

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
GRADUATION THESIS**

**Проектирование и реализация расчетного модуля системы автоматического
формирования генеральных планов площадных объектов капитального
строительства**

Обучающийся / Student Степанов Сергей Владимирович

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования

Группа/Group M42051

Направление подготовки/ Subject area 09.04.02 Информационные системы и технологии

Образовательная программа / Educational program Программирование и интернет-технологии 2020


Язык реализации ОП / Language of the educational program Русский

Статус ОП / Status of educational program

Квалификация/ Degree level Магистр

Руководитель ВКР/ Thesis supervisor Пантенков Сергей Александрович, Университет ИТМО, факультет информационных технологий и программирования, преподаватель (квалификационная категория "преподаватель практики")

Обучающийся/Student

Документ подписан	
Степанов Сергей Владимирович	
23.05.2022	

(эл. подпись/ signature)

Степанов
Сергей
Владимирович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Пантенков Сергей Александрович	
18.05.2022	

(эл. подпись/ signature)

Пантенков
Сергей
Александрович

(Фамилия И.О./ name
and surname)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**ЗАДАНИЕ НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ /
OBJECTIVES FOR A GRADUATION THESIS**

Обучающийся / Student Степанов Сергей Владимирович

Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования

Группа/Group M42051

Направление подготовки/ Subject area 09.04.02 Информационные системы и технологии

Образовательная программа / Educational program Программирование и интернет-технологии 2020

Язык реализации ОП / Language of the educational program Русский

Статус ОП / Status of educational program

Квалификация/ Degree level Магистр

Тема ВКР/ Thesis topic Проектирование и реализация расчетного модуля системы автоматического формирования генеральных планов площадных объектов капитального строительства

Руководитель ВКР/ Thesis supervisor Пантенков Сергей Александрович, Университет ИТМО, факультет информационных технологий и программирования, преподаватель (квалификационная категория "преподаватель практики")

Основные вопросы, подлежащие разработке / Key issues to be analyzed

Техническое задание: на основании определения и анализа бизнес-процессов сопутствующих проведению исследований в области автоматического формирования генеральных планов площадных объектов требуется сформировать требования к средству автоматизации. Выполнить проектирование и разработку средства автоматизации, удовлетворяющего сформированным требованиям.

Целью работы является упрощение процесса проведения научных изысканий в области автоматического формирования генеральных планов площадных объектов путём создания программного компонента.

Задачи:

- сбор требований пользователей системы,
- анализ возможной нагрузки и вариативности используемых данных,
- формирование системной и программной архитектуры,
- реализация получившегося решения.

Перечень подлежащих разработке вопросов:

- Сбор и анализ требований пользователей
- Формирование функциональных и нефункциональных требований

- Выбор и обоснование технологического стека
- Проектирование системной и программной архитектуры средства автоматизации
- Разработка спроектированных компонент средства автоматизации

Рекомендуемые материалы и пособия:

- Документация Python 3.8 [Электронный ресурс] URL: <https://docs.python.org/3.8/>
- Документация Gitlab CI/CD [Электронный ресурс] URL: <https://docs.gitlab.com/ee/ci/>
- Документация Docker [Электронный ресурс] URL: <https://docs.docker.com>
- Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2021. – 352 с.
- Мартин Р. Чистый код. Создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464 с.

Форма представления материалов ВКР / Format(s) of thesis materials:

Текст ВКР, презентация, программный код, диаграммы UML.

Дата выдачи задания / Assignment issued on: 29.11.2021

Срок представления готовой ВКР / Deadline for final edition of the thesis 12.05.2022

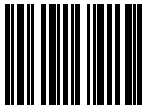
Характеристика темы ВКР / Description of thesis subject (topic)

Тема в области фундаментальных исследований / Subject of fundamental research: нет / not

Тема в области прикладных исследований / Subject of applied research: да / yes

СОГЛАСОВАНО / AGREED:

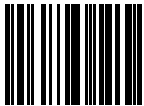
Руководитель ВКР/
Thesis supervisor

Документ подписан	
Пантенков Сергей Александрович	
16.05.2022	

(эл. подпись)

Пантенков
Сергей
Александрович


Задание принял к
исполнению/ Objectives
assumed BY

Документ подписан	
Степанов Сергей Владимирович	
16.05.2022	

(эл. подпись)

Степанов
Сергей
Владимирович

Руководитель ОП/ Head
of educational program

Документ подписан	
Парфенов Владимир Глебович	
23.05.2022	

Парфенов
Владимир
Глебович

(эл. подпись)

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

**АННОТАЦИЯ
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
SUMMARY OF A GRADUATION THESIS**

Обучающийся / Student Степанов Сергей Владимирович
Факультет/институт/кластер/ Faculty/Institute/Cluster факультет информационных технологий и программирования
Группа/Group M42051
Направление подготовки/ Subject area 09.04.02 Информационные системы и технологии
Образовательная программа / Educational program Программирование и интернет-технологии 2020
Язык реализации ОП / Language of the educational program Русский
Статус ОП / Status of educational program
Квалификация/ Degree level Магистр
Тема ВКР/ Thesis topic Проектирование и реализация расчетного модуля системы автоматического формирования генеральных планов площадных объектов капитального строительства
Руководитель ВКР/ Thesis supervisor Пантенков Сергей Александрович, Университет ИТМО, факультет информационных технологий и программирования, преподаватель (квалификационная категория "преподаватель практики")

**ХАРАКТЕРИСТИКА ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ
DESCRIPTION OF THE GRADUATION THESIS**

Цель исследования / Research goal

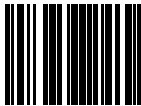
Упростить процесс проведения научных изысканий в области автоматического формирования генеральных планов площадных объектов путём создания программного компонента.

Задачи, решаемые в ВКР / Research tasks

1. Собрать и проанализировать требования пользователей системы. 2. Проанализировать возможную нагрузку и вариативность используемых данных. 3. Спроектировать системную и программную архитектуру. 4. Реализовать запроектированное решение.

Краткая характеристика полученных результатов / Short summary of results/findings

Были сформулированы требования к расчётному модулю системы автоматического формирования генеральных планов площадных объектов. На основе сформированных требований был спроектирован и реализован расчётный модуль, состоящий из пяти программных компонент: математической библиотеки, расчётной модели данных, сервиса запуска расчётных задач, сервиса хранения расчётных данных, сервиса запуска математических методов.


подписан	
Степанов Сергей Владимирович	
23.05.2022	

(эл. подпись/ signature)

Степанов
Сергей
Владимирович

(Фамилия И.О./ name
and surname)

Руководитель ВКР/
Thesis supervisor

Документ подписан	
Пантенков Сергей Александрович	
18.05.2022	

(эл. подпись/ signature)

Пантенков
Сергей
Александрович

(Фамилия И.О./ name
and surname)

СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ	6
ВВЕДЕНИЕ	7
ПОСТАНОВКА ЗАДАЧИ	8
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К СРЕДСТВУ АВТОМАТИЗАЦИИ	10
1.1 Предметная область задачи	10
1.2 Особенности выполнения расчётных задач	11
1.3 Пользователи системы и их потребности	11
1.4 Бизнес-процессы	14
1.5 Формирование требований	18
1.5.1 Функциональные требования	18
1.5.2 Нефункциональные требования	18
2 ПРОЕКТИРОВАНИЕ СРЕДСТВА АВТОМАТИЗАЦИИ	20
2.1 Стек технологий	20
2.2 Системная архитектура	24
2.2.1 Выбор архитектуры системы	24
2.2.2 Сервис запуска математических методов	27
2.2.3 Хранилище расчётных данных	29
2.2.4 Сервис запуска расчётных задач	30
2.3 Программная архитектура	34
2.3.1 Математическая библиотека	34
2.3.2 Расчётная модель данных	36
3 РАЗРАБОТКА СРЕДСТВА АВТОМАТИЗАЦИИ	39
3.1 Разработка расчётной модели данных	39

3.2	Разработка математической библиотеки	40
3.3	Разработка сервиса по запуску математических методов	42
3.4	Развертывание приложения	44
ЗАКЛЮЧЕНИЕ		47
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ		48
ПРИЛОЖЕНИЯ		50
	Приложение А. API сервисов	50
	Приложение Б. Фрагменты пользовательского интерфейса	52

ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

В данной работе применяют следующие термины с соответствующими определениями.

Генеральный план (генплан, ГП) в общем смысле — проектный документ, на основании которого осуществляется планировка, застройка, реконструкция и иные виды градостроительного освоения территорий. Основной частью генерального плана (также называемой собственно генеральным планом) является масштабное изображение, полученное методом графического наложения чертежа проектируемого объекта на топографический, инженерно-топографический или фотографический план территории[1].

Площадными объектами капитального строительства (ПО) в данной работе называются здания, строения, сооружения, а также объекты, строительство которых не завершено, за исключением некапитальных строений, сооружений и неотделимых улучшений земельного участка (замощение, покрытие и другие)[1].

Здание — результат строительства, представляющий собой объемную строительную систему, имеющую надземную и (или) подземную части, включающую в себя помещения, сети инженерно-технического обеспечения и системы инженерно-технического обеспечения и предназначенную для проживания и (или) деятельности людей, размещения производства, хранения продукции или содержания животных[2].

Сооружение — результат строительства, представляющий собой объемную, плоскостную или линейную строительную систему, имеющую наземную, надземную и (или) подземную части, состоящую из несущих, а в отдельных случаях и ограждающих строительных конструкций и предназначенную для выполнения производственных процессов различного вида, хранения продукции, временного пребывания людей, перемещения людей и грузов[2].

Строения — общее понятие зданий и сооружений[3].

ВВЕДЕНИЕ

Чтобы создать любой объект, необходимо сначала его представить. Если объект достаточно простой, то весь процесс его создания может спокойно уместиться в голове человека. Но по мере усложнения его устройства становится невозможно удержать весь контекст в голове, процесс создания начинает требовать документации. Когда объект становится совсем сложным, то его создание требует привлечения большого количества специалистов из разных областей, знакомых с тонкостями той или иной сферы деятельности.

Описанное выше применимо и для сферы строительства. Любому сложному строительству предшествует этап проектирования. Результатом этого этапа является генеральный план площадного объекта (см. рисунок 1). Именно на его основе проводится оценка, как капитальных, так и эксплуатационных затрат. За проектирование крупных технологических объектов отвечают проектные институты, а также инженеры-проектировщики, которые в них работают.

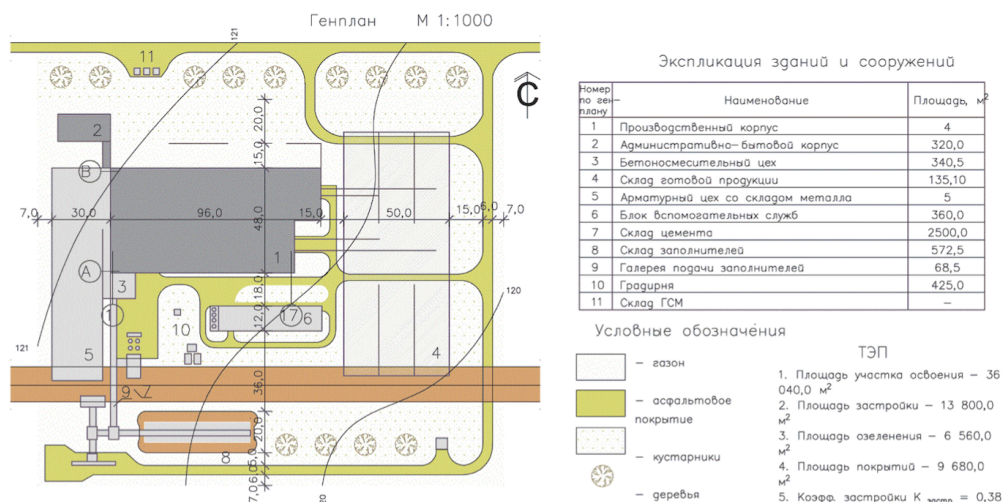


Рис. 1. Пример генерального плана площадного объекта

Основным рабочим инструментом для инженера-проектировщика являются системы автоматического проектирования. В данных инструментах отсутствует возможность полностью автоматизировать процесс построения конструкции. По сути, инженер строит решение вручную с нуля, базируясь на требованиях ГОСТ, СНиП, а также сводах правил эксплуатации того или иного сооружения, что является сложным процессом, требующим высокой квалификации специалиста.

ПОСТАНОВКА ЗАДАЧИ

Процесс проектирования генерального плана площадного объекта является непростым и требует большого объема знаний от специалиста, как в сфере государственного законодательства, так и сфере правил эксплуатации сооружений. Из-за большого количества объектов, используемых на генплане, специалисту легко допустить ошибку, которая может быть замечена лишь на поздних этапах проекта.

Набор правил минимальных допустимых расстояний между объектами является одним из основных критериев, которым руководствуются инженеры-проектировщики при размещении объекта на генплане. Основное требование заключается в нахождении объекта на достаточном удалении от остальных, с целью снижения риска для здоровья и жизни людей во время эксплуатации площадного объекта. Примером подобных правил являются требования к ограничению распространения пожара[4]. Они вычисляются на основе класса конструктивной пожарной опасности и степени огнестойкости. Но для предотвращения распространения пожара также важны активные средства, какими являются пожарные гидранты, количество и производительность которых определяется дополнительно строительным объемом сооружений.

Также при проектировании важно учитывать различные технические ограничения используемых объектов. Например, для трубопроводов такими являются диаметр, радиус поворота и пр.

У каждого проектного решения есть стоимость, включающая в себя как капитальные затраты, так и эксплуатационные. САПР не обладают встроенным инструментом стоимостного моделирования, поэтому инженер не может заранее узнать, насколько его решение экономически эффективно для площадного объекта в целом, так как полноценный расчёт затрат происходит позже на этапе формирования сметы и оказывается несколько отделенным от процесса проектирования.

Таким образом, рынок нуждается в системе способной как формировать генеральные планы в автоматическом режиме, так и верифицировать существующие решения в соответствии с заданным набором правил, а так-

же способной оценивать и сравнивать решения с точки зрения экономической эффективности.

Создание инструмента для аналитики и формирования генпланов площадных объектов позволит снизить капитальные затраты при строительстве промышленных объектов благодаря повышению качества проектирования. Подобный инструмент позволит сравнить несколько различных вариантов планировок промышленного объекта, чтобы выбрать оптимальный по ряду критериев, а также получить обоснование, почему выбранный вариант планировки действительно является оптимальным.

Формирование генплана площадного объекта в автоматическом режиме является крайне сложной алгоритмической задачей, для которой отсутствуют готовые методики решения.

Процесс создания системы для решения подобной задачи требует проведения массы исследований в сфере алгоритмов и может занимать не один год. В силу столь длительных сроков проведения научных изысканий остро стоит вопрос по созданию инструмента, способного упростить данный процесс. Процесс исследований напрямую связан с взаимодействием с техническими экспертами в области проектирования генеральных планов. Для достижения высоких результатов в сфере исследований очень важно получение обратной связи, качество и оперативность которой позволяет сделать процесс исследований максимально продуктивным.

Целью данной работы является упрощение процесса проведения научных изысканий в области автоматического формирования генеральных планов площадных объектов путём создания программного компонента.

Исходя данной цели можно выделить следующие *задачи*:

- сбор и анализ требований пользователей системы,
- анализ возможной нагрузки и вариативности используемых данных,
- формирование системной и программной архитектуры,
- реализация полученного решения.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ФОРМИРОВАНИЕ ТРЕБОВАНИЙ К СРЕДСТВУ АВТОМАТИЗАЦИИ

1.1 Предметная область задачи

Задача по автоматическому формированию генерального плана площадного объекта представляет существенные трудности не только в алгоритмическом плане, но еще и в техническом. Для расчёта генплана необходимо учесть большое количество входных данных. Ниже представлен пример набора входных данных:

- допустимая для строительства область на карте;
- стоимостная модель расчета стоимости инженерной подготовки;
- перечень сооружений с указанием габаритов (ширина, длина, радиус), степени огнестойкости, категорий взрывопожарной и пожарной опасности, конструктивной пожарной опасности;
- параметры технологических коммуникаций между сооружениями проектируемого объекта (электрокабели, трубопроводы и т.д.);
- местоположения внешних точек подключения площадного объекта (дорога, воздушная линия, трубопроводы внешнего транспорта и т.п.);
- параметры цифровой модели рельефа;

Результатом работы системы является генеральный план площадного объекта. Он, в свою очередь, состоит из множества технологических элементов. Ниже представлены примеры объектов, которые могут быть включены в генплан:

- фигура площадного объекта, содержащая заданные сооружения;
- местоположения сооружений;

- схема технологических эстакад минимальной длины, соединяющей все сооружения, заданные пользователем;
- схема внутриплощадочных проездов;
- стоимость инженерной подготовки, коэффициент застройки территории и длина технологических связей;
- зоны распространения теплового потока;
- зоны распространения взрывной волны;

Все перечисленные выше объекты очень разнородны по своему содержанию и структуре.

1.2 Особенности выполнения расчётных задач

Каждый объект на генеральном плане имеет собственную методику расчёта. Для одних объектов методика расчета предоставляется заказчиком, для других формируется в результате проведения исследований и экспериментов.

Результат работы ряда методик зависит от результата выполнения предыдущих. Например, вычисление местоположения ограждений для групп сооружений невозможно без рассчитанного местоположения сооружений.

В силу сложности задачи построения генплана возникают типичные проблемы использования сложных математических алгоритмов, обрабатывающих большой набор входных данных: долгое время выполнения, высокую нагрузку на центральный процессор и высокий уровень потребления оперативной памяти.

1.3 Пользователи системы и их потребности

На текущий момент невозможно полноценно реализовать систему по автоматическому формированию генеральных планов в силу отсутствия методики. Для выработки этой методики требуется провести ряд исследований в области алгоритмов формирования генпланов.

Любые исследования для решения промышленных задач связаны с активной консультацией с техническими экспертами, а также требуют грамотного оформления всех результатов и обеспечения их воспроизводимости.

На время научных изысканий система будет являться исследовательским прототипом. Она должна позволять интерактивно отображать полученные результаты и фиксировать проведенные эксперименты.

Можно выделить три группы пользователей, которые будут взаимодействовать с системой:

1. *Технические эксперты со стороны заказчика.* Они обладают профессиональными знаниями в сфере проектирования генеральных планов площадных объектов. На их экспертизе базируется итоговое качество получаемого решения.
2. *Аналитики.* Они являются связкой между техническими экспертами и исследователями. Именно они презентуют полученные результаты техническим экспертам и подготавливают данные в том формате, которым могут воспользоваться исследователи.
3. *Исследователи.* Основной их деятельностью является решение именно математической задачи, применение и обоснование методики, которая будет давать наилучший результат.

С каждой из групп пользователей было проведено интервью с целью выяснения их потребностей в работе с системой, и был составлен список этих потребностей, который представлен ниже.

1. *Технические эксперты со стороны заказчика*
 - хотят иметь интерактивный доступ к результатам исследований на различных кейсах.
2. *Аналитики*
 - хотят иметь возможность загрузить данные, полученные от технических экспертов,

- хотят оперативно видеть результаты работы команды исследователей,
- хотят проводить анализ результатов различных методик, полученных на разных этапах развития проекта.

3. Исследователи

- хотят иметь удобный и простой доступ к данным, полученные от технических экспертов,
- хотят иметь простой способ для отправки результатов новой методики для дальнейшего анализа команде аналитиков,
- хотят иметь возможность оперативно добавлять новые методики в действующий функционал системы.

На основе потребностей пользователей можно составить диаграмму вариантов использования системы(см. рис 2).

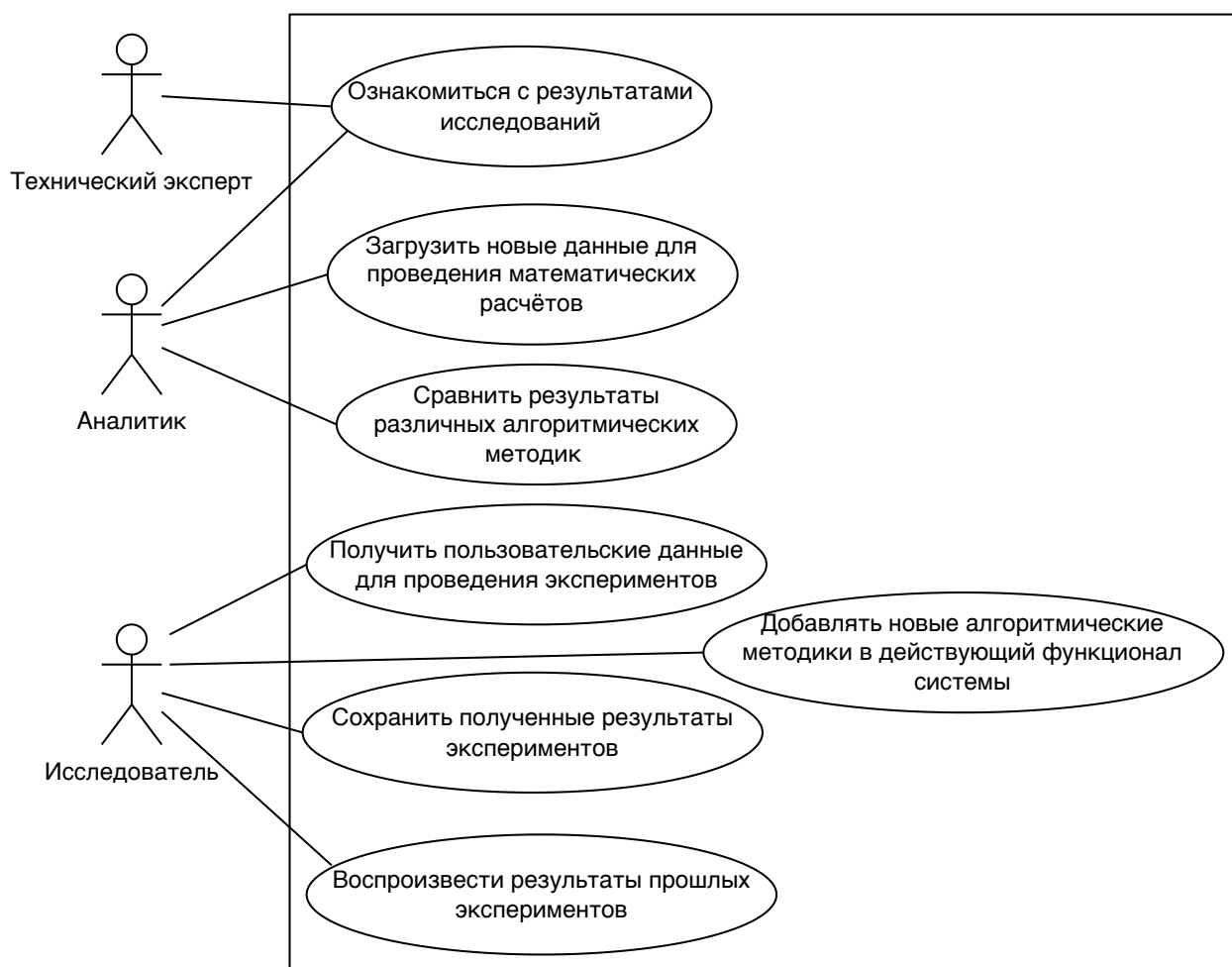


Рис. 2. Диаграмма вариантов использования

1.4 Бизнес-процессы

Наиболее распространёнными вариантами использования системы являются следующие:

1. Добавлен новый или изменены требования размещения к существующему элементу генерального плана.
2. Подбор эффективных методик расчёта для конкретного набора входных данных.

Первый вариант использования системы порождает процесс проведения исследований и внедрения алгоритмической методики. Ниже представлена ЕРС-диаграмма этого процесса(см. рисунок 3).

Главным инициатором исследований является заказчик. Именно заказчик знает, где найти людей, обладающих экспертными знаниями в проектировании генеральных планов. Заказчик так же определяет приоритет добавления новых элементов на генплане площадного объекта, а также набор требований к их размещению.

Сначала заказчик решает, требуется ли ему улучшить качество размещения уже существующего элемента или добавить на генплан новый элемент. Затем заказчик сообщает аналитику контактную информацию технического эксперта. Технический эксперт предоставляет презентативные данные аналитику для отладки методики, способной удовлетворить требованиям заказчика.

После получения данных от технического эксперта аналитик оформляет их в виде доступном для загрузки в расчётный модуль. После того как данные загружены в расчётный модуль, исследователи могут приступать к выработке решения правильного размещения заданного объекта.

У исследователя всегда есть два варианта решения поставленной задачи: создать новую методику решения или использовать уже существующую. После того как сделан выбор, проводится ряд экспериментов, наиболее удачные результаты экспериментов сохраняются и отправляются на рассмотрение аналитику.

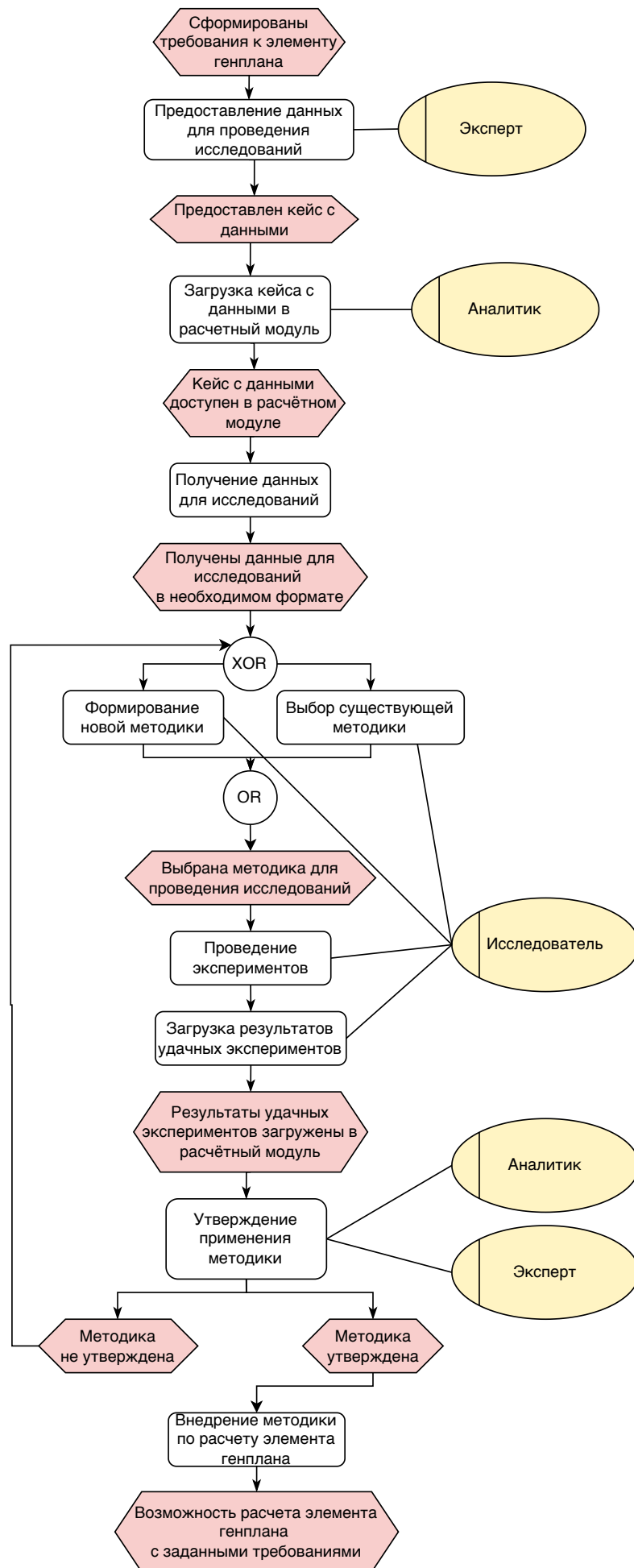


Рис. 3. Диаграмма процесса внедрения алгоритмической методики

Если аналитик делает вывод, что данная методика, позволяет получить решение, соответствующее требованиям заказчика, то результаты эксперимента уже демонстрируются техническому эксперту со стороны заказчика, чтобы он дополнительно убедился, что реализация данной методики не противоречит другим требованиям проектирования генпланов.

Если замечаний к результату нет, то методика встраивается в существующее решение. Если же есть какие-либо недостатки, то они фиксируются и устраняются.

Второй вариант использования системы порождает бизнес-процесс выбора наиболее эффективных методик расчёта для конкретного набора входных данных. Он изображен на ЕРС-диаграмме(см. рисунок 4).

Все действия изображенные на этой диаграмме выполняются аналитиком в системе автоматического расчёта генерального плана площадного объекта. Входные данные задачи определены, на них происходит проведение анализа. Аналитик выбирает необходимые этапы задачи и методы для проведения исследований, производит расчёты, анализирует получившееся решение. На основании определенных критериев аналитик выбирает наиболее эффективный набор методик для определенного набора входных данных.

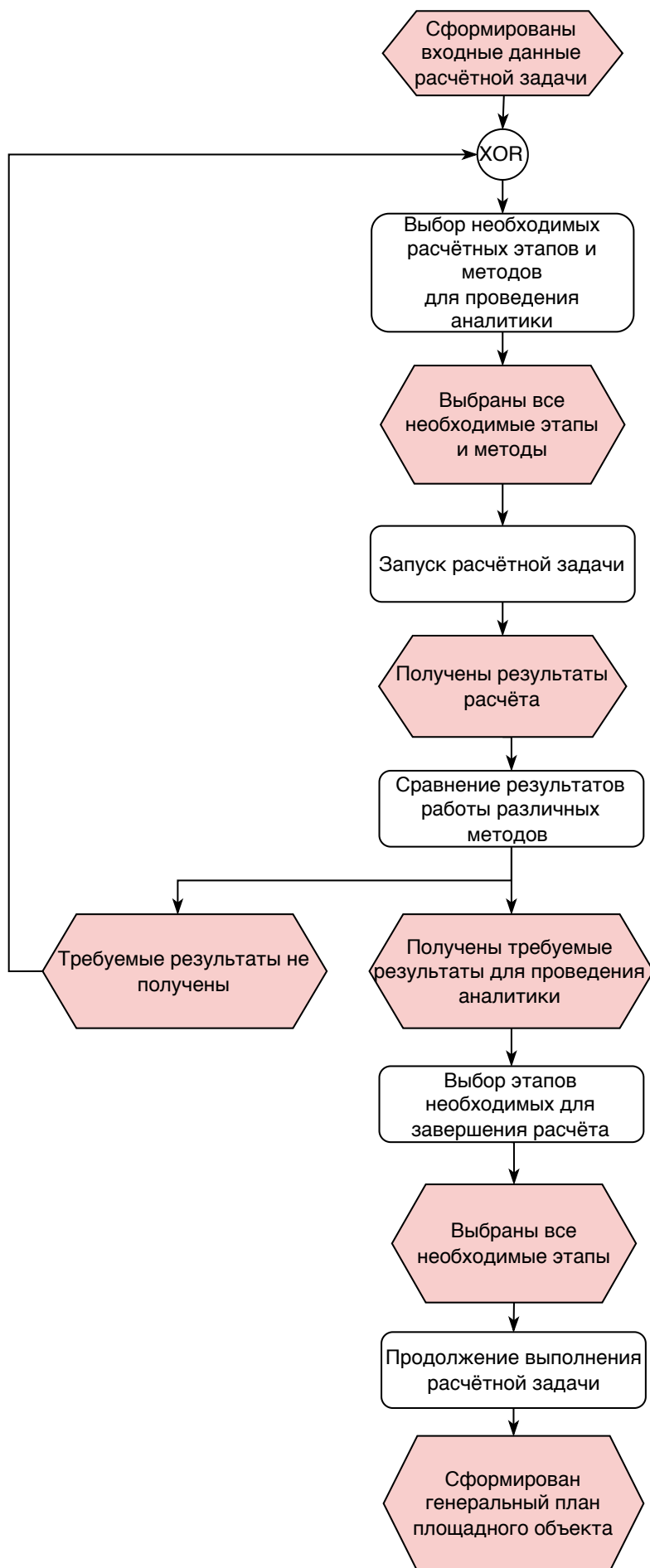


Рис. 4. Диаграмма процесса выбора эффективной методики расчёта

1.5 Формирование требований

1.5.1 Функциональные требования

На основе анализа сформированы следующие функциональные требования:

- Должна присутствовать возможность расчёта генерального плана площадного объекта в автоматическом режиме.
- Автоматический расчёт генерального плана площадного объекта должен быть разбит на этапы.
- Результат каждого этапа расчёта должен быть сохранён в долговременное хранилище.
- Должна присутствовать возможность продолжить расчёт с последнего успешно завершённого этапа.
- Должна быть возможность сравнения одинаковых расчётных объектов, полученных с путем применения разных методик.
- Должна присутствовать возможность загрузки данных, полученных от технических экспертов, в расчётный модуль.
- Должна присутствовать возможность загрузки результатов экспериментов, а также информации об особенностях проведения экспериментов в расчётный модуль.

1.5.2 Нефункциональные требования

Из нефункциональных требований отметим наиболее важные, которые определяют сам процесс проектирования системы и выбора технологий для реализации.

1. *Вычислительные ресурсы.* Одним из требований заказчика является использование собственных серверов компании. Для проведения исследований выделен один вычислительный сервер на операционной системе Ubuntu 20.04 LTS.

2. *Особенность выполнения расчётов.* Вызов алгоритмически сложной части системы должен осуществляться в отдельном процессе.
3. *Временные ресурсы.* Основной задачей проекта является демонстрация перспективности развития применяемого набора математических методов. Эту перспективность требуется показать в короткие сроки.
4. *Гибкость системной архитектуры.* Проект является исследовательским, поэтому отсутствует достаточно глубокий уровень проработки предметной области для качественного проектирования системы. Необходимо предусмотреть высокий уровень гибкости системы, так как требования к системе могут оперативно и непредсказуемо изменяться.
5. *Требования к отчётной документации.* Алгоритмы являются основным результатом деятельности в проекте. Все разработанные алгоритмы должны быть оформлены в отдельную библиотеку. Библиотека должна быть версионизируема.

2 ПРОЕКТИРОВАНИЕ СРЕДСТВА АВТОМАТИЗАЦИИ

2.1 Стек технологий

Во время проектирования системы всегда возникает ряд вопросов, связанных с выбором стека технологий для решения следующих задач.

1. Реализация программного кода
2. Хранение данных
3. Развертывание
4. Тестирование
5. Логирование
6. Документация
7. Протоколы взаимодействия

Рассмотрим каждую из задач более подробно и выберем технологию, которая наиболее подходит для её решения в нашем случае.

1. Язык программирования.

В качестве основного языка программирования был выбран *Python* 3.8[5]. Самым ценным ресурсом при реализации нашей системы является время разработки. Python является языком программирования с динамической типизацией, имеет простой синтаксис, а также очень большую базу готовых библиотек для решения любых задач, в том числе и математических. Одними из основных библиотек для работы с алгоритмами являются *numpy*, предоставляющий удобные механизмы для работы с массивами любой размерности, и *scipy*, предоставляющий набор алгоритмов из различных областей математики.

Все эти преимущества позволяют, как быстро реализовывать функционал со стороны разработки, так и проводить проверку разных алгоритмических гипотез за короткое время.

Такие недостатки Python, как невысокая скорость выполнения программ и избыточное потребление памяти[6], в нашем случае не играют решающей роли, в силу исследовательской направленности проекта. В Ubuntu 20.04 LTS – операционной системе, на которой ведется разработка, установлен по умолчанию Python 3.8, поэтому была выбрана именно эта версия языка.

2. Хранение данных.

Данные, используемые в проекте, хорошо структурируемы и могут быть легко представлены в табличном виде. Поэтому в качестве технологии для хранения данных будем использовать реляционную базу данных. Так как проектирование генеральных планов напрямую связано с геоданными, необходимо учитывать и это требование.

В качестве технического решения предлагается связка *PostgreSQL 12* и расширения для работы с геоданными *PostGIS 3.1*. PostgreSQL является популярным open-source решением в мире баз данных, а PostGIS содержит большой набор функций для преобразований геоданных[7].

3. Развертывание.

Необходимо предусмотреть механизм автоматического развертывания. В компании используется система управления git-репозиториями *GitLab*. В этой системе доступен набор инструментов CI/CD, позволяющих решить данные задачи. Он называется *GitLab CI/CD*. Современным стандартом развертывания и управления приложениями является технология контейнеризации Docker[9].

Для объединения нескольких *Docker*-контейнеров вместе можно воспользоваться средствами *docker-compose*. Более сложные инструменты оркестрации контейнеров, такие как *kubernetes* нам не требуются, в силу того, что нам доступен только один вычислительный сервер.

4. Тестирование.

Наша система состоит из нескольких компонентов. Поэтому нужно предусмотреть следующие виды тестирования[8].

- *Модульное тестирование.* Для проверки корректности работы различных функций и методов.

- *Функциональное тестирование.* Для проверки соответствия результатов методов бизнес-требованиям.
- *Интеграционное тестирование.* Для проверки, что все компоненты системы работают согласованно.

В языке Python современным стандартом запуска тестов является библиотека `pytest`.

Gitlab CI/CD имеет интеграцию с отчётами по тестированию в формате *JUnit*. Будем использовать данный формат для отображения результатов модульного тестирования. Библиотека `pytest` имеет плагин `pytest-junit` для генерации отчётов в подобном формате.

Для отображения отчётов о функциональном тестировании воспользуемся инструментом *Allure*. Данный инструмент используется в компании для визуализации отчётов о функциональном тестировании. Библиотека `pytest` имеет плагин `pytest-allure` для генерации отчётов в подобном формате.

5. Логирование.

Помимо логирования в коде, требуется подумать о системе сбора, агрегации и хранения логов. Одним из лучших решений в современной разработке является ELK. Это решение состоит из трех компонентов: `ElasticSearch` для полнотекстового поиска и аналитики, `Logstash` для сбора логов из приложений, `Kibana` для визуализации логов в веб-интерфейсе.

6. Документация.

Необходимо предусмотреть возможность генерации документации API и документации математической библиотеки.

В первом случае, есть стандартные средства, как `Swagger` или `GraphiQL`, которые позволяют автоматически генерировать документацию на основе модели данных.

Во втором случае требуется сгенерировать описание алгоритмов, используемых в программе. Важная особенность в документировании алгоритмов заключается в том, что так как они зачастую очень сложны в понимании только по текстовому описанию и требуют наличия иллюстраций. Это удобно делать в `Markdown`- и `LaTeX`-файлах. Также нужно учесть,

что в алгоритмах присутствует не только теоретическое обоснование, но и реализация, которая также требует документации, но уже стандартными средствами разработки.

Чтобы объединить в себе два формата документации, теоретическую часть и реализацию, воспользуемся инструментом Sphinx[10].

7. Протоколы взаимодействия.

В качестве основного протокола взаимодействия между сервисами будет использоваться REST API over HTTP как легковесный и широко распространенный. В качестве формата данных для передачи между веб-клиентом и сервером выберем JSON. Выберем его по следующим причинам:

- Данный формат является человеко-читаемым и легко редактируется с помощью обычного текстового редактора, что в исследовательских задачах очень важно.
- В силу малого количества пользователей в системе, нам не требуется уделять особое внимание размеру передаваемых пакетов данных, что предоставляют бинарные форматы, такие как protobuf[11].
- Данный формат является очень популярным в индустрии и он имеет поддержку во всех современных языках программирования.

Итого при разработке системы будут использоваться следующий стек технологий:

- Операционная система *Ubuntu 20.04 LTS*
- Язык программирования *Python 3.8.12*
- База данных *PostgreSQL 12* с расширением *PostGIS 3.1*
- Автоматическое развертывание *Gitlab CI/CD*
- Контейнеризация *Docker* и *docker-compose*
- Документация *Swagger* и *Sphinx*
- Тестирование *pytest*, *JUnit*, *Allure*

- Система сборки логов *ELK*
- Протоколы взаимодействия *REST API over HTTP*
- Формат данных *JSON*

2.2 Системная архитектура

2.2.1 Выбор архитектуры системы

Общий вид системы для автоматического проектирования генеральных планов площадных объектов можно представить в виде двух независимых друг от друга частей: бизнес-части и расчётной. Эти части системы схематично изображены на диаграмме ниже(см. рисунок 5).

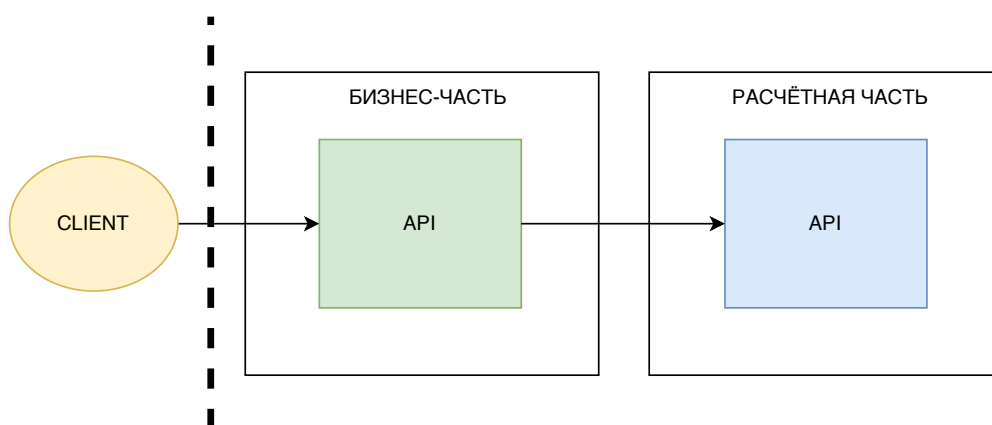


Рис. 5. Общий вид системы

Бизнес-часть приложения используется для отображения полученных результатов внешним экспертам со стороны заказчика. Именно в этой части определены все правила и те объекты, которые используются в экспертной оценке. В расчётной части системы используются объекты внутренней модели данных. Эти внутренние объекты уже не могут быть интерпретированы внешними экспертами.

Также стоит отметить, что внутренняя модель данных подвержена сильным изменениям в процессе исследований. Новые алгоритмы решения задачи могут давать куда более качественный результат, но только с учетом того, что будут использованы дополнительные структуры данных. В то время как внешняя модель данных относительно стабильна и представлена объектами целиком и полностью понятными внешним экспертам.

Целью данной работы является проектирование и реализация только расчётной части системы, поэтому на всех остальных диаграммах представленных в работе рассматривается именно эта часть.

Для удовлетворения описанных в предыдущей главе требований предлагается архитектура системы, использующая микросервисный подход. Данный подход позволяет разбить систему на отдельные блоки, каждый из которых будет выполнять только свою задачу и не зависеть от функционала других блоков. Независимость каждого из сервисов существенно упростит процесс разработки и отладки в силу изолированности каждой задачи. Помимо этого каждый сервис может быть развернут отдельно от остальных, в зависимости от требований распределения нагрузки. Микросервисный подход к разработке позволит разрабатывать автономные и слабосвязанные друг с другом сервисы.

Для решения поставленной задачи предлагается следующая архитектура расчётного модуля(см. рисунок 6).

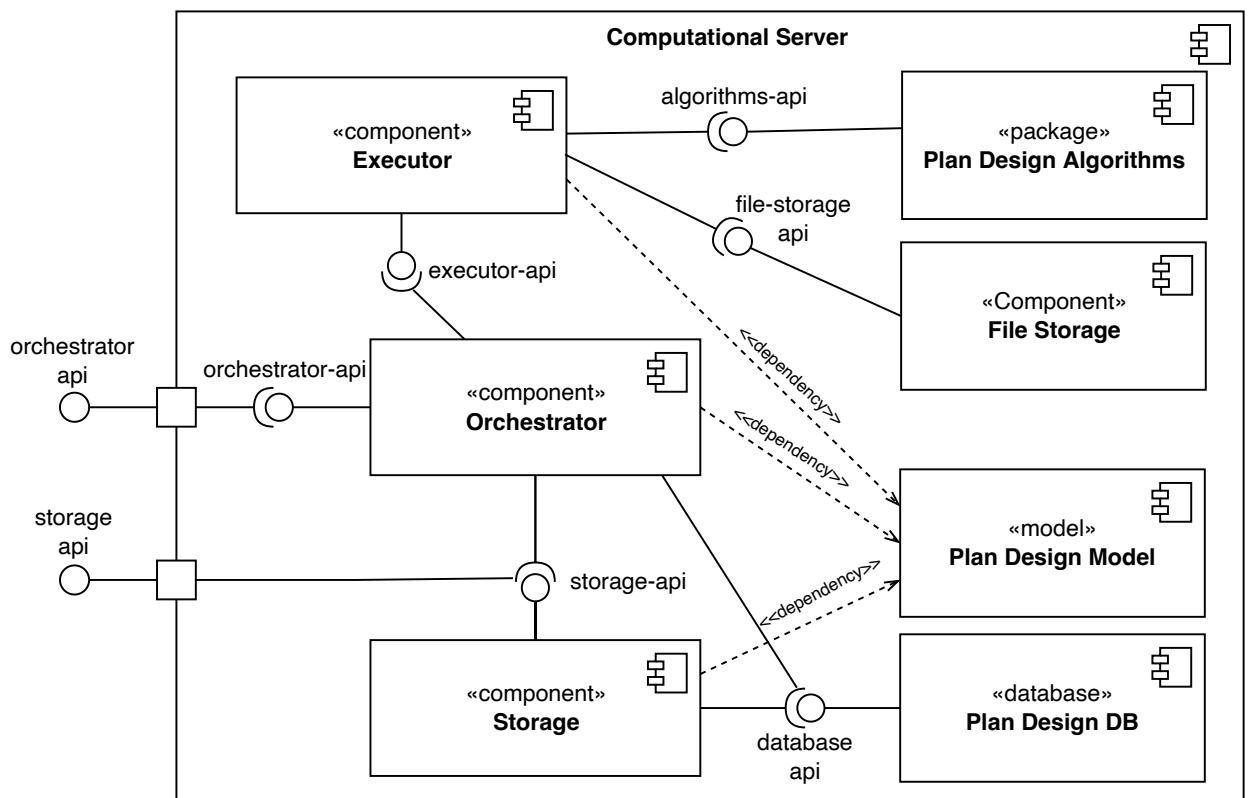


Рис. 6. Верхнеуровневая диаграмма компонентов

Схема развертывания компонентов, указанных выше, изображена на диаграмме развёртывания(см. рисунок 7).

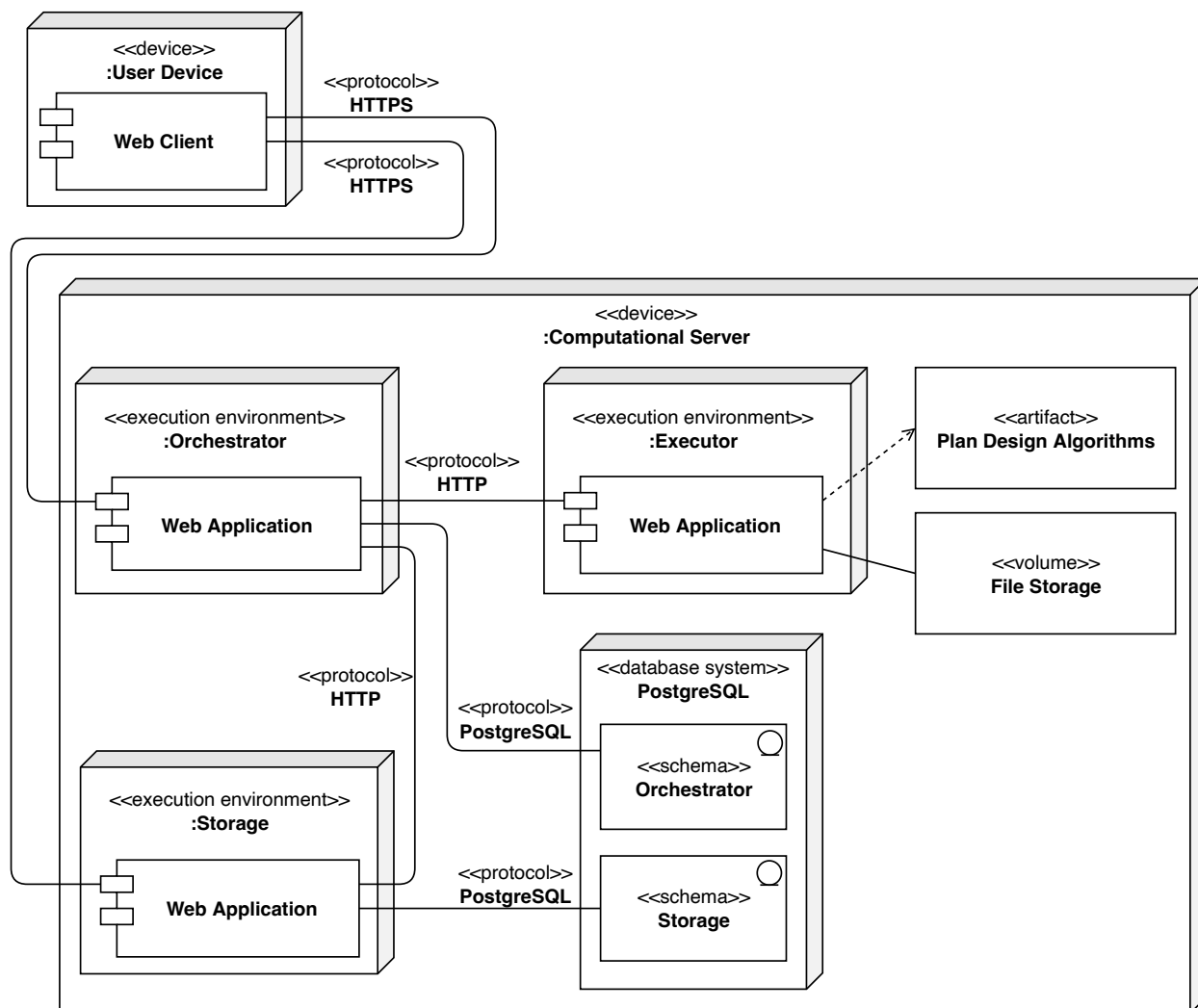


Рис. 7. Диаграмма развёртывания

Объекты представленные на диаграмме компонентов:

- *Computational Server* – сервер для запуска расчётных задач.
- *Plan Design Algorithms* – математическая библиотека, содержащая алгоритмы для построения генеральных планов.
- *Plan Design Model* – расчётная модель данных.
- *Executor* – сервис, отвечающий за запуск методов математической библиотеки.
- *Storage* – сервис, отвечающий за чтение/запись расчётной модели данных.
- *Orchestrator* – сервис, контролирующий выполнение расчётных задач.
- *File Storage* – файловое хранилище.

- *Plan Design DB* – база данных.

Самой ресурсоёмкой частью приложения является запуск математических методов. Сервис *Executor*, отвечающий за это, отвязан от использования других компонент системы и может работать полностью изолированно. Поэтому, в случае необходимости, возможно оперативное развертывание нескольких экземпляров данного сервиса на других вычислительных узлах.

Так как проект является исследовательским, то основными пользователями является немногочисленная команда проекта. Поэтому использование дополнительных архитектурных компонент, таких как балансировщики нагрузки и системы автоматического развертывания новых экземпляров приложения здесь просто не требуется.

Основной причиной применения микросервисного подхода в данном случае является изолированность задач с целью упрощения проведения исследований. В монолитном приложении из-за высокой связности между компонентами спустя время становится крайне сложно добавление нового функционала. В исследовательских проектах в силу неопределенных требований это очень важное ограничение, которое в какой-то момент просто может не позволить дальше продолжать научные изыскания без переработки архитектуры проекта.

2.2.2 Сервис запуска математических методов

Верхнеуровневая архитектура сервиса запуска математических методов представлена на диаграмме компонент (см. рисунок 8).

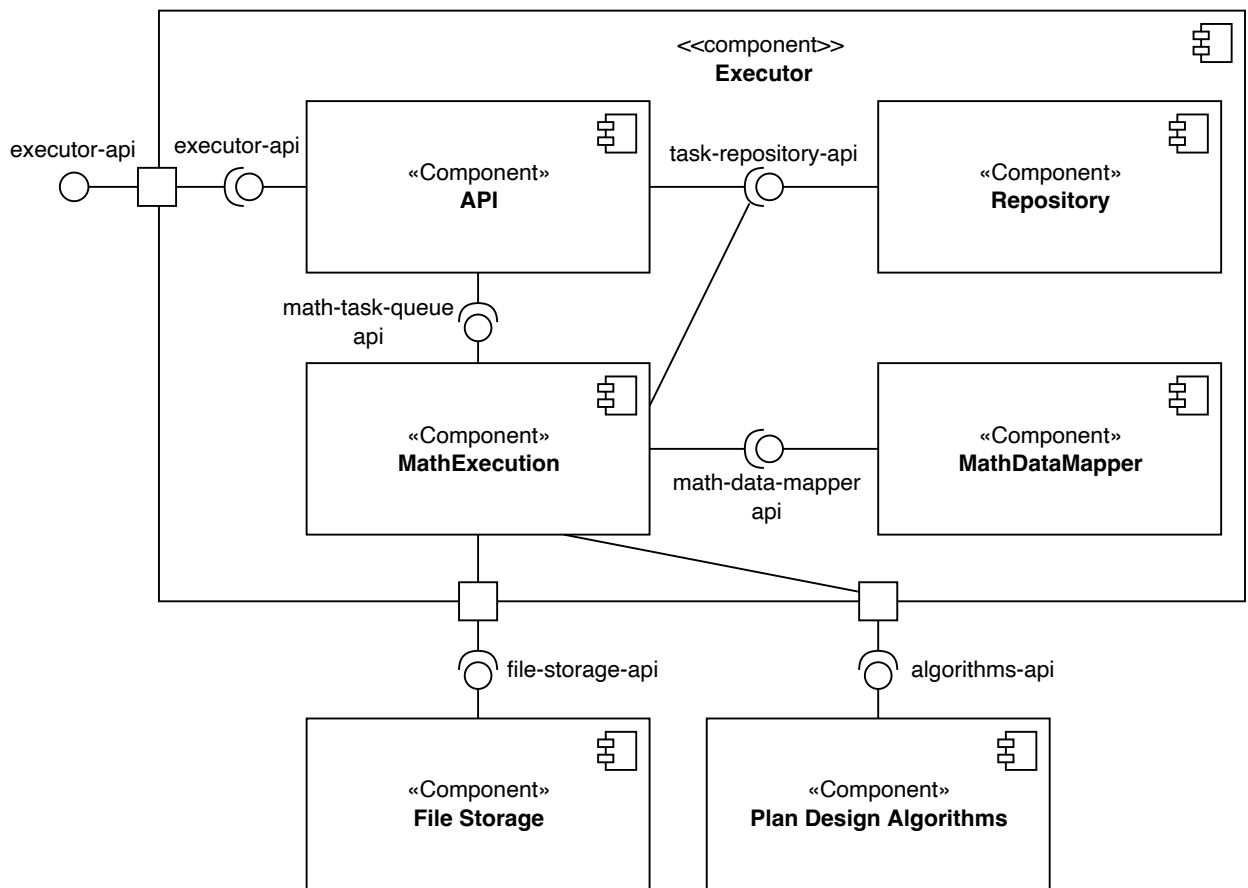


Рис. 8. Верхнеуровневая диаграмма компонент сервиса запуска математических методов

Сервис представлен следующими компонентами:

1. *API* – отвечает за предоставление REST API и отправки задачи в очередь.
2. *Repository* – отвечает за получение и сохранение данных.
3. *MathDataMapper* – отвечает преобразование расчётной модели данных в модель данных математических методик.
4. *MathExecution* – отвечает за запуск математических методов в отдельном процессе.

Самой технически сложной частью сервиса является компонент, отвечающий за запуск математических методов в отдельном процессе. Детальное изображение его компонентов представлено на диаграмме ниже (см. рисунок 9).

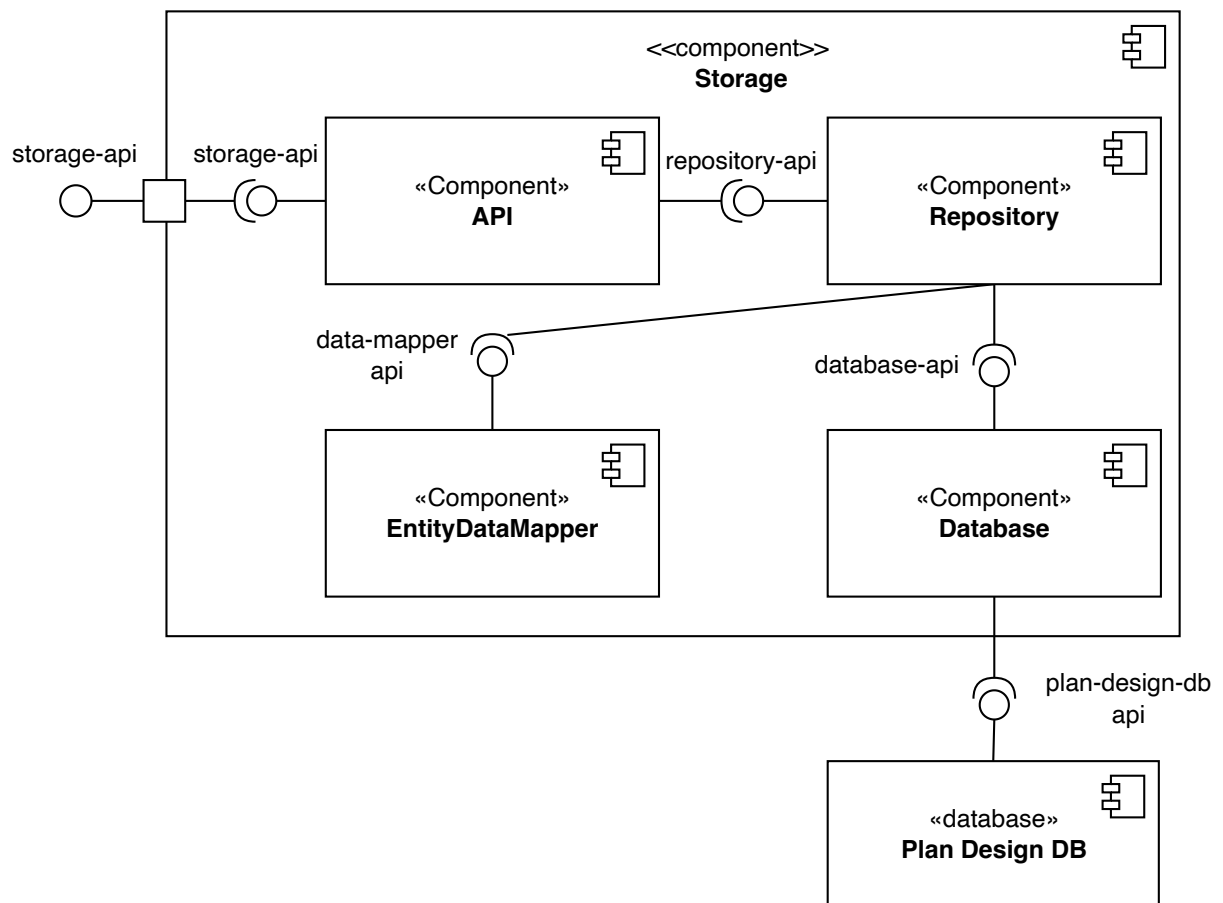


Рис. 10. Диаграмма компонент хранилища расчётных данных

Сервис представлен следующими компонентами:

1. *API* – отвечает за предоставление REST API.
2. *Repository* – отвечает за получение и сохранение данных, а также логику их преобразований.
3. *EntityDataMapper* – преобразование сущностей базы данных в сущности расчётной модели данных.
4. *Database* – чтение/сохранение сущностей базы данных.

2.2.4 Сервис запуска расчётных задач

Архитектура сервиса запуска расчётных задач представлена на диаграмме компонентов (см. рисунок 11).

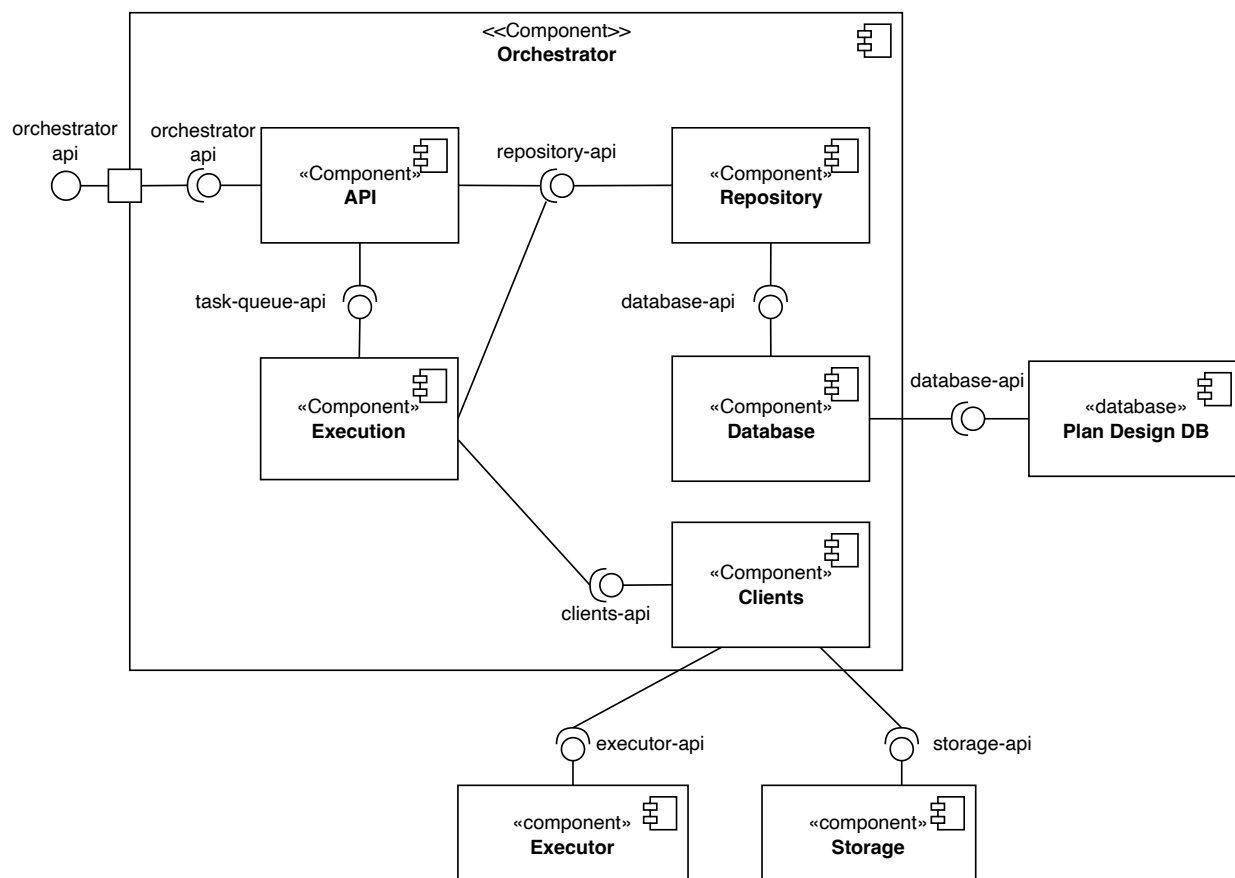


Рис. 11. Диаграмма компонентов сервиса запуска расчётных задач

Сервис представлен следующими компонентами:

1. *API* – отвечает за предоставление REST API и отправки задачи в очередь.
2. *Repository* – отвечает за получение и сохранение данных и логику их преобразований.
3. *Execution* – компонент, отвечающий за выполнение расчётных задач.
4. *Clients* – клиенты для взаимодействия с хранилищем расчётных данных и сервисом запуска математических методов.
5. *Database* – чтение/сохранение сущностей базы данных.

Для расчёта генерального плана площадного объекта в автоматическом режиме используются четыре типа сущностей, представленные на диаграмме ниже(см. рисунок .12)

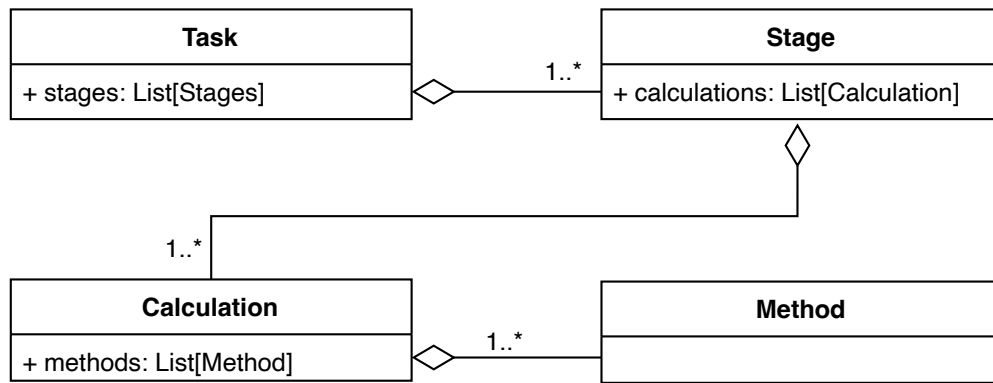


Рис. 12. Диаграмма классов основных сущностей сервиса запуска расчётных задач

- *Task* – расчётная задача. Результатом выполнения задачи является генеральный план площадного объекта. Задача состоит из набора этапов, выполняющихся строго последовательно. Результат выполнения следующего этапа зависит от результатов предыдущего.
- *Stage* – этап расчётной задачи. Для каждого этапа определен хотя бы один расчёт. Расчёты могут выполняться параллельно. Результатом этапа может быть только один расчёт.
- *Calculation* – расчёт. Состоит из последовательности методов.
- *Method* – метод. Способ применения математической методики.

Такое разбиение на классы обусловлены высокой сложностью и вариативностью расчёта генплана.

Наименьшим блоком в расчёте генерального плана является метод. Метод является одним из способов применения определенной математической методики, которая представлена в математической библиотеке.

Блок, который отвечает за последовательность вызова математических методик называется расчёт. Для получения результата более высокого качества требуется иметь возможность последовательно выполнять несколько математических методик. Например, для того чтобы рассчитать правила минимальных допустимых расстояний с учётом зон теплового излучения требуется вычислить параметры этих зон, а затем на основе полученных параметров произвести расчёт минимальных допустимых расстояний. Таким образом последовательно вызываются две методики для получения одного расчётного элемента.

Объектом, который объединяет расчёты является этап. Этап состоит хотя бы из одного расчёта. В рамках этапа расчёты могут выполняться параллельно. Входные данные расчёта зависят только от результата выполнения предыдущих этапов задачи. Результатом этапа может быть только один результат расчёта.

Задача представлена набором этапов. Все этапы выполняются строго последовательно, результат последующего этапа зависит от результата выполнения предыдущих.

Для оперирования с указанными сущностями предлагается следующая архитектура компонентов(см. рисунок 13).

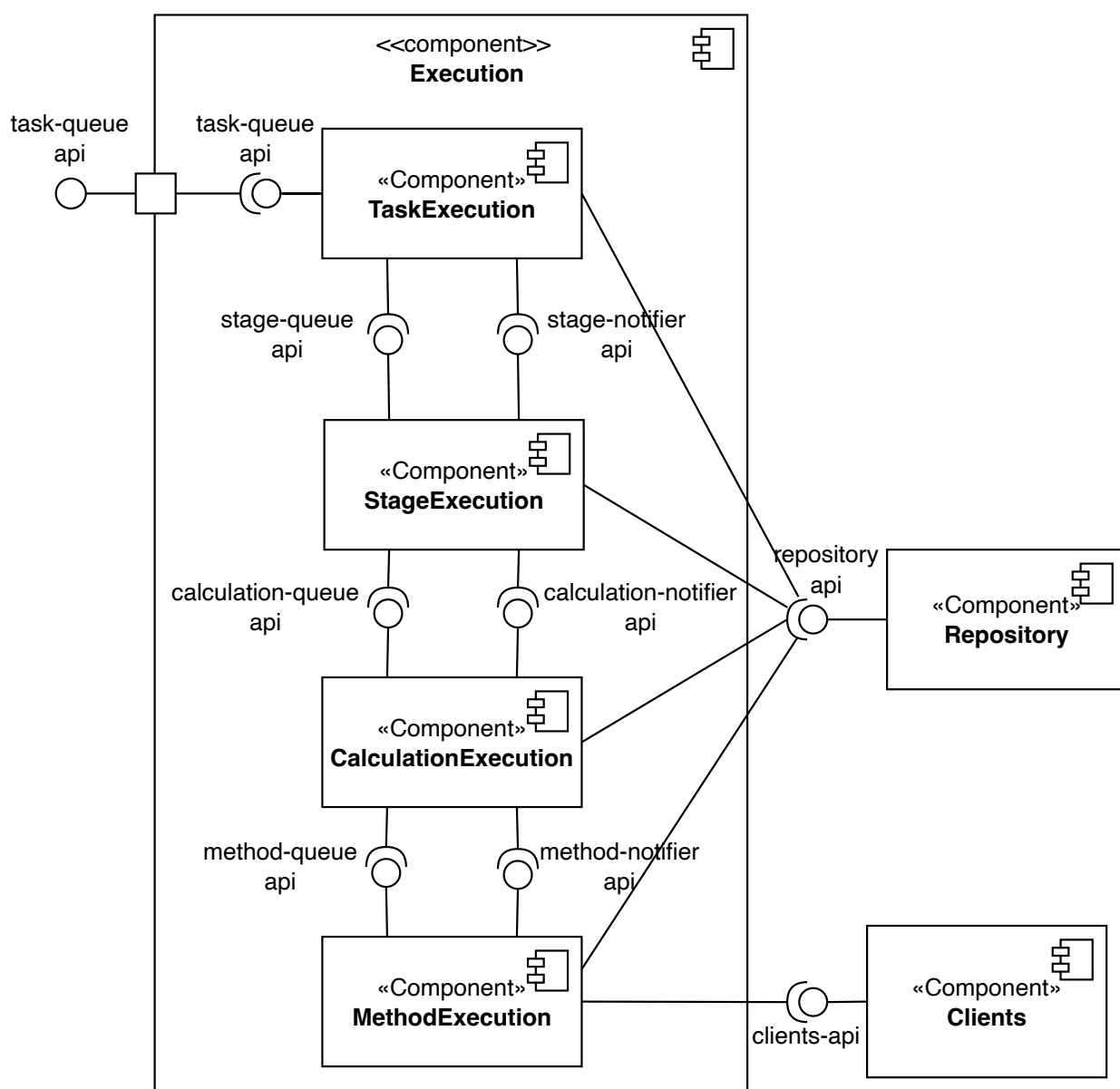


Рис. 13. Диаграмма компонентов запуска расчётных задач

1. *Task Execution* – выполнение расчётной задачи.

2. *Stage Execution* – выполнение этапа задачи.
3. *Calculation Execution* – выполнение расчёта.
4. *Method Execution* – выполнение математической методики.

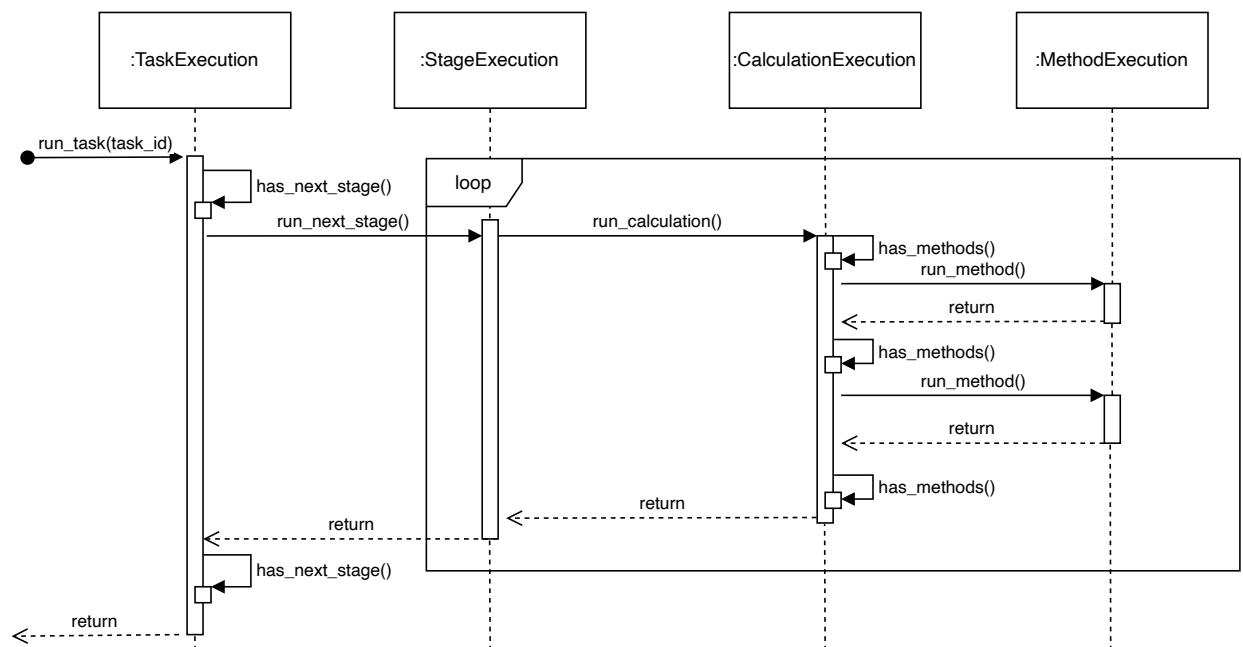


Рис. 14. Диаграмма последовательности выполнения расчётной задачи

Выше представлена диаграмма последовательности выполнения расчётной задачи (см. рисунок 14). Для её выполнения происходит вызов компонента *TaskExecution*. Пока присутствуют нерасчитанные этапы, происходит последовательный вызов компонента *StageExecution*. В рамках выполнения этапа расчётной задачи вызывается компонент, отвечающий за выполнение расчётов *CalculationExecution*. Для выполнения одного расчёта последовательно вызывается компонент *MethodExecution*.

2.3 Программная архитектура

2.3.1 Математическая библиотека

Математическая библиотека занимает главенствующую роль в проекте, так как именно она содержит набор алгоритмов, позволяющих получить генеральный план площадного объекта в автоматическом режиме.

Прежде чем изобразить архитектуру математической библиотеки, сформулируем основные концепции, которые будут её описывать. Для иссле-

довательских проектов именно общие принципы реализации приобретают главенствующую роль, так как в силу очень высокой динамичности изменений требований к результату работы и несистемности этих требований отсутствует какая-либо возможность спроектировать архитектуру заранее.

Математическая методика является классом алгоритмов, которые могут решать ту или иную задачу. Каждая математическая методика используется хотя бы в одном *математическом методе*.

- *Аккуратное использование внешних зависимостей.* Исходный код математической библиотеки является главным результатом работы команды исследователей, поэтому, по-возможности, он должен содержать наименьшее количество внешних зависимостей. Допускается использование математических и научных библиотек языка Python.
- *Каждая математическая методика должна быть выделена в отдельный программный модуль.* Так как каждая методика олицетворяет собой определенный класс математических алгоритмов, а сами алгоритмы являются концептуально сложными, то для упрощения их поддержки и модернизации следует ввести их явное логическое разделение.
- *Минимализм во входных и выходных данных каждой методики.* В рамках каждой методики должны использоваться только лишь те данные, которые необходимы для её работы. Использование лишних данных следует избегать. Данная концепция обусловлена тем, что в силу высокой сложности алгоритмов, очень важно уменьшать накладные расходы на их отладку, что может сильно осложняться из-за большого количества входных параметров.
- *Все параметры каждой методики должны быть выделены в отдельный класс.* Внутри реализации каждой математической методики недопустимо появление различных "магических чисел". Применение этой концепции обусловлено необходимостью повторяемости научного эксперимента. Если спрятать параметры алгоритма внутрь исходного кода, то результат на одних и тех же исходных данных может

неудастся повторить в силу потерянного значения одного из чисел.

- *Входные/выходные данные и параметры каждой методики должны быть сериализуемыми.* Так как расчёты могут производиться, как на сервере, так и на компьютере исследователя, то необходимо предусмотреть механизм простой передачи данных с компьютера на сервер и обратно. Самым простым способом является сериализация входных и выходных данных.

Ниже представлена верхнеуровневая диаграмма классов математического модуля(см. рисунок 15).

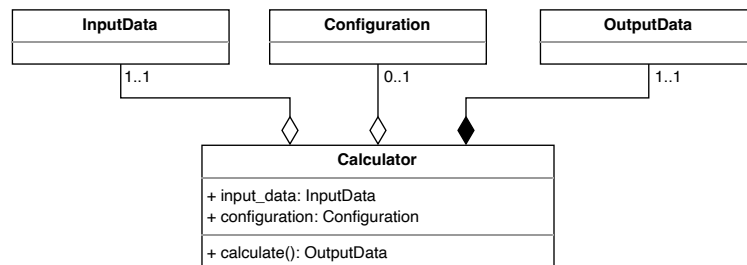


Рис. 15. Верхнеуровневая диаграмма классов математического модуля

Каждая алгоритмическая методика запускается путём вызова метода *calculate* в классе *Calculator*. Для каждой методики может быть реализовано несколько интерфейсов *Calculator*, которые будут использовать один класс алгоритмов, но давать несколько иной результат.

Каждая реализация *Calculator* принимает входные данные *InputData* и возвращает выходные данные *OutputData*. Если методика подразумевает использование различных параметров настройки, то *Calculator* дополнительно принимает на вход *Configuration*. Если параметры настройки не требуются, то класс *Configuration* является *NULL*.

2.3.2 Расчётная модель данных

Так как генплан состоит из большого количества разнородных объектов, то необходимо предусмотреть легко расширяемую и понятную модель данных, которая будет удобна в использовании исследователям и аналитикам. Для генплана используются следующие объекты:

- допустимая для строительства область на карте;

- стоимостная модель расчета стоимости инженерной подготовки;
- перечень сооружений с указанием габаритов (ширина, длина, радиус), степени огнестойкости, категории взрывопожарной и пожарной опасности, конструктивной пожарной опасности;
- параметры коммуникаций между сооружениями проектируемого объекта;
- зоны распространения теплового потока;
- зоны распространения взрывной волны;
- и т.д.

Для решения поставленной задачи предлагается следующая расчётная модель данных, изображенная на верхнеуровневой диаграмме классов. Основная цель создания этой модели данных – это снижение издержек на обмен данными между сервисами расчётного модуля. В силу сложности предметной области и большого количества технической терминологии, критерий понятности чрезвычайно важен.

Ниже представлена верхнеуровневая диаграмма классов расчётной модели данных (см. рисунок 16).

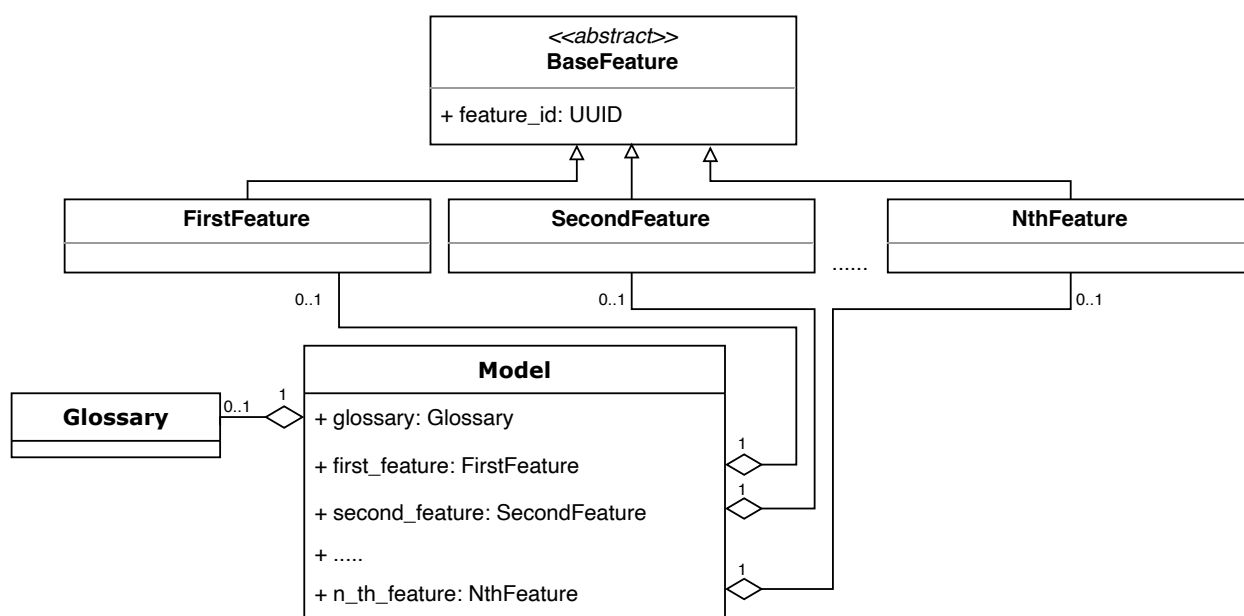


Рис. 16. Верхнеуровневая диаграмма классов расчётной модели данных

Расчётная модель данных представлена следующими классами.

- *Model* – верхнеуровневый класс, который и является расчётной моделью данных, использующийся при передаче данных между сервисами. Состоит из расчётных элементов и глобального регистра типов объектов.
- *Feature* – расчётный элемент. Являются составными частями расчётной модели данных. Например, расчётными элементами являются местоположения сооружений, внутриплощадочные проезды и т.д.
- *Glossary* – объект, являющийся глобальным перечнем типов различных объектов. Например, типы сооружений, такие как "Сепаратор "Трансформаторная подстанция "Насосная" хранятся именно здесь. Помимо описания типа, именно здесь хранятся его различные характеристики. Для типа "Электрокабель 35 кВ" определены параметра диаметра сечения, удельного сопротивления и используемого материала.

3 РАЗРАБОТКА СРЕДСТВА АВТОМАТИЗАЦИИ

3.1 Разработка расчётной модели данных

Расчётная модель данных состоит из расчётных элементов. Ниже представлена общая диаграмма классов расчётной модели данных (см. рисунок 17).

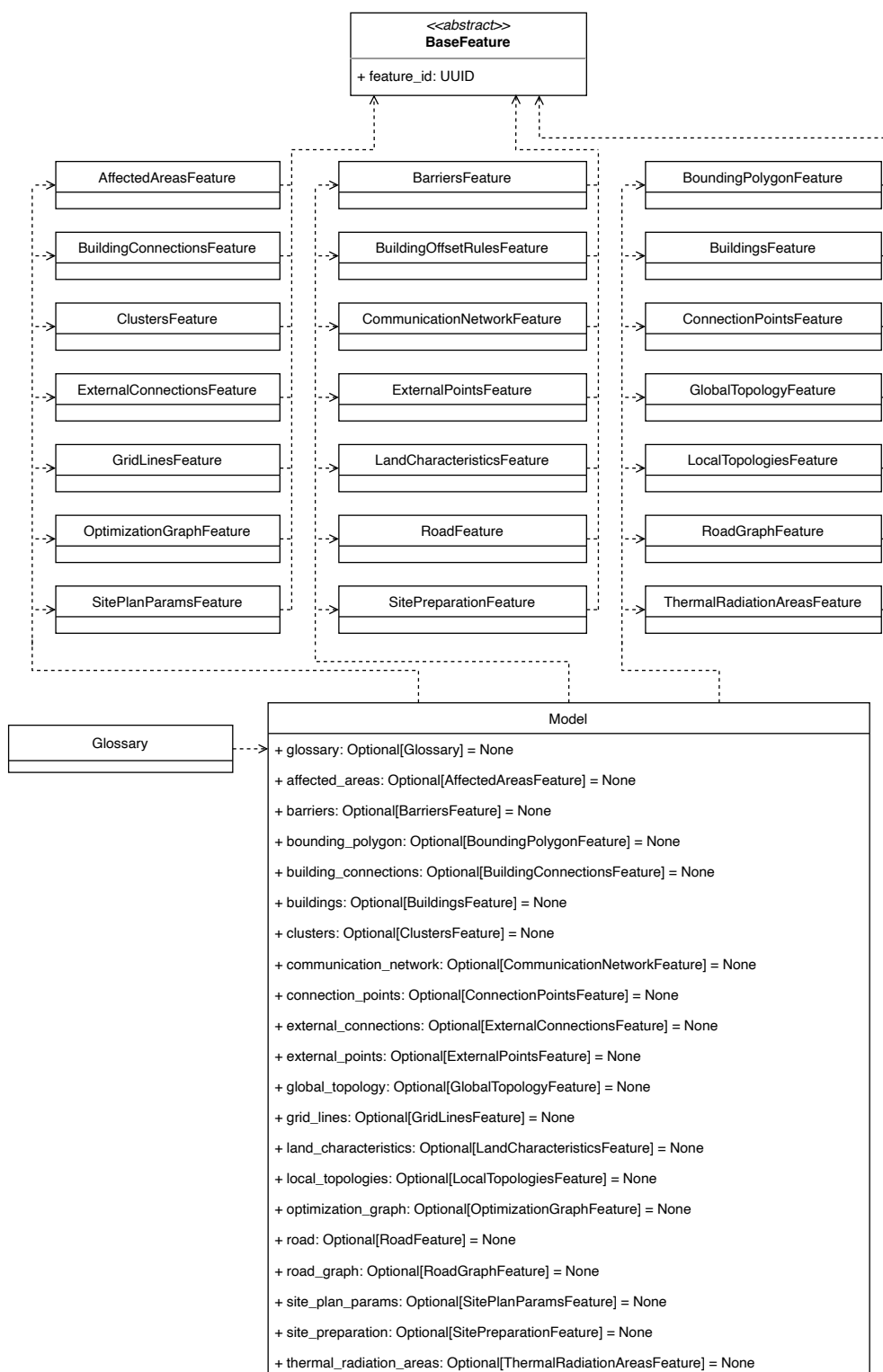


Рис. 17. Общая диаграмма классов расчётной модели

А вот диаграмма классов для расчётного элемента, описывающее сооружение (см. рисунок .18).

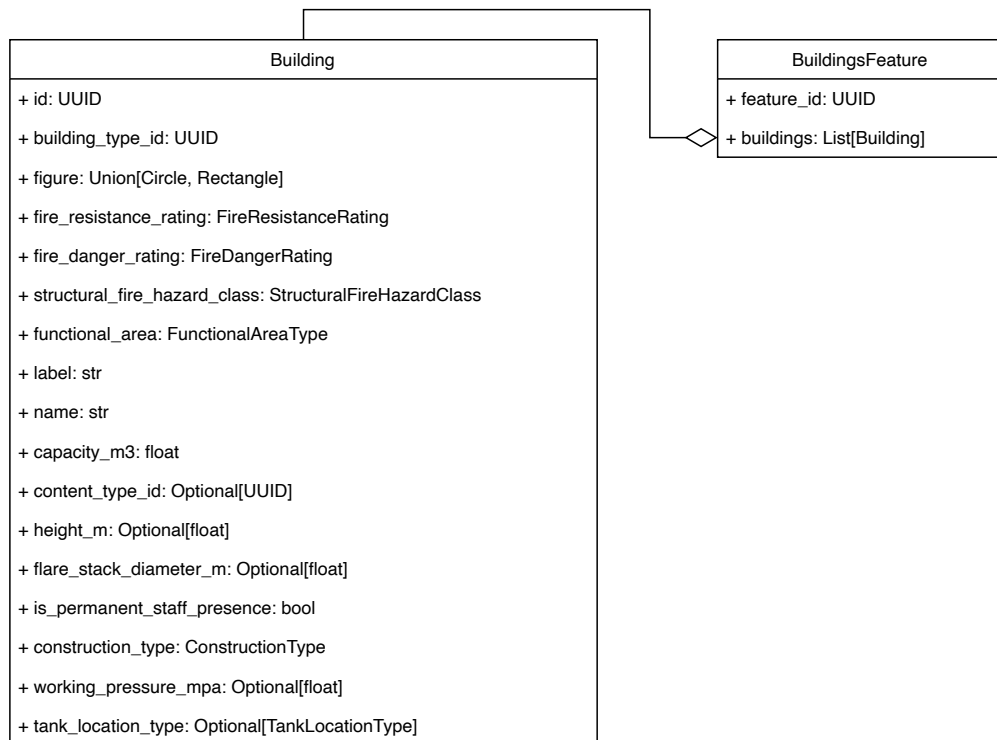


Рис. 18. Общая диаграмма классов расчётной модели

3.2 Разработка математической библиотеки

В главе с проектированием системы приведена абстрактная архитектура математической библиотеки и описание общих принципов реализации математических методов. Ниже приведем конкретное воплощение этих общих принципов. Написание кода, его группировка в методы и объединение методов в классы осуществляется с учетом лучших практик, указанных в [15].

Математическая библиотека имеет следующую структуру (см. рисунок .19).

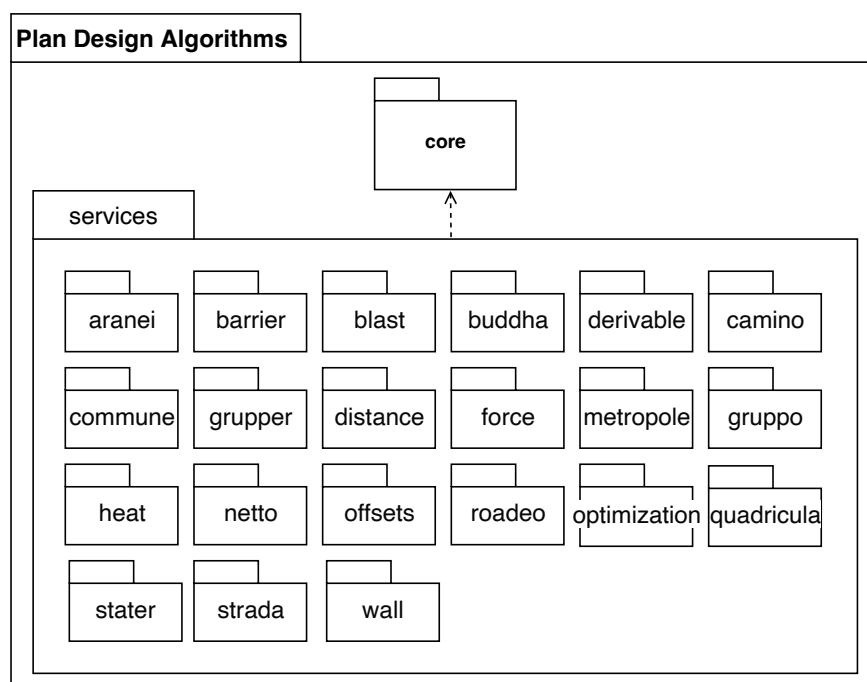


Рис. 19. Диаграмма пакетов математической библиотеки

Пакет *core* содержит методы и классы, которые могут быть свободно переиспользованы. К таковым, например, относится вычислений расстояний между сооружениями. Каждый пакет расположенный в пакете *services* это реализация отдельной математической методики.

Приведем пример сервиса, реализующий методику по расчёту расстояний теплового излучения от факелов. Данный сервис называется *heat*. Диаграмма классов данного сервиса изображена ниже (см. рисунок 20).

Так как мы знаем, что расчет теплового излучения может быть произведен только для факелов, то все сооружения, используемые в этом сервисе, мы можем сразу же обозначить, как факелы. Название класса *Flare* сразу позволит понять, с какими объектами оперирует методика, не заглядывая в документацию. Класс *Building*, описанный выше в расчётной модели данных содержит 17 полей, а класс *Flare*, описанного для методики расчёта зон теплового излучения содержит всего 4 поля. В модели данных, используемой в методике следует указывать только те поля, которые необходимы для работы данной методики. Это позволяет упростить взаимодействие с объектами, которые описаны в общей модели данных.

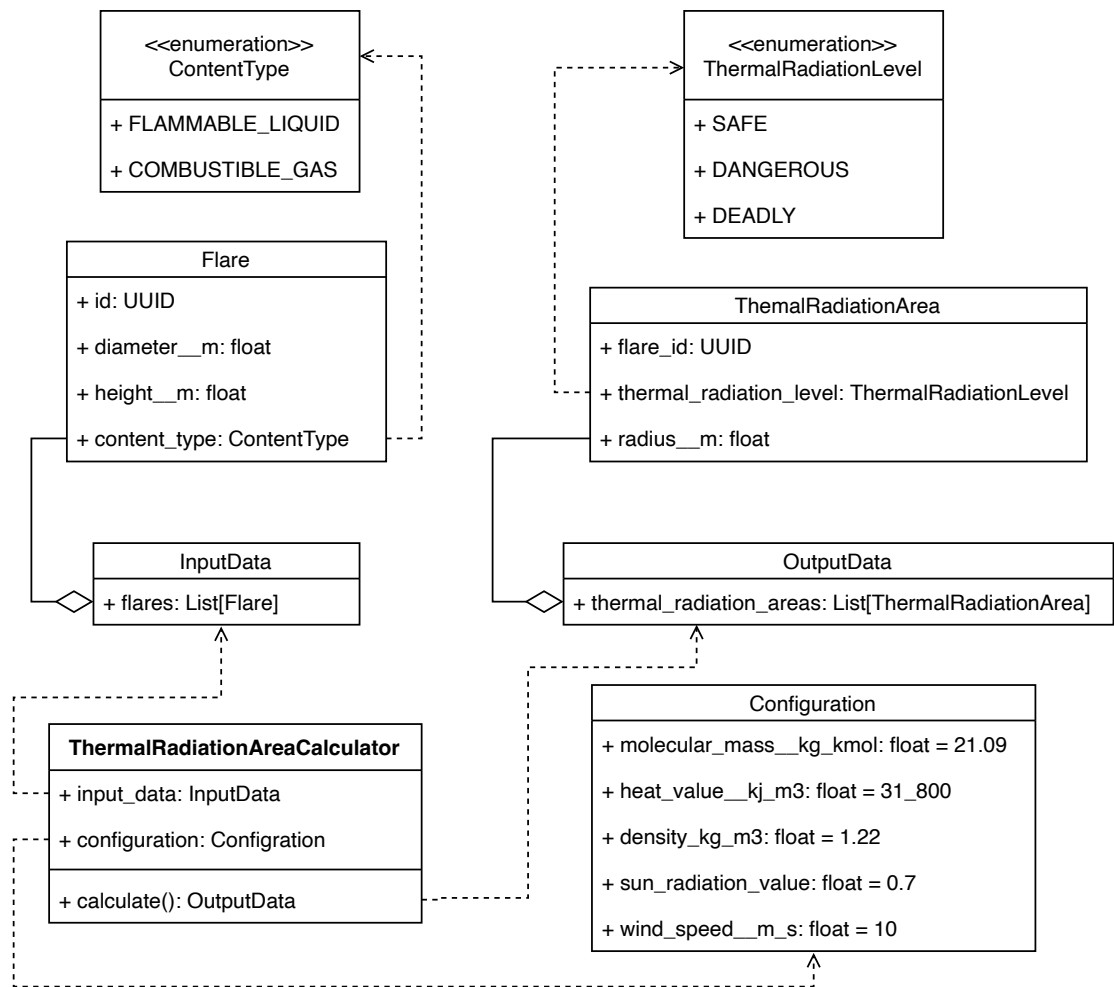


Рис. 20. Диаграмма классов модели данных сервиса *heat*

3.3 Разработка сервиса по запуску математических методов

Одним из современных стандартов разработки веб-приложений на языке Python является использование библиотеки *asyncio*. Обычно при использовании данной библиотеки для запуска задач в отдельном процессе используется экземпляр класса *concurrent.futures.ProcessPoolExecutor*.

```

result = await loop.run_in_executor(
    process_pool_executor,
    execute_task,
    execution_config,
    task.task_id
)

```

Листинг 1. Пример запуска с использованием *ProcessPoolExecutor*

Но так как время выполнения математических методов при расчёте

генерального плана является значительным, то необходимо предусмотреть механизм, который позволит завершать расчёт досрочно при необходимости.

Для завершения процессов операционная система использует механизм сигналов. Но если с помощью сигнала завершить процесс, созданный экземпляром класс *ProcessPoolExecutor*, то экземпляр больше не сможет создавать новые процесс и для продолжения работы будет требоваться создание экземпляра класса заново.

Таким образом данный способ является неприменимым в нашем случае. Чтобы корректно работал механизм отмены предлагается запуск задач с помощью прямого создания экземпляров класса *multiprocessing.Process*.

```
async def handle_long_cpu_bound_task(
    execution_config: ExecutionConfig,
    task_id: int
) -> TaskExecutionProcess:
    args = (
        execution_config,
        task_id,
    )
    process = Process(
        target=execute_long_task,
        args=args
    )

    task_execution_process = TaskExecutionProcess(
        task_id,
        process,
    )

    process.start()

    return task_execution_process
```

Листинг 2. Пример запуска с использованием *multiprocessing.Process*

3.4 Развертывание приложения

Помимо программной реализации необходимо уделить внимание вопросам развертывания приложения. Gitlab CI/CD является инструментом, используемым в компании для непрерывной интеграции и развертывания. Он предоставляет большой набор возможностей для развертывания приложения. Код системы хранится в монорепозитории. Это сделано с целью повышения прозрачности развертывания.

Главным объектом в системе Gitlab CI/CD является Pipeline. Pipeline состоит из Stage, где каждый Stage состоит из Job. Pipeline – это верхне-уровневый элемент набора операций, который будет применяться к актуальной кодовой базе.

Правила развертывания в Gitlab CI/CD представляют собой Directed Acyclic Graph(DAG). Stage выполняются строго последовательно, но между Job-ами можно устанавливать зависимости.

Пример зависимостей в Pipeline можно увидеть на рисунке ниже(см. рисунок 21).

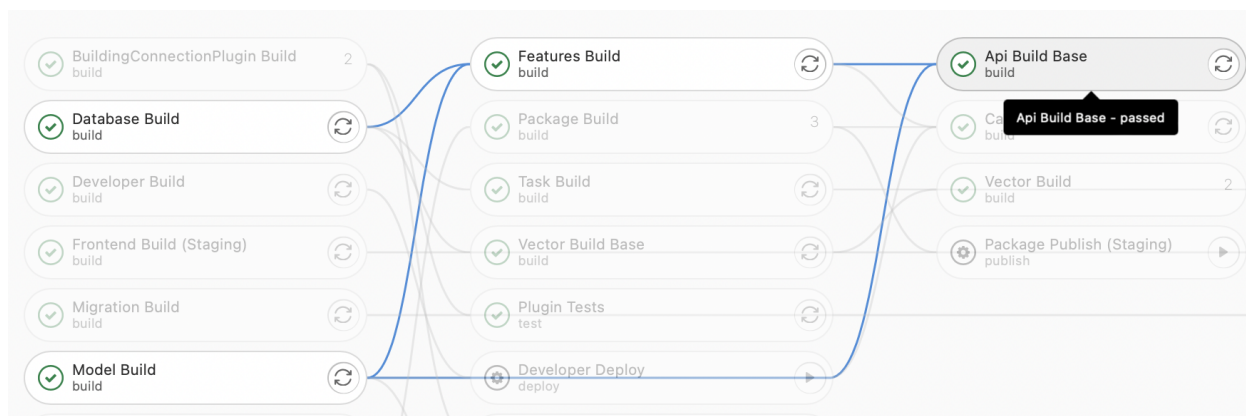


Рис. 21. Зависимости в Pipeline в Gitlab CI/CD

Динамические окружения

Gitlab CI/CD имеет возможность создания, как статических окружений, так и динамических. Статические окружения представляют собой окружения, с которыми взаимодействуют пользователи. В нашем случае это *staging* и *production*. То динамические окружения – это отличное решение для изолированного тестирования нового функционала отдельно от окружений, с которыми взаимодействуют пользователи. Именно они позволяют

развернуть изолированный прототип системы путем нажатия одной кнопки в интерфейсе. Пример окружений можно увидеть на рисунке ниже (см. рисунок 22).

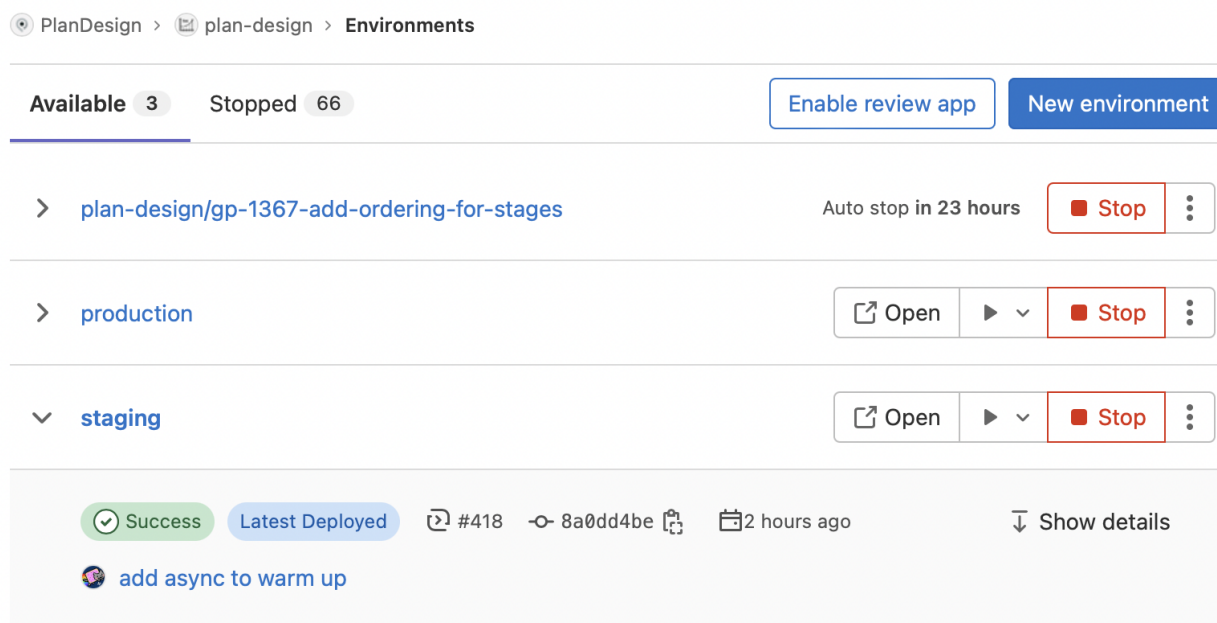


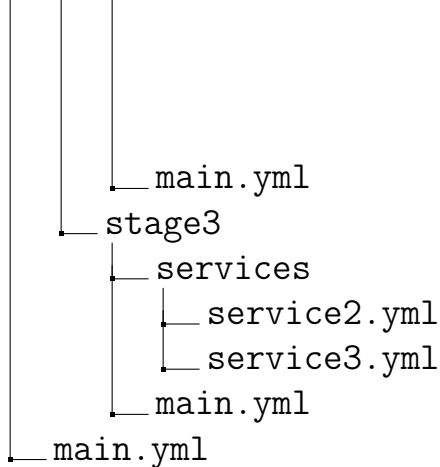
Рис. 22. Пример динамических окружений

Структура файлов CI/CD

Все правила развертывания и сборки описываются в файле *.gitlab-ci.yml*, его название и местоположение может быть изменено на любое другое в настройках репозитория.

Сложную конфигурацию CI/CD можно разбить на несколько разных логических компонент, поместив каждую из них в отдельный файл. С целью упрощения поддержки и добавления новых правил развертывания предлагается следующая структура файлов.

```
gitlab-ci
├── stage1
│   ├── services
│   │   ├── service1.yml
│   │   ├── service2.yml
│   │   └── service3.yml
│   └── main.yml
└── stage2
    ├── services
    │   ├── service1.yml
    │   ├── service3.yml
    │   └── service7.yml
```

Данная структура является иерархической. Корневая директория называется *gitlab-ci*. В корневой директории находится файл *gitlab-ci/main.yml* (см. листинг3), внутри которого содержатся на верхнеуровневые файлы для каждого *Stage*. В директории каждого *Stage* находится файл *main.yml* (см. листинг4) который содержит включения на файлы с описанием конфигурации для отдельного сервиса на этом этапе.

```
stages:
```

- stage1
- stage2
- stage3

```
include:
```

- local: stage1/main.yml
- local: stage2/main.yml
- local: stage3/main.yml

Листинг 3. Пример кода *gitlab-ci/main.yml*

```
stages:
```

- stage1

```
include:
```

- local: stage1/services/service1.yml
- local: stage1/services/service2.yml
- local: stage1/services/service3.yml

Листинг 4. Пример кода *gitlab-ci/stage1/main.yml*

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной выпускной квалификационной работы были выполнены все поставленные задачи.

Путём проведения интервью были собраны и проанализированы требования пользователей к системе. Была детально рассмотрена предметная область задачи с точки зрения структур данных и выполнения математических задач. Были сформированы функциональные и нефункциональные требования к программному компоненту, целью которого является упрощение процесса проведения научных изысканий в области автоматического формирования генеральных планов площадных объектов.

На основе сформированных требований была спроектирована системная и программная архитектура расчётного модуля системы автоматического проектирования генеральных планов площадных объектов капитального строительства.

Был реализован расчётный модуль, состоящий из пяти программных компонент: математической библиотеки, расчётной модели данных, сервиса запуска расчётных задач, сервиса хранения расчётных данных, сервиса запуска математических методов. Разработанный расчётный модуль удовлетворяет всем сформированным функциональным и нефункциональным требованиям.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. «Градостроительный кодекс Российской Федерации» от 29.12.2004 N 190-ФЗ (ред. от 01.05.2022)
2. Федеральный закон «Технический регламент о безопасности зданий и сооружений» от 30.12.2009 N 384-ФЗ.
3. «Гражданский кодекс Российской Федерации (часть первая)» от 30.11.1994 N 51-ФЗ (ред. от 06.04.2011), Статья 233
4. «СВОД ПРАВИЛ СП 4.13130.2013 СИСТЕМЫ ПРОТИВОПОЖАРНОЙ ЗАЩИТЫ ОГРАНИЧЕНИЕ РАСПРОСТРАНЕНИЯ ПОЖАРА НА ОБЪЕКТАХ ЗАЩИТЫ ТРЕБОВАНИЯ К ОБЪЕМНО-ПЛАНИРОВОЧНЫМ И КОНСТРУКТИВНЫМ РЕШЕНИЯМ» от 24.06.2013
5. Python 3.8 Documentation [Интернет ресурс]:
URL:<https://docs.python.org/3.8/> (дата обращения: 23.05.22)
6. Python Advantages and Disadvantages [Интернет ресурс]:
URL:<https://techvidvan.com/tutorials/python-advantages-and-disadvantages/>
(дата обращения: 23.05.22)
7. PostGIS Reference [Интернет ресурс]:
URL:<https://postgis.net/documentation/> (дата обращения: 23.05.22)
8. Glenford J. Myers, Corey Sandler, Tom Badgett "The Art of Software Testing 3rd Edition, 16 Dec 2011
9. Docker Reference [Интернет ресурс]:
URL:<https://docs.docker.com/> (дата обращения: 23.05.22)
10. Sphinx Python Documentation [Интернет ресурс]:
URL:<https://www.sphinx-doc.org/en/master/> (дата обращения: 23.05.22)

11. Protobuf Documentation [Интернет ресурс]:
<https://developers.google.com/protocol-buffers/docs/overview>
(дата обращения: 23.05.22)
12. Gitlab CI/CD Documentation [Интернет ресурс]:
<https://docs.gitlab.com/ee/ci/> (дата обращения: 23.05.22)
13. Microservices Architecture [Интернет ресурс]:
<https://microservices.io/index.html> (дата обращения: 23.05.22)
14. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides "Design Patterns: Elements of Reusable Object-Oriented Software 1st Edition
15. Martin Robert C. "Clean Code: A Handbook of Agile Software Craftsmanship" Aug 1, 2008

ПРИЛОЖЕНИЯ

Приложение А. API сервисов

Executor Service API 0.1.0 OAS3

[/openapi.json](#)

Task



POST	/tasks/create	Create Task	▼
POST	/tasks/run	Run Task	▼
POST	/tasks/cancel	Cancel Task	▼
GET	/tasks/{task_id}/status	Get Task Status	▼
GET	/tasks/{task_id}/output	Get Task Output Data	▼

Рис. 1. API сервиса запуска математических методов

Storage Service API

0.1.0 OAS3

/openapi.json

features

POST	/feature	Create Feature	✓
GET	/feature/{feature_id}	Read Feature	✓
PUT	/feature/{feature_id}	Update Feature	✓
DELETE	/feature/{feature_id}	Delete Feature	✓

glossary

GET	/glossary/{glossary_items_name}	Read Glossary Items	✓
PUT	/glossary/{glossary_items_name}	Update Glossary Item	✓
POST	/glossary/{glossary_items_name}	Create Glossary Item	✓
DELETE	/glossary/{glossary_items_name}/{glossary_item_id}	Delete Glossary Item	✓

model

POST	/model/create	Create Model	✓
POST	/model/read	Read Model	✓

Рис. 2. API сервиса хранения расчётных данных

Orchestrator Service API

0.1.0 OAS3

/openapi.json

Schemes

HTTPS

feature

task

stage

calculation

method

Рис. 3. API сервиса запуска расчётных задач

Приложение Б. Фрагменты пользовательского интерфейса

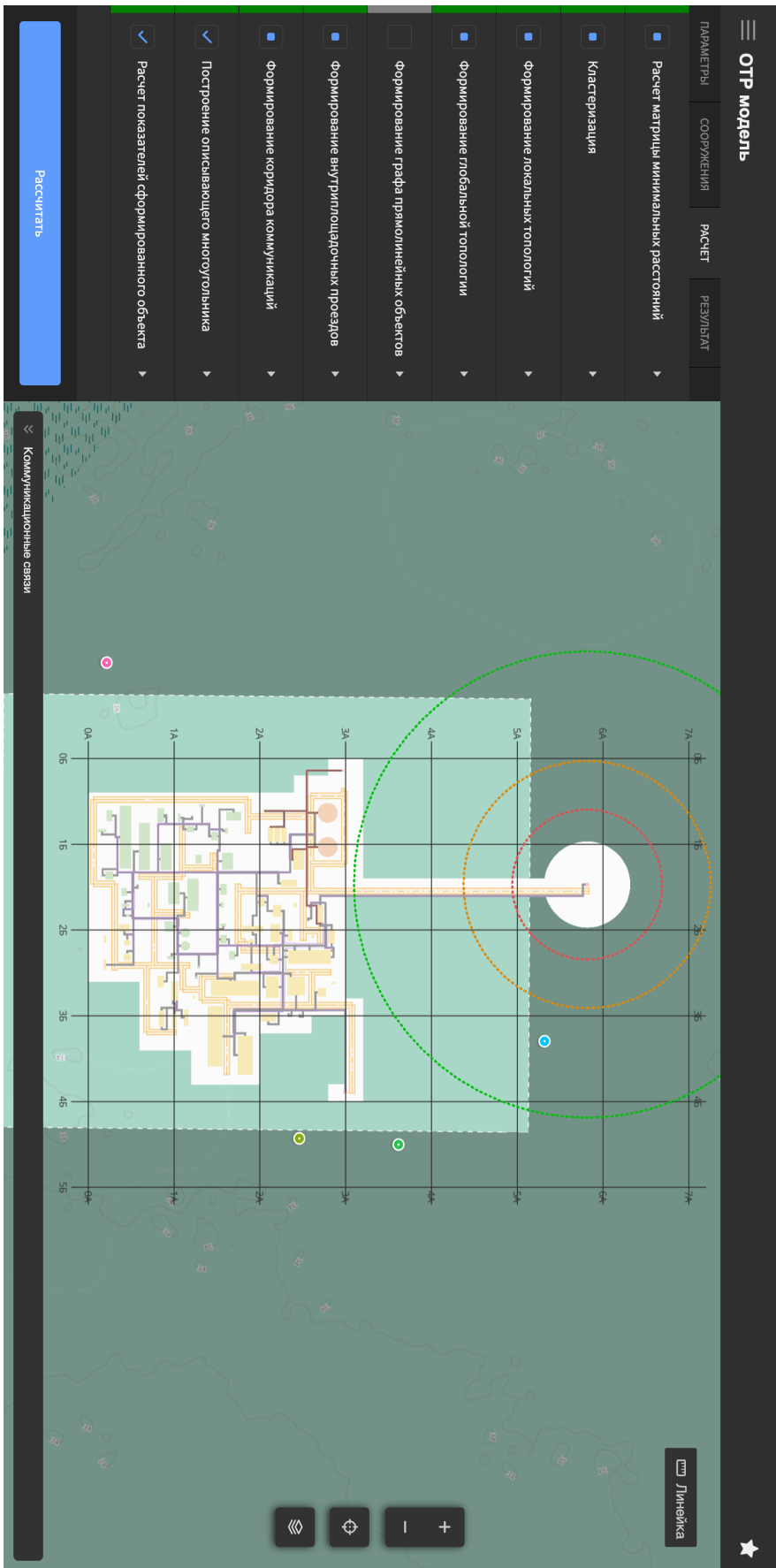


Рис. 1. Скриншоты интерфейса системы