Сервисы шлюза

Proxy

Единая точка входа для http и websocket

Fastify

Один из вариантов коммуникации с клиентами и сторонними сервисами - веб сервер Fastify. Сервис-обвертка инициализирет библиотеку fastify с расширениями: `@fastify/cors, @fastify/swagger, @fastify/swagger-ui, @fastify/multipart`. В отличии от оригинального Fastify, сервис позволяет динамически добавлять и удалять новые маршруты без прерывания работы запущенных запросов после запуска сервера. Также реализована плавная(greceful) остановка сервиса.

Для начала, надо добавить пакет из corelibs:

```
npm install ./corelibs/fastify
```

Пример работы с Fastify:

```
import {Fastify} from  '@morphcluster/fastify'
...
//Добавление Fastify к хосту
let fastify = new Fastify( host, {"host" : "127.0.0.1", "port" : 80} )
host.addService(fastify)
...
//Добавление маршрута, в формате fastify
fastify.addRoute({...})
//Запуск пересоздания машрутов, чтобы отобразились изменения. Начатые вызовы сброшены не будут
await fastify.restart()
```

Формат для добавления новых маршрутов можно посмотреть в официальной документации

Gateway

Шлюз это HTTP сервер, который позволяет общаться из внешней среды с сервисами. Чтобы запрос сервиса был доступен через шлюз, надо описать его схеме, что он доступен во внешнем контуре. Также внешний контур проверяет наличие сессии. При отсутствии сессии будут работать только анонимные методы.

Чтобы публичные запросы сервисов хоста были доступны по http(s), можно воспользоваться сервисом Gateway. Он автоматически обнаружит запросы сервиса к публикации и создаст маршруты.

```
import {ServiceHost} from '@morphcluster/core'
import {Fastify,FastifyRest} from  '@morphcluster/fastify'
...
//Fastify - зависимость FastifyRest
host.addService(new Fastify( host, {"host" : "127.0.0.1", "port" : 80} ))
//Добавление FastifyRest
host.addService(new FastifyRest( host ))
//Для публикации, сервис должен обязательно быть добавлен в ServiceHost
host.addService(new Test())
```

Проверить работу можно будет по ссылке `http://127.0.0.1/documentation`. Там автоматически разворачиватся OpenApi/Swagger со всеми доступными маршрутами. Вызов, производимый извне имеет проверку доступа, при этом запросы между сервисами не замедляются этой проверки, сессия передается напрямую.

Sessions

Сессии соответственно хранят информацию о доступе, подтвержденную токеном. Токен создается при авторизации или иным способом. Пользователю или внешнему сервису нужен токен для взаимодействия с системой через шлюз. Система через него идентифицирует пользователя. Сессия может быть определена как привилегированная(isAdmin), что расширяет возможности такого пользователя.

Auth

Eventer

Bridge

Мост. Мост позволяет общаться привилегированным сессиям с внутренним контуром, недоступным через шлюз.

Revision #5
Created 17 June 2026 18:13:02 by Admin
Updated 17 June 2026 18:16:48 by Admin