Разработка программного обеспечения для автоматизации работы с приложением Cytoscape

Цель

Необходимо создать многопользовательское микросервисное приложение GraBLEE, которое позволяет с помощью вызова функции из программного кода на языке Python получать файл Cytoscape сессии, с возможностью применения стилей и укладок для графа.

Актуальность приложения

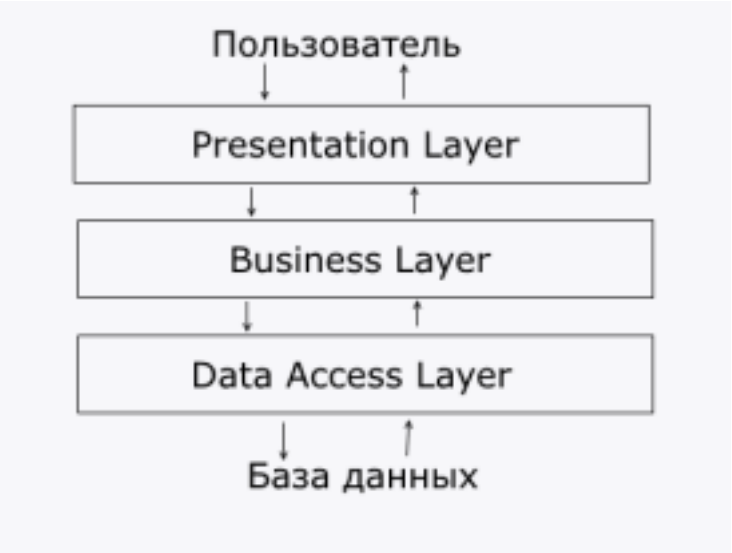
Команда алгоритмов на графах работает с огромными графами, содержащими сотни тысяч вершин, что вынуждает использовать инструмент визуализации Cytoscape, так как текстовое представление графа не дает необходимой информации. Это приложение обладает перегруженным неудобным и медленным интерфейсом, поэтому на однотипные задачи по получению визуализации вручную тратится много времени и сил. Так же существует проблема визуальной презентации решения задачи заказчикам, так как либо разработчикам, либо заказчикам придется конфигурировать наглядный, красивый вид cytoscape session файла. Cytoscape session файл – это файл, который содержит визуализацию уже в готовом, оформленным нужным образом виде. Именно его получение так нужно команде для повышения эффективности рабочего процесса путем делегирования приложению GraBLEE воспроизведения однотипных действий по открытию файлов с графами, применения к ним стилей, укладок, создания файлов cytoscape session.

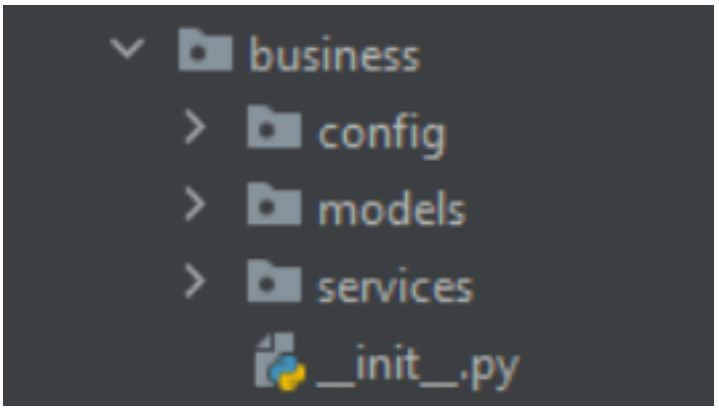
Используемые технологии

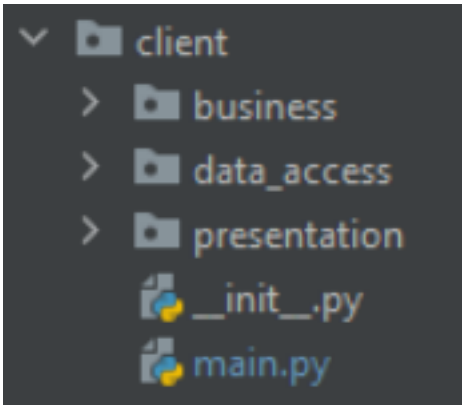
По условию поставленной задачи требовалось реализовать приложение на языке Python. Основные используемые библиотеки: socket, py4cytoscape, networkx, threading, colorlog, os. На удаленном сервере установлена ОС Ubuntu 22.04, его приходилось настраивать с использованием командной строки и shell. Серверная часть приложения взаимодействует с приложением Cytoscape, которое под управлением GraBLEE создает файл cytoscape session. Для написания кода и отладки приложения использовалась среда разработки Pycharm. Весь прогресс выполнения работы отмечался в Notion. В качестве системы контроля версий выбран Github. Инструкция по использованию GraBLEE написана на Markdown.

Архитектура

Для каждого микросервиса была выбрана трехслойная архитектура с анемичными моделями и сервисами для этих моделей. Приложение содержит микросервис-клиент и микросервис-сервер, которые общаются по TCP протоколу.

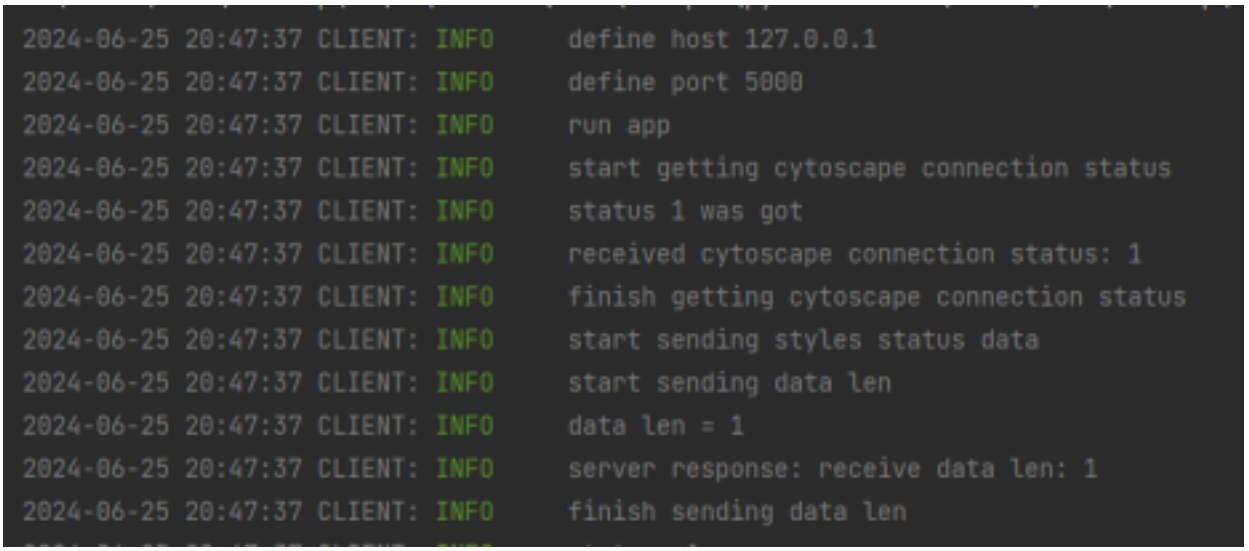




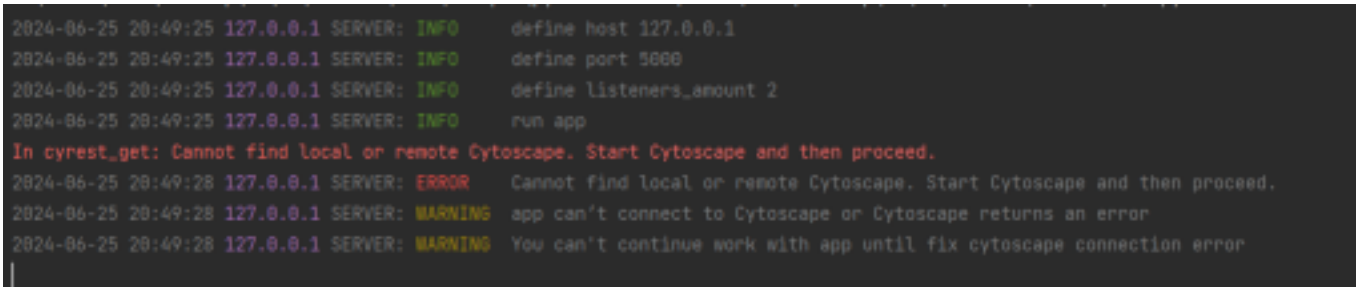


Микросервисы запускаются с помощью main.py файлов. В них создаются и конфигурируются все компоненты, включая socket и logger.

В приложении и клиент, и сервер предоставляют логи в красивом удобном формате, содержащие всю необходимую информацию, для устранения неполадок.



Пример логов у пользователя



Пример логов на сервере, когда GraBLEE server не может связаться с Cytoscape

Data access layer – содержит репозитории, которые взаимодействуют с файловой системой устройства, на котором запущено приложение, обеспечивая запись и чтение файлов в нужных форматах. Так же здесь происходит архивирование и распаковывание файлов. Это необходимо для обеспечения приемлемой скорости передачи данных по сети, так как работать приходится с тяжелыми файлами.

Business layer – содержит бизнес логику приложения. Здесь происходит взаимодействие с Cytoscape, NetworkX, Scocket, OGDF, вызовы репозиториев. Каждый сервис содержит множество методов для поэтапной обработки запроса. Каждый метод выполняет только одну конкретную задачу. Это позволяет сохранять код чистым, легко расширяемым и легко читаемым.

Presentation layer – производит поэтапную обработку после получения запроса микросервисом, путем вызовов методов сервисов из Business layer для обработки поступивших данных и доставки результатов обработки отправителю.

Так как приложением в один момент времени должны иметь возможность пользоваться несколько человек, было применено многопоточное программирование. Все операции по работе с Cytoscape были защищены мьютексами. В main.py для каждого нового подключения (запрос от нового пользователя) создается новый поток с запуском Handler.Handle() из Presentation layer. В итоге запросы от всех пользователей обрабатываются конкурентно, что создает иллюзию параллельности.

```
while True:
    try:
        conn, address = socket.accept()
        logger.info("accept connection from: " + str(address))
        conn.settimeout(timeout_seconds)

    except sct.error as e:
        logger.error(f'accepting connection: {e}')

    else:
        thread = threading.Thread(target=handle, args=(address[0], conn, lock))
        thread.start()
```

Получение запросов в main.py в GraBLEE server

Принцип работы

Как работает клиент:

Когда пользователь в процессе работы с NetworkX графом хочет получить визуализацию, он опционально предоставляет путь до файла со стилями и параметр укладки. Клиент оформлен, как расширение класса networkx.Graph, поэтому данные о графе он получает непосредственно из экземпляра класса.

После вызова функции получения сессии клиент применяет к графу укладку, сохраняет граф в виде файла и в архивированном виде отправляет файлы с графом и стилем на сервер. После этого пользователь просто ожидает, когда сервер отправит в ответ файл Cytoscape сессии.

Как работает сервер:

Серверная часть может быть запущена как на удаленном сервере, так и на локальном компьютере. Так же на устройстве, выполняющем роль сервера, должен быть запущен Cytoscape.

Сервер получает все необходимые данные и файлы, распаковывает их. К графу применяется укладка с помощью OGDF_LayoutService. Граф импортируется в Cytoscape. К нему применяется стиль, если он был задан. Все это экспортируется как файл Cytoscape сессии и отправляется на клиент пользователю.

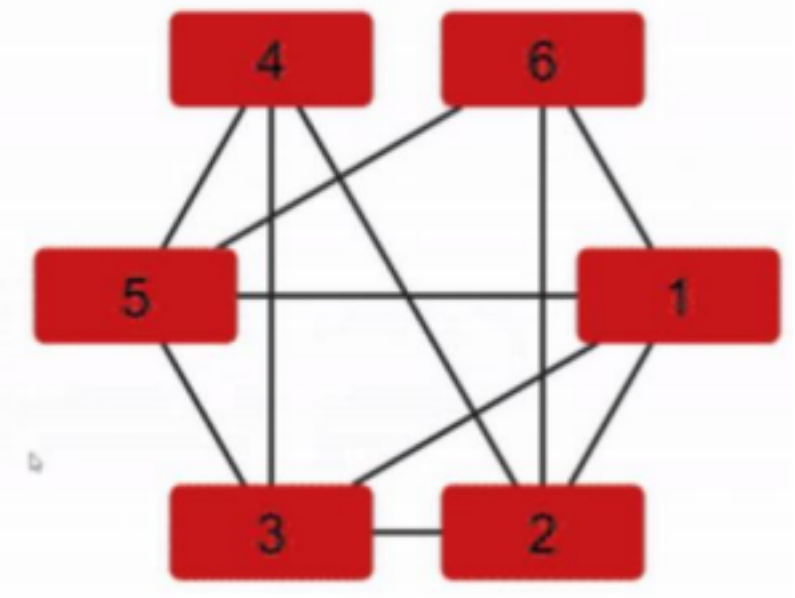
Принцип использования

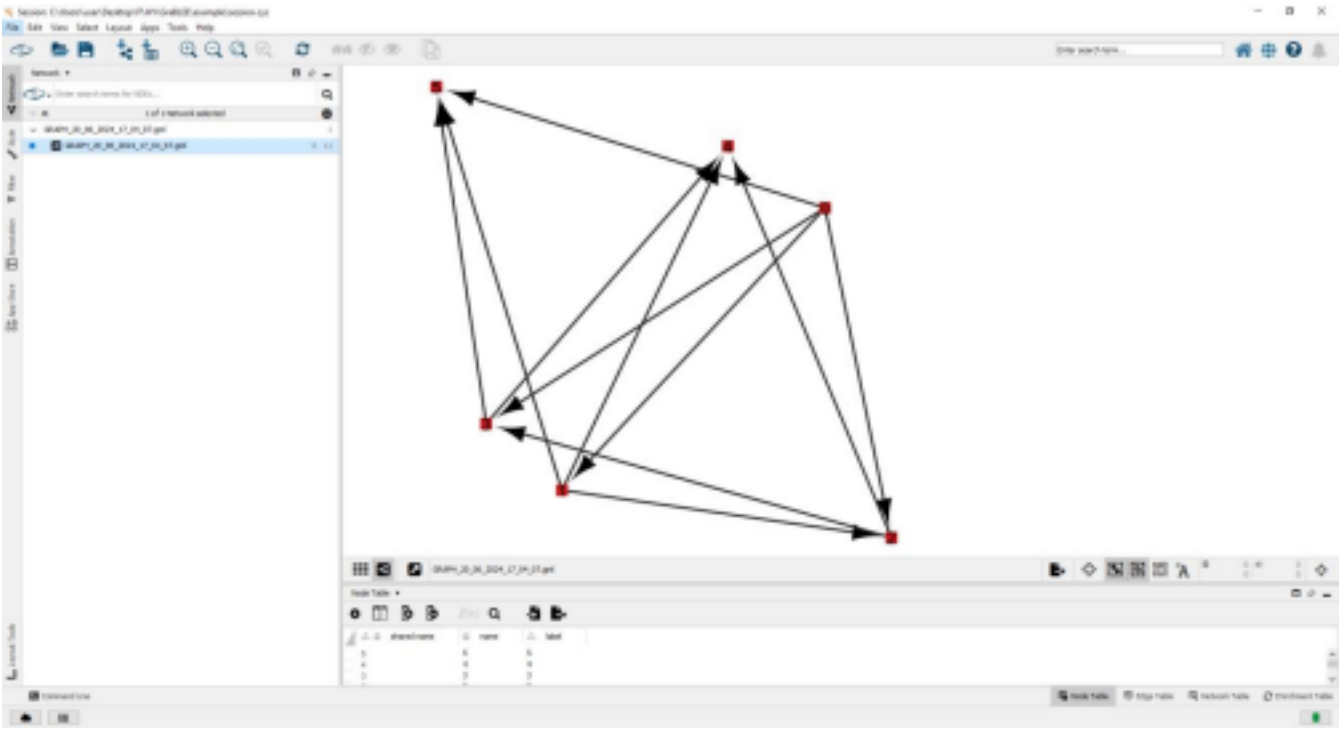
Если говорить про сервер, то на нем нужно запустить GraBLEE сервер и убедиться, что Cytoscape работает.

Для того, что использовать GraBLEE клиент у себя в коде, нужно сначала инициализировать его для netxorkx.Graph. Сделать это нужно всего один раз, затем в любой момент можно вызывать функцию получения cytoscape session.

```python
import networkx as nx
from GraBLEE.client import init_cytoscape_extension

init_cytoscape_extension()

graph = nx.Graph()
graph.add_edge(1, 2)
graph.add_edge(1, 3)
graph.add_edge(1, 5)

graph.to_cytoscape_session(cs_session_name='new_session',
                           layout_algorith='circular',
                           styles_filename='styles_red.xml')
```

Пример использования GraBLEE Client в пользовательском коде





Примеры cytoscape session, созданных с помщью GraBLEE

Результаты

В результате выполненных работ команда алгоритмов на графах получила приложение GraBLEE, с помощью которого с легкостью можно получить полностью настроенный и оформленный в нужном виде файл cytoscape session, который можно быстро открыть для просмотра или отправить коллегам и заказчикам.

Благодаря использованию GraBLEE, пользователь тратит около 10 секунд, чтобы передать параметры в функцию и запустить программу. Для сравнения: прошлым ручным способом пользователь тратил по 2-3 минуты между всеми шагами настройки визуализации при каждой необходимости посмотреть на граф.

Приложение повысило производительность команды не только за счет экономии времени, но и благодаря уменьшению стресса и утомляемости разработчиков.