков решения.

В корпоративной версии также следует выделить:

- Высокая масштабируемость.

- Возможность работы с использованием облачных технологий.
- Модульность. Возможно расширение в рамках HELYX ADD-ONS,

В соответствии с рисунком 2.1 изображен рабочий экран программы.

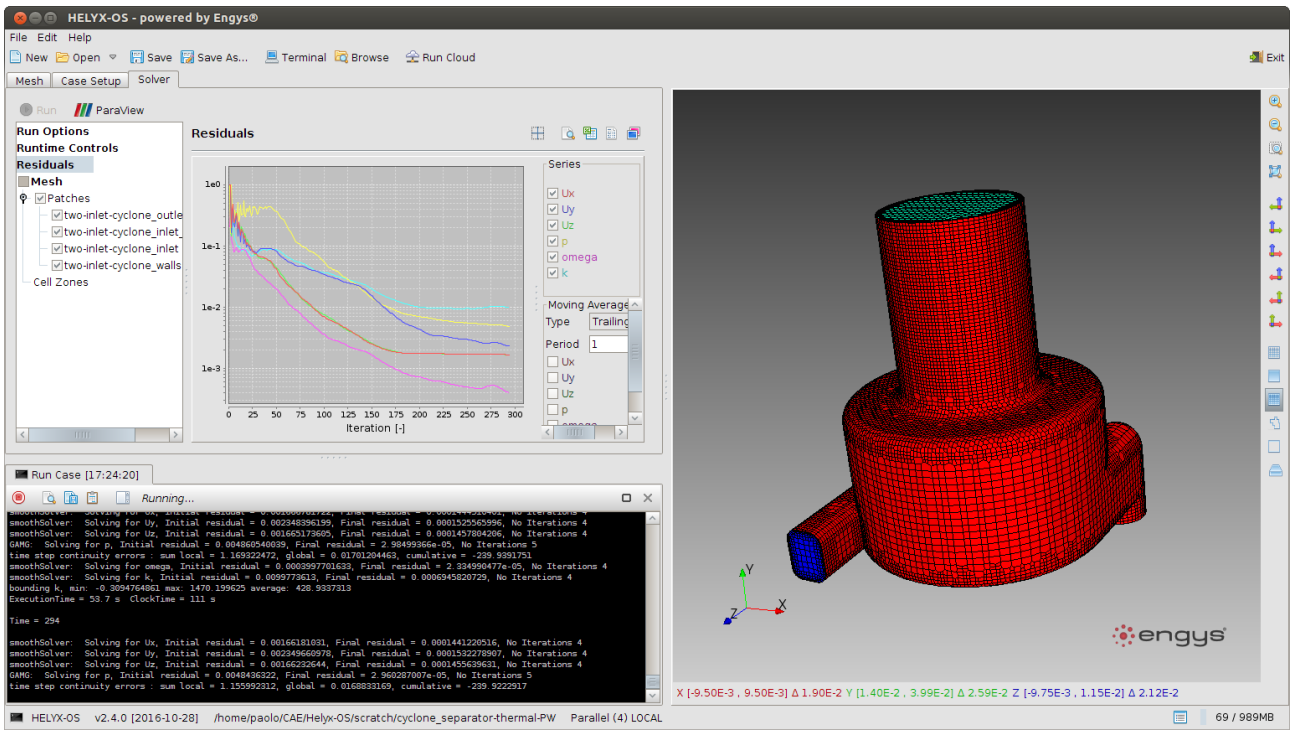


Рисунок 2.1 — Рабочий экран приложения Helyx OS

## 2.2 ANSA

ANSA - это инструмент препроцессинга CAE, который предоставляет все необходимые функциональные возможности для построения полной модели, от CAD-данных до готового к вводу файла решателя, в единой интегрированной среде [7].

Все функции программного обеспечения размещены в интегрированной среде с настраиваемым графическим интерфейсом. Программное обеспечение доступно для всех современных популярных операционных систем в 32-битной и 64-битной архитектуре с использованием многоядерных процессоров.

Преимущества:

- Эффективная обработка данных для сложных структур моделей.

- Быстрое и качественное моделирование сложных геометрических моделей.
- Возможность взаимодействия между моделями, созданными для разных решателей.
- Высокоавтоматизированные процессы и инструменты настройки модели в одной программе.
- Уменьшены зависящие от пользователя подверженные ошибкам операции.
- Полное построение модели для многочисленных решателей в одной среде.

Рабочий экран приложения представлен в соответствии с рисунком 2.2.

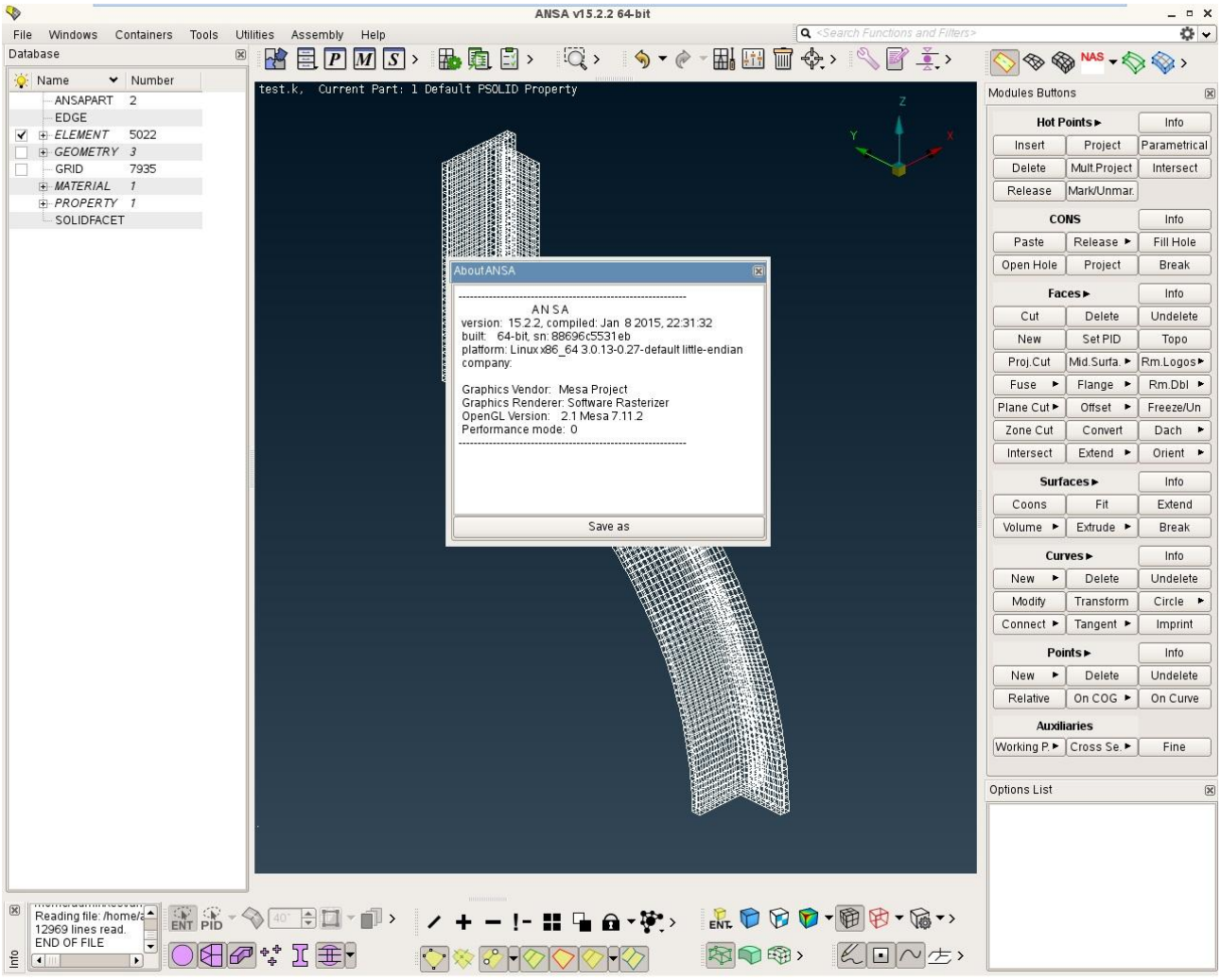


Рисунок 2.2 — Рабочий экран приложения ANSA

## 2.3 CastNet

CastNet упрощает использование технологических решений CAE для решателей с открытым исходным кодом: кроме типичного редактирования текстовых файлов, предоставляется альтернативный способ работы с OpenFOAM на основе графического интерфейса, сохраняя полную совместимость со стандартными выпусками OpenFOAM. В результате рабочий процесс становится достаточно гибким, и пользователь может в любой момент переключаться между настройкой рабочего примера на основе текстового файла и графического интерфейса пользователя.

Вид рабочего экрана приложения представлен в соответствии с рисунком 2.3

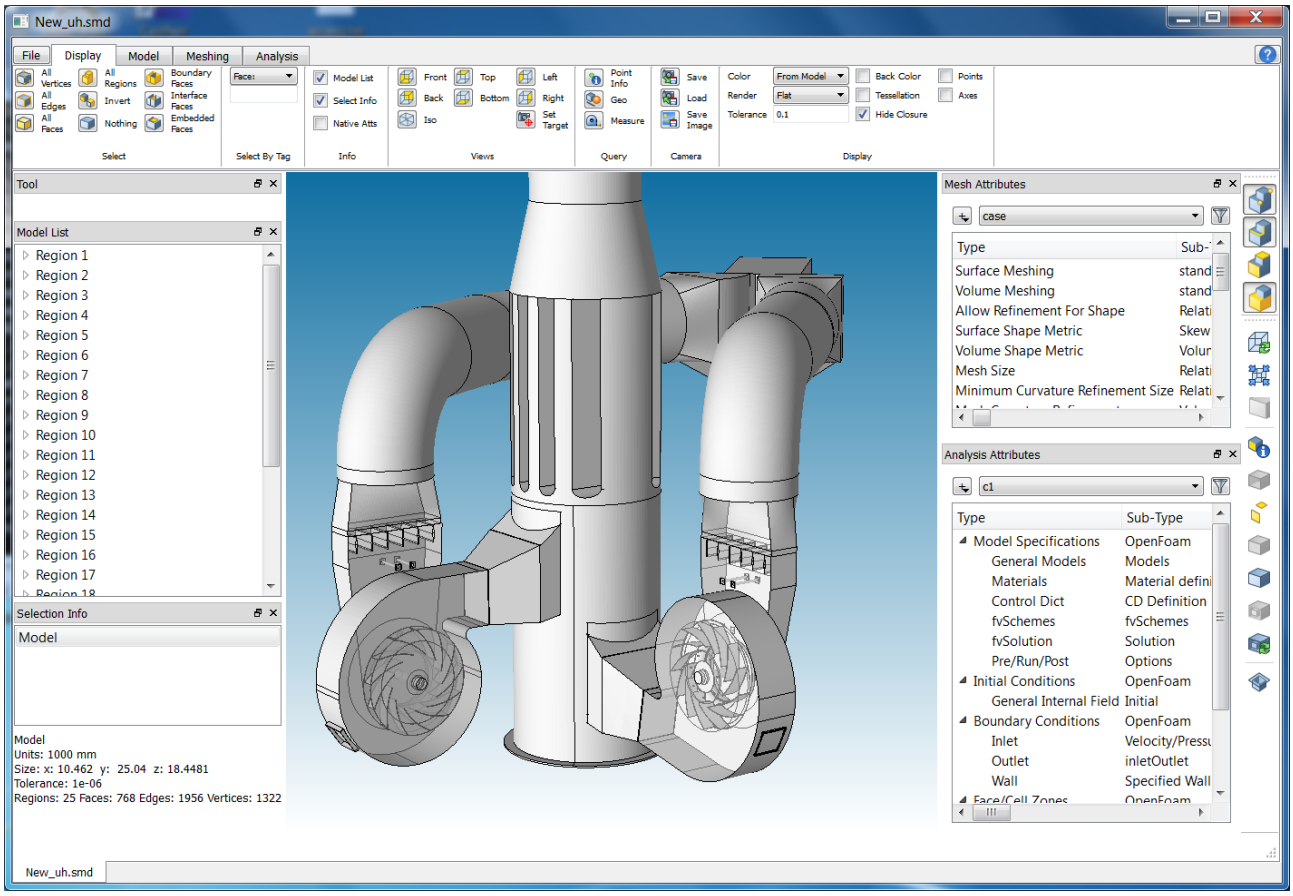


Рисунок 2.3 — Рабочий экран приложения CastNet

Ключевые особенности CastNet:

- Среда разработки на основе графического интерфейса пользователя, включающая предварительную обработку (создание сетки, настройку

примера), мониторинг решения и последующую обработку. Таким образом: доступ к мощным функциям решателя с открытым исходным кодом без редактирования текстовых файлов или необходимости детального изучения структуры ключевых слов OpenFOAM.

- Кроссплатформенное использование: поддержка среды для пакета программ OpenFOAM в операционных системах Windows и Linux.
- Библиотека шаблонов позволяет настраивать пример для более чем 30 решателей.
- Больше надежности в отношении результатов моделирования благодаря контролю сходимости.

## 2.4 Выводы

Таким образом, рассмотренные существующие решения предлагают разнообразные и гибкие возможности по работе с примерами:

- CastNet интегрирована с OpenFoam, имеет графический интерфейс, но работает только одним конкретным примером.
- Ansa – мощный инструмент препроцессинга. Имеет высокоавтоматизированные процессы и инструменты настройки модели в одной программе. Возможность быстро и качественно моделировать сложных геометрических модели, но ничего не сказано о какой-либо работе с группой моделей.
- HELYX-OS имеет встроенную поддержку OpenFOAM, предоставляет возможности отображения геометрии и непосредственного построения примера прямо в окне, но также не концентрируется на работе с набором примеров.

  Однако, ни одно из них не предоставляет функциональности по работе сразу с группой примеров, что в свою очередь вызывает трудности при анализе экспериментальных данных, которые состоят из набора примеров.

# 3 Проектирование информационной системы

## 3.1 Постановка задачи

Необходимо спроектировать приложение, которое бы выполняло процесс постобработки, то есть строило графики, используя экспериментальные данные, полученные из пакета программ OpenFOAM. Приложение должно также работать с группами экспериментальных данных, то есть выполнять конкретное действие построения графика, например срез, с группой из разных мало отличающихся примеров. Программа должна хранить экспериментальные данные и историю операций примера, также должна быть реализована возможность экспорта графиков в файлы. Для более подробного понимания информационной системы были построены UML-диаграммы.

## 3.2 Диаграмма прецедентов

Прецеденты – это технология определения функциональных требований к системе [9]. Диаграмма прецедентов (use case diagram) предназначена для описания взаимодействия проектируемой системы с любыми внешними или внутренними объектами - пользователями, другими системами и тому подобное. Основными понятиями при работе с диаграммой вариантов использования являются Актор (Actor) – это роль, которую выполняет пользователь или другая система, при взаимодействии с проектируемой системой. Вариант использования – это конечная единица взаимодействия актора и системы.

Диаграмма вариантов использования представлена в соответствии с рисунком 3.1

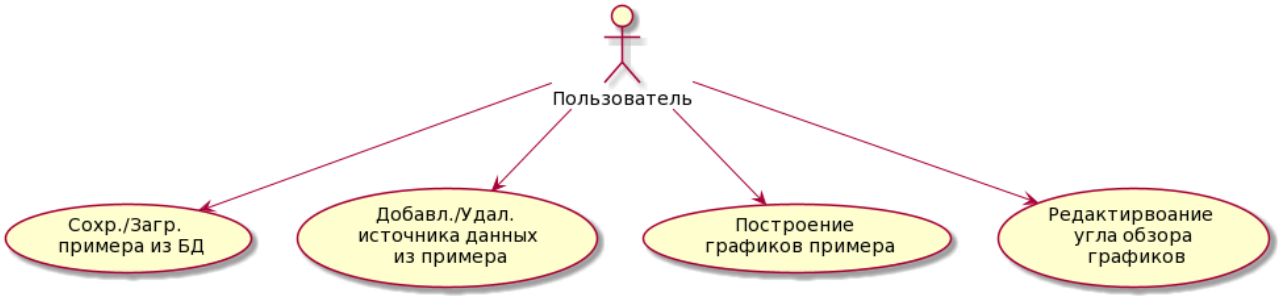


Рисунок 3.1 — Диаграмма прецедентов

Ниже представлен код генерации диаграммы прецедентов при помощи инструмента PlantUML.

```
@startuml
actor "Пользователь" as user
user-->(Сохр./Загр. \n примера из БД)
user-->(Добавл./Удал. \n источника данных \n из примера)
user-->(Построение \n графиков примера)
user-->(Редактирвоание \n угла обзора \n графиков)
@enduml
```

В соответствии с рисунком 3.1 представлена базовая функциональность проектируемой программы. Пример или основная сущность программы состоит из списка источников данных. Каждый источник данных – это результат конкретного численного эксперимента. Таким образом достигается цель – работа сразу с несколькими источниками данных.

В случае описанной программы, встречаются следующие прецеденты:

- Сохранение и загрузка примера из базы данных. Работа с программой строится вокруг анализа расчетов полученных от пакета программ OpenFOAM, конфигурации данных должны сохраняться в БД для обеспечения сохранности информации и быстрого доступа к ней, при возрастающим объеме примеров.
- Добавление и удаление источника данных из примера. Также необходимая функция для возможности быстрой и гибкой настройки примеров для получения различных срезов и углов обзора графиков.
- Редактирование угла обзора графиков – изменение положения камеры в соответствии с необходимостью анализа данных.
- Построение графиков примера – получение изображение в соответствии с описанными выше настройками примера.

## 3.3 Диаграмма классов

Диаграмма классов описывает типы объектов системы и различного рода статические отношения, которые существуют между ними. На диаграммах

классов отображаются также свойства классов, операции классов и ограничения, которые накладываются на связи между объектами [9].

Для удобства рассмотрения разобьем диаграмму классов на три рисунка. Каждый из которых соответствует определенной решаемой задачи.

### 3.3.1 Диаграмма модели данных

В соответствии с рисунком 3.2 представленная диаграмма классов, иллюстрирует способ организации модели данных хранящих кейс для генерации данных. Кейс или пример – основная сущность программы. В ней хранится текущее частичное состояние. Модель состоит из набора связанных сущностей, модели хранятся в списке – который описывает полное состояние системы.

### 3.3.2 Диаграмма, представляющая способ организации экспериментальных данных

В соответствии с рисунком 3.2 представленная диаграмма классов иллюстрирует способ организации экспериментальных данных в приложении.

В соответствии с рисунком 3.2 основным классом модели является FoamCase. В нем хранится все необходимое для генерации различных графиков. Соответственно его поля – это параметры генерации:

- CasesDir – класс, который хранит путь к примеру. Пример это директория, содержащая один или несколько кейсов OpenFOAM.

  Код для генерации класса CaseDir:

```
1         class CasesDir{
2                 +idn: int
3                 +path: Path
4         }
```

- CameraProps – класс, который содержит настройки камеры для построения графика.

```
1         class CameraProps{
2                 +idn: int
```

```
        +name: str
        +focal_point: Point
        +cam_position: Point
        +viewup: Point
        +viewangle: float
    }
```

- Point – утилитарный класс, необходимый для представления данных с точки зрения позиции камеры для обзора.

```
    class Point{
        +x: float
        +y: float
        +z: float
    }
```

- SharedState – Класс хранящий общую информацию для контролера и представления. Необходим пересылки общей универсальной информации высокого уровня между виджетами разных уровней отображения. В этом классе отдельно выделено важное поле case_list – оно хранит все текущие «живые», иными словами доступные в бд и не удаленные пользователем, экземпляры класса FoamCase.

```
        class FoamCase{
            +idn: int
            +name: str
            +cases_dir: CaseDir
            +cam_prop_list: List[CameraoProps]
        }
        class Controller {}
        hide Controller members
        class SharedState {
            +case_list: List[FoamCase]
            +state1
            +state2
            + ...
        }
```

А теперь необходимо установить связи между классами диаграммы:

```
Controller -- FoamCase
SharedState --o Controller
```

```
3  Point -- CameraProps
4  FoamCase "*" --o "1" SharedState
5  CasesDir "1" --* "1" FoamCase
6  CameraProps "*" --* "1" FoamCase
```

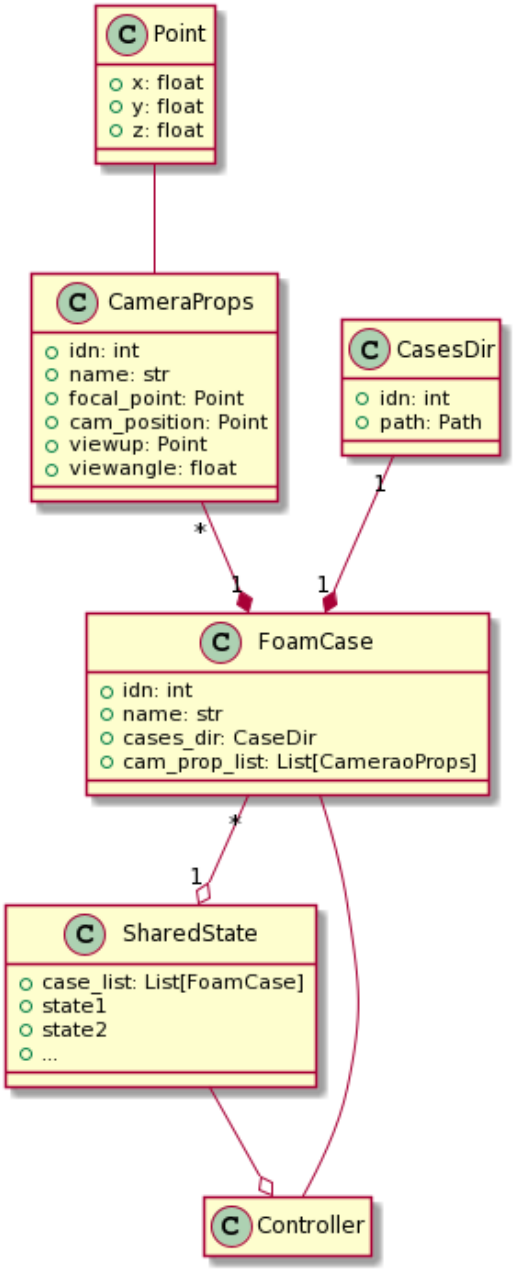


Рисунок 3.2 — Диаграмма классов, представляющая способ организации данных в приложении

### 3.3.3 Диаграмма для работы с базой данных. Слой DAO

В соответствии с рисунком 3.3 представленная диаграмма классов, иллюстрирует структуру DAO слоя (Data Access Object). Этот слой необходим для получения различных сущностей из базы данных, абстрагируясь от того какая конкретно СУБД используется. Ход генерации данной диаграммы, как и разработку данных классов в приложении можно разбить на несколько этапов:

I. Построение абстрактных классов фабрики классов DAO. Про паттерн фабрика будет подробнее сказано чуть ниже. Нужно отметить, что абстрактные классы, учитывая множественное наследование в Python выполняю и в некотором смысле роль интерфейсов. Ниже приведен код генерации абстрактного класса фабрики:

```
1         abstract class DaoFactory{
2                 - DBConnection _get_connection()
3                 + AbstractDAO get_dao(connection, dao_class)
4         }
```

II. Следующий этап – это расширение абстракции до конкретной фабрики DAO компонентов БД MongoDB:

```
1         class MongoDaoFactory {
2                 - LOGIN
3                 - PASSWORD
4                 - MongoDBConnection MongoDBConnection _get_connection()
5                 + MongoAbstractDAO MongoDget_dao(connection, dao_class)
6         }
```

III. На этом этапе строиться абстрактные класс для DAO компоненты (выступает и в качестве интерфейса). Но также и абстрактный класс для MongoDB, созданный для того, чтобы избежать дублирования кода в при написании программы. Код генерации представлен ниже:

```
1         abstract class AbstractDAO{
2                 +connection
3                 +Any create(obj)
4                 +Any create_or_update(obj)
5                 +Any read(key)
```

```
6                 +Any update(obj)
7                 +Any delete(key)
8                 +List[Any] get_all()
9         }
10
11        abstract class MongoAbstractDAO{
12                +Any delete(key): implemented
13                +Any get_all(): implemented
14        }
```

IV. Далее идет генерация уже конкретных классов DAO базы данных MongoDB:

```
1         class MongoFoamCaseDAO{}
2         hide MongoFoamCaseDAO members
3
4         class MongoCaseDirDAO{}
5         hide MongoCaseDirDAO members
6
7         class MongoCameraPropsDAO{}
8         hide MongoCameraPropsDAO members
```

V. На последней этапе были сгенерированы классы из сторонних модулей, ассоциациированные с DAO. Код представлен ниже:

```
1         class Config {}
2         hide Config members
3
4
5         class Controller {}
6         hide Controller members
7
8         DaoFactory <|-- MongoDaoFactory
9         Config ..> MongoDaoFactory: inject LOGIN, PASSWORD
10        Controller -- DaoFactory
11        Controller -- AbstractDAO
12        DaoFactory --> AbstractDAO
13        AbstractDAO <|-- MongoAbstractDAO
14        MongoAbstractDAO <|--MongoFoamCaseDAO
15        MongoAbstractDAO <|--MongoCaseDirDAO
16        MongoAbstractDAO <|--MongoCameraPropsDAO
```

При проектирование также необходимо предусмотреть неуказанный в диаграмме класс DTO (Data Transfer Object). Он используется как объект в котором хранятся данные модели сущности при передаче в БД или внешний сервис. Это необходимо для того, чтобы при изменении стороннего модуля или представления в базе данных локализовать изменения, которые необходимо внести в разрабатываемой программе, в одном модуле.

Также был разработан отдельный модуль modelmapper. В нем единственный класс Mapper отвечает за преобразования объектов модели в их DTO-аналог и наоборот. Таким образом при возникновении изменений на стороне необходимо будет изменить соответствующий DTO-класс, а также его метод в классе Mapper.

Для установления соединения и работы с базой данных используется класс из PyMongo, не отраженный на диаграмме, также для доступа к базе данных используется класс Config, который передает логин, пароль и дополнительную специфичную для конкретного соединения информацию. Класс Config (как и предоставляющий соединение с базой данных класс из PyMongo) реализован при помощи шаблона проектирования синглтон (Singleton). Данный паттерн гарантирует, что у класса есть только один экземпляр, и предоставляет к нему глобальную точку доступа [10]. Такой подход дает позволяет только один раз при первом обращении произвести ресурсоемкую операцию установления соединения с базой данных или чтения config-файла, а после каждый раз при последующих обращениях возвращать уже созданный экземпляр класса.

Так как имеется три класса DAO, а вообще, их может быть и больше, был реализован паттерн фабрика. Фабричный метод — это порождающий паттерн проектирования, который определяет общий интерфейс для создания объектов в суперклассе, позволяя подклассам изменять тип создаваемых объектов [11].

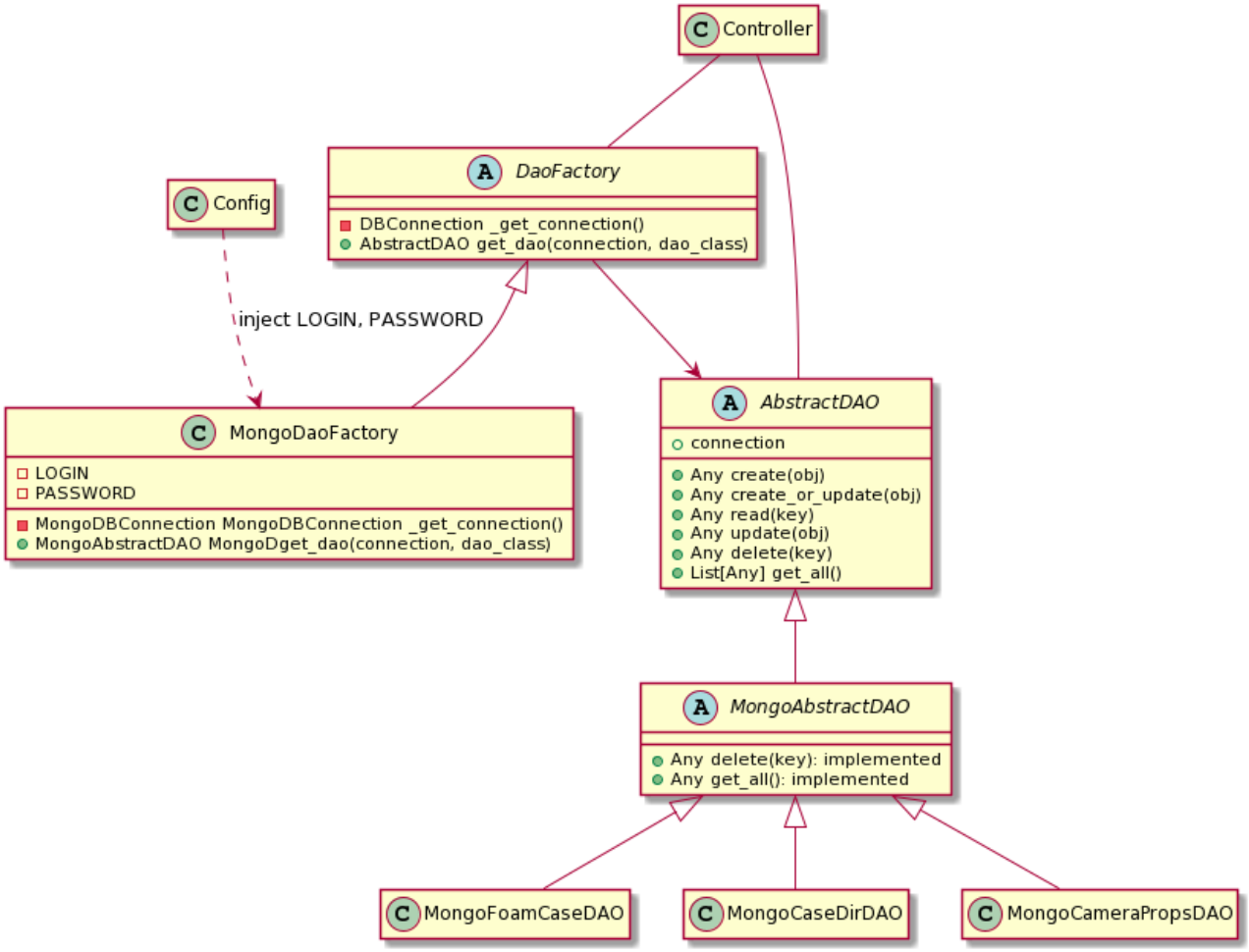


Рисунок 3.3 — Диаграмма классов. Слой DAO и его фабрика

Использование этого шаблона дает следующие преимущества:

- Избавляет класс от привязки к конкретным классам продуктов.
- Выделяет код производства продуктов в одно место, упрощая поддержку кода.
- Упрощает добавление новых продуктов в программу.
- Реализует принцип открытости/закрытости.

К недостаткам можно отнести возможность создания больших параллельных иерархий классов, так как для каждого класса продукта надо создать свой подкласс создателя.

Здесь же опишем подробнее класс Config. В нем, помимо одиночки, был реализован паттерн заместитель. Заместитель — это структурный паттерн проектирования, который позволяет подставлять вместо реальных объектов специальные объекты-заменители. Эти объекты перехватывают вызовы

к оригинальному объекту, позволяя сделать что-то до или после передачи вызова оригиналу [12].

При использовании библиотеки ConfigParser есть возможность для удобства пользователя разделять параметры по разделам. Поэтому удобно возвращать вместо конечных данных класс ConfigProxy, который в зависимости от раздела будет каким-то образом предобрабатывать данные. Например, приводить к нужному типу, или добавлять префикс к возвращаемому пути папки для сохранения данных.

### 3.3.4 Диаграмма классов для представлений (View)

Здесь необходимо упомянуть что все классы кроме Controller наследуются от QWidget. И по своей сути это инструкции как собрать графический интерфейс пользователя. В дополнение к этому в этих классах происходит привязка функций обратного вызова (callback) контроллера к отображению через конструктор классов отображения. Этот факт отражен да диаграмме в виде ассоциаций.

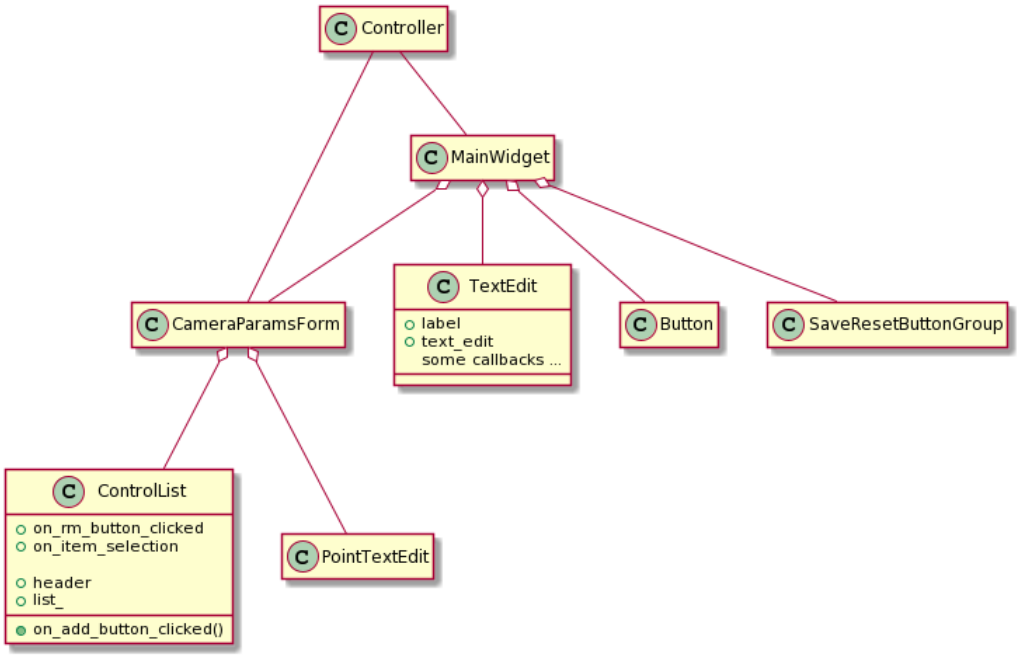


Рисунок 3.4 — Диаграмма классов. Отображение (View)

В один из обработчиков событий кнопок «на нажатие» в Controller классе включен вызов метода takeScreenshot класса Screenshot. Этот метод обращается к Paraview API и генерирует график по заданным параметрам.

Далее приведен код генерации данной диаграммы:

```
@startuml
class ControlList {
        +on_add_button_clicked()
        +on_rm_button_clicked
        +on_item_selection
        +header
        +list_
class TextEdit {
        + label
        + text_edit
        some callbacks ...
}
class PointTextEdit{}
hide PointTextEdit members
class CameraParamsForm {}
hide CameraParamsForm members
class Button {}
hide Button members
class SaveResetButtonGroup {}
hide SaveResetButtonGroup members
class MainWidget {}
hide MainWidget members
class Controller {}
hide Controller members

MainWidget o-- SaveResetButtonGroup
MainWidget o-- Button
MainWidget o-- TextEdit
MainWidget o-- CameraParamsForm
CameraParamsForm o-- PointTextEdit
CameraParamsForm o-- ControlList
Controller -- MainWidget
Controller -- CameraParamsForm
@enduml
```

## 3.4 Диаграмма последовательностей

Диаграммы последовательностей или взаимодействия (interaction diagrams) описывают взаимодействие групп объектов в различных условиях их поведения. UML определяет диаграммы взаимодействия нескольких типов, из которых наиболее употребительными являются диаграммы последовательности (sequence diagram) [9].

На диаграмме последовательностей отображаются системные события для одного сценария некоторого прецедента. Поэтому сама диаграмма строится на основе описания прецедента [13].

Если прецедент отвечает на вопрос «Что делает актор?», то последовательность отвечает на вопрос «Как работает система при выполнении данного прецедента?». Каждый прецедент может содержать несколько диаграмм последовательностей, на тот случай, если они описывают несколько альтернативных вариантов развития событий.

Диаграмма последовательностей будет построена только для прецедента «Построение графиков примера», «Сохранение примера в базу данных», «Добавление источника данных».

Ниже представлен код генерации диаграммы последовательностей для прецедента «Построение графиков примера»:

```
1  @startuml
2  participant Пользователь
3
4  Пользователь -> ":Controller" : построить графики
5  create ":Screenshot"
6  activate ":Controller"
7  ":Controller" -> ":Screenshot" : обратиться к классу
8  ":Controller" -> ":Screenshot" : take_screenshot(cases_list, output)
9  activate ":Screenshot"
10 deactivate ":Controller"
11 ":Screenshot" -> ":FoamCase" : get_data()
12 activate ":FoamCase"
13 ":FoamCase" --> ":Screenshot" : вернуть data
14 deactivate ":FoamCase"
15 ":Screenshot" -> ":ParaView" : set_data()
```

```
activate ":ParaView"
":Screenshot" -> ":ParaView" : show()
":ParaView" -> ":Screenshot" : вернуть графики
":Screenshot" -> ":ParaView": сохранить графики
deactivate ":Screenshot"
":ParaView" -> Пользователь: вернуть файлы с построенными графиками
deactivate ":ParaView"
@enduml
```

В соответствии с рисунком 3.5 представлена диаграмма последовательностей для прецедента «Построение графиков примера».

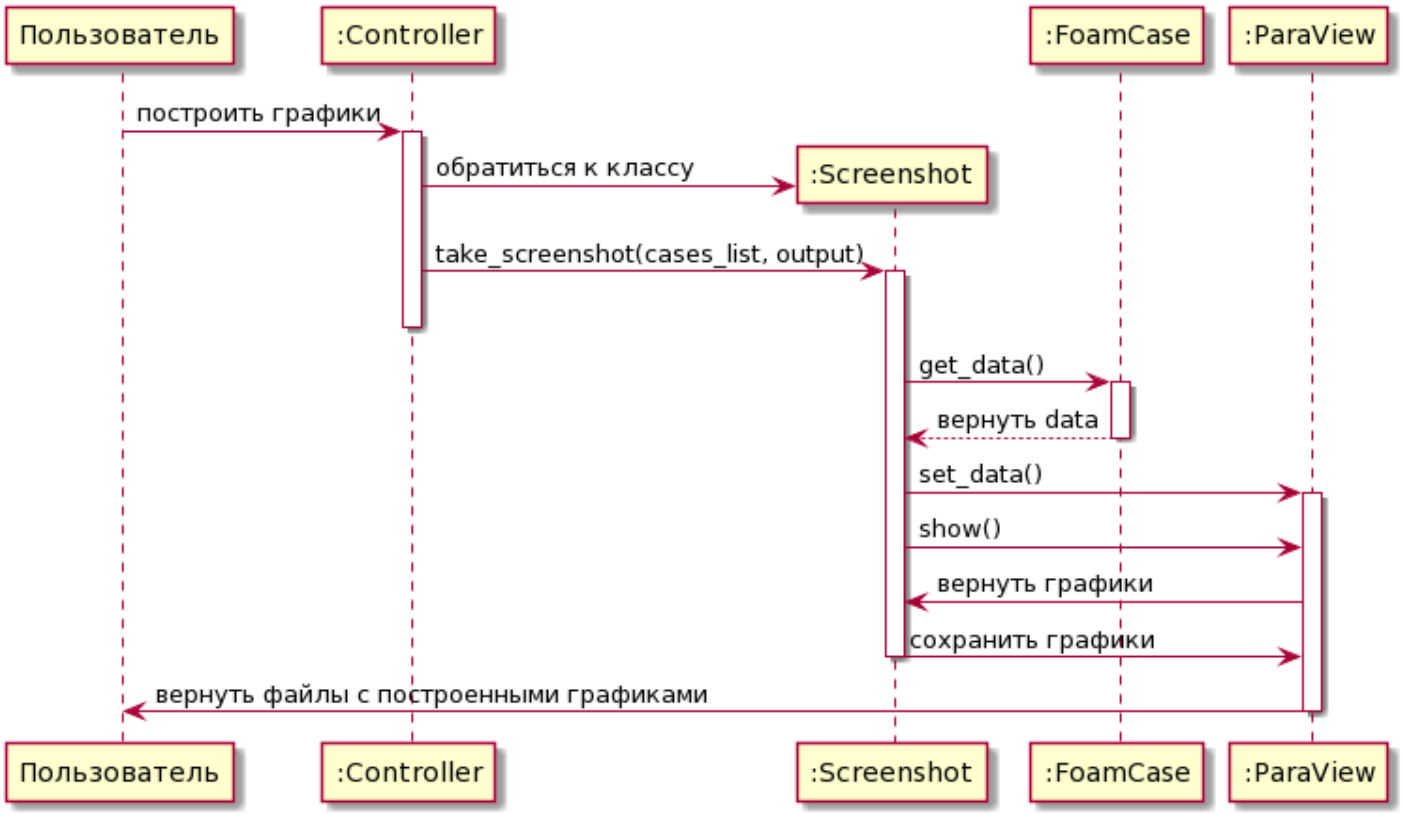


Рисунок 3.5 — Диаграмма последовательностей для прецедента «Построение графиков примера»

Код генерации диаграммы последовательностей для прецедента «Сохранение примера в базу данных»:

```
@startuml
Пользователь -> ":Controller" : сохранить пример
":Controller" -> ":MongoDaoFactory" : get_dao()
activate ":MongoDaoFactory"
```

```
5  ":MongoDaoFactory" -> ":Controller" : инстанцировать и вернуть экземпляр MongoFoamC
6  deactivate ":MongoDaoFactory"
7  ":Controller" -> ":MongoFoamCaseDAO": create_or_update(case)
8  activate ":MongoFoamCaseDAO"
9  ":MongoFoamCaseDAO" -> ":Controller": успех
10 deactivate ":MongoFoamCaseDAO"
11 ":Controller" -> Пользователь: запись завершена
12 deactivate ":Controller"
13 @enduml
```

В соответствии с рисунком 3.6 представлена диаграмма последовательностей для прецедента «Сохранение примера в базу данных». Как видно из диаграммы обращение к базе данных происходит по средствам соответствующего DAO-класса, экземпляр которого в свою очередь создается методом класса MongoDaoFactory.

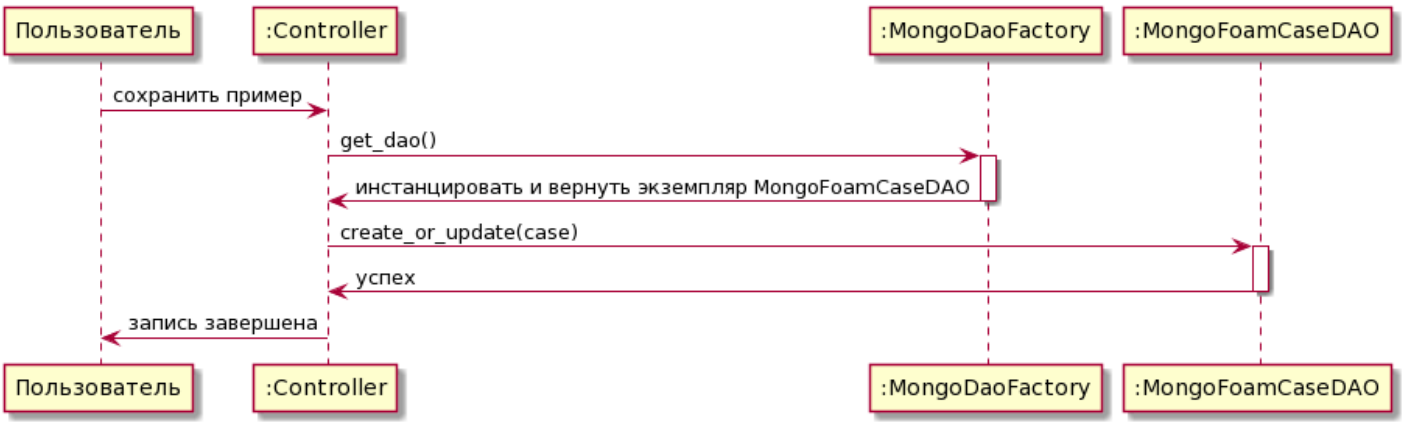


Рисунок 3.6 — Диаграмма последовательностей для прецедента «Сохранение примера в базу данных»

В соответствии с рисунком 3.7 рассмотрена диаграмма последовательностей для прецедента «Добавление источника данных».

На данной диаграмме хорошо видно как взаимодействуют классы, меняя общее состояние программы хранящиеся в классе SharedState. Также на этой диаграмме хорошо видно как класс Controller инкапсулирует основную логику работы программы.

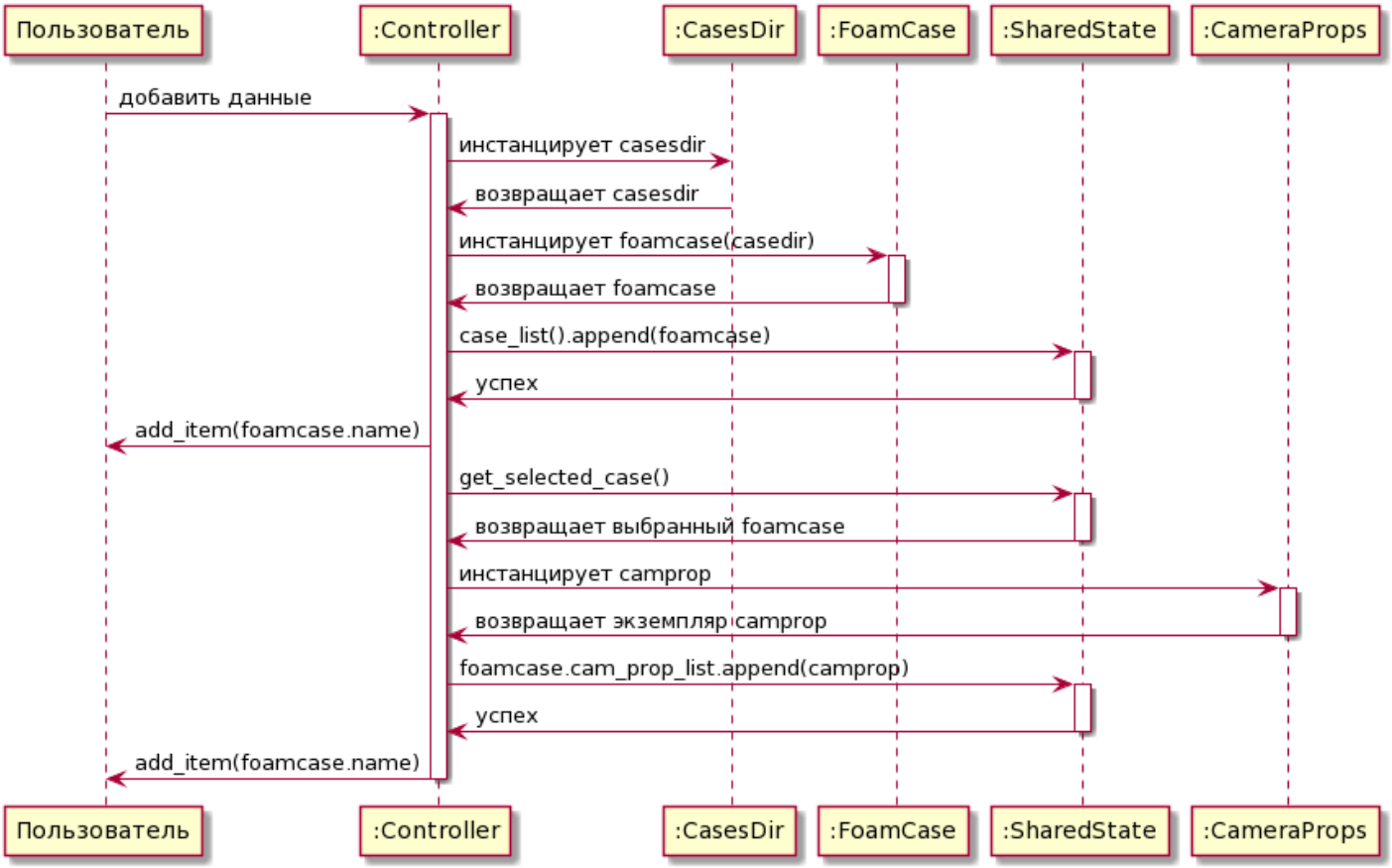


Рисунок 3.7 — Диаграмма последовательностей для прецедента «Добавление источника данных»

Код генерации:

```
@startuml
Пользователь -> ":Controller" : добавить данные
activate ":Controller"
":Controller" -> ":CasesDir" : инстанцирует casesdir
":CasesDir" -> ":Controller" : возвращает casesdir
":Controller" -> ":FoamCase" : инстанцирует foamcase(casedir)
activate ":FoamCase"
":FoamCase" -> ":Controller" : возвращает foamcase
deactivate ":FoamCase"
":Controller" -> ":SharedState" : case_list().append(foamcase)
activate ":SharedState"
":SharedState" -> ":Controller" : успех
deactivate ":SharedState"
":Controller" -> Пользователь : add_item(foamcase.name)
deactivate ":FoamCase"
deactivate ":CasesDir"
```

```
":Controller" -> ":SharedState" : get_selected_case()
activate ":SharedState"
":SharedState" -> ":Controller" : возвращает выбранный foamcase
deactivate ":SharedState"
":Controller" -> ":CameraProps" : инстанцирует camprop
activate ":CameraProps"
":CameraProps" -> ":Controller" : возвращает экземпляр camprop
deactivate ":CameraProps"
":Controller" -> ":SharedState" : foamcase.cam_prop_list.append(camprop)
activate ":SharedState"
":SharedState" -> ":Controller" : успех
deactivate ":SharedState"
":Controller" -> Пользователь : add_item(foamcase.name)
deactivate ":Controller"
@enduml
```

# 4 Выбор средств разработки

Для хранения данных был выбран NoSQL подход. Этот выбор обусловлен теми преимуществами, которые предоставляют СУБД этого вида, а именно:

- Не требуется иметь строгую схему данных.
- Линейная масштабируемость.

В качестве базы данных приложения была выбрана документоориентированная СУБД MongoDB, как одна из наиболее известных СУБД это вида. Рассмотрим данную базу данных и сам подход NoSQL.

## 4.1 NoSQL

NoSQL (с английского not only SQL, что означает не только SQL). Это обозначение широкого класса различных систем управления базами данных, появившихся в период с конца 2000-х – начала 2010-х годов и существенно отличающихся от традиционных реляционных СУБД с доступом к данным средствами языка SQL. Применяется к системам, в которых делается попытка решить проблемы масштабируемости и доступности за счёт полного или частичного отказа от требований атомарности и согласованности данных [14].

Существует несколько видов NoSQL СУБД:

- База данных «ключ-значение». Это наиболее простой вариант хранилища данных, использующий ключ для доступа к значению в рамках большой хэш-таблицы [15]. Такие СУБД применяются для хранения изображений, создания специализированных файловых систем, в качестве кэшей для объектов, а также в масштабируемых Big Data системах, включая игровые и рекламные приложения, а также проекты интернета вещей (Internet of Things, IoT). Наиболее известными представителями нереляционных СУБД типа key-value считаются Oracle NoSQL Database, Berkeley DB, MemcacheDB, Redis, Riak, Amazon DynamoDB, которые поддерживают высокую разделяемость, обеспечивая беспрецедентное горизонтальное масштабирование, недостижимое при использовании других типов БД [17].
- Документоориентированные СУБД. В ней находятся данные представленные парами ключ-значение, которые сжимаются в виде полуструк-

турированного документа из тегированных элементов, подобно JSON, XML, BSON и другим подобным форматам [15]. Такая модель хорошо подходит для каталогов, пользовательские профилей и систем управления контентом, где каждый документ уникален и изменяется со временем [17]. Поэтому чаще всего документные NoSQL-СУБД используются в CMS-системах, издательском деле и документальном поиске. Примеры документно-ориентированных нереляционных баз данных – это CouchDB, Couchbase, MongoDB, eXist, Berkeley DB XML [18].

- Колоночная база данных. Хранит информацию в виде разреженной матрицы, строки и столбцы которой используются как ключи. В мире Big Data к колоночным хранилищам относятся базы типа «семейство столбцов» (Column Family). В таких системах сами значения хранятся в столбцах (колонках), представленных в отдельных файлах. Благодаря такой модели данных можно хранить большое количество атрибутов в сжатом виде, что ускоряет выполнение запросов к базе, особенно операции поиска и агрегации данных. Наличие временных меток (timestamp) позволяет использовать такие СУБД для организации счётчиков, регистрации и обработки событий, связанных со временем: системы биржевой аналитики, IoT/IIoT-приложения, систему управления содержимым и так далее Самой известной колоночной базой данных является Google Big Table, а также основанные на ней Apache HBase и Cassandra. Также к этому типу относятся менее популярные ScyllaDB, Apache Accumulo и Hypertable.
- Графовая база данных. Такие хранилища представляют собой сетевую базу, которая использует узлы и рёбра для отображения и хранения данных [17]. Поскольку рёбра графа являются хранимыми, его обход не требует дополнительных вычислений (как соединение в SQL). При этом для нахождения начальной вершины обхода необходимы индексы. Обычно графовые СУБД поддерживают ACID-требования и специализированные языки запросов (Gremlin, Cypher, SPARQL, GraphQL и так далее). Такие СУБД используются в задачах, ориентированных на связи: социальные сети, выявление мошенничества, маршруты общественного транспорта, дорожные карты, сетевые топологии. Примеры гра-

фовых баз: InfoGrid, Neo4j, Amazon Neptune, OrientDB, AllegroGraph, Blazegraph, InfiniteGraph, FlockDB, Titan, ArangoDB.

Обратимся еще раз к документоориентированной СУБД. Она включает в себя кодировку документа, хранит метаданные, связанные с хранимой информацией, что даёт возможность делать запросы на основе это информации. Учитывая сказанное выше и также то, что такие базы данных работают без схемы данных, делает добавление полей в JSON-документы достаточно простой задачей.

Таким образом, имеется возможность сразу хранить некий аналог объектно-ориентированной модели, что упрощает написание соответствующих классов в программе.

## 4.2 MongoDB

В качестве документоориентированной СУБД была использована MongoDB, ввиду ее популярности и простоте освоения.

В рейтинге СУБД MongoDB прочно удерживается среди десяти лучших баз данных, занимая в нем пятую позицию [16].

MongoDB – это база данных документов с открытым исходным кодом, построенная на горизонтально-масштабируемой архитектуре. Каждая строка в базе данных MongoDB представляет собой документ, описанный на языке форматирования JSON. В ниже для примера представлен простой документ JSON с описанием контактной информации [19].

```
{
        "name" :  "Carlos Smith",
        "title" : "Product Manager",
        "location" : "New York, NY",
        "twitter" : "@MongoDB",
        "facebook" : "@MongoDB"
}
```

JSON эффективен по многим причинам:

- Это естественная форма хранения данных.
- Легко воспринимается людьми.

- Структурированная и неструктурированная информация может храниться в одном документе.
- Благодаря JSON, документы могут иметь неограниченную вложенность друг в друга.
- JSON имеет гибкую и динамическую схему, поэтому добавление или удаление полей из документа не представляет каких-либо сложностей. Возможно хранить, например, частично завершенные документы.

Также важно отметить, что структура данных находится под контролем разработчика. Это в свою очередь предоставляет возможность корректировать и переформатировать базу данных по мере развития приложения без помощи администрирования базы данных. При необходимости MongoDB может координировать и контролировать изменения в структуре документов. Поля в документе играют роль столбцов в базе данных SQL, и, как столбцы, их можно индексировать для повышения производительности поиска.

MongoDB имеет хороший пользовательский web-интерфейс, что также является плюсом.

Из недостатков следует отметить ложащуюся на разработчика ответственность за формализацию базы данных. Недостаточный контроль может привести к путанице и лишним издержкам на больших проектах.

## 4.3 Python

Выбор языка программирования должен быть обусловлен возможностью интеграции с выбранной базой данных, а также обеспечивать доступ к Paraview API. В этой связи, будет рассмотрен язык программирования Python как наиболее подходящий кандидат.

Высокоуровневый язык программирования Python является языком общего назначения с динамической строгой типизацией и автоматическим управлением памятью, ориентированный на повышение производительности разработчика, читаемости кода и его качества, а также на обеспечение переносимости написанных на нём программ. Язык является полностью объектно-ориентированным – всё является объектами [20]. Характерное отличие языка – выделение блоков кода пробельными символами или табуляцией. Синтак-

сис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации [21]. Python – интерпретируемый мультипарадигмальный язык программирования, который используется для различных задач [20]. Недостатками языка являются зачастую более низкая скорость работы и более высокое потребление памяти написанных на нём программ по сравнению с аналогичным кодом, написанным на компилируемых языках, таких как Си или C++ [21].

Тем не менее, Python будет легко использовать при дальнейшем расширении программы. Например, для добавления веб-интерфейса приложения можно будет использовать Django и flask. Или при что-то более легковесное, например FastAPI и React.

У приложения Paraview имеется API для двух языков: Python и C++. Это сразу ограничивает выбор языка разработки. Python более современный, комфортный и гибкий, часто выступает как более удобный интерфейс для C++ библиотек. Также Python имеет интеграцию с MongoDB через библиотеку PyMongo

В качестве языка программирования для написания работы был выбран Python, так как он достаточно гибкий и удобный, а также имеет интеграцию с ParaView через библиотеку ParaView Python API и с MongoDB через библиотеку PyMongo.

Немаловажная деталь – это выбор среды разработки. К этому следует также подходить внимательно, так как верно выбранная среда разработки сильно сокращает время написания, отладки и тестирования программы. Остановимся PyCharm Community Edition от компании JetBrains. Среда разработки обеспечивает:

- Помощь в написании кода и анализ с автозавершением кода, подсветкой синтаксиса и ошибок, быстрыми исправлениями.
- Навигация по проекту и коду: специализированные представления проекта, представления структуры файлов и быстрое переключение между файлами, классами, методами и способами использования.
- Рефакторинг Python: включает переименование, извлечение метода, введение переменной, введение константы, подтягивание, нажатие и другие,

- Встроенный отладчик Python.
- Интегрированное модульное тестирование с построчным покрытием кода.
- Интеграция контроля версий: единый пользовательский интерфейс для Mercurial, Git, Subversion, Perforce и CVS со списками изменений и слиянием.

Выбор обусловлен популярностью и удобством программного продукта и тем, что средства отладки и автозаполнения кода предоставляются из коробки. Это дает возможность не тратить время на установку и налаживание работы. Также в PyCharm присутствует интеграция виртуальным окружением Anaconda, которое сильно упрощает процесс подключения ParaView Python API. Скриншот окна программы представлен в соответствии с рисунком 4.1.

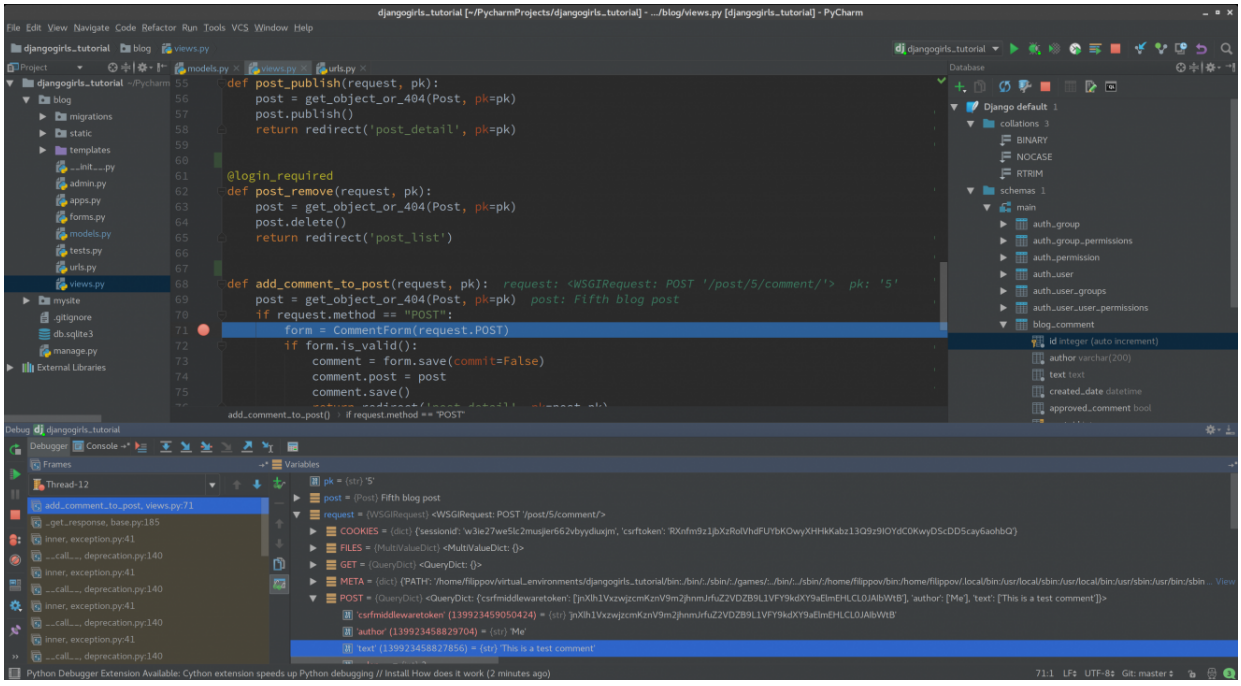

Рисунок 4.1 — Рабочее окно программы PyCharm

Таким образом, перечисленные выше преимущества объясняют выбор конкретно этой среды разработки.

## 4.4 Обзор средств разработки графического интерфейса пользователя

Также необходимо выбрать фреймворк для разработки графического интерфейса пользователя приложения. Принимая во внимание, что мы разрабатываем на Python и, учитывая, потенциальную расширяемость программы.

### 4.4.1 Kivy

Kivy – это бесплатный фреймворк Python с открытым исходным кодом для разработки мобильных приложений и другого программного обеспечения для широкого назначения с естественным пользовательским интерфейсом. Он распространяется в соответствии с условиями лицензии MIT, что дает право использовать библиотеку бесплатно в личных и коммерческих целях. Kivy может работать на Android, iOS, Linux, macOS и Windows [23].

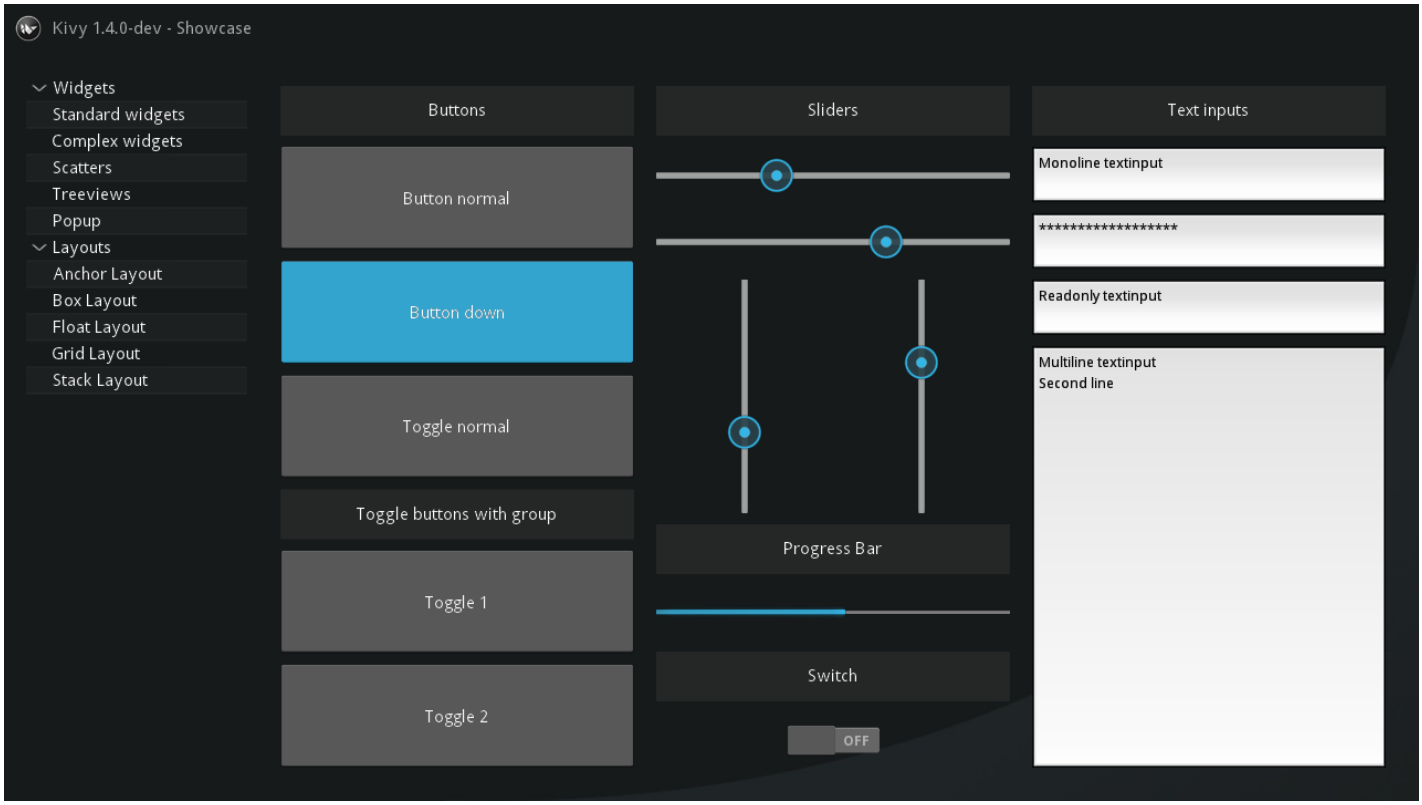


Рисунок 4.2 — Приложение Kivy

Фреймворк содержит все элементы для создания приложения, такие как:

- Расширенная поддержка ввода для мыши, клавиатуры, TUIO и событий мультитач ОС.
- Графическая библиотека, использующая только OpenGL ES 2 и основанная на Vertex Buffer Object и шейдерах.
- Широкий выбор виджетов, поддерживающих мультитач.
- Промежуточный язык, используемый для простой разработки пользовательских виджетов.

### 4.4.2 TKinter

Tkinter – это событийно-ориентированная кросс-платформенная графическая библиотека. Она достаточно широко распространена в мире GNU/Linux и других UNIX-подобных систем, портирована также и на Microsoft Windows. Входит в стандартную библиотеку Python и является свободным программным обеспечением.

Преимущества:

- Краткость. Программы Python, использующие Tkinter, могут быть весьма короткими. Например, для многих используемых для создания виджетов параметров, значениям по умолчанию заданы разумные значения, что в комбинации с Python дает весьма компактный код.
- Кросс-платформенный. Tk предоставляет виджеты для Windows, Mac и большинства реализаций Unix. При это существует зависимость от платформы, но небольшая.
- Ядро хорошо разработано и стабильно
- Расширяемость. Существует множество расширений Tk.

Недостатки Tkinter – скорость. Большинство вызовов Tkinter форматируются как Tcl-команда и интерпретируются с помощью Tcl. Отсюда фактические и выполняются вызовы Tkinter. Таким образом происходит последовательный вызов двух интерпретируемых языков [26].

### 4.4.3 wxPython

wxPython - это кроссплатформенный набор инструментов с графическим интерфейсом для языка программирования Python. Он позволяет программистам Python просто и легко создавать программы с графическим пользовательским интерфейсом. Он реализован в виде набора модулей расширения Python, которые обертывают компоненты графического интерфейса популярной кроссплатформенной библиотеки wxWidgets, написанной на C++ [27].

Подобно Python и wxWidgets, wxPython является открытым исходным кодом. Также он является кроссплатформенным инструментарием. В настоящее время поддерживаемыми платформами являются Microsoft Windows, Mac OS X и macOS, а также Linux или другие unix-подобные системы с библиотеками GTK2 или GTK3.

Присутствует официальная документация, но не хватает сторонних источников и примеров использования библиотеки.

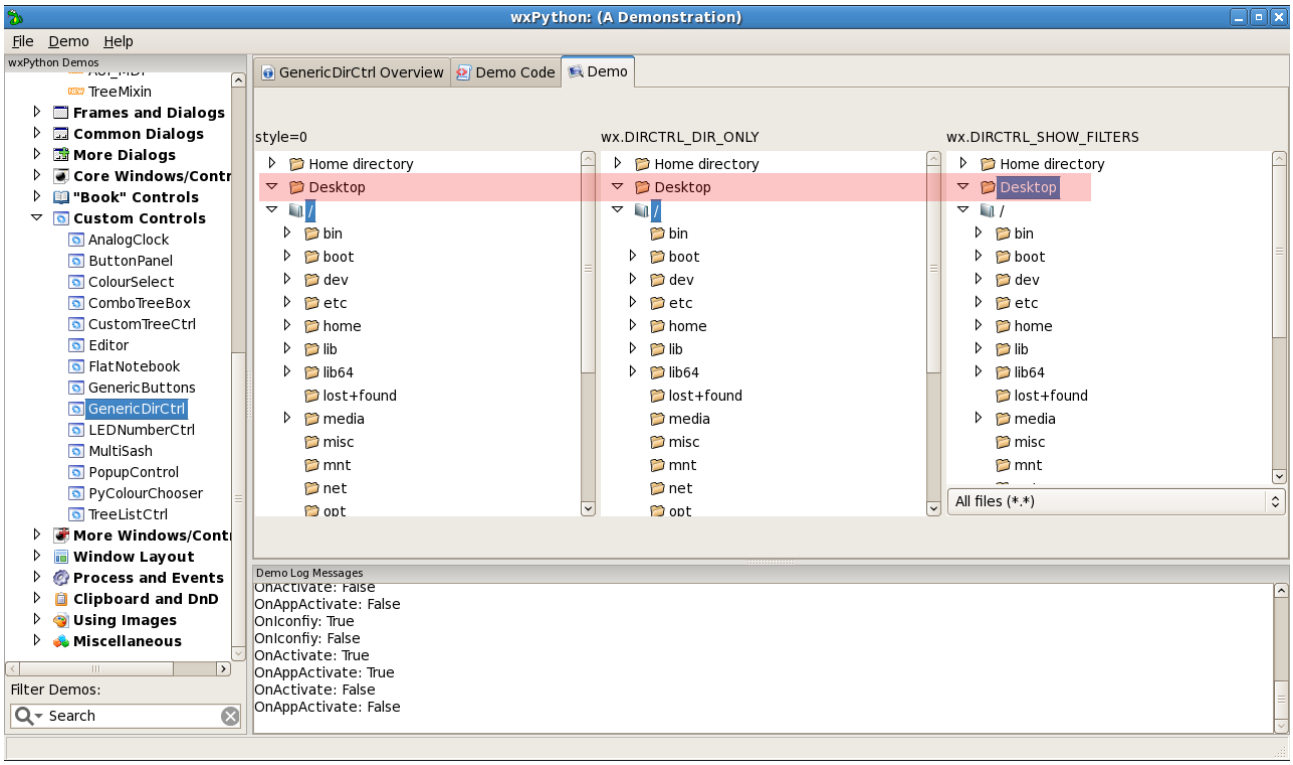


Рисунок 4.3 — Пример использующей wxPython программы

### 4.4.4 PyQt/PySide

PyQt — набор расширений (биндингов) графического фреймворка Qt для языка программирования Python, выполненный в виде расширения Python.

PyQt работает на всех платформах, поддерживаемых Qt: Linux и другие UNIX-подобные ОС, Mac OS X и Windows. Существует 2 версии: PyQt5, поддерживающий Qt 5, и PyQt4, поддерживающий Qt 4. PyQt распространяется под лицензиями GPL (2 и 3 версии) и коммерческой [24].

PyQt также включает в себя Qt Designer (Qt Creator) — дизайнер графического интерфейса пользователя. Программа pyuic генерирует Python код из файлов, созданных в Qt Designer. Это делает PyQt очень полезным инструментом для быстрого прототипирования. Кроме того, можно добавлять новые графические элементы управления, написанные на Python, в Qt Designer.

Однако, для PyQt существуют ограничения связанные с лицензией GPL, поэтому был разработан PySide. PySide – привязка языка Python к инструментарию Qt, совместимая на уровне API с PyQt. В отличие от PyQt, PySide доступна для свободного использования как в открытых, так и закрытых, в частности, коммерческих проектах, поскольку лицензирована по LGPL [25].

Все представленные библиотеки имеют примерно одинаковые преимущества, но стоит выбирать наиболее зрелые и популярные. Это даст возможность в случаи необходимости легко найти решение проблем, возникающих в процессе разработки приложения. PySide кажется хорошим вариантом.

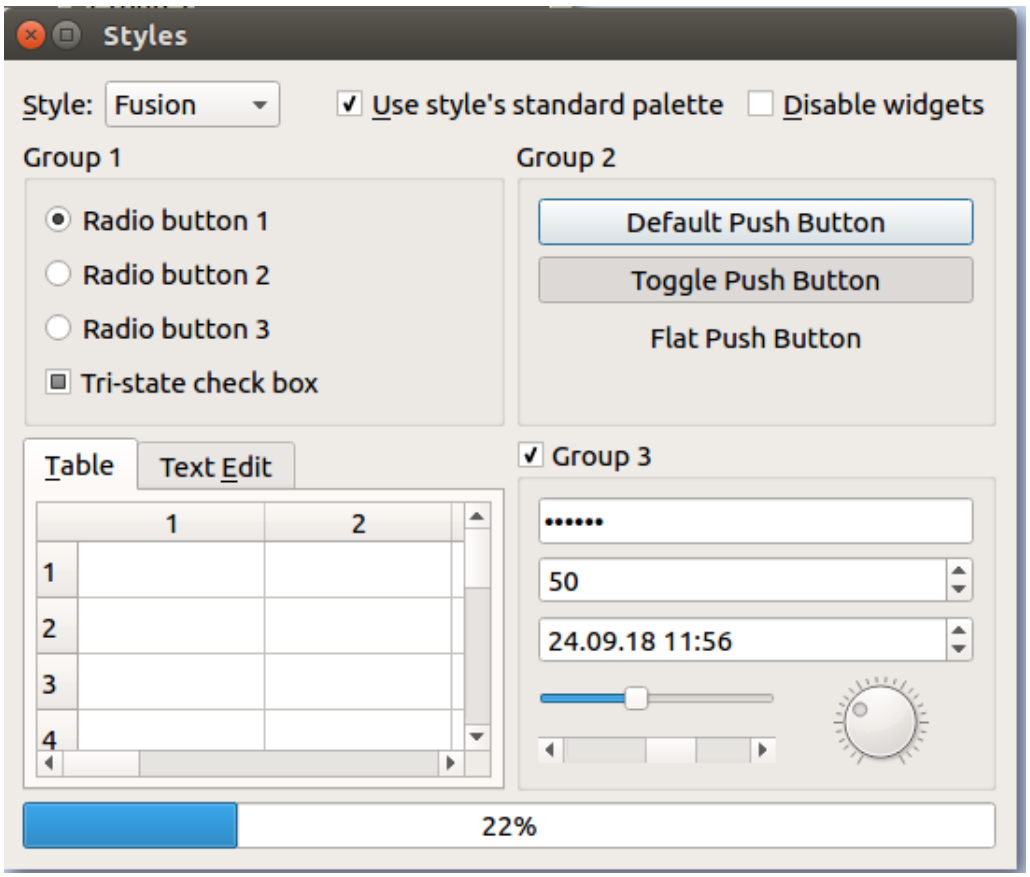


Рисунок 4.4 — Приложение, написанное с помощью PySide

Преимущества PySide с точки зрения выбора фреймворка:

- Стабильность.
- Подробнейшая документация.
- Обилие дополнительной информации на различных форумах и сайтах.

Недостатки:

- Сложность в освоении из-за большого размера.
- Не вся документация доступна для языка Python.

Но эти недостатки нивелируются. В частности, первый из них не имеет значения, если есть некий опыт работы и общее понимание принципов устройства фреймворка. А второй недостаток, также преодолевается опытом,подобием синтаксиса в целом и большим количеством сторонней информации в сети интернет.

Таким образом, предпочтение было отдано PySide.

## 5 Разработка информационной системы

В соответствии с рисунком 5.1 представлен скриншот разработанной программы.

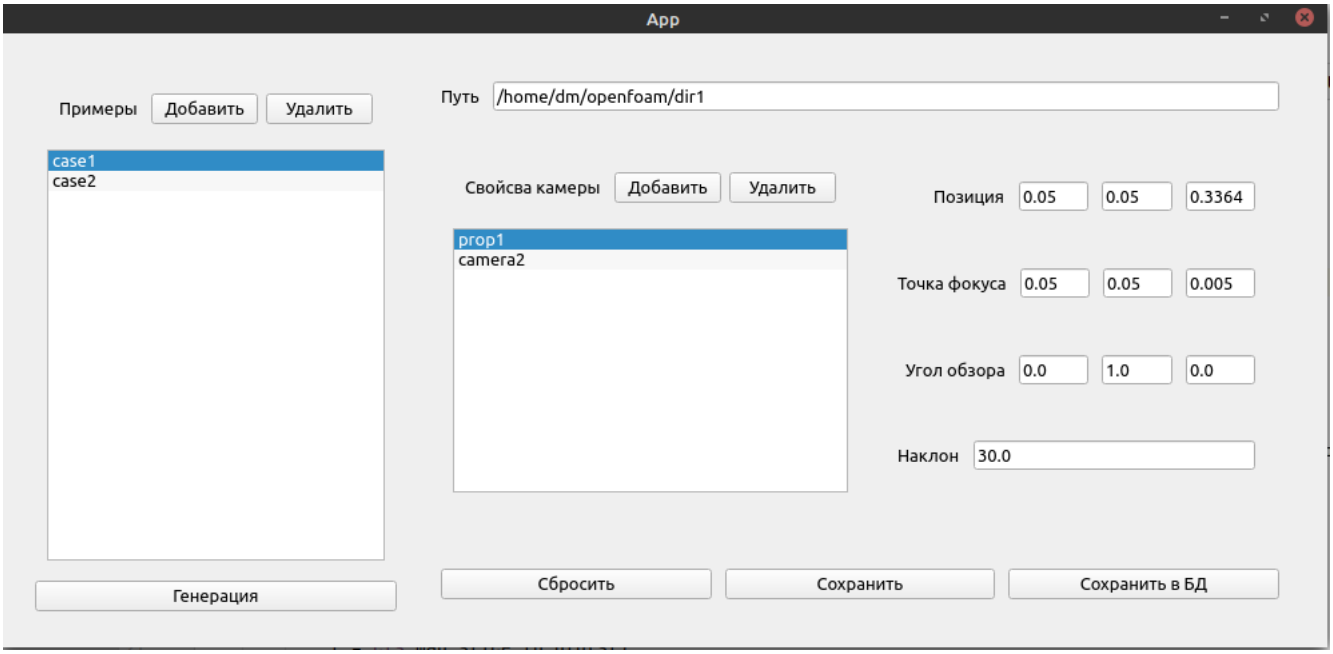


Рисунок 5.1 — Разработанное приложение

Разработка была начата с модуля model. В нем содержаться классы модели данных: FoamCase, CameraProps, Point, CaseDir. Затем были разработаны соответствующие им DTO-классы (Data Trasfer Object). После этого был написан класс Mapper для отображения DTO в model и обратно. Затем нужно было приступить к разработке DAO слоя и схемы базы данных.

На самом деле, так как была использована документоориентированная СУБД MongoDB, то необходимость в проектировки схемы базы данных отпала, однако, все же нужно было создать коллекции и получить доступ к хранилищу. В соответствии с рисунком 5.2 показан web-интерфейс созданного в результате экземпляра базы данных MongoDB.

Также для объектной модели требовалось добавить поля _id во все классы и DTO объекты соответственно. Поля _id нужно было инициализировать, используя класс ObjectId из библиотеки PyMongo. С этим были связаны некоторые трудности, так как класс Mapper также пришлось переписывать, учитывая нововведения.

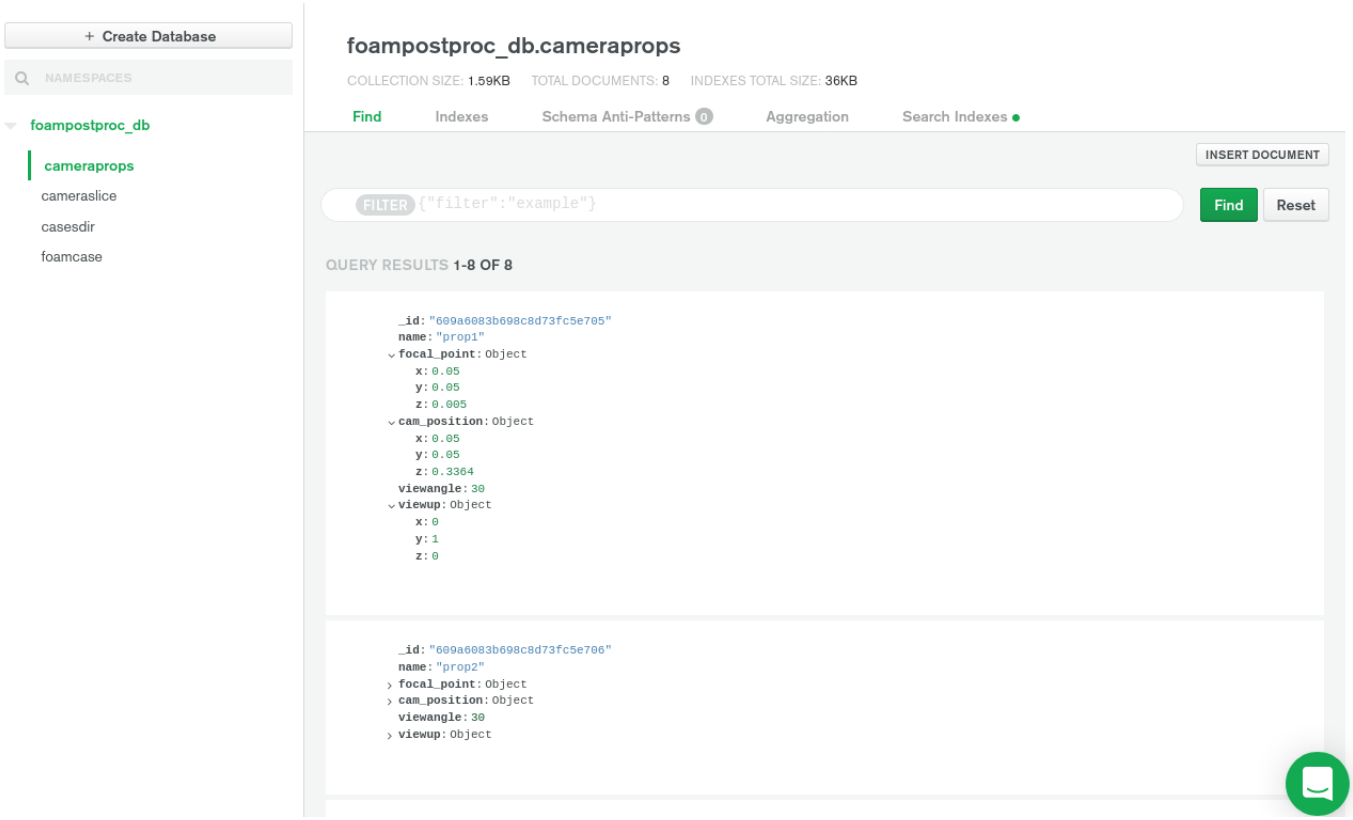


Рисунок 5.2 — Web-интерфейс экземпляра базы данных MongoDB

Параллельно с этим возникали трудности с работой в ParaView Python API. Данная библиотека плохо документирована. На официальном сайте есть краткие пояснения назначения методов, но совершенно отсутствует описание классов и их полей, также на официальном сайте отсутствует описание процесса работы с библиотекой и какие-либо полезные примеры.

Большое время заняло подключение ParaView Python API к виртуальному окружению Python. Следует отметить, что результата удалось достичь только используя окружение Anaconda. В окружении Anaconda, в свою очередь, возникли проблемы с установкой соответствующих пакетов, что в последствии привело к трудностям с запуском PySide6.

Более того, после успешного подключения библиотеки возникали трудности с подсказками методов, что очень сильно тормозило работу (принимая во внимание отсутствие полезной документации).

Тем не менее, получения тестовых графиков от ParaView Python API, был разработан DAO-слой, который отвечает за доступ к базе данных. После чего был сделан графический интерфейс пользователя. И оставшееся время было

потрачено на написание класса Controller. Он отвечает за обработку событий вызванных графическим интерфейсом пользователя и объединяет элементы программы, а именно:

- Модель
- Графический интерфейс пользователя
- Логику отвечающую за создание и редактирование примеров
- Логику генерацию графиков

В завершении работы был добавлен класс Config, который обращается к соответствующему файлу, куда были вынесены параметры программы, например логин и пароль от базы данных и передает их классам для работы.

Исходный код программы приведен в соотвествии с приложением А.

## 6 Пример анализа экспериментальных данных

Как было сказано ранее, при обработке экспериментальных данных, когда проводится серия экспериментов, в которых входные данные незначительно изменяются, часто порождается большой объем результатов, подлежащих анализу или, иными словами, постобработке. Для анализа таких данных необходимо проводить однотипные манипуляции.

Например, рассматривается движение вязкой жидкости в каверне. Такое движение описывают уравнения Навье-Стокса. Уравнения Навье-Стокса – система дифференциальных уравнений в частных производных, описывающая движение вязкой ньютоновской жидкости. Уравнения Навье – Стокса являются одними из важнейших в гидродинамике и применяются в математическом моделировании многих природных явлений и технических задач.

Геометрия течения показана в соответствии с рисунком 6.1 в которой все границы квадрата являются твердыми стенками длинной 0.1 м. Верхняя стенка перемещается в x-направлении со скоростью 1 м/с, в то время как другие 3 неподвижны.

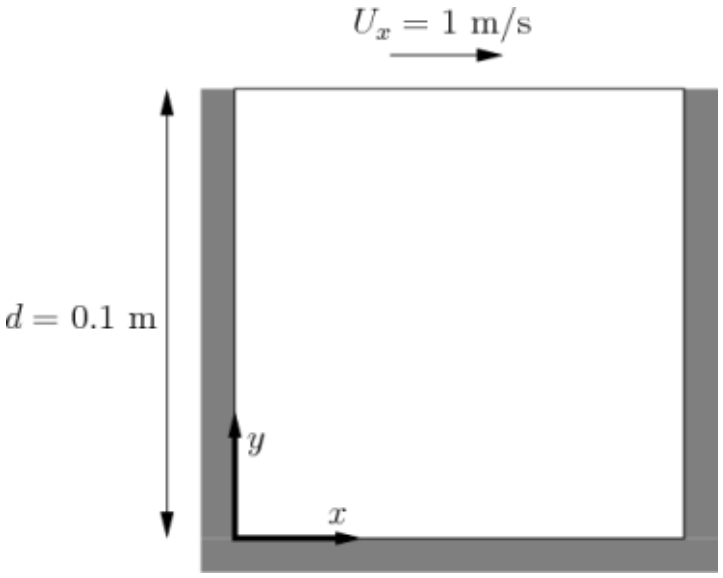


Рисунок 6.1 — Геометрия расчетной области задачи

Пусть для уравнениях Навье-Стокса в случае несжимаемых жидкостей потребовалось построить графики скорости для последовательно увеличивающейся кинематической вязкости.

Задав положение камеры и директорию с расчетами или загрузив эти данные из базы данных запустим генерацию графиков и оценим результаты.

Графики были построены для коэффициентов вязкости соответственно от 0.01 до 0.18 с шагом 3, либо 2. Для краткости приведем три из них: первый, затем из середины и последний график.

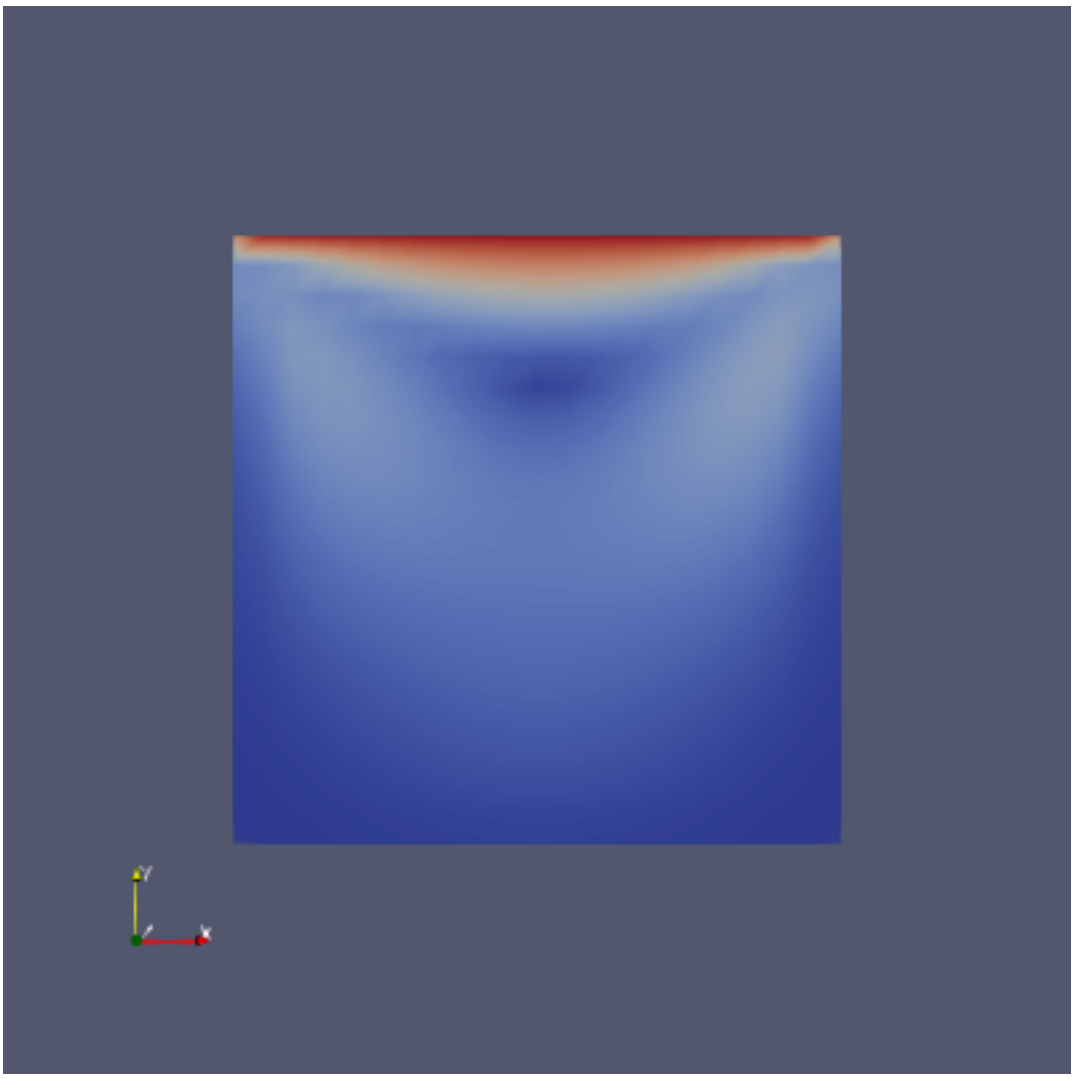

Рисунок 6.2 — График с коэффициентов вязкости 0.01

В соответствии с рисунком 6.2, в соответствии с рисунком 6.3 и в соответствии с рисунком 6.4 видно, что пока рос коэффициент вязкости графики практически не менялись, но ближе к концу заметен резкий всплеск.

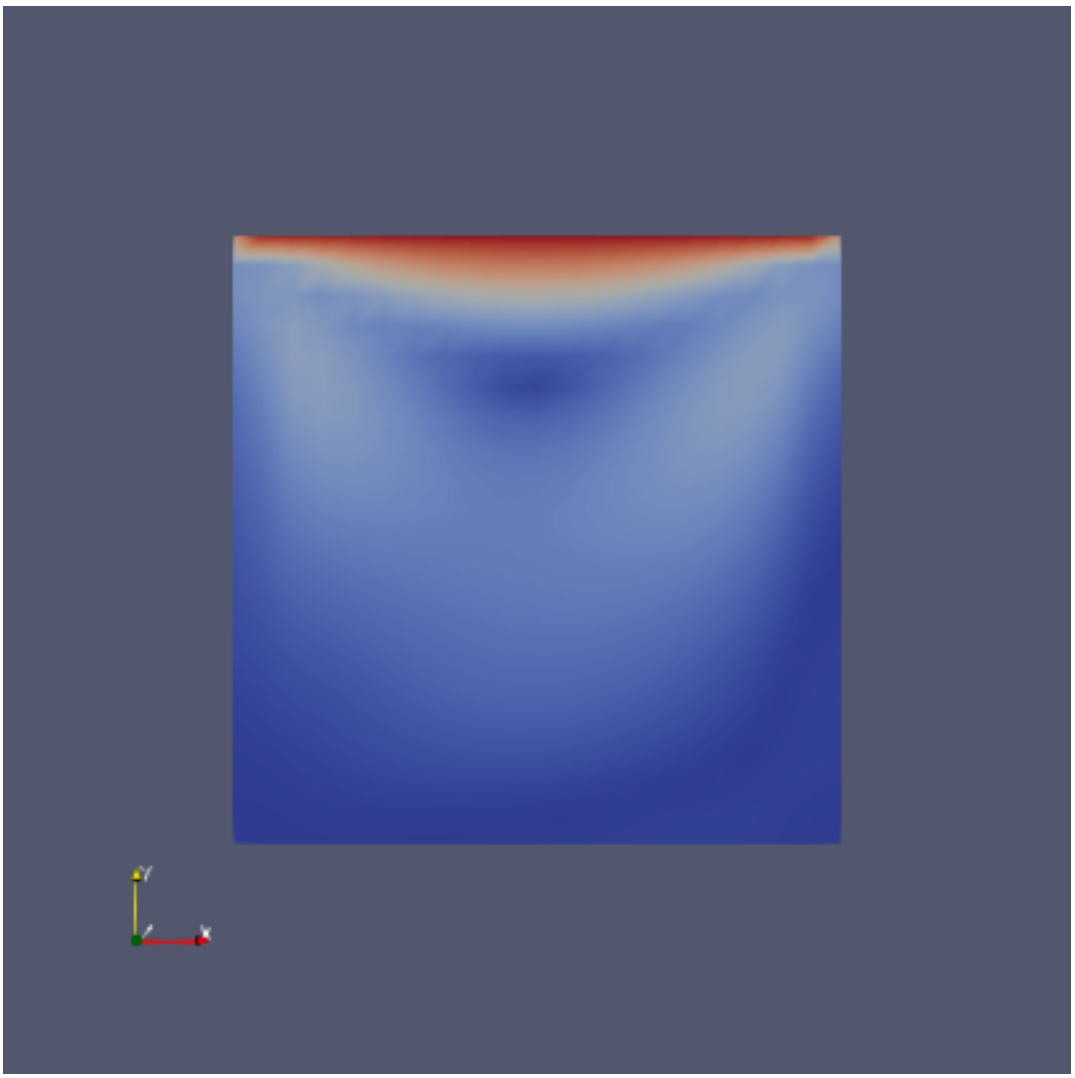

Рисунок 6.3 — График с коэффициентов вязкости 0.09

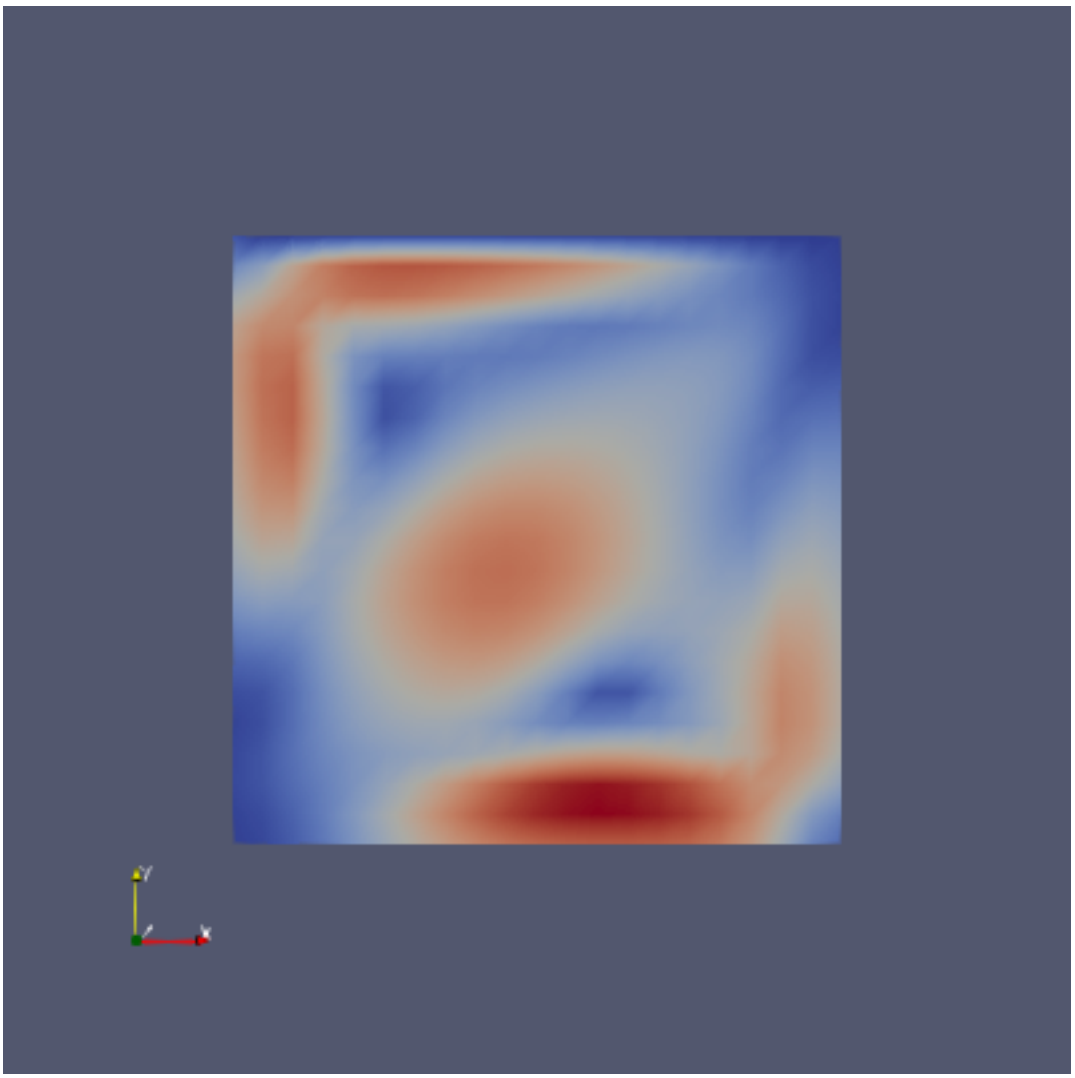

Рисунок 6.4 — График с коэффициентов вязкости 0.18

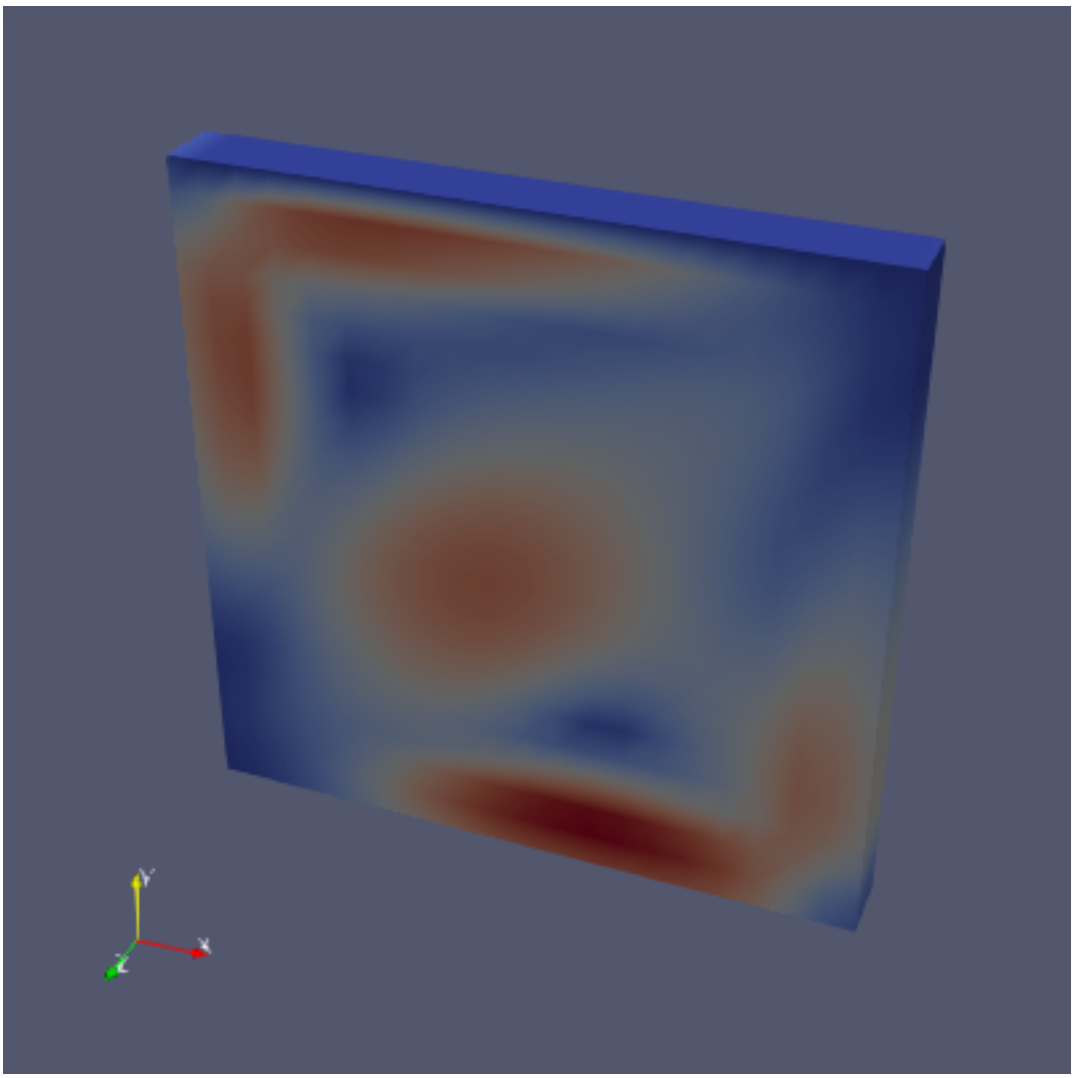

Рисунок 6.5 — График с коэффициентов вязкости 0.18 с другого ракурса

# ЗАКЛЮЧЕНИЕ

При обработке экспериментальных данных и расчетов, полученных в результате математического моделирования физических процессов когда проводится серия экспериментов, в которых входные данные незначительно изменяются порождается большой объем результатов, подлежащих постобработке.

Как было указано во введении, целью данной магистерской работы является проектирование и разработка информационной системы, автоматизирующей рутинные операции анализа экспериментальных данных для CAD/CAE систем. Конкретизируя цель, были поставлены следующие задачи:

- Кратко рассмотреть платформу для численного моделирования OpenFOAM и пакет для визуализации ParaView.

  Благодаря решению этой задачи понятие CAD/CAE приложение было уточнено до конкретного решения – пакет программ OpenFOAM и инструмент постобработки ParaView, таким образом неявно был очерчен контекст разработки приложения.
- Рассмотреть существующие решения по данной тематике.

  При решении этой задачи были проанализированные существующие на данный момент программы, предоставляющие похожий функционал автоматизации моделирования. Были приведены их преимущества и недостатки, а также было дано некоторое обоснование актуальности разрабатываемого приложения.
- Спроектировать информационную систему и создать UML-диаграммы для ее описания.

  В рамках этой задачи была проведена основная работа по проектированию приложения, очерчен его контур и структура связей. Также были подробно рассмотрены основные классы программы, приведены примеры кода построения UML-диаграмм и результат их генерации.
- Провести обзор средств разработки.

  Здесь были рассмотрены средства разработки, последовательно обоснован выбор определенных инструментов. В некоторых случаях приведено сравнения с аналогичными по функционалу библиотеками и фреймворками.

- Разработать помогающую в анализе экспериментальных данных информационную систему.

  Решая эту задачу, был подробно описан процесс разработки и возникшие в ходе нее проблемы.

Дополнительно был добавлен раздел с примером анализа экспериментальных данных при помощи разработанного приложения.

Таким образом, в данной работе был кратко рассмотрен пакет программ для численного моделирования OpenFOAM, приложение для постпроцессинга ParaView. Было выполнено проектирование, в частности построены UML-диаграммы прецедентов, классов, последовательностей. Также были рассмотрены существующие на данный момент решения постобработки экспериментальных данных и выбраны средства разработки.

Было разработано приложение, которое выполняет одну из подзадач разработки и анализа экспериментальных данных, в частности программа строит графики указанных расчетов по заданным параметрам и хранит данные, используемые для построения в БД, используя NoSQL подход. Программа позволяет не тратить время при постобработке ряда слабо отличающихся друг от друга геометрией и условиями данных, генерируя графики группами сразу для всех расчетов. Этим она отличается и выделяется на фоне рассмотренных решений.

Сама разработка носит итеративный процесс, поэтому трудно судить о полноте выполненных задач в плане написания кода. Однако, можно с некоторой долей уверенности сказать, что минимально жизнеспособный продукт был создан, но еще много деталей требуют внимания и дальнейшего развития.

Таким образом, задачи данной работы были выполнены и, следовательно, поставленная цель была достигнута.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ


1. Статья OpenFOAM [Электронный ресурс] : (на 08 апреля 2021 года) // Официальный веб-сайт Wikipedia [Электронный ресурс] : [сайт]. - URL: https://ru.wikipedia.org/wiki/OpenFOAM (дата обращения 08.04.2021). - Загл с экрана - Яз. рус.

2. Страница User guide [Электронный ресурс] : (на 08 апреля 2021 года) // Официальный веб-сайт OpenFOAM [Электронный ресурс] : [сайт]. - URL: https://cfd.direct/openfoam/user-guide (дата обращения 08.04.2021). - Загл с экрана - Яз. англ.

3. Страница About [Электронный ресурс] : (на 08 апреля 2021 года) // Официальный веб-сайт ParaView [Электронный ресурс] : [сайт]. - URL: https://www.paraview.org/overview (дата обращения 08.04.2021). - Загл с экрана - Яз. англ.

4. Статья ParaView [Электронный ресурс] : (на 09 апреля 2021 года) // Официальный веб-сайт Wikipedia [Электронный ресурс] : [сайт]. - URL: https://ru.wikipedia.org/wiki/ParaView (дата обращения 09.04.2021). - Загл с экрана - Яз. рус.

5. Страница ParaView and Python [Электронный ресурс] : (на 10 апреля 2021 года) // Официальный веб-сайт ParaView [Электронный ресурс] : [сайт]. - URL: https://www.paraview.org/Wiki/ParaView_and_Python (дата обращения 10.04.2021). - Загл с экрана - Яз. англ.

6. Продукт Helyx OS [Электронный ресурс] : (на 29 марта 2021 года) // Официальный веб-сайт Helyx [Электронный ресурс] : [сайт]. - URL: https://engys.com/products/helyx-os (дата обращения 29.03.2021). - Загл с экрана - Яз. англ.

7. Продукт ANSA [Электронный ресурс] : (на 11 апреля 2021 года) // Официальный веб-сайт компании Beta [Электронный ресурс] : [сайт]. - URL: http://www.beta-cae.gr/ansa.html (дата обращения 11.04.2021). - Загл с экрана - Яз. англ.

8. Продукт CastNet [Электронный ресурс] : (на 11 апреля 2021 года) // Официальный веб-сайт компании DHCAE Tools [Электронный ресурс] : [сайт]. - URL: http://www.dhcae-tools.com/CastNet.html (дата обращения 11.04.2021). - Загл с экрана - Яз. англ.

9. Фаулер, М. UML. Основы, 3-е издание/ М. Фаулер – Спб: Символ-Плюс, 2004. - 192 с.

10. Гамма, Э. Приемы объектно-ориентированного проектирования/ Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес – Спб: Питер, 2019. - 368 с.

11. Паттерны проектирования. Фабрика [Электронный ресурс] : (на 19 апреля 2021 года) // Веб-сайт refactoring guru [Электронный ресурс] : [сайт]. - URL: https://refactoring.guru/ru/design-patterns/factory-method (дата обращения 19.04.2021). - Загл с экрана - Яз. рус.

12. Паттерны проектирования. Заместитель [Электронный ресурс] : (на 19 апреля 2021 года) // Веб-сайт refactoring guru [Электронный ресурс] : [сайт]. - URL: https://refactoring.guru/ru/design-patterns/proxy (дата обращения 19.04.2021). - Загл с экрана - Яз. рус.

13. Ларман, К. Применение UML 2.0 и шаблонов проектирования. Практическое руководство. 3-е издание/ К. Ларман – М: И.Д. Вильямс, 2013. - 736 с.

14. Фаулер М. NoSQL. Методология разработки нереляционных баз данных / М. Фаулер, П. Садаладж – Диалектика-Вильямс, 2021. - 192 с.

15. Обзор NoSQL [Электронный ресурс] : (на 19 апреля 2021 года) // Веб-сайт Amazon Web Services [Электронный ресурс] : [сайт]. - URL: https://aws.amazon.com/ru/nosql/ (дата обращения 19.04.2021). - Загл с экрана - Яз. рус.

16. DB-Engines Ranking [Электронный ресурс] : (на 19 апреля 2021 года) // Веб-сайт DB Engines [Электронный ресурс] : [сайт]. - URL: https://db-engines.com/en/ranking (дата обращения 19.04.2021). - Загл с экрана - Яз. англ.

17. Разбираемся в типах NoSQL СУБД [Электронный ресурс] : (на 19 апреля 2021 года) // Веб-сайт Tproger [Электронный ресурс] : [сайт]. - URL: https://tproger.ru/translations/types-of-nosql-db/ (дата обращения 19.04.2021). - Загл с экрана - Яз. рус.

18. NoSQL СУБД [Электронный ресурс] : (на 20 апреля 2021 года) // Веб-сайт wikipedia [Электронный ресурс] : [сайт]. - https://ru.wikipedia.org/wiki/NoSQL (дата обращения 20.04.2021). - Загл с экрана - Яз. рус.

19. MongoDB [Электронный ресурс] : (на 23 апреля 2021 года) // Веб-сайт mongodb [Электронный ресурс] : [сайт]. - https://www.mongodb.com/why-use-mongodb (дата обращения 23.04.2021). - Загл с экрана - Яз. англ.

20. Yogesh R., Python: Simple though an Important Programming language [Текст] / R. Yogesh // International Research Journal of Engineering and Technology (IRJET). – 2019. - том 06, номер 2. – С. 1856—1858.

21. Лутц, М. Программирование на Python, том 1 / М. Лутц – СПб: Символ-Плюс, 2011. - 992 с.

22. Сравнение интегрированных сред разработки [Электронный ресурс] : (на 24 апреля 2021 года) // Веб-сайт wikipedia [Электронный ресурс] : [сайт]. - https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments#Python (дата обращения 24.04.2021). - Загл с экрана - Яз. англ.

23. Страница о нас [Электронный ресурс] : (на 30 апреля 2021 года) // Веб-сайт фреймворка Kivy [Электронный ресурс] : [сайт]. - https://kivy.org/#aboutus (дата обращения 30.04.2021). - Загл с экрана - Яз. англ.

24. PyQt. Введение [Электронный ресурс] : (на 30 апреля 2021 года) // Веб-сайт Riverbank Computing [Электронный ресурс] : [сайт]. - https://riverbankcomputing.com/software/pyqt/intro (дата обращения 30.04.2021). - Загл с экрана - Яз. англ.

25. Qt for Python [Электронный ресурс] : (на 30 апреля 2021 года) // Веб-сайт Wiki Qt [Электронный ресурс] : [сайт]. - https://wiki.qt.io/Qt_for_Python/ru (дата обращения 30.04.2021). - Загл с экрана - Яз. рус.

26. Robinson A. Python Programming On Win32 / A. Robinson, M. Hammond – O'Reilly Media, 2000. - 674 с.

27. Overview of wxPython [Электронный ресурс] : (на 30 апреля 2021 года) // Веб-сайт wxPython [Электронный ресурс] : [сайт]. - https://www.wxpython.org/pages/overview/ (дата обращения 30.04.2021). - Загл с экрана - Яз. англ.

# ПРИЛОЖЕНИЕ А

## Исходный код программы

### A.1 foampostproc.core.screenshot.taker.py

Модуль используется для генерации графиков.

```
from pathlib import Path
from glob import glob
from paraview.simple import *

from foampostproc.utils import FileHandling


class Screenshot:
    @classmethod
    def take_screenshots(cls, foam_case_list, out: Path):
        for i, foam_case in enumerate(foam_case_list):
            cases = glob(str(foam_case.cases_dir.path / "*/"))
            for case in cases:
                foamcase_path = Path(case)
                foamcase = FileHandling.read_foamcase(foamcase_path)
                foamcase.MeshRegions = ['internalMesh']
                servermanager.Fetch(foamcase)
                view = CreateRenderView(foamcase)
                print(foamcase.TimestepValues)
                view.ViewTime = max(foamcase.TimestepValues)
                for j, cam_prop in enumerate(foam_case.cam_prop_list):
                    camera = view.GetActiveCamera()
                    camera.SetFocalPoint(cam_prop.focal_point.x, cam_prop.focal_point.y,
                                         cam_prop.focal_point.z)
                    camera.SetPosition(cam_prop.cam_position.x, cam_prop.cam_position.y,
                                       cam_prop.cam_position.z)
                    camera.SetViewUp(cam_prop.viewup.x, cam_prop.viewup.y,
                                     cam_prop.viewup.z)
                    camera.SetViewAngle(cam_prop.viewangle)

                    display = Show(foamcase, view)
                    ColorBy(display, ('POINTS', 'U'))
                    display.RescaleTransferFunctionToDataRange(True)
                    SaveScreenshot(str(out / f"view-{foamcase_path.name}-{i}-{j}.png"),
                                   view)
```

### A.2 foampostproc.core.controller.py

Модуль содержит класс Controller, который инкапсулирует основную логику работы с моделью.

```
from copy import deepcopy
from pathlib import Path

from PySide6 import QtWidgets as QtW, QtCore as QtC
from bson import ObjectId

from foampostproc.config import Config
from foampostproc.core.model import Point, FoamCase, CasesDir, CameraProps, CameraSlice
from foampostproc.core.screenshot.taker import Screenshot
from foampostproc.dao.dao import MongoFoamCaseDAO, MongoCameraPropsDAO, MongoCameraSliceDAO
from foampostproc.dao.daofactory import MongoDaoFactory
from foampostproc.dto.modelmapper import Mapper
from foampostproc.utils import SharedState


def handle_cases_add_button_clicked():
```

```
17      case_dir = CasesDir(ObjectId(), "")
18      foamcase = FoamCase(ObjectId(), f"case{len(SharedState.case_list) + 1}", case_dir, [], [])
19      SharedState.case_list.append(foamcase)
20      SharedState.m_widget.cases_control_list.list_.addItem(foamcase.name)
21      if len(SharedState.case_list) - 1 > 0:
22          SharedState.m_widget.cases_control_list.list_.setCurrentRow(len(SharedState.case_list) - 1)
23      handle_camera_props_add_button_clicked()
24
25
26  def handle_cases_rm_button_clicked():
27      list_ = SharedState.m_widget.cases_control_list.list_
28      for item in list_.selectedItems():
29          item_row = list_.row(item)
30          SharedState.cases_for_del.append(SharedState.case_list[item_row])
31          SharedState.case_list.pop(item_row)
32          list_.takeItem(item_row)
33      if len(SharedState.case_list) - 1 > 0:
34          list_.setCurrentRow(len(SharedState.case_list) - 1)
35
36
37  def handle_cases_item_selection():
38      case = _get_selected_case()
39      list_ = SharedState.m_widget.camera_props_control_list.list_
40      SharedState.m_widget.camera_props_params_form.clear()
41      list_.clear()
42      list_.addItems([prop.name for prop in case.cam_prop_list])
43      list_.setCurrentRow(0)
44      handle_camera_props_item_selection()
45
46      #
47      # list_ = SharedState.m_widget.slice_props_control_list.list_
48      # SharedState.m_widget.slice_params_form.clear()
49      # list_.clear()
50      # list_.addItems([prop.name for prop in case.slice_list])
51      # list_.setCurrentRow(0)
52
53      SharedState.m_widget.case_path_field.clear()
54
55      SharedState.m_widget.case_path_field.text_edit.setText(str(case.cases_dir.path))
56
57
58  def handle_generate_button_clicked():
59      Screenshot.take_screenshots(SharedState.case_list, Config.get_section("Paths")
60                                  .get_path("output"))
61
62
63  def handle_case_path_field_on_text_changed():
64      pass
65
66
67  def handle_camera_props_add_button_clicked():
68      case = _get_selected_case()
69      cam_prop = CameraProps(ObjectId(), f"camera{len(case.cam_prop_list) + 1}",
70                             Point(0, 0, 0), Point(0, 0, 0), 0, Point(0, 0, 0))
71      case.cam_prop_list.append(cam_prop)
72      SharedState.m_widget.camera_props_control_list.list_.addItem(cam_prop.name)
73      if len(case.cam_prop_list) - 1 > 0:
74          SharedState.m_widget.camera_props_control_list.list_\
75              .setCurrentRow(len(case.cam_prop_list) - 1)
76
77
78  def handle_camera_props_rm_button_clicked():
79      case = _get_selected_case()
80
81      list_ = SharedState.m_widget.camera_props_control_list.list_
82      for item in list_.selectedItems():
83          item_row = list_.row(item)
84          SharedState.cam_props_for_del.append(case.cam_prop_list[item_row])
85          case.cam_prop_list.pop(item_row)
86          list_.takeItem(item_row)
87      if len(case.cam_prop_list) - 1 > 0:
88          list_.setCurrentRow(len(case.cam_prop_list) - 1)
89
```

```
90
91  def handle_camera_props_item_selection():
92      # if len(_get_selected_case().slice_list) == 0 or len(_get_selected_case()\
93      # .cam_prop_list) == 0:
94      #     return
95
96      cam_prop = _get_selected_cam_prop()
97      if cam_prop is not None:
98          form = SharedState.m_widget.camera_props_params_form
99          form.position_field.text_edit1.setText(str(cam_prop.cam_position.x))
100         form.position_field.text_edit2.setText(str(cam_prop.cam_position.y))
101         form.position_field.text_edit3.setText(str(cam_prop.cam_position.z))
102
103         form.focal_point_field.text_edit1.setText(str(cam_prop.focal_point.x))
104         form.focal_point_field.text_edit2.setText(str(cam_prop.focal_point.y))
105         form.focal_point_field.text_edit3.setText(str(cam_prop.focal_point.z))
106
107         form.view_up_field.text_edit1.setText(str(cam_prop.viewup.x))
108         form.view_up_field.text_edit2.setText(str(cam_prop.viewup.y))
109         form.view_up_field.text_edit3.setText(str(cam_prop.viewup.z))
110
111         form.view_angle_field.text_edit.setText(str(cam_prop.viewangle))
112
113
114 def handle_slice_props_add_button_clicked():
115     case = _get_selected_case()
116     cam_slice = CameraSlice(ObjectId(), f"slice{len(case.slice_list)␣+␣1}",
117                             None, None, None)
118     case.slice_list.append(cam_slice)
119     SharedState.m_widget.slice_props_control_list.list_.addItem(cam_slice.name)
120     if len(case.slice_list) - 1 > 0:
121         SharedState.m_widget.slice_props_control_list.list_.setCurrentRow(
122             len(case.slice_list) - 1)
123
124
125 def handle_slice_props_rm_button_clicked():
126     case = _get_selected_case()
127
128     list_ = SharedState.m_widget.slice_props_control_list.list_
129     for item in list_.selectedItems():
130         item_row = list_.row(item)
131         SharedState.slices_for_del.append(case.slice_list[item_row])
132         case.slice_list.pop(item_row)
133         list_.takeItem(item_row)
134     if len(case.slice_list) - 1 > 0:
135         list_.setCurrentRow(len(case.slice_list) - 1)
136
137
138 def handle_slice_props_item_selection():
139     slice_ = _get_selected_slice_prop()
140
141     if slice_ is not None:
142         form = SharedState.m_widget.slice_params_form
143
144         x = "" if slice_.sl_x is None else str(slice_.sl_x.x)
145         y = "" if slice_.sl_x is None else str(slice_.sl_x.y)
146         z = "" if slice_.sl_x is None else str(slice_.sl_x.z)
147
148         form.x_slice_field.text_edit1.setText(x)
149         form.x_slice_field.text_edit2.setText(y)
150         form.x_slice_field.text_edit3.setText(z)
151
152         x = "" if slice_.sl_y is None else str(slice_.sl_y.x)
153         y = "" if slice_.sl_y is None else str(slice_.sl_y.y)
154         z = "" if slice_.sl_y is None else str(slice_.sl_y.z)
155
156         form.y_slice_field.text_edit1.setText(x)
157         form.y_slice_field.text_edit2.setText(y)
158         form.y_slice_field.text_edit3.setText(z)
159
160         x = "" if slice_.sl_z is None else str(slice_.sl_z.x)
161         y = "" if slice_.sl_z is None else str(slice_.sl_z.y)
162         z = "" if slice_.sl_z is None else str(slice_.sl_z.z)
```

```
163
164         form.z_slice_field.text_edit1.setText(x)
165         form.z_slice_field.text_edit2.setText(y)
166         form.z_slice_field.text_edit3.setText(z)
167
168
169 def handle_save_btn():
170     case = _get_selected_case()
171     case_path_field_text = SharedState.m_widget.case_path_field.text_edit.text()
172     case.cases_dir.path = Path(case_path_field_text)
173
174     cam_prop = _get_selected_cam_prop()
175     # slice_ = _get_selected_slice_prop()
176
177     cam_form = SharedState.m_widget.camera_props_params_form
178     cam_prop.cam_position = deepcopy(cam_form.position)
179     cam_prop.viewup = deepcopy(cam_form.view_up)
180     cam_prop.focal_point = deepcopy(cam_form.focal_point)
181     cam_prop.viewangle = float(cam_form.view_angle)
182
183     # slice_form = SharedState.m_widget.slice_params_form
184     # if slice_form.x_slice_field.text_edit1.text() == "" \
185     #         or slice_form.x_slice_field.text_edit2.text() == "" \
186     #         or slice_form.x_slice_field.text_edit3.text() == "":
187     #     slice_.sl_x = None
188     # else:
189     #     slice_.sl_x = slice_form.x_slice
190     #
191     # if slice_form.y_slice_field.text_edit1.text() == "" \
192     #         or slice_form.y_slice_field.text_edit2.text() == "" \
193     #         or slice_form.y_slice_field.text_edit3.text() == "":
194     #     slice_.sl_y = None
195     # else:
196     #     slice_.sl_y = slice_form.y_slice
197     #
198     # if slice_form.z_slice_field.text_edit1.text() == "" \
199     #         or slice_form.z_slice_field.text_edit2.text() == "" \
200     #         or slice_form.z_slice_field.text_edit3.text() == "":
201     #     slice_.sl_z = None
202     # else:
203     #     slice_.sl_z = slice_form.z_slice
204
205
206 def handle_reset_btn():
207     case_dtos = MongoDaoFactory().get_dao(MongoFoamCaseDAO).get_all()
208     cases = list(map(Mapper.map_foam_case_dto, case_dtos))
209     SharedState.case_list = cases
210     list_ = SharedState.m_widget.cases_control_list.list_
211     list_.clear()
212     list_.addItems([case.name for case in cases])
213     if len(cases) > 0:
214         list_.setCurrentRow(0)
215
216
217 def handle_save_db_btn():
218     cases = SharedState.case_list
219     for case in cases:
220         handle_save_btn()
221         MongoDaoFactory().get_dao(MongoFoamCaseDAO)\
222             .create_or_update(Mapper.map_foam_case_to_dto(case))
223
224     for case in SharedState.cases_for_del:
225         MongoDaoFactory().get_dao(MongoFoamCaseDAO).delete(str(case.idn))
226
227     for cam_prop in SharedState.cam_props_for_del:
228         MongoDaoFactory().get_dao(MongoCameraPropsDAO).delete(str(cam_prop.idn))
229
230     for sl in SharedState.slices_for_del:
231         MongoDaoFactory().get_dao(MongoCameraSliceDAO).delete(str(sl.idn))
232
233     SharedState.cam_props_for_del = []
234     SharedState.slices_for_del = []
235     SharedState.cases_for_del = []
```

```
236
237
238 def _get_selected_case_row():
239     selected_item = SharedState.m_widget.cases_control_list.list_.selectedItems()[0]
240     return SharedState.m_widget.cases_control_list.list_.row(selected_item)
241
242
243 def _get_selected_case():
244     row = _get_selected_case_row()
245     return SharedState.case_list[row]
246
247
248 def _get_selected_cam_prop_row():
249     selected_item = SharedState.m_widget.camera_props_control_list.list_\
250         .selectedItems()
251     if not selected_item:
252         return None
253     return SharedState.m_widget.camera_props_control_list\
254         .list_.row(selected_item[0])
255
256
257 def _get_selected_cam_prop():
258     case = _get_selected_case()
259     cam_row = _get_selected_cam_prop_row()
260     return None if cam_row is None else case.cam_prop_list[cam_row]
261
262
263 def _get_selected_slice_prop_row():
264     selected_item = SharedState.m_widget.slice_props_control_list.list_.selectedItems()
265     if not selected_item:
266         return None
267     return SharedState.m_widget.slice_props_control_list.list_.row(selected_item[0])
268
269
270 def _get_selected_slice_prop():
271     case = _get_selected_case()
272     row = _get_selected_slice_prop_row()
273     return None if row is None else case.slice_list[row]
274
275
276 handlers = {
277     "handle_cases_add_button_clicked": handle_cases_add_button_clicked,
278     "handle_cases_rm_button_clicked": handle_cases_rm_button_clicked,
279     "handle_cases_item_selection": handle_cases_item_selection,
280     "handle_generate_button_clicked": handle_generate_button_clicked,
281     "handle_case_path_field_on_text_changed": handle_case_path_field_on_text_changed,
282     "handle_camera_props_add_button_clicked": handle_camera_props_add_button_clicked,
283     "handle_camera_props_rm_button_clicked": handle_camera_props_rm_button_clicked,
284     "handle_camera_props_item_selection": handle_camera_props_item_selection,
285     "handle_slice_props_add_button_clicked": handle_slice_props_add_button_clicked,
286     "handle_slice_props_rm_button_clicked": handle_slice_props_rm_button_clicked,
287     "handle_slice_props_item_selection": handle_slice_props_item_selection,
288     "handle_save_btn": handle_save_btn,
289     "handle_reset_btn": handle_reset_btn,
290     "handle_save_db_btn": handle_save_db_btn
291 }
```

## A.3 foampostproc.core.model.py

Модуль состоит из классов, описывающих структуру данных приложения.

```
1 from pathlib import Path
2 from typing import List
3
4
5 class Point:
6     def __init__(self, x: float, y: float, z: float):
7         self.x = x
```

```
8           self.y = y
9           self.z = z
10
11      def __repr__(self):
12          return f"<Point: {self.x} {self.y} {self.z}>"
13
14
15  class CameraProps:
16      def __init__(self, idn, name: str, focal_point: Point,
17                   cam_position: Point, viewangle: int, viewup: Point):
18          self.idn = idn
19          self.name = name
20          self.focal_point = focal_point
21          self.cam_position = cam_position
22          self.viewup = viewup
23          self.viewangle = viewangle
24
25
26  class CameraSlice(object):
27      def __init__(self, idn, name: str,
28                   sl_x: Point = None, sl_y: Point = None, sl_z: Point = None):
29          self.idn = idn
30          self.name = name
31          self.sl_x = sl_x
32          self.sl_y = sl_y
33          self.sl_z = sl_z
34
35
36  class CasesDir(object):
37      def __init__(self, idn, path: str):
38          self.idn = idn
39          self.path = Path(path)
40
41
42  class FoamCase(object):
43      def __init__(self, idn, name: str, case_dir: CasesDir,
44                   cam_prop_list: List[CameraProps], slice_list: List[CameraSlice]):
45          self.idn = idn
46          self.name = name
47          self.cases_dir = case_dir
48          self.cam_prop_list = cam_prop_list
49          self.slice_list = slice_list
```

## A.4 foampostproc.core.view.py

В данному модуле содержится код графического интерфейса пользователя.

```
1   import sys
2   from typing import List, Callable
3
4   from PySide6 import QtWidgets as QtW, QtCore as QtC
5   from PySide6.QtWidgets import QAbstractItemView
6
7   from foampostproc.core.model import Point
8
9
10  class ControlList(QtW.QWidget):
11      ADD_BUTTON_TEXT = "Добавить"
12      RM_BUTTON_TEXT = "Удалить"
13      LABEL_SPACING = 4
14      ALTERNATING_ROW_COLORS = True
15
16      def __init__(self, label_text: str, list_items: List[str],
17                   on_add_button_clicked: Callable,
18                   on_rm_button_clicked: Callable,
19                   on_item_selection: Callable):
```

```
20          super().__init__()
21
22          self.on_add_button_clicked = on_add_button_clicked
23          self.on_rm_button_clicked = on_rm_button_clicked
24          self.on_item_selection = on_item_selection
25
26          self.header = self._create_header(label_text)
27          self.list_ = self._create_list(list_items)
28
29          self.setLayout(QtW.QVBoxLayout())
30          self.layout().addWidget(self.header)
31          self.layout().addSpacing(self.LABEL_SPACING)
32          self.layout().addWidget(self.list_)
33
34      def _create_header(self, label_text: str) -> QtW.QWidget:
35          header_widget = QtW.QWidget()
36          header_layout = QtW.QHBoxLayout()
37          header_widget.setLayout(header_layout)
38
39          header_widget.label = QtW.QLabel(label_text)
40
41          header_widget.add_button = Button(self.ADD_BUTTON_TEXT,
42                                            self.handle_header_add_button)
43          header_widget.remove_button = Button(self.RM_BUTTON_TEXT,
44                                               self.handle_header_remove_button)
45
46          header_layout.addWidget(header_widget.label)
47          header_layout.addSpacing(4)
48          header_layout.addWidget(header_widget.add_button)
49          header_layout.addWidget(header_widget.remove_button)
50
51          return header_widget
52
53      def _create_list(self, list_items: List[str]) -> QtW.QListWidget:
54          list_ = QtW.QListWidget()
55          list_.setAlternatingRowColors(self.ALTERNATING_ROW_COLORS)
56          list_.setSelectionMode(QAbstractItemView.SingleSelection)
57          list_.addItems(list_items)
58          list_.itemSelectionChanged.connect(self.handle_item_selection_changed)
59          return list_
60
61      @QtC.Slot()
62      def handle_header_add_button(self):
63          self.on_add_button_clicked()
64
65      @QtC.Slot()
66      def handle_header_remove_button(self):
67          self.on_rm_button_clicked()
68
69      @QtC.Slot()
70      def handle_item_selection_changed(self):
71          self.on_item_selection()
72
73
74  class TextEdit(QtW.QWidget):
75      LABEL_SPACING = 4
76
77      def __init__(self, label_text: str,
78                   handle_on_text_changed: Callable, edit_text: str = ""):
79          super().__init__()
80          self.handle_on_text_changed = handle_on_text_changed
81
82          self.label = QtW.QLabel(label_text)
83          self.text_edit = QtW.QLineEdit(edit_text)
84          self.text_edit.textChanged.connect(self.handle_on_text_changed)
85
86          self.setLayout(QtW.QHBoxLayout())
87          self.layout().addWidget(self.label)
88          self.layout().addSpacing(self.LABEL_SPACING)
89          self.layout().addWidget(self.text_edit)
90
91      @QtC.Slot()
92      def _handle_on_text_changed(self):
```

```
93          self.handle_on_text_changed()
94
95      def clear(self):
96          self.text_edit.clear()
97
98
99  class PointTextEdit(QtW.QWidget):
100     SPACING = 4
101
102     def __init__(self, label_text: str,
103                  handle_on_text_changed1: Callable,
104                  handle_on_text_changed2: Callable,
105                  handle_on_text_changed3: Callable,
106                  text1: str = "",
107                  text2: str = "",
108                  text3: str = ""
109                  ):
110         super().__init__()
111         self.handle_on_text_changed1 = handle_on_text_changed1
112         self.handle_on_text_changed2 = handle_on_text_changed2
113         self.handle_on_text_changed3 = handle_on_text_changed3
114         self.label = QtW.QLabel(label_text)
115         self.text_edit1 = QtW.QLineEdit(text1)
116         self.text_edit1.textChanged.connect(self._handle_on_text_changed1)
117         self.text_edit2 = QtW.QLineEdit(text2)
118         self.text_edit2.textChanged.connect(self._handle_on_text_changed2)
119         self.text_edit3 = QtW.QLineEdit(text3)
120         self.text_edit3.textChanged.connect(self._handle_on_text_changed3)
121
122         self.setLayout(QtW.QHBoxLayout())
123         self.layout().addWidget(self.label)
124         self.layout().addSpacing(self.SPACING)
125         self.layout().addWidget(self.text_edit1)
126         self.layout().addSpacing(self.SPACING)
127         self.layout().addWidget(self.text_edit2)
128         self.layout().addSpacing(self.SPACING)
129         self.layout().addWidget(self.text_edit3)
130
131     @QtC.Slot()
132     def _handle_on_text_changed1(self):
133         self.handle_on_text_changed1(self.text_edit1.text())
134
135     @QtC.Slot()
136     def _handle_on_text_changed2(self):
137         self.handle_on_text_changed2(self.text_edit2.text())
138
139     @QtC.Slot()
140     def _handle_on_text_changed3(self):
141         self.handle_on_text_changed3(self.text_edit3.text())
142
143     def clear(self):
144         self.text_edit1.clear()
145         self.text_edit2.clear()
146         self.text_edit3.clear()
147
148
149 class CameraParamsForm(QtW.QWidget):
150     POSITION_FIELD_TEXT = "␣␣␣␣␣␣␣␣␣Позиция"
151     FOCAL_POINT_FIELD_TEXT = "Точка␣фокуса"
152     VIEW_ANGLE_FIELD_TEXT = "␣␣Угол␣обзора"
153     VIEW_UP_TEXT = "Наклон"
154
155     SPACING = 4
156
157     def __init__(self):
158         super().__init__()
159         self.setLayout(QtW.QVBoxLayout())
160         self.position = Point(0, 0, 0)
161         self.focal_point = Point(0, 0, 0)
162         self.view_up = Point(0, 0, 0)
163         self._view_angle = [0]
164
165         self.position_field \
```

```
166             = PointTextEdit(self.POSITION_FIELD_TEXT,
167                             lambda text: self.position
168                             .__setattr__("x", 0 if text == "" else float(text)),
169                             lambda text: self.position
170                             .__setattr__("y", 0 if text == "" else float(text)),
171                             lambda text: self.position
172                             .__setattr__("z", 0 if text == "" else float(text)))
173
174         self.focal_point_field \
175             = PointTextEdit(self.FOCAL_POINT_FIELD_TEXT,
176                             lambda text: self.focal_point
177                             .__setattr__("x", 0 if text == "" else float(text)),
178                             lambda text: self.focal_point
179                             .__setattr__("y", 0 if text == "" else float(text)),
180                             lambda text: self.focal_point
181                             .__setattr__("z", 0 if text == "" else float(text)))
182
183         self.view_up_field \
184             = PointTextEdit(self.VIEW_ANGLE_FIELD_TEXT,
185                             lambda text: self.view_up.__setattr__(
186                                 "x", 0 if text == "" else float(text)),
187                             lambda text: self.view_up.__setattr__(
188                                 "y", 0 if text == "" else float(text)),
189                             lambda text: self.view_up.__setattr__(
190                                 "z", 0 if text == "" else float(text)))
191
192         self.view_angle_field \
193             = TextEdit(self.VIEW_UP_TEXT, lambda text: self._view_angle
194                        .__setitem__(0, 0 if text == "" else float(text)))
195
196         self.setLayout(QtW.QVBoxLayout())
197         self.layout().addWidget(self.position_field)
198         self.layout().addSpacing(self.SPACING)
199         self.layout().addWidget(self.focal_point_field)
200         self.layout().addSpacing(self.SPACING)
201         self.layout().addWidget(self.view_up_field)
202         self.layout().addSpacing(self.SPACING)
203         self.layout().addWidget(self.view_angle_field)
204
205     @property
206     def view_angle(self):
207         return self._view_angle[0]
208
209     @view_angle.setter
210     def view_angle(self, value):
211         self._view_angle[0] = value
212
213     def clear(self):
214         self.position_field.clear()
215         self.focal_point_field.clear()
216         self.view_up_field.clear()
217         self.view_angle_field.clear()
218
219
220 class SliceParamsForm(QtW.QWidget):
221     X_SLICE_LABEL_TEXT = "Срез X"
222     Y_SLICE_LABEL_TEXT = "Срез Y"
223     Z_SLICE_LABEL_TEXT = "Срез Z"
224     X_LABEL_TEXT = "x"
225     Y_LABEL_TEXT = "y"
226     Z_LABEL_TEXT = "z"
227
228     SPACING = 4
229
230     def __init__(self):
231         super().__init__()
232         self.setLayout(QtW.QVBoxLayout())
233
234         self.x_slice = Point(0, 0, 0)
235         self.x_slice_field \
236             = PointTextEdit(self.X_SLICE_LABEL_TEXT,
237                             lambda text: self.x_slice.__setattr__(
238                                 "x", 0 if text == "" else float(text)),
```

```
239                             lambda text: self.x_slice.__setattr__(
240                                 "y", 0 if text == "" else float(text)),
241                             lambda text: self.x_slice.__setattr__(
242                                 "z", 0 if text == "" else float(text)))
243
244         self.y_slice = Point(0, 0, 0)
245         self.y_slice_field \
246             = PointTextEdit(self.Y_SLICE_LABEL_TEXT,
247                             lambda text: self.y_slice.__setattr__(
248                                 "x", 0 if text == "" else float(text)),
249                             lambda text: self.y_slice.__setattr__(
250                                 "y", 0 if text == "" else float(text)),
251                             lambda text: self.y_slice.__setattr__(
252                                 "z", 0 if text == "" else float(text)))
253
254         self.z_slice = Point(0, 0, 0)
255         self.z_slice_field \
256             = PointTextEdit(self.Z_SLICE_LABEL_TEXT,
257                             lambda text: self.z_slice.__setattr__(
258                                 "x", 0 if text == "" else float(text)),
259                             lambda text: self.z_slice.__setattr__(
260                                 "y", 0 if text == "" else float(text)),
261                             lambda text: self.z_slice.__setattr__(
262                                 "z", 0 if text == "" else float(text)))
263
264         self.setLayout(QtW.QVBoxLayout())
265         self.layout().addWidget(self.x_slice_field)
266         self.layout().addSpacing(self.SPACING)
267         self.layout().addWidget(self.y_slice_field)
268         self.layout().addSpacing(self.SPACING)
269         self.layout().addWidget(self.z_slice_field)
270
271     def clear(self):
272         self.x_slice_field.clear()
273         self.y_slice_field.clear()
274         self.z_slice_field.clear()
275
276
277 class Button(QtW.QPushButton):
278     def __init__(self, button_text: str, handle_on_click: Callable):
279         super().__init__(button_text)
280         self.handle_on_click = handle_on_click
281         self.clicked.connect(self._handle_on_click)
282
283     @QtC.Slot()
284     def _handle_on_click(self):
285         self.handle_on_click()
286
287
288 class SaveResetButtonGroup(QtW.QWidget):
289     def __init__(self, save_btn_txt: str, save_db_btn_txt: str,
290                  handle_save: Callable, handle_save_db: Callable,
291                  handle_reset: Callable,
292                  reset_btn_text: str):
293         super().__init__()
294
295         self.handle_save = handle_save
296         self.handle_save_db = handle_save_db
297         self.handle_reset = handle_reset
298
299         self.setLayout(QtW.QHBoxLayout())
300         self.button_save = Button(save_btn_txt, handle_save)
301         self.button_save_db = Button(save_db_btn_txt, handle_save_db)
302         self.button_reset = Button(reset_btn_text, handle_reset)
303
304         self.layout().addWidget(self.button_reset)
305         self.layout().addSpacing(4)
306         self.layout().addWidget(self.button_save)
307         self.layout().addSpacing(4)
308         self.layout().addWidget(self.button_save_db)
309
310     @QtC.Slot()
311     def _handle_save(self):
```

```
312             self.handle_save()
313
314         @QtC.Slot()
315         def _handle_save_db(self):
316             self.handle_save_db()
317
318         @QtC.Slot()
319         def _handle_reset(self):
320             self.handle_reset()
321
322
323     class MainWidget(QtW.QWidget):
324         SPACING = 4
325         CASES_LABEL = "Примеры"
326         GENERATE_BTN_TEXT = "Генерация"
327         CAMERA_PROPS_LABEL = "Свойсва камеры"
328         SLICE_PROPS_LABEL = "Срезы"
329         CASE_PATH_FIELD_TEXT = "Путь"
330         SAVE_BTN_TEXT = "Сохранить"
331         SAVE_DB_BTN_TEXT = "Сохранить в БД"
332         RESET_BTN_TEXT = "Сбросить"
333
334         def __init__(self, case_list: List[str],
335                      handle_cases_add_button_clicked,
336                      handle_cases_rm_button_clicked,
337                      handle_cases_item_selection,
338                      handle_generate_button_clicked,
339                      handle_case_path_field_on_text_changed,
340                      handle_camera_props_add_button_clicked,
341                      handle_camera_props_rm_button_clicked,
342                      handle_camera_props_item_selection,
343                      handle_slice_props_add_button_clicked,
344                      handle_slice_props_rm_button_clicked,
345                      handle_slice_props_item_selection,
346                      handle_save_btn,
347                      handle_save_db_btn,
348                      handle_reset_btn,
349                      ):
350             super().__init__()
351
352             # first col ---------------------------------------------------
353             self.first_col_widget = QtW.QWidget()
354             self.first_col_widget.setLayout(QtW.QVBoxLayout())
355             self.cases_control_list = ControlList(self.CASES_LABEL, case_list,
356                                                   handle_cases_add_button_clicked,
357                                                   handle_cases_rm_button_clicked,
358                                                   handle_cases_item_selection)
359             self.generate_button = Button(self.GENERATE_BTN_TEXT, handle_generate_button_clicked)
360             self.first_col_widget.layout().addWidget(self.cases_control_list)
361             self.first_col_widget.layout().addWidget(self.generate_button)
362
363             # second col --------------------------------------------------
364             self.handle_case_path_field_on_text_changed \
365                 = handle_case_path_field_on_text_changed
366             self.case_path_field = TextEdit(self.CASE_PATH_FIELD_TEXT,
367                                             self.handle_case_path_field_on_text_changed)
368
369             handle_camera_props_add_button_clicked = handle_camera_props_add_button_clicked
370             handle_camera_props_rm_button_clicked = handle_camera_props_rm_button_clicked
371             handle_camera_props_item_selection = handle_camera_props_item_selection
372             self.camera_props_control_list \
373                 = ControlList(self.CAMERA_PROPS_LABEL, [],
374                               handle_camera_props_add_button_clicked,
375                               handle_camera_props_rm_button_clicked,
376                               handle_camera_props_item_selection)
377             self.camera_props_params_form = CameraParamsForm()
378
379             handle_slice_props_add_button_clicked = handle_slice_props_add_button_clicked
380             handle_slice_props_rm_button_clicked = handle_slice_props_rm_button_clicked
381             handle_slice_props_item_selection = handle_slice_props_item_selection
382             self.slice_props_control_list = ControlList(self.SLICE_PROPS_LABEL, [],
383                                                         handle_slice_props_add_button_clicked,
384                                                         handle_slice_props_rm_button_clicked,
```

```
385                                                     handle_slice_props_item_selection)
386         self.slice_params_form = SliceParamsForm()
387
388         handle_save_btn = handle_save_btn
389         handle_save_db_btn = handle_save_db_btn
390         handle_reset_btn = handle_reset_btn
391         self.save_reset_btn_group \
392             = SaveResetButtonGroup(self.SAVE_BTN_TEXT, self.SAVE_DB_BTN_TEXT,
393                                    handle_save_btn, handle_save_db_btn,
394                                    handle_reset_btn, self.RESET_BTN_TEXT)
395
396         self.second_col_widget = QtW.QWidget()
397         self.second_col_widget.setLayout(QtW.QVBoxLayout())
398         self.second_col_widget.layout().addWidget(self.case_path_field)
399         self.second_col_widget.layout().addSpacing(self.SPACING)
400
401         self.first_row_second_col_widget = QtW.QWidget()
402         self.first_row_second_col_widget.setLayout(QtW.QHBoxLayout())
403         self.first_row_second_col_widget.layout()\
404             .addWidget(self.camera_props_control_list)
405         self.first_row_second_col_widget.layout().addSpacing(self.SPACING)
406         self.first_row_second_col_widget.layout()\
407             .addWidget(self.camera_props_params_form)
408         self.second_col_widget.layout().addWidget(self.first_row_second_col_widget)
409
410         self.second_row_second_col_widget = QtW.QWidget()
411         self.second_row_second_col_widget.setLayout(QtW.QVBoxLayout())
412         # self.second_row_second_col_widget.layout()\
413         #     .addWidget(self.slice_props_control_list)
414         # self.second_row_second_col_widget.layout()\
415         #     .addWidget(self.slice_params_form)
416         self.second_col_widget.layout().addWidget(self.second_row_second_col_widget)
417
418         self.second_col_widget.layout().addWidget(self.save_reset_btn_group)
419
420         self.setLayout(QtW.QHBoxLayout())
421         self.layout().addWidget(self.first_col_widget)
422         self.layout().addWidget(self.second_col_widget)
```

## A.5 foampostproc.dao.dao.py

Модуль, хранящий классы DAO-слоя программы.

```
1  from abc import ABC, abstractmethod
2  from typing import Any, List
3  from foampostproc.dao.daofactory import MongoDaoFactory
4  from foampostproc.dto.dto import FoamCaseDTO, CasesDirDTO, CameraPropsDTO, SliceDTO
5
6
7  class AbstractDAO(ABC):
8      def __init__(self, collection_conn):
9          self._connection = collection_conn
10
11     @property
12     def connection(self):
13         return self._connection
14
15     @abstractmethod
16     def create(self, obj: Any) -> Any:
17         pass
18
19     def create_or_update(self, obj: Any) -> Any:
20         pass
21
22     @abstractmethod
23     def read(self, key: str) -> Any:
24         pass
25
```

```
26      @abstractmethod
27      def update(self, obj: Any):
28          pass
29
30      @abstractmethod
31      def delete(self, key: str):
32          pass
33
34      @abstractmethod
35      def get_all(self) -> List[Any]:
36          pass
37
38
39  class MongoAbstractDAO(AbstractDAO, ABC):
40      def delete(self, key: str) -> Any:
41          try:
42              self.connection.delete_one({"_id": key})
43          except Exception:
44              raise RuntimeError("Something went wrong with object deletion")
45
46      def get_all(self) -> List[Any]:
47          db_obj_list = self.connection.find()
48          return [self.read(db_obj["_id"]) for db_obj in db_obj_list]
49
50
51  class MongoFoamCaseDAO(MongoAbstractDAO):
52      def create(self, dto_obj: FoamCaseDTO):
53          try:
54              camera_props_ids = [str(prop._id) for prop in dto_obj.camera_props]
55              camera_slices_ids = [str(sl._id) for sl in dto_obj.camera_slices]
56              cases_dir_id = str(dto_obj.cases_dir._id)
57              _id = str(dto_obj._id)
58
59              if self.connection.count_documents({"_id": _id}) == 0:
60                  self.connection.insert_one({
61                      "_id": _id,
62                      "name": dto_obj.name,
63                      "cases_dir": cases_dir_id,
64                      "camera_props": camera_props_ids,
65                      "camera_slices": camera_slices_ids
66                  })
67
68              camera_props_dao = MongoDaoFactory().get_dao(MongoCameraPropsDAO)
69              for dto_camera_prop in dto_obj.camera_props:
70                  camera_props_dao.create(dto_camera_prop)
71
72              camera_slices_dao = MongoDaoFactory().get_dao(MongoCameraSliceDAO)
73              for dto_camera_slice in dto_obj.camera_slices:
74                  camera_slices_dao.create(dto_camera_slice)
75
76              MongoDaoFactory().get_dao(MongoCaseDirDAO).create(dto_obj.cases_dir)
77              return self.connection.count_documents({"_id": _id}) == 0
78          except Exception:
79              raise RuntimeError("Something went wrong with object creation")
80
81      def read(self, key: str) -> FoamCaseDTO:
82          try:
83              obj_dct = self.connection.find_one({"_id": key})
84              dao_fact = MongoDaoFactory()
85              cases_dir = dao_fact.get_dao(MongoCaseDirDAO).read(obj_dct["cases_dir"])
86              camera_props = [dao_fact.get_dao(MongoCameraPropsDAO).read(prop)
87                              for prop in obj_dct["camera_props"]]
88              camera_slices = [dao_fact.get_dao(MongoCameraSliceDAO).read(sl)
89                               for sl in obj_dct["camera_slices"]]
90              return FoamCaseDTO(cases_dir, camera_props, camera_slices,
91                                 obj_dct["name"], obj_dct["_id"])
92          except Exception:
93              raise RuntimeError("Something went wrong with object reading")
94
95      def update(self, dto_obj: FoamCaseDTO):
96          try:
97              camera_props_ids = [str(prop._id) for prop in dto_obj.camera_props]
98              camera_slices_ids = [str(sl._id) for sl in dto_obj.camera_slices]
```

```
99              cases_dir_id = str(dto_obj.cases_dir._id)
100             print(camera_props_ids, dto_obj._id)
101             self.connection.update_one({"_id": str(dto_obj._id)},
102                                        {"$set": {
103                                            "name": dto_obj.name,
104                                            "cases_dir": cases_dir_id,
105                                            "camera_props": camera_props_ids,
106                                            "camera_slices": camera_slices_ids
107                                        }})
108
109             camera_props_dao = MongoDaoFactory().get_dao(MongoCameraPropsDAO)
110             for dto_camera_prop in dto_obj.camera_props:
111                 camera_props_dao.create(dto_camera_prop)
112             camera_slices_dao = MongoDaoFactory().get_dao(MongoCameraSliceDAO)
113             for dto_camera_slice in dto_obj.camera_slices:
114                 camera_slices_dao.create(dto_camera_slice)
115             MongoDaoFactory().get_dao(MongoCaseDirDAO).create(dto_obj.cases_dir)
116         except Exception:
117             raise RuntimeError("Something went wrong with object updating")
118
119     def create_or_update(self, dto_obj: FoamCaseDTO):
120         result = self.create(dto_obj)
121         if not result:
122             self.update(dto_obj)
123             camera_props_dao = MongoDaoFactory().get_dao(MongoCameraPropsDAO)
124             for dto_camera_prop in dto_obj.camera_props:
125                 camera_props_dao.create_or_update(dto_camera_prop)
126             camera_slices_dao = MongoDaoFactory().get_dao(MongoCameraSliceDAO)
127             for dto_camera_slice in dto_obj.camera_slices:
128                 camera_slices_dao.create(dto_camera_slice)
129             MongoDaoFactory().get_dao(MongoCaseDirDAO) \
130                 .create_or_update(dto_obj.cases_dir)
131
132
133 class MongoCaseDirDAO(MongoAbstractDAO):
134     def create(self, dto_obj: CasesDirDTO):
135         try:
136             if self.connection.count_documents({"_id": str(dto_obj._id)}) == 0:
137                 self.connection.insert_one({"_id": str(dto_obj._id),
138                                             "cases_path": dto_obj.cases_path})
139                 return True
140             return False
141         except Exception:
142             raise RuntimeError("Something went wrong with object creation")
143
144     def read(self, key: str) -> CasesDirDTO:
145         try:
146             db_obj = self.connection.find_one({"_id": key})
147             return CasesDirDTO(**db_obj)
148         except Exception:
149             raise RuntimeError("Something went wrong with object reading")
150
151     def update(self, dto_obj: CasesDirDTO):
152         try:
153             self.connection.update_one({"_id": str(dto_obj._id)},
154                                        {"$set": {
155                                            "cases_path": dto_obj.cases_path
156                                        }})
157         except Exception:
158             raise RuntimeError("Something went wrong with object updating")
159
160     def create_or_update(self, dto_obj: CasesDirDTO):
161         result = self.create(dto_obj)
162         if not result:
163             self.update(dto_obj)
164
165
166 class MongoCameraPropsDAO(MongoAbstractDAO):
167     def create(self, dto_obj: CameraPropsDTO):
168         try:
169             if self.connection.count_documents({"_id": str(dto_obj._id)}) == 0:
170                 self.connection.insert_one({
171                     "_id": str(dto_obj._id),
```

```
172                     "name": dto_obj.name,
173                     "focal_point": dto_obj.focal_point.__dict__,
174                     "cam_position": dto_obj.cam_position.__dict__,
175                     "viewangle": dto_obj.viewangle,
176                     "viewup": dto_obj.viewup.__dict__,
177                 })
178                 return True
179             return False
180         except Exception:
181             raise RuntimeError("Something went wrong with object creation")
182 
183     def read(self, key: str) -> CameraPropsDTO:
184         try:
185             db_obj = self.connection.find_one({"_id": key})
186             return CameraPropsDTO(**db_obj)
187         except Exception:
188             raise RuntimeError("Something went wrong with object reading")
189 
190     def update(self, dto_obj: CameraPropsDTO):
191         try:
192             self.connection.update_one({"_id": str(dto_obj._id)},
193                                        {"$set": {
194                                            "name": dto_obj.name,
195                                            "focal_point": dto_obj.focal_point.__dict__,
196                                            "cam_position": dto_obj.cam_position.__dict__,
197                                            "viewangle": dto_obj.viewangle,
198                                            "viewup": dto_obj.viewup.__dict__
199                                        }})
200         except Exception:
201             raise RuntimeError("Something went wrong with object updating")
202 
203     def create_or_update(self, dto_obj: CameraPropsDTO):
204         result = self.create(dto_obj)
205         if not result:
206             self.update(dto_obj)
207 
208 
209 class MongoCameraSliceDAO(MongoAbstractDAO):
210     def create(self, dto_obj: SliceDTO):
211         try:
212             if self.connection.count_documents({"_id": str(dto_obj._id)}) == 0:
213                 self.connection.insert_one({
214                     "_id": str(dto_obj._id),
215                     "name": dto_obj.name,
216                     "sl_x": None if dto_obj.sl_x is None else dto_obj.sl_x.__dict__,
217                     "sl_y": None if dto_obj.sl_y is None else dto_obj.sl_y.__dict__,
218                     "sl_z": None if dto_obj.sl_z is None else dto_obj.sl_z.__dict__
219                 })
220                 return True
221             return False
222         except Exception:
223             raise RuntimeError("Something went wrong with object creation")
224 
225     def read(self, key: str) -> SliceDTO:
226         try:
227             return SliceDTO(**self.connection.find_one({"_id": key}))
228         except Exception:
229             raise RuntimeError("Something went wrong with object reading")
230 
231     def update(self, dto_obj: SliceDTO):
232         try:
233             self.connection.update_one(
234                 {"_id": str(dto_obj._id)},
235                 {"$set": {
236                     "name": dto_obj.name,
237                     "sl_x": None if dto_obj.sl_x is None else dto_obj.sl_x.__dict__,
238                     "sl_y": None if dto_obj.sl_y is None else dto_obj.sl_y.__dict__,
239                     "sl_z": None if dto_obj.sl_z is None else dto_obj.sl_z.__dict__
240                 }})
241         except Exception:
242             raise RuntimeError("Something went wrong with object updating")
243 
244     def create_or_update(self, dto_obj: SliceDTO):
```

```
245             result = self.create(dto_obj)
246             if not result:
247                 self.update(dto_obj)
```

## A.6 foampostproc.dao.daofactory.py

Данный модуль хранит класс-фабрику для классов DAO-слоя.

```
1   from abc import ABC, abstractmethod
2   from pymongo import MongoClient
3
4   from foampostproc.config import Config
5
6
7   class DaoFactory(ABC):
8       @abstractmethod
9       def _get_connection(self):
10          pass
11
12      @abstractmethod
13      def get_dao(self, connection, dao_class):
14          pass
15
16
17  class MongoDaoFactory(DaoFactory):
18      LOGIN = Config.get_section("DataBaseUser").get("login")
19      PASSWORD = Config.get_section("DataBaseUser").get("password")
20      DB_PROJ_NAME = Config.get_section("DataBaseUser").get("db_proj_name")
21      DB_CONNECT_LINK = f"mongodb+srv://{LOGIN}:{PASSWORD}@cluster0.ecqqe.mongodb.net/" \
22                        f"{DB_PROJ_NAME}?retryWrites=true&w=majority"
23
24      def _get_connection(self):
25          cluster = MongoClient(self.DB_CONNECT_LINK)
26          return cluster.foampostproc_db
27
28      map_collection = {
29          "MongoFoamCaseDAO": "foamcase",
30          "MongoCaseDirDAO": "casesdir",
31          "MongoCameraSliceDAO": "cameraslice",
32          "MongoCameraPropsDAO": "cameraprops"
33      }
34
35      def _get_collection(self, connection, dao_class):
36          return connection[self.map_collection[dao_class.__name__]]
37
38      def get_dao(self, dao_class, connection=None):
39          if connection is None:
40              connection = self._get_connection()
41          collection = self._get_collection(connection, dao_class)
42          return dao_class(collection)
```

## A.7 foampostproc.dto.dto.py

Этот модуль хранит DTO-классы.

```
1   from typing import List, Dict, Optional
2
3   from bson import ObjectId
4
5
6   class FoamCaseDTO(object):
7       def __init__(self, cases_dir: 'CasesDirDTO',
8                    camera_props: List['CameraPropsDTO'],
```

```
9                   camera_slices: List['SliceDTO'],
10                  name,
11                  _id=None, ):
12          if _id is None:
13              _id = ObjectId()
14          self._id = ObjectId(_id)
15          self.cases_dir = cases_dir
16          self.camera_props = camera_props
17          self.camera_slices = camera_slices
18          self.name = name
19
20      @staticmethod
21      def parse(d: Dict):
22          try:
23              res = FoamCaseDTO(**d)
24          except Exception:
25              res = None
26          return res
27
28
29  class CasesDirDTO(object):
30      def __init__(self, cases_path: str, _id=None):
31          if _id is None:
32              _id = ObjectId()
33          self._id = ObjectId(_id)
34          self.cases_path = cases_path
35
36      @staticmethod
37      def parse(d: Dict) -> Optional['CasesDirDTO']:
38          try:
39              res = CasesDirDTO(**d)
40          except Exception:
41              res = None
42          return res
43
44
45  class PointDTO(object):
46      def __init__(self, x: float, y: float, z: float):
47          self.x = x
48          self.y = y
49          self.z = z
50
51      @staticmethod
52      def parse(d: Dict) -> Optional['PointDTO']:
53          try:
54              res = PointDTO(**d)
55          except Exception:
56              res = None
57          return res
58
59
60  class CameraPropsDTO(object):
61      def __init__(self, focal_point: PointDTO, cam_position: PointDTO,
62                   viewangle: int, viewup: PointDTO, name, _id=None):
63          if _id is None:
64              _id = ObjectId()
65          self._id = ObjectId(_id)
66          self.name = name
67          self.focal_point = focal_point
68          self.cam_position = cam_position
69          self.viewangle = viewangle
70          self.viewup = viewup
71
72      @staticmethod
73      def parse(d: Dict) -> Optional['CameraPropsDTO']:
74          try:
75              res = CameraPropsDTO(**d)
76          except Exception:
77              res = None
78          return res
79
80
81  class SliceDTO(object):
```

```
82      def __init__(self, name, sl_x: PointDTO = None,
83                   sl_y: PointDTO = None, sl_z: PointDTO = None, _id=None):
84          if _id is None:
85              _id = ObjectId()
86          self._id = ObjectId(_id)
87          self.name = name
88          self.sl_x = sl_x
89          self.sl_y = sl_y
90          self.sl_z = sl_z
91
92      @staticmethod
93      def parse(d: Dict) -> Optional['SliceDTO']:
94          try:
95              res = SliceDTO(**d)
96          except Exception:
97              res = None
98          return res
99
100
101 parse_functions = [FoamCaseDTO.parse, CasesDirDTO.parse,
102                    PointDTO.parse, CameraPropsDTO.parse, SliceDTO.parse]
103
104
105 def parse_config_json_hook(dct: Dict):
106     res = None
107     for parse in parse_functions:
108         res = parse(dct)
109         if res is not None:
110             break
111
112     if res is None:
113         raise RuntimeError("Can't parse config file")
114
115     return res
```

## A.8 foampostproc.dto.modelmapper.py

Модуль содержит функции, которые отображают DTO-объекты в соответствующие им классы модели и наоборот.

```
1  from foampostproc.dto.dto import CasesDirDTO, PointDTO, SliceDTO, CameraPropsDTO, FoamCaseDTO
2  from foampostproc.core.model import Point, CameraSlice, CameraProps, FoamCase, CasesDir
3
4
5  class Mapper:
6      @classmethod
7      def map_foam_case_dto(cls, foam_case: FoamCaseDTO) -> FoamCase:
8          cam_props = [cls.map_camera_props_dto(cam_prop) for cam_prop in foam_case.camera_props]
9          slices_ = []
10         for sl in foam_case.camera_slices:
11             t = cls.map_slice_dto(sl)
12             slices_.append(t)
13
14         return FoamCase(foam_case._id,
15                         foam_case.name,
16                         cls.map_foam_cases_path_dto(foam_case.cases_dir),
17                         cam_props,
18                         slices_)
19
20     @classmethod
21     def map_foam_case_to_dto(cls, foam_case: FoamCase) -> FoamCaseDTO:
22         cam_prop_dtos = [cls.map_camera_props_to_dto(cam_prop)
23                          for cam_prop in foam_case.cam_prop_list]
24         slice_dtos = []
25         for sl in foam_case.slice_list:
26             t = cls.map_slice_to_dto(sl)
27             slice_dtos.append(t)
```

```
28          cases_path = cls.map_foam_cases_path_to_dto(foam_case.cases_dir)
29
30          return FoamCaseDTO(cases_path,
31                             cam_prop_dtos,
32                             slice_dtos,
33                             foam_case.name,
34                             _id=foam_case.idn)
35
36      @classmethod
37      def map_foam_cases_path_dto(cls, foam_cases_path_dto: CasesDirDTO) -> CasesDir:
38          return CasesDir(foam_cases_path_dto._id, foam_cases_path_dto.cases_path)
39
40      @classmethod
41      def map_foam_cases_path_to_dto(cls, foam_cases_path: CasesDir) -> CasesDirDTO:
42          return CasesDirDTO(str(foam_cases_path.path), _id=foam_cases_path.idn)
43
44      @classmethod
45      def map_point_dto(cls, p_dto) -> Point:
46          if isinstance(p_dto, PointDTO):
47              return Point(p_dto.x, p_dto.y, p_dto.z)
48          else:
49              return Point(p_dto["x"], p_dto["y"], p_dto["z"])
50
51      @classmethod
52      def map_point_to_dto(cls, p: Point) -> PointDTO:
53          return PointDTO(p.x, p.y, p.z)
54
55      @classmethod
56      def map_slice_dto(cls, s_dto: SliceDTO) -> CameraSlice:
57          sl_x = None if s_dto.sl_x is None else cls.map_point_dto(s_dto.sl_x)
58          sl_y = None if s_dto.sl_y is None else cls.map_point_dto(s_dto.sl_y)
59          sl_z = None if s_dto.sl_z is None else cls.map_point_dto(s_dto.sl_z)
60          return CameraSlice(s_dto._id, s_dto.name, sl_x, sl_y, sl_z)
61
62      @classmethod
63      def map_slice_to_dto(cls, s: CameraSlice) -> SliceDTO:
64          sl_x = None if s.sl_x is None else cls.map_point_to_dto(s.sl_x)
65          sl_y = None if s.sl_y is None else cls.map_point_to_dto(s.sl_y)
66          sl_z = None if s.sl_z is None else cls.map_point_to_dto(s.sl_z)
67          return SliceDTO(s.name, sl_x, sl_y, sl_z, _id=s.idn)
68
69      @classmethod
70      def map_camera_props_dto(cls, c_dto: CameraPropsDTO) -> CameraProps:
71          return CameraProps(c_dto._id,
72                             c_dto.name,
73                             cls.map_point_dto(c_dto.focal_point),
74                             cls.map_point_dto(c_dto.cam_position),
75                             c_dto.viewangle,
76                             cls.map_point_dto(c_dto.viewup))
77
78      @classmethod
79      def map_camera_props_to_dto(cls, c: CameraProps) -> CameraPropsDTO:
80          focal_point_dto = cls.map_point_to_dto(c.focal_point)
81          cam_position_dto = cls.map_point_to_dto(c.cam_position)
82          viewup = cls.map_point_to_dto(c.viewup)
83          return CameraPropsDTO(focal_point_dto, cam_position_dto,
84                                c.viewangle, viewup, c.name, _id=c.idn)
```

## A.9 foampostproc.config.py

Модуль, содержащий код для работы с файлом конфигурации программы.

```
1  import configparser
2  from pathlib import Path
3  from typing import List
4
5  from foampostproc.utils import PROJ_DIR
6
```

```
7  CONFIG_PATH = PROJ_DIR / Path("config/app.ini")
8
9
10 class Config:
11     __config: configparser.RawConfigParser = configparser.ConfigParser()
12     __is_read = False
13
14     @classmethod
15     def get_section(cls, section: str) -> 'ConfigSectionProxy':
16         if not cls.__is_read:
17             cls._read_config()
18         return ConfigSectionProxy(cls.__config[section])
19
20     @classmethod
21     def _read_config(cls):
22         cls.__config.read(CONFIG_PATH)
23         cls.__is_read = True
24
25
26 class ConfigSectionProxy:
27     COMMON_SECTION: str = "Common"
28     USE_PROJ_PREFIX_FOR_PATHS = "use_proj_prefix_for_paths"
29
30     __raw_config: configparser.RawConfigParser = configparser.ConfigParser()
31     __raw_config.read(CONFIG_PATH)
32     _common_section_config = __raw_config[COMMON_SECTION]
33
34     def __init__(self, section_proxy: configparser.SectionProxy):
35         self.__section_proxy = section_proxy
36
37     def get_int(self, option: str) -> int:
38         return self.__section_proxy.getint(option)
39
40     def get_float(self, option: str) -> float:
41         return self.__section_proxy.getfloat(option)
42
43     def get_boolean(self, option: str) -> bool:
44         return self.__section_proxy.getboolean(option)
45
46     def get(self, option: str) -> str:
47         return self.__section_proxy.get(option)
48
49     def get_path(self, option: str) -> Path:
50         option_path = Path(self.__section_proxy.get(option))
51         use_prefix = self._common_section_config.getboolean(
52             self.USE_PROJ_PREFIX_FOR_PATHS)
53         return Path(PROJ_DIR) / option_path if use_prefix else option_path
54
55     def get_list(self, option: str) -> List[str]:
56         return list(map(lambda s: s.strip(), self.get(option).split(',')))
```

## A.10 foampostproc.main.py

Модуль, содержащий точку входа в программу и производящий начальные настройки приложения.

```
1  from paraview.simple import *
2  from PySide6 import QtWidgets as QtW
3
4  import foampostproc.dto.dto as dto
5  from foampostproc.config import Config
6  from foampostproc.core.controller import handlers
7  from foampostproc.core.screenshot.taker import Screenshot
8  from foampostproc.core.view import MainWidget
9  from foampostproc.dao.dao import MongoFoamCaseDAO
10 from foampostproc.dao.daofactory import MongoDaoFactory
11 from foampostproc.utils import FileHandling, SharedState
```

```
12 from foampostproc.dto.modelmapper import Mapper
13
14 WINDOW_X = Config.get_section("ViewProperties").get_int("window_x")
15 WINDOW_Y = Config.get_section("ViewProperties").get_int("window_y")
16 WINDOW_W = Config.get_section("ViewProperties").get_int("window_w")
17 WINDOW_H = Config.get_section("ViewProperties").get_int("window_h")
18 WINDOW_TITLE = Config.get_section("ViewProperties").get("window_title")
19 TEST_DATA_PATH = Config.get_section("Paths").get_path("test_data")
20
21
22 def __run0():
23     case_dto = FileHandling.read_json(TEST_DATA_PATH,
24                                       object_hook_=dto.parse_config_json_hook)
25
26
27 def __run1():
28     case_dtos = MongoDaoFactory().get_dao(MongoFoamCaseDAO).get_all()
29     cases = list(map(Mapper.map_foam_case_dto, case_dtos))
30     Screenshot.take_screenshots(cases, Config.get_section("Paths").get_path("output"))
31
32
33 def run():
34     case_dtos = MongoDaoFactory().get_dao(MongoFoamCaseDAO).get_all()
35     cases = list(map(Mapper.map_foam_case_dto, case_dtos))
36
37     app = QtW.QApplication(sys.argv)
38
39     window = QtW.QMainWindow()
40     m_widget = MainWidget([case.name for case in cases], **handlers)
41     SharedState.m_widget = m_widget
42     SharedState.case_list = cases
43
44     layout = QtW.QGridLayout()
45     central_widget = QtW.QWidget()
46     central_widget.setLayout(layout)
47     layout.addWidget(m_widget)
48     window.setCentralWidget(central_widget)
49
50     window.setWindowTitle(WINDOW_TITLE)
51     window.setGeometry(WINDOW_X, WINDOW_Y, WINDOW_W, WINDOW_H)
52     window.show()
53     sys.exit(app.exec_())
54
55
56 if __name__ == "__main__":
57     run()
```

## A.11 foampostproc.utils.py

Модуль, содержащий разные полезные функции например чтение JSON файлов или константы указывающие путь к корневой директории приложения. Также здесь находится общее состояние программы.

```
1  from os.path import dirname, abspath
2  from pathlib import Path
3  from typing import Any
4  from ntpath import split
5  from os import makedirs
6  from json import dumps, load
7  from paraview.simple import *
8
9  SRC_DIR = Path(dirname(abspath(__file__)))
10 PROJ_DIR = SRC_DIR.parent
11
12
13 class SharedState(object):
```

```
14      case_list = []
15      m_widget = None
16      cam_props_for_del = []
17      slices_for_del = []
18      cases_for_del = []
19
20
21  class FileHandling(object):
22      @classmethod
23      def write_json(cls, obj, path: str) -> None:
24          def get_object_dict(d):
25              return d.__dict__
26
27          filepath, _ = split(path)
28          if filepath:
29              makedirs(filepath, exist_ok=True)
30          with open(path, "w") as out:
31              json_string = dumps(obj, default=get_object_dict)
32              out.write(json_string)
33
34      @classmethod
35      def read_json(cls, inp: Path, object_hook_=None) -> Any:
36          with open(inp, "r") as fin:
37              data = load(fin, object_hook=object_hook_)
38
39          return data
40
41      @classmethod
42      def read_foamcase(cls, inp: Path):
43          foamcase_path = inp / "temp.foam"
44          cls.write_file(foamcase_path)
45          foamcase = OpenFOAMReader(FileName=str(foamcase_path))
46          Path.unlink(foamcase_path)
47          return foamcase
48
49      @classmethod
50      def write_file(cls, file_path: Path, text: str = "", mode: str = 'w'):
51          dir_path = file_path.parent
52          dir_path.mkdir(parents=True, exist_ok=True)
53          with open(file_path, mode) as out:
54              out.write(text)
55
56
57  if __name__ == "__main__":
58      print(PROJ_DIR)
```