education

HTTP и компания, ч.2

Разработка веб-сервисов на Golang

education

Виктор Горячкин

Старший программист VK Tech

education

О чём поговорим?

О чём поговорим?

- ➔ Шаблоны

О чём поговорим?

- ➔ Шаблоны
- ➔ REST

О чём поговорим?

- ➔ Шаблоны
- ➔ REST
- ➔ Аутентификация на основе сессий

О чём поговорим?

- ➔ Шаблоны
- ➔ REST
- ➔ Аутентификация на основе сессий
- ➔ Telegram-бот

О чём поговорим?

- ➔ Шаблоны
- ➔ REST
- ➔ Аутентификация на основе сессий
- ➔ Telegram-бот
- ➔ Vendoring

Виды контента

Виды контента

- Статический — неизменный контент, который остается одинаковым для всех пользователей и отображается так, как определено на сервере

Виды контента

- Статический — неизменный контент, который остается одинаковым для всех пользователей и отображается так, как определено на сервере
- Динамический — контент, который динамически изменяется в зависимости от действий пользователя и других факторов, формируется в момент запроса

Варианты клиент-серверного взаимодействия

Варианты взаимодействия

- RPC — remote procedure call

Варианты взаимодействия

- RPC — remote procedure call
- REST — representational state transfer

Варианты взаимодействия

- RPC — remote procedure call
- REST — representational state transfer
- GraphQL — query language for API, разработан в Facebook

RPC

Вызов процедуры на удаленном сервере

- ➔ https://localhost/rpc?get_all_users
- ➔ https://localhost/rpc?get_all_users_s

RPC

Вызов процедуры на удаленном сервере

- https://localhost/rpc?get_all_users
- https://localhost/rpc?get_all_users_s

Примеры

- JSON-RPC

RPC

Вызов процедуры на удаленном сервере

- https://localhost/rpc?get_all_users
- https://localhost/rpc?get_all_users_s

Примеры

- JSON-RPC
- gRPC

REST - принципы

➔ Client-server model — клиент-серверная модель взаимодействия

REST - принципы

- Client-server model — клиент-серверная модель взаимодействия
- Stateless — состояние клиента, его сессии не хранится на сервере, запросы не должны зависеть друг от друга, каждый запрос содержит всю необходимую информацию для его обработки

REST - принципы

- Client-server model — клиент-серверная модель взаимодействия
- Stateless — состояние клиента, его сессии не хранится на сервере, запросы не должны зависеть друг от друга, каждый запрос содержит всю необходимую информацию для его обработки
- Uniform interface — единообразие интерфейсов, единый способ обращения ко всем ресурсам

REST - принципы

- Client-server model — клиент-серверная модель взаимодействия
- Stateless — состояние клиента, его сессии не хранится на сервере, запросы не должны зависеть друг от друга, каждый запрос содержит всю необходимую информацию для его обработки
- Uniform interface — единообразие интерфейсов, единый способ обращения ко всем ресурсам
- Layered system — многоуровневая система, клиент или промежуточный сервер абстрагированы от инфраструктуры

REST - принципы

- Client-server model — клиент-серверная модель взаимодействия
- Stateless — состояние клиента, его сессии не хранится на сервере, запросы не должны зависеть друг от друга, каждый запрос содержит всю необходимую информацию для его обработки
- Uniform interface — единообразие интерфейсов, единый способ обращения ко всем ресурсам
- Layered system — многоуровневая система, клиент или промежуточный сервер абстрагированы от инфраструктуры
- Code on demand — предоставление кода по требованию, необязательное требование

REST структура HTTP сообщения

	Request	Response
Start line	GET hello/world HTTP 1.1	HTTP 1.1 200 OK
Headers	Accept-language: ru,en	Content-type: application/json
Empty line		

Body		{
		"message": "Hello world !"
		}

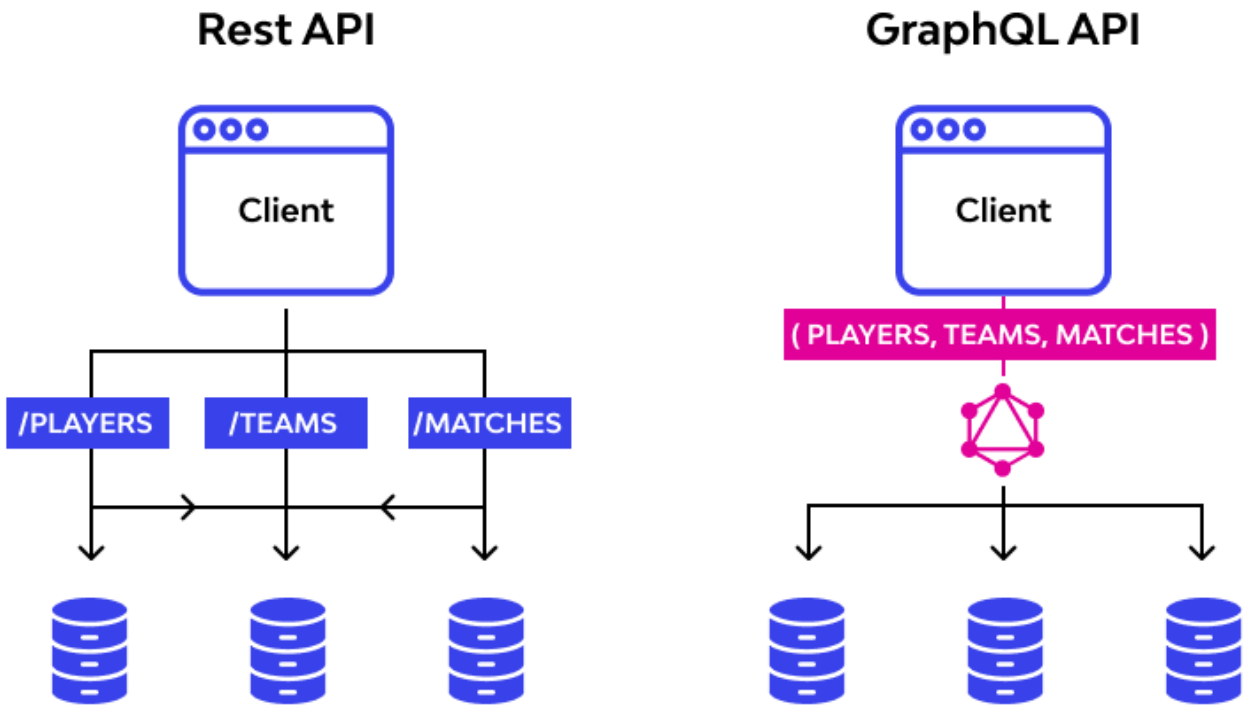
Request methods

- GET /user?sort=name&limit=10 — получить список
- GET /user/101 — единичная запись
- POST /user — создание
- PUT /user — полное обновление
- PATCH /user — частичное обновление
- DELETE /user

Response codes

- 1xx - information
- 2xx - success ; 201 - created; 202 - accepted; 200 - ok
- 3xx - redirect; 307 - temporary redirect; 308 - permanent redirect
- 4xx - client error; 400 - bad request; 401 - unauthorized
- 5xx - server error; 500 - internal server error; 502 - bad gateway

GraphQL


Rest API
Client
/PLAYERS
/TEAMS
/MATCHES
GraphQL API
Client
(PLAYERS, TEAMS, MATCHES)

GraphQL схема

Gitlab — https://docs.gitlab.com/api/graphql

```
{
  users(usernames: ["user1", "user3", "user4"]) {
    pageInfo {
      endCursor
      startCursor
      hasNextPage
    }
    nodes {
      id
      username,
      publicEmail
      location
      webUrl
      userPermissions {
        createSnippet
      }
    }
  }
}
```

education

Вопросы?