# 0. Видове deploy-ване

Два варианта:

## 0.1. On-Premise (On-Prem)

имаме собствен сървър или си ко-локираме купената от нас машина в специално наето помещение.

## 0.2. Cloud

Ние не притежаваме такива сървърни машини, но си плащаме да ползваме

### Unmanaged Solution

Имаме достъп до виртуална cloud машина (лимит за трафик, ядра на машината, и т.н.)

Например Tier plan – първоначално безплатно, и когато имаме повече потребители/трафик, то си плащаме

### Managed Solution

Не само не се грижим за физически за ядрата и машината, но направо си качваме проекта някъде.

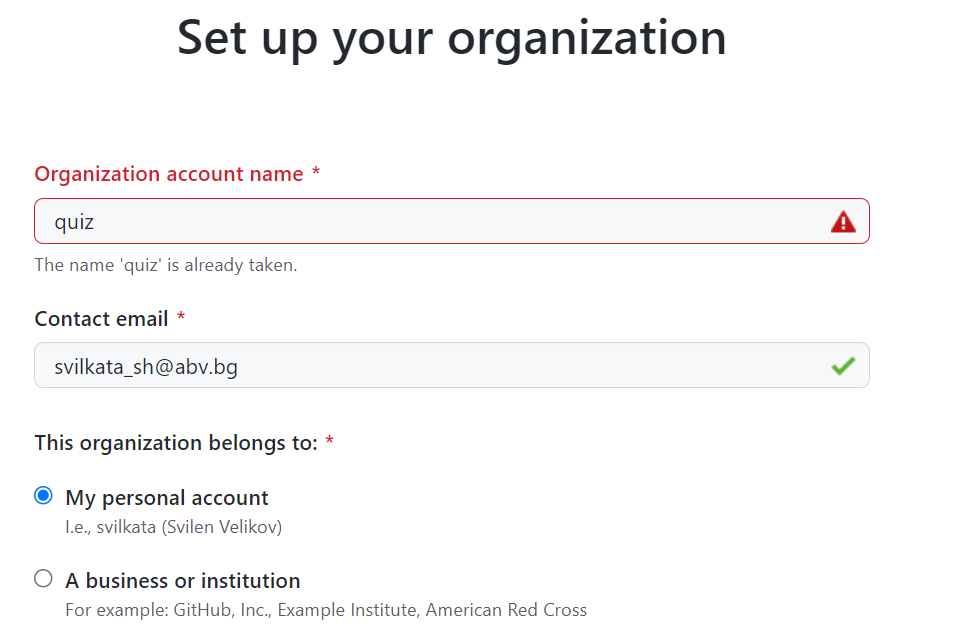
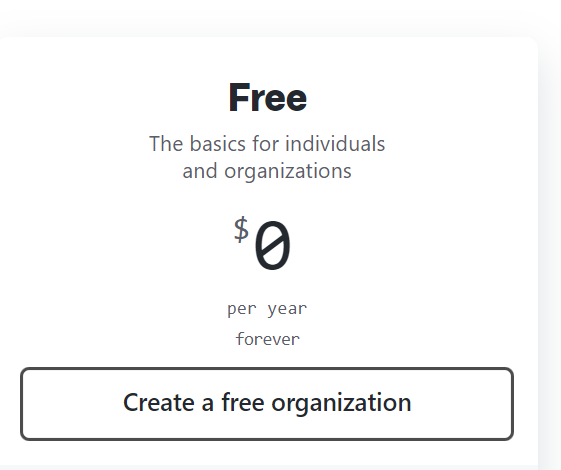
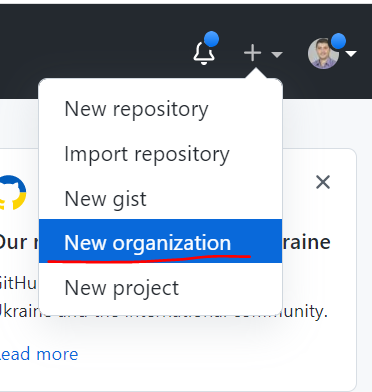
#### Server application

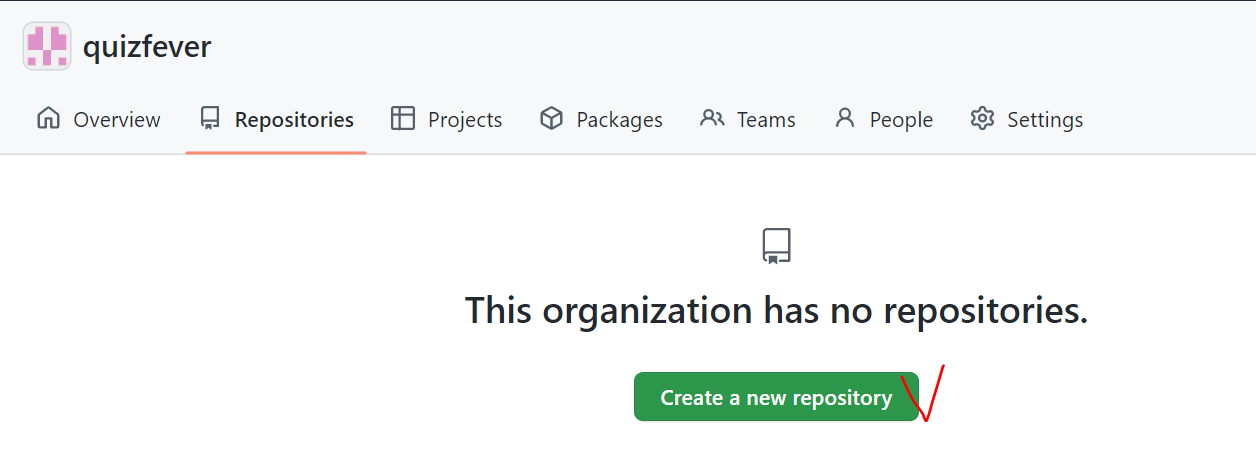
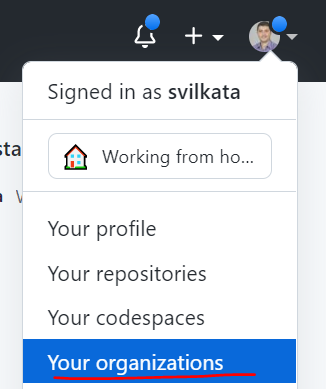
API Rest application – Back4Up, SoftUni server, database services, etc.

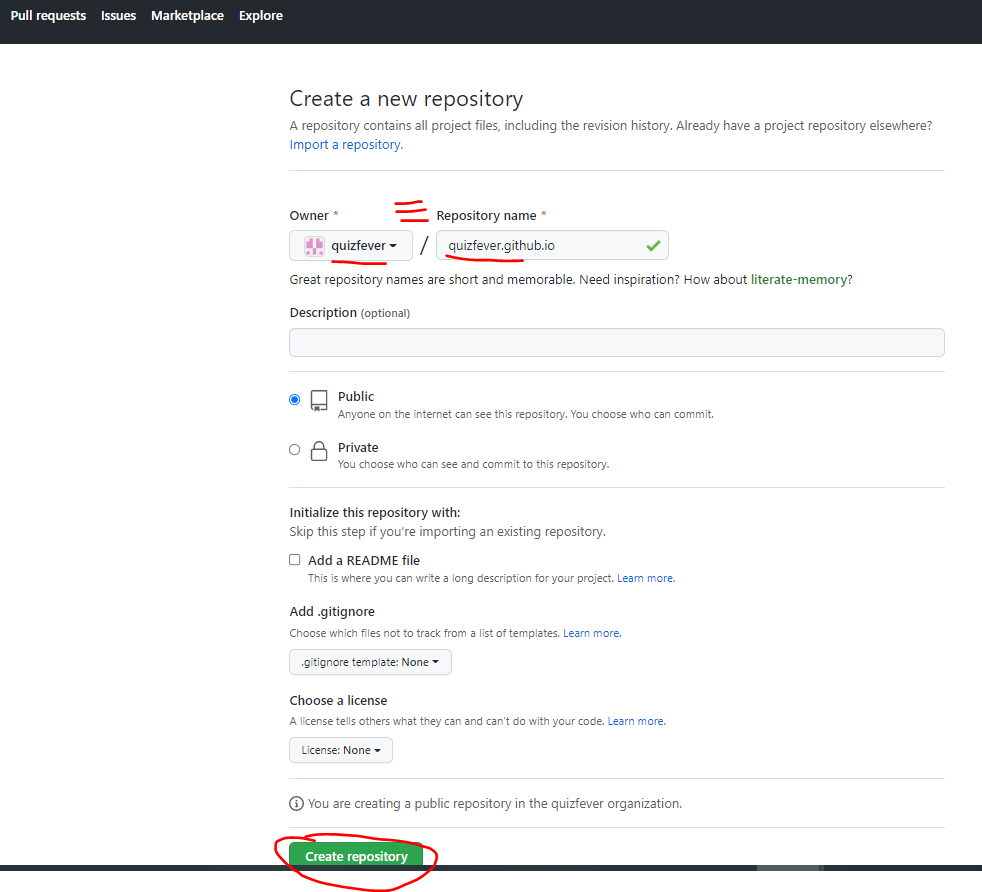
#### Client-Side application

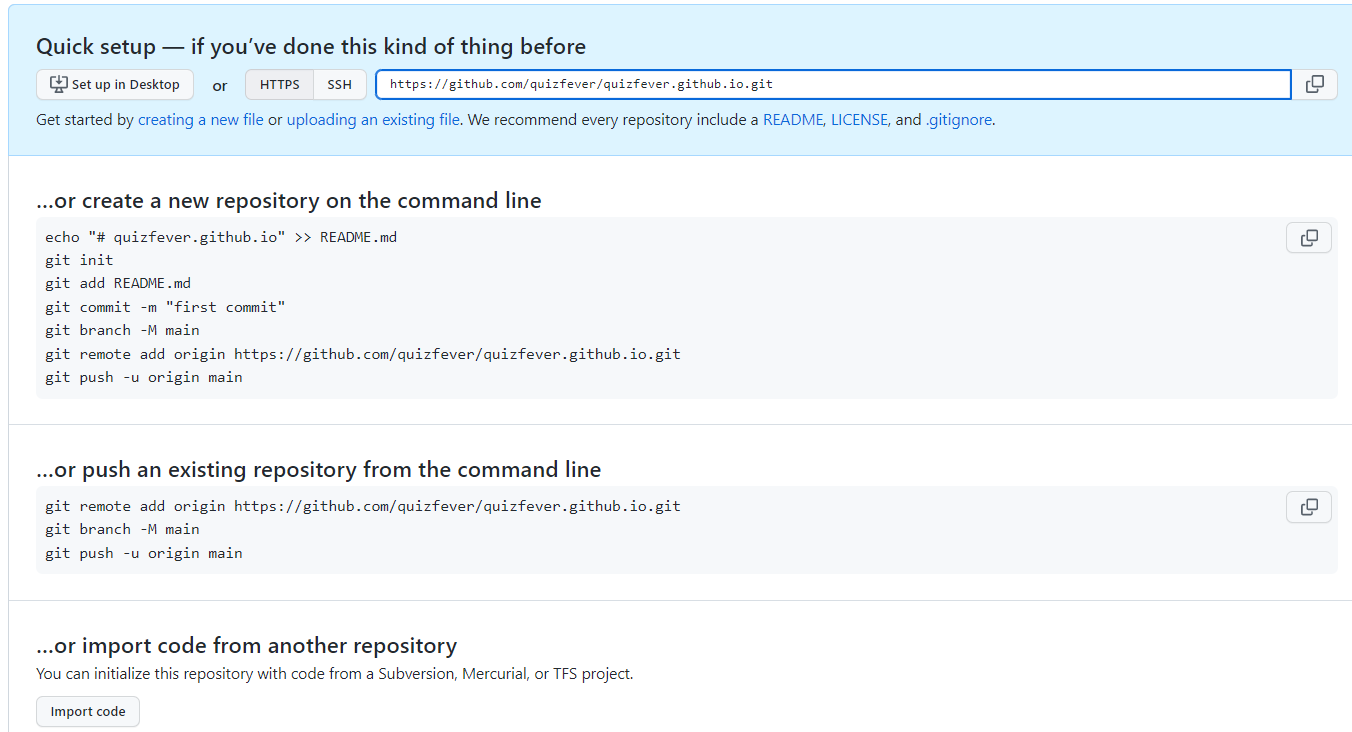
# 1. Deploying site in Host Github.io (GitHub pages) + VSC (JS Node)

## 1.1. Настройки в GitHub

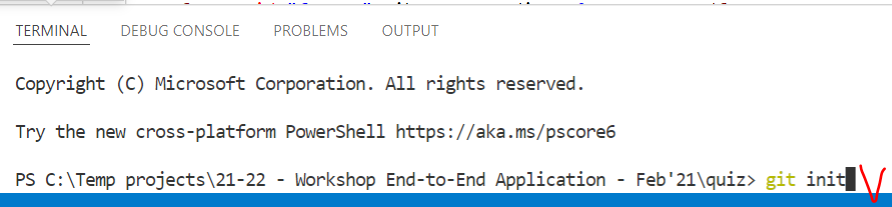




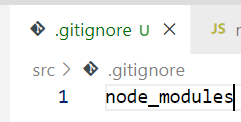


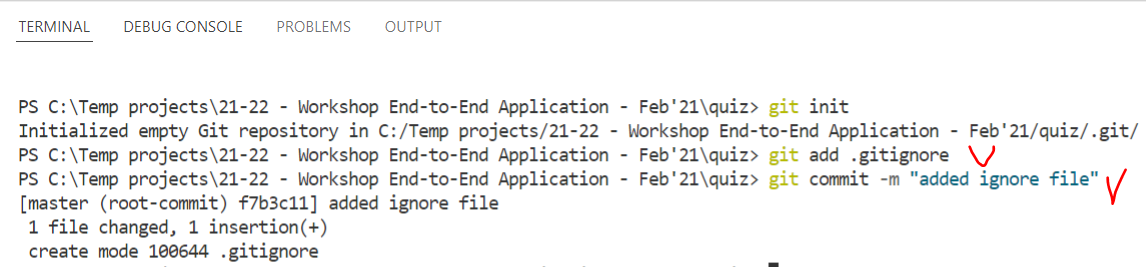


## 1.2. Инициализиране на GitHub в VSC(Node.js)



## 1.3. Добавяне на .gitignore файл



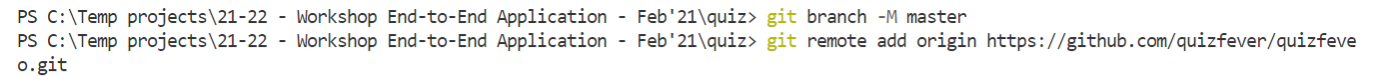


## 1.4. Добавяне на readme.md файл

Ние сме си го създали вече този файл локално – описание/документация на проекта

## 1.5. Определяне на главна директория на repo-то и свързването й с GitHub

…or push an existing repository from the command line



git remote add origin https://github.com/movies-softuni/movies-softuni.github.io.git

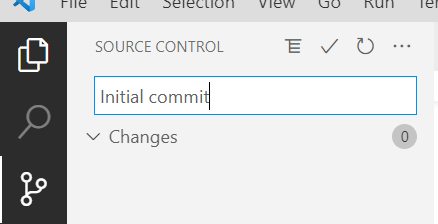
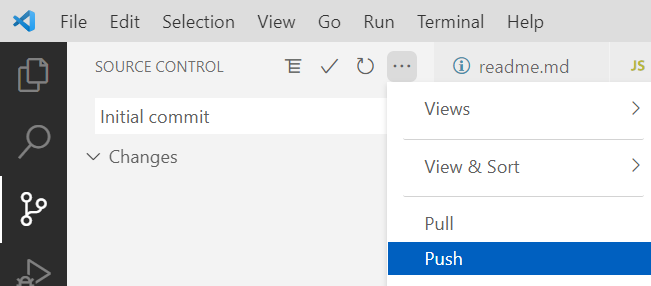
git branch -M main да **не** го правим – да си сменим master на main

## 1.6. Push

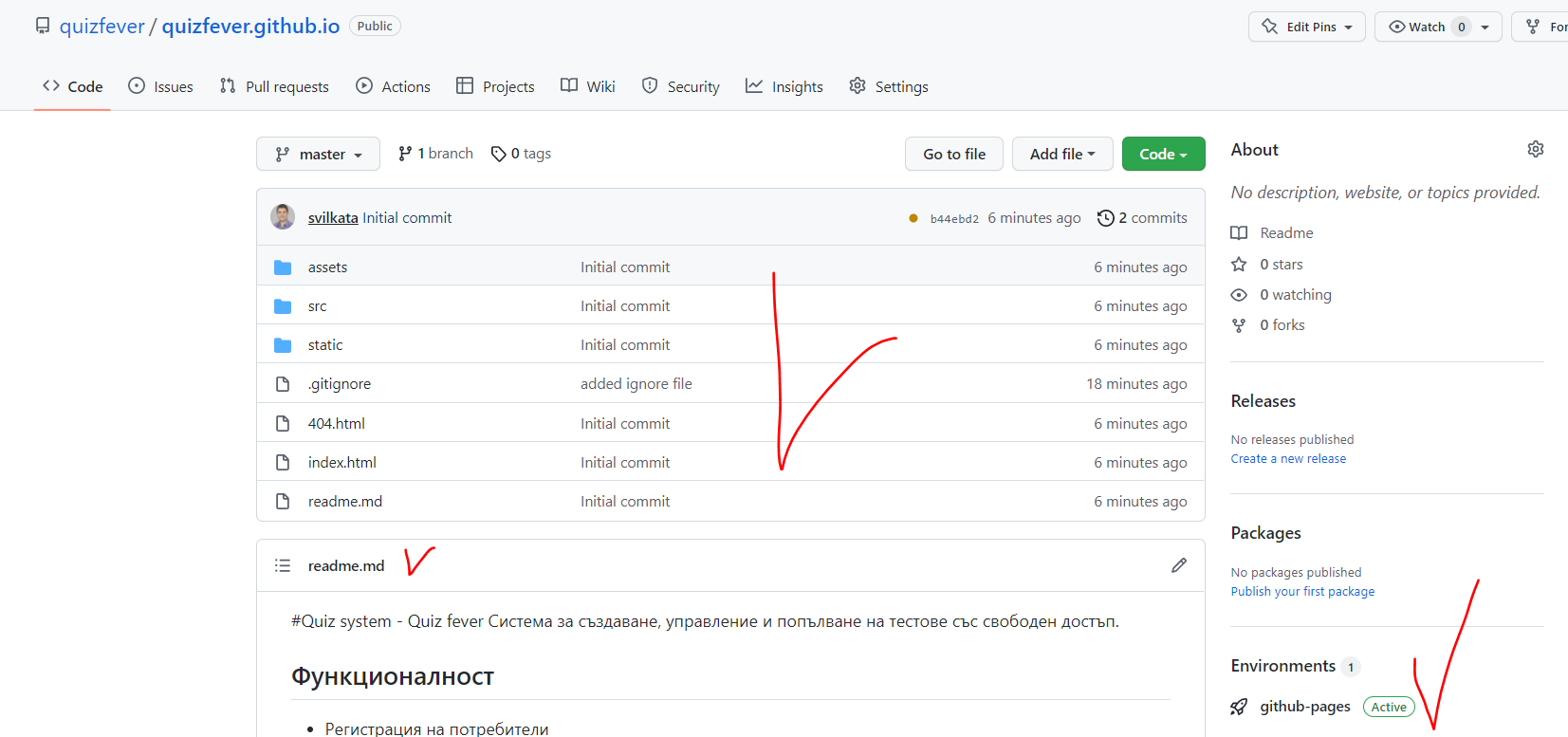


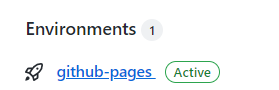
## 1.7. Initial commit from VSC interface

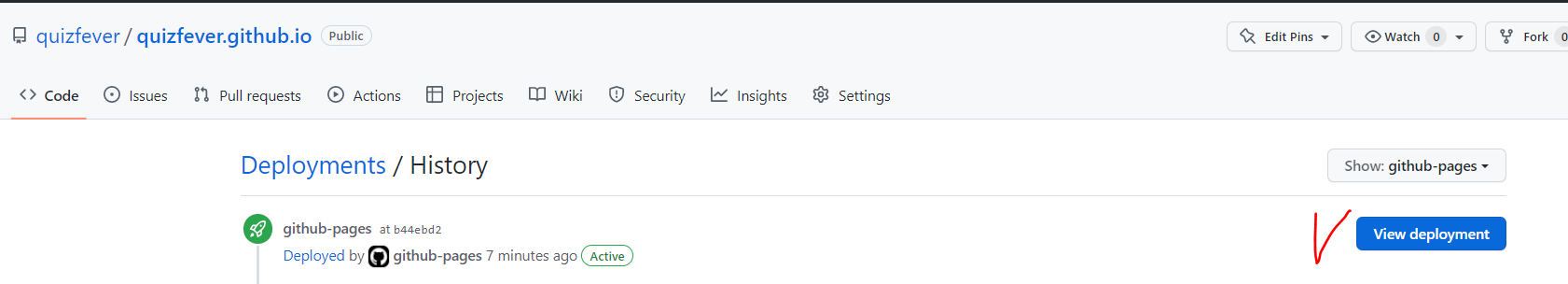
**Ctrl + Enter** – за да ги качи всичките файлове

## 1.8. We can see the result in the web of GitHub







## 1.9. Back4Up или друг BaaS – тестване на http заявки

1. x Initialize project

2. x Copy / api

3. x Create Back4Up app

4. View documentation on Users

5. x Adjust api.js to include AppID, APIKey, correct headers, register/login/logout routes and bodies

6. x Test register/login/logout

7. Examine Database Browser, create Quiz collection

8. x Create CRUD functions for Quiz collection

9. x Test Quiz collection, confirm it is pubpic for read/write, NO Add field CLP

10. View documentation on security, ACL, CLP

11. x Configure public read, authenticated write, NO add field CLP

12. x Test read/write

13. x Configure owner pointer

14. x Adjust data.js to include owner pointer on create - { \_\_type: 'Pointer', className: 'User'}

15. x Test owner protection

16. x Add query to include owner on GET

17. x Create Question collection, configure CLP, owner pointer, Quiz pointer

18. x Create CRUD functions for Question collection, with owner protection

19. x Test Question collection

(20. Proceed with implementation of views)

## 1.10. Generating the package.json file and installing external libraries

npm init -y

npm install -E lit-html page -E е за запазване

{

  "name": "quiz",

  "version": "1.0.0",

  "description": "Система за създаване, управление и попълване на тестове със свободен достъп.",

  "main": "index.html",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

  "repository": {

    "type": "git",

    "url": "git+https://github.com/quizfever/quizfever.github.io.git" //автоматично се генерира и път към git репото

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "bugs": {

    "url": "https://github.com/quizfever/quizfever.github.io/issues"

  },

  "homepage": "https://github.com/quizfever/quizfever.github.io#readme",

  "dependencies": {

    "lit-html": "2.2.3",

    "page": "1.11.6"

  }

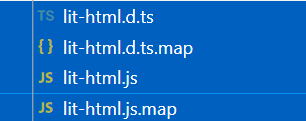
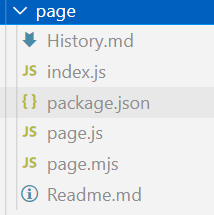
}

## 1.11. Re-commit from VSC interface

**Ctrl + Enter** – за да ги качи всичките файлове

## 1.12. How to deploy external libraries

### Option 1) Копираме само определени файлове от node-modules

Налага се да докопираме допълнителни 2-3 файла, за да не хвърля грешка

### Option 2) Вземаме директно от online CDN-а в уеб пространството

import page from '//unpkg.com/page/page.mjs';

import {html, render} from '//unpkg.com/lit-html?module';

export {

    page,

    html,

    render

}

### Option 3) Най-правилния production вариант с build стъпка

Работим си локално, и след това минава скрипт, който пакетира външните библиотеки.

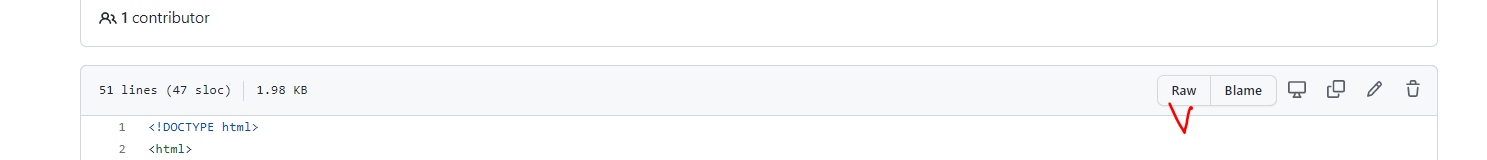
* Webpack
* Minify.js
* Browerify
* RollUp

## File 404.html - we can manually add from <https://github.com/rafgraph/spa-github-pages> - когато рефрешнем дадена страница на SPA, то дава грешка. И затова трябва да използваме 404.html

<https://github.com/rafgraph/spa-github-pages>

<https://github.com/rafgraph/spa-github-pages>

To easily copy it



На негово място слагаме този от <https://github.com/rafgraph/spa-github-pages>

<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <title>Single Page Apps for GitHub Pages</title>

    <script type="text/javascript">

      // Single Page Apps for GitHub Pages

      // MIT License

      // https://github.com/rafgraph/spa-github-pages

      // This script takes the current url and converts the path and query

      // string into just a query string, and then redirects the browser

      // to the new url with only a query string and hash fragment,

      // e.g. https://www.foo.tld/one/two?a=b&c=d#qwe, becomes

      // https://www.foo.tld/?/one/two&a=b~and~c=d#qwe

      // Note: this 404.html file must be at least 512 bytes for it to work

      // with Internet Explorer (it is currently > 512 bytes)

      // If you're creating a Project Pages site and NOT using a custom domain,

      // then set pathSegmentsToKeep to 1 (enterprise users may need to set it to > 1).

      // This way the code will only replace the route part of the path, and not

      // the real directory in which the app resides, for example:

      // https://username.github.io/repo-name/one/two?a=b&c=d#qwe becomes

      // https://username.github.io/repo-name/?/one/two&a=b~and~c=d#qwe

      // Otherwise, leave pathSegmentsToKeep as 0.

      var pathSegmentsToKeep = 0;

      var l = window.location;

      l.replace(

        l.protocol + '//' + l.hostname + (l.port ? ':' + l.port : '') +

        l.pathname.split('/').slice(0, 1 + pathSegmentsToKeep).join('/') + '/?/' +

        l.pathname.slice(1).split('/').slice(pathSegmentsToKeep).join('/').replace(/&/g, '~and~') +

        (l.search ? '&' + l.search.slice(1).replace(/&/g, '~and~') : '') +

        l.hash

      );

    </script>

  </head>

  <body>

  </body>

</html>

И слагаме в index.html, преди всички други script-ове, този скрипт от <https://github.com/rafgraph/spa-github-pages>:

1)     <!-- Start Single Page Apps for GitHub Pages -->

    <script type="text/javascript">

        // Single Page Apps for GitHub Pages

        // MIT License

        // https://github.com/rafgraph/spa-github-pages

        // This script checks to see if a redirect is present in the query string,

        // converts it back into the correct url and adds it to the

        // browser's history using window.history.replaceState(...),

        // which won't cause the browser to attempt to load the new url.

        // When the single page app is loaded further down in this file,

        // the correct url will be waiting in the browser's history for

        // the single page app to route accordingly.

        (function(l) {

          if (l.search[1] === '/' ) {

            var decoded = l.search.slice(1).split('&').map(function(s) {

              return s.replace(/~and~/g, '&')

            }).join('?');

            window.history.replaceState(null, null,

                l.pathname.slice(0, -1) + decoded + l.hash

            );

          }

        }(window.location))

      </script>

      <!-- End Single Page Apps for GitHub Pages -->

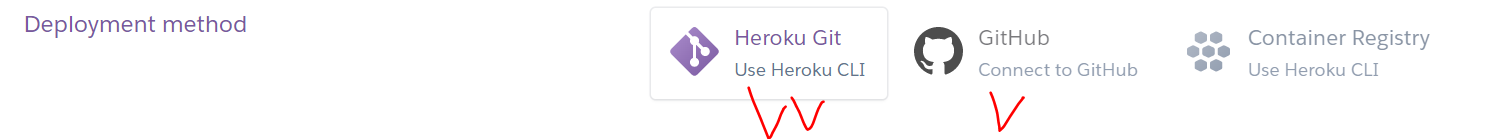
2)     <script src="src/app.js" type="module"></script>

# 2. Deploung in Host Heroku + VSC (JS Node) – we can deploy the RestApi backend softuni server

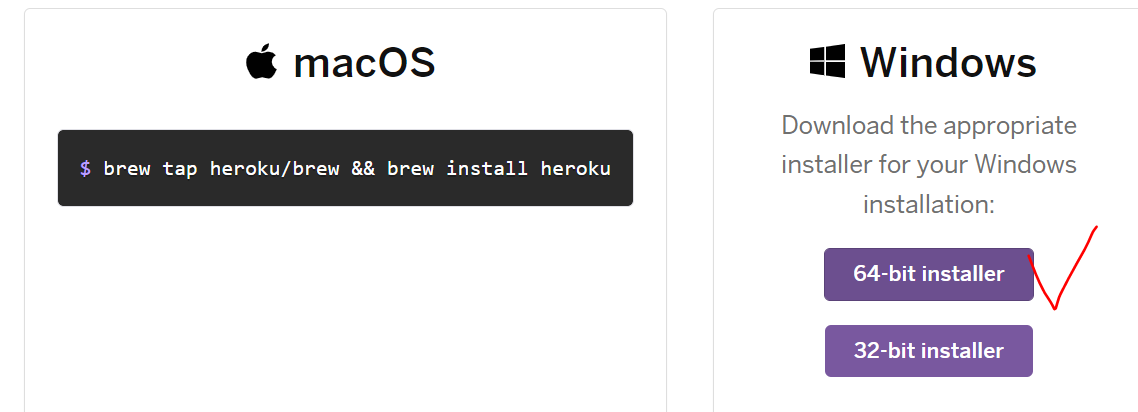
<https://www.heroku.com/>

## 2.1. Регистриране в heroku

* Register in Heroku
* Create New App



Download and install the [Heroku CLI](https://devcenter.heroku.com/articles/heroku-command-line). - Command-Line interface



## 2.2. Нагласяне на SoftUni server-a

В папката apiRest\_SoftUniServer

npm init -y

package.json

{

  "name": "apiRest\_SoftUniServer",

  "version": "1.0.0",

  "description": "",

  "main": "server.js",

  "scripts": {

    "start": **"node server.js"**

  },

  "keywords": [],

  "author": "",

  "license": "ISC"

}

npm run start или npm start работи.

## 2.3. Да влезем в профила си в Heroku CLI

С тази команда в Shell-a, която пак ще ни отведе до login страницата на Heroku.

heroku login

## 2.4. Create a new Git repository and deploy it

git init

heroku git:remote -a movies-softuni

git add .

git status

git commit -am "Initial commit" - -a означава всичко, което е модифицирано, го качвай

git push heroku master

нашият софтУни сървър не е направен да се качва в Heroku, затова трябва да му зададем автоматичен порт, и хероку сам да си определя на кой порт да се рънва.

const server = http\_\_default['default'].createServer(requestHandler(plugins, services));

    const port = 3030;

    const port = process.env.PORT || 3030; //така ще тръгне в Heroku, и ще бачка и без да използваме heroku

    server.listen(port);

    console.log(`Server started on port ${port}. You can make requests to http://localhost:${port}/`);

    console.log(`Admin panel located at http://localhost:${port}/admin`);

    var softuniPracticeServer = {

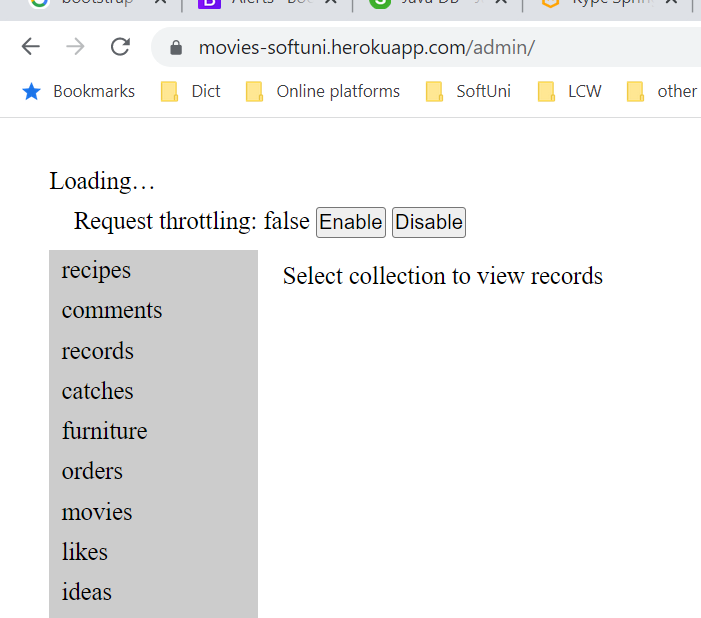
    };

    return softuniPracticeServer;

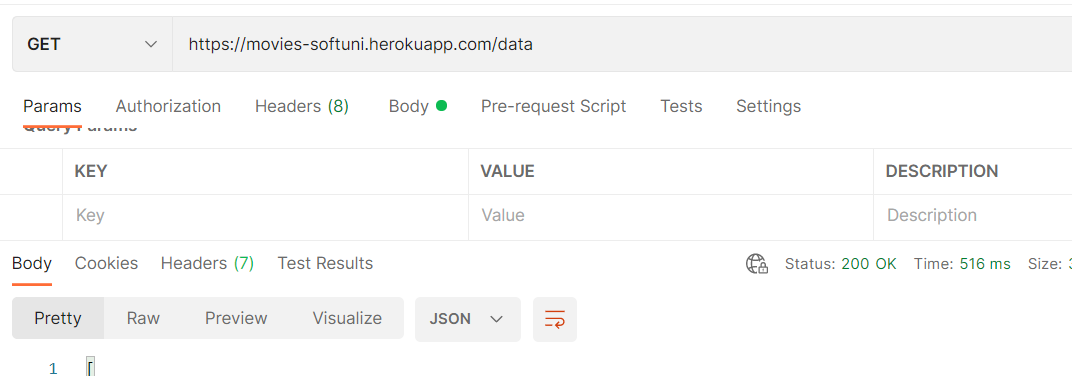
git add . може и без тази команда, защото следващата команда е -a (-a означава всичко, което е модифицирано, го качвай)

git commit -am "Add port env. variable"

git push heroku master

https://**movies-softuni**.herokuapp.com/ 

И вече softUni сервър работи стартиран от интернет пространството



## 2.5. Front-End/Client server we make via github pages -> new organization

Сменяме и base URL адреса на backend услугата:

// export const baseUrl = 'http://localhost:3030';

export const baseUrl = 'https://movies-softuni.herokuapp.com'; //URL-то на SoftUni server, който качихме в Heroku.

#### CORS - Access-Control-Allow-Origin

Няма нужда

options.headers['Access-Control-Allow-Origin'] = '\*';

re-run on the local lite-server for example

Access-Control-Allow-Origin: \*

Access-Control-Allow-Origin: <origin>

Access-Control-Allow-Origin: null

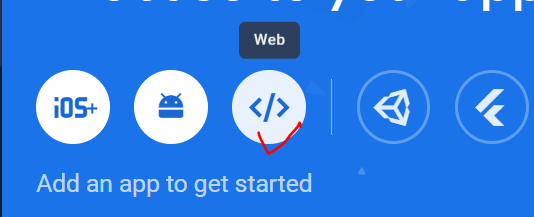
Ако имаме в requester.js console.log(response) и се чупи.

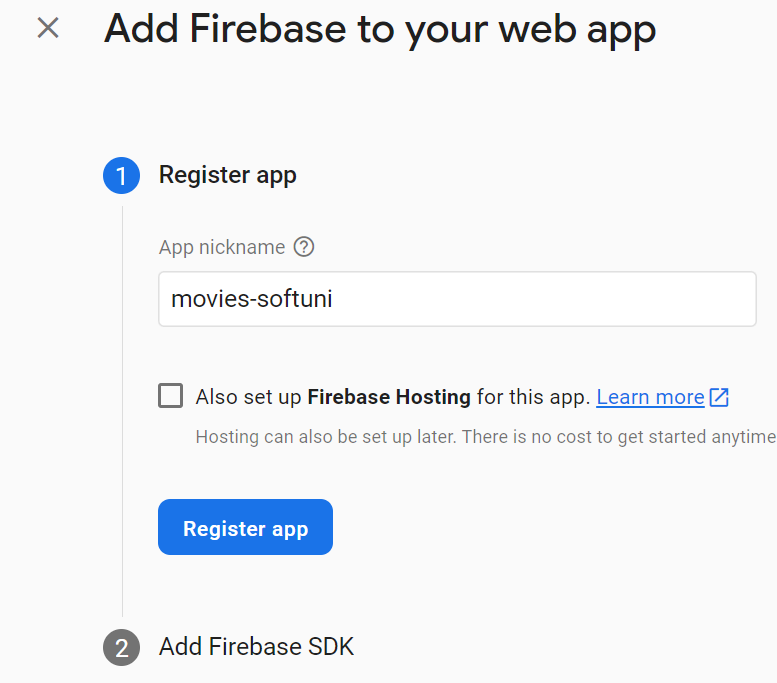
# 3. Deploung in Host Firebase + VSC (JS Node) – we can deploy front-end client server

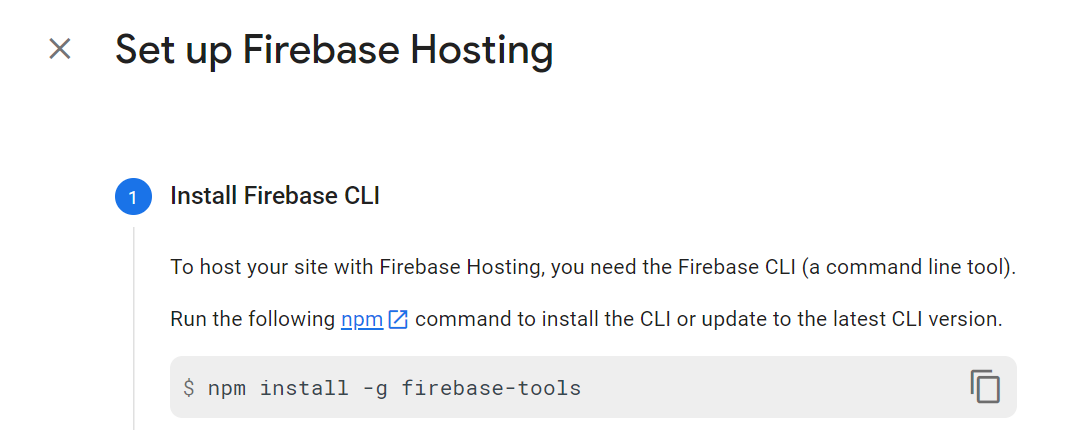
<https://console.firebase.google.com> - login with gmail account

## **movies-softuni**

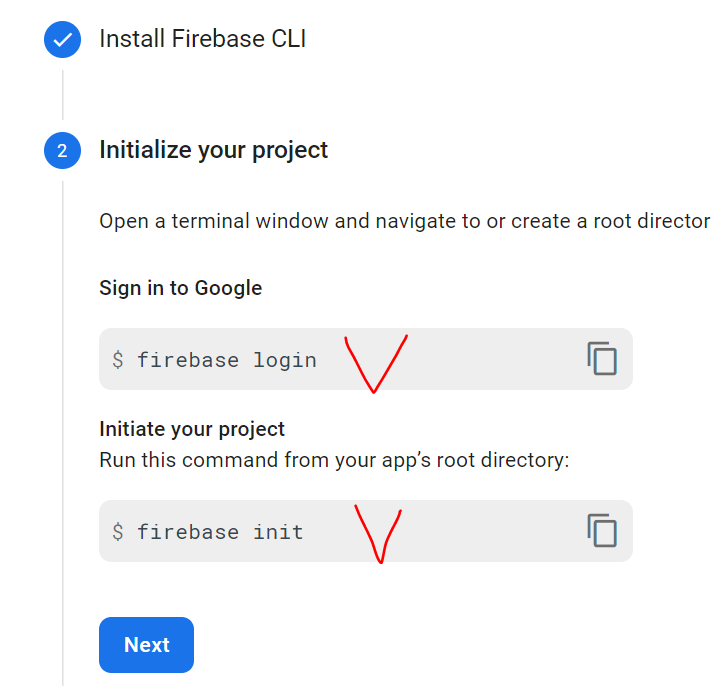
Към един проект, може да има няколко вида front-end – за мобилно устройство Android, за web, за apple,





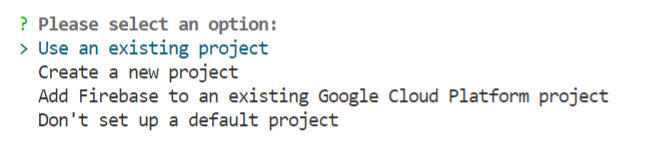


Command-Line interface

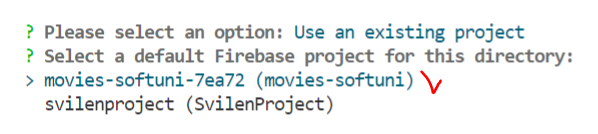




Enter

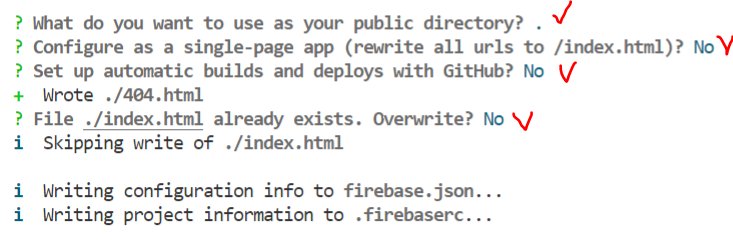


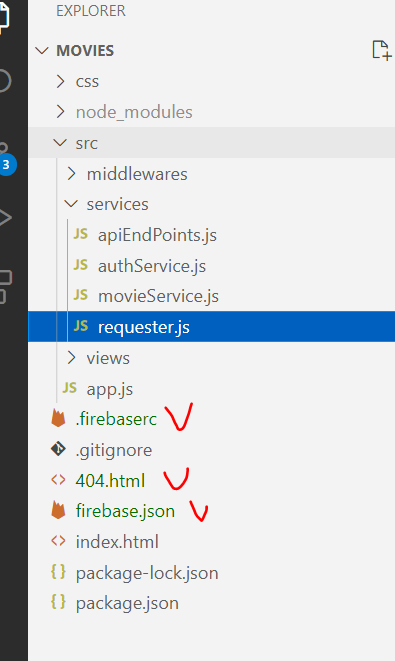
Enter

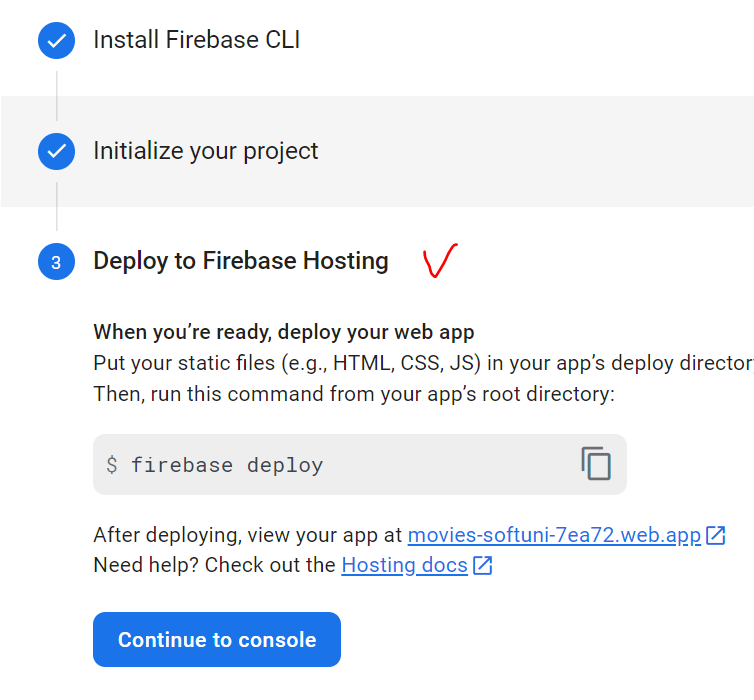


Enter

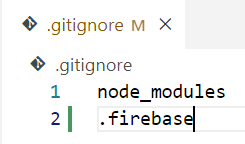
Точката представлява, че нашата public directory е текущата директория.



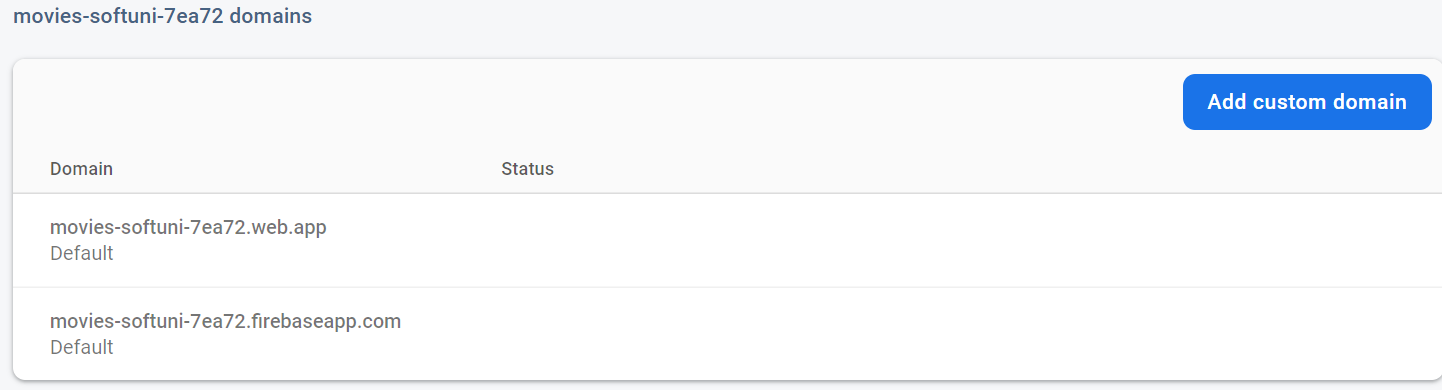




Тази папкa .firebase трябва да я ignore-нем.



Два домейна ни прави – могат да се ползват и те



## File 404.html - automatically generated by firebase – когато рефрешнем дадена страница на SPA, то дава грешка. И затова трябва да използваме 404.html

### <!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>Page Not Found</title>

    <style media="screen">

      body { background: #ECEFF1; color: rgba(0,0,0,0.87); font-family: Roboto, Helvetica, Arial, sans-serif; margin: 0; padding: 0; }

      #message { background: white; max-width: 360px; margin: 100px auto 16px; padding: 32px 24px 16px; border-radius: 3px; }

      #message h3 { color: #888; font-weight: normal; font-size: 16px; margin: 16px 0 12px; }

      #message h2 { color: #ffa100; font-weight: bold; font-size: 16px; margin: 0 0 8px; }

      #message h1 { font-size: 22px; font-weight: 300; color: rgba(0,0,0,0.6); margin: 0 0 16px;}

      #message p { line-height: 140%; margin: 16px 0 24px; font-size: 14px; }

      #message a { display: block; text-align: center; background: #039be5; text-transform: uppercase; text-decoration: none; color: white; padding: 16px; border-radius: 4px; }

      #message, #message a { box-shadow: 0 1px 3px rgba(0,0,0,0.12), 0 1px 2px rgba(0,0,0,0.24); }

      #load { color: rgba(0,0,0,0.4); text-align: center; font-size: 13px; }

      @media (max-width: 600px) {

        body, #message { margin-top: 0; background: white; box-shadow: none; }

        body { border-top: 16px solid #ffa100; }

      }

    </style>

  </head>

  <body>

    <div id="message">

      <h2>404</h2>

      <h1>Page Not Found</h1>

      <p>The specified file was not found on this website. Please check the URL for mistakes and try again.</p>

      <h3>Why am I seeing this?</h3>

      <p>This page was generated by the Firebase Command-Line Interface. To modify it, edit the <code>404.html</code> file in your project's configured <code>public</code> directory.</p>

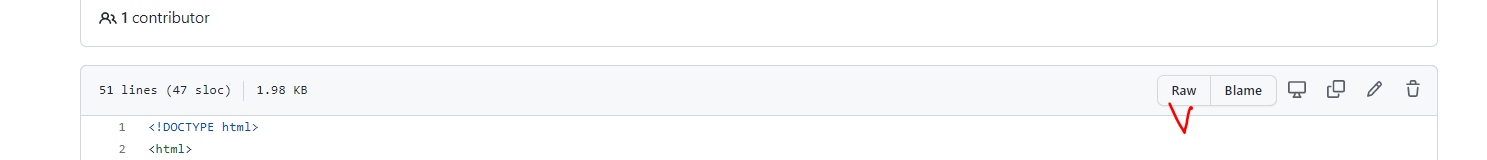
    </div>

  </body>

</html>

<https://github.com/rafgraph/spa-github-pages>

To easily copy it



На негово място слагаме този от <https://github.com/rafgraph/spa-github-pages>

<!DOCTYPE html>

<html>

  <head>

    <meta charset="utf-8">

    <title>Single Page Apps for GitHub Pages</title>

    <script type="text/javascript">

      // Single Page Apps for GitHub Pages

      // MIT License

      // https://github.com/rafgraph/spa-github-pages

      // This script takes the current url and converts the path and query

      // string into just a query string, and then redirects the browser

      // to the new url with only a query string and hash fragment,

      // e.g. https://www.foo.tld/one/two?a=b&c=d#qwe, becomes

      // https://www.foo.tld/?/one/two&a=b~and~c=d#qwe

      // Note: this 404.html file must be at least 512 bytes for it to work

      // with Internet Explorer (it is currently > 512 bytes)

      // If you're creating a Project Pages site and NOT using a custom domain,

      // then set pathSegmentsToKeep to 1 (enterprise users may need to set it to > 1).

      // This way the code will only replace the route part of the path, and not

      // the real directory in which the app resides, for example:

      // https://username.github.io/repo-name/one/two?a=b&c=d#qwe becomes

      // https://username.github.io/repo-name/?/one/two&a=b~and~c=d#qwe

      // Otherwise, leave pathSegmentsToKeep as 0.

      var pathSegmentsToKeep = 0;

      var l = window.location;

      l.replace(

        l.protocol + '//' + l.hostname + (l.port ? ':' + l.port : '') +

        l.pathname.split('/').slice(0, 1 + pathSegmentsToKeep).join('/') + '/?/' +

        l.pathname.slice(1).split('/').slice(pathSegmentsToKeep).join('/').replace(/&/g, '~and~') +

        (l.search ? '&' + l.search.slice(1).replace(/&/g, '~and~') : '') +

        l.hash

      );

    </script>

  </head>

  <body>

  </body>

</html>

И слагаме в index.html, преди всички други script-ове, този скрипт от <https://github.com/rafgraph/spa-github-pages>:

1)     <!-- Start Single Page Apps for GitHub Pages -->

    <script type="text/javascript">

        // Single Page Apps for GitHub Pages

        // MIT License

        // https://github.com/rafgraph/spa-github-pages

        // This script checks to see if a redirect is present in the query string,

        // converts it back into the correct url and adds it to the

        // browser's history using window.history.replaceState(...),

        // which won't cause the browser to attempt to load the new url.

        // When the single page app is loaded further down in this file,

        // the correct url will be waiting in the browser's history for

        // the single page app to route accordingly.

        (function(l) {

          if (l.search[1] === '/' ) {

            var decoded = l.search.slice(1).split('&').map(function(s) {

              return s.replace(/~and~/g, '&')

            }).join('?');

            window.history.replaceState(null, null,

                l.pathname.slice(0, -1) + decoded + l.hash

            );

          }

        }(window.location))

      </script>

      <!-- End Single Page Apps for GitHub Pages -->

2)     <script src="src/app.js" type="module"></script>

firebase deploy и готово.

# 4. Deploying site in Host <https://www.netlify.com/>

# 5. Deploying SPRING project – see Spring notes