KEVIN THE HACK BUSTER – API DOCS

version: *1.0*

by *Patryk Ryak*

Table of contents:

INTRODUCTION

BASIC INFO

This is first public version of this **API**.

Kevin's api is organized around **REST**.

For presentation purposes we will use the base URL as:

```
https://kevin.com/api
```

NOTE: Final URL will differ.

Please, report to me if you find a bug! 🐛

API LEVELS

API consists of three level of access:

- Root level
- Public level

- Company level

Root level allows to add and delete companies. It can also set API public key for clients to autenticate. It is not supposed to have GUI. It is designed to be used by company that hosts API and wants root access.

Public level is an access level for client apps (game distributed to many platforms). It is used for sending selected questions for quiz and reciving data about answers. In order to make sure only trusted clients can use api, "*PublicKey*" is required to connect. It is also usefull to forbid older versions of client apps from connecting to the API. Public level request must include statment showed below:

```
{
  "publicKey": "testKey"
}
```

Company Level is the most advanced of all layers. It uses JWT cookie to authenticate clients. It is designed for administration of a single client company account. It allows to manage questions and see telemetry data gathered from answered questions.

Errors

How API handles errors?

Every error has specific message that will further explain a problem.

Errors table:

Error	Code	Meaning
200	Success	Everything went well
404	Not Found	No such route
403	Forbidden	Permission denied
400	syntax error	Something is wrong with .json (message will specify)
500	Server error	Cannot process something (message will specify)

Error example:

```
{
  "error": "403",
  "message": "Access Denied"
}
```

ENDPOINTS

ROOT LEVEL:

Set public key:

Path:

```
https://kevin.com/api/root/setApiPublicKey
```

Type: POST

Request:

```
{
  "rootKey": "RootExampleKey",
  "newPublicKey": "testKey"
}
```

Response:

```
{
  "Success": "200",
  "message": "Public key updated successfully",
  "previous": "exampleKey",
  "current": "testKey"
}
```

Info: All fields in request are required

Adding a company:

Path:

```
https://kevin.com/api/root/addCompany
```

Type: POST

Request:

```
{
  "rootKey": "RootSuperSecretKey",
  "companyName": "TestCompany",
  "password": "example"
}
```

Response:

```
{
  "status": "200",
  "message": "Company added"
}
```

Info: All fields in request are required

Deleting a company

Path:

```
https://kevin.com/api/root/delCompany
```

Type: POST

Request:

```
{
  "rootKey": "RootSuperSecretKey",
  "companyName": "TestCompany"
}
```

Response:

```
{
  "status": "200",
  "message": "Company deleted"
}
```

Info: All fields in request are required

PUBLIC LEVEL:

Requesting a quiz:

Path:

```
https://kevin.com/api/quiz
```

Type: POST

Request:

```
{
  "publicKey": "testKey",
  "company": "TestCompany"
}
```

"company" field is a data that user has to provide. If it's correct then the access to a quiz will be granted and response will end with success.

Response:

```
{
  "status": "200",
  "questions": [
    {
      "QID": "c9d14faa-6eb8-430d-8a05-d328010a7c85",
      "Options": [
        [
          "Option 1",
          true
        ],
        [
          "Option 2",
          false
        ]
```

```
        ],
        "Company": "TestCompany",
        "InUse": true,
        "Category": "Examples",
        "Body": "My question"
      }
    ]
}
```

Successful request will return status and array of questions. In this example only one question has been associated with this company. Learn more about question structure.

Submit an answer:

Path:

```
https://kevin.com/api/answer
```

Type: POST

Request:

```
{
  "publicKey": "testKey",
  "company": "TestCompany",
  "qid": "c9d14faa-6eb8-430d-8a05-d328010a7c85",
  "platform": "PC",
  "options": [false, true]
}
```

In order to submit an answer about question qou need to provide its QID (Question ID) and array of selected options by user in order as provided in .json from quiz request.

For example:

There were two options:

```
"Options": [
        [
          "Option 1",
          true
        ],
        [
          "Option 2",
          false
        ]
    ]
```

So submited array of options looks like: "options": (false, true). That means that user selected "Option 2" as an answer.

Response:

```
{
  "status": "200",
  "message": "Answer submited"
}
```

Info: All fields in request are required

COMPANY LEVEL:

Login:

Path:

```
https://kevin.com/api/company/auth/login
```

Type: POST

Request:

```
{
  "login": "TestCompany",
  "password": "example"
}
```

Response:

```
{
  "status": "200",
  "message": "Logged as TestCompany"
}
```

Logout:

Path:

```
https://kevin.com/api/company/auth/logout
```

Type: POST

Request: No data needs to be sent.

Response:

```
{
  "status": "200",
  "message": "Logged out"
}
```

Setting a new password:

Path:

```
https://kevin.com/api/company/newPassword
```

Type: POST

Request:

```
{
  "login": "TestCompany",
  "password": "example"
}
```

Response:

```
{
  "status": "200",
  "message": "Logged as TestCompany"
}
```

Telemetry:

Path:

```
https://kevin.com/api/company/telemetry
```

Type: POST

Request: No data needs to be sent.

Response:

```
{
  "status": "200",
```

```
  "telemetry": [
    {
      "Platform": "PC",
      "Date": "2024-02-03T20:34:41.564Z",
      "QID": "c9d14faa-6eb8-430d-8a05-d328010a7c85",
      "Options": [
        false,
        true
      ],
      "Company": "TestCompany"
    }
  ]
}
```


Response returns telemetry array that contains previously sent values. They are not filtered to reduce ammount of requests to database. Processing and presenting that data is a front-end job. To get information about all questionss associated with this company use request showed in the next paragraph.

Getting questions:

Path:

```
https://kevin.com/api/company/questions
```

Type: POST

Request: No data needs to be sent.

Response:


```
{
  "status": "200",
  "questions": [
    {
      "QID": "c9d14faa-6eb8-430d-8a05-d328010a7c85",
      "Options": [
        [
          "Option 1",
```

```
            true
          ],
          [
            "Option 2",
            false
          ]
        ],
        "Company": "TestCompany",
        "InUse": true,
        "Category": "Examples",
        "Body": "My question"
      }
    ]
}
```

This route will return all questions associated with company even if "InUse" parameter is set to *false*.

Adding new question:

Path:

```
https://kevin.com/api/company/questions/add
```

Type: POST

Request:

```
{
  "body": "Question",
  "options": [
      [
          "A",
          true
      ],
      [
          "B",
          false
```

```
        ],
        [
            "C",
            false
        ]
    ],
    "category": "Programming",
    "inUse": true
}
```

About question structure:

All fields are required.

"inUse" field decides wether API will serve this question as part of quiz or not.

Array options is built from small answer arrays. Those arrays must contain possible answer and boolean value that tells if answer is correct or not.

Info: Keep your request structured exactly like an example!

Response:

```
{
    "status": "200",
    "message": "Question added"
}
```

Editing question:

Path:

```
https://kevin.com/api/company/questions/edit
```

Type: POST

Request:

```
{
  "qid": "c9d14faa-6eb8-430d-8a05-d328010a7c85",
  "body": "Question",
  "options": [
      [
          "X",
          false
      ],
      [
          "Y",
          true
      ],
      [
          "Z",
          false
      ]
  ],
  "category": "Testing",
  "inUse": false
}
```


Response:


```
{
  "status": "200",
  "message": "Question updated"
}
```


Info: Editing question is basically adding with specified QID.

Deleting question:

Path:

```
https://kevin.com/api/company/questions/delete
```

Type: POST

Request:

```
{
  "qid": "b3633a8f-752c-4143-ac99-2f232fb6ad56"
}
```

Response:

```
{
  "status": "200",
  "message": "Question deleted"
}
```

THE END!