Пояснительная записка

направление Devops

Ермин Михаил Александрович

2025

Оглавление

[Введение 2](#_Toc193238990)

[Актуальность проекта 2](#_Toc193238991)

[Реализация 2](#_Toc193238992)

[Архитектура 2](#_Toc193238993)

[Вычисления 3](#_Toc193238994)

[Использование 4](#_Toc193238995)

[Использованные технологии 5](#_Toc193238996)

[Заключение 6](#_Toc193238997)

[Список использованной литературы 6](#_Toc193238998)

[Приложения 7](#_Toc193238999)

# Введение

PathFinder - это высокопроизводительный gRPC-сервис, предназначенный для поиска оптимального пути в лабиринте с учётом различных типов местности и количества персонажей. Данный сервис разработан с использованием различных технологий и может быть легко интегрирован в другие системы. Кроме того, удобство взаимодействия с сервисом является его преимуществом. Взаимодействие осуществляется через веб запросы

# Актуальность проекта

Оптимизация маршрутов является важной задачей в различных областях, таких как логистика, игры и навигационные системы. Использование gRPC позволяет достичь высокой производительности и минимальной задержки при передаче данных. Проект демонстрирует архитектурные решения, для обеспечения отказоустойчивости и надёжности системы.

# Реализация

## Архитектура

Изображение выглядит как диаграмма, текст, План, линия

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 1 – Взаимодействие компонентов архитектуры

Всего используется 5 контейнеров. Пользователь пишет запрос с текстовым файлом на Gateway, этот файл конвертируется в поток байтов и транслируется на сам Gateway, затем ретранслируется на loadbalancer, который распределяет запросы между двумя вычислительными Core серверами. После того как поток байтов попадает на Core, он конвертируется в txt и сохранятеся на самой машине. Далее выполня.тся сами вычисления.

На Core серверах происходят вычисления и затем на Gateway отправляется ответ в виде "Точки сбора" и ссылки на файл с отрисованной картой. Сама отрисованная карта отправляется на fileserver и оттуда становится доступна для пользователя.

## Вычисления

**Описание работы программы PathFinder**

Программа предназначена для поиска оптимальной точки сбора персонажей в лабиринте с различными типами местности. Каждый тип местности имеет свою стоимость перемещения. Программа загружает лабиринт из файла, находит оптимальную точку сбора и строит кратчайшие пути для каждого героя с учётом сложности местности.

**Вербальная модель**

1. Загрузка лабиринта из текстового файла, который содержит размеры лабиринта и его структуру.
2. Определение начальных позиций героев, отмеченных цифрами от '1' до '9'.
3. Поиск оптимальной точки сбора:
   * Перебор всех доступных клеток.
   * Для каждой клетки вычисление общей стоимости пути от всех героев.
   * Выбор точки с минимальной суммарной стоимостью.
4. Построение маршрута для каждого героя до найденной точки с использованием алгоритма Дейкстры (поиск кратчайшего пути на графе с весами).
5. Вывод информации о путях и визуализация лабиринта с отображением маршрутов и точки сбора.

**Математическая модель**

Пусть:

* M – матрица лабиринта размером h \* w, где M[i][j] – тип местности.
* H={h1,h2,…,hn} – множество позиций героев, где hi=(xi,yi).
* d(h, p) – длина кратчайшего пути от героя h до точки p с учётом стоимости клеток.

Целевая функция:

Где p – точка сбора, которая минимизирует суммарные затраты на перемещение.

**Блок-схема**

Приложение №1

**Комментарии к модулям и объектам (core/pathfinder)**

1. **Point**: структура, представляющая координаты (x,y)(x, y) на карте.
2. **terrainCost**: карта с типами местности и соответствующими им весами.
3. **readMaze**: функция чтения лабиринта из файла. Определяет размеры, загружает карту и находит героев.
4. **isValid**: проверяет, доступна ли точка для перемещения.
5. **findOptimalMeetingPoint**: перебирает все клетки и находит точку с минимальной стоимостью до всех героев.
6. **findPath**: поиск кратчайшего пути с использованием алгоритма Дейкстры и очереди с приоритетами (PriorityQueue).
7. **PriorityQueue**: реализация очереди с приоритетами на основе контейнера heap.
8. **printMazeWithPaths**: визуализация лабиринта с маршрутами и точкой сбора.

## Использование

Для использования необходимо:

**Клонирование репозитория**

git clone <https://github.com/your-username/tbolimpiada-grpc-2025.git>

**Переход в директорию**

cd tbolimpiada-grpc-2025

**Запуск через Docker Compose**

docker-compose up --build -d

Должно запуститься 5 контейнеров.

Взаимодействие с сервисом осуществляется через http запросы на Gateway сервер, который в свою очередь через балансировщик отправляет gRPC запросы на Core сервера.

Для взаимодействия с сервисом необходимо выполнить http запрос:

curl -X POST http://localhost:8081/upload -F "file=@maze.txt"

Вместо maze.txt необходимо вставить текстовый файл с лабиринтом в формате как представлен в задании с усложнением 2го уровня. Пример такого в приложении. (Приложение 2)

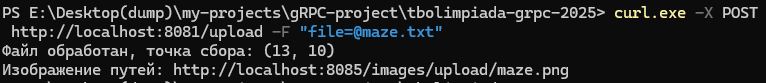


Рисунок 2 – Пример вывода ответа

Пример карты в текстовом файле. (Приложение 2)  
 Код преобразует её в цветную карту с отмеченными маршрутами и точкой сбора.

Изображение выглядит как пиксель, Красочность, прямоугольный, Прямоугольник

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 3 – Пример карты

Легенда карты:  
'R': Дорога (жёлтый)  
'G': Поле (зелёный)  
'S': Болото (коричневый)  
'H': Холмы (светло-зелёный)  
'F': Лес (тёмно-зелёный)  
'W': Вода (голубой)  
'M': Горы (серый)  
'X': Точка сбора (голубой)  
'Герой': Белый цвет

У каждого типа местности имеется свой коэффициент проходимости, согласно выданному заданию.

## Использованные технологии

Go - основной язык разработки

gRPC - для высокопроизводительного взаимодействия

Protobuf - для определения контрактов сервиса

Docker & Docker Compose - для контейнеризации и оркестрации

Nginx - балансировщик нагрузки и файловый сервер для хранения и раздачи изображений

# Заключение

PathFinder демонстрирует применение gRPC, микросервисного подхода и автоматизации развёртывания в реальном проекте. С доработками в рамках безопасности его можно использовать в реальных проектах.

# Список использованной литературы

Таненбаум Э., Уэзеролл Д. – **"Распределенные системы. Принципы и парадигмы"** – 2017.

Документация Go (Golang) – <https://go.dev/doc/>

Документация Docker – <https://docs.docker.com/>

Документация gRPC – <https://grpc.io/docs/>

Документация Nginx – <https://nginx.org/en/docs/>

# Приложения

Изображение выглядит как диаграмма, текст, линия, оригами

Контент, сгенерированный ИИ, может содержать ошибки.

Приложение №1 – Блок схема алгоритма поиска пути

30

20

MRGGGGRRRMMMMMMMMRFFWWWRRRMMMM

W2WGGRGRRRMMMMMMRRFFRWWF4FFRMM

WWWGGRGGRRWMMMMMRFFFWWWFFFFRRM

WSWGGGGRRRWRRRRRRFFFFFGGFFFRRR

SSSSRGGGRRWRGGGGRRRWWWHHHHRHHR

RSSHHHHWWWWWGGGGRRRWWWHHHHHRRR

RRGGWFGGRRWWGGGGRRRRWWWGWSSSSR

RRGGHWGGMRWWWGGGRRRRRRWWWSSSRS

RRGGHWGGMRWWWWGGRRRRWWWWWGRRRR

RRRRRRRRRRWRGGGGRRRWRRWWWGRRRR

WRGGGGRRRRWRG3GGHHRRRRGGGGRRRR

WRWGGRGHHHRRGHHHRRRRWRGGGGRRRR

RWGGGRHHHRWRGHHHFFFWWRGGFFFFFF

RSGGFFFFFFWRGGGGFFFWWRHHFFFWWW

RRRRRFFFFFFFFFFFFFFWWRHHWWSSSS

RFFFFFFFFWWWFFFFRRRWWRGGGGWWWS

RFFFFFFFWWWFFFFFFRRWWRGGGGWWWW

RRGGHWGGMRWRGGGGRRRGWWGGGGRRRR

RRGGHWGGMRWRGGGGRRRRWWGGGGRR1R

RRRRRRRRRRWRGGGGRRRRWWGGGGRRRM

Приложение 2 – Листинг файла карты из примера