МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

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

Кафедра Математического и компьютерного моделирования

Анализ экспериментальных данных с помощью NoSQL

МАГИСТЕРСКАЯ РАБОТА

студента 2 курса 247 группы

направления 09.03.04 — Прикладная информатика

механико-математического факультета

Иванова Дмитрия Алексеевича

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

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

Саратов 2021

СОДЕРЖАНИЕ

Стр.

ВВЕДЕНИЕ **4**

1 Краткая информация об OpenFOAM и ParaView **5**
- 1.1 OpenFOAM 5
- 1.2 ParaView 7

2 Обзор существующих решений **9**
- 2.1 HELYX OS 9
- 2.2 ANSA 10
- 2.3 CastNet 12
- 2.4 Итог 13

3 Проектирование информационной системы **14**
- 3.1 Постановка задачи 14
- 3.2 Диаграмма прецедентов 14
- 3.3 Диаграмма классов 15
  - 3.3.1 Диаграмма модели данных 15
  - 3.3.2 Диаграмма, представляющая способ организации экспериментальных данных 16
  - 3.3.3 Диаграмма для работы с базой данных. Слой DAO 17
  - 3.3.4 Диаграмма классов для представлений 20
- 3.4 Диаграмма последовательностей 21

4 Выбор средств разработки **24**
- 4.1 NoSQL 24
- 4.2 MongoDB 26
- 4.3 Python 27
- 4.4 Обзор средств разработки графического интерфейса пользователя 30
  - 4.4.1 Kivy 30
  - 4.4.2 TKinter 31
  - 4.4.3 wxPython 32
  - 4.4.4 PyQt/PySide 32

5 Разработка **35**

6 Пример помощи в анализе экспериментальных данных 38

ЗАКЛЮЧЕНИЕ 43

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 44

ПРИЛОЖЕНИЕ А Исходный код программы 48

A.1 foampostproc.core.screenshot.taker.py 48
A.2 foampostproc.core.controller.py 48
A.3 foampostproc.core.model.py 52
A.4 foampostproc.core.view.py 53
A.5 foampostproc.dao.dao.py 58
A.6 foampostproc.dao.daofactory.py 62
A.7 foampostproc.dto.dto.py 62
A.8 foampostproc.dto.modelmapper.py 64
A.9 foampostproc.config.py 65
A.10 foampostproc.main.py 66
A.11 foampostproc.utils.py 67

ВВЕДЕНИЕ

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

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

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

- Кратко рассмотреть CAD/CAE систему – OpenFOAM и 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 Краткая информация 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.

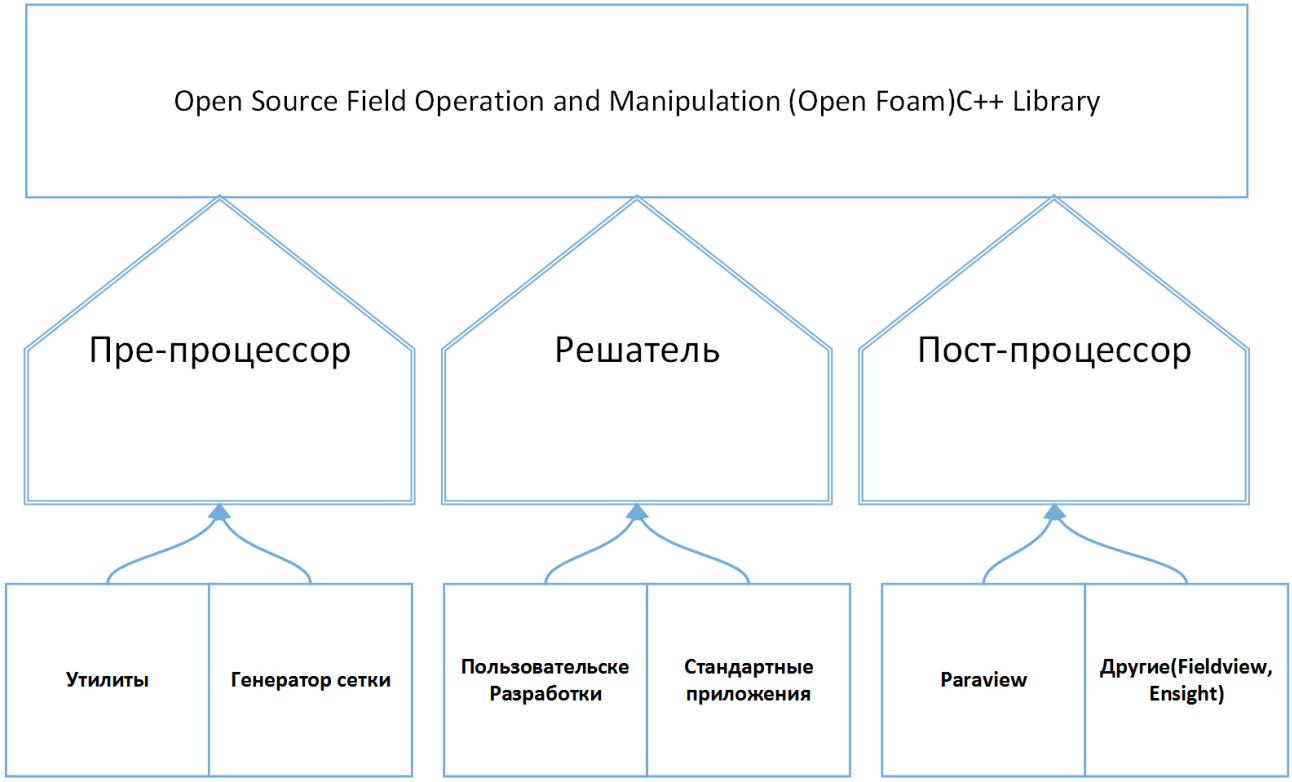

Рисунок 1.1 — Утилиты и программы входящие в пакет OpenFOAM, сгруппированные по этапам работы с расчетом

Работа с программой делится на три этапа:

1. Пре-процессинг;
2. Решение;
3. Пост-процессинг.

На этапе пре-процессинга в специальных файлах задаются входные данные для рассчета примера, такие как: начальное время, конечное время, шаг и так далее. Также параметры для хранения решения: время, формат, тип сжатия. Также в препроцессинг включены настройки выбора различных схем рассчета, котоыре 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 пост-процессинга 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

Рассматриваемая в данной курсовой работе информационная система должна выполнять пост-обработку данных, полученных в результате численно эксперимента в пакете OpenFOAM, делая упор на автоматизацию функций для работы с серией данных. Рассмотрим доступные приложение 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.

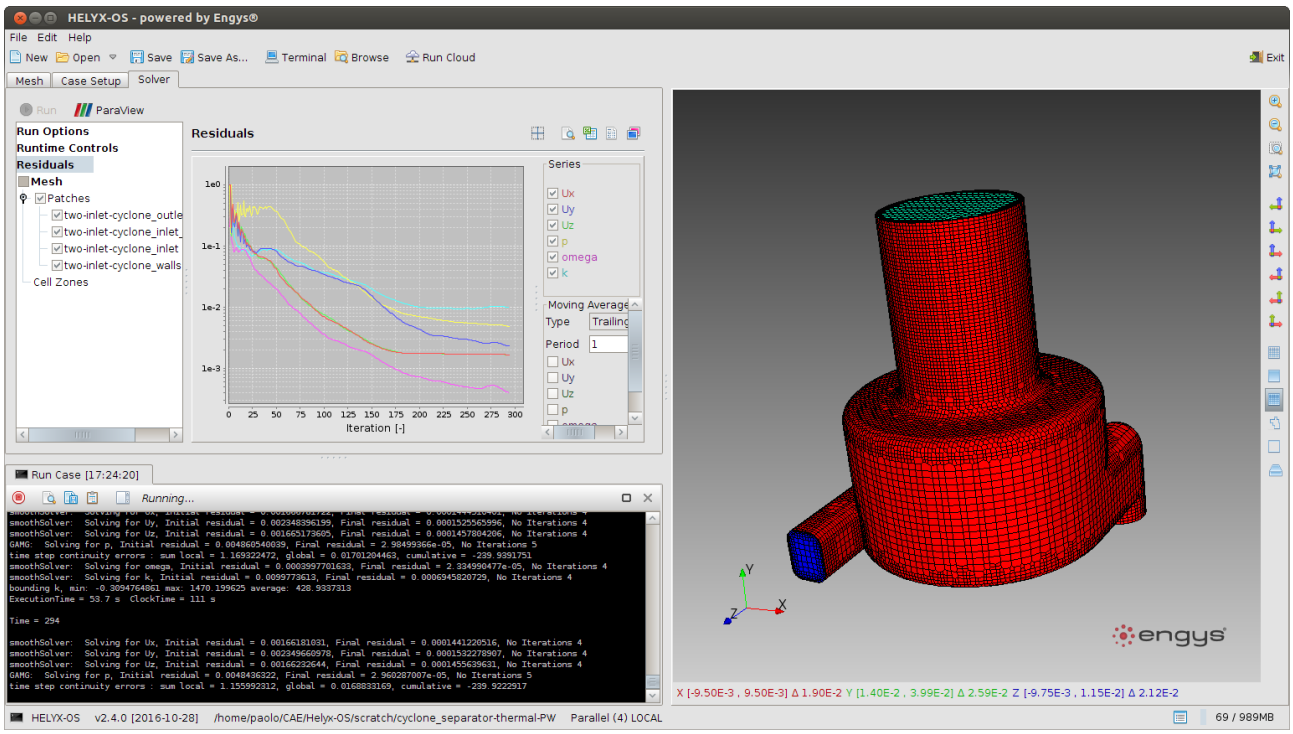

Рисунок 2.1 — Рабочий экран приложения Helyx OS

2.2 ANSA

ANSA - это инструмент пре-процессинга 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.

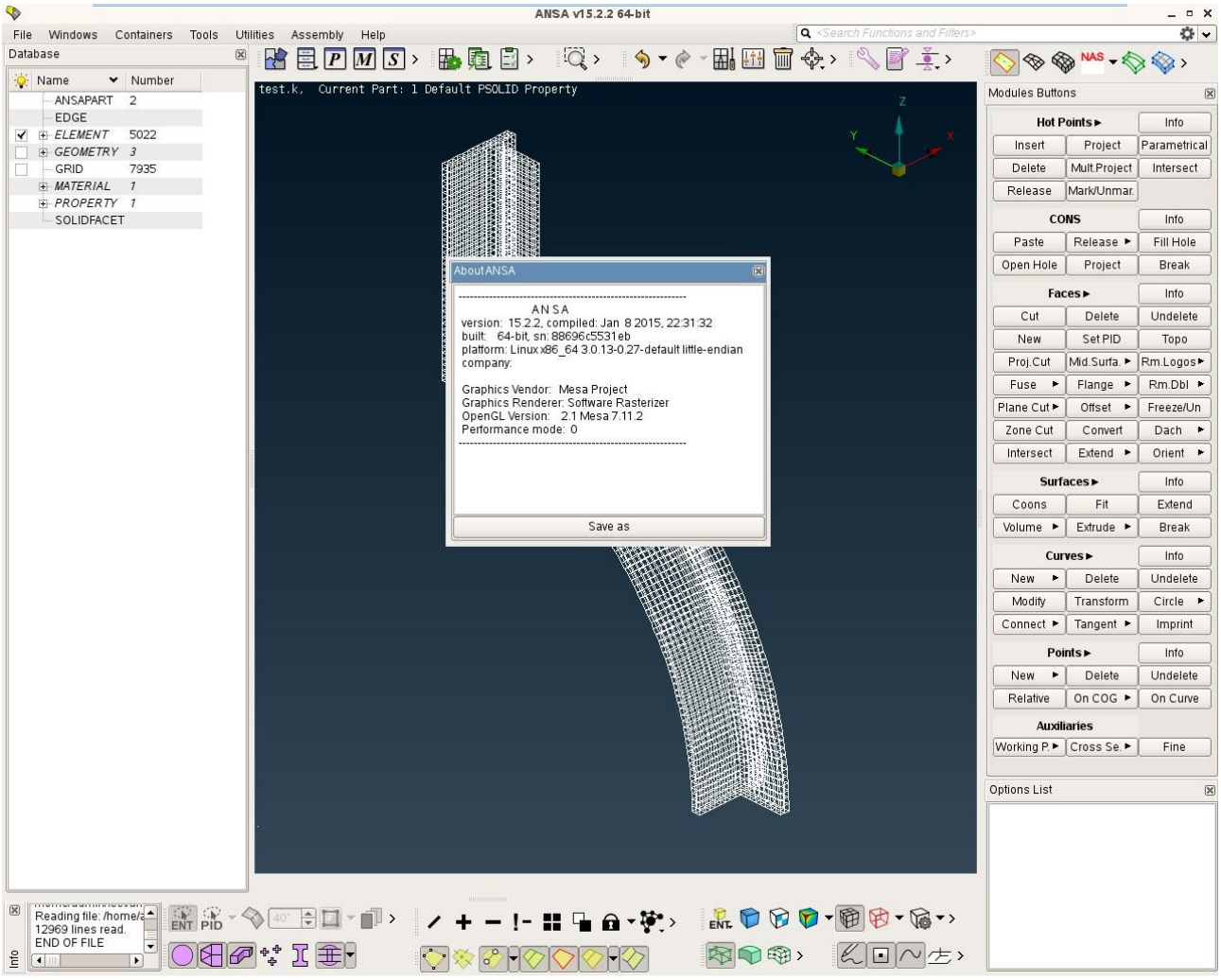

Рисунок 2.2 — Рабочий экран приложения ANSA

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

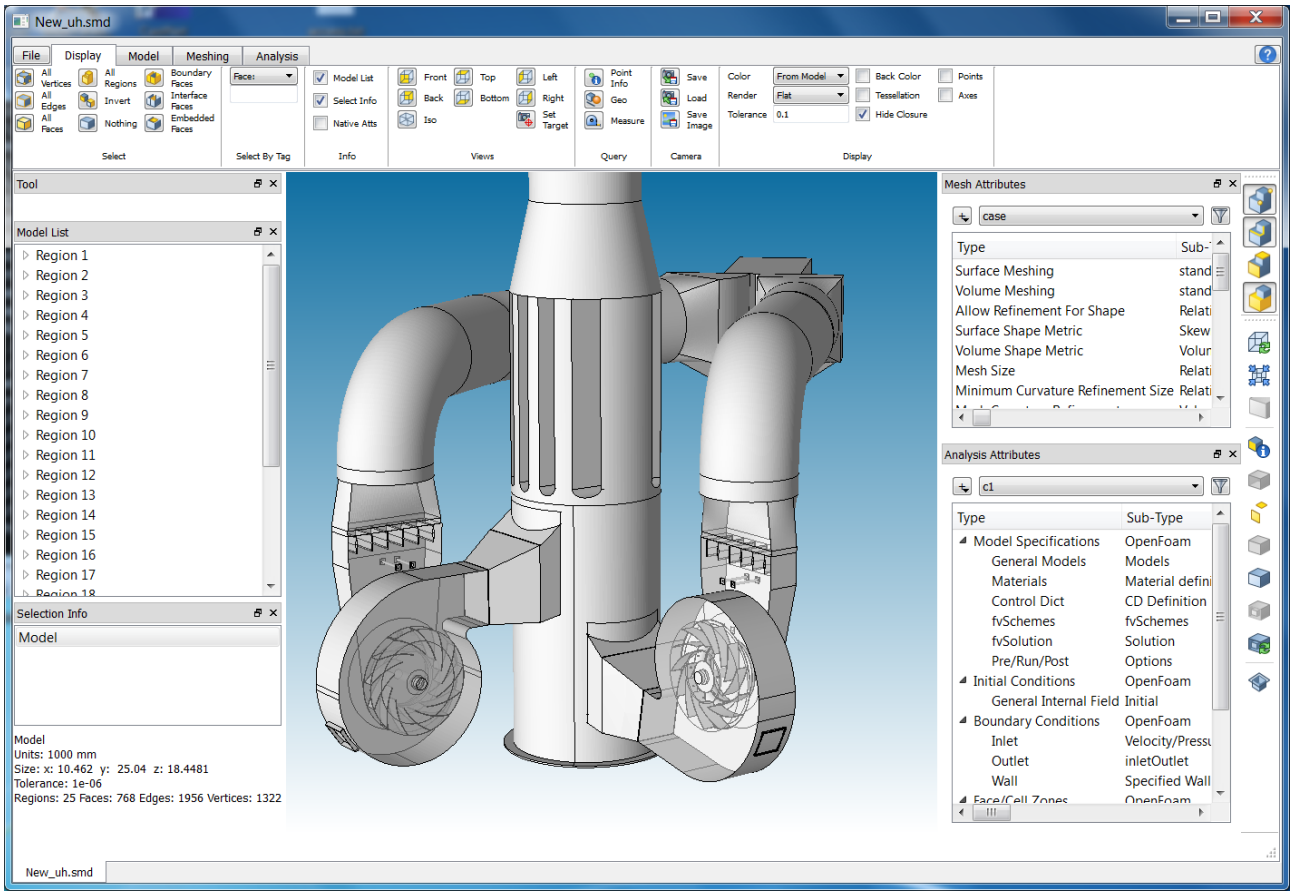

Рисунок 2.3 — Рабочий экран приложения CastNet

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 Итог

Таким образом, рассмотренные существующие решения предлагают разнообразные и гибкие возможности по работе с примерами, 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 пост-обработки, 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

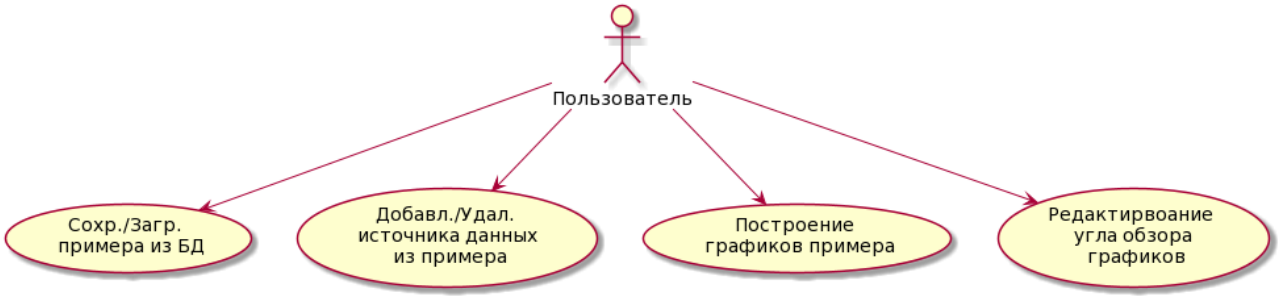

Рисунок 3.1 — Диаграмма прецедентов

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 базе данных 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 примера. Пример это директория содержащая один или несколько кейсов OpenFOAM.
- CameraProps – класс, который содержит настройки камеры для построения графика
- Point – утилитарный класс, необходимый для представления данных с точки зрения позиции камеры для обзора.
- SharedState – Класс хранящий общую информацию для контролера и представления. Необходим пересылки общей универсальной информации высокого уровня между виджетами разных уровней отображения. Можно мыслить как некий «шорт-кат» или кроткий путь в смысле термина подъема состояния.

  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.

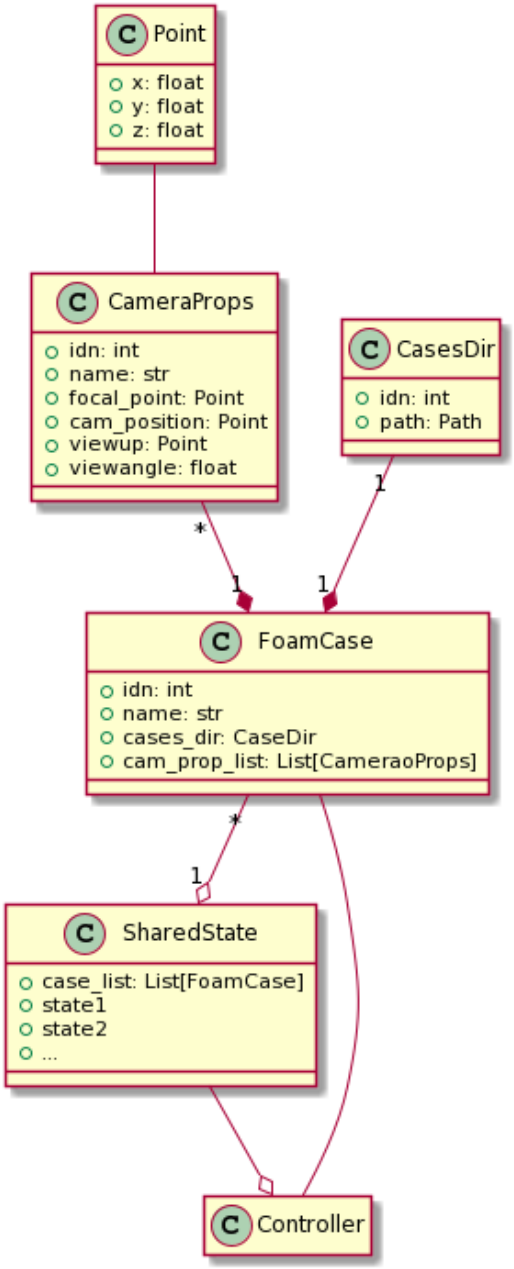

Рисунок 3.2 — Диаграмма классов, представляющая способ организации данных в приложении

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.

Статический класс ApplicationUtils реализует чтение экспериментальных данных и настроек приложения из файла. Класс DTO (Data Transfer Object) используется в качестве промежуточной сущности для валидации данных.

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 паттерна 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 в общем говоря, 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].

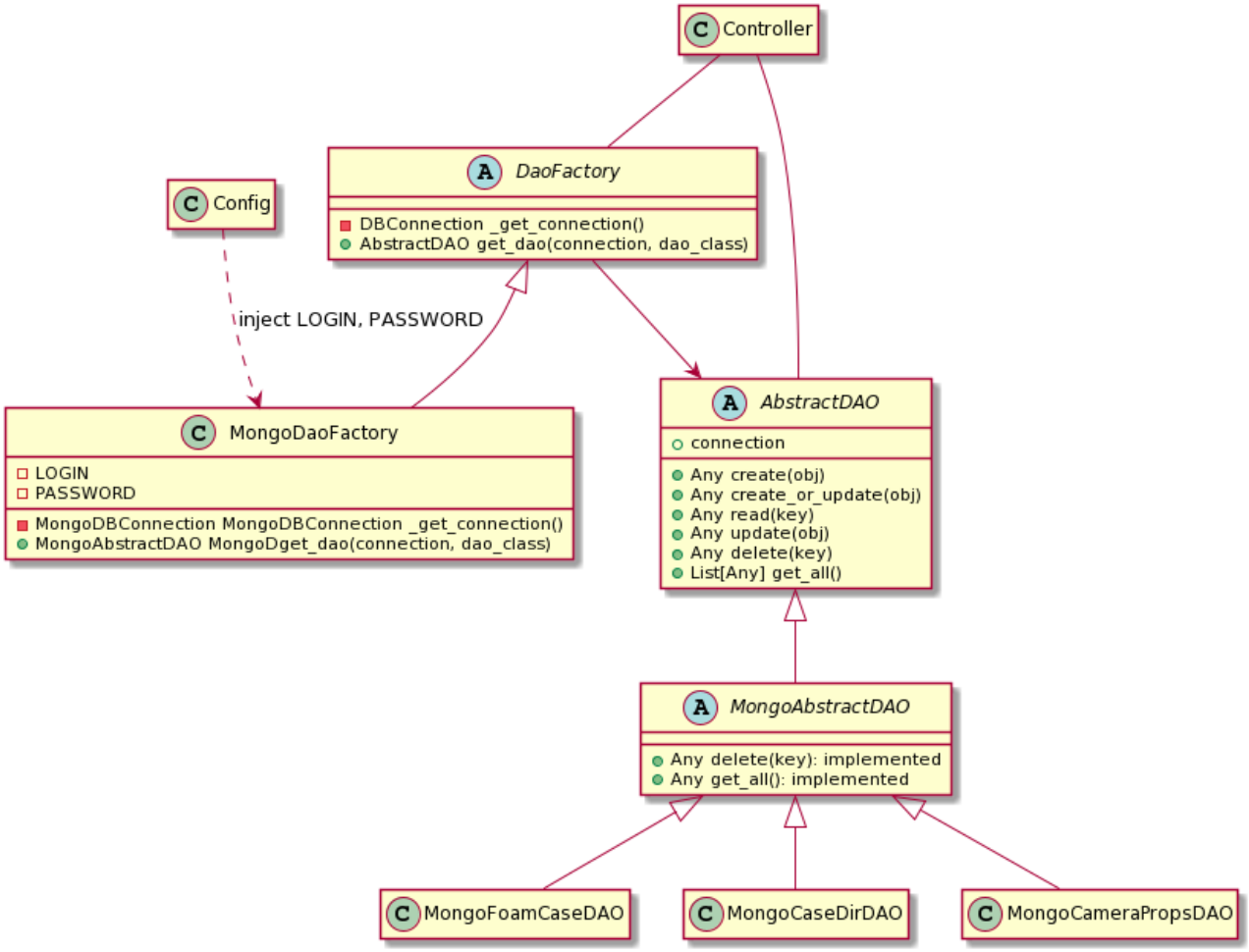

Рисунок 3.3 — Диаграмма классов. Слой DAO и его фабрика

Использование этого шаблона дает следующий 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 поговорим чуть подробнее про 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 перехватыва-

ют 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

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.

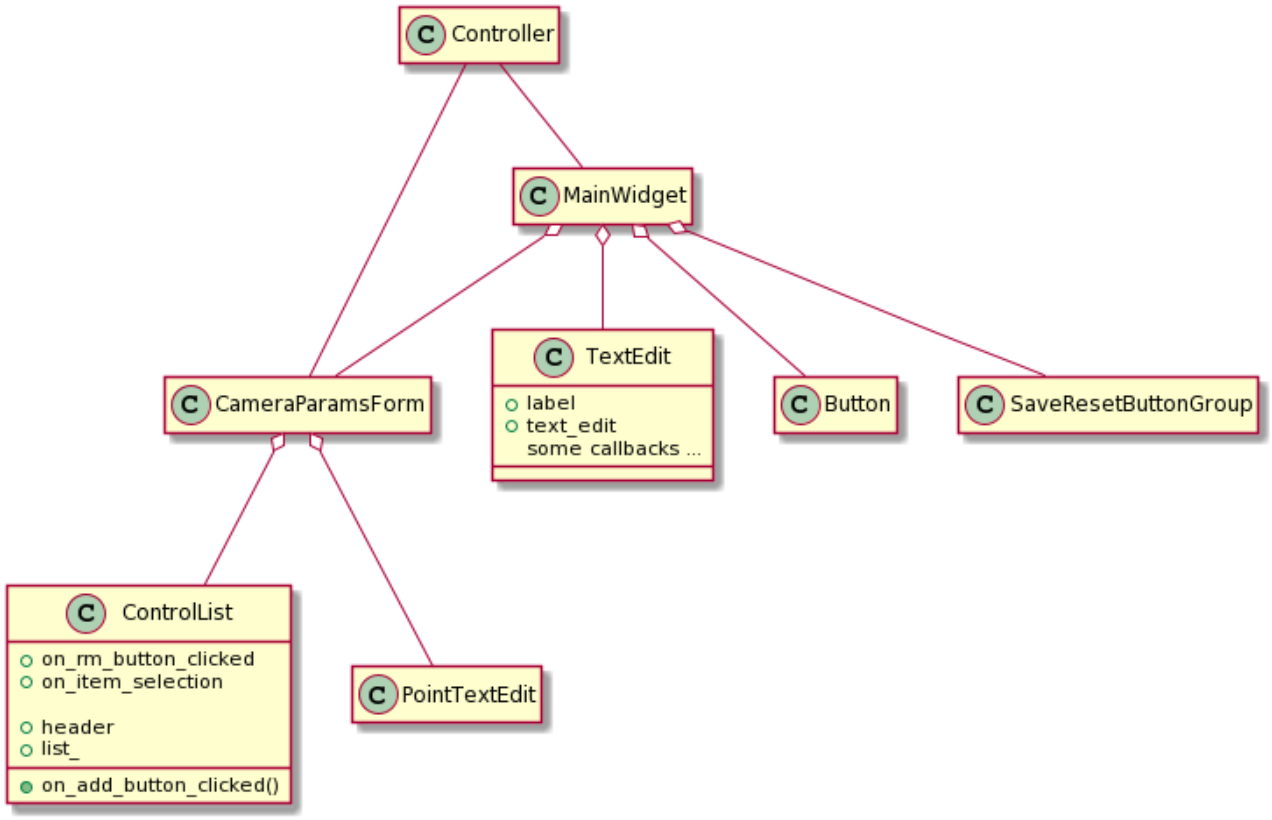

Рисунок 3.4 — Диаграмма классов. Отображение (View)

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.

3.4 Диаграмма последовательностей

Диаграммы 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».

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

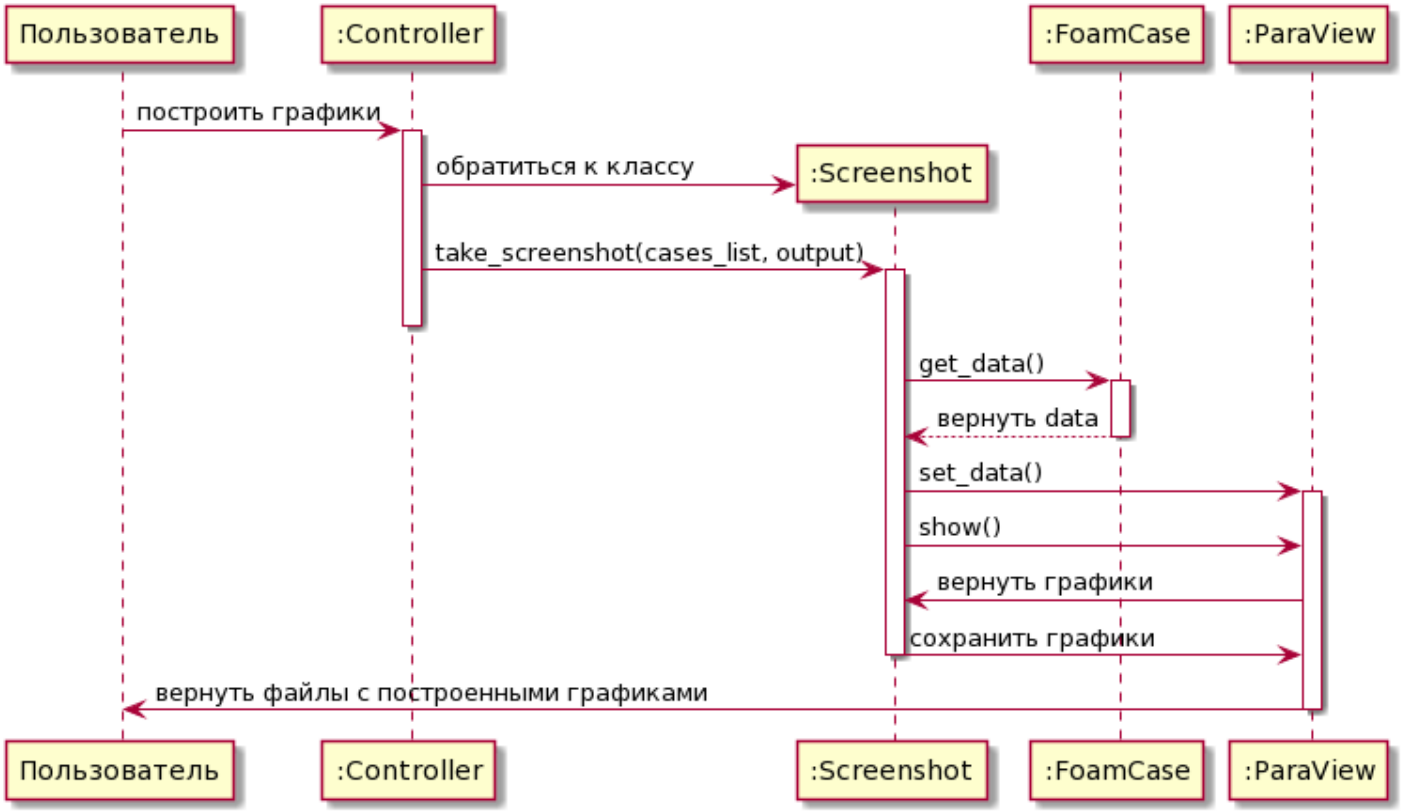

Рисунок 3.5 — Диаграмма последовательностей для прецедента «Построение графиков примера»

В соответствии с рисунком 3.6 представлена диаграмма последовательностей для прецедента «Сохранение примера в базу данных».

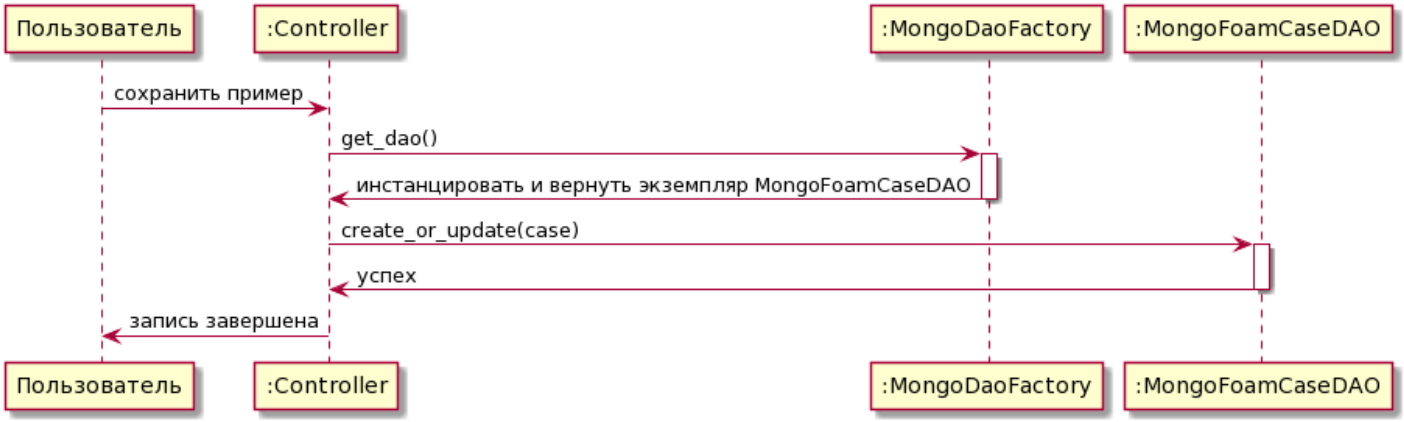

Рисунок 3.6 — Диаграмма последовательностей для прецедента «Сохранение примера в базу данных»

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

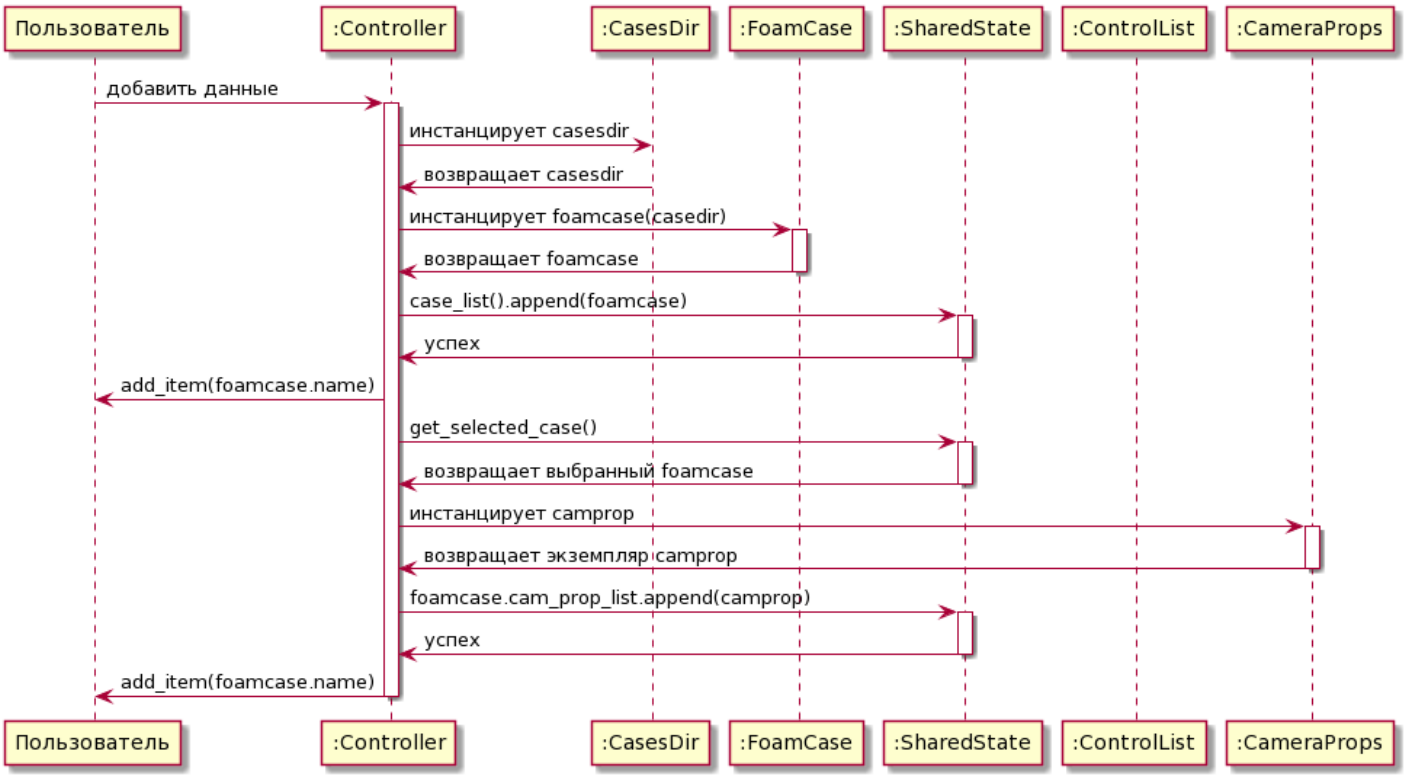

Рисунок 3.7 — Диаграмма последовательностей для прецедента «Добавление источника данных»

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 [16].
- 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 [16]. 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 [17].

- 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 [16]. 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 документоориентированный СУБД будет использована MongoDB. Ввиду популярности и простоте освоения.

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 [18].

```
1 {
2         "name" :  "Carlos␣Smith",
3         "title" : "Product␣Manager",
4         "location" : "New␣York,␣NY",
5         "twitter" : "@MongoDB",
6         "facebook" : "@MongoDB"
7 }
```

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 отличным 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 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 [19]. Характерное отличие языка – выделение блоков кода пробельными символами или табуляцией. Синтаксис ядра языка минималистичен, за счёт чего на практике редко возникает необходимость обращаться к документации [20]. Python – интерпретируемый мультипарадигмальный язык программирования, который используется для различных задач [19]. 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++ [20].

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 этом следует также подходить внимательно, так как верно выбранная среда сильно сокращает время написания, отладки и тестирования программы. Остановимся PyCharm Community Edition от компании JetBrains. Скриншот окна программы представлен в соответствии с рисунком 4.1.

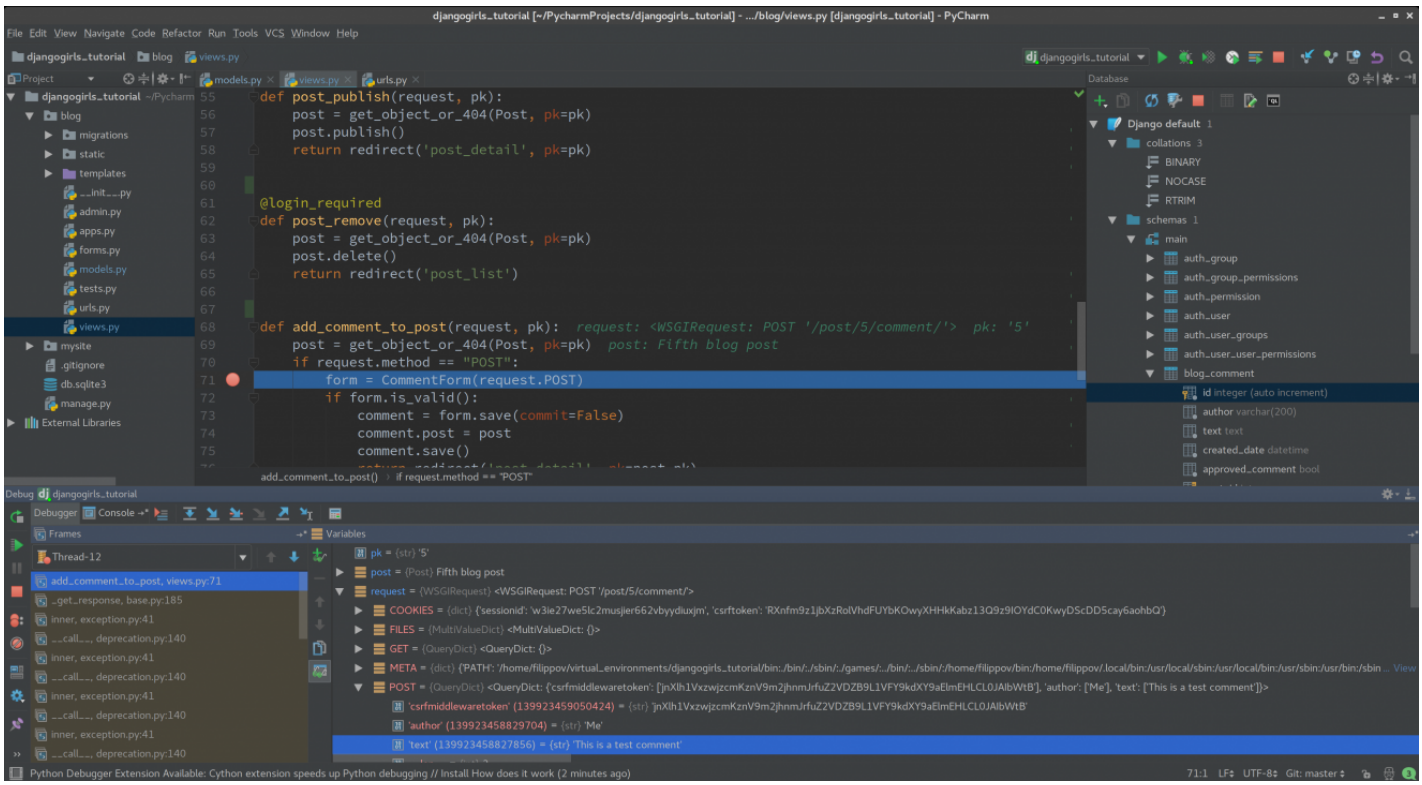

Рисунок 4.1 — Рабочее окно программы PyCharm

Выбор обусловлен тем, что во-первых, данное интегрированное средство разработки доступно под операционной системой Linux [21], что необходимо для тестирования так как средствами ParaView будут считываться данные расчетов полученные от пакета программ OpenFOAM, который в свою очередь работает только под операционной системой из семейства Linux. Во-вторых, средства отладки и автозаполнения кода предоставляются из коробки, что дает возможность не тратить время на установку и налаживание работы. В-третих, в PyCharm присутствует нативная интеграция виртуальным окружением Conda, которое сильно упрощает установки ParaView Python API.

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 бесплатного в личных и коммерческих целях. Kivy может работать на Android, iOS, Linux, macOS и Windows [22].

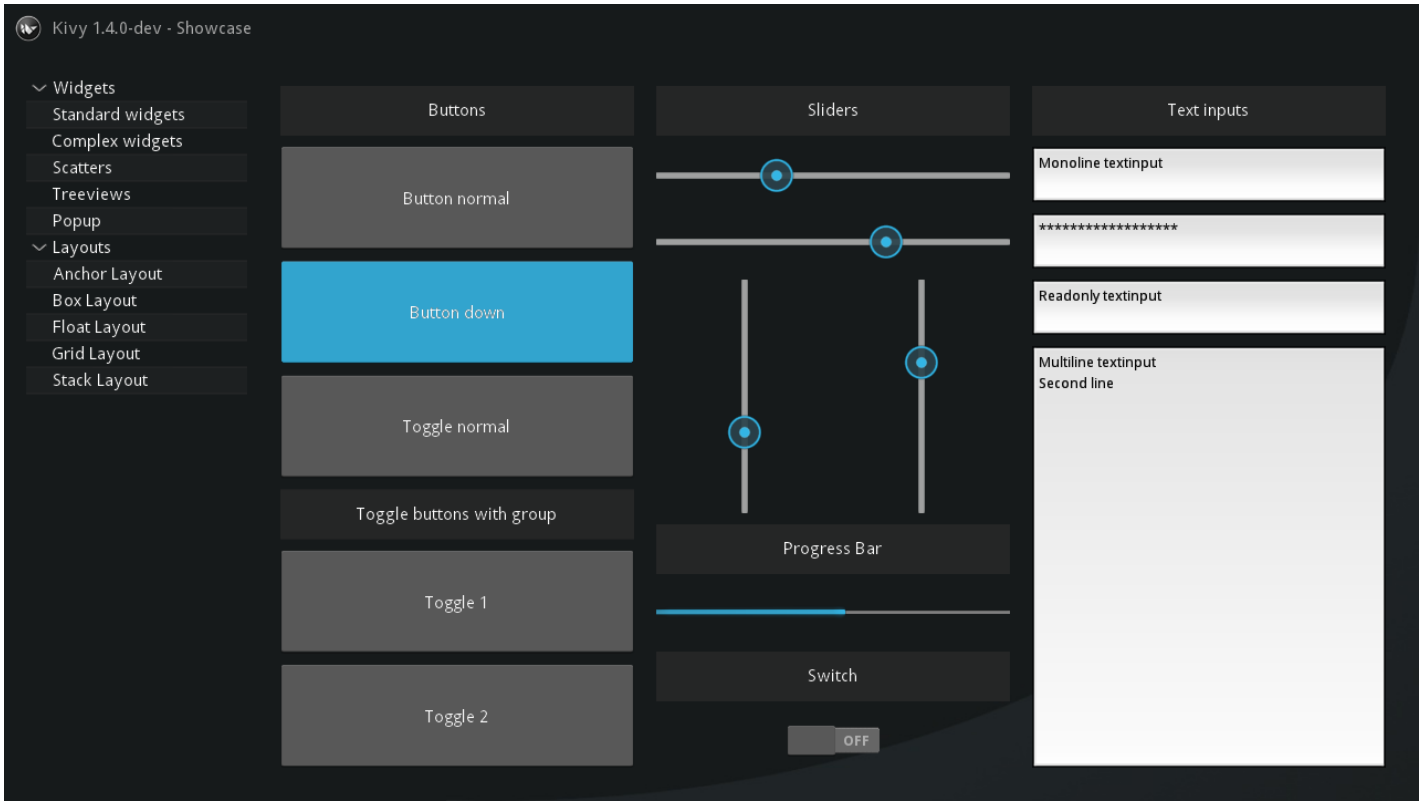

Рисунок 4.2 — Приложение Kivy

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 [25].

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++ [26].

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.

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 [23].

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 [24].

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.

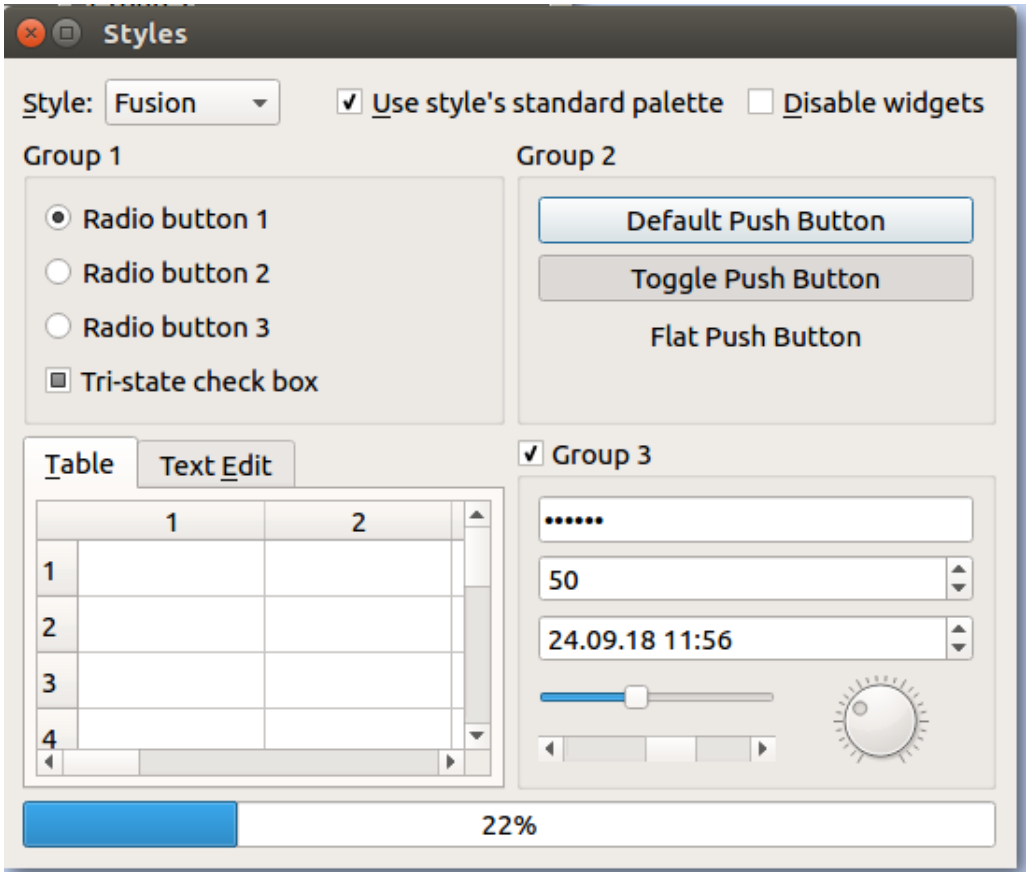

Рисунок 4.3 — Приложение PySide

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 PySide6.

5 Разработка

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 5.1 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.

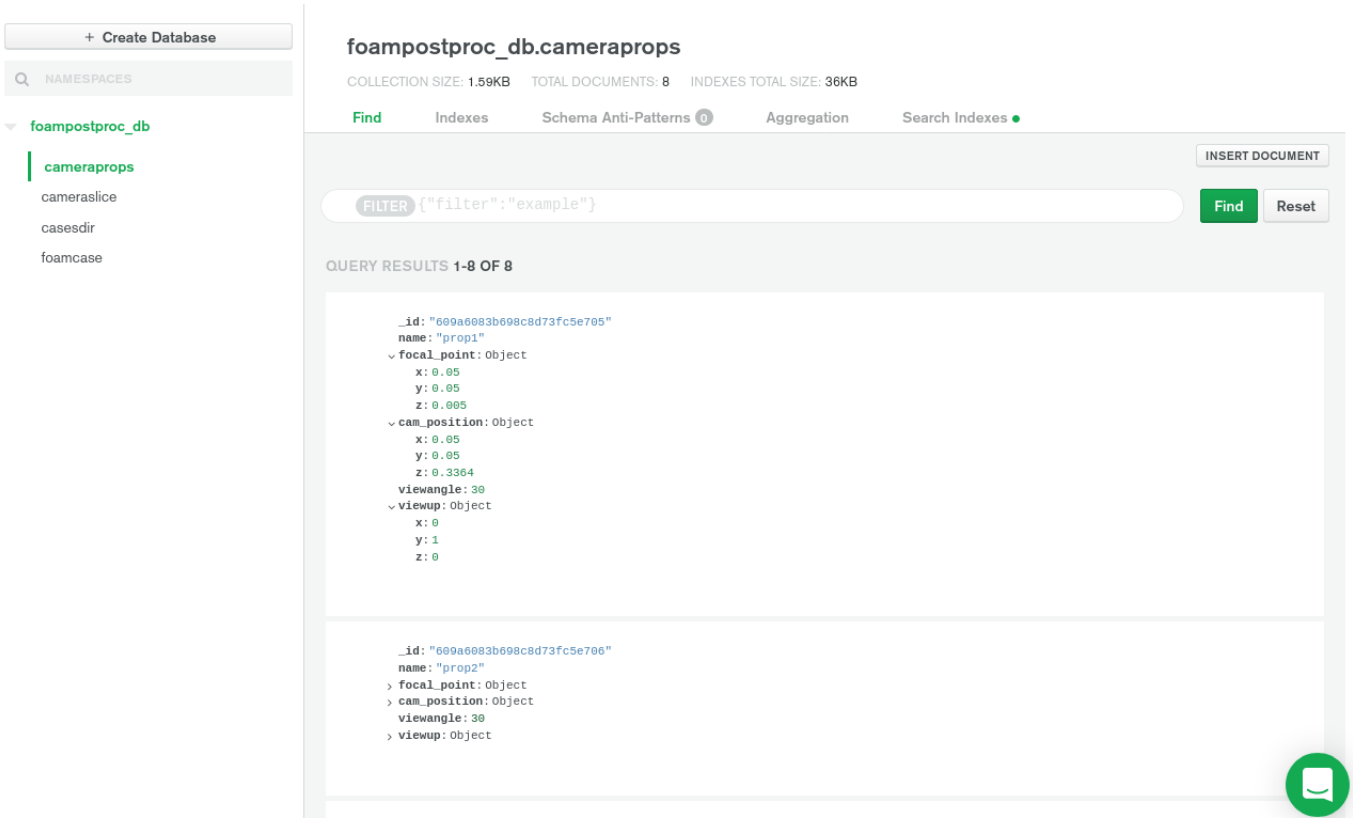

Рисунок 5.1 — Web-интерфейс экземпляра базы данных MongoDB

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

Большое время заняло подключение ParaView Python API к виртуальному окружению Python. Не вдаваясь в детали, 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, была 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.

В соответствии с рисунком 5.2 представлен скриншот программы.

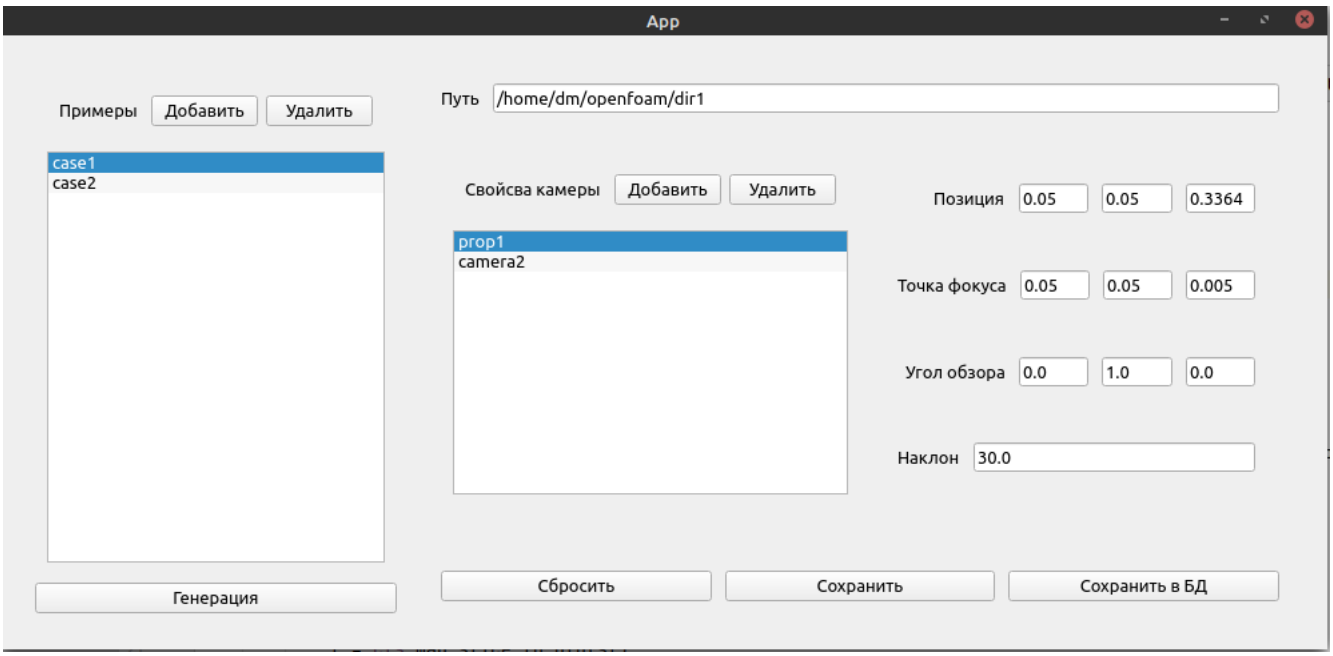

Рисунок 5.2 — Разработанное приложение

6 Пример помощи в анализе экспериментальных данных

Как было сказано ранее, при обработке экспериментальных данных когда проводиться серия экспериментов, в которых входные данные незначительно изменяются часто порождается большой объем результатов, надлежащих анализу или иными словами – пост-обработке. Для анализа данных необходимо провести однотипные манипуляции.

Например, для уравнениях Навье-Стокса в случае несжимаемых жидкостей для чего-то 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, из середины и последний.

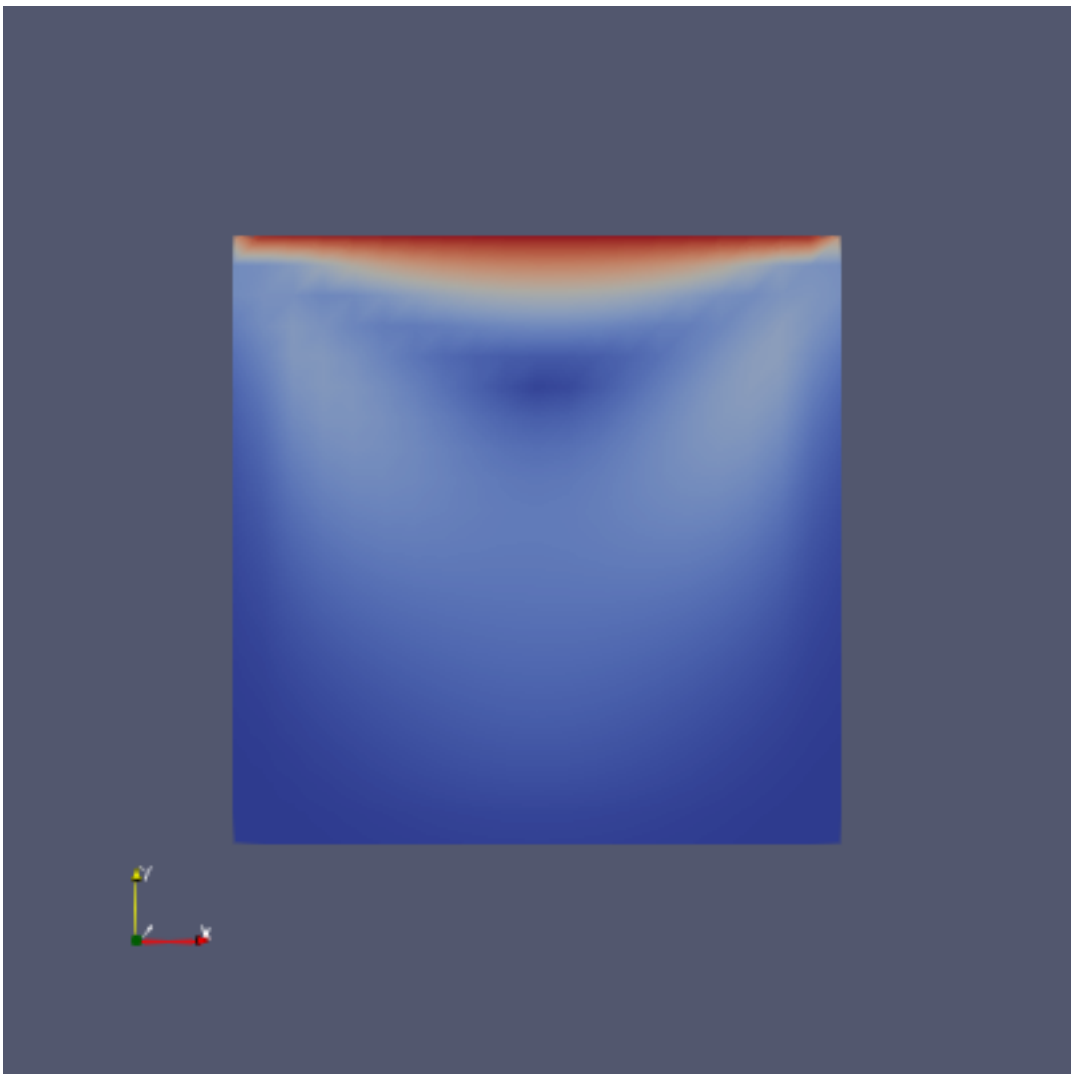

Рисунок 6.1 — График с коэффициентов вязкости 0.01

В соответствии с рисунком 6.1, в соответствии с рисунком 6.2 и в соответствии с рисунком 6.3 видно что пока рос коэффициент вязкости графики практически не менялись, но ближе к концу заметен резкий всплеск.

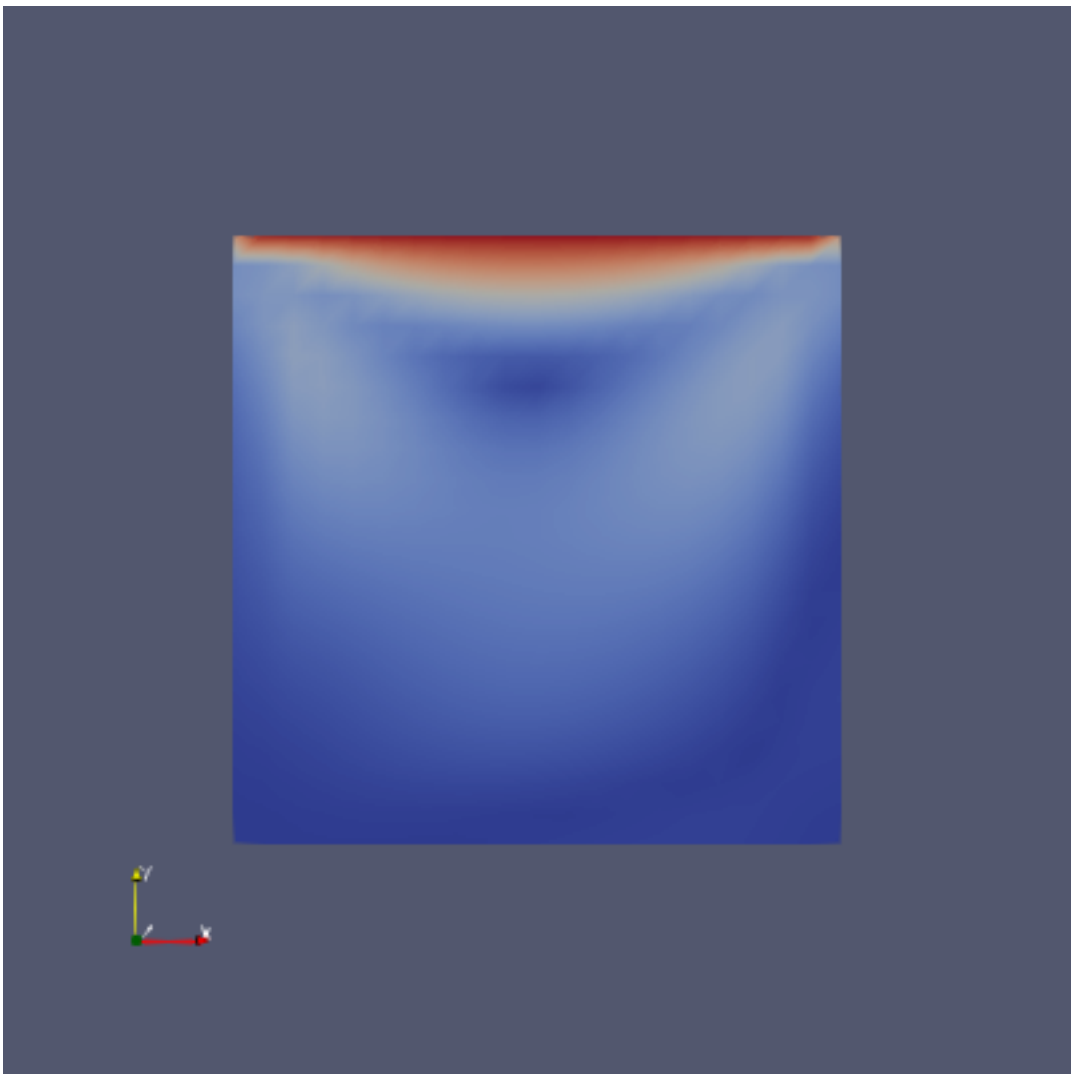

Рисунок 6.2 — График с коэффициентов вязкости 0.09

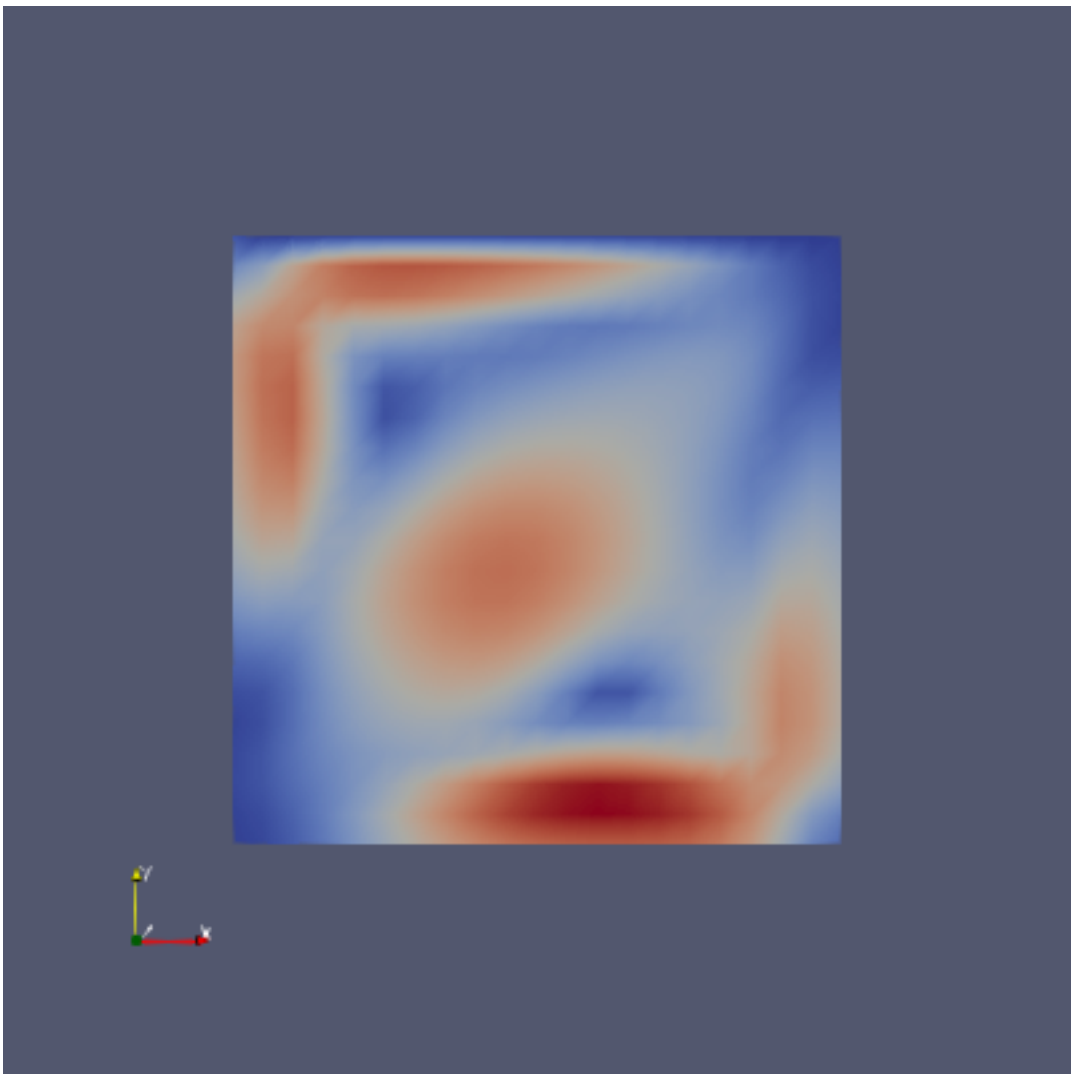

Рисунок 6.3 — График с коэффициентов вязкости 0.18

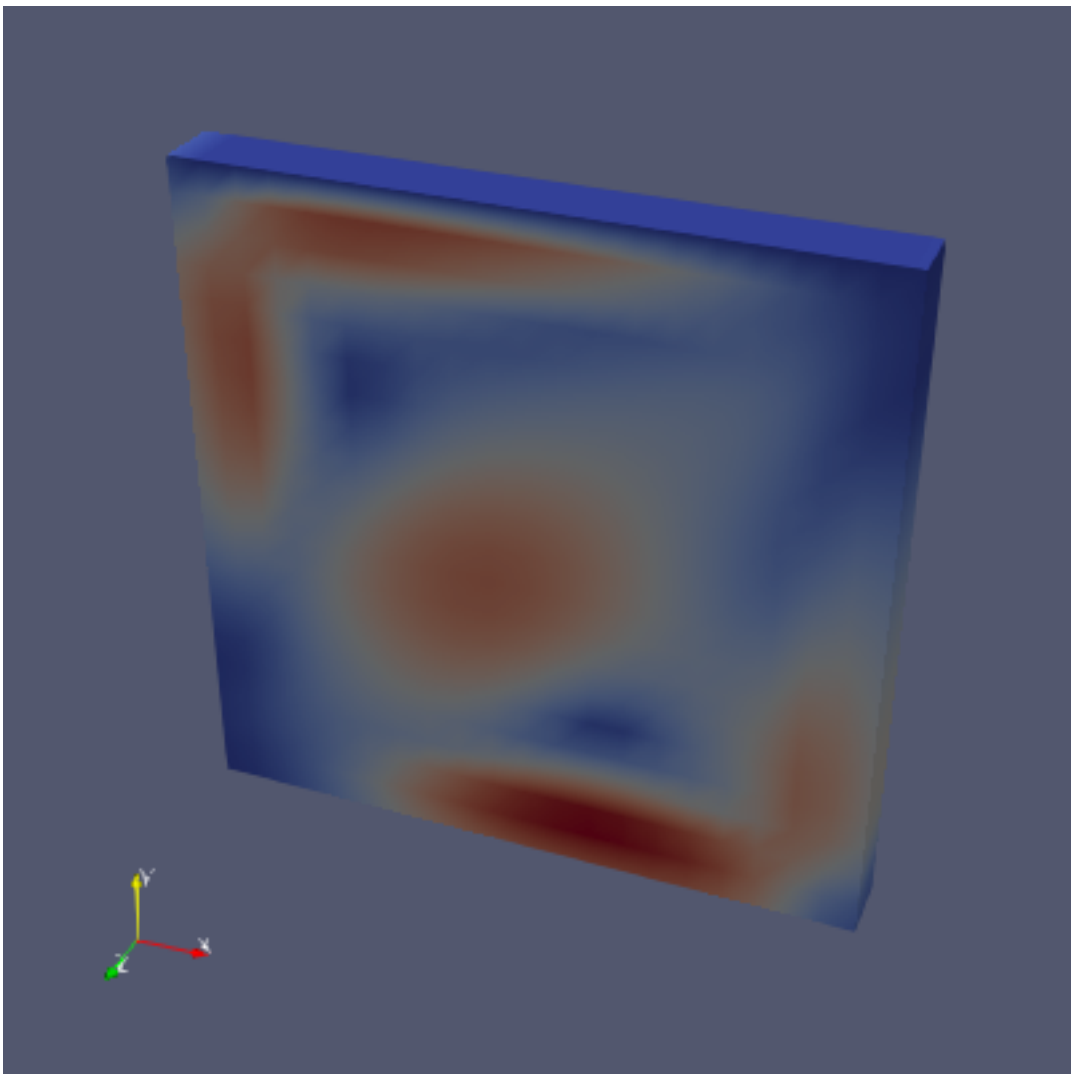

Рисунок 6.4 — График с коэффициентов вязкости 0.18 с другого ракурса

ЗАКЛЮЧЕНИЕ

В данной работе кратко рассмотрен пакет программ для численного моделирования OpenFOAM, приложения для пост-процессинга ParaView. Была поставлена задача проектирования и разработки информационной системы. Было выполнена проектирование, в частности построены UML-диаграммы прецедентов, классов, последовательностей. Также были рассмотрены существующие на данным момент решение по пост-обработки экспериментальных данных и выбраны средства разработки. 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 вещей 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 2020 года) // Официальный веб-сайт Wikipedia [Электронный ресурс] : [сайт]. - URL: https://ru.wikipedia.org/wiki/OpenFOAM (дата обращения 08.04.2020). - Загл с экрана - Яз. рус.

2. Страница User guide [Электронный ресурс] : (на 08 апреля 2020 года) // Официальный веб-сайт OpenFOAM [Электронный ресурс] : [сайт]. - URL: https://cfd.direct/openfoam/user-guide (дата обращения 08.04.2020). - Загл с экрана - Яз. англ.

3. Страница About [Электронный ресурс] : (на 08 апреля 2020 года) // Официальный веб-сайт ParaView [Электронный ресурс] : [сайт]. - URL: https://www.paraview.org/overview (дата обращения 08.04.2020). - Загл с экрана - Яз. англ.

4. Статья ParaView [Электронный ресурс] : (на 09 апреля 2020 года) // Официальный веб-сайт Wikipedia [Электронный ресурс] : [сайт]. - URL: https://ru.wikipedia.org/wiki/ParaView (дата обращения 09.04.2020). - Загл с экрана - Яз. рус.

5. Страница ParaView and Python [Электронный ресурс] : (на 10 апреля 2020 года) // Официальный веб-сайт ParaView [Электронный ресурс] : [сайт]. - URL: https://www.paraview.org/Wiki/ParaView_and_Python (дата обращения 10.04.2020). - Загл с экрана - Яз. англ.

6. Продукт Helyx OS [Электронный ресурс] : (на 29 марта 2020 года) // Официальный веб-сайт Helyx [Электронный ресурс] : [сайт]. - URL: https://engys.com/products/helyx-os (дата обращения 29.03.2020). - Загл с экрана - Яз. англ.

7. Продукт ANSA [Электронный ресурс] : (на 11 апреля 2020 года) // Официальный веб-сайт компании Beta [Электронный ресурс] : [сайт]. - URL: http://www.beta-cae.gr/ansa.html (дата обращения 11.04.2020). - Загл с экрана - Яз. англ.

8. Продукт CastNet [Электронный ресурс] : (на 11 апреля 2020 года) // Официальный веб-сайт компании DHCAE Tools [Электронный ресурс] : [сайт]. - URL: http://www.dhcae-tools.com/CastNet.html (дата обращения 11.04.2020). - Загл с экрана - Яз. англ.

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

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 2020 года) // Веб-сайт refactoring guru [Электронный ресурс] : [сайт]. - URL: https://refactoring.guru/ru/design-patterns/factory-method (дата обращения 19.04.2020). - Загл с экрана - Яз. рус.

12. Паттерны проектирования. Заместитель [Электронный ресурс] : (на 19 апреля 2020 года) // Веб-сайт refactoring guru [Электронный ресурс] : [сайт]. - URL: https://refactoring.guru/ru/design-patterns/proxy (дата обращения 19.04.2020). - 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., Садаладж П. NoSQL. Методология разработки нереляционных баз данных / М. Фаулер – Диалектика-Вильямс, 2020. - 192 с.

15. Обзор NoSQL [Электронный ресурс] : (на 19 апреля 2020 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. Разбираемся в типах NoSQL СУБД [Электронный ресурс] : (на 19 апреля 2020 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.

17. NoSQL СУБД [Электронный ресурс] : (на 20 апреля 2020 года) // Веб-сайт wikipedia [Электронный ресурс] : [сайт]. - https://ru.wikipedia.org/wiki/NoSQL (дата обращения 20.04.2021). - Загл с экрана - Яз. рус.

18. MongoDB [Электронный ресурс] : (на 23 апреля 2020 года) // Веб-сайт mongodb [Электронный ресурс] : [сайт]. - https://www.mongodb.com/why-use-mongodb (дата обращения 23.04.2021). - Загл с экрана - Яз. англ.

19. 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.

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

21. Сравнение интегрированных сред разработки [Электронный ресурс] : (на 24 апреля 2020 года) // Веб-сайт wikipedia [Электронный ресурс] : [сайт]. - https://en.wikipedia.org/wiki/Comparison_of_integrated_development_environmen (дата обращения 24.04.2021). - Загл с экрана - Яз. англ.

22. Страница о нас [Электронный ресурс] : (на 30 апреля 2020 года) // Веб-сайт фреймворка Kivy [Электронный ресурс] : [сайт]. - https://kivy.org/#aboutus (дата обращения 30.04.2021). - Загл с экрана - Яз. англ.

23. PyQt. Введение [Электронный ресурс] : (на 30 апреля 2020 года) // Веб-сайт Riverbank Computing [Электронный ресурс] : [сайт]. - https://riverbankcomputing.com/software/pyqt/intro (дата обращения 30.04.2021). - Загл с экрана - Яз. англ.

24. Qt for Python [Электронный ресурс] : (на 30 апреля 2020 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.

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

26. Overview of wxPython [Электронный ресурс] : (на 30 апреля 2020 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

```
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
                    camera.SetPosition(cam_prop.cam_position.x, cam_prop.cam_position.y, cam_prop.cam_p
                    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

```
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():
    case_dir = CasesDir(ObjectId(), "")
    foamcase = FoamCase(ObjectId(), f"case{len(SharedState.case_list) + 1}", case_dir, [], [])
    SharedState.case_list.append(foamcase)
    SharedState.m_widget.cases_control_list.list_.addItem(foamcase.name)
    if len(SharedState.case_list) - 1 > 0:
        SharedState.m_widget.cases_control_list.list_.setCurrentRow(len(SharedState.case_list) - 1)
    handle_camera_props_add_button_clicked()


def handle_cases_rm_button_clicked():
    list_ = SharedState.m_widget.cases_control_list.list_
    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").get_path("output"))
60
61
62  def handle_case_path_field_on_text_changed():
63      pass
64
65
66  def handle_camera_props_add_button_clicked():
67      case = _get_selected_case()
68      cam_prop = CameraProps(ObjectId(), f"camera{len(case.cam_prop_list) + 1}", Point(0, 0, 0),
69                             Point(0, 0, 0), 0, Point(0, 0, 0))
70      case.cam_prop_list.append(cam_prop)
71      SharedState.m_widget.camera_props_control_list.list_.addItem(cam_prop.name)
72      if len(case.cam_prop_list) - 1 > 0:
73          SharedState.m_widget.camera_props_control_list.list_.setCurrentRow(len(case.cam_prop_list) - 1)
74
75
76  def handle_camera_props_rm_button_clicked():
77      case = _get_selected_case()
78
79      list_ = SharedState.m_widget.camera_props_control_list.list_
80      for item in list_.selectedItems():
81          item_row = list_.row(item)
82          SharedState.cam_props_for_del.append(case.cam_prop_list[item_row])
83          case.cam_prop_list.pop(item_row)
84          list_.takeItem(item_row)
85      if len(case.cam_prop_list) - 1 > 0:
86          list_.setCurrentRow(len(case.cam_prop_list) - 1)
87
88
89  def handle_camera_props_item_selection():
90      # if len(_get_selected_case().slice_list) == 0 or len(_get_selected_case().cam_prop_list) == 0:
91      #     return
92
93      cam_prop = _get_selected_cam_prop()
94      if cam_prop is not None:
95          form = SharedState.m_widget.camera_props_params_form
96          form.position_field.text_edit1.setText(str(cam_prop.cam_position.x))
97          form.position_field.text_edit2.setText(str(cam_prop.cam_position.y))
98          form.position_field.text_edit3.setText(str(cam_prop.cam_position.z))
99
100         form.focal_point_field.text_edit1.setText(str(cam_prop.focal_point.x))
101         form.focal_point_field.text_edit2.setText(str(cam_prop.focal_point.y))
```

```
102         form.focal_point_field.text_edit3.setText(str(cam_prop.focal_point.z))
103
104         form.view_up_field.text_edit1.setText(str(cam_prop.viewup.x))
105         form.view_up_field.text_edit2.setText(str(cam_prop.viewup.y))
106         form.view_up_field.text_edit3.setText(str(cam_prop.viewup.z))
107
108         form.view_angle_field.text_edit.setText(str(cam_prop.viewangle))
109
110
111 def handle_slice_props_add_button_clicked():
112     case = _get_selected_case()
113     cam_slice = CameraSlice(ObjectId(), f"slice{len(case.slice_list) + 1}", None, None, None)
114     case.slice_list.append(cam_slice)
115     SharedState.m_widget.slice_props_control_list.list_.addItem(cam_slice.name)
116     if len(case.slice_list) - 1 > 0:
117         SharedState.m_widget.slice_props_control_list.list_.setCurrentRow(len(case.slice_list) - 1)
118
119
120 def handle_slice_props_rm_button_clicked():
121     case = _get_selected_case()
122
123     list_ = SharedState.m_widget.slice_props_control_list.list_
124     for item in list_.selectedItems():
125         item_row = list_.row(item)
126         SharedState.slices_for_del.append(case.slice_list[item_row])
127         case.slice_list.pop(item_row)
128         list_.takeItem(item_row)
129     if len(case.slice_list) - 1 > 0:
130         list_.setCurrentRow(len(case.slice_list) - 1)
131
132
133 def handle_slice_props_item_selection():
134     slice_ = _get_selected_slice_prop()
135
136     if slice_ is not None:
137         form = SharedState.m_widget.slice_params_form
138
139         x = "" if slice_.sl_x is None else str(slice_.sl_x.x)
140         y = "" if slice_.sl_x is None else str(slice_.sl_x.y)
141         z = "" if slice_.sl_x is None else str(slice_.sl_x.z)
142
143         form.x_slice_field.text_edit1.setText(x)
144         form.x_slice_field.text_edit2.setText(y)
145         form.x_slice_field.text_edit3.setText(z)
146
147         x = "" if slice_.sl_y is None else str(slice_.sl_y.x)
148         y = "" if slice_.sl_y is None else str(slice_.sl_y.y)
149         z = "" if slice_.sl_y is None else str(slice_.sl_y.z)
150
151         form.y_slice_field.text_edit1.setText(x)
152         form.y_slice_field.text_edit2.setText(y)
153         form.y_slice_field.text_edit3.setText(z)
154
155         x = "" if slice_.sl_z is None else str(slice_.sl_z.x)
156         y = "" if slice_.sl_z is None else str(slice_.sl_z.y)
157         z = "" if slice_.sl_z is None else str(slice_.sl_z.z)
158
159         form.z_slice_field.text_edit1.setText(x)
160         form.z_slice_field.text_edit2.setText(y)
161         form.z_slice_field.text_edit3.setText(z)
162
163
164 def handle_save_btn():
165     case = _get_selected_case()
166     case_path_field_text = SharedState.m_widget.case_path_field.text_edit.text()
167     case.cases_dir.path = Path(case_path_field_text)
168
169     cam_prop = _get_selected_cam_prop()
170     # slice_ = _get_selected_slice_prop()
171
172     cam_form = SharedState.m_widget.camera_props_params_form
173     cam_prop.cam_position = deepcopy(cam_form.position)
174     cam_prop.viewup = deepcopy(cam_form.view_up)
```

```
175     cam_prop.focal_point = deepcopy(cam_form.focal_point)
176     cam_prop.viewangle = float(cam_form.view_angle)
177 
178     # slice_form = SharedState.m_widget.slice_params_form
179     # if slice_form.x_slice_field.text_edit1.text() == "" \
180     #         or slice_form.x_slice_field.text_edit2.text() == "" \
181     #         or slice_form.x_slice_field.text_edit3.text() == "":
182     #     slice_.sl_x = None
183     # else:
184     #     slice_.sl_x = slice_form.x_slice
185     #
186     # if slice_form.y_slice_field.text_edit1.text() == "" \
187     #         or slice_form.y_slice_field.text_edit2.text() == "" \
188     #         or slice_form.y_slice_field.text_edit3.text() == "":
189     #     slice_.sl_y = None
190     # else:
191     #     slice_.sl_y = slice_form.y_slice
192     #
193     # if slice_form.z_slice_field.text_edit1.text() == "" \
194     #         or slice_form.z_slice_field.text_edit2.text() == "" \
195     #         or slice_form.z_slice_field.text_edit3.text() == "":
196     #     slice_.sl_z = None
197     # else:
198     #     slice_.sl_z = slice_form.z_slice
199 
200 
201 def handle_reset_btn():
202     case_dtos = MongoDaoFactory().get_dao(MongoFoamCaseDAO).get_all()
203     cases = list(map(Mapper.map_foam_case_dto, case_dtos))
204     SharedState.case_list = cases
205     list_ = SharedState.m_widget.cases_control_list.list_
206     list_.clear()
207     list_.addItems([case.name for case in cases])
208     if len(cases) > 0:
209         list_.setCurrentRow(0)
210 
211 
212 def handle_save_db_btn():
213     cases = SharedState.case_list
214     for case in cases:
215         handle_save_btn()
216         MongoDaoFactory().get_dao(MongoFoamCaseDAO).create_or_update(Mapper.map_foam_case_to_dto(case))
217 
218     for case in SharedState.cases_for_del:
219         MongoDaoFactory().get_dao(MongoFoamCaseDAO).delete(str(case.idn))
220 
221     for cam_prop in SharedState.cam_props_for_del:
222         MongoDaoFactory().get_dao(MongoCameraPropsDAO).delete(str(cam_prop.idn))
223 
224     for sl in SharedState.slices_for_del:
225         MongoDaoFactory().get_dao(MongoCameraSliceDAO).delete(str(sl.idn))
226 
227     SharedState.cam_props_for_del = []
228     SharedState.slices_for_del = []
229     SharedState.cases_for_del = []
230 
231 
232 def _get_selected_case_row():
233     selected_item = SharedState.m_widget.cases_control_list.list_.selectedItems()[0]
234     return SharedState.m_widget.cases_control_list.list_.row(selected_item)
235 
236 
237 def _get_selected_case():
238     row = _get_selected_case_row()
239     return SharedState.case_list[row]
240 
241 
242 def _get_selected_cam_prop_row():
243     selected_item = SharedState.m_widget.camera_props_control_list.list_.selectedItems()
244     if not selected_item:
245         return None
246     return SharedState.m_widget.camera_props_control_list.list_.row(selected_item[0])
247 
```

```
248
249 def _get_selected_cam_prop():
250     case = _get_selected_case()
251     cam_row = _get_selected_cam_prop_row()
252     return None if cam_row is None else case.cam_prop_list[cam_row]
253
254
255 def _get_selected_slice_prop_row():
256     selected_item = SharedState.m_widget.slice_props_control_list.list_.selectedItems()
257     if not selected_item:
258         return None
259     return SharedState.m_widget.slice_props_control_list.list_.row(selected_item[0])
260
261
262 def _get_selected_slice_prop():
263     case = _get_selected_case()
264     row = _get_selected_slice_prop_row()
265     return None if row is None else case.slice_list[row]
266
267
268 handlers = {
269     "handle_cases_add_button_clicked": handle_cases_add_button_clicked,
270     "handle_cases_rm_button_clicked": handle_cases_rm_button_clicked,
271     "handle_cases_item_selection": handle_cases_item_selection,
272     "handle_generate_button_clicked": handle_generate_button_clicked,
273     "handle_case_path_field_on_text_changed": handle_case_path_field_on_text_changed,
274     "handle_camera_props_add_button_clicked": handle_camera_props_add_button_clicked,
275     "handle_camera_props_rm_button_clicked": handle_camera_props_rm_button_clicked,
276     "handle_camera_props_item_selection": handle_camera_props_item_selection,
277     "handle_slice_props_add_button_clicked": handle_slice_props_add_button_clicked,
278     "handle_slice_props_rm_button_clicked": handle_slice_props_rm_button_clicked,
279     "handle_slice_props_item_selection": handle_slice_props_item_selection,
280     "handle_save_btn": handle_save_btn,
281     "handle_reset_btn": handle_reset_btn,
282     "handle_save_db_btn": handle_save_db_btn
283 }
```

A.3 foampostproc.core.model.py

```
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, cam_position: Point, viewangle: int, viewup:
17         self.idn = idn
18         self.name = name
19         self.focal_point = focal_point
20         self.cam_position = cam_position
21         self.viewup = viewup
22         self.viewangle = viewangle
23
24
25 class CameraSlice(object):
26     def __init__(self, idn, name: str, sl_x: Point = None, sl_y: Point = None, sl_z: Point = None):
27         self.idn = idn
28         self.name = name
29         self.sl_x = sl_x
30         self.sl_y = sl_y
31         self.sl_z = sl_z
```

```
32
33
34  class CasesDir(object):
35      def __init__(self, idn, path: str):
36          self.idn = idn
37          self.path = Path(path)
38
39
40  class FoamCase(object):
41      def __init__(self, idn, name: str, case_dir: CasesDir, cam_prop_list: List[CameraProps],
42                   slice_list: List[CameraSlice]):
43          self.idn = idn
44          self.name = name
45          self.cases_dir = case_dir
46          self.cam_prop_list = cam_prop_list
47          self.slice_list = slice_list
```

A.4 foampostproc.core.view.py

```
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, self.handle_header_add_button)
42          header_widget.remove_button = Button(self.RM_BUTTON_TEXT, self.handle_header_remove_button)
43
44          header_layout.addWidget(header_widget.label)
45          header_layout.addSpacing(4)
46          header_layout.addWidget(header_widget.add_button)
47          header_layout.addWidget(header_widget.remove_button)
48
49          return header_widget
50
51      def _create_list(self, list_items: List[str]) -> QtW.QListWidget:
```

```
52          list_ = QtW.QListWidget()
53          list_.setAlternatingRowColors(self.ALTERNATING_ROW_COLORS)
54          list_.setSelectionMode(QAbstractItemView.SingleSelection)
55          list_.addItems(list_items)
56          list_.itemSelectionChanged.connect(self.handle_item_selection_changed)
57          return list_
58
59      @QtC.Slot()
60      def handle_header_add_button(self):
61          self.on_add_button_clicked()
62
63      @QtC.Slot()
64      def handle_header_remove_button(self):
65          self.on_rm_button_clicked()
66
67      @QtC.Slot()
68      def handle_item_selection_changed(self):
69          self.on_item_selection()
70
71
72  class TextEdit(QtW.QWidget):
73      LABEL_SPACING = 4
74
75      def __init__(self, label_text: str, handle_on_text_changed: Callable, edit_text: str = ""):
76          super().__init__()
77          self.handle_on_text_changed = handle_on_text_changed
78
79          self.label = QtW.QLabel(label_text)
80          self.text_edit = QtW.QLineEdit(edit_text)
81          self.text_edit.textChanged.connect(self.handle_on_text_changed)
82
83          self.setLayout(QtW.QHBoxLayout())
84          self.layout().addWidget(self.label)
85          self.layout().addSpacing(self.LABEL_SPACING)
86          self.layout().addWidget(self.text_edit)
87
88      @QtC.Slot()
89      def _handle_on_text_changed(self):
90          self.handle_on_text_changed()
91
92      def clear(self):
93          self.text_edit.clear()
94
95
96  class PointTextEdit(QtW.QWidget):
97      SPACING = 4
98
99      def __init__(self, label_text: str,
100                 handle_on_text_changed1: Callable,
101                 handle_on_text_changed2: Callable,
102                 handle_on_text_changed3: Callable,
103                 text1: str = "",
104                 text2: str = "",
105                 text3: str = ""
106                 ):
107         super().__init__()
108         self.handle_on_text_changed1 = handle_on_text_changed1
109         self.handle_on_text_changed2 = handle_on_text_changed2
110         self.handle_on_text_changed3 = handle_on_text_changed3
111         self.label = QtW.QLabel(label_text)
112         self.text_edit1 = QtW.QLineEdit(text1)
113         self.text_edit1.textChanged.connect(self._handle_on_text_changed1)
114         self.text_edit2 = QtW.QLineEdit(text2)
115         self.text_edit2.textChanged.connect(self._handle_on_text_changed2)
116         self.text_edit3 = QtW.QLineEdit(text3)
117         self.text_edit3.textChanged.connect(self._handle_on_text_changed3)
118
119         self.setLayout(QtW.QHBoxLayout())
120         self.layout().addWidget(self.label)
121         self.layout().addSpacing(self.SPACING)
122         self.layout().addWidget(self.text_edit1)
123         self.layout().addSpacing(self.SPACING)
124         self.layout().addWidget(self.text_edit2)
```

```
125             self.layout().addSpacing(self.SPACING)
126             self.layout().addWidget(self.text_edit3)
127
128         @QtC.Slot()
129         def _handle_on_text_changed1(self):
130             self.handle_on_text_changed1(self.text_edit1.text())
131
132         @QtC.Slot()
133         def _handle_on_text_changed2(self):
134             self.handle_on_text_changed2(self.text_edit2.text())
135
136         @QtC.Slot()
137         def _handle_on_text_changed3(self):
138             self.handle_on_text_changed3(self.text_edit3.text())
139
140         def clear(self):
141             self.text_edit1.clear()
142             self.text_edit2.clear()
143             self.text_edit3.clear()
144
145
146     class CameraParamsForm(QtW.QWidget):
147         POSITION_FIELD_TEXT = "␣␣␣␣␣␣␣␣␣Позиция"
148         FOCAL_POINT_FIELD_TEXT = "Точка␣фокуса"
149         VIEW_ANGLE_FIELD_TEXT = "␣␣Угол␣обзора"
150         VIEW_UP_TEXT = "Наклон"
151
152         SPACING = 4
153
154         def __init__(self):
155             super().__init__()
156             self.setLayout(QtW.QVBoxLayout())
157             self.position = Point(0, 0, 0)
158             self.focal_point = Point(0, 0, 0)
159             self.view_up = Point(0, 0, 0)
160             self._view_angle = [0]
161
162             self.position_field = PointTextEdit(self.POSITION_FIELD_TEXT,
163                                                 lambda text: self.position
164                                                 .__setattr__("x", 0 if text == "" else float(text)),
165                                                 lambda text: self.position
166                                                 .__setattr__("y", 0 if text == "" else float(text)),
167                                                 lambda text: self.position
168                                                 .__setattr__("z", 0 if text == "" else float(text)))
169
170             self.focal_point_field = PointTextEdit(self.FOCAL_POINT_FIELD_TEXT,
171                                                    lambda text: self.focal_point
172                                                    .__setattr__("x", 0 if text == "" else float(text)),
173                                                    lambda text: self.focal_point
174                                                    .__setattr__("y", 0 if text == "" else float(text)),
175                                                    lambda text: self.focal_point.__setattr__("z",
176                                                                                              0 if text == "
177                                                                                                  text)))
178
179             self.view_up_field = PointTextEdit(self.VIEW_ANGLE_FIELD_TEXT,
180                                                lambda text: self.view_up.__setattr__("x", 0 if text == "" e
181                                                lambda text: self.view_up.__setattr__("y", 0 if text == "" e
182                                                lambda text: self.view_up.__setattr__("z", 0 if text == "" e
183
184             self.view_angle_field = TextEdit(self.VIEW_UP_TEXT, lambda text: self._view_angle
185                                              .__setitem__(0, 0 if text == "" else float(text)))
186
187             self.setLayout(QtW.QVBoxLayout())
188             self.layout().addWidget(self.position_field)
189             self.layout().addSpacing(self.SPACING)
190             self.layout().addWidget(self.focal_point_field)
191             self.layout().addSpacing(self.SPACING)
192             self.layout().addWidget(self.view_up_field)
193             self.layout().addSpacing(self.SPACING)
194             self.layout().addWidget(self.view_angle_field)
195
196         @property
197         def view_angle(self):
```

```
198             return self._view_angle[0]
199
200         @view_angle.setter
201         def view_angle(self, value):
202             self._view_angle[0] = value
203
204         def clear(self):
205             self.position_field.clear()
206             self.focal_point_field.clear()
207             self.view_up_field.clear()
208             self.view_angle_field.clear()
209
210
211     class SliceParamsForm(QtW.QWidget):
212         X_SLICE_LABEL_TEXT = "Срез X"
213         Y_SLICE_LABEL_TEXT = "Срез Y"
214         Z_SLICE_LABEL_TEXT = "Срез Z"
215         X_LABEL_TEXT = "x"
216         Y_LABEL_TEXT = "y"
217         Z_LABEL_TEXT = "z"
218
219         SPACING = 4
220
221         def __init__(self):
222             super().__init__()
223             self.setLayout(QtW.QVBoxLayout())
224
225             self.x_slice = Point(0, 0, 0)
226             self.x_slice_field = PointTextEdit(self.X_SLICE_LABEL_TEXT,
227                                                lambda text: self.x_slice.__setattr__("x", 0 if text == "" e
228                                                lambda text: self.x_slice.__setattr__("y", 0 if text == "" e
229                                                lambda text: self.x_slice.__setattr__("z", 0 if text == "" e
230
231             self.y_slice = Point(0, 0, 0)
232             self.y_slice_field = PointTextEdit(self.Y_SLICE_LABEL_TEXT,
233                                                lambda text: self.y_slice.__setattr__("x", 0 if text == "" e
234                                                lambda text: self.y_slice.__setattr__("y", 0 if text == "" e
235                                                lambda text: self.y_slice.__setattr__("z", 0 if text == "" e
236
237             self.z_slice = Point(0, 0, 0)
238             self.z_slice_field = PointTextEdit(self.Z_SLICE_LABEL_TEXT,
239                                                lambda text: self.z_slice.__setattr__("x", 0 if text == "" e
240                                                lambda text: self.z_slice.__setattr__("y", 0 if text == "" e
241                                                lambda text: self.z_slice.__setattr__("z", 0 if text == "" e
242
243             self.setLayout(QtW.QVBoxLayout())
244             self.layout().addWidget(self.x_slice_field)
245             self.layout().addSpacing(self.SPACING)
246             self.layout().addWidget(self.y_slice_field)
247             self.layout().addSpacing(self.SPACING)
248             self.layout().addWidget(self.z_slice_field)
249
250         def clear(self):
251             self.x_slice_field.clear()
252             self.y_slice_field.clear()
253             self.z_slice_field.clear()
254
255
256     class Button(QtW.QPushButton):
257         def __init__(self, button_text: str, handle_on_click: Callable):
258             super().__init__(button_text)
259             self.handle_on_click = handle_on_click
260             self.clicked.connect(self._handle_on_click)
261
262         @QtC.Slot()
263         def _handle_on_click(self):
264             self.handle_on_click()
265
266
267     class SaveResetButtonGroup(QtW.QWidget):
268         def __init__(self, save_btn_txt: str, save_db_btn_txt: str, handle_save: Callable, handle_save_db:
269                      handle_reset: Callable,
270                      reset_btn_text: str):
```

```
271             super().__init__()
272
273             self.handle_save = handle_save
274             self.handle_save_db = handle_save_db
275             self.handle_reset = handle_reset
276
277             self.setLayout(QtW.QHBoxLayout())
278             self.button_save = Button(save_btn_txt, handle_save)
279             self.button_save_db = Button(save_db_btn_txt, handle_save_db)
280             self.button_reset = Button(reset_btn_text, handle_reset)
281
282             self.layout().addWidget(self.button_reset)
283             self.layout().addSpacing(4)
284             self.layout().addWidget(self.button_save)
285             self.layout().addSpacing(4)
286             self.layout().addWidget(self.button_save_db)
287
288         @QtC.Slot()
289         def _handle_save(self):
290             self.handle_save()
291
292         @QtC.Slot()
293         def _handle_save_db(self):
294             self.handle_save_db()
295
296         @QtC.Slot()
297         def _handle_reset(self):
298             self.handle_reset()
299
300
301     class MainWidget(QtW.QWidget):
302         SPACING = 4
303         CASES_LABEL = "Примеры"
304         GENERATE_BTN_TEXT = "Генерация"
305         CAMERA_PROPS_LABEL = "Свойсва камеры"
306         SLICE_PROPS_LABEL = "Срезы"
307         CASE_PATH_FIELD_TEXT = "Путь"
308         SAVE_BTN_TEXT = "Сохранить"
309         SAVE_DB_BTN_TEXT = "Сохранить в БД"
310         RESET_BTN_TEXT = "Сбросить"
311
312         def __init__(self, case_list: List[str],
313                      handle_cases_add_button_clicked,
314                      handle_cases_rm_button_clicked,
315                      handle_cases_item_selection,
316                      handle_generate_button_clicked,
317                      handle_case_path_field_on_text_changed,
318                      handle_camera_props_add_button_clicked,
319                      handle_camera_props_rm_button_clicked,
320                      handle_camera_props_item_selection,
321                      handle_slice_props_add_button_clicked,
322                      handle_slice_props_rm_button_clicked,
323                      handle_slice_props_item_selection,
324                      handle_save_btn,
325                      handle_save_db_btn,
326                      handle_reset_btn,
327                      ):
328             super().__init__()
329
330             # first col ---------------------------------------------------
331             self.first_col_widget = QtW.QWidget()
332             self.first_col_widget.setLayout(QtW.QVBoxLayout())
333             self.cases_control_list = ControlList(self.CASES_LABEL, case_list,
334                                                   handle_cases_add_button_clicked,
335                                                   handle_cases_rm_button_clicked,
336                                                   handle_cases_item_selection)
337             self.generate_button = Button(self.GENERATE_BTN_TEXT, handle_generate_button_clicked)
338             self.first_col_widget.layout().addWidget(self.cases_control_list)
339             self.first_col_widget.layout().addWidget(self.generate_button)
340
341             # second col --------------------------------------------------
342             self.handle_case_path_field_on_text_changed = handle_case_path_field_on_text_changed
343             self.case_path_field = TextEdit(self.CASE_PATH_FIELD_TEXT, self.handle_case_path_field_on_text_
```

```
344
345             handle_camera_props_add_button_clicked = handle_camera_props_add_button_clicked
346             handle_camera_props_rm_button_clicked = handle_camera_props_rm_button_clicked
347             handle_camera_props_item_selection = handle_camera_props_item_selection
348             self.camera_props_control_list = ControlList(self.CAMERA_PROPS_LABEL, [],
349                                                          handle_camera_props_add_button_clicked,
350                                                          handle_camera_props_rm_button_clicked,
351                                                          handle_camera_props_item_selection)
352             self.camera_props_params_form = CameraParamsForm()
353
354             handle_slice_props_add_button_clicked = handle_slice_props_add_button_clicked
355             handle_slice_props_rm_button_clicked = handle_slice_props_rm_button_clicked
356             handle_slice_props_item_selection = handle_slice_props_item_selection
357             self.slice_props_control_list = ControlList(self.SLICE_PROPS_LABEL, [],
358                                                         handle_slice_props_add_button_clicked,
359                                                         handle_slice_props_rm_button_clicked,
360                                                         handle_slice_props_item_selection)
361             self.slice_params_form = SliceParamsForm()
362
363             handle_save_btn = handle_save_btn
364             handle_save_db_btn = handle_save_db_btn
365             handle_reset_btn = handle_reset_btn
366             self.save_reset_btn_group = SaveResetButtonGroup(self.SAVE_BTN_TEXT, self.SAVE_DB_BTN_TEXT,
367                                                              handle_save_btn, handle_save_db_btn,
368                                                              handle_reset_btn, self.RESET_BTN_TEXT)
369
370             self.second_col_widget = QtW.QWidget()
371             self.second_col_widget.setLayout(QtW.QVBoxLayout())
372             self.second_col_widget.layout().addWidget(self.case_path_field)
373             self.second_col_widget.layout().addSpacing(self.SPACING)
374
375             self.first_row_second_col_widget = QtW.QWidget()
376             self.first_row_second_col_widget.setLayout(QtW.QHBoxLayout())
377             self.first_row_second_col_widget.layout().addWidget(self.camera_props_control_list)
378             self.first_row_second_col_widget.layout().addSpacing(self.SPACING)
379             self.first_row_second_col_widget.layout().addWidget(self.camera_props_params_form)
380             self.second_col_widget.layout().addWidget(self.first_row_second_col_widget)
381
382             self.second_row_second_col_widget = QtW.QWidget()
383             self.second_row_second_col_widget.setLayout(QtW.QVBoxLayout())
384             # self.second_row_second_col_widget.layout().addWidget(self.slice_props_control_list)
385             # self.second_row_second_col_widget.layout().addWidget(self.slice_params_form)
386             self.second_col_widget.layout().addWidget(self.second_row_second_col_widget)
387
388             self.second_col_widget.layout().addWidget(self.save_reset_btn_group)
389
390             self.setLayout(QtW.QHBoxLayout())
391             self.layout().addWidget(self.first_col_widget)
392             self.layout().addWidget(self.second_col_widget)
```

A.5 foampostproc.dao.dao.py

```
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) for prop in obj_dct["camera
87              camera_slices = [dao_fact.get_dao(MongoCameraSliceDAO).read(sl) for sl in obj_dct["camera_s
88              return FoamCaseDTO(cases_dir, camera_props, camera_slices, obj_dct["name"], obj_dct["_id"])
89          except Exception:
90              raise RuntimeError("Something went wrong with object reading")
91
```

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

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

```
238
239     def create_or_update(self, dto_obj: SliceDTO):
240         result = self.create(dto_obj)
241         if not result:
242             self.update(dto_obj)
```

A.6 foampostproc.dao.daofactory.py

```
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

```
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', camera_props: List['CameraPropsDTO'],
8                   camera_slices: List['SliceDTO'], name, _id=None, ):
9          if _id is None:
10             _id = ObjectId()
11         self._id = ObjectId(_id)
12         self.cases_dir = cases_dir
13         self.camera_props = camera_props
14         self.camera_slices = camera_slices
```

```
15          self.name = name
16
17      @staticmethod
18      def parse(d: Dict):
19          try:
20              res = FoamCaseDTO(**d)
21          except Exception:
22              res = None
23          return res
24
25
26  class CasesDirDTO(object):
27      def __init__(self, cases_path: str, _id=None):
28          if _id is None:
29              _id = ObjectId()
30          self._id = ObjectId(_id)
31          self.cases_path = cases_path
32
33      @staticmethod
34      def parse(d: Dict) -> Optional['CasesDirDTO']:
35          try:
36              res = CasesDirDTO(**d)
37          except Exception:
38              res = None
39          return res
40
41
42  class PointDTO(object):
43      def __init__(self, x: float, y: float, z: float):
44          self.x = x
45          self.y = y
46          self.z = z
47
48      @staticmethod
49      def parse(d: Dict) -> Optional['PointDTO']:
50          try:
51              res = PointDTO(**d)
52          except Exception:
53              res = None
54          return res
55
56
57  class CameraPropsDTO(object):
58      def __init__(self, focal_point: PointDTO, cam_position: PointDTO, viewangle: int, viewup: PointDTO,
59                   name, _id=None):
60          if _id is None:
61              _id = ObjectId()
62          self._id = ObjectId(_id)
63          self.name = name
64          self.focal_point = focal_point
65          self.cam_position = cam_position
66          self.viewangle = viewangle
67          self.viewup = viewup
68
69      @staticmethod
70      def parse(d: Dict) -> Optional['CameraPropsDTO']:
71          try:
72              res = CameraPropsDTO(**d)
73          except Exception:
74              res = None
75          return res
76
77
78  class SliceDTO(object):
79      def __init__(self, name, sl_x: PointDTO = None, sl_y: PointDTO = None, sl_z: PointDTO = None, _id=N
80          if _id is None:
81              _id = ObjectId()
82          self._id = ObjectId(_id)
83          self.name = name
84          self.sl_x = sl_x
85          self.sl_y = sl_y
86          self.sl_z = sl_z
87
```

```
88      @staticmethod
89      def parse(d: Dict) -> Optional['SliceDTO']:
90          try:
91              res = SliceDTO(**d)
92          except Exception:
93              res = None
94          return res
95
96
97  parse_functions = [FoamCaseDTO.parse, CasesDirDTO.parse, PointDTO.parse, CameraPropsDTO.parse, SliceDTO
98
99
100 def parse_config_json_hook(dct: Dict):
101     res = None
102     for parse in parse_functions:
103         res = parse(dct)
104         if res is not None:
105             break
106
107     if res is None:
108         raise RuntimeError("Can't parse config file")
109
110     return res
```

A.8 foampostproc.dto.modelmapper.py

```
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, foam_case.name, cls.map_foam_cases_path_dto(foam_case.cases_dir)
15                          slices_)
16
17      @classmethod
18      def map_foam_case_to_dto(cls, foam_case: FoamCase) -> FoamCaseDTO:
19          cam_prop_dtos = [cls.map_camera_props_to_dto(cam_prop) for cam_prop in foam_case.cam_prop_list]
20          slice_dtos = []
21          for sl in foam_case.slice_list:
22              t = cls.map_slice_to_dto(sl)
23              slice_dtos.append(t)
24          cases_path = cls.map_foam_cases_path_to_dto(foam_case.cases_dir)
25
26          return FoamCaseDTO(cases_path, cam_prop_dtos, slice_dtos, foam_case.name, _id=foam_case.idn)
27
28      @classmethod
29      def map_foam_cases_path_dto(cls, foam_cases_path_dto: CasesDirDTO) -> CasesDir:
30          return CasesDir(foam_cases_path_dto._id, foam_cases_path_dto.cases_path)
31
32      @classmethod
33      def map_foam_cases_path_to_dto(cls, foam_cases_path: CasesDir) -> CasesDirDTO:
34          return CasesDirDTO(str(foam_cases_path.path), _id=foam_cases_path.idn)
35
36      @classmethod
37      def map_point_dto(cls, p_dto) -> Point:
38          if isinstance(p_dto, PointDTO):
39              return Point(p_dto.x, p_dto.y, p_dto.z)
40          else:
41              return Point(p_dto["x"], p_dto["y"], p_dto["z"])
42
43      @classmethod
44      def map_point_to_dto(cls, p: Point) -> PointDTO:
```

```
45          return PointDTO(p.x, p.y, p.z)
46
47      @classmethod
48      def map_slice_dto(cls, s_dto: SliceDTO) -> CameraSlice:
49          sl_x = None if s_dto.sl_x is None else cls.map_point_dto(s_dto.sl_x)
50          sl_y = None if s_dto.sl_y is None else cls.map_point_dto(s_dto.sl_y)
51          sl_z = None if s_dto.sl_z is None else cls.map_point_dto(s_dto.sl_z)
52          return CameraSlice(s_dto._id, s_dto.name, sl_x, sl_y, sl_z)
53
54      @classmethod
55      def map_slice_to_dto(cls, s: CameraSlice) -> SliceDTO:
56          sl_x = None if s.sl_x is None else cls.map_point_to_dto(s.sl_x)
57          sl_y = None if s.sl_y is None else cls.map_point_to_dto(s.sl_y)
58          sl_z = None if s.sl_z is None else cls.map_point_to_dto(s.sl_z)
59          return SliceDTO(s.name, sl_x, sl_y, sl_z, _id=s.idn)
60
61      @classmethod
62      def map_camera_props_dto(cls, c_dto: CameraPropsDTO) -> CameraProps:
63          return CameraProps(c_dto._id,
64                             c_dto.name,
65                             cls.map_point_dto(c_dto.focal_point),
66                             cls.map_point_dto(c_dto.cam_position),
67                             c_dto.viewangle,
68                             cls.map_point_dto(c_dto.viewup))
69
70      @classmethod
71      def map_camera_props_to_dto(cls, c: CameraProps) -> CameraPropsDTO:
72          focal_point_dto = cls.map_point_to_dto(c.focal_point)
73          cam_position_dto = cls.map_point_to_dto(c.cam_position)
74          viewup = cls.map_point_to_dto(c.viewup)
75          return CameraPropsDTO(focal_point_dto, cam_position_dto, c.viewangle, viewup, c.name, _id=c.idn
```

A.9 foampostproc.config.py

```
1   import configparser
2   from pathlib import Path
3   from typing import List
4
5   # gui generator config ------------------------------------------------------------------------------
6   from foampostproc.utils import PROJ_DIR
7
8   CONFIG_PATH = PROJ_DIR / Path("config/app.ini")
9
10
11  class Config:
12      __config: configparser.RawConfigParser = configparser.ConfigParser()
13      __is_read = False
14
15      @classmethod
16      def get_section(cls, section: str) -> 'ConfigSectionProxy':
17          if not cls.__is_read:
18              cls._read_config()
19          return ConfigSectionProxy(cls.__config[section])
20
21      @classmethod
22      def _read_config(cls):
23          cls.__config.read(CONFIG_PATH)
24          cls.__is_read = True
25
26
27  class ConfigSectionProxy:
28      COMMON_SECTION: str = "Common"
29      USE_PROJ_PREFIX_FOR_PATHS = "use_proj_prefix_for_paths"
30
31      __raw_config: configparser.RawConfigParser = configparser.ConfigParser()
32      __raw_config.read(CONFIG_PATH)
33      _common_section_config = __raw_config[COMMON_SECTION]
34
35      def __init__(self, section_proxy: configparser.SectionProxy):
36          self.__section_proxy = section_proxy
```

```
37
38      def get_int(self, option: str) -> int:
39          return self.__section_proxy.getint(option)
40
41      def get_float(self, option: str) -> float:
42          return self.__section_proxy.getfloat(option)
43
44      def get_boolean(self, option: str) -> bool:
45          return self.__section_proxy.getboolean(option)
46
47      def get(self, option: str) -> str:
48          return self.__section_proxy.get(option)
49
50      def get_path(self, option: str) -> Path:
51          option_path = Path(self.__section_proxy.get(option))
52          use_prefix = self._common_section_config.getboolean(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

```
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 # https://docs.paraview.org/en/v5.8.1/UsersGuide/displayingData.html?highlight=slice#slice-view
15
16 WINDOW_X = Config.get_section("ViewProperties").get_int("window_x")
17 WINDOW_Y = Config.get_section("ViewProperties").get_int("window_y")
18 WINDOW_W = Config.get_section("ViewProperties").get_int("window_w")
19 WINDOW_H = Config.get_section("ViewProperties").get_int("window_h")
20 WINDOW_TITLE = Config.get_section("ViewProperties").get("window_title")
21 TEST_DATA_PATH = Config.get_section("Paths").get_path("test_data")
22
23
24 def __run0():
25     case_dto = FileHandling.read_json(TEST_DATA_PATH, object_hook_=dto.parse_config_json_hook)
26
27
28 def __run1():
29     case_dtos = MongoDaoFactory().get_dao(MongoFoamCaseDAO).get_all()
30     cases = list(map(Mapper.map_foam_case_dto, case_dtos))
31     Screenshot.take_screenshots(cases, Config.get_section("Paths").get_path("output"))
32
33
34 def run():
35     case_dtos = MongoDaoFactory().get_dao(MongoFoamCaseDAO).get_all()
36     cases = list(map(Mapper.map_foam_case_dto, case_dtos))
37
38     app = QtW.QApplication(sys.argv)
39
40     window = QtW.QMainWindow()
41     m_widget = MainWidget([case.name for case in cases], **handlers)
42     SharedState.m_widget = m_widget
43     SharedState.case_list = cases
44
45     layout = QtW.QGridLayout()
46     central_widget = QtW.QWidget()
47     central_widget.setLayout(layout)
```

```
48      layout.addWidget(m_widget)
49      window.setCentralWidget(central_widget)
50
51      window.setWindowTitle(WINDOW_TITLE)
52      window.setGeometry(WINDOW_X, WINDOW_Y, WINDOW_W, WINDOW_H)
53      window.show()
54      sys.exit(app.exec_())
55
56
57  if __name__ == "__main__":
58      run()
```

A.11 foampostproc.utils.py

```
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)
```