

# **JS QA automation framework for API**

## Table of Contents

1 Test automation theory.....	3
2 Linux basics.....	3
3 Linux basics practice.....	3
4 SSH keys.....	3
5 Test project setup.....	4
6 Syntax for common test runners / Work with a new branch. Install BABEL. Pull request. First tests.....	8
7 Syntax for HTTP client / First API tests.....	12
7.1 Installation Supertest.....	12
7.2 Creating a new specification's file (spec file) for authorization tests.....	12
8 Environment variables.....	14
8.1 Installation dotenv npm package.....	14
8.1 Work with environment variables.....	14
9 Test runner (mocha) hooks.....	15
9.1 Hooks.....	15
9.2 Global hooks.....	16
10 Wrappers (helpers) for API tests.....	17
11 API tests practice.....	18
12 Setting up a mock server.....	18
13 CI/CD theory.....	18
14 Integrating API tests with GitHub Actions.....	18

# 1 Test automation theory

Lots of super boring stuff should be presented over here!

## 2 Linux basics

ls, cd, mkdir, cp, mv, rm, cat, nano, grep, regExp

## 3 Linux basics practice

ls, cd, mkdir, cp, mv, rm, cat, nano, grep, regExp

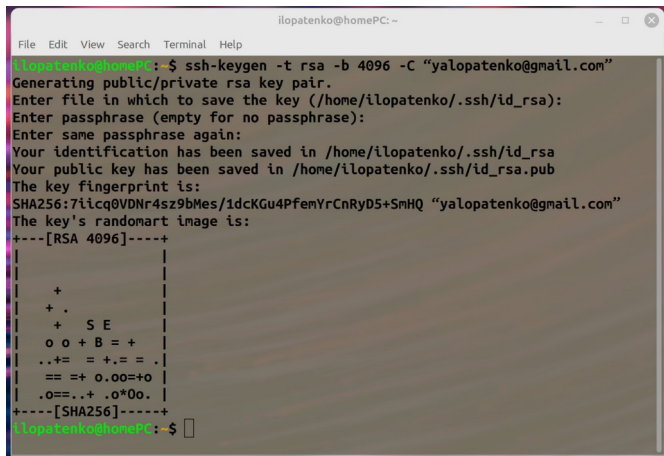
## 4 SSH keys

Open an terminal

CD to a user home directory

Run next command:

CLI=> `ssh-keygen -t rsa -b 4096 -C "ENTER_HERE_YOUR_EMAIL_OR_WHATEVER"`



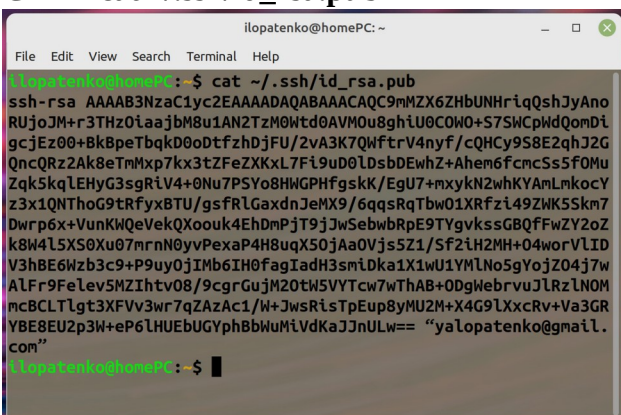
```
ilopatenko@homePC: ~  
File Edit View Search Terminal Help  
ilopatenko@homePC:~$ ssh-keygen -t rsa -b 4096 -C "yalopatenko@gmail.com"  
Generating public/private rsa key pair.  
Enter file in which to save the key (/home/ilopatenko/.ssh/id_rsa):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in /home/ilopatenko/.ssh/id_rsa  
Your public key has been saved in /home/ilopatenko/.ssh/id_rsa.pub  
The key fingerprint is:  
SHA256:71lcq0VDNr4sz9bMes/1dcKGu4PfemYrCnRyD5+SmHQ "yalopatenko@gmail.com"  
The key's randomart image is:  
+--[RSA 4096]-----+  
+  
+ . S E  
+ o o + B = +  
...+ = + + = .  
==+ 0.00=+0  
..0=..+ .0*00.  
+-----[SHA256]-----+  
ilopatenko@homePC:~$
```

Go to gitHub. Go to settings <https://github.com/settings/profile>

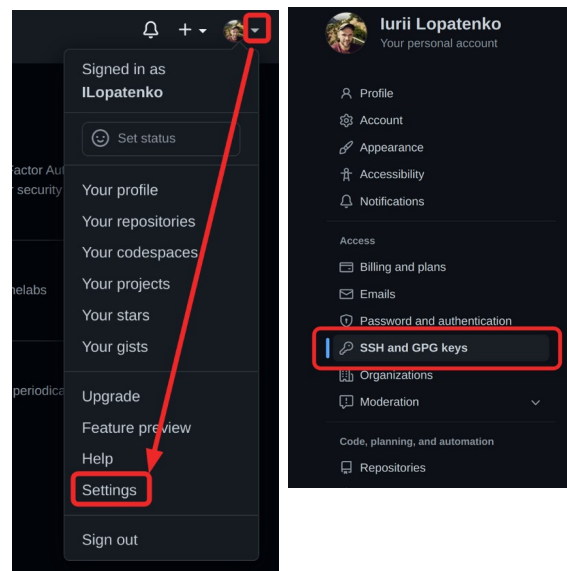
Go to SSH and GPG keys <https://github.com/settings/keys>

Copy SSH key to buffer:

CLI=> `cat ~/.ssh/id_rsa.pub`



```
ilopatenko@homePC: ~  
File Edit View Search Terminal Help  
ilopatenko@homePC:~$ cat ~/.ssh/id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQAC9mMZX6ZHbUNHrlqQshJyAno  
RUjoJM+r3THz0iaaBm8u1AN2TzM0Wtd0AVM0u8ghiU0COW0+S7SWCpWdQomDi  
gcjEz00+Bk8pTbqkD00DtfzhDjFU/2vA3K7QWftrV4nyf/cQHCy9S8E2qhJ2G  
QncQRz2Ak8eTmMxp7kx3tZFeZXKxL7Fi9u00LdsbDEwhZ+Ahem6fcmcSs5F0Mu  
Zqk5kqLEHvG3sgriv4+0Nu7PSYo8HWGPHfgskK/EgU7+mxykN2whKYAmLnkocY  
z3x1QNTHoG9trfyxBTU/gsfRLGaxdnJeMX9/6qqsRqTbw01XRfzi49ZWK5SkM7  
DwRp6x+VunKWQeVekQXoouk4EhDmPjT9jJwSebwRpe9TYgvkssGBQfFwZY2oZ  
k8W4LSX0XU07mrnN0yvPexaP4H8uqX50jAa0Vjs5Z1/Sf2LH2MH+04worVLID  
V3hBE6WzB3c9+P9uy0jIMb6IH0fagIadH3smiDka1X1wU1YMLNo5gYoJZ04j7w  
ALFr9FeLev5MZHItv08/9cgrGuJM20tW5VYTw7wThAB+0DgWebrvuJlRzLNOM  
mcBCLTlgt3XFVv3wr7qZaZAc1/W+JwsRisTepE8yMU2M+X4G9LXcRv+Va3GR  
YBE8EU2p3M+eP6LHUEbUGYphBbWuMivdKaJJnULw== "yalopatenko@gmail.  
com"  
ilopatenko@homePC:~$
```



Click the button **Add SSH** and Insert an SSH key from buffer and click the button **Add SSH key** and confirm (re enter your gitHub password).

## SSH keys / Add new

Title

This is an SSH key for homePC

Key


```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQCAQC9mMZx6ZHbUNHriqQshJvAnoRUjoJM+r3THzOiaajbM8u1AN2TzM0Wtd
0AVMOu8ghiU0COWO+S7SWCpWdQomDiacjEz00+BkBpeTbqkD0oDtfzhDjFU/2vA3K7QWftrV4nyf/cQHcy9S8E2qhJ
2GQncQRz2Ak8eTmMxp7kx3tZFeZXKxL7Fi9uD0IDsbDEwhZ+Ahem6fcmcSs5fOMuZqk5kqLEHyG3sgRiV4+0Nu7PSYo
8HWGPHfqsK/EgU7+mxykN2whKYAmLmkocYz3x1QNTHoG9tRfyzBTU/gsfRIGaxdnJeMX9/6qqsRqTbwO1XRfzi49Z
WK5Skm7Dwrp6x+VunKWQeVekQXoouk4EhDmPJt9jJwSebwRpE9TYgvkssGBQfWZY2oZk8W4l5XS0Xu07mrnN0y
vPexaP4H8uqX5OjAaOVjs5Z1/Sf2iH2MH+O4worVlIDV3hBE6Wzb3c9+P9uyOjIMb6IH0fagladH3smiDka1X1wU1YMIN
o5gYojZO4j7wAlFr9Felev5MZlhtvO8/9cgrGuJM2OtW5VYTw7wThAB+ODgWebrvuJIRzlNOMmcBCLTlgt3XfVv3wr7qZ
AzAc1/W+JwsRisTpEup8yMU2M+X4G9lXxcRv+Va3GRYBE8EU2p3W+eP6lHUEbUGYphBbWuMiVdKaJJnULw==
"yalopatenko@gmail.com"
```

Add SSH key

## SSH keys

New SSH key

This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.



**This is an SSH key for homePC**

SHA256:7iicq0VDNr4sz9bMes/1dcKGu4PfemYrCnRyD5+SmHQ

Added on Feb 5, 2022

Never used — Read/write

Delete

Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#).

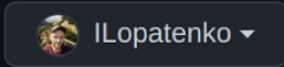
## 5 Test project setup

Create a new gitHub repository for a new project (JS automation framework for API testing)

# Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner \*



Repository name \*

/ QAA-API-framework ✓

Great repository names are short and memorable. Need inspiration? How about **solid-system**?

Description (optional)

JS test automation framework for API



**Public**

Anyone on the internet can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.

**Initialize this repository with:**

Skip this step if you're importing an existing repository.



**Add a README file**

This is where you can write a long description for your project. [Learn more.](#)



**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)


.gitignore template: Node ▾



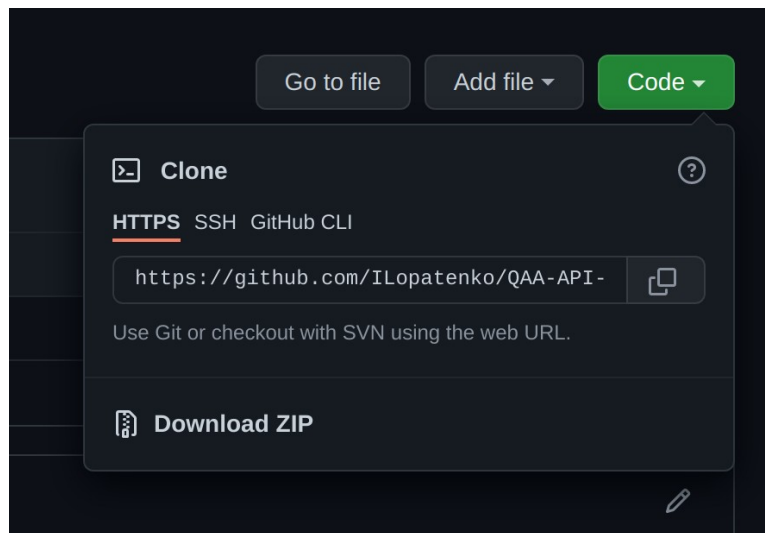
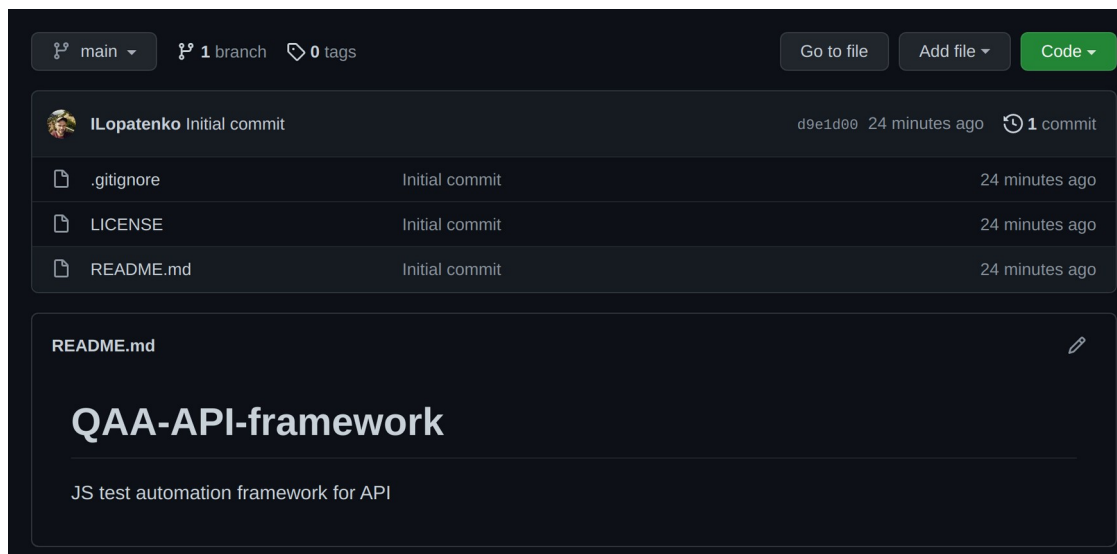
**Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

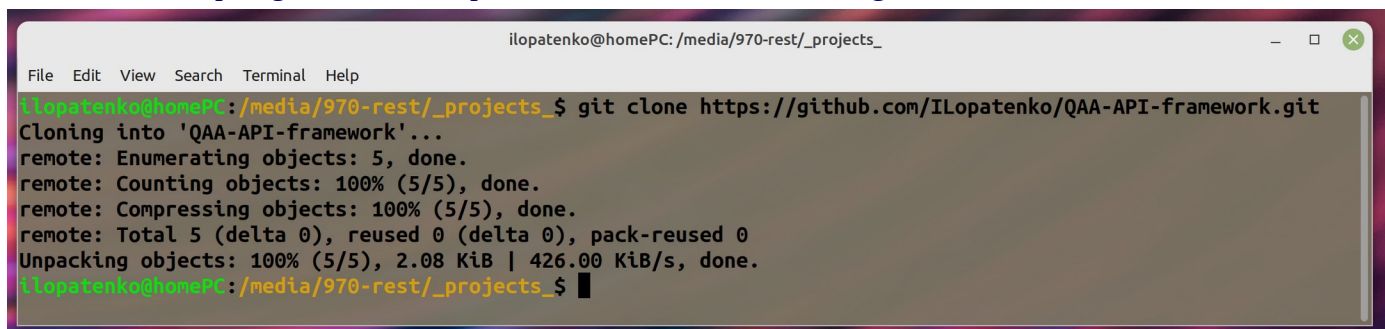
Create repository



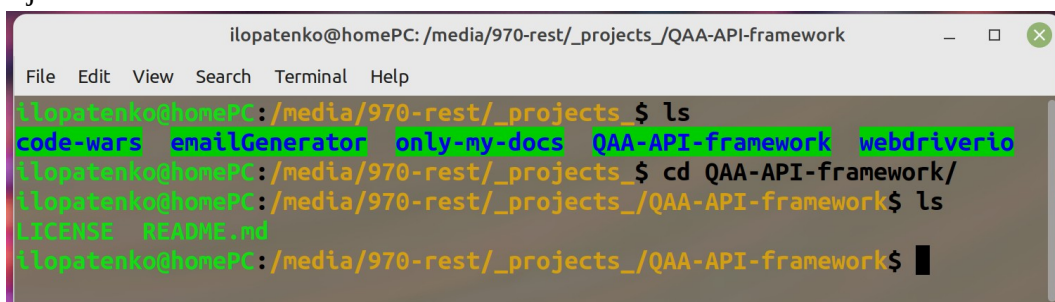
Copy your new repository's URL

Open terminal and **CD** to a directory where you want to keep your framework and clone it

CLI=> **git clone** <https://github.com/Ilopatenko/QAA-API-framework.git>



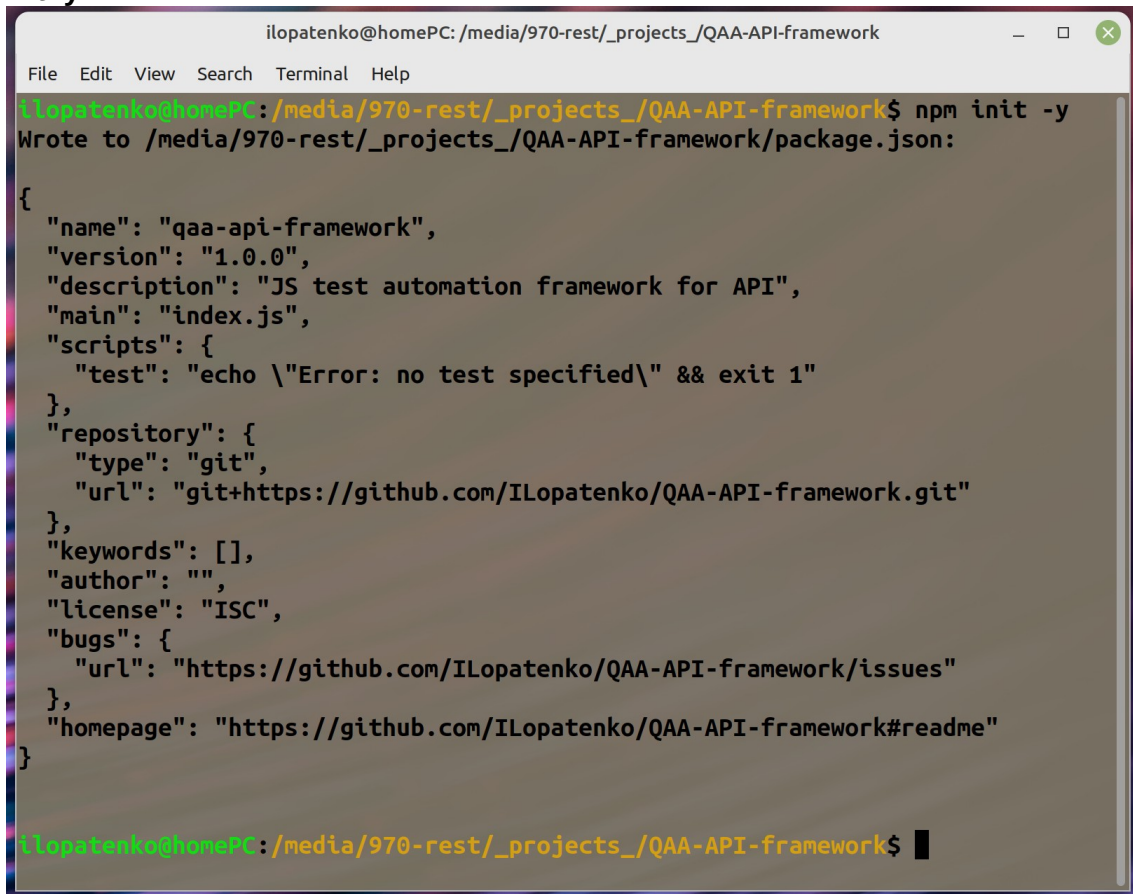
**CD** into your project's folder





Initialize a new NPM project

CLI=> **npm init -y**

A screenshot of a terminal window titled 'ilopatenko@homePC: /media/970-rest/\_projects\_/QAA-API-framework'. The terminal shows the command 'npm init -y' being executed. The output indicates that a package.json file was written to the current directory. The contents of the package.json file are displayed as a JSON object with the following fields: name, version, description, main, scripts, repository, keywords, author, license, bugs, and homepage. The terminal prompt is now ready for the next command.

```
ilopatenko@homePC: /media/970-rest/_projects_/QAA-API-framework$ npm init -y
Wrote to /media/970-rest/_projects_/QAA-API-framework/package.json:

{
  "name": "qaa-api-framework",
  "version": "1.0.0",
  "description": "JS test automation framework for API",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "repository": {
    "type": "git",
    "url": "git+https://github.com/ILopatenko/QAA-API-framework.git"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "bugs": {
    "url": "https://github.com/ILopatenko/QAA-API-framework/issues"
  },
  "homepage": "https://github.com/ILopatenko/QAA-API-framework#readme"
}

ilopatenko@homePC: /media/970-rest/_projects_/QAA-API-framework$
```

## 6 Syntax for common test runners / Work with a new branch. Install BABEL. Pull request. First tests

<https://dev.to/bormando/babel-setup-for-rest-api-tests-1dhf>

Create a new branch babel-setup (and switch to it)

CLI=> **git checkout -B babel-setup**

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

iIopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git checkout -B babel-setup
Switched to a new branch 'babel-setup'
iIopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Install Babel

CLI=> **npm i -D @babel/cli @babel/core @babel/plugin-transform-runtime @babel/preset-env @babel/register**

Create a new file .babelrc in the root directory of your project

CLI=> **touch .babelrc**

And add this text

```
{
  "presets": ["@babel/preset-env"],
  "plugins": [
   ["@babel/transform-runtime"]
  ]
}
```

CLI=> **nano .babelrc**

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

GNU nano 4.8
{
  "presets": ["@babel/preset-env"],
  "plugins": [
   ["@babel/transform-runtime"]
  ]
}
```

Now we can add all the changes to a STAGE. Create a new commit and push it to the repository.

CLI=> **git add .**

CLI=> **git commit -m 'Added Babel'**

CLI=> **git push --set-upstream origin babel-setup**

```
iIopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git push --set-upstream origin babel-setup
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 16 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 44.15 KiB | 5.52 MiB/s, done.
Total 5 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote:
remote: Create a pull request for 'babel-setup' on GitHub by visiting:
remote:   https://github.com/Ilopatenko/QAA-API-framework/pull/new/babel-setup
remote:
To https://github.com/Ilopatenko/QAA-API-framework.git
 * [new branch]   babel-setup -> babel-setup
Branch 'babel-setup' set up to track remote branch 'babel-setup' from 'origin'.
iIopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```



Go to your gitHub repository

The screenshot shows the GitHub repository page for 'babel-setup'. At the top, a notification bar indicates 'babel-setup had recent pushes less than a minute ago' with a green 'Compare & pull request' button. Below this, the repository navigation bar shows 'main' as the selected branch, '1 branch', and '0 tags'. There are buttons for 'Go to file', 'Add file', and 'Code'. The commit history table shows a recent commit by 'ILOpatenko' titled 'The new NPM project was initialized' (commit hash f3809d3, 30 minutes ago, 2 commits). The file list includes '.gitignore', 'LICENSE', 'README.md', 'package-lock.json', and 'package.json'. The 'README.md' file is expanded, showing the title 'QAA-API-framework' and the description 'JS test automation framework for API'.

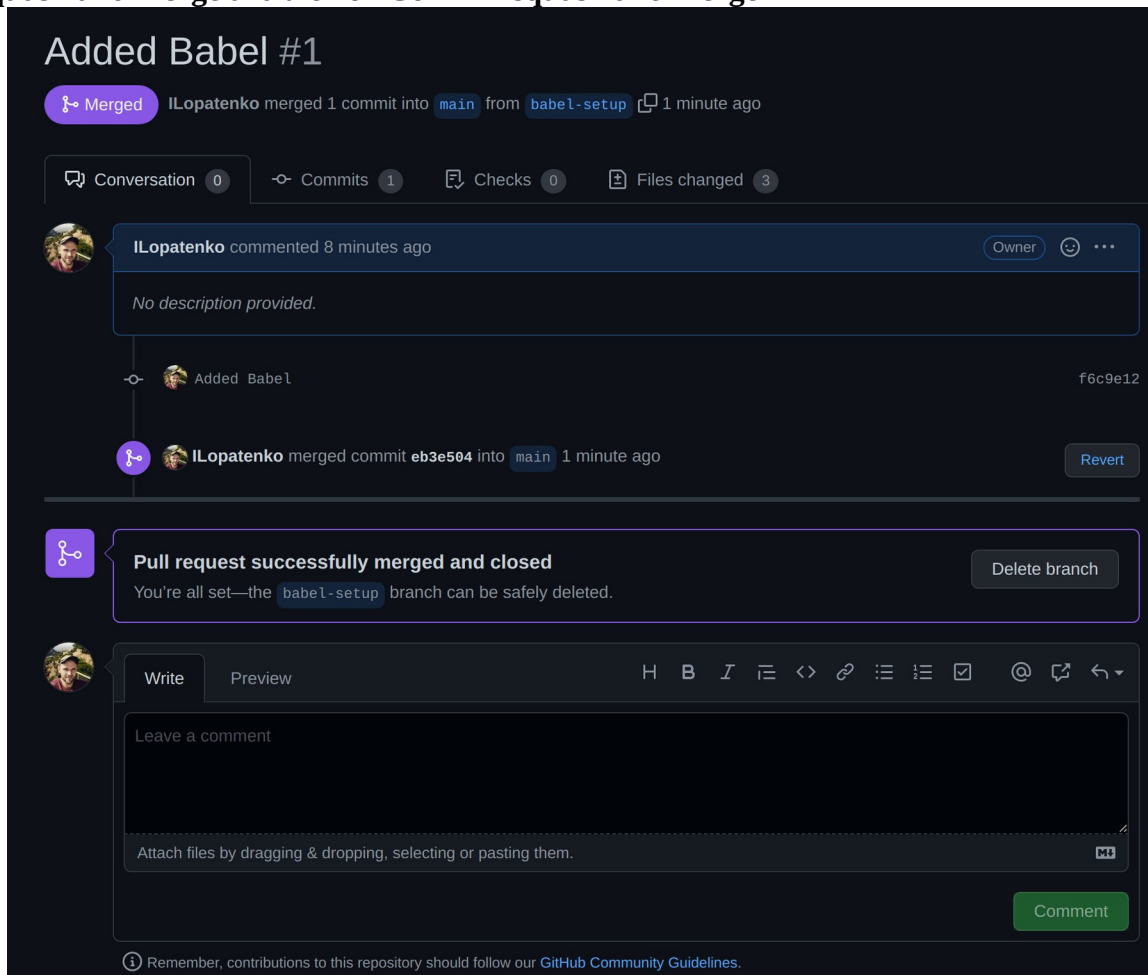
Click on **Compare & pull request** button

Check all the changes and if everything is good click on **Create pull request** button

The screenshot shows the 'Open a pull request' page. The title is 'Open a pull request' with a subtitle 'Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).' The comparison bar shows 'base: main' and 'compare: babel-setup', with a green checkmark and the text 'Able to merge. These branches can be automatically merged.' The pull request title is 'Added Babel'. Below the title are 'Write' and 'Preview' tabs. The 'Write' tab is active, showing a large text area for the pull request description with a placeholder 'Leave a comment'. Below the text area is a section for attaching files. At the bottom right is a green 'Create pull request' button. A footer note states: 'Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).'

Change default merge options to **Squash and merge**

Click on **Squash and merge** and then on **Confirm squash and merge**



Now we can delete branch **babel-setup** by click on **Delete branch** button

Change a branch to **main** in your IDE

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Pull all the changes to our local main branch from gitHub

CLI=> **git pull**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git pull
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 1), reused 4 (delta 1), pack-reused 0
Unpacking objects: 100% (5/5), 44.52 KiB | 337.00 KiB/s, done.
From https://github.com/ILOpatenko/QAA-API-framework
   f3809d3..eb3e504  main       -> origin/main
Updating f3809d3..eb3e504
Fast-forward
 .babelrc      |      6 +
 package-lock.json | 4866 ++++++
 package.json   |      9 +-
 3 files changed, 4878 insertions(+), 3 deletions(-)
 create mode 100644 .babelrc
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Create a new branch **first-tests**

CLI=> **git checkout -B first-tests**

Install Mocha and Chai packages

CLI=> **npm i -D mocha chai**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm i -D mocha chai
added 62 packages, and audited 278 packages in 2s

31 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Create a new directory **specs**

CLI=> **mkdir specs**

Create a new file **example.spec.js** in directory **specs**

CLI=> **touch specs/example.spec.js**

Open **specs/example.spec.js** with IDE import **Chai** and **Mocha** and start writing first tests

```
JS example.spec.js U X
specs > JS example.spec.js > ...
1  import { expect } from 'chai';
2
3  describe('Math functions', () => {
4    it('A + B = C', () => {
5      const a = 4;
6      const b = 7;
7      const c = a + b;
8      expect(c).to.eq(11);
9    });
10 });
11
```

Open **package.json** file and make some changes in test script:

```
"scripts": {
  "test": "npx mocha --config .mocharc.js"
},
```

Create a new file **.mocharc.js**

CLI=> **touch .mocharc.js**

And add next text

```
module.exports = {
  require: '@babel/register',
  spec: 'specs/**/*.spec.js',
};
```

Run a first test

CLI=> **npm run test**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm run test
> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js

Math functions
  ✓ A + B = C

1 passing (3ms)
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

Create a new commit, push it to gitHub, Create a pull request, merge pull request and pull all the changes to a local computer

## 7 Syntax for HTTP client / First API tests

### 7.1 Installation Supertest

For this lesson I'm going to create a new branch in my project: first-api-tests and switch to the new branch

CLI=> **git checkout -b first-api-tests**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ git checkout -b first-api-tests
Switched to a new branch 'first-api-tests'
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

I'm going to use SUPERTEST as an HTTP client. Install the supertest as a DEV dependency.

CLI=> **npm i -D supertest**

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm i -D supertest
added 28 packages, and audited 306 packages in 1s

37 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$
```

### 7.2 Creating a new specification's file (spec file) for authorization tests

Delete example spec **example.spec.js**

Create a new file **auth.spec.js** inside **/QAA-API-framework/specs/**

CLI=> **touch specs/auth.spec.js**

Open this new file in VSCode and add next content:

```
//Imports block
import { expect } from 'chai';
import supertest from 'supertest';
//
//describe is a test suit that can contain another test sub suits or test cases
describe('Auth', () => {
  //Create a request object
  const request = supertest('https://paysis.herokuapp.com');

  //it is a test case with a couple of assertions
  it('Successful login (happy path, positive tests)', () => {
    //Use inner methods to add all the parameters to a request using chaining methods
    request
      //Setup a request method - POST and an endpoint - /auth
      .post('/auth')
      //Setup payload - object with 2 keys - login and password (and their values)
      .send({ login: 'adminius', password: 'supers3cret' })
      //Tests block that can handle error or response
      .end((err, res) => {
        //Test that status code in response equals to 200
        expect(res.statusCode).to.eq(200);
        //Test that response has body, body has token key and that token key is not undefined
        expect(res.body.token).not.to.be.undefined;
      }));
  });
});
```

//it is a test case with a couple of assertions

```
it('Unsuccessful login (unhappy path, negative tests)', () => {
```

//Use inner methods to add all the parameters to a request using chaining methods

request

//Setup a request method - POST and an endpoint - /auth

.post('/auth')

//Setup payload - object with 2 keys - login and password (and their values)

.send({ login: 'wrongLogin', password: 'wrongPassword' })

//Tests block that can handle error or response

.end((err, res) => {

//Test that status code in response equals to 404

expect(res.statusCode).to.eq(404);

//Test that error message equals to 'Wrong login or password.'

expect(res.body.message).to.eq('Wrong login or password.'))));

```
JS auth.spec.js X
specs > JS auth.spec.js > ...
1 //Imports block
2 import { expect } from 'chai';
3 import supertest from 'supertest';
4 //
5 //describe is a test suite that can contain another test sub suits or test cases
6 describe('Auth', () => {
7   //Create a request object
8   const request = supertest('https://paysis.herokuapp.com');
9
10  //it is a test case with a couple of assertions
11  it('Successfull login (happy path, positive tests)', () => {
12    //Use inner methods to add all the parameters to a request using chaining methods
13    request
14      //Setup a request method - POST and an endpoint - /auth
15      .post('/auth')
16      //Setup payload - object with 2 keys - login and password (and their values)
17      .send({ login: 'adminius', password: 'supers3cret' })
18      //Tests block that can handle error or response
19      .end((err, res) => {
20        //Test that status code in response equals to 200
21        expect(res.statusCode).to.eq(200);
22        //Test that response has body, body has token key and that token key is not undefined
23        expect(res.body.token).not.to.be.undefined;
24      });
25  });
26
27  //
28  //it is a test case with a couple of assertions
29  it('Unsuccessfull login (unhappy path, negative tests)', () => {
30    //Use inner methods to add all the parameters to a request using chaining methods
31    request
32      //Setup a request method - POST and an endpoint - /auth
33      .post('/auth')
34      //Setup payload - object with 2 keys - login and password (and their values)
35      .send({ login: 'wrongLogin', password: 'wrongPassword' })
36      //Tests block that can handle error or response
37      .end((err, res) => {
38        //Test that status code in response equals to 404
39        expect(res.statusCode).to.eq(404);
40        //Test that error message equals to 'Wrong login or password.'
41        expect(res.body.message).to.eq('Wrong login or password.');
42      });
43  });
44 });
45
```

Run all the specs with 1 command:

CLI=> **npm run test**

```
ilopatenko@homePC:/media/970-rest/_projects/QAA-API-framework$ npm run test
> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js

Auth
  ✓ Successfull login (happy path, positive tests)
  ✓ Unsuccessfull login (unhappy path, negative tests)

2 passing (16ms)

ilopatenko@homePC:/media/970-rest/_projects/QAA-API-framework$
```

So far so good. Make a commit, push it to gitHub, create a pull request, merge it. Pull all the changes.

## 8 Environment variables

For this lesson I'm going to create a new branch in my project: **env-vars** and switch to the new branch

CLI=> **git checkout -b env-vars**

### 8.1 Installation dotenv npm package

Dotenv is a zero-dependency module that loads environment variables from a **.env** file into **process.env**.

CLI=> **npm i -D dotenv**

Create a new file **.env** in root project's folder to store all the global variables

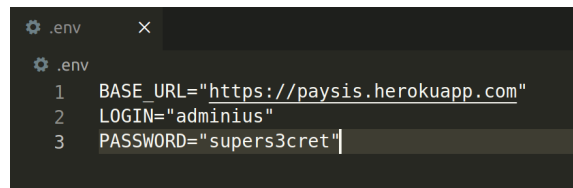
CLI=> **touch .env**

Add **.env** file to **.gitignore** file

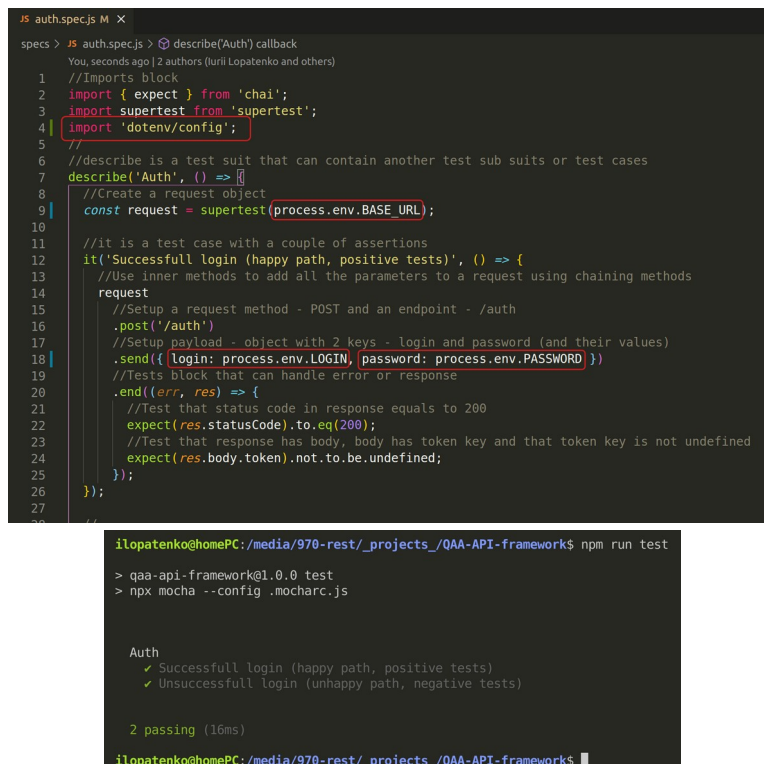
A screenshot of a code editor showing the .gitignore file. The file contains several entries: .gitignore, .yarn-integrity, # dotenv environment variables file, .env, and .env.test. The .env entry is highlighted with a red box.

### 8.1 Work with environment variables

Add all the variables (with their values) to **.env** file:

A screenshot of a code editor showing the .env file. The file contains three lines of text: 1 BASE\_URL="https://paysis.herokuapp.com", 2 LOGIN="adminius", and 3 PASSWORD="supers3cret". The entire content is highlighted with a red box.

Now I'm going to import dotenv package to my specs and get rid of all hard coded values inside all the specs.

A screenshot showing two parts of a development environment. The top part is a code editor showing the auth.spec.js file. It includes imports for expect, supertest, and dotenv/config. The test suite describes an 'Auth' endpoint with a 'Successful login (happy path, positive tests)' case. The test code uses process.env.BASE\_URL, process.env.LOGIN, and process.env.PASSWORD. The bottom part is a terminal window showing the command npm run test and the output of the test suite. The output shows that the 'Auth' test suite passed, with 2 tests passing in 16ms.

Make a commit, push, merge pull request, pull all the changes to local main branch.



## 9 Test runner (mocha) hooks

For this lesson I'm going to create a new branch in my project: **env-vars** and switch to the new branch

CLI=> **git checkout -b mocha-hooks**

### 9.1 Hooks

```
JS auth.spec.js M X
specs > JS auth.spec.js > ...
You, seconds ago | 2 authors (You and others)
1 //Imports block
2 import { expect } from 'chai';
3 import supertest from 'supertest';
4 import 'dotenv/config';
5 //This describe is a main test suite
6 describe('Auth test suite', () => {
7   //Create a request object
8   const request = supertest(process.env.BASE_URL);
9   //Create a new variable to store a response from server
10  let result;
11  describe('Successful login sub suite (happy path, positive tests)', () => {
12    //BEFORE hook - will be runned 1st (before all other suits/tests)
13    //this hook will send a request to a server and save a response from a server to a request variable
14    before(async () => {
15      //Send async request
16      await request
17        //Setup a request method - POST and an endpoint - /auth
18        .post('/auth')
19        //Setup payload - object with 2 keys - login and password (and their values)
20        .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
21        //Save a response from server to result variable
22        .then((res) => {
23          result = res;
24        });
25    });
26    it('Checking that response status code is 200', () => {
27      expect(result.statusCode).to.eq(200);
28    });
29    it('Checking that response contains an authorization token', () => {
30      expect(result.body.token).not.to.be.undefined;
31    });
32  });
33  describe('Unsuccessful login sub suite (unhappy path, negative tests)', () => {
34    //BEFORE hook - will be runned 1st (before all other suits/tests)
35    //this hook will send a request to a server and save a response from a server to a request variable
36    before(async () => {
37      //Send async request
38      await request
39        //Setup a request method - POST and an endpoint - /auth
40        .post('/auth')
41        //Setup payload - object with 2 keys - login and password (and their values)
42        .send({ login: 'tralala', password: 'trulala' })
43        //Save a response from server to result variable
44        .then((res) => {
45          result = res;
46        });
47    });
48    it('Checking that response status code is 404', () => {
49      expect(result.statusCode).to.eq(404);
50    });
51    it('Checking that error message is "Wrong login or password."', () => {
52      expect(result.body.message).to.eq('Wrong login or password.');
```

```
ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ npm run test
> qaa-api-framework@1.0.0 test
> npx mocha --config .mocharc.js

Auth test suite
  Successful login sub suite (happy path, positive tests)
    ✓ Checking that response status code is 200
    ✓ Checking that response contains an authorization token
  Unsuccessful login sub suite (unhappy path, negative tests)
    ✓ Checking that response status code is 404
    ✓ Checking that error message is "Wrong login or password."

  4 passing (772ms)

ilopatenko@homePC:/media/970-rest/_projects_/QAA-API-framework$ +
```

## 9.2 Global hooks

Create a new directory **config** and a new file **setup.js** inside to store global mocha hooks

CLI=> **mkdir config && touch config/setup.js**

I'm going to move successful log in action to a global Mocha hook (to be able so save a token from response to a .env variable)

```
JS setup.js U X
config > JS setup.js > ...
1 import 'dotenv/config';
2 import supertest from 'supertest';
3
4 before(async () => {
5   //Create a request object
6   const request = await supertest(process.env.BASE_URL);
7   await request
8     //Setup a request method - POST and an endpoint - /auth
9     .post('/auth')
10    //Setup payload - object with 2 keys - login and password (and their values)
11    .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
12    //Save a response from server to result variable
13    .then((res) => {
14      process.env['TOKEN'] = res.body.token;
15    });
16 });
17
```

```
JS auth.spec.js M
specs > JS auth.spec.js > describe('Auth test suite') callback > describe('Unsuccessful login sub suite (unhappy path, negative tests)')
You, seconds ago | 2 authors (You and others)
1 import { expect } from 'chai';
2 import supertest from 'supertest';
3 //This describe is a main test suite
4 describe('Auth test suite', () => {
5   //Create a request object
6   const request = supertest(process.env.BASE_URL);
7   //Create a new variable to store a response from server
8   let result;
9   describe('Successful login sub suite (happy path, positive tests)', () => {
10    //BEFORE hook - will be runned 1st (before all other suits/tests)
11    before(async () => {
12      //Send async request
13      await request
14        //Setup a request method - POST and an endpoint - /auth
15        .post('/auth')
16        //Setup payload - object with 2 keys - login and password (and their values)
17        .send({ login: process.env.LOGIN, password: process.env.PASSWORD })
18        //Save a response from server to result variable
19        .then((res) => {
20          result = res;
21        });
22    });
23    it('Checking that response status code is 200', () => {
24      expect(result.statusCode).to.eq(200);
25    });
26    it('Checking that response contains an authorization token', () => {
27      expect(result.body.token).not.to.be.undefined;
28    });
29  });
30  describe('Unsuccessful login sub suite (unhappy path, negative tests)', () => {
31    //BEFORE hook - will be runned 1st (before all other suits/tests)
32    before(async () => {
33      //Send async request
34      await request
35        //Setup a request method - POST and an endpoint - /auth
36        .post('/auth')
37        //Setup payload - object with 2 keys - login and password (and their values)
38        .send({ login: 'tralala', password: 'trulala' })
39        //Save a response from server to result variable
40        .then((res) => {
41          result = res;
42        });
43    });
44    it('Checking that response status code is 404', () => {
45      expect(result.statusCode).to.eq(404);
46    });
47    it('Checking that error message is "Wrong login or password."', () => {
48      expect(result.body.message).to.eq('Wrong login or password.')}));
49  });
50
```

## **10 Wrappers (helpers) for API tests**

**11 API tests practice**

**12 Setting up a mock server**

**13 CI/CD theory**

**14 Integrating API tests with GitHub Actions**