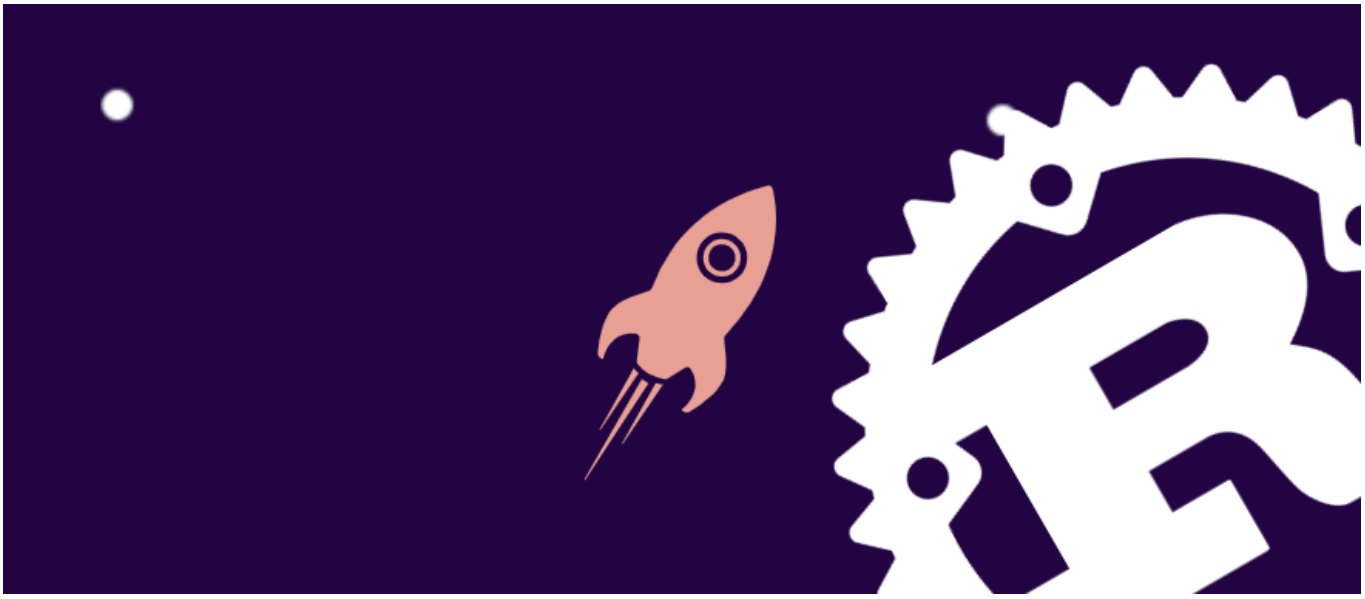
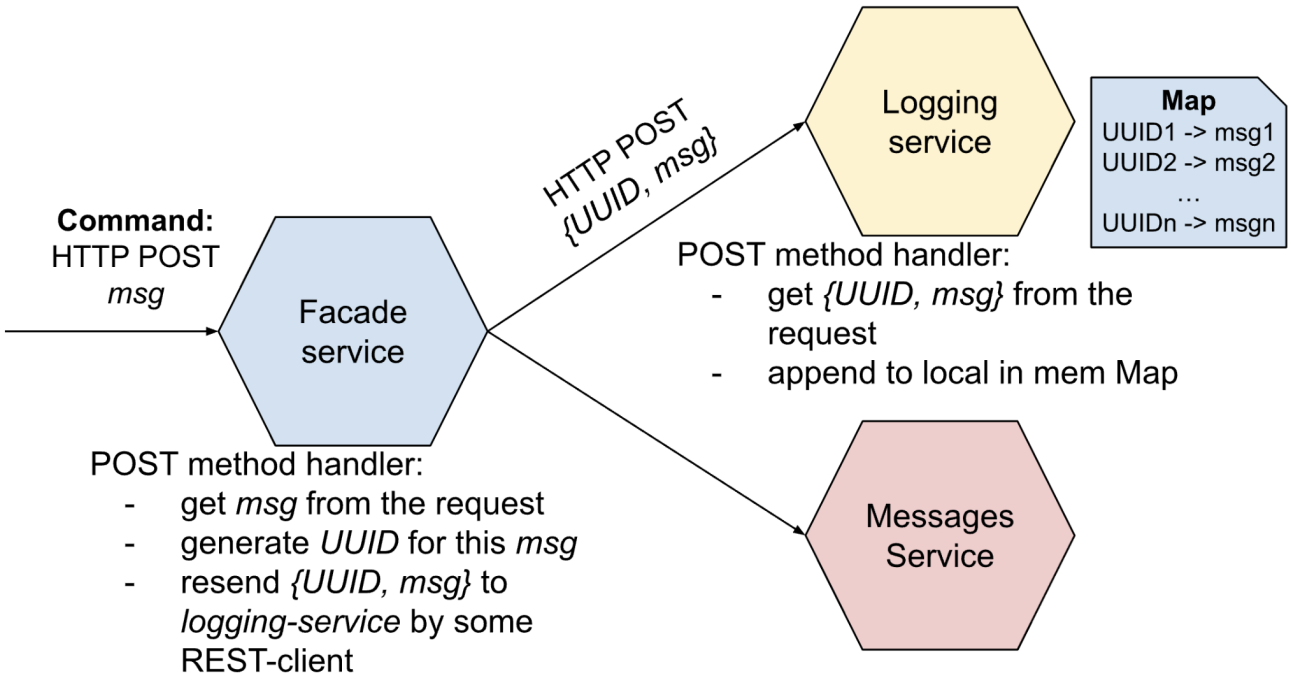
Report 1

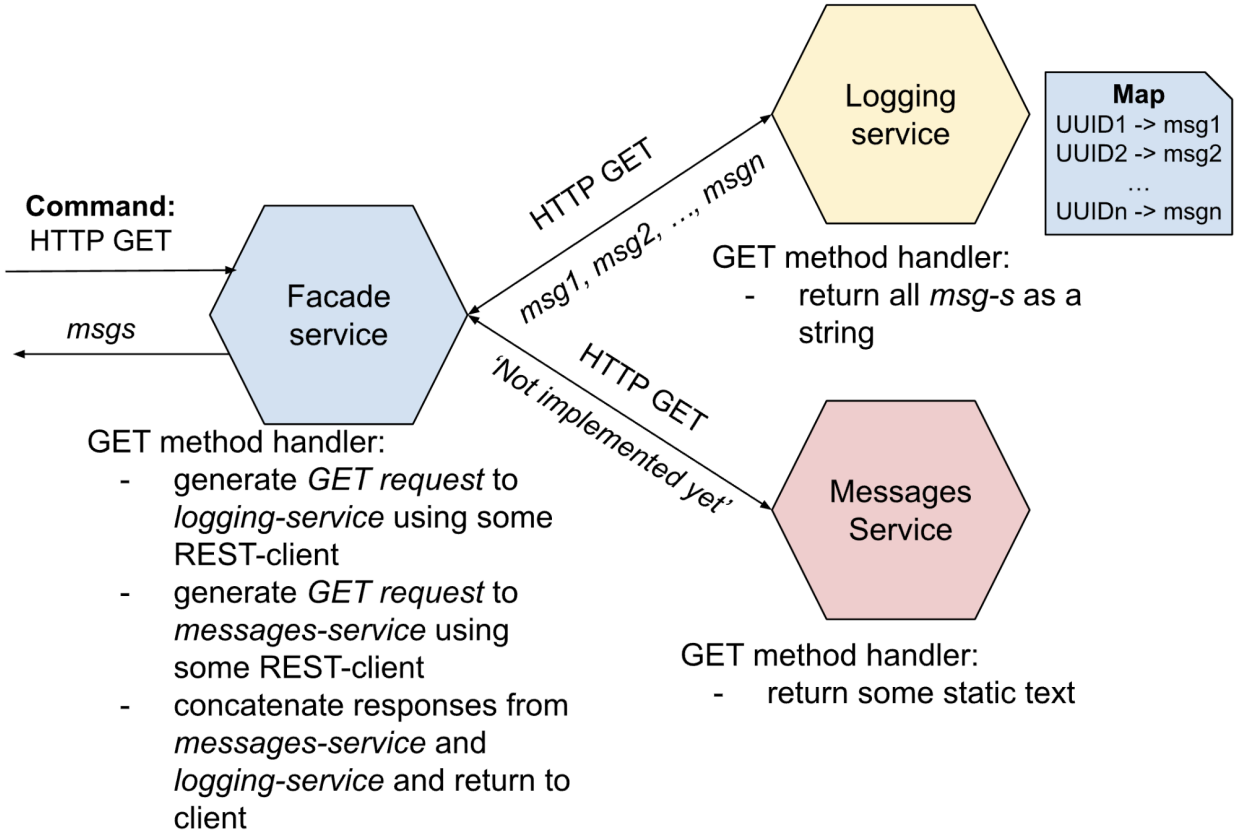


У цій лабораторній я створив три незалежні веб застосунки, які спілкуються між умовним клієнтом та обмінюються інформацією за принципами REST.

Лабораторну я виконував на мові програмування Rust у зв'язці із потужним фреймворком поверх - rocket. Він унікальний своєю простотою та повністю інкапсулює використання http-повідомлень за допомогою request - іншої спеціалізованої бібліотеки.

Спілкування між трьома сервісами організовано таким же чином, як і описано у завданні:





Отже, код організований наступним чином:

- корінь проекту містить так званий workspace - набір із незалежних пакетів

- всі три пакети лежать на рівень нижче і також містять свій `Dockerfile`.
- загальний `docker-compose.yml` лежить у корені проекту, і описує процес збору.

Щоб запустити проект за допомогою Docker, просто напишіть:

```
docker compose up
```

Або, щоб зібрати локально, встановіть Rust на вашу платформу, і введіть наступні команди в термінал з кореня проекту (вводьте окремо в різні сесії, щоб бачити результат):

```
cd facade-service
cargo run
```

```
cd logging-service
cargo run
```

```
cd messages-service
cargo run
```

Кожен із веб-застосунків повідомить про успішний запуск:

Отже, наші сервіси готові обмінюватися повідомленнями. Давайте ж перевіримо це. Для цього треба перейти в нове вікно терміналу, і спробувати послати повідомлення для facade-service. Зробимо це кілька разів, використавши curl (виконуйте команди одна за одною, щоб отримати результати):

```
curl -X POST http://localhost:8000/ -H "Content-Type:
application/json" -d '{"msg":"Hello World"}'

curl -X POST http://localhost:8000/ -H "Content-Type:
application/json" -d '{"msg":"I love Rust!"}'

curl -X POST http://localhost:8000/ -H "Content-Type:
application/json" -d '{"msg":"Give me the highest point, pls
=)"}'
```

Як бачимо, унікальні guid-и дійсно генеруються після кожного запиту:

```
➜  lab1 git:(micro_basics) ✗ curl -X POST http://localhost:8000/ -H "Content-Type: application/json" -d '{"msg":"Hello World"}'
07191436-459c-47db-bc4e-85ba3a1ec234%
➜  lab1 git:(micro_basics) ✗ curl -X POST http://localhost:8000/ -H "Content-Type: application/json" -d '{"msg":"I love Rust!"}'
777be0f1-8bed-4d89-8a7d-560a5ef8f51e%
➜  lab1 git:(micro_basics) ✗ curl -X POST http://localhost:8000/ -H "Content-Type: application/json" -d '{"msg":"Give me the highest point, pls =)"}'
f08bab42-7249-479b-9d6a-394a54fb80b1%
➜  lab1 git:(micro_basics) ✗
```

Поглянемо, чи отримав наші повідомлення facade-service:

```
POST / application/json:
   >> Matched: (index_post) POST / application/json
   >> Outcome: Success(200 OK)
   >> Response succeeded.
POST / application/json:
   >> Matched: (index_post) POST / application/json
   >> Outcome: Success(200 OK)
   >> Response succeeded.
POST / application/json:
   >> Matched: (index_post) POST / application/json
   >> Outcome: Success(200 OK)
   >> Response succeeded.
```

Дійсно, отримав. Як щодо logging-service?

```
POST /log application/json:
   >> Matched: (log_message) POST /log application/json
Logged: Hello World
   >> Outcome: Success(200 OK)
   >> Response succeeded.
POST /log application/json:
   >> Matched: (log_message) POST /log application/json
Logged: I love Rust!
   >> Outcome: Success(200 OK)
   >> Response succeeded.
POST /log application/json:
   >> Matched: (log_message) POST /log application/json
Logged: Give me the highest point, pls =)
   >> Outcome: Success(200 OK)
   >> Response succeeded.
```

І тут усе в порядку. Обожнюю Rust =)

Тепер давайте спробуємо отримати щось від facade-service:

```
curl http://localhost:8000/
```

Результат:

```
➜  lab1 git:(micro_basics) ✗ curl http://localhost:8000/

Hello World, I love Rust!, Give me the highest point, pls =) not implemented yet%
```

Як бачимо, маємо конкатеновану відповідь, і "заглушку" у вигляді повідомлення "`not implemented yet`".

Подивимося, чи дійсно messages-service перехопив повідомлення:

```
GET /message:
   >> Matched: (message) GET /message
   >> Outcome: Success(200 OK)
   >> Response succeeded.
```

І так, він дійсно це зробив.

Отже, сервіси працюють і виконують поставлену задачу. Дякую що прочитали це!