Содержание

1.1 Телеграм-бот 2 1.2 User 2 1.3 Admin 2 1.4 MessageParser 2 1.5 TripHelperWorker 2 2 Схемы 4 2.1 Сервер 4 2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Onucahue paзвертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.2 Продуктивный режим 16 6.2 Продуктивный режим 16	1	Опи	исания		
1.3 Admin 2 1.4 MessageParser 2 1.5 TripHelperWorker 2 2 Схемы 4 2.1 Сервер 4 2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Onucatue passeptisbahus приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоснособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		1.1	Телеграм-бот		
1.4 MessageParser 2 1.5 TripHelperWorker 2 2 Cxемы 4 2.1 Сервер 4 2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Описание развертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоснособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		1.2	User		
1.5 TripHelperWorker 2 2 Схемы 4 2.1 Сервер 4 2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Описание развертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16		1.3	Admin		
2 Схемы 4 2.1 Сервер 4 2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Oписание развертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		1.4	MessageParser		
2.1 Сервер 4 2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Onucahue paзвертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		1.5	TripHelperWorker		
2.2 MongoDB 5 2.3 HealthCheckModule 6 2.4 AdminModule 7 2.5 PlannedTripsModule 9 2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Onucahue pasbeptubahus приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	2	Cxe	емы 4		
2.3 HealthCheckModule62.4 AdminModule72.5 PlannedTripsModule92.6 TripHelperModule112.7 TripHistoryModule133 Описание развертывания приложения153.1 Подготовка окружения153.2 Конфигурация инфраструктуры153.3 Развертывание бота154 Сборка приложения154.1 Установка зависимостей154.2 Сборка Fat JAR154.3 Тестирование154.4 Создание Docker-образа155 Деплой приложения155.1 Загрузка образа в Docker Hub155.2 Развертывание на сервере165.3 Проверка работоспособности166 Запуск приложения166.1 Локальный запуск (для разработки)166.2 Продуктивный режим16		2.1	Сервер		
2.4 AdminModule72.5 PlannedTripsModule92.6 TripHelperModule112.7 TripHistoryModule133 Описание развертывания приложения153.1 Подготовка окружения153.2 Конфигурация инфраструктуры153.3 Развертывание бота154 Сборка приложения154.1 Установка зависимостей154.2 Сборка Гат JAR154.3 Тестирование154.4 Создание Docker-образа155 Деплой приложения155.1 Загрузка образа в Docker Hub155.2 Развертывание на сервере165.3 Проверка работоспособности166 Запуск приложения166.1 Локальный запуск (для разработки)166.2 Продуктивный режим16		2.2	MongoDB		
2.5PlannedTripsModule92.6TripHelperModule112.7TripHistoryModule133Описание развертывания приложения153.1Подготовка окружения153.2Конфигурация инфраструктуры153.3Развертывание бота154Сборка приложения154.1Установка зависимостей154.2Сборка Fat JAR154.3Тестирование154.4Создание Docker-образа155Деплой приложения155.1Загрузка образа в Docker Hub155.2Развертывание на сервере165.3Проверка работоспособности166Запуск приложения166.1Локальный запуск (для разработки)166.2Продуктивный режим16		2.3	HealthCheckModule		
2.6 TripHelperModule 11 2.7 TripHistoryModule 13 3 Описание развертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		2.4	AdminModule		
2.7 TripHistoryModule 13 3 Описание развертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		2.5	PlannedTripsModule		
3 Описание развертывания приложения 15 3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		2.6	TripHelperModule		
3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		2.7			
3.1 Подготовка окружения 15 3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	3	Опи	исание развертывания приложения		
3.2 Конфигурация инфраструктуры 15 3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	-				
3.3 Развертывание бота 15 4 Сборка приложения 15 4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		3.2			
4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		3.3			
4.1 Установка зависимостей 15 4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	4	Сборка приложения			
4.2 Сборка Fat JAR 15 4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16			± ±		
4.3 Тестирование 15 4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16					
4.4 Создание Docker-образа 15 5 Деплой приложения 15 5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16					
5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16			<u> </u>		
5.1 Загрузка образа в Docker Hub 15 5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	5	Лег	глой приложения		
5.2 Развертывание на сервере 16 5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	•		1		
5.3 Проверка работоспособности 16 6 Запуск приложения 16 6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16			± v		
6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16		_	1 1		
6.1 Локальный запуск (для разработки) 16 6.2 Продуктивный режим 16	6	Зап	уск приложения		
6.2 Продуктивный режим	•				
- rw r					
		6.3	Команды для управления		

1 Описания

1.1 Телеграм-бот

- Компонент: интерфейс для взаимодействия с пользователем через Telegram.
 - Передача команд и запросов от пользователя на сервер.
 - Отображает ответы сервера в чате бота.

1.2 User

- Роль:
 - Telegram пользователь.
- Функции:
 - Инициализирует запросы через Telegram бота, такие как: запланировать поездку, добавить заметку к путевой точке и другие.
 - Делится геолокацией.
 - Получает сообщения от Telegram бота.
 - Имеет доступ к эндпоинту /healthcheck без авторизации.

1.3 Admin

- Роли:
 - администратор системы.
- Функции:
 - Имеет доступ к эндпоинту /users с авторизацией по API ключу.
 - Имеет доступ к эндпоинту /healthcheck без авторизации.

1.4 MessageParser

- Компонент:
 - обработка сообщений от Telegram бота.
- Функции:
 - Обрабатывает сообщения от Telegram бота.
 - Направляет команды в соответствующий модуль.

1.5 TripHelperWorker

- Компонент:
 - модуль помощник с поездкой.
- Функции:
 - Фоново следит за состоянием поездок пользователей.
 - Уведомляет пользователей об их ближайшей поездке неделю до её начала и один день до её начала.

- Автоматически делает запланированную поездку текущей при наступлений даты её начала.
- Автоматически перемещает текущую поездку в историю поездок при истечению даты её конца.
- Заращивает у пользователя геолокацию в ходе поездки.
- Автоматически отмечает путевую точку посещенной, если геолокация пользователя входит в определённый радиус вокруг неё.

2 Схемы

2.1 Сервер

Основной компонент.

Функции:

- Обрабатывает все веб запросы и запросы от телеграмм бота.
- Обеспечение взаимодействия между всеми компонентами.

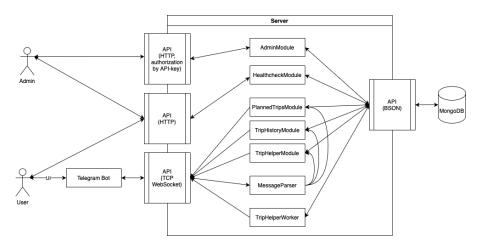


Рис. 1: Схема сервера

2.2 MongoDB

- База данных:
 - документоориентированная база данных.
- Функции:
 - Хранит информацию о администраторах.
 - Хранит информацию о пользователях и их поездках.

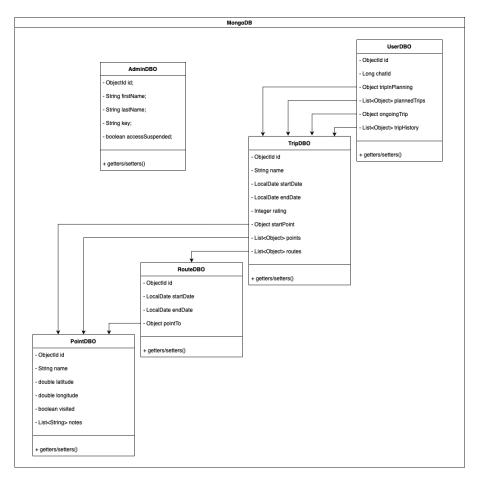


Рис. 2: Схема MongoDB (NoSQL)

2.3 HealthCheckModule

- Компонент:
 - модуль проверки состояния сервера.

• Функции:

- Реализует эндпоинт /healthcheck для проверки работоспособности сервера и получения списка авторов.
- Доступен без авторизации.

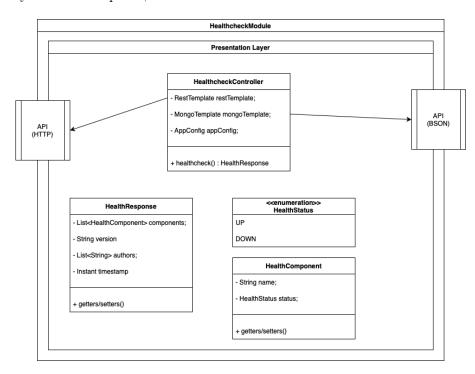
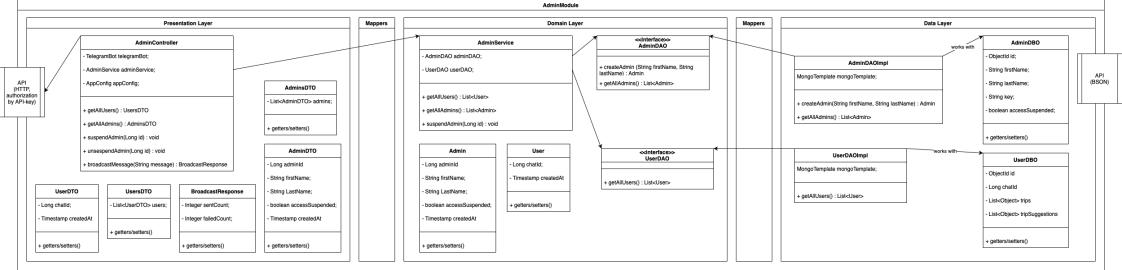


Рис. 3: Схема HealthChecker

2.4 AdminModule

- Компонент:
 - модуль администрации системы.
- Функции:
 - Реализует эндпоинт /users, отображающий данные о всех пользователях.
 - Доступен администраторам с авторизацией.



2.5 PlannedTripsModule

• Компонент:

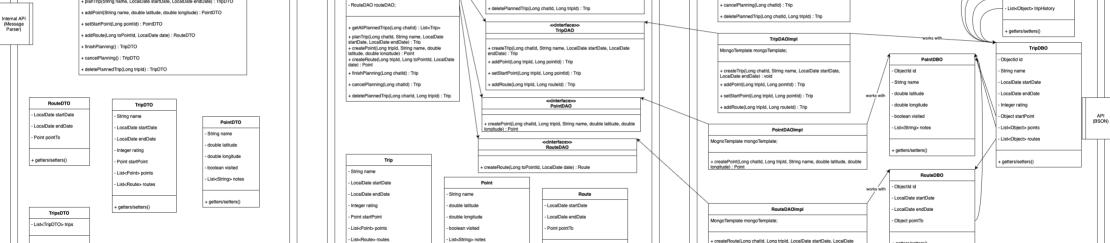
– модуль планирования поездок.

• Функции:

- Обрабатывает команду /showplanned, получение списка запланированных поездок.
- Обрабатывает команду /plantrip, запланирование новой поездки.
- Обрабатывает команду /addpoint, добавление путевой точки в поездку.
- Обрабатывает команду /setstartpoint, установление начальной путевой точки в поездке.
- Обрабатывает команду /addroute, добавление маршрута между точками.
- Обрабатывает команду /finishplanning, запланирование поездки.
- Обрабатывает команду /cancelplanning, отмета планирования.
- Обрабатывает команду /deleteplanned, удаление запланированной поездки.

Presentation Laver Mappers Bussines Logic Mappers Data Laver UserDBO UserDAO PlannedTripsControlle UserDAOImpi - ObjectId id TelegramBot telegramBot: PlannedTripsService + getAllPlannedTrips(Long chatId) : List<Trip> - Long chatld (TCP PlannedTripsService plannedTripsService: WebSocket) - UserDAO userDAO: + getAllPlannedTrips(Long chatId) : List<Trip> + getTripInPlanning(Long chatId) : Trip - Object tripInPlanning - TripDAO tripDAO: + getTripInPlanning(Long chatId) : Trip + finishPlanning(Long chatld): Trip - List<Object> plannedTrips + showAllPlannedTrips() : TripsDTO - PointDAO pointDAO + finishPlanning(Long chatld) : Trip + cancelPlanning(Long chatld) : Trip - Object ongoingTrip + planTrip(String name, LocalDate startDate, LocalDate endDate) : TripDTO - RouteDAO routeDAO: + cancelPlanning(Long chatld) : Trip + deletePlannedTrip(Long chatld, Long tripId) : Trip List<Object> tripHistory + addPoint(String name, double latitude, double longitude) : PointDTO

PlannedTripsModule



+ getters/setters()

+ getters/setters()

+ getters/setters()

+ getters/setters()

+ aetters/setters()

endDate, Long toPointId) : Route

2.6 TripHelperModule

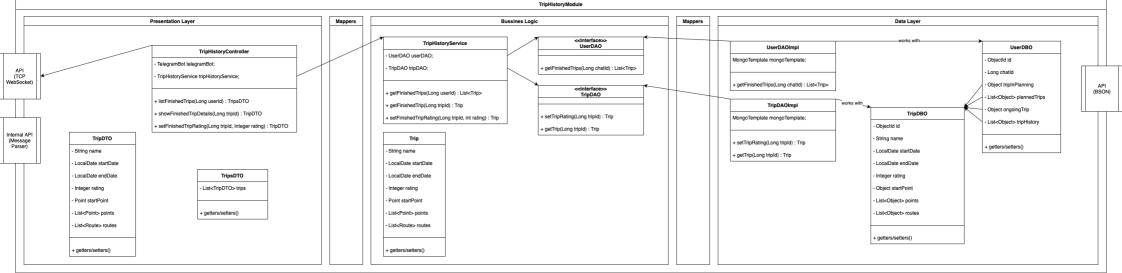
- Компонент:
 - модуль помощник с поездкой.

• Функции:

- Обрабатывает команду /showongoingtrip, отображение информации о текущей поездке.
- $-\,$ Обрабатывает команду /addnote, добавление заметки к путевой точке одной из поездкой.
- $-\,$ Обрабатывает команду /markpoint, мануальная отметка путевой точки текущей поездки как посещенной.

2.7 TripHistoryModule

- Компонент:
 - модуль истории поездок.
- Функции:
 - $-\,$ Обрабатывает команду /triphistory, отображает список всех завершенных поездок.
 - $-\,$ Обрабатывает команду /finisheddetails, отображает подробную информацию о конкретной завершенной поездке.
 - Обрабатывает команду /ratefinished, установливает оценку завершенной поездке.



3 Описание развертывания приложения

3.1 Подготовка окружения

- Установка Docker и Docker Compose на сервер.
- Настройка переменных окружения ('.env') для подключения к Telegram API, MongoDB и Kafka.

3.2 Конфигурация инфраструктуры

- Запуск MongoDB и Kafka через Docker Compose.
- Настройка репликации и шардинга для MongoDB (если требуется).
- Проверка доступности брокера сообщений Kafka.

3.3 Развертывание бота

- Сборка Docker-образа приложения ('docker build').
- Публикация образа в Docker Hub.
- Запуск контейнера с ботом на сервере ('docker-compose up').

4 Сборка приложения

4.1 Установка зависимостей

- Запуск 'gradle build' для загрузки зависимостей (Spring Boot, Telegram API, MongoDB Driver и т.д.).
- Проверка совместимости версий Java (SE 23) и Spring (6.2).

4.2 Сборка Fat JAR

- Генерация исполняемого JAR-файла с помощью Gradle ('./gradlew bootJar').
- Проверка наличия всех зависимостей в 'build/libs/'.

4.3 Тестирование

- Запуск unit-тестов ('./gradlew test').
- Проверка покрытия кода (минимум 60%).

4.4 Создание Docker-образа

- Написание 'Dockerfile' с базой на 'openjdk:23'.
- Копирование JAR-файла и запуск приложения в контейнере.

5 Деплой приложения

5.1 Загрузка образа в Docker Hub

• Авторизация ('docker login').

• Пуш образа ('docker push username/trip-planner-bot:latest').

5.2 Развертывание на сервере

- Запуск MongoDB и Kafka через 'docker-compose.yml'.
- Развертывание бота в отдельном контейнере.

5.3 Проверка работоспособности

- Тестирование команд бота через Telegram.
- Мониторинг логов на предмет ошибок ('docker logs < container id>').

6 Запуск приложения

6.1 Локальный запуск (для разработки)

- Запуск MongoDB и Kafka через Docker Compose.
- Запуск бота через './gradlew bootRun'.

6.2 Продуктивный режим

- Запуск всех сервисов через 'docker-compose up -d'.
- Настройка авто-рестарта при падении ('restart: always').

6.3 Команды для управления

- Остановка: 'docker-compose down'.
- Обновление: 'docker-compose pull && docker-compose up -d'.