Министерство науки высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
(Университет ИТМО)

Факультет информационных технологий и программирования (ФИТиП)

Образовательная программа Программирование и интернет-технологии

Направление подготовки (специальность) 09.04.02 Информационные системы и технологии

ОТЧЕТ
О ПРЕДДИПЛОМНОЙ ПРАКТИКЕ

по теме:

РЕАЛИЗАЦИЯ СЕРВИСА ДЛЯ ЗАПУСКА ЗАДАЧ ДЛЯ РАСЧЁТА ГЕНЕРАЛЬНЫХ ПЛАНОВ ПЛОЩАДНЫХ ОБЪЕКТОВ

Обучающийся
Академическая группа М42051 — Степанов С. В.

Руководитель практики от профильной организации,
ООО «Цифровое проектирование»,
руководитель научной лаборатории — Ашихмин И. А.

Руководитель практики от университета,
Университет ИТМО,
факультет информационных технологий и программирования,
доцент — Зубок Д. А.

Практика пройдена с оценкой _____

Дата 28.05.2022

Санкт-Петербург
2022

РЕФЕРАТ

Данный отчет состоит из 14 страниц. Содержит в себе 6 иллюстраций, 3 листинга кода, 7 использованных источников. В данном отчете описывается проектирование и реализация сервиса, обеспечивающего выполнения расчётных задач при построении генерального плана площадного объекта. В рамках данной работы были проанализированы функциональные и нефункциональные требования, спроектирована архитектура сервиса и реализована запроектированная архитектура.

СОДЕРЖАНИЕ

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.

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).

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.

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.

Строения — общее понятие зданий и сооружений.

ВВЕДЕНИЕ

Автоматическое формирование генерального плана (ГП) площадного объекта капитального строительства является чрезвычайно сложной задачей, как в алгоритмическом, так и в технологическом плане. На генеральном плане помимо сооружений отражены внутриплощадочные проезды, различные трубопроводы, линии электропередач, технологические эстакады, пожарные гидранты и прочие объекты, необходимые для функционирования того или иного площадного объекта.

Для каждого объекта ГП заданы определенные требования к его размещению. Так как объектов много и они очень разнообразны по своей структуре, то использовать единый алгоритм для их расстановки невозможно.

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.

ПОСТАНОВКА ЗАДАЧИ

Основной целью данной работы является спроектировать и реализовать сервис, который обеспечит выполнение расчётных задач при построении генерального плана площадного объекта.

Исходя из поставленной цели можно выделить следующие *задачи*:

1. Проанализировать функциональные и нефункциональные требования к сервису.
2. Опираясь на требования спроектировать архитектуру сервиса.
3. Реализовать спроектированную архитектуру.

ТРЕБОВАНИЯ

Ниже перечислены те функциональные и нефункциональные требования для сервиса запуска расчётных задач.

Функциональные требования

В рамках функциональных требований к сервису можно выделить следующее:

1. Должна присутствовать возможность расчёта генерального плана площадного объекта в автоматическом режиме.
2. Автоматический расчёт генерального плана площадного объекта должен быть разбит на этапы.
3. Результат каждого этапа расчёта должен быть сохранён в долговременное хранилище.
4. Должна присутствовать возможность продолжить расчёт с последнего успешно завершенного этапа.
5. API должен придерживаться концепции REST.

6. API должен предусматривать асинхронный запуск расчётной задачи.

7. API должен использовать JSON в качестве обмена данных с клиентом.

Нефункциональные требования

Из нефункциональных требований выделим:

1. Язык программирования **Python 3.8**

2. Развертывание сервиса осуществлять с помощью **Docker**

АРХИТЕКТУРА СЕРВИСА

На диаграмме размещения системы(см. рис 1) расчетный сервис выделен голубым цветом.

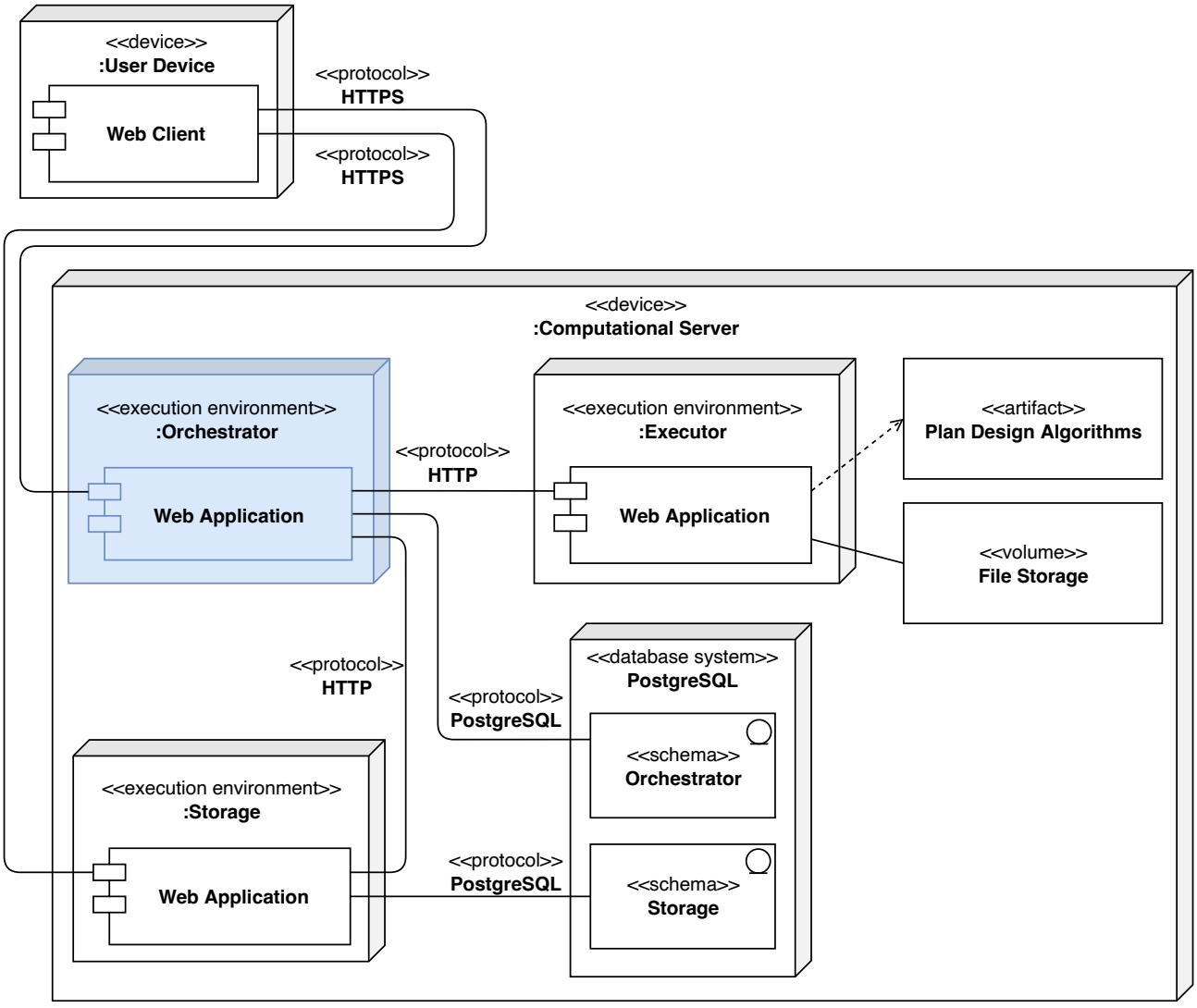

Рис. 1. Диаграмма компонентов запуска расчётных задач

Архитектура сервиса запуска расчётных задач представлена на диаграмме компонентов(см. рисунок 2).

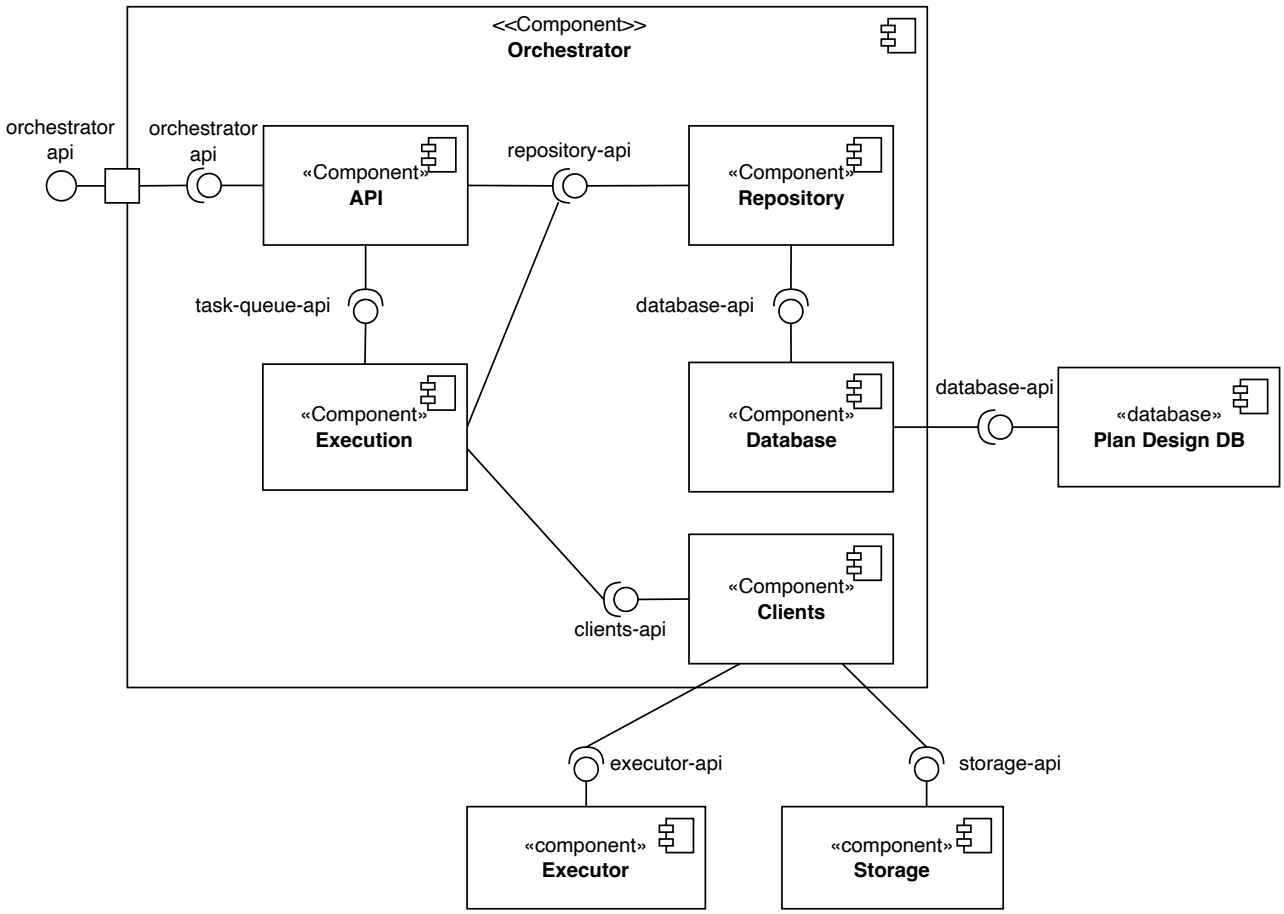

Рис. 2. Диаграмма компонентов сервиса запуска расчётных задач

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 .3)

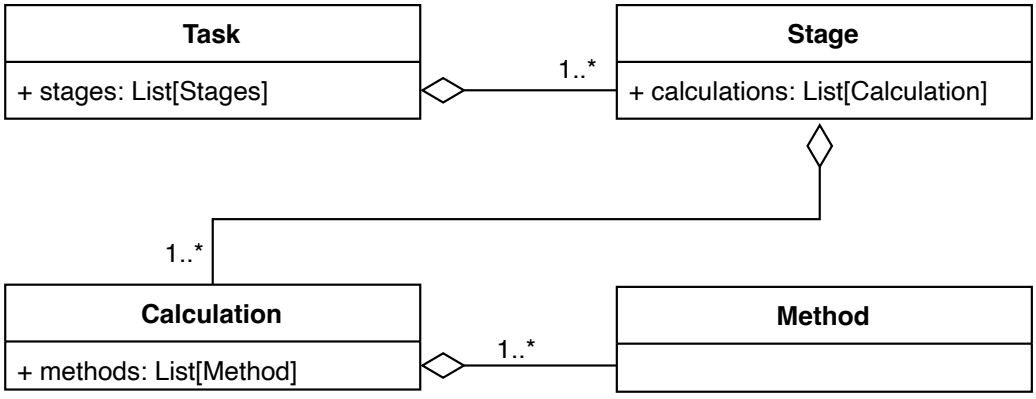

Рис. 3. Диаграмма классов основных сущностей сервиса запуска расчётных задач

- *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 4).

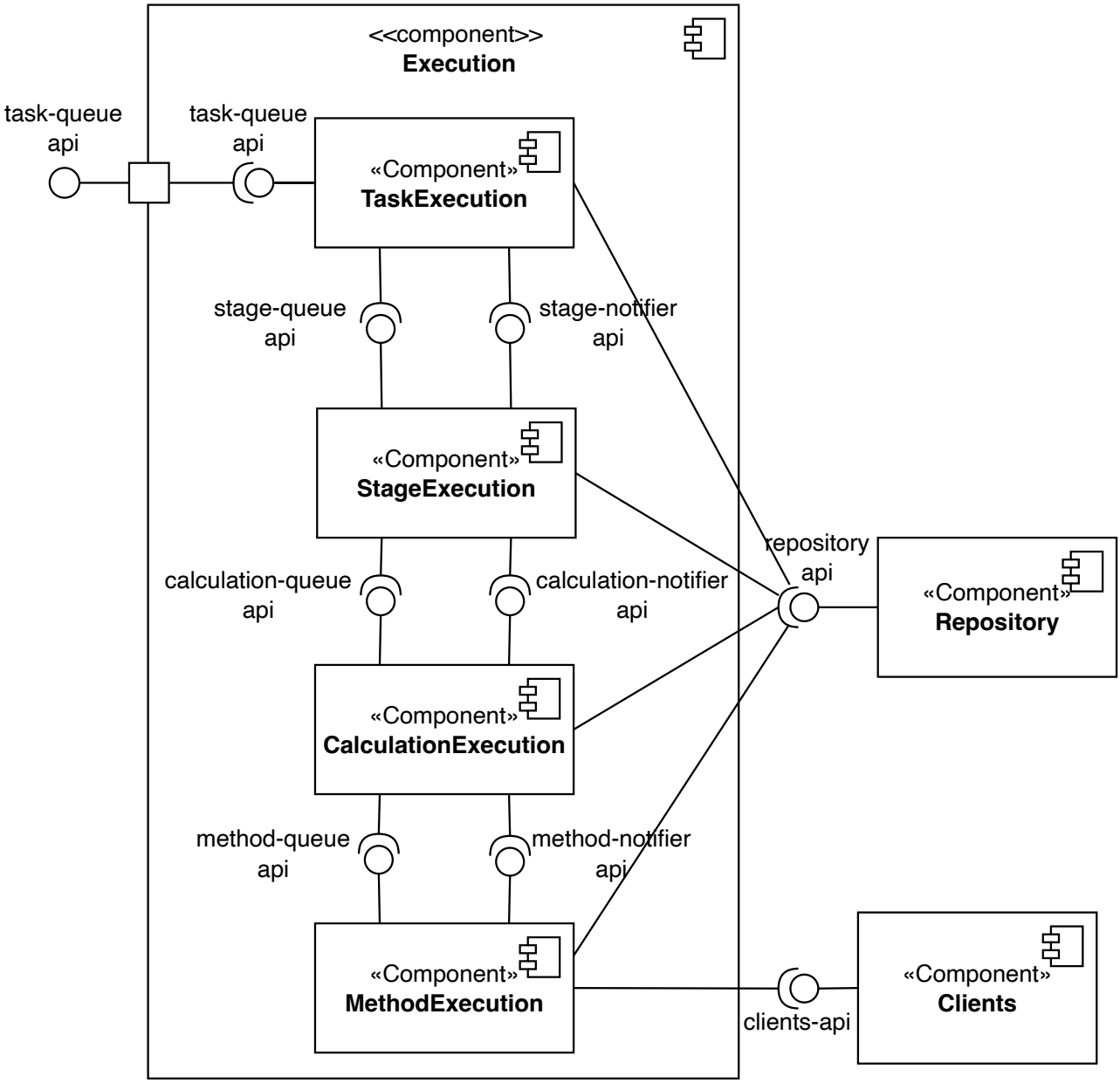

Рис. 4. Диаграмма компонентов запуска расчётных задач

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

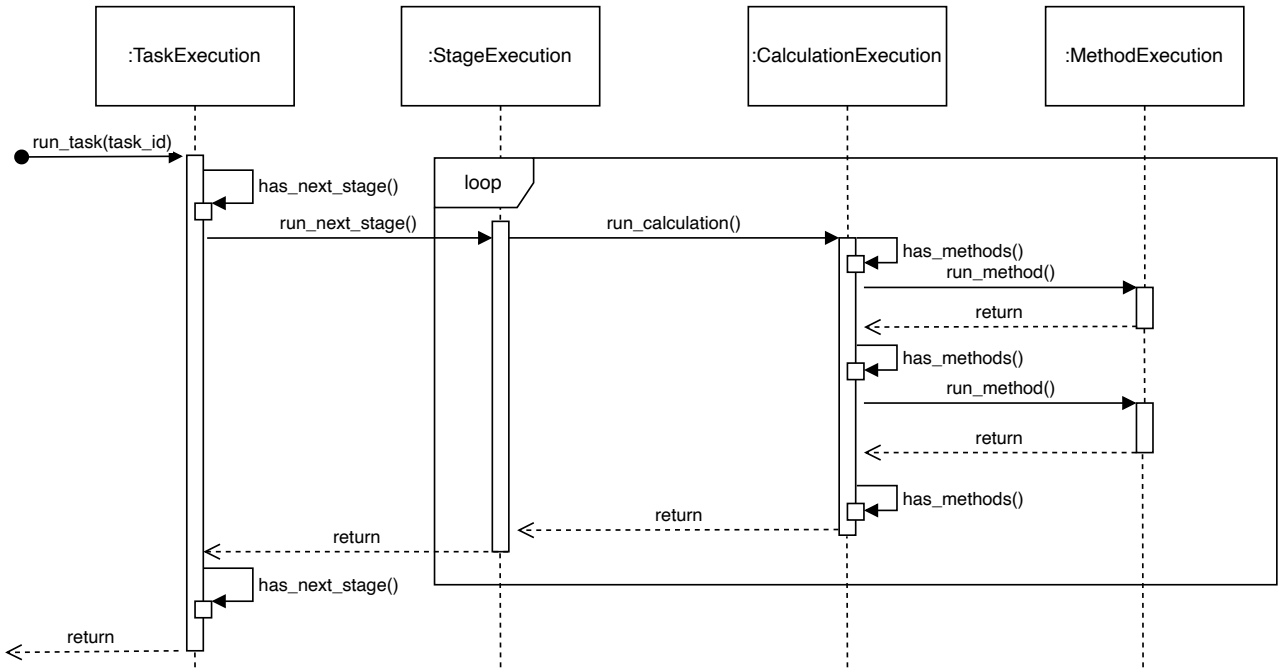

Рис. 5. Диаграмма последовательности выполнения расчётной задачи

Выше представлена диаграмма последовательности выполнения расчётной задачи(см. рисунок 5). 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*.

РЕАЛИЗАЦИЯ

При реализации описанной архитектуры была получена следующая структура проекта(см. рисунок 6).

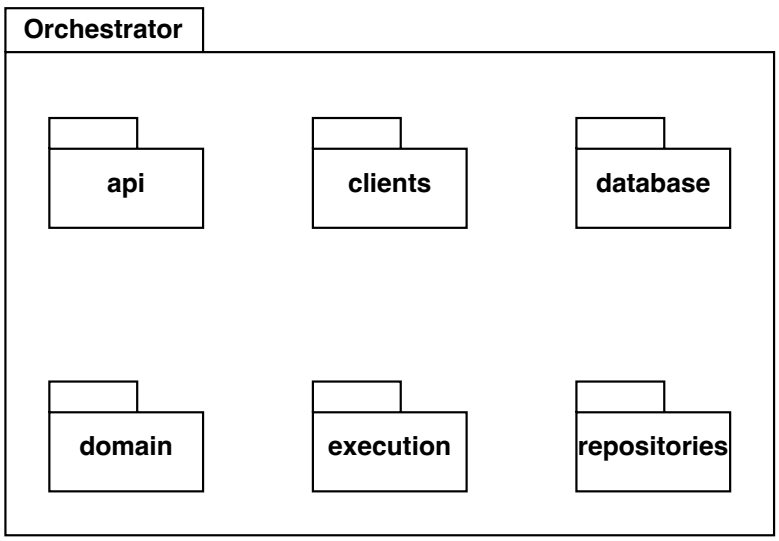

Рис. 6. Диаграмма пакетов сервиса запуска расчётных задач

Всего получилось 6 python-пакетов, в которых и скрыта основная логика работы.

1. *api* – реализует API сервисы через HTTP. То есть endpoint-ы, модели запросов/ответов к серверу, а также взаимодействие с *ITaskRepository* и *ITaskDataManager*
2. *database* – обеспечивает взаимодействие с базой данных.
3. *clients* – пакеты, в котором находятся клиенты для взаимодействия с сервисами хранилища и сервиса запуска математических методов.
4. *domain* – пакет, в котором определены только интерфейсы взаимодействия между модулями программы. Непосредственная реализация обозначенных интерфейсов находится в других пакетах.
5. *execution* – пакет, в котором находится код, отвечающий за запуск расчётных задач.
6. *repositories* – пакет, который обеспечивает логику сохранения/получения сущностей проекта из БД.

Примеры кода

```python
from app import startup_app
import uvicorn

if __name__ == '__main__':
    app = startup_app()
    uvicorn.run(app, host="0.0.0.0", port=8080)
```

Листинг 1. main.py

```
FROM python:3.8@sha256:4c4e6735f46e7727965d1523015874ab08f71377b3536b8789ee5742fc737

WORKDIR /app

ENV LC_ALL C.UTF-8
ENV LANG C.UTF-8
ENV N_WORKERS 8

COPY requirements.txt .
RUN pip3 install --no-cache-dir -r requirements.txt

RUN pip3 check

COPY main.py .

COPY /app ./app

ENTRYPOINT /bin/bash -c "gunicorn run_app:app --workers=${N_WORKERS} --bind 0.0.0.0:
```

Листинг 2. Dockerfile

```python
class Status(Enum):
    CREATED = "CREATED"
    QUEUED = "QUEUED"
    RUNNING = "RUNNING"
    CANCELLED = "CANCELLED"
    SUCCESS = "SUCCESS"
    FAILURE = "FAILURE"


@dataclass
class Feature:
    id: UUID = UUID(int=0)
    feature_type_id: UUID = UUID(int=0)


@dataclass
class Method:
    id: UUID = UUID(int=0)
    method_type_id: UUID = UUID(int=0)
    status: Status = Status.CREATED


@dataclass
class Calculation:
    id: UUID = UUID(int=0)
    calculation_type_id: UUID = UUID(int=0)
    status: Status = Status.CREATED


@dataclass
class Stage:
    id: UUID = UUID(int=0)
    stage_type_id: UUID = UUID(int=0)
    status: Status = Status.CREATED


@dataclass
class Task:
    id: UUID = UUID(int=0)
    name: str = ""
    task_type_id: UUID = UUID(int=0)
    status: Status = Status.CREATED
```

Листинг 3. domain/model.py

ЗАКЛЮЧЕНИЕ

В данной работе требовалось спроектировать и реализовать сервис, обеспечивающий выполнение расчётных задач при построении генерального плана площадного объекта. С учетом функциональных и нефункциональных требований была спроектирована архитектура сервиса, а так же эта архитектура была реализована.

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. Python 3.8 Documentation [Интернет ресурс]: `URL:https://docs.python.org/3.8/`(дата обращения: 23.05.22)
5. Glenford J. Myers, Corey Sandler, Tom Badgett "The Art of Software Testing 3rd Edition, 16 Dec 2011
6. Мартин Р. Чистый код. Создание, анализ и рефакторинг. – СПб.: Питер, 2019. – 464 с.
7. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. – СПб.: Питер, 2021. – 352 с.