

ЗастУВим бэкенд

Шакшина Мария

Rooh Solutions



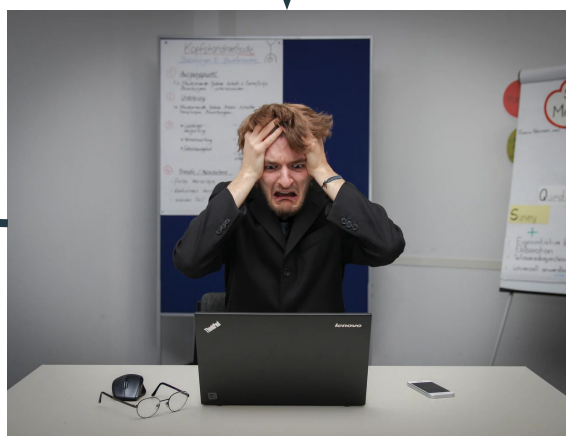
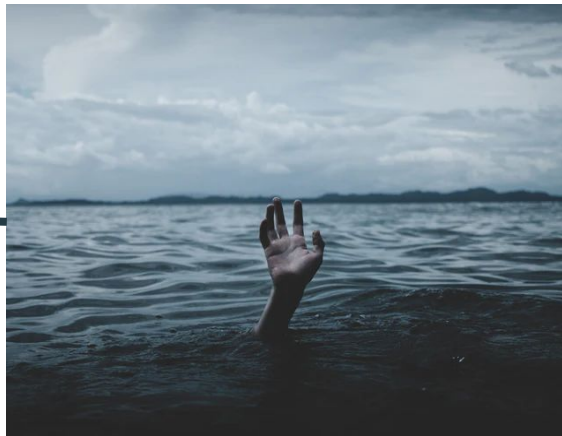
КТО Я?

- Senior JS developer в Rooh Solutions
- Пишу на React
- Опыт в разработке ~6 лет
- Интересуюсь различными направлениями в IT области





Фильмец!



Где взять бэкенд?





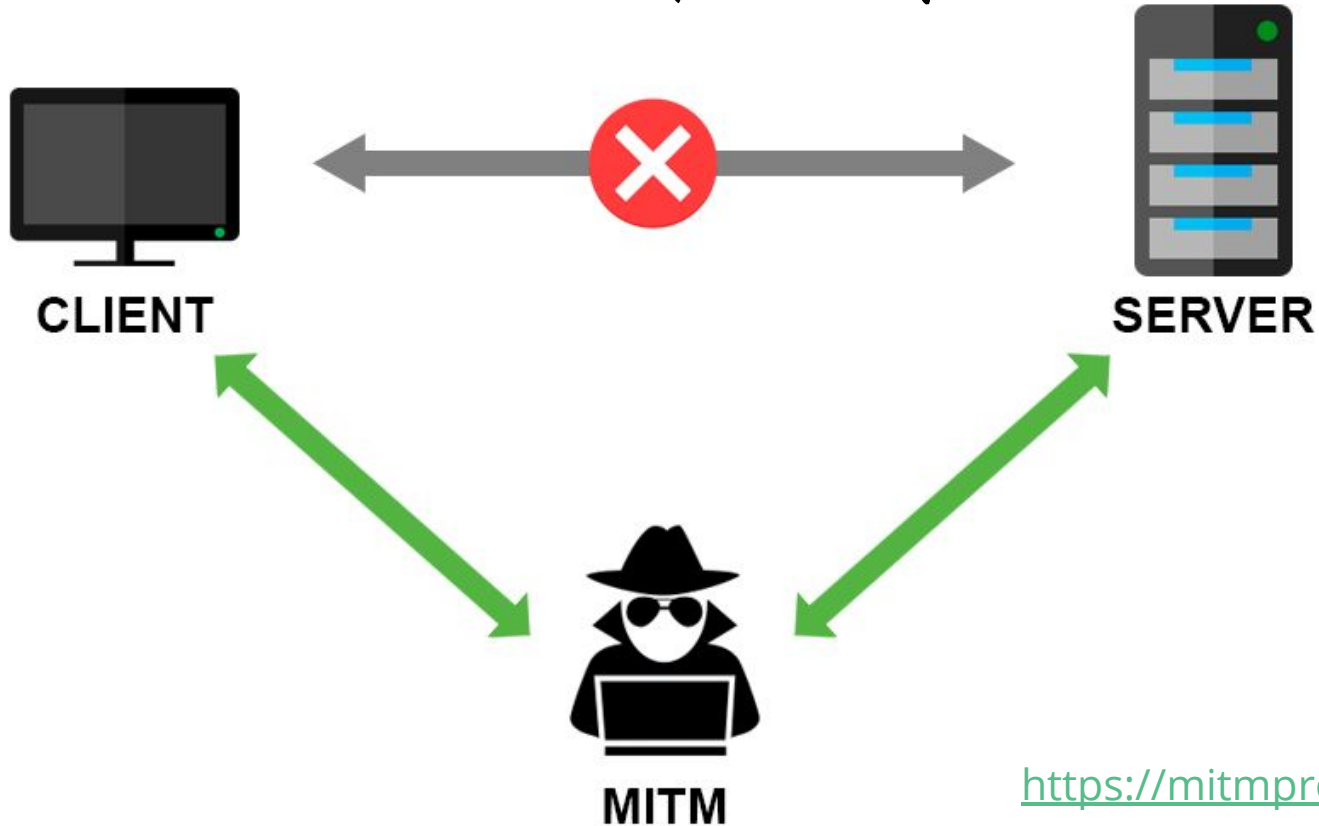
Проксирование



Инструменты для проксирования

- Mitmproxy
- Fiddler
- Charles
- Server-replay

Mitmpoxy





Mitmproxy

Using mitmproxy as a mock server

```
from mitmproxy import http
import json, re

with open('user.json') as json_file:
    userData = json.load(json_file)

def request(flow: http.HTTPFlow):
    if flow.request.pretty_host == "api.github.com" and re.match('/users', flow.request.path):
        flow.response = http.HTTPResponse.make(
            200, # (optional) status code
            json.dumps(userData), # (optional) content
            {
                "Content-Type": "application/json",
                "Access-Control-Allow-Origin": "*"
            } # (optional) headers
        )
```

Примеры на Python

<https://github.com/mitmproxy/mitmproxy/tree/master/examples>



Потрясающе!





Charles

WEB DEBUGGING PROXY APPLICATION

for Windows, Mac OS and Linux

HOME

OVERVIEW

DOCUMENTATION

DOWNLOAD

BUY

SUPPORT

Charles is an HTTP proxy / HTTP monitor / Reverse Proxy that enables a developer to view all of the HTTP and SSL / HTTPS traffic between their machine and the Internet. This includes requests, responses and the HTTP headers (which contain the cookies and caching information).

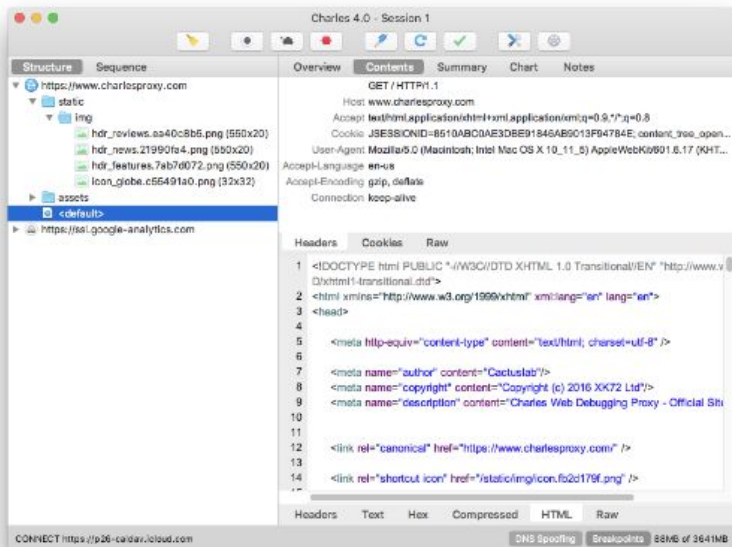
[READ MORE](#)

<https://www.charlesproxy.com/>

DOWNLOAD



Download a free trial
Version 4.5.6



Limited time offer: Get 10 free
[Adobe Stock images.](#)

[ads via Carbon](#)

FEATURES



Records all traffic between
your browser and the Internet



Reveals the contents of all
requests, responses, cookies
and headers



A wide-angle shot of a modern movie theater. The room is dimly lit with blue ambient lighting on the walls. The front wall is dominated by a large white screen displaying the name 'Charles' in a large, white, sans-serif font. The theater is filled with rows of dark, upholstered seats, all facing the screen. On either side of the screen, the walls feature vertical panels with a textured, golden-brown pattern. A small doorway with a green exit sign is visible on the right wall. The ceiling has recessed lighting fixtures.

Charles

Fiddler

<https://www.telerik.com/fiddler>



Protocol	Host	URL	Body	Ca
HTTPS	www.fiddler2.com	/UpdateCheck.aspx?isBet...	785	pr
HTTP	Tunnel to	www.facebook.com:443	754	
HTTP	Tunnel to	www.facebook.com:443	0	
HTTP	Tunnel to	www.facebook.com:443	520	
HTTP	Tunnel to	www.facebook.com:443	-1	

Request Headers

CONNECT www.facebook.com:443 HTTP/1.1

[\[Raw\]](#) [\[Header Definitions\]](#)

Client

User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/81.0.4044.138 Safari/537.36

Transport

Connection: keep-alive

Host: www.facebook.com:443

Breakpoint Hit Tamper the Break on Response Run to Completion Choose Response...

Transformer Headers TextView SyntaxView ImageView HexView WebView Auth Caching Cookies Raw JSON XML

HTTP/200 responses are cacheable by default, unless Expires, Pragma, or Cache-Control headers are present and forbid caching.

This response does not specify explicit HTTP Cache Lifetime information and does not specify a Last-Modified date. Heuristic expiration is typically based on Last-Modified date. Lacking Last-Modified, this response may be revalidated on every use or once per browsing session, depending on the browser configuration.

This response contains neither an ETag nor a Last-Modified time. This will prevent a Conditional Revalidation of this response.

=====

Learn more about caching at <http://fiddler2.com/r/2httpperf>

#	Edit	Rules	Tools	View	Help	#	Result	Protocol	Host	URL	
659						200	HTTP	Tunnel	da1891jfb0i.doudfon		
660						200	HTTP	Tunnel	fonts.gstatic.com:443		
661						200	HTTP	Tunnel	logx.optimizely.com:4		
662						200	HTTP	Tunnel	beacons.gcp.gvt2.cor		
663						200	HTTP	Tunnel	www.google.ru:443		
664	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
665						200	HTTP	Tunnel	cdn.cookiecslaw.org:44		
666						200	HTTP	Tunnel	dec.azureedge.net:4		
667						200	HTTP	Tunnel	script.hotjar.com:443		
668						200	HTTP	Tunnel	docs.telerik.com:443		
669						200	HTTP	Tunnel	api.dec.sitefinity.com		
670						200	HTTP	Tunnel	stats.g.doubleclick.ne		
671	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
672						200	HTTP	Tunnel	ssl.gstatic.com:443		
673	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
674						200	HTTP	Tunnel	google.com:443		
675	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
676						200	HTTP	Tunnel	play.google.com:443		
677						200	HTTP	Tunnel	play.google.com:443		
678						200	HTTP	Tunnel	play.google.com:443		
679						200	HTTP	Tunnel	beacons.gcp.gvt2.cor		
680						200	HTTP	Tunnel	play.google.com:443		
681						200	HTTP	Tunnel	beacons.gcp.gvt2.cor		
682	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
683						200	HTTP	Tunnel	consumer.entitlement		
684	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
685	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
686						200	HTTP	Tunnel	browser.pipe.aria.mic		
687	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
688	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
689	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
690	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
691	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
692	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
693	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
694	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
695	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
696						200	HTTP	Tunnel	activity.windows.com		
697	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
698	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
699	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
700	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
701	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
702						200	HTTP	Tunnel	d27xxe7juh.tus6.dou		
703						200	HTTP	Tunnel	export.yandex.ru:44		
704						200	HTTP	Tunnel	api.lingualco.com:443		
705	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
706	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
707	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
708	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
709	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
710	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			
711	▲	502	HTTP			127.0.0.1:63342	/browserConnection/	/browserConnection/			

Save Script Go to:

```

BindPref("Fiddlerscript.ephemeral.bpMethod")
public static var bpMethod: String = null;

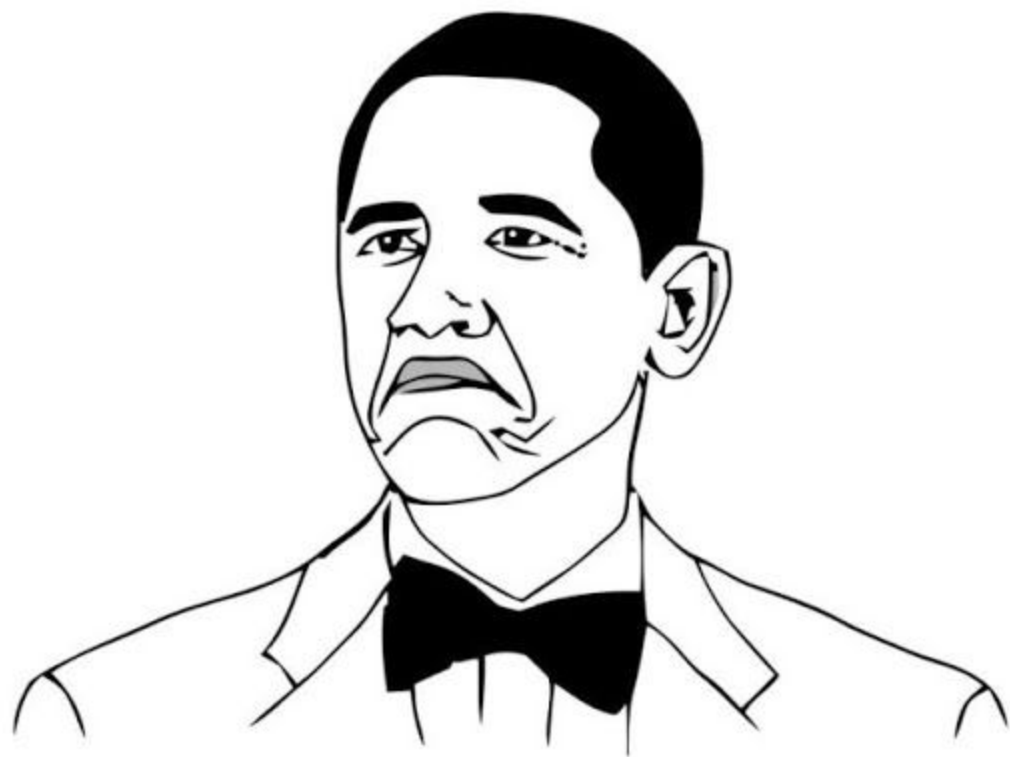
static var bpStatus: int = -1;
static var uiBoldURI: String = null;
static var gs_ReplaceToken: String = null;
static var gs_ReplaceTokenWith: String = null;
static var gs_OverrideHost: String = null;
static var gs_OverrideHostWith: String = null;

// The OnExecAction function is called by either the QuickExec box in the Fiddler window,
// or by the ExecAction.exe command line utility.
static function OnExecAction(sParams: String[]): Boolean {
    FiddlerObject.StatusText = "ExecAction:" + sParams[0];

    var sAction = sParams[0].toLowerCase();
    switch (sAction) {
        case "bold":
            if (sParams.Length < 2) { uiBoldURI = null; FiddlerObject.StatusText = "Bolding cleared"; return false; }
            uiBoldURI = sParams[1]; FiddlerObject.StatusText = "Bolding requests for " + uiBoldURI;
            return true;
        case "bp":
            FiddlerObject.alert("bpu = breakpoint request for uri\nbpm = breakpoint request method\nbprs=breakpoint response status\nbpaftr = breakpoint response for URI");
            return true;
        case "bps":
            if (sParams.Length < 2) { bpStatus = -1; FiddlerObject.StatusText = "Response Status breakpoint cleared"; return false; }
            bpStatus = parseInt(sParams[1]); FiddlerObject.StatusText = "Response status breakpoint for " + sParams[1];
            return true;
        case "bpv":
        case "bpm":
            if (sParams.Length < 2) { bpMethod = null; FiddlerObject.StatusText = "Request Method breakpoint cleared"; return false; }
            bpMethod = sParams[1].toUpperCase(); FiddlerObject.StatusText = "Request Method breakpoint for " + bpMethod;
            return true;
        case "bpu":
            if (sParams.Length < 2) { bpRequestURI = null; FiddlerObject.StatusText = "RequestURI breakpoint cleared"; return false; }
            bpRequestURI = sParams[1];
            FiddlerObject.StatusText = "RequestURI breakpoint for " + sParams[1];
            return true;
        case "bpa":
        case "bpaftr":
            if (sParams.Length < 2) { bpResponseURI = null; FiddlerObject.StatusText = "ResponseURI breakpoint cleared"; return false; }
            bpResponseURI = sParams[1];
            FiddlerObject.StatusText = "ResponseURI breakpoint for " + sParams[1];
            return true;
        case "overridehost":
            if (sParams.Length < 3) { gs_OverrideHost = null; FiddlerObject.StatusText = "Host Override cleared"; return false; }
            gs_OverrideHost = sParams[1].toLowerCase();
            gs_OverrideHostWith = sParams[2];
            FiddlerObject.StatusText = "Connecting to [" + gs_OverrideHostWith + "] for requests to [" + gs_OverrideHost + "];";
            return true;
        case "urlreplace":
            if (sParams.Length < 3) { gs_ReplaceToken = null; FiddlerObject.StatusText = "URL Replacement cleared"; return false; }
            gs_ReplaceToken = sParams[1];
            gs_ReplaceTokenWith = sParams[2].Replace(" ", "%20"); // Simple helper
            FiddlerObject.StatusText = "Replacing [" + gs_ReplaceToken + "] in URIs with [" + gs_ReplaceTokenWith + "];";
            return true;
        case "allbut":
        case "keeponly":
            if (sParams.Length < 2) { FiddlerObject.StatusText = "Please specify Content-Type to retain during wipe."; return false; }
            UI.actSelectSessionsWithResponseHeaderValue("Content-Type", sParams[1]);
            UI.actRemoveUnselectedSessions();
            UI.lvSessions.SelectedItems.Clear();
            FiddlerObject.StatusText = "Removed all but Content-Type: " + sParams[1];
            return true;
        case "stop":
            UI.actDetachProxy();
            return true;
        case "start":
            UI.actAttachProxy();
            return true;
        case "cls":
        case "clear":
            UI.actRemoveAllSessions();
            return true;
    }
}

```

<https://docs.telerik.com/fiddler/KnowledgeBase/FiddlerScript/ModifyRequestOrResponse>



NOT BAD

server-replay <https://www.npmjs.com/package/server-replay>

Replay server responses from a **.har file**.

Useful if...

- ...you want to develop offline and your development server isn't local
- ...your development server is very slow and you want to go faster
- ...you are developing against an API with rate limits

It works by starting a proxy server and serving content from a previously saved .har file overlaid with files from your local system, configurable with mappings.

Installation

```
npm install -g server-replay
```

Running

You need to have a `.har` file, run `server-replay`, and then set up your browser to use it as a proxy.

A `.server-replay.json` in the current directory is used by default if no config option is given.

```
server-replay [options] <.har file>
```

Options:

- c, --config The config file to use
- p, --port The port to run the proxy server on [default: 8080]
- d, --debug Turn on debug logging

Install

```
> npm i server-replay
```

± Weekly Downloads

6



Version

1.1.0

License

Apache-2.0

Issues

3

Pull Requests

2

Homepage

github.com/Stuk/server-replay

Repository

github.com/Stuk/server-replay

Last publish

4 years ago

Collaborators



Try on RunKit

Report a vulnerability

https -> http

```
{  
  "version": 1,  
  "replacements": [  
    // Proxy only works over http  
    {"match": "https", "replace": "http"}  
  ]  
}
```


http-proxy-middleware

src/setupProxy.js

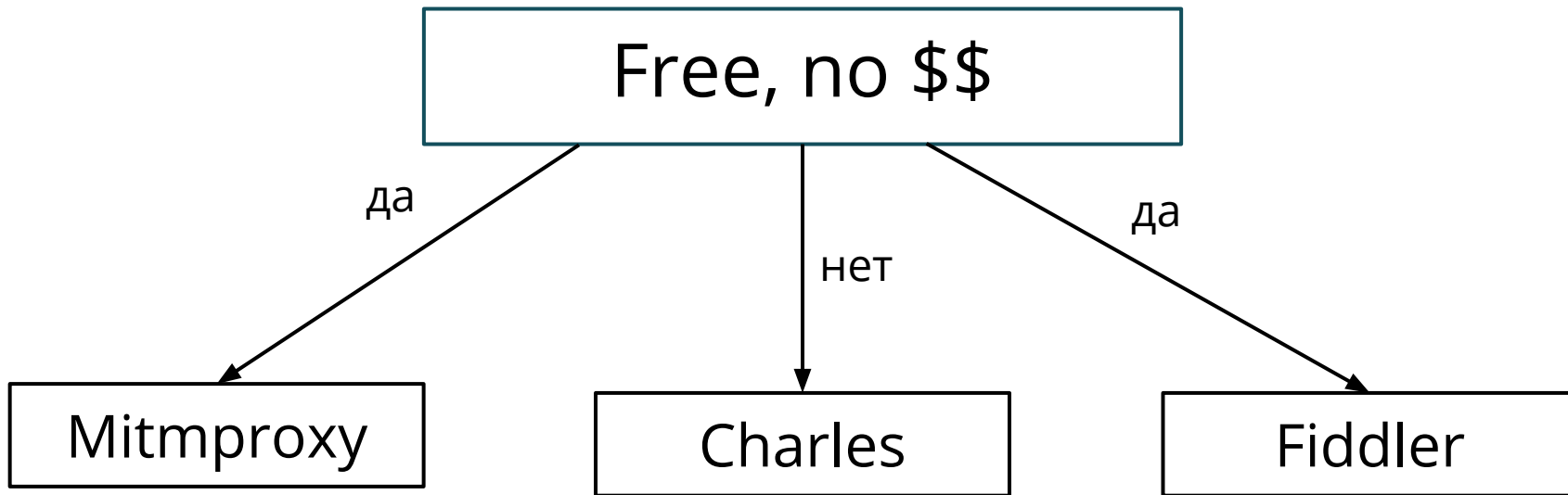
```
const { createProxyMiddleware } = require('http-proxy-middleware');
module.exports = function(app) {
  app.use(
    '/api',
    createProxyMiddleware({
      target: 'http://localhost:5000',
      changeOrigin: true,
    })
  );
};
```

Invalid Host header и др проблемы

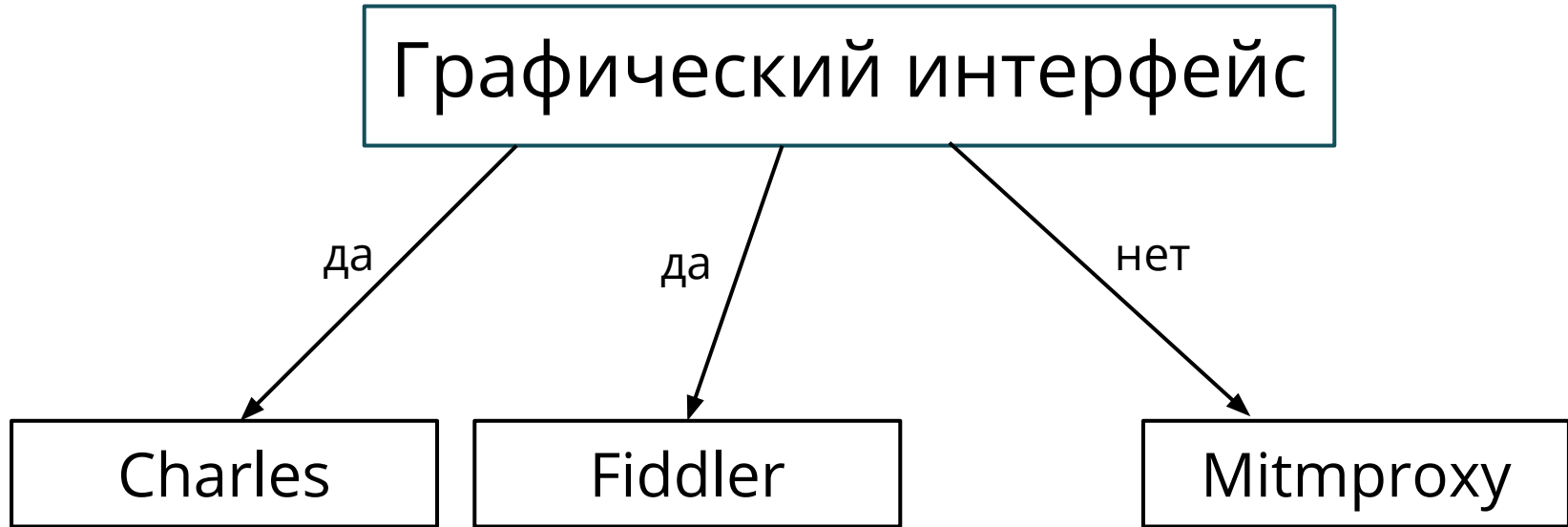
```
.env.development
```

```
HOST=local.devhost.com
```

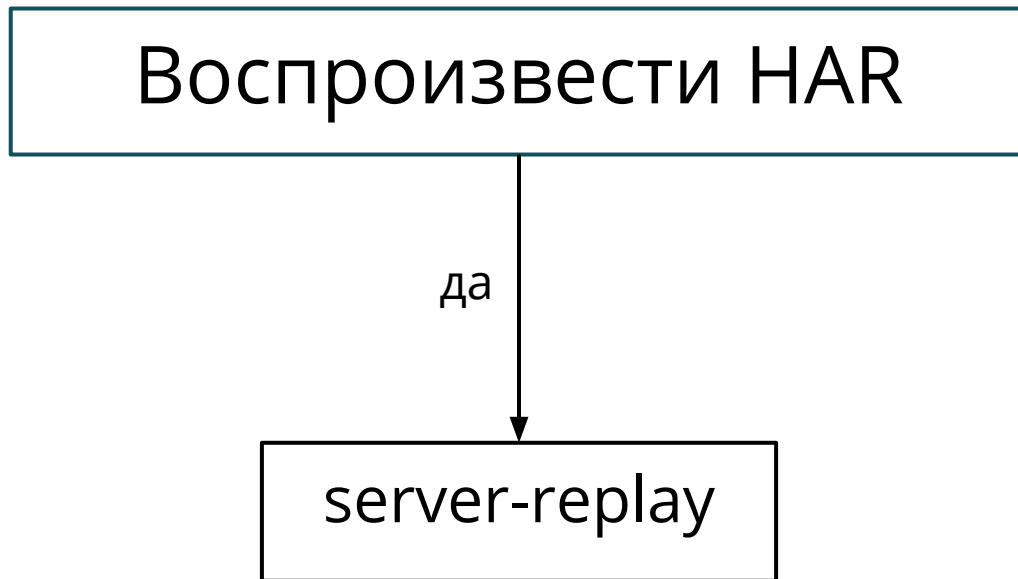
СТОИМОСТЬ



Наличие UI



Ответы из HAR-ника



Проксирование во фреймворке

Просто прокси запросов?

да

`http-proxy-middleware`

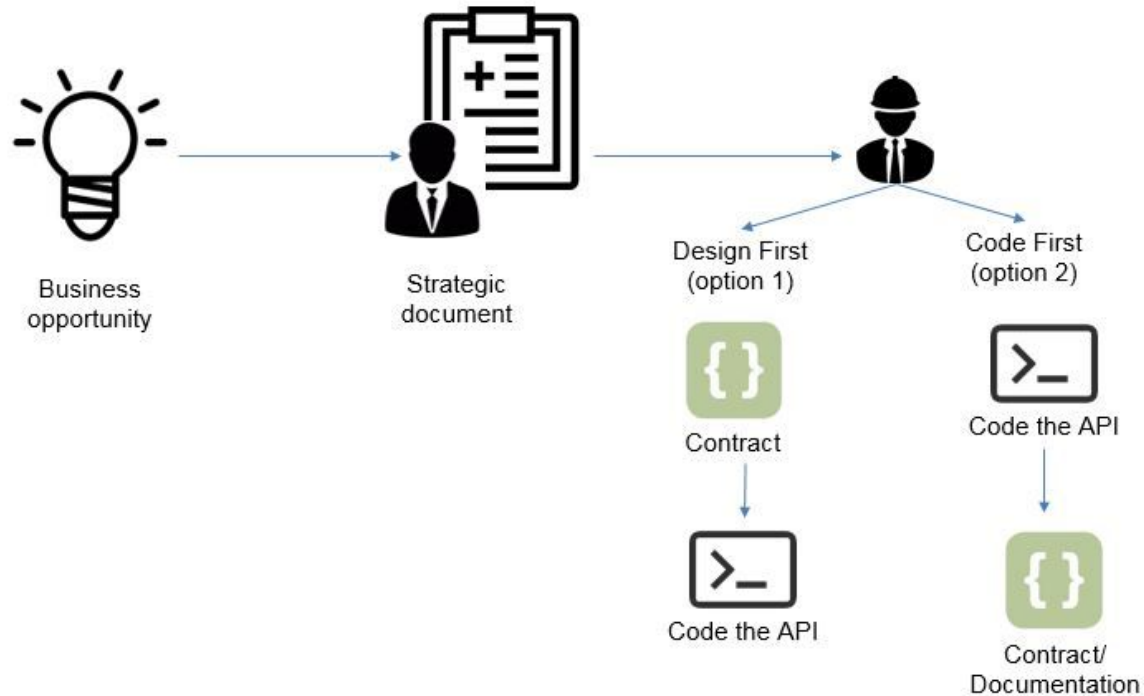
The background of the slide is a black and white photograph of numerous stacks of papers. Each stack is bound with a white paper strip that has a handwritten number and a circled number. Visible labels include '2016 (10)', '2002 (11)', '1992 (11)', and '2001 (11)'. The stacks are arranged in a grid-like pattern, creating a sense of depth and volume.

STUВим по спецификации

Виды спецификаций

- OpenAPI v2/v3;
- API Blueprint;
- RAML

Design First vs Code First









Инструменты для OpenAPI

- Swagger;
- SoapUI;
- Stoplight

Mock Servers

<https://openapi.tools/>

Fake servers that take description document as input, then route incoming HTTP requests to example responses or dynamically generates examples.

Name	Language	v2	v3	GitHub
API Sprout - Lightweight, blazing fast, cross-platform OpenAPI 3 mock server with validation	cli / Docker	✗	✓	
Aptive Studio - A platform for Digital Product Managers and API Consultants to design REST APIs with in-built mock and documentation.	Angular 7.0, Java / Saas	✓	✓	
Connexion - OpenAPI First framework for Python on top of Flask with automatic endpoint validation & OAuth2 support	Python	✓	✓	
Fakeit - Create mock server from OpenAPI 3 specification with random response generation and request validation.	cli / Docker	✗	✓	
Falcon Heavy - The framework for building app backends and microservices via the API design-first workflow.	Python	✗	✓	
Meeshkan - Mock HTTP APIs through a combination of API definitions, recorded traffic and code. Used for sandboxes, as well as automated and exploratory testing.	Python	✗	✓	
Microcks - Mocking and testing platform for API and microservices. Turn your OAI contract examples into ready to use mocks. Use examples to test and validate implementations according schema elements.	Self- hosted / SaaS	✓	✓	

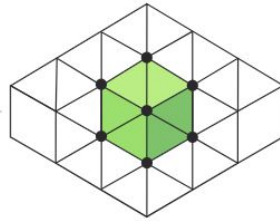
Swagger

<https://swagger.io/>

SwaggerHub

The design and documentation platform for teams and individuals working with the OpenAPI Specification.

Create Free Account



Swagger Editor



Swagger UI



Swagger Codegen

A wide-angle shot of a modern cinema interior. The room is dimly lit, with the primary light source being a large, bright blue screen at the front. The screen displays the word "Swagger" in a large, white, sans-serif font. The walls are a deep blue, and the ceiling is dark with recessed lighting. The floor is a dark, polished surface. The seats are dark grey or black, arranged in neat rows, and are currently empty. On the side walls, there are decorative vertical panels with a textured, golden-brown pattern. A small green exit sign is visible on the right wall near the back of the room.

Swagger

SoapUI


<https://www.soapui.org/>



Stoplight Studio

Design APIs 10x faster with our free OpenAPI editor. Prototype and share your API within minutes.

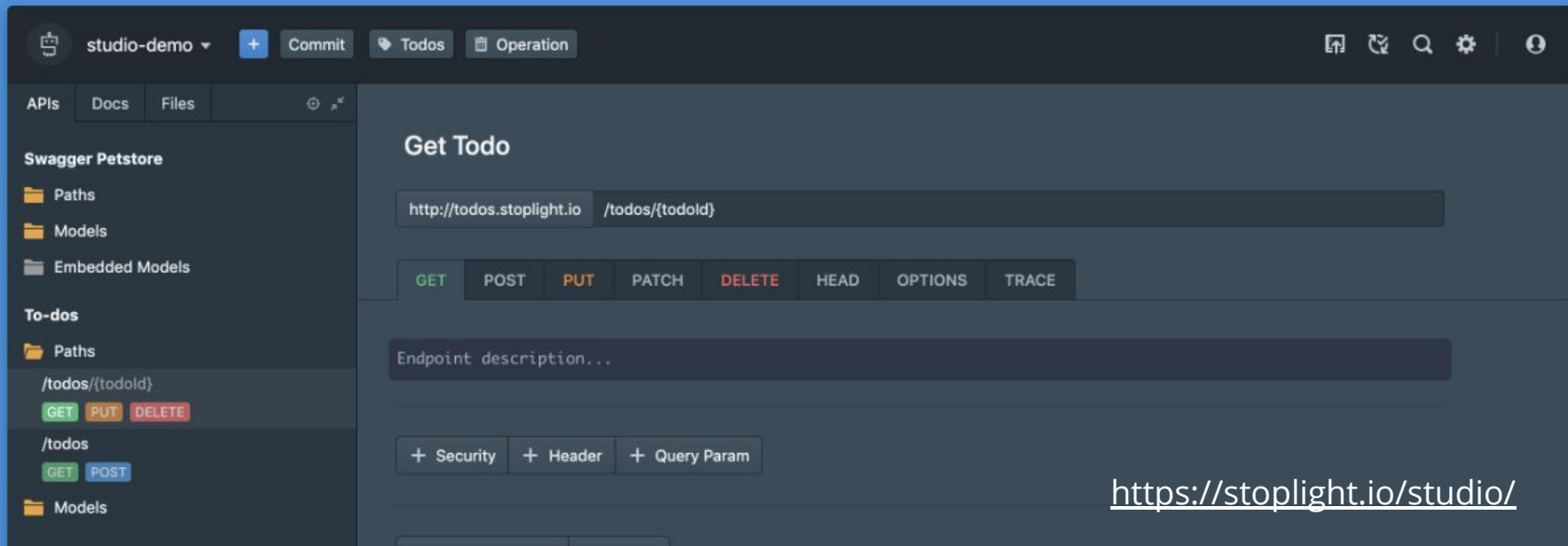
 On the Web

 Mac App

 Windows App

 Linux App

 Read the Docs



The screenshot displays the Stoplight Studio interface for editing an API. The top navigation bar includes a logo, a dropdown menu for 'studio-demo', a '+' button, and buttons for 'Commit', 'Todos', and 'Operation'. The left sidebar shows a tree view for 'Swagger Petstore' with folders for 'Paths', 'Models', and 'Embedded Models'. Under 'Paths', there are two entries: '/todos/{todoid}' with methods GET, PUT, and DELETE, and '/todos' with methods GET and POST. The main workspace is titled 'Get Todo' and shows the URL 'http://todos.stoplight.io /todos/{todoid}'. Below the URL, there are buttons for HTTP methods: GET (selected), POST, PUT, PATCH, DELETE, HEAD, OPTIONS, and TRACE. A text area below the methods contains the placeholder text 'Endpoint description...'. At the bottom, there are buttons for '+ Security', '+ Header', and '+ Query Param'. The bottom right corner of the image contains the URL <https://stoplight.io/studio/>.

Prism, an Open-Source HTTP Mock & Proxy Server

Accelerate API development with realistic mock servers, powered by OpenAPI documents.

[GitHub](#)[Watch the Video](#)[Read the Docs](#)[✔ Quick Iterations](#)[✔ Dynamic Examples](#)[✔ Validation](#)[✔ Callbacks](#)[✔ Proxy](#)

Work with your API before you write any code

Prism is an open-source HTTP mock server that can mimic your API's behavior as if you already built it. Mock HTTP servers are generated from your OpenAPI v2/v3 (formerly known as Swagger) documents.

Stoplight & Prism



Swagger to ts



```
npx @manifoldco/swagger-to-ts schema.yaml --output schema.ts
```

<https://github.com/manifoldco/swagger-to-ts>

Спецификация API Blueprint

<https://apiblueprint.org/tools.html>

```
26 Lines (20 sloc) | 1.08 KB
```

Raw Blame History

FORMAT: 1A

The Simplest API

This is one of the simplest APIs written in the **API Blueprint**. One plain resource combined with a method and that's it! We will explain what is going on in the next installment - [Resource and Actions](#).

Note: As we progress through the examples, do not also forget to view the [Raw](#) code to see what is really going on in the API Blueprint, as opposed to just seeing the output of the Github Markdown parser.

Also please keep in mind that every single example in this course is a **real API Blueprint** and as such you can **parse** it with the [API Blueprint parser](#) or one of its [bindings](#).

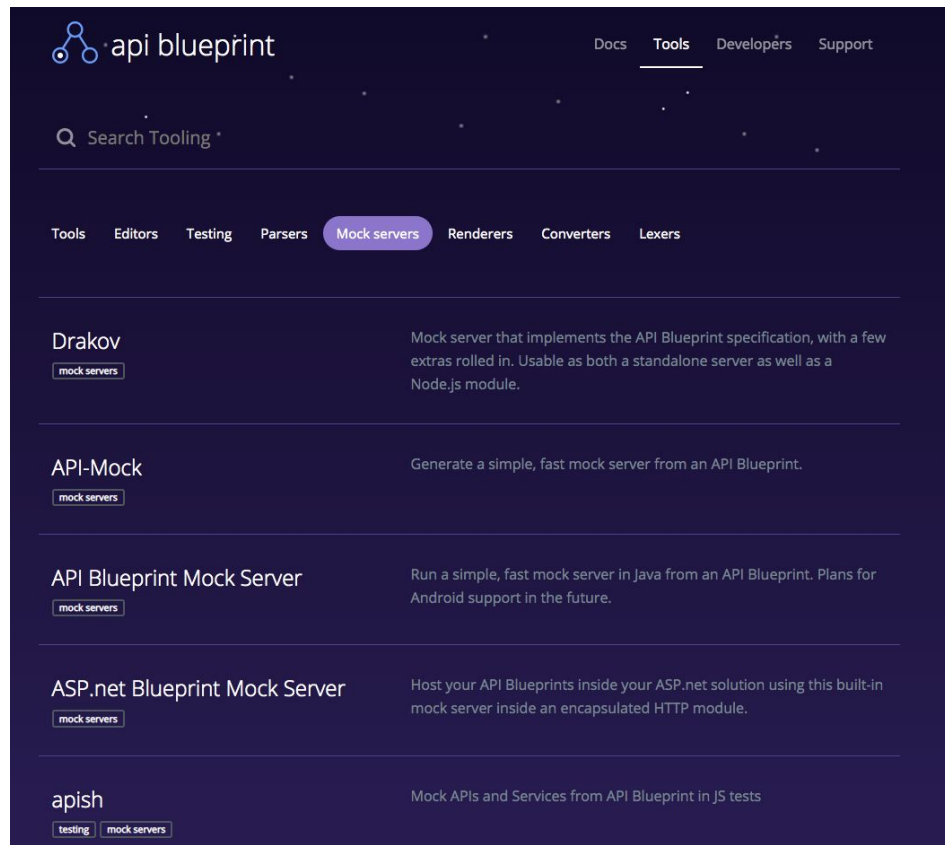
API Blueprint

- This: [Raw API Blueprint](#)
- Next: [Resource and Actions](#)

GET /message

- Response 200 (text/plain)

```
Hello World!
```



The screenshot shows the 'Tools' page of the API Blueprint website. The header includes the API Blueprint logo, navigation links for 'Docs', 'Tools' (which is active), 'Developers', and 'Support', and a search bar. Below the header is a horizontal menu with categories: 'Tools', 'Editors', 'Testing', 'Parsers', 'Mock servers' (highlighted), 'Renderers', 'Converters', and 'Lexers'. The main content area lists several tools:

- Drakov**: Mock server that implements the API Blueprint specification, with a few extras rolled in. Usable as both a standalone server as well as a Node.js module. (tagged 'mock servers')
- API-Mock**: Generate a simple, fast mock server from an API Blueprint. (tagged 'mock servers')
- API Blueprint Mock Server**: Run a simple, fast mock server in Java from an API Blueprint. Plans for Android support in the future. (tagged 'mock servers')
- ASP.net Blueprint Mock Server**: Host your API Blueprints inside your ASP.net solution using this built-in mock server inside an encapsulated HTTP module. (tagged 'mock servers')
- apish**: Mock APIs and Services from API Blueprint in JS tests. (tagged 'testing' and 'mock servers')

Плюсы спецификаций

Есть файл со спецификацией ==
есть мок-сервер

Минусы спецификаций

Спецификация может быть
необновленной => неактуальный
мок-сервер



Спецификация!

Fake your data



JS-библиотеки для фейковых данных

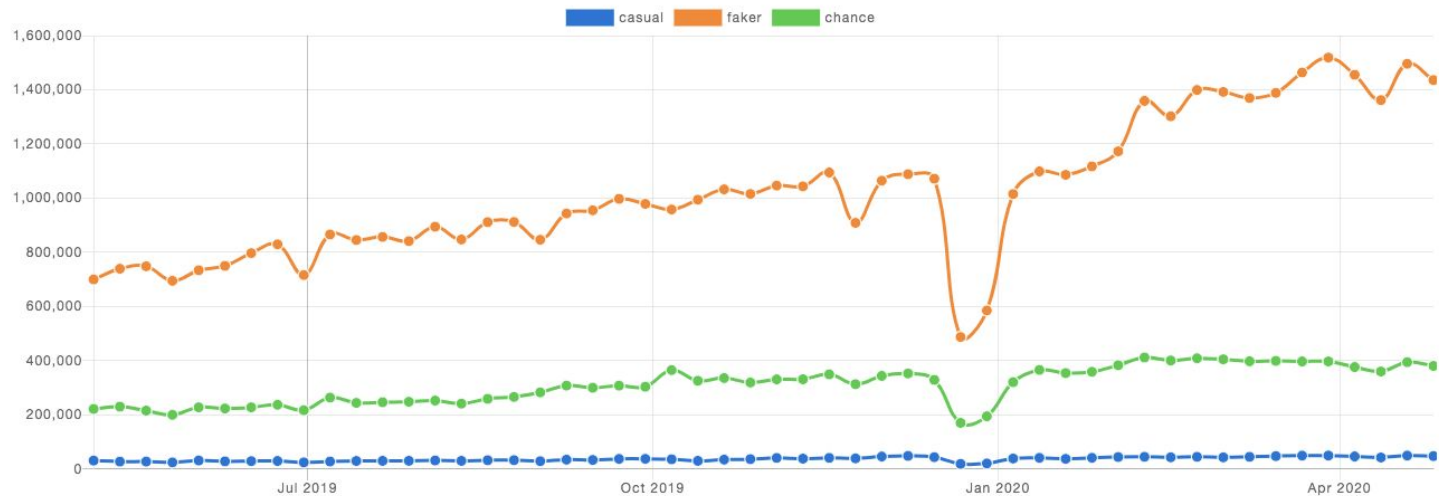
- Faker.js <https://github.com/Marak/faker.js>;
- Chancejs <https://chancejs.com/index.html>;
- Casual <https://github.com/boo1ean/casual>;

Пример использования Faker.js

```
users.push({  
  id: faker.random.uuid(),  
  fullName: faker.name.findName(),  
  email: faker.internet.email(),  
  country: faker.address.country(),  
  phoneNumber: `+7${faker.phone.phoneNumber()}`  
});
```

Сравнение fake data библиотек

Downloads in past 1 Year



Stats

	stars 🌟	forks 🍴	issues 🚨	updated 🛠	created 🗓	size 📦
casual	2,519	143	33	Mar 23, 2020	Jan 31, 2014	minzipped size 178.2 KB
faker	23,418	1,933	321	May 6, 2020	May 15, 2010	minzipped size 321.1 KB
chance	5,115	363	120	Apr 30, 2020	Jun 21, 2013	minzipped size 67.9 KB

ГОТОВИМ СТУВЫ



SinonJs FakeServer

```
const fakeServerWrapper = {
  init: function() {
    faker.locale = "ru"; //Устанавливаем локаль

    // Создаем сервер
    this.fs = sinon.createFakeServer({autoRespond: true});
    this.fs.xhr.useFilters = true;

    // Если фильтр возвратит true, запрос не будет сэмулирован,
    // а пойдет на настоящий сервер
    this.fs.xhr.addFilter(
      function(method, url, async, username, password) {
        return (new RegExp(externalHost)).test(url);
      });

    this.fs.respondWith("GET", url,
      (xhr) => {
        const users = generateUsers(3);
        xhr.respond(200, //Устанавливаем любой статус код ответа
          { "Content-Type": "application/json" }, JSON.stringify(users));
      }
    );
  },

  restore: function() {
    this.fs.restore();
  }
};
```

<https://sinonjs.org/releases/latest/fake-xhr-and-server/>

Поддержка Fetch API

Поддерживает только **XMLHttpRequest** объекты. Но можно использовать фейковый Fetch.

<https://www.npmjs.com/package/fake-fetch>

FakeServer



JSON Server

<https://github.com/typicode/json-server>

- Создаем файл `db.json`
- Запускаем сервер
`json-server --watch db.json`
- Дергаем, например,
<http://localhost:3000/comments/1>

• Получаем в ответе:
`{ "id": 1, "body": "Some new comment", "postId": 1 }`

db.json

```
{
  "posts": [
    {
      "id": 1,
      "title": "json-server",
      "author": "typicode"
    }
  ],
  "comments": [
    {
      "id": 1,
      "body": "Some new comment",
      "postId": 1
    }
  ],
  "profile": {
    "name": "Mary"
  }
}
```

Запуск JSON Server

`node server.js` - запустить сервер.

`npm nodemon server.js` - запустить

сервер со слежением за изменением
файлов.


```
const server = jsonServer.create();
```

```
const router = jsonServer.router('data.json');
```

```
// Добавить кастомные пути в роутер (/operations?accountNumber={number})  
server.get('/operations', (req, res) => {  
  res.jsonp(getOperations(req.query.accountNumber))  
});
```

```
// Добавить правила в роутер  
server.use(jsonServer.rewriter({  
  '/api/*': '/$1',  
}));
```

```
server.use(router);
```

```
server.use((req, res, next) => {  
  if (isAuthorized(req)) { // Добавить свою логику авторизации  
    next() // Продолжаем  
  } else {  
    res.sendStatus(401);  
  }  
});
```

```
server.post('/messages', (req, res, next) => {  
  const userId = req.body['senderId'];  
  // Проверим, передан ли senderId  
  if (userId) {  
    req.body.createdAt = moment().format();  
  } else {  
    // Проставить свой статус код ответа  
    res.status(400).jsonp({  
      error: "No valid senderId"  
    });  
  }  
  next();  
});
```

My JSON Server

Fake Online REST server for teams

Create a **JSON file** on **GitHub**

Get **instantly** a **fake server**



github.com/user/repo/master/db.json

```
{
  "posts": [
    {
      "id": 1,
      "title": "hello"
    }
  ],
  "profile": {
    "name": "typicode"
  }
}
```



my-json-server.typicode.com/user/repo/posts/1

```
{
  "id": 1,
  "title": "hello"
}
```

<https://my-json-server.typicode.com>

JSON Server



mockoon

The screenshot displays the Mockoon application interface. On the left sidebar, there is a list of routes under the 'Example' environment (0.0.0.0:3001):

- /users** (GET) - Get all users
- /users/id** (POST) - Create a user
- /users/id** (PUT) - Update a user
- /users/id** (DELETE) - Delete a user

The main panel shows the configuration for the **GET /users** route. The status is set to **200 - OK**. The response body is a JSON object with the following structure:

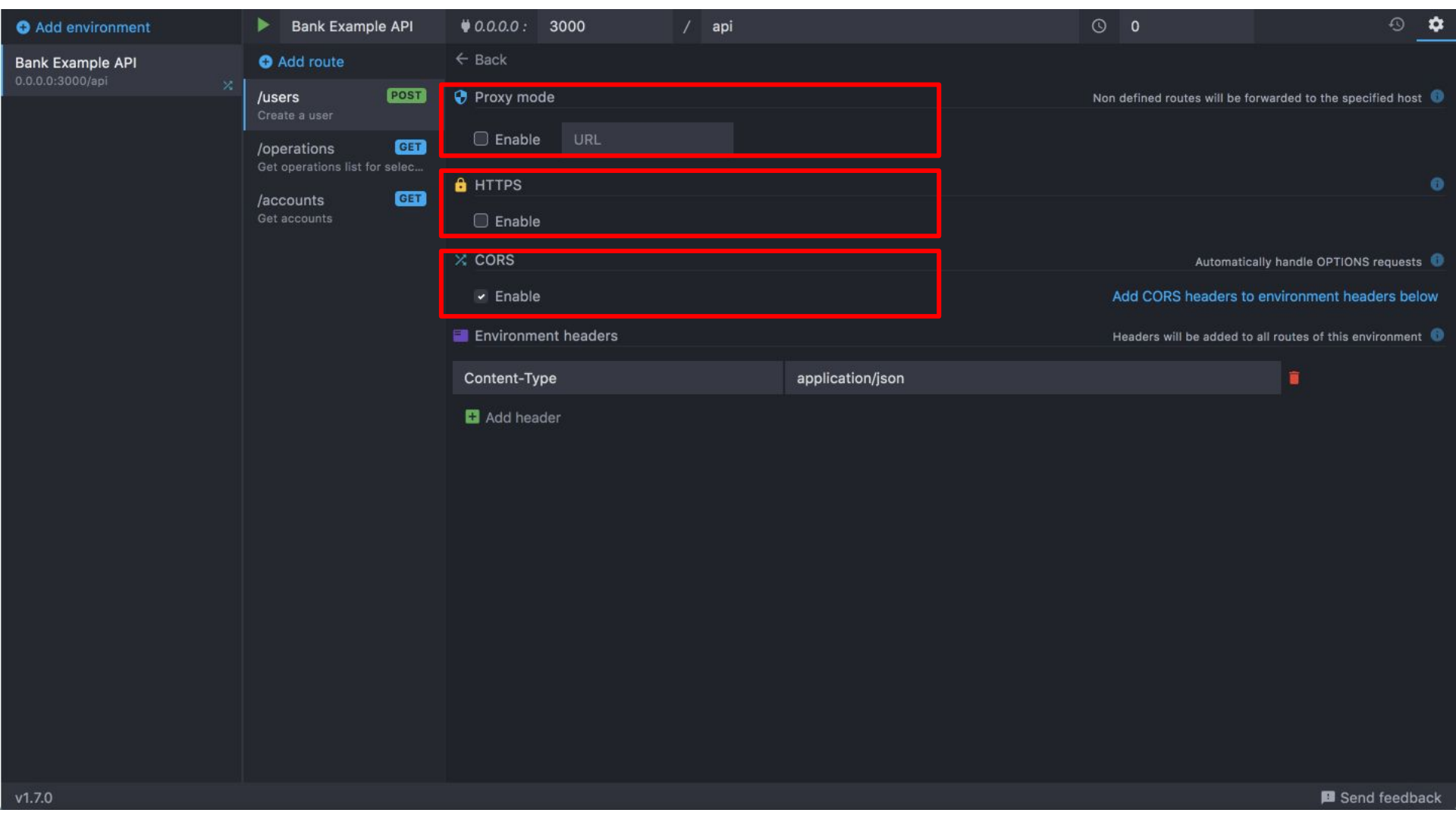
```
1- {
2   "userId": "{{ urlParam 'id' }}",
3   "name": "{{ queryParam 'name' 'John' }}",
4   "lang": "{{ header 'Accept-Language' 'en' }}",
5   "elementTitle": "{{ body 'elements.0.title' 'default' }}",
6   "ip": "{{ ip }}",
7   "method": "{{ method }}",
8   "hostname": "{{ hostname }}",
9   "friends": [
10    {{#repeat 2}}
11    {
12      "id": {{@index}},
13      "name": "{{ firstName }} {{ lastName }}"
14    }
15    {{/ repeat}}
16  ],
17  "oneItem": "{{ oneOf (array 'item1' 'item2' 'item3') }}",
18  "someItems": "{{ someOf (array 'item1' 'item2' 'item3') 1 2 }}",
19  "userName":
20    {{#switch (urlParam 'id')}}
21    {{#case "1"}}John{{/case}}
22    {{#case "2"}}Jack{{/case}}
23    {{#default}}Peter{{/default }}
24    {{/switch}}
25 }
```

<https://mockoon.com/>

A wide-angle shot of a modern cinema interior. The room is dimly lit, with the primary light source being a large, bright blue screen at the front. The screen displays the word "Mockoon" in a large, white, sans-serif font. The walls are a deep blue, and the ceiling is dark with recessed lighting. On either side of the screen, there are vertical panels with a textured, golden-brown pattern. The floor is dark, and the overall atmosphere is clean and contemporary.

Mockoon

```
{{#repeat 2}}
{
  "date": "{{date '2020-06-14' '2020-06-10' 'YYYY-MM-DD'}}",
  "currency": "810",
  "documents": [
    {{#repeat 3}}
    {
      "documentId": "{{guid}}",
      "documentNum": "{{int 0 1000}}",
      "amount": "-{{float 0 100000 round=0.01}}",
      "paymentDesc": "{{lorem}}",
      "accountNumber": "{{queryParam 'accountNumber'}}"
    }
    {{/repeat}}
  ]
}
{{/repeat}}
```

➕ Add environment

▶ Bank Example API

🌐 0.0.0.0 : 3000

/ api

🕒 0

🔄 ⚙️

Bank Example API

0.0.0.0:3000/api

➕ Add route

← Back

/users

POST

Create a user

🛡️ Proxy mode

Enable

URL

Non defined routes will be forwarded to the specified host ⓘ

/operations

GET

Get operations list for selec...

🔒 HTTPS

Enable

ⓘ

/accounts

GET

Get accounts

✕ CORS

Enable

Automatically handle OPTIONS requests ⓘ

Add CORS headers to environment headers below

📁 Environment headers

Headers will be added to all routes of this environment ⓘ

Content-Type

application/json

➕ Add header

Есть импорт, нет синхронизации



Где же баг?



Error while parsing JSON during the
call



mockoon

Стабильность прямо на фронте

<https://clck.ru/Md3oX>

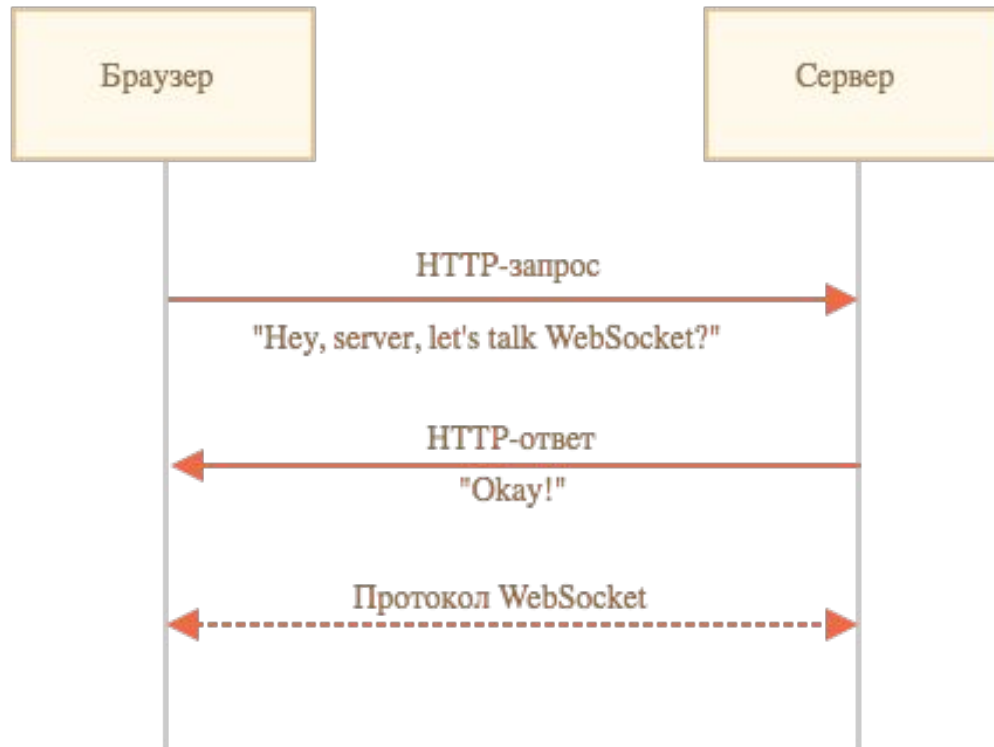


Стаб на фронте из практики

```
export const someMethod = (params = {}) => {  
  if (useStub) {  
    return fetchStub('someMethod');  
  }  
  ...  
}
```

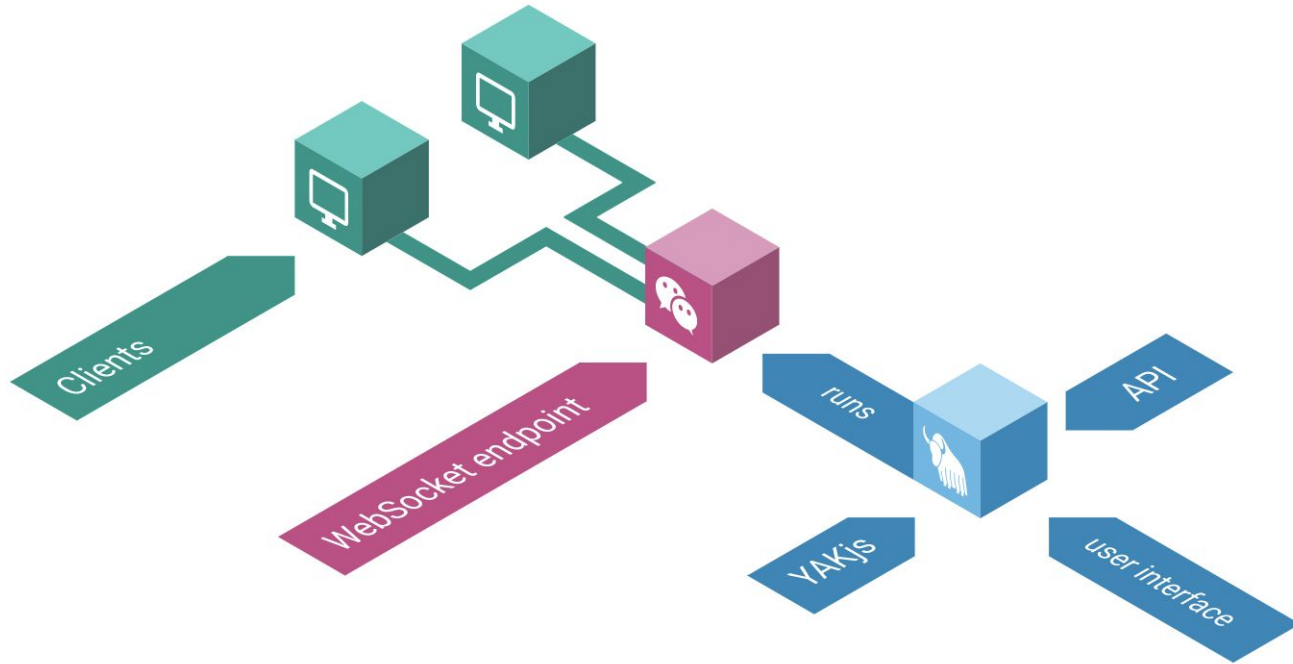
```
export function* fetchStub(methodName) {  
  yield wait(1000);  
  return data[methodName];  
}
```

WebSocket



YAKjs

<http://www.yakjs.com/>



A wide-angle shot of a modern cinema interior. The room is dimly lit, with the primary light source being a large, bright blue screen at the front. The screen displays the text 'YAKjs' in a bold, white, sans-serif font. The walls are a deep blue, and the ceiling is dark with recessed lighting. On either side of the screen, there are vertical panels with a textured, golden-brown pattern. The floor is dark, and the overall atmosphere is clean and contemporary.

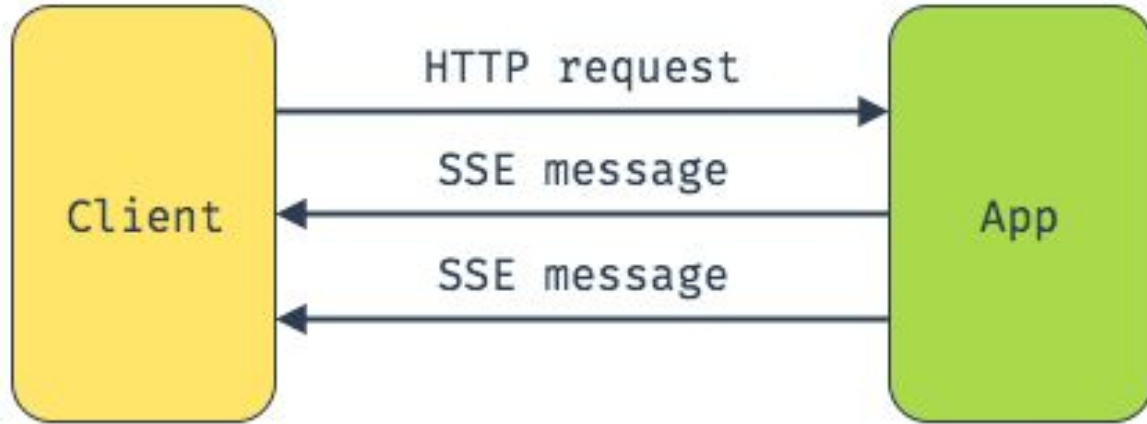
YAKjs

И здесь тоже нет синхронизации





Server-Sent Events



local-web-server

Инструмент на все случаи
ЖИЗНИ:

- REST API,
- WebSocket,
- Server Sent Events service

SSE

```
{
  route: '/notifications',
  responses: [
    {
      response: async function (ctx) {
        ctx.body = new require('stream').PassThrough();
        ctx.type = 'text/event-stream';
        ctx.set('Cache-Control', 'no-cache');
        ctx.set('Connection', 'keep-alive');

        const interval = setInterval(() => {
          ++i;
          ctx.body.write(`event: message\n`);
          ctx.body.write(`data: ${JSON.stringify(getNotification(i))}\n\n`);
        }, 3000);

        function finished () {
          clearInterval(interval);
          ctx.body.end();
        }

        ctx.req.on('close', finished);
        ctx.req.on('finish', finished);
        ctx.req.on('error', finished);
      }
    }
  ]
}
```

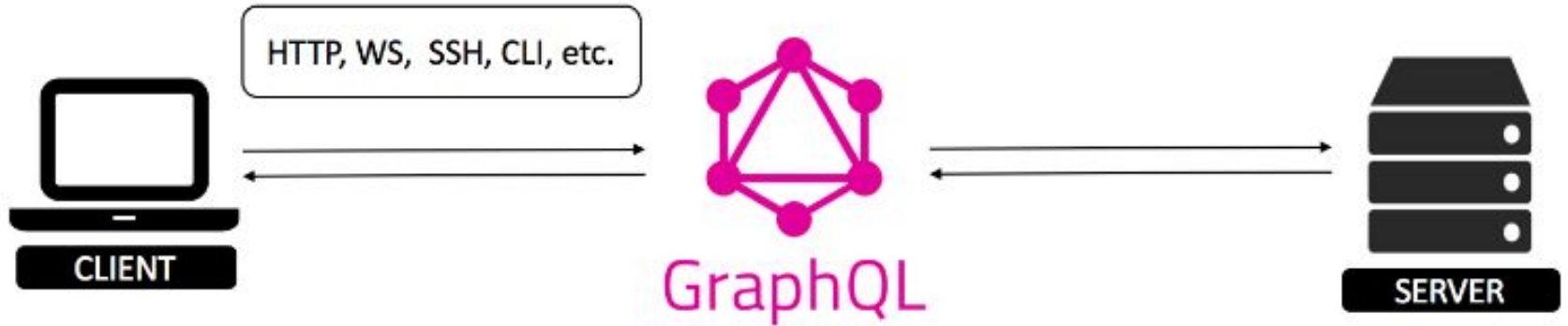
WebSocket

```
class Websocket {  
  middleware (config, lws) {  
    const WebSocket = require('ws');  
    const wss = new WebSocket.Server({ server: lws.server });  
  
    wss.on('connection', socket => {  
      socket.on('message', message => {  
        socket.send(`"${message}"? Yep, that's right`);  
      })  
    })  
  }  
}  
  
module.exports = Websocket;
```

REST API

```
const router = require('koa-route');
return [
  router.get('/users', function (ctx) {
    if (isAuthorized()) {
      ctx.response.type = 'json';
      ctx.response.body = users;
    } else {
      ctx.response.status = 401;
    }
  }),
  router.post('/users', function (ctx) {
    const newUser = ctx.request.body;
    users.push(newUser);
    newUser.id = users.length;
    ctx.response.status = 201;
    ctx.response.set('Location', `/users/${newUser.id}`);
  }),
  router.put('/users', function (ctx) {
    ctx.response.status = 405;
  }),
  router.get('/users/:id', function (ctx, id) {
    ctx.response.type = 'json';
    ctx.response.body = users.find(user => user.id === Number(id));
  })
]
```

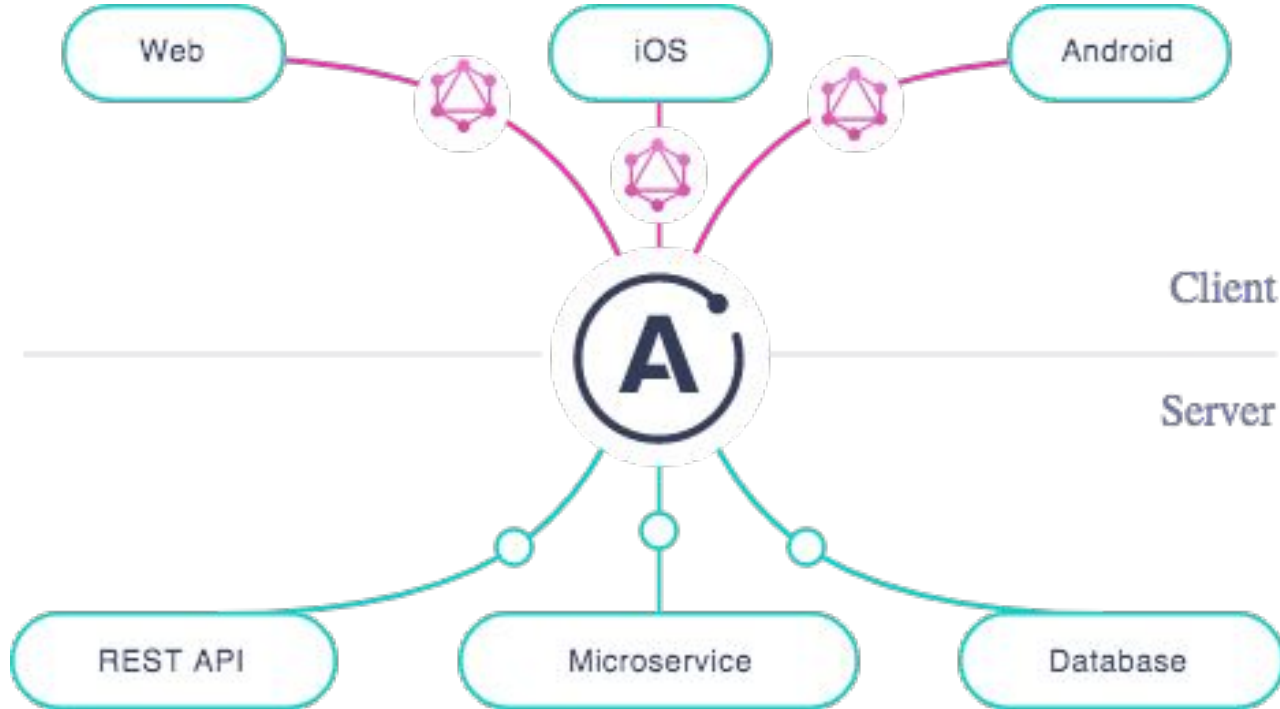

GraphQL



„GraphQL на проде похож
на привидение: все о нем
говорят, но мало кто
видел.“

— Франсуа де Ларошфуко, «Максимы и моральные
размышления»

Apollo Server



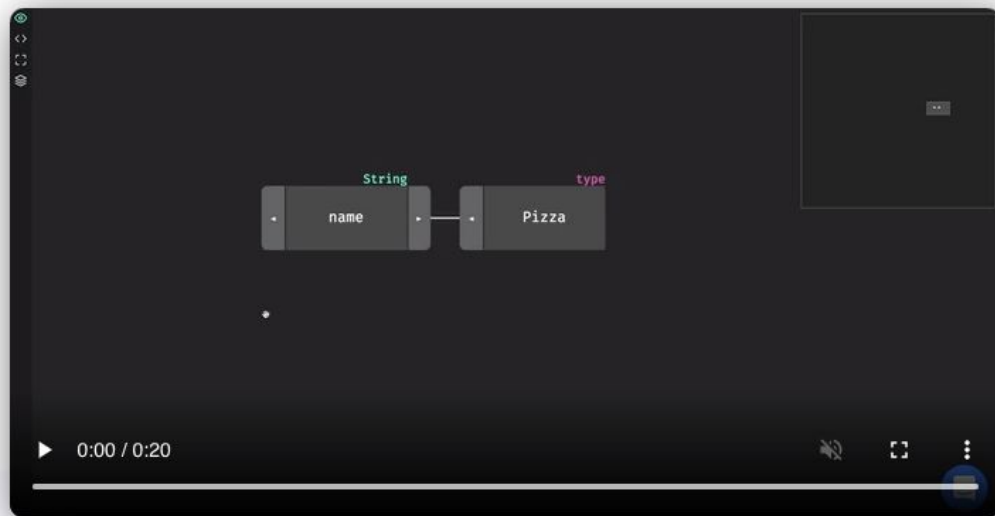
Apollo Server

The image shows a wide-angle view of a modern, empty theater. The seats are black and arranged in neat rows, receding towards a large, bright blue screen at the front. The screen displays the text 'Apollo Server' in a clean, white, sans-serif font. The theater's walls are a deep blue, and the side walls feature vertical panels with a textured, golden-brown pattern. The ceiling is dark with recessed lighting. The overall atmosphere is clean and professional.

A Better Way to GraphQL

Take your GraphQL schema to the **next level**

Start now

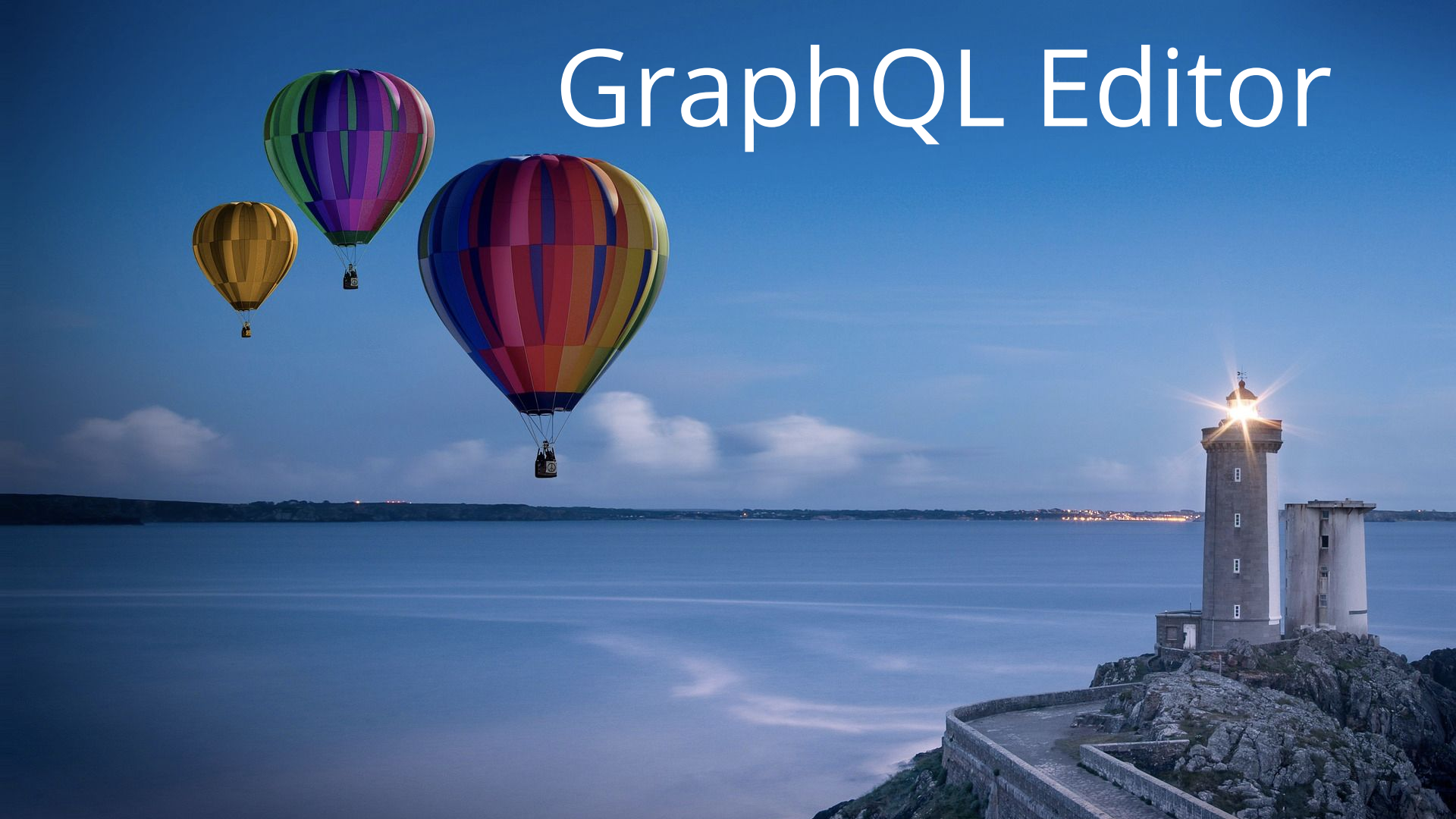


<https://graphqleditor.com/>

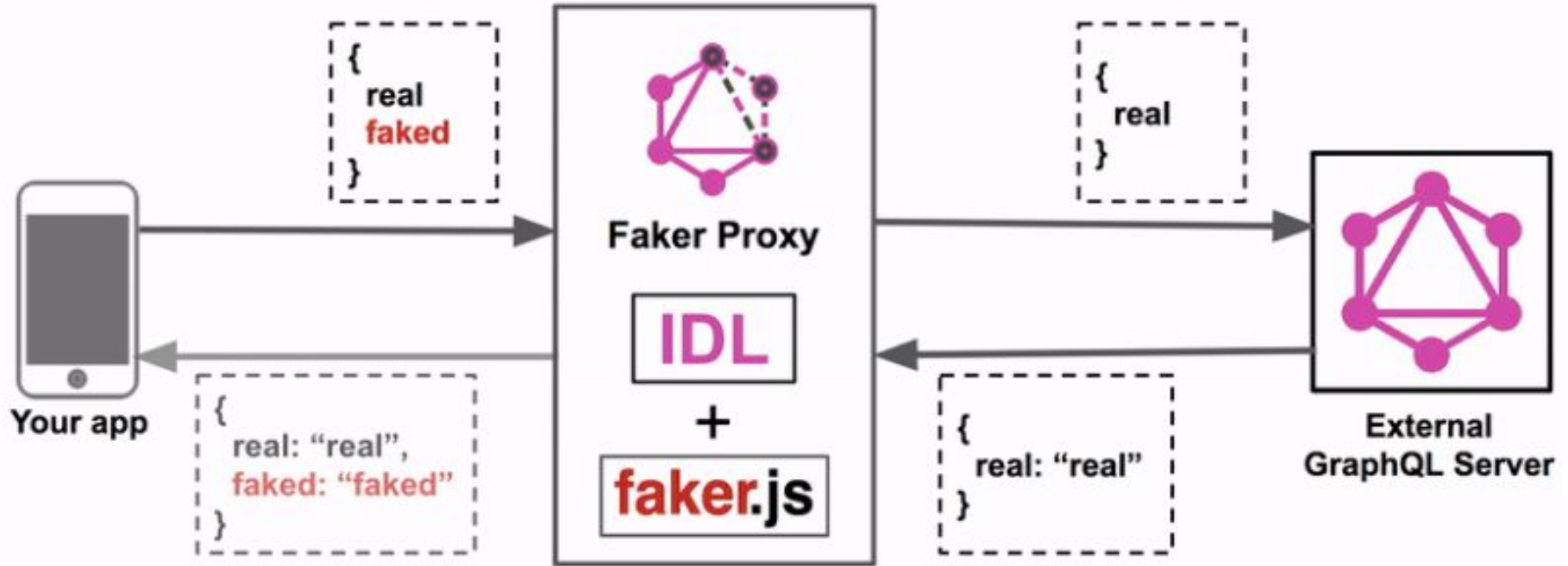
GraphQL Editor

A 3D rendering of a movie theater. The room is dimly lit with blue walls and gold-colored decorative panels on the side walls. The ceiling has recessed lighting. The floor is dark. The seats are black and arranged in rows, facing a large white screen. The screen displays the text 'GraphQL Editor' in white. There are some small details like a fire extinguisher and a door on the right wall.

GraphQL Editor



GraphQL-faker



<https://github.com/APIs-guru/graphql-faker>



```
7 #
8 # Press save or Cmd+Enter to apply the changes and update server. Switch to GraphQL
9 # on the left panel to immediately test your changes.
10 # This tool also supports extending existing APIs. Check graphql-faker --help
11 #
12 # Developed with ❤ by APIs.guru | https://github.com/APIs-guru/graphql-faker
13
14 type Company {
15   name: String @fake(type:companyName)
16   industry: String @examples(values: ["IT", "Manufacturing", "Medicine", "Media"])
17   employees: [Employee!] @listLength(min: 5, max: 10)
18 }
19
20 type Employee {
21   firstName: String @fake(type: firstName, locale:en_CA)
22   lastName: String @fake(type: lastName, locale:en_CA)
23   address: String @fake(type:streetAddress, options: { useFullAddress: true })
24   subordinates: [Employee!] @listLength(min: 0, max: 3)
25   company: Company
26 }
27
28 type Query {
29   employee(id: ID): Employee
30   company(id: ID): Company
31   allCompanies: [Company!]
32 }
33
```

SAVE

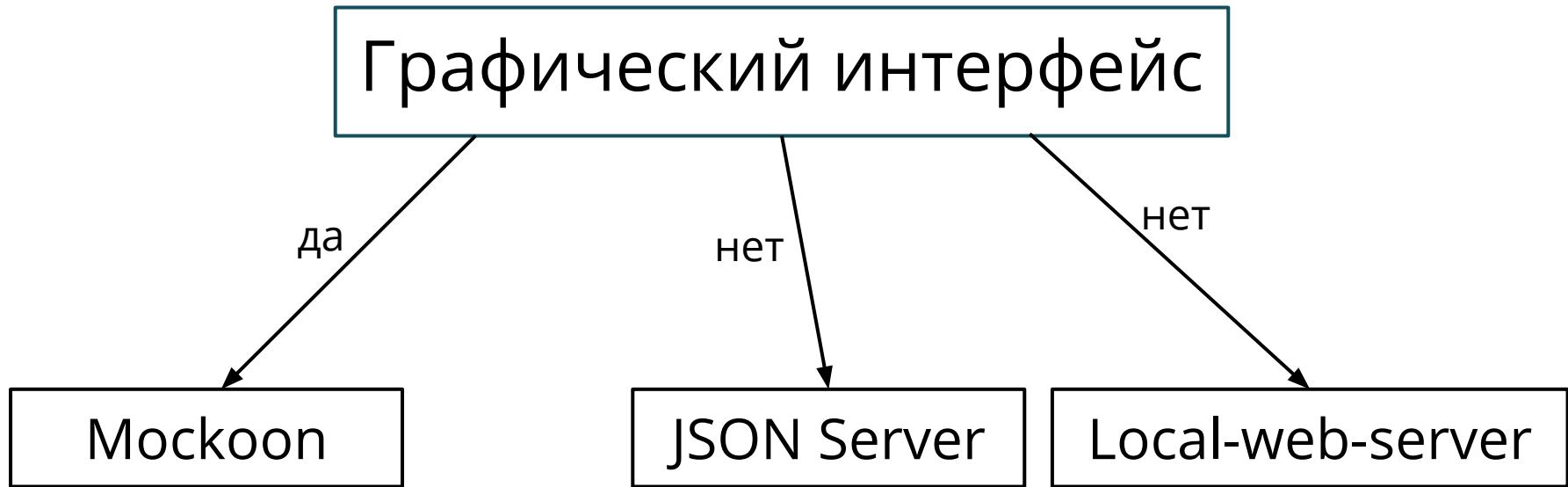


GraphQL-faker

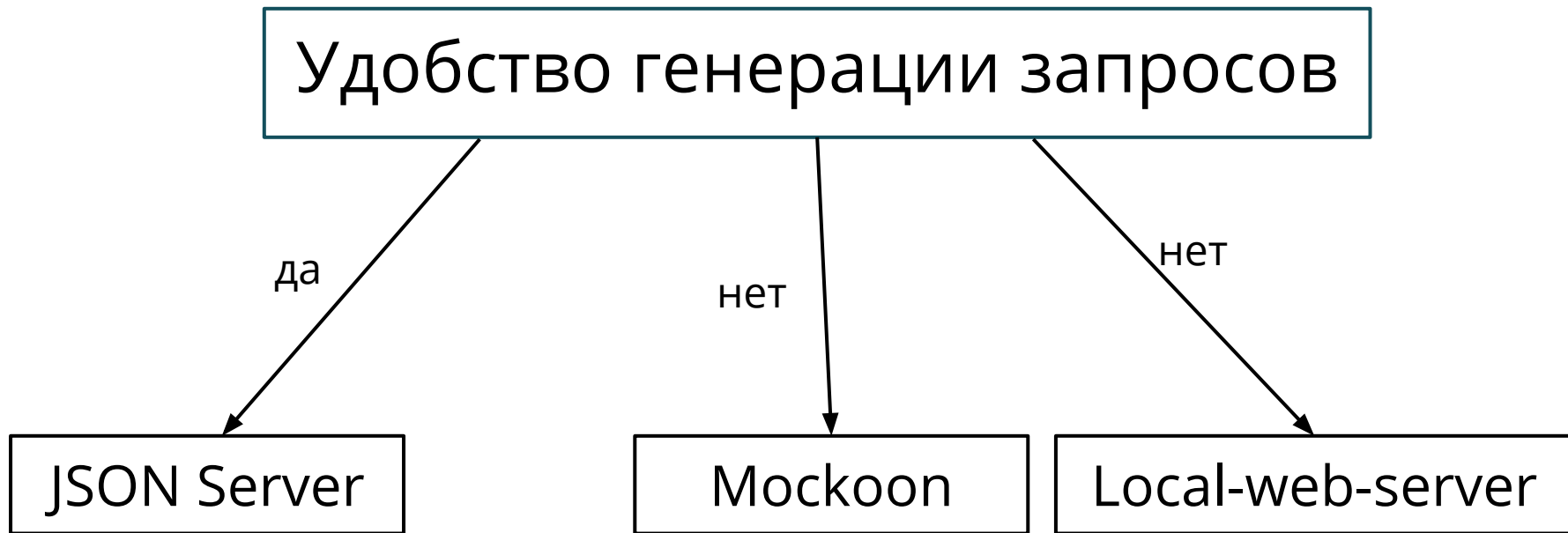
- Более 60 видов различных типов данных
- Множество локалей
- Автозаполнение
- Сохранение обновленной схемы в файл

ВЫВОДЫ

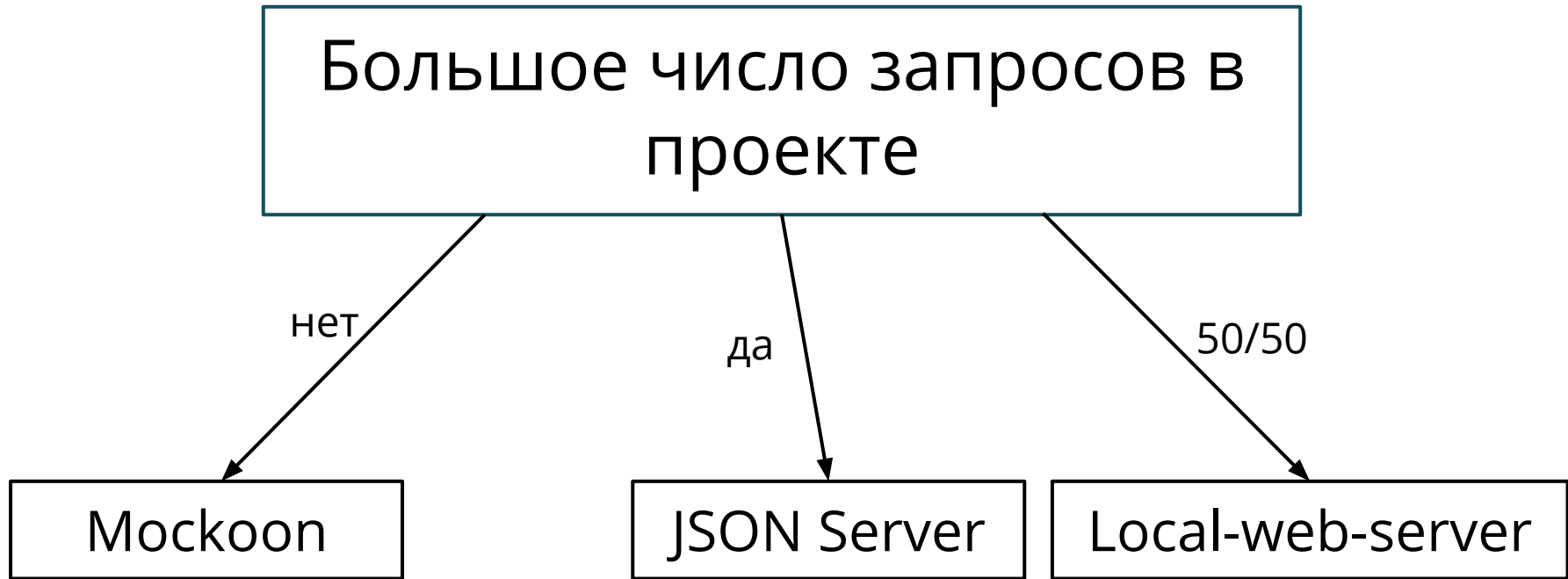
REST API



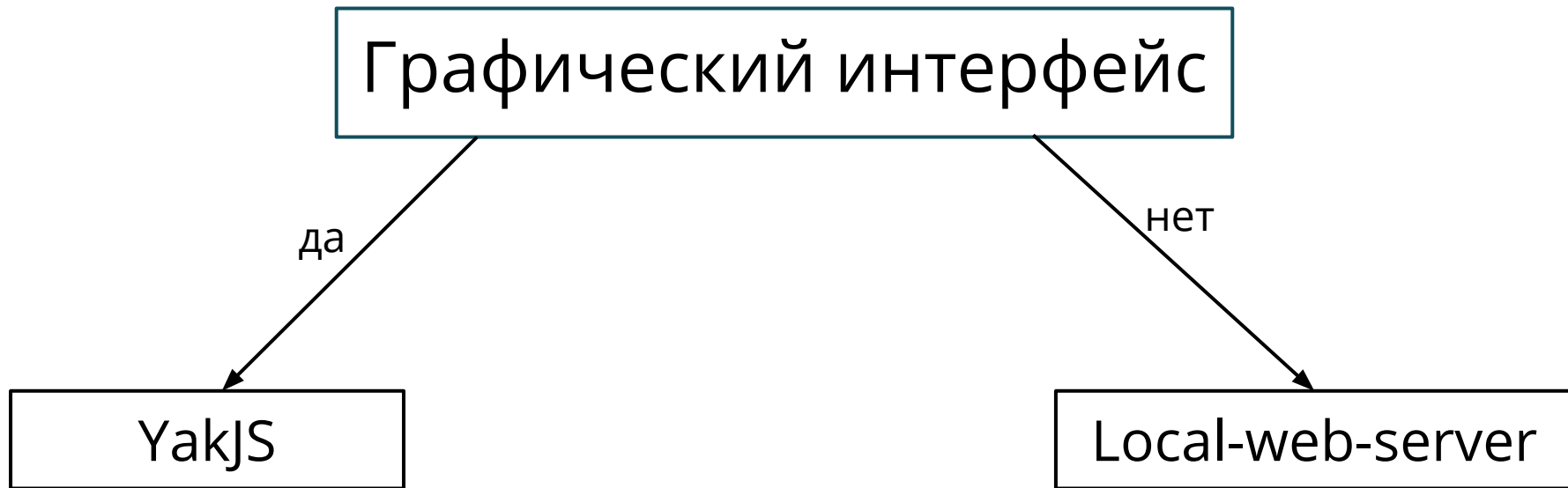
REST API



REST API



WebSocket



WebSocket

Разработка в IDE

```
graph TD; A[Разработка в IDE] -- да --> B[Local-web-server]; A -- нет --> C[YakJS];
```

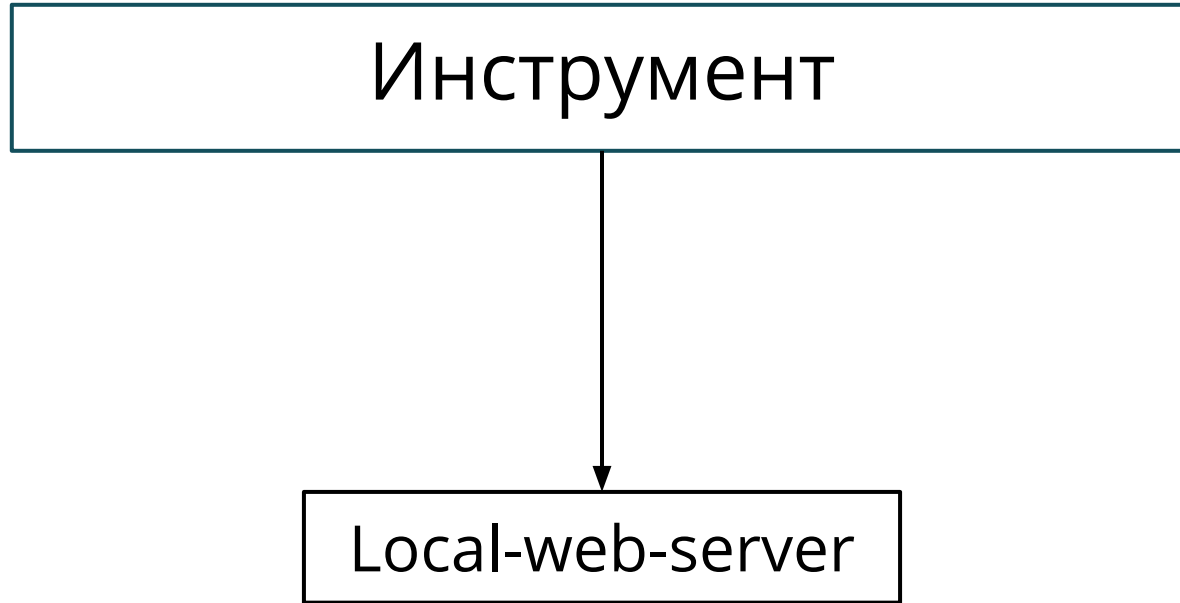
да

Local-web-server

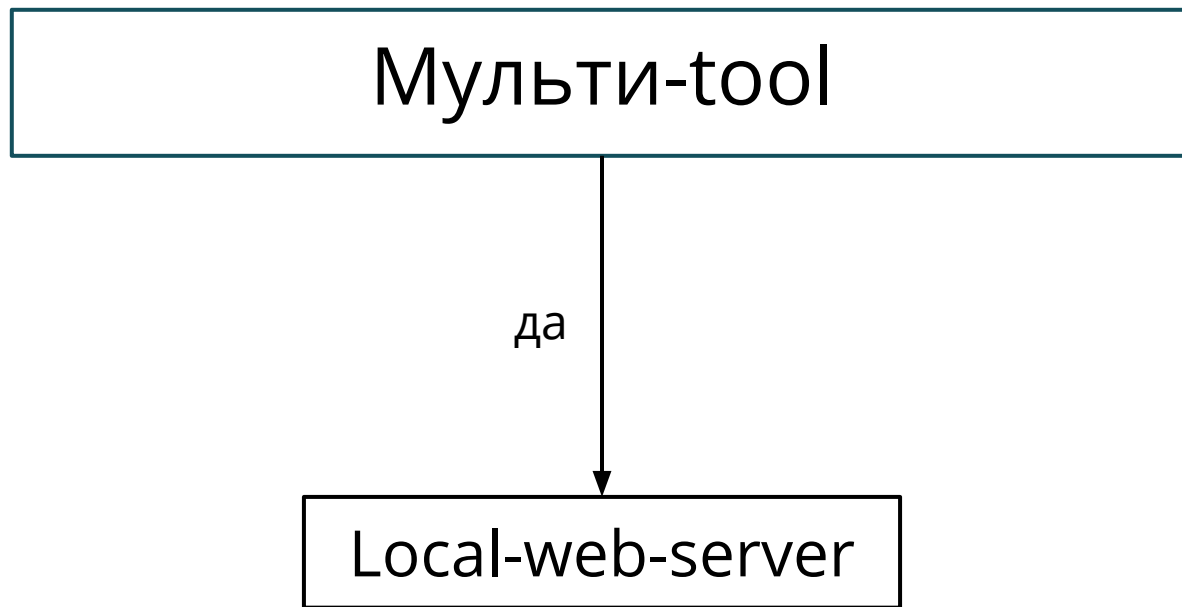
нет

YakJS

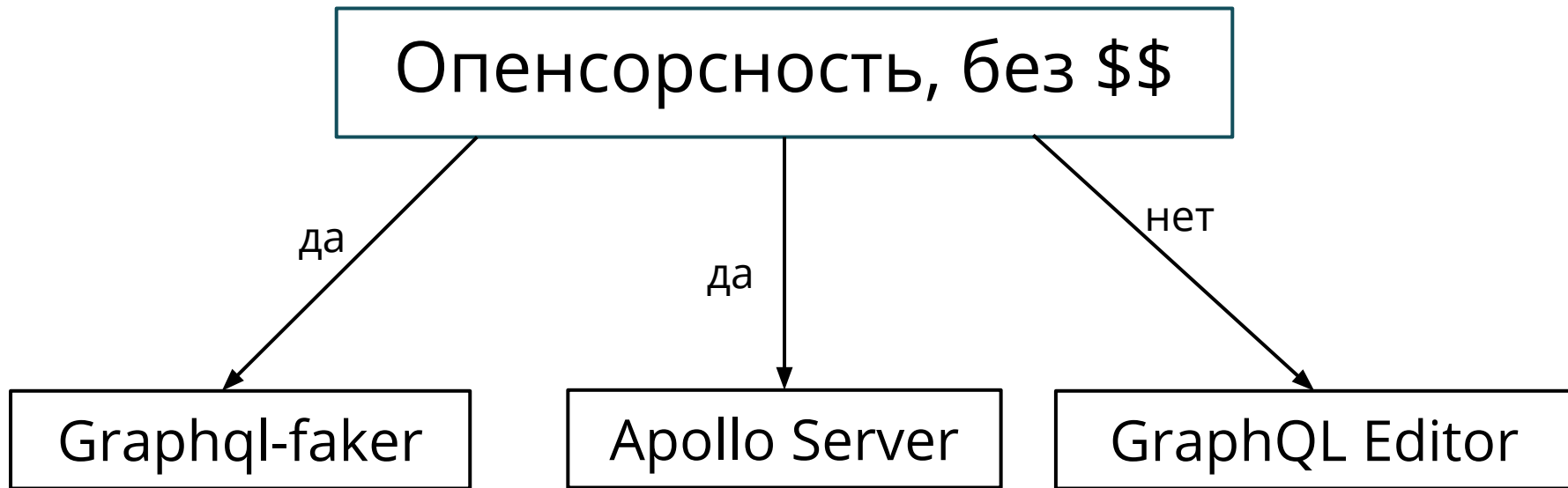
SSE



REST API, WebSocket, SSE



GraphQL



GraphQL

Локальный запуск



```
graph TD; A[Локальный запуск] -- да --> B[GraphQL-faker]; A -- да --> C[Apollo Server]; A -- нет --> D[GraphQL Editor];
```

The diagram is a flowchart starting with a central box labeled 'Локальный запуск' (Local launch). Three arrows point downwards from this box to three separate boxes below. The left arrow is labeled 'да' (yes) and points to 'GraphQL-faker'. The middle arrow is also labeled 'да' (yes) and points to 'Apollo Server'. The right arrow is labeled 'нет' (no) and points to 'GraphQL Editor'.

GraphQL-faker

Apollo Server

GraphQL Editor

Хотите? ЗаSTUBим!

 **YouTube** <https://clck.ru/NRygG>



<https://github.com/Feyweyter/stub-servers>

Стубы бэкенда



ЗаSTUVить бэк просто!

Шакшина Мария



[Feyweyter](#)



[marflenx](#)



[marflenx](#)

