

# HTTP and REST Services

HTTP, Request Headers, RESTful Web Services



**REST API**

SoftUni Team  
Technical Trainers



**SoftUni**



Software University

<https://softuni.bg>

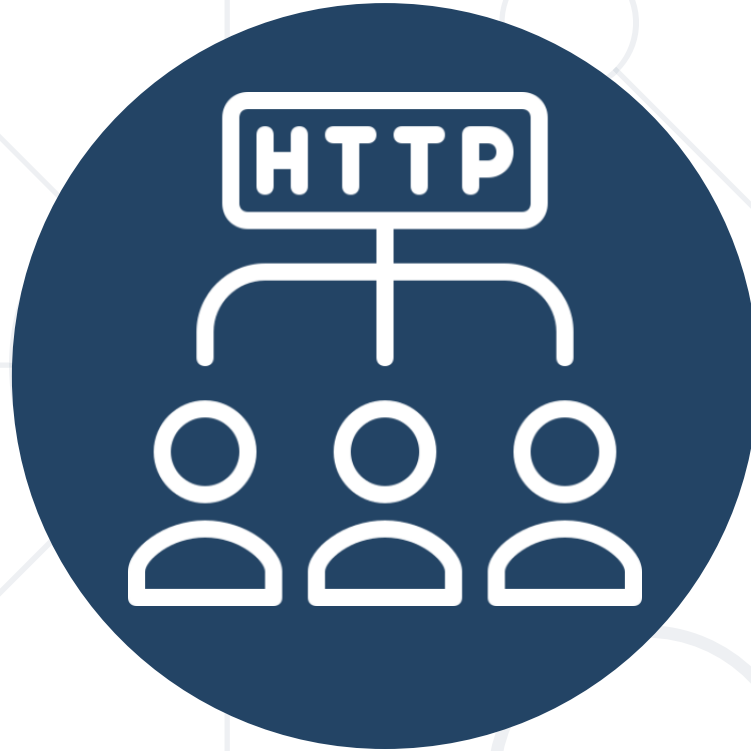
sli.do

**#js-front-end**

# Table of Contents

1. HTTP Overview & Developer Tools
2. REST & RESTful Services
3. Accessing the GitHub API
4. NPM & package.json
5. CRUD Operations to a NodeJS REST API

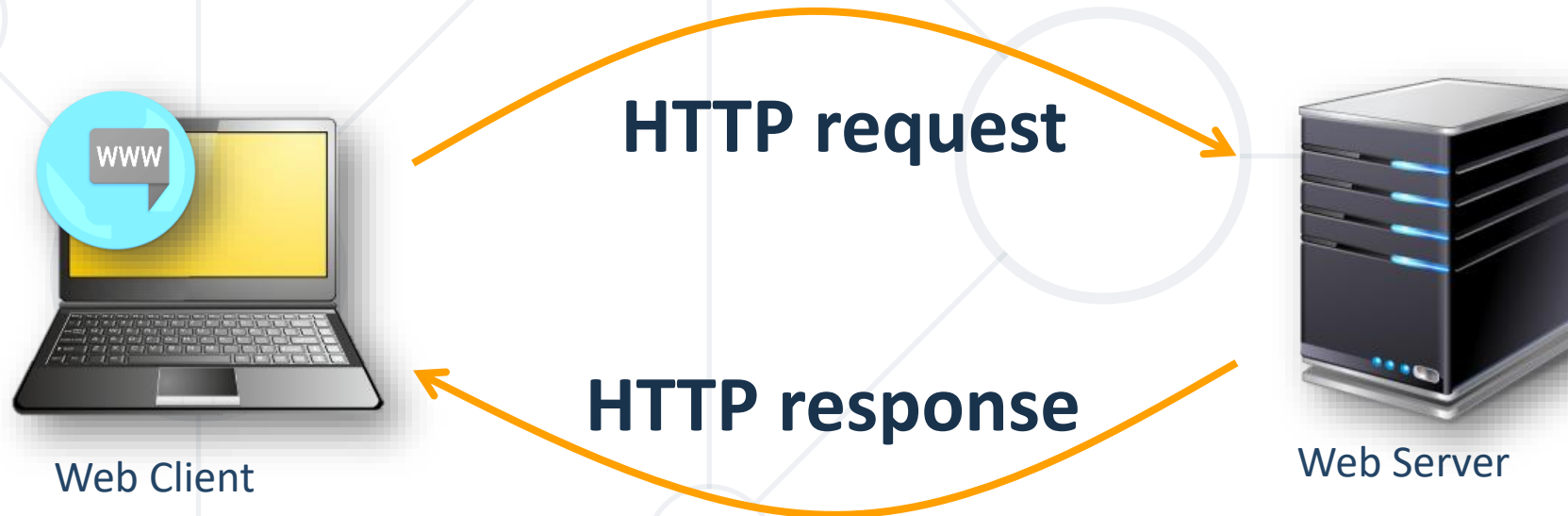










# HTTP Overview

Hypertext Transfer Protocol

- HTTP (**H**yper **T**ext **T**ransfer **P**rotocol)
  - Text-based client-server protocol for the Internet
  - For transferring Web resources (HTML files, images, styles, etc.)
  - Request-response based



- **HTTP** defines **methods** to indicate the desired action to be performed on the identified resource

Method		Description
GET		Retrieve / load a resource
POST		Create / store a resource
PUT		Update a resource
DELETE		Delete (remove) a resource
PATCH		Update resource partially
HEAD		Retrieve the resource's headers
OPTIONS		Returns the HTTP methods that the server supports for the specified URL

# HTTP GET Request – Example

**GET** /users/testnakov/repos **HTTP/1.1**

HTTP request line

Host: api.github.com

Accept: \*/\*

Accept-Language: en

HTTP headers

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/54.0.2840.71 Safari/537.36

Connection: Keep-Alive

Cache-Control: no-cache

**<CRLF>**

The request body is empty

# HTTP POST Request – Example

**POST** /repos/testnakov/test-nakov-repo/issues **HTTP/1.1**

Host: api.github.com

Accept: \*/\*

Accept-Language: en

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible;MSIE 6.0; Windows NT 5.0)

Connection: Keep-Alive

Cache-Control: no-cache

**<CRLF>**

{"title": "Found a bug",

"body": "I'm having a problem with this.",

"labels": ["bug", "minor"]}

**<CRLF>**

HTTP request line

HTTP headers

The request body holds  
the submitted data



# HTTP Response – Example

HTTP/1.1 200 OK

HTTP response status line

Date: Fri, 11 Nov 2016 16:09:18 GMT+2

Server: Apache/2.2.14 (Linux)

Accept-Ranges: bytes

Content-Length: 84

Content-Type: text/html

HTTP response headers

<CRLF>

<html>

<head><title>Test</title></head>

<body>Test HTML page.</body>

</html>

HTTP response body

# HTTP Response Status Codes

Status Code	Action	Description
200	OK	Successfully retrieved resource
201	Created	A new resource was created
204	No Content	Request has nothing to return
301 / 302	Moved	Moved to another location (redirect)
400	Bad Request	Invalid request / syntax error
401 / 403	Unauthorized	Authentication failed / Access denied
404	Not Found	Invalid resource
409	Conflict	Conflict was detected, e.g. duplicated email
500 / 503	Server Error	Internal server error / Service unavailable

# Content-Type and Disposition

- The **Content-Type** / **Content-Disposition** headers specify how the HTTP request / response body should be processed

JSON-encoded data

Content-Type: **application/json**

UTF-8 encoded HTML page.  
Will be shown in the browser

Content-Type: **text/html**; charset=utf-8

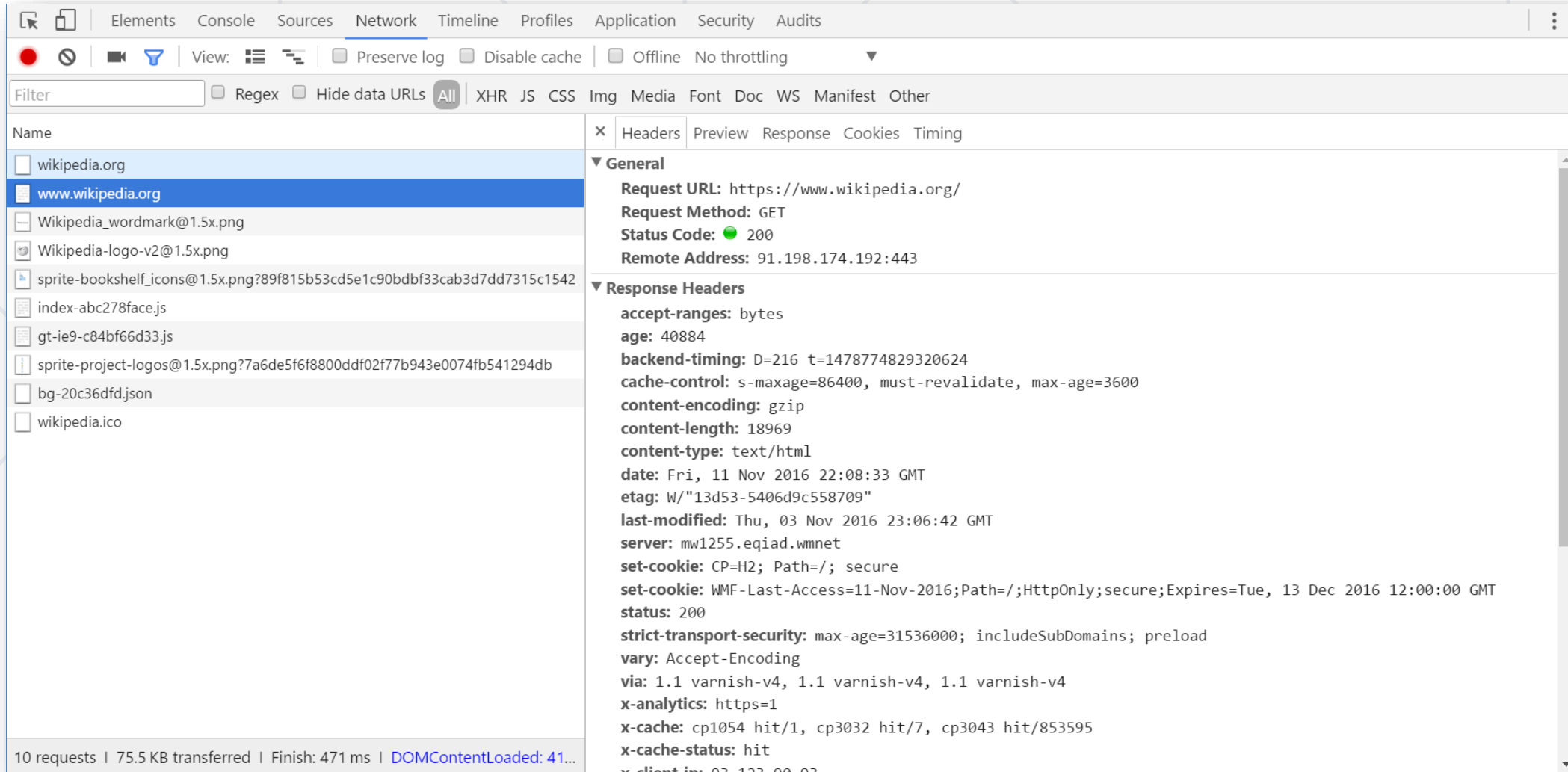
This will download a PDF file named  
Financial-Report-April-2016.pdf

Content-Type: **application/pdf**

Content-Disposition: attachment;

filename="Financial-Report-April-2016.pdf"

# Browser Developer Tools

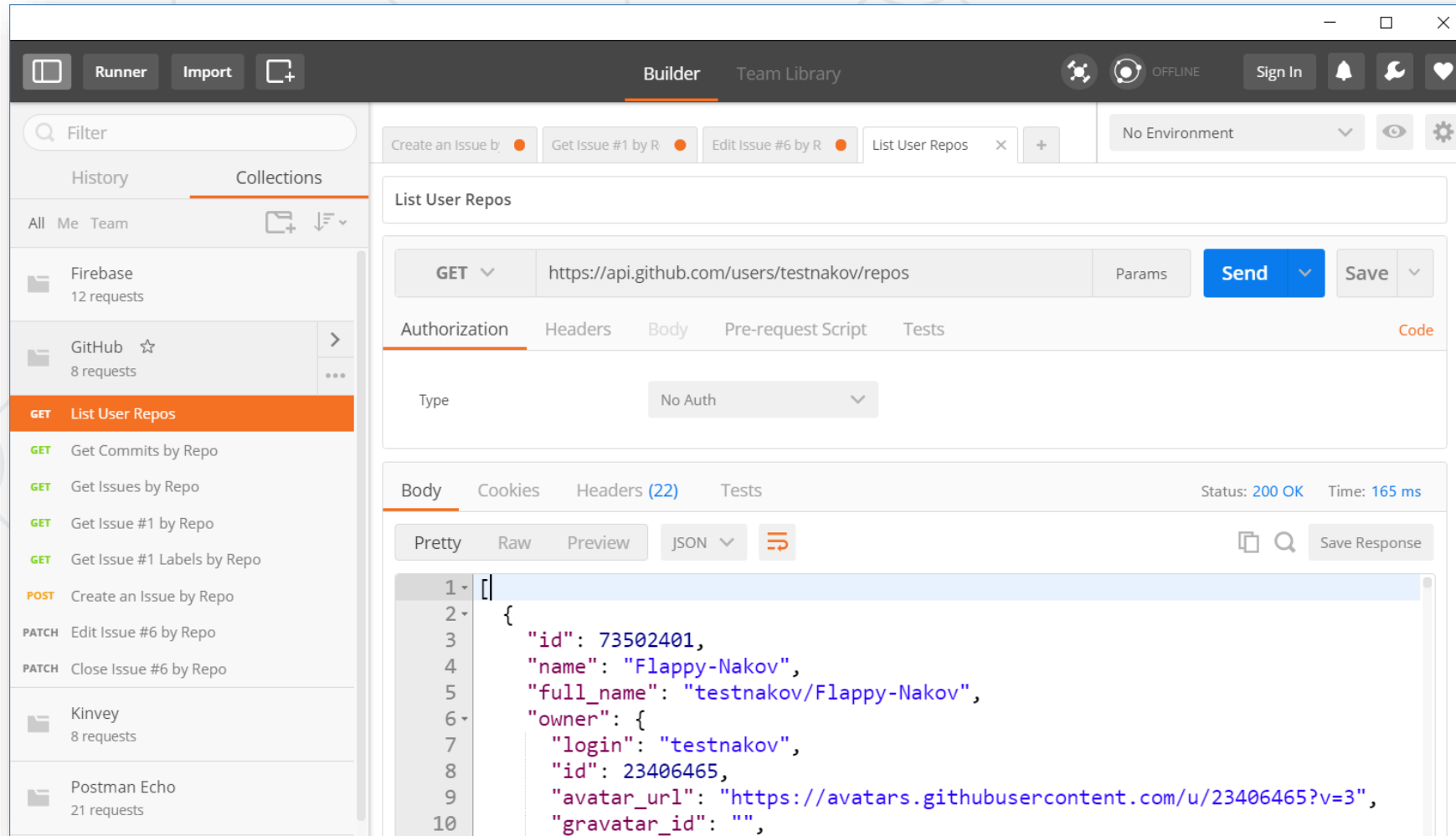


The screenshot displays the Chrome DevTools Network tab. The left sidebar shows a list of network requests, with `www.wikipedia.org` selected. The main panel shows the details for this request, including the Request URL, Request Method, Status Code, and Remote Address. The Response Headers section is expanded, showing various headers such as `accept-ranges`, `age`, `backend-timing`, `cache-control`, `content-encoding`, `content-length`, `content-type`, `date`, `etag`, `last-modified`, `server`, `set-cookie`, `status`, `strict-transport-security`, `vary`, `via`, `x-analytics`, `x-cache`, and `x-cache-status`.

**Network Tab Details:**

- Name:** `www.wikipedia.org`
- Request URL:** `https://www.wikipedia.org/`
- Request Method:** `GET`
- Status Code:** `200`
- Remote Address:** `91.198.174.192:443`
- Response Headers:**
  - `accept-ranges:` bytes
  - `age:` 40884
  - `backend-timing:` D=216 t=1478774829320624
  - `cache-control:` s-maxage=86400, must-revalidate, max-age=3600
  - `content-encoding:` gzip
  - `content-length:` 18969
  - `content-type:` text/html
  - `date:` Fri, 11 Nov 2016 22:08:33 GMT
  - `etag:` W/"13d53-5406d9c558709"
  - `last-modified:` Thu, 03 Nov 2016 23:06:42 GMT
  - `server:` mw1255.eqiad.wmnet
  - `set-cookie:` CP=H2; Path=/; secure
  - `set-cookie:` WMF-Last-Access=11-Nov-2016;Path=/;HttpOnly;secure;Expires=Tue, 13 Dec 2016 12:00:00 GMT
  - `status:` 200
  - `strict-transport-security:` max-age=31536000; includeSubDomains; preload
  - `vary:` Accept-Encoding
  - `via:` 1.1 varnish-v4, 1.1 varnish-v4, 1.1 varnish-v4
  - `x-analytics:` https=1
  - `x-cache:` cp1054 hit/1, cp3032 hit/7, cp3043 hit/853595
  - `x-cache-status:` hit

10 requests | 75.5 KB transferred | Finish: 471 ms | DOMContentLoaded: 41...



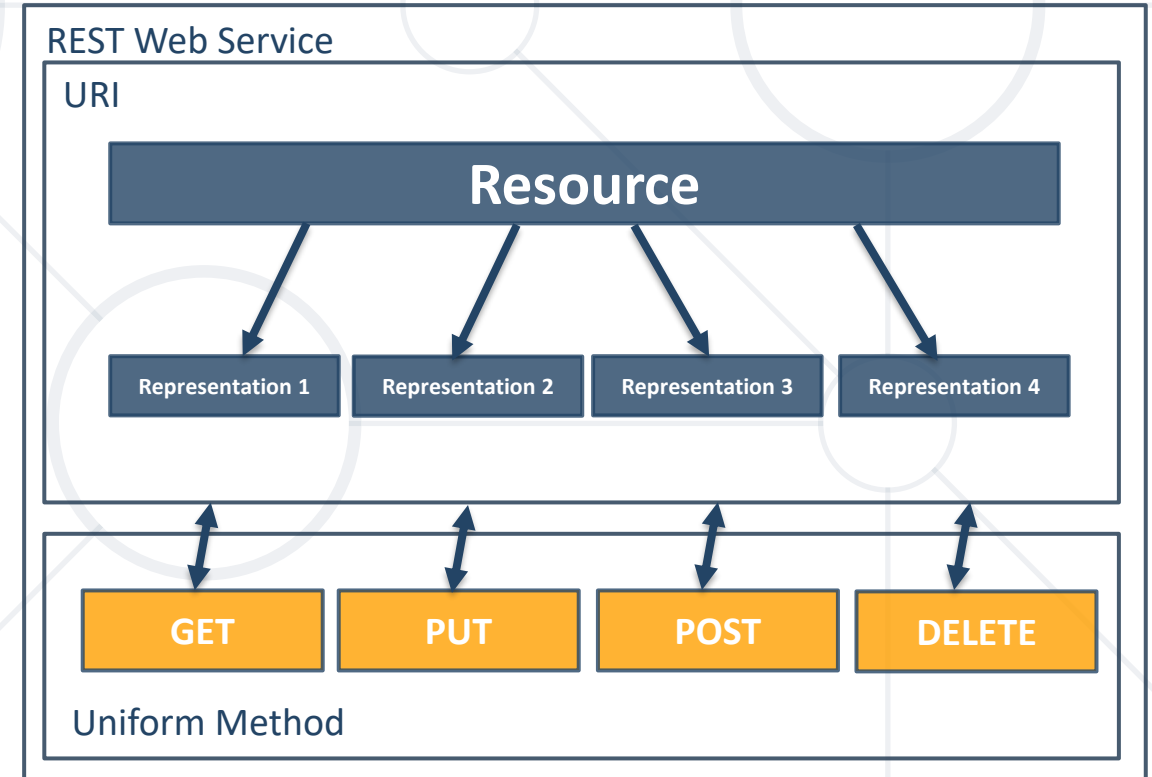
[Read more about Postman REST Client](#)



{REST}

**REST and RESTful Services**

- **R**epresentational **S**tate **T**ransfer (**REST**)
  - Architecture for **client-server communication** over HTTP
  - Resources have **URI** (address)
  - Can be **created/retrieved/modified/deleted**/etc.
- RESTful API/RESTful Service
  - Provides access to **server-side resources** via **HTTP** and **REST**



# REST and RESTful Services – Example

- Create a new post

POST	<a href="http://some-service.org/api/posts">http://some-service.org/api/posts</a>
------	---

- Get all posts / specific post

GET	<a href="http://some-service.org/api/posts">http://some-service.org/api/posts</a>
-----	---

GET	<a href="http://some-service.org/api/posts/17">http://some-service.org/api/posts/17</a>
-----	---

- Delete existing post

DELETE	<a href="http://some-service.org/api/posts/17">http://some-service.org/api/posts/17</a>
--------	---

- Replace / modify existing post

PUT/PATCH	<a href="http://some-service.org/api/posts/17">http://some-service.org/api/posts/17</a>
-----------	---





# Accessing GitHub Through HTTP

GitHub REST API

- List user's all public repositories:

GET	<a href="https://api.github.com/users/testnakov/repos">https://api.github.com/users/testnakov/repos</a>
-----	---

- Get all commits from a public repository:

GET	<a href="https://api.github.com/repos/testnakov/softuniada-2016/commits">https://api.github.com/repos/testnakov/softuniada-2016/commits</a>
-----	---

- Get all issues/issue #1 from a public repository

GET	<a href="/repos/testnakov/test-nakov-repo/issues">/repos/testnakov/test-nakov-repo/issues</a>
-----	---

GET	<a href="/repos/testnakov/test-nakov-repo/issues/1">/repos/testnakov/test-nakov-repo/issues/1</a>
-----	---

- Get the first issue from the "**test-nakov-repo**" repository
- Send a **GET** request to:
  - <https://api.github.com/repos/testnakov/test-nakov-repo/issues/:id>
  - Where **:id** is the current issue



- Get all labels for certain issue from a public repository:

GET	<a href="https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/labels">https://api.github.com/repos/testnakov/test-nakov-repo/issues/1/labels</a>
-----	---

- Create a new issue to certain repository (with authentication)

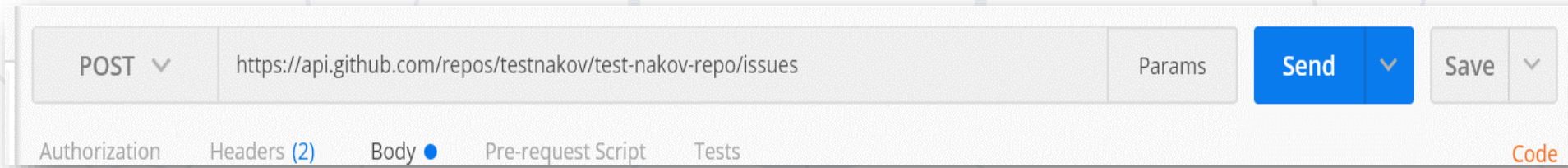
POST	<a href="https://api.github.com/repos/testnakov/test-nakov-repo/issues">https://api.github.com/repos/testnakov/test-nakov-repo/issues</a>
------	---

Headers	Authorization: Basic base64(user:pass)
---------	--

Body	<pre>{"title": "Found a bug",   "body": "I'm having a problem with this."}</pre>
------	--

# Github: Create Issue

- Create an issue when you send a "**POST**" request
- Use your Github account **credentials** to submit the issue



The image features a central dark blue circle with the text 'NPM' in white. Surrounding this central circle is a network diagram consisting of several light gray circles of varying sizes connected by thin gray lines. The background is white.

**NPM**

**Node Package Manager (NPM)**

# NPM: Package Management for JavaScript Projects

- **Node Package Manager (NPM)** is a package manager for JavaScript and Node.js projects
- Simplifies installing, managing, and sharing libraries and tools in web development
- Bundled with Node.js, it offers a command-line interface for various tasks
- Facilitates integrating third-party packages to reuse existing code and speed up development
- Helps manage project dependencies, ensuring required packages are available
- Allows defining and running scripts through the project's package.json file

- **package.json** is a metadata file used in Node.js projects to provide information about the project, its dependencies, and various configurations
- It includes details such as the project's name, version, description, author, license, and more, making it a central place to document essential project information
- integral to the Node.js and NPM ecosystems. When you share your project, including the **package.json** file lets others quickly understand and recreate your project's environment. It also makes it easy for others to install the correct dependencies with a single command (**npm install**)



# Package.json– Example

```
{  
  "name": "test",  
  "version": "1.0.0",  
  "description": "",  
  "main": "app.js",  
  "scripts": {  
    "start": "http-server -a localhost -p 3000 -P http://localhost:3000? -c-1"  
  },  
  "author": "",  
  "license": "ISC",  
  "devDependencies": {  
    "http-server": "0.12.3"  
  }  
}
```

# Install Packages From The package.json

- **Navigate to Project:** Open your command-line interface and navigate to your project's directory
- Run the following command to install the project's dependencies listed in the **package.json**

```
npm install
```

- NPM will read the **package.json** file and install all the dependencies

# Problem: Book Library

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <meta http-equiv="X-UA-Compatible" content="ie=edge">
7   <title>Books</title>
8   <link rel="stylesheet" href="./styles.css">
9 </head>
10 <body>
11   <button id="loadBooks">LOAD ALL BOOKS</button>
12   <table>
13     <thead>
14       <tr>
15         <th>Title</th>
16         <th>Author</th>
17         <th>Action</th>
```

LOAD ALL BOOKS

Title	Author	Action
Book 1	Author 1	<div>EditDelete</div>
Book 2	Author 2	<div>EditDelete</div>

FORM

TITLE

Title...

AUTHOR

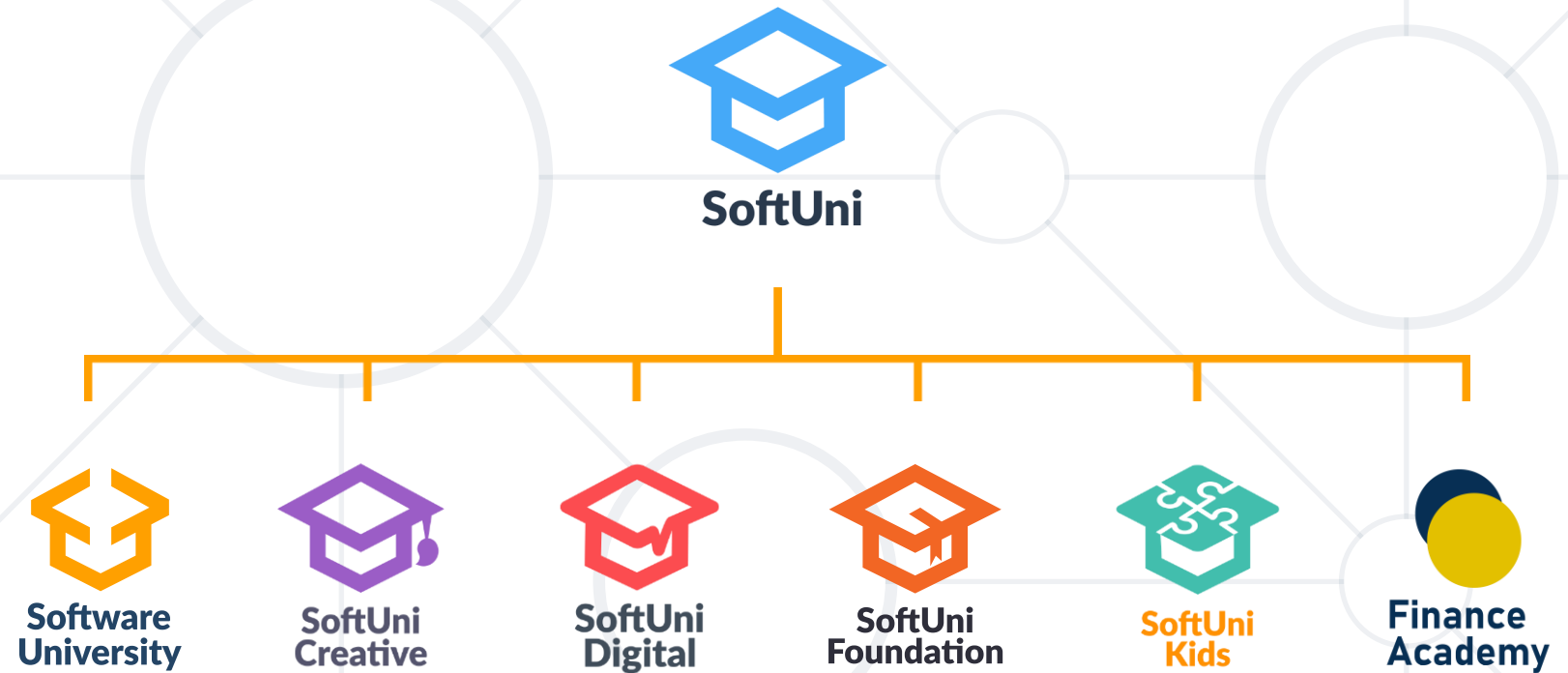
Author...

Submit

- **HTTP** is text-based request-response protocol
- **RESTful** services address resources by URL
- Provide **CRUD** operations over HTTP
- **RESTful** services address resources by URL
- **NPM** is used for installing, managing, and sharing libraries and tools



# Questions?



# SoftUni Diamond Partners



- Software University – High-Quality Education, Profession and Job for Software Developers
  - [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)
- Software University Foundation
  - [softuni.foundation](http://softuni.foundation)
- Software University @ Facebook
  - [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)



- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://softuni.org>
- © Software University – <https://softuni.bg>

