FULL STACK ENGINEER BOOTCAMP

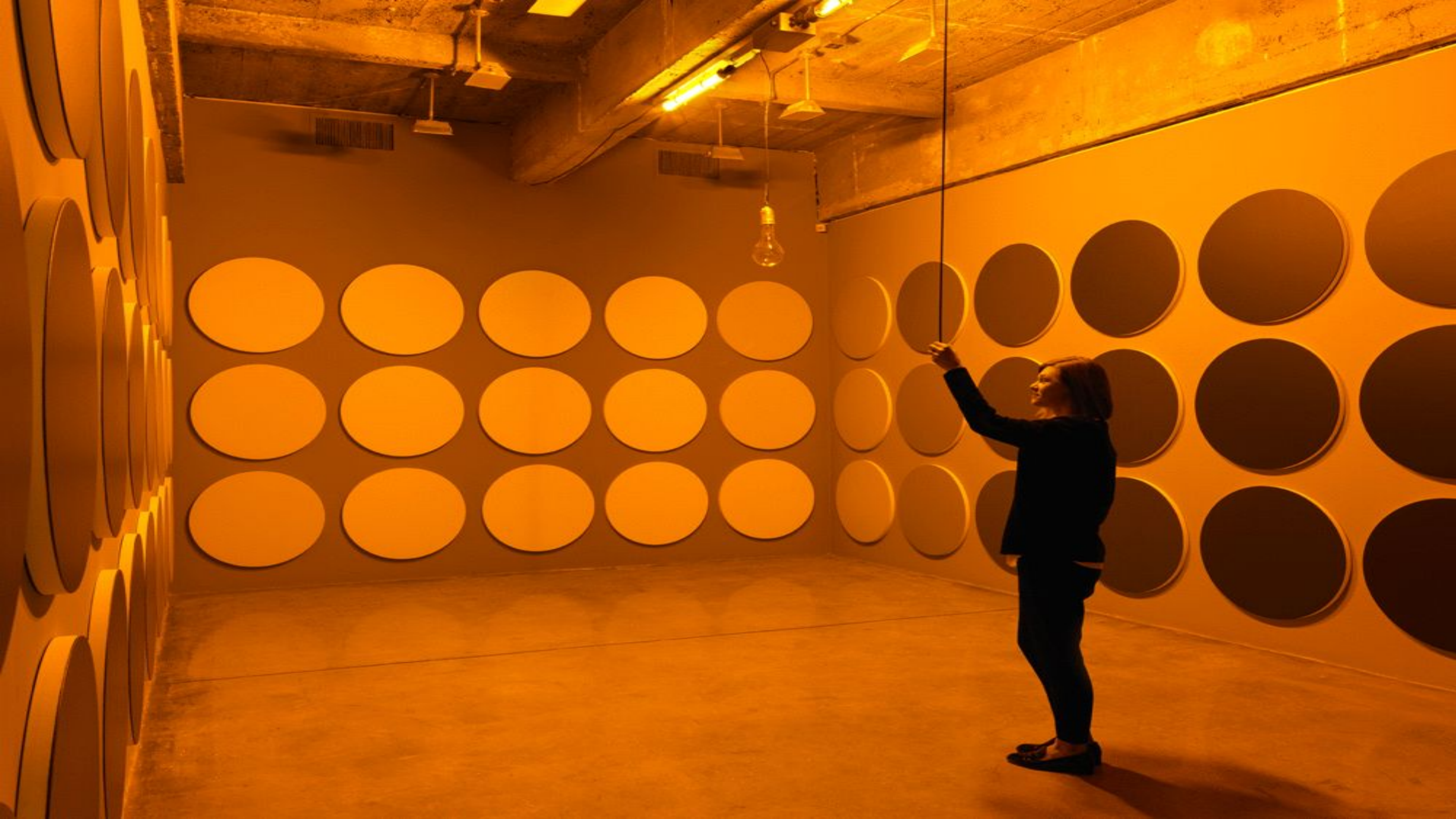
ITJ {...} BOOTCAMP

HELLO!

I am Aylin Yepez

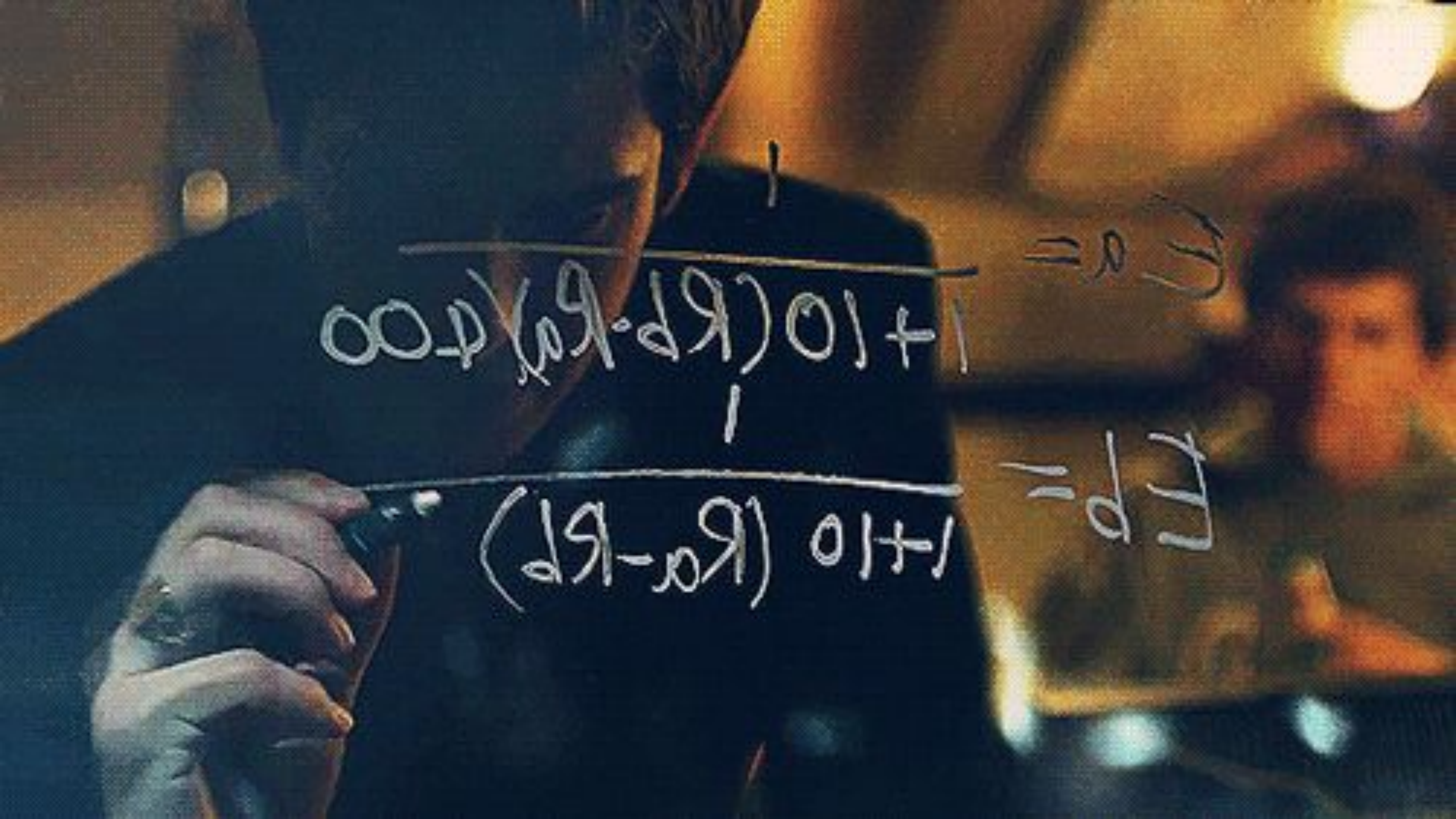
FullStack Developer
Coded to Win

Front vs Back



Front-End

Is the branch that specializes in interfaces and how the page looks like, in other words, the visual aspect taht our website will have on all devices.


Ea = 1 / 1+10(Rb-Ra)/400
Eb = 1 / 1+10 (Ra-Rb)

Back-End

Is the branch that specializes in the logical part of the website. It is responsable for codding the algorithms, manipulating databases and making the website optimal and secure.

FullStack Developer

It is the mixture of Front-End and Back-End. Today the Fullstack developer is the profile most demanded by companies.

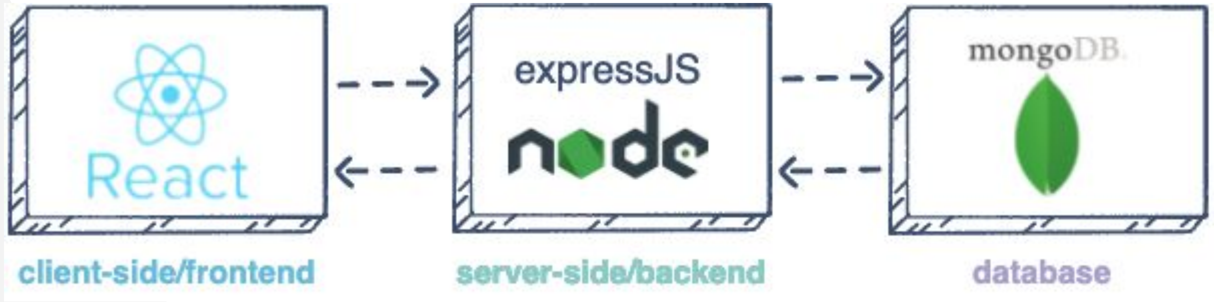
Backend

Bootcamp project

Project: ITJuana Blog clone

MERN (**M**ongoDB, **E**xpressJS, **R**eact, **N**odeJS)

CRUD (**C**reate, **R**ead, **U**pdate, **D**elete)



REST API Concepts

What is a REST API?

What is a REST API?

REST + API

REpresentational
State
Transfer

Application
Programming
Interface

What is a REST API?

REST

- Software architectural style
  - Defines six constraints:
    - **Client–server architecture**
    - **Statelessness**
    - Uniform interface
      - **Resource** identification in **requests**
      - **Resource** manipulation through *representations*

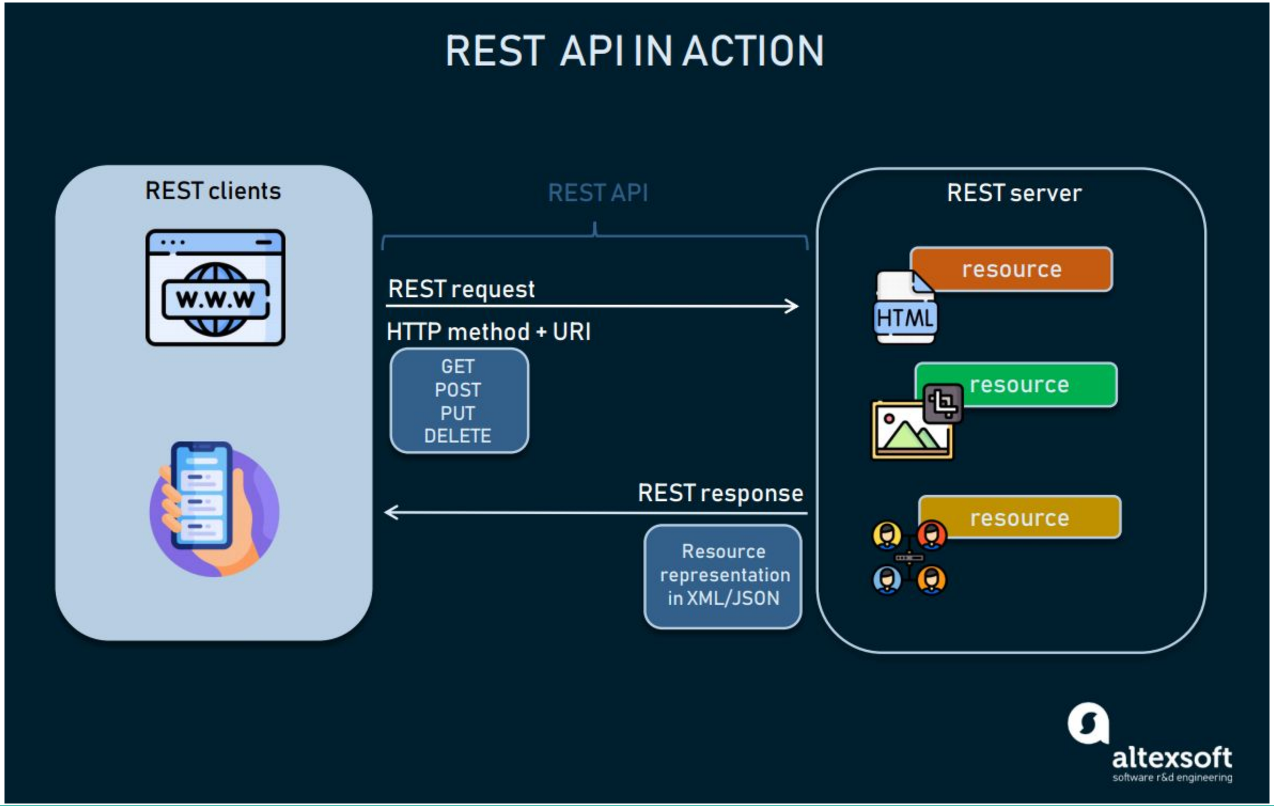
API

- Enable two software components to communicate with each other using a set of **definitions / methods / operations**

REST~~ful~~ API

An **API** that complies with some or all of the **REST constraints** is called a RESTful API, better known as a **REST API**.

How do REST APIs work?



HTTP Request

A request includes four essential parts:

- **HTTP method:** Defines what kind of **operation** to perform.

HTTP method	Operation
GET	Used to **retrieve** resources
POST	Used to **create** a resource
PUT	Used to **update** a resource
DELETE	Used to **delete** a resource
PATCH	Use to **update a part** of a resource

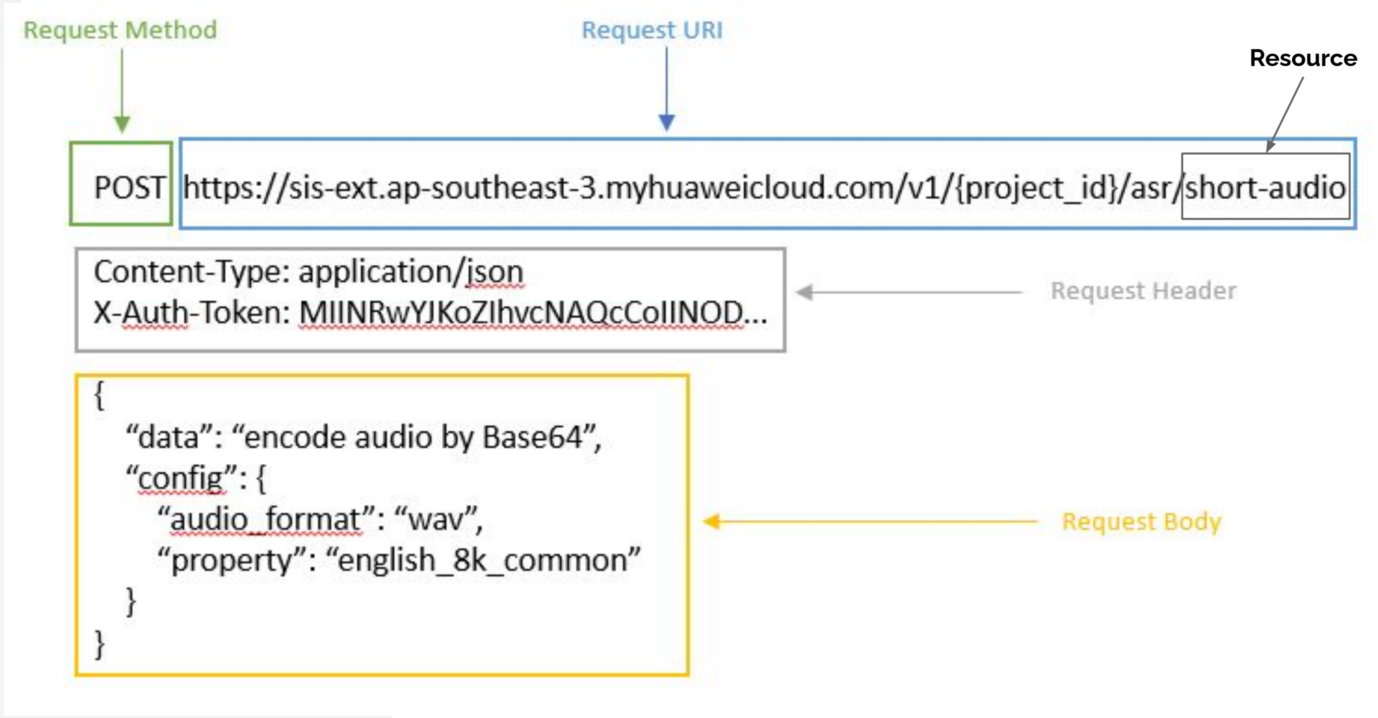
- **URI:** To specify the **resource** to work with.

HTTP Request

- **Header:** Allows the client to pass along information about the request. Mainly, headers provide authentication data - such as an API key and what type of data is sent.

- **Body (Optional):** Is used to send information to the server. For example, the ***resource*** information to <u>create</u> or <u>update</u>.

HTTP Request



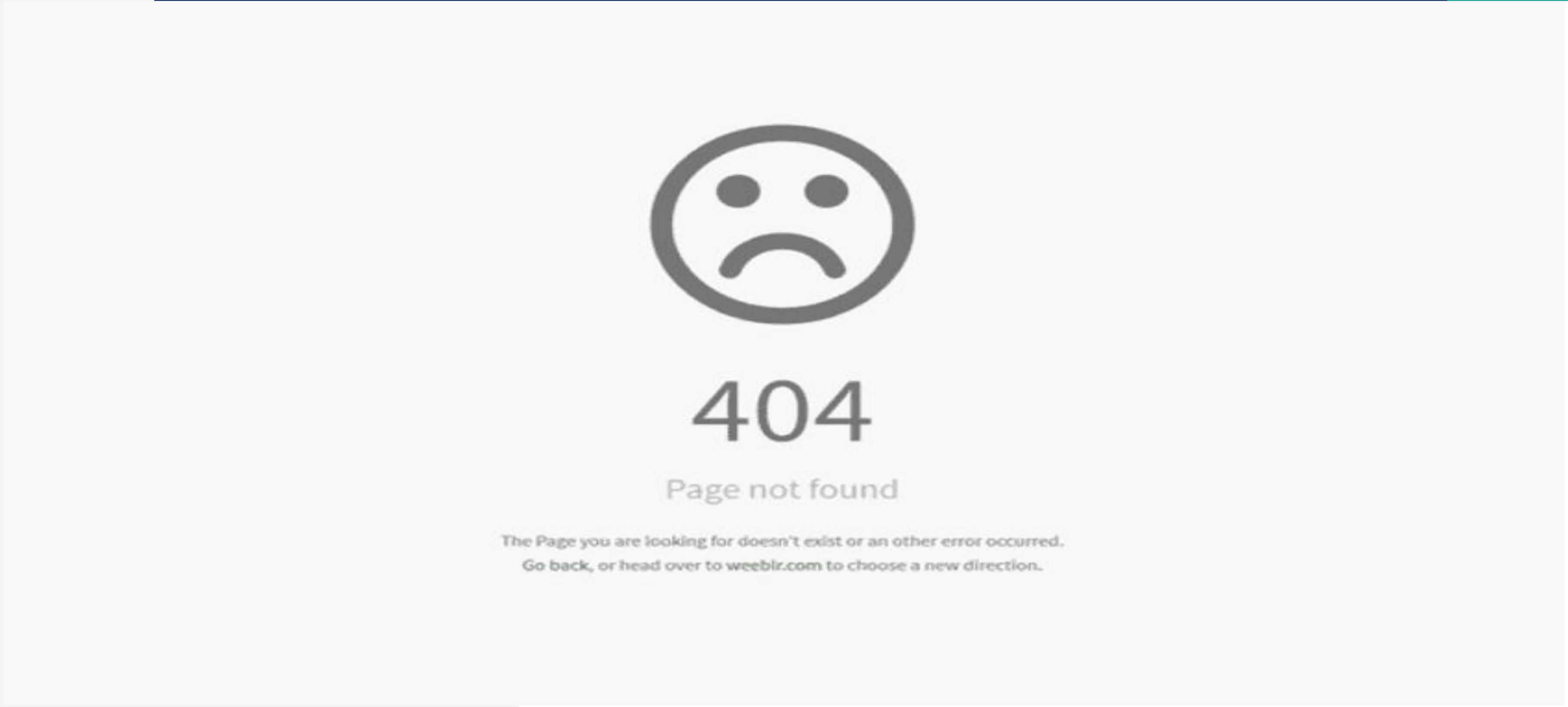
```
POST https://sis-ext.ap-southeast-3.myhuaweicloud.com/v1/{project_id}/asr/short-audio
Content-Type: application/json
X-Auth-Token: MIINRwYJKoZIhvcNAQcCoIINOD...
{
   "data": "encode audio by Base64",
   "config": {
     "audio_format": "wav",
     "property": "english_8k_common"
   }
}
```

HTTP Request

```
POST    http://myblocl/v1/{Project_id}/users

    Content-Type_application/jason
    X-Auth-Token: MIFNJSNFKJSNKFJ

    {
        “nickname”: “GYPZ”
        “email”: example@example.com,
        “password”: miSuperPassword

    }
```

Request Method
Request URI
Resource
Headers
Body

HTTP Response



404

Page not found

The Page you are looking for doesn't exist or an other error occurred.
Go back, or head over to weeblr.com to choose a new direction.

HTTP Response

A response includes three essential parts:

- **Status code:** Tells the client information about the success of the operation.

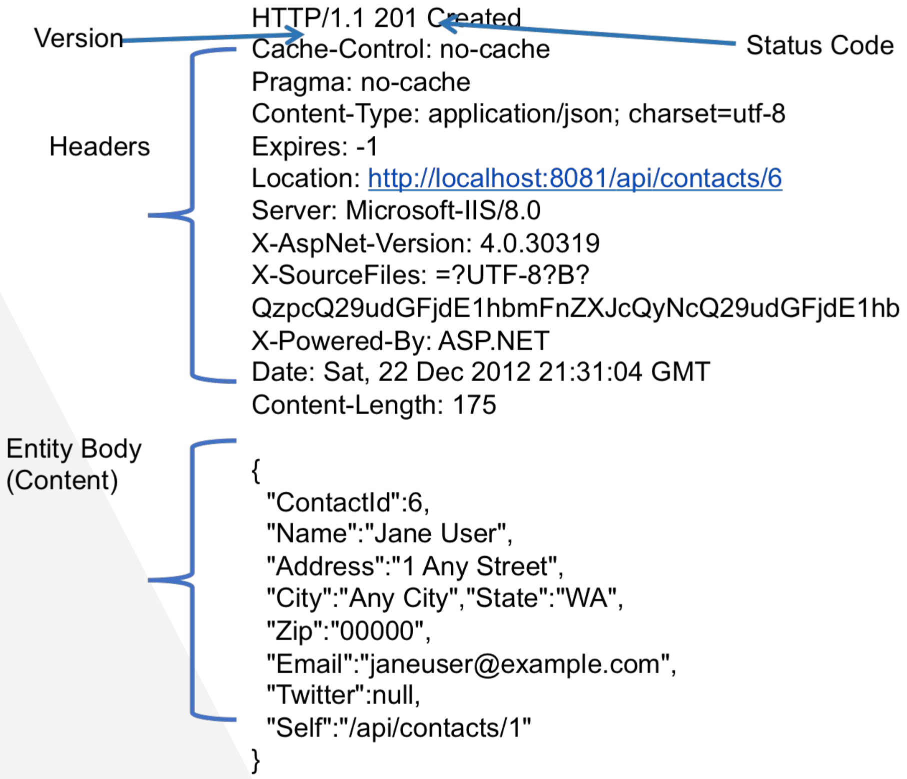
Status code	Meaning
200 (OK)	Response for successful HTTP requests.
201 (Created)	Response for an HTTP request that resulted in a resource being successfully created.
204 (No Content)	Response for successful HTTP requests, where nothing is returned in the response body.
400 (Bad Request)	Response when the request cannot be processed because of malformed request or another client error.
403 (Forbidden)	Response when the client does not have permission to access the resource.
404 (Not Found)	Response when the resource could not be found at this time.
500 (Internal Server Error)	Response for an unexpected failure if there is no more specific information available.

HTTP Response

- **Header:** Similar to request header, response headers also contain useful information, in case the server is sending data, the server must include a content-type.

- **Body (Optional)**: A representation of the ***resource***, it can be represented in different formats, but the most popular ones are **JSON** and XML.

HTTP Response



```
HTTP/1.1 201 Created
Cache-Control: no-cache
Pragma: no-cache
Content-Type: application/json; charset=utf-8
Expires: -1
Location: http://localhost:8081/api/contacts/6
Server: Microsoft-IIS/8.0
X-AspNet-Version: 4.0.30319
X-SourceFiles: =?UTF-8?B?
QzpcQ29udGFjdE1hbmFnZXJcQyNcQ29udGFjdE1hb
X-Powered-By: ASP.NET
Date: Sat, 22 Dec 2012 21:31:04 GMT
Content-Length: 175

{
 "ContactId":6,
 "Name":"Jane User",
 "Address":"1 Any Street",
 "City":"Any City","State":"WA",
 "Zip":"00000",
 "Email":"janeuser@example.com",
 "Twitter":null,
 "Self":"/api/contacts/1"
}
```

BOOTCAMP

Any questions?

NodeJS

NODE JS

NODE JS

Nodejs is a JavaScript runtime

NODEJS ISN'T A PROGRAMMING LANGUAGE...!

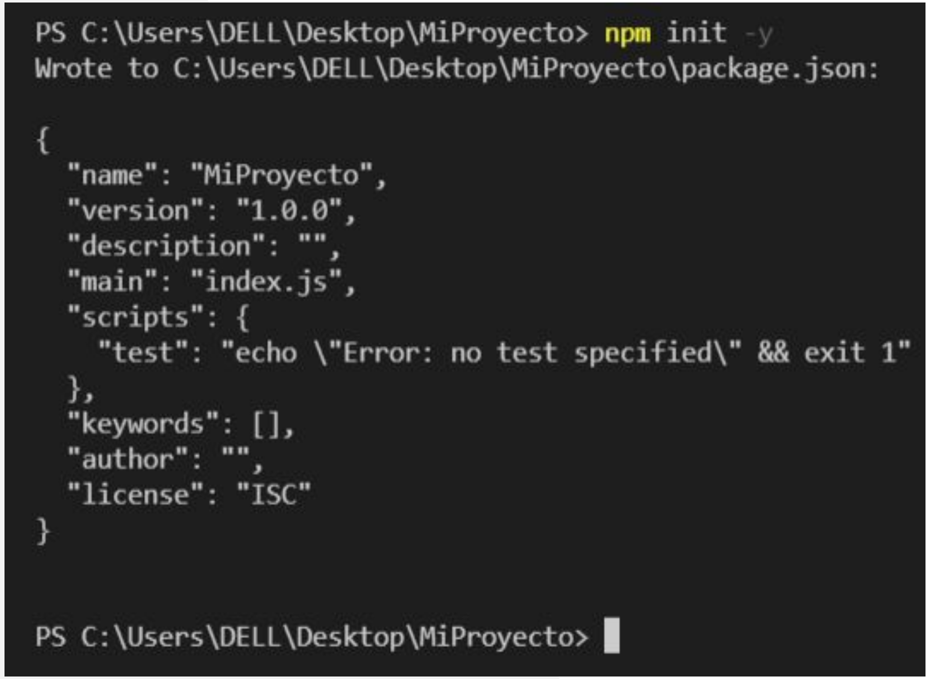
NPM

NPM

Node Package Manager is the package manager for NodeJS

NPM ISN'T A PROGRAMMING LANGUAGE...!

STEP 1



```
PS C:\Users\DELL\Desktop\MiProyecto> npm init -y
Wrote to C:\Users\DELL\Desktop\MiProyecto\package.json:

{
  "name": "MiProyecto",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}


PS C:\Users\DELL\Desktop\MiProyecto>
```

STEP 1

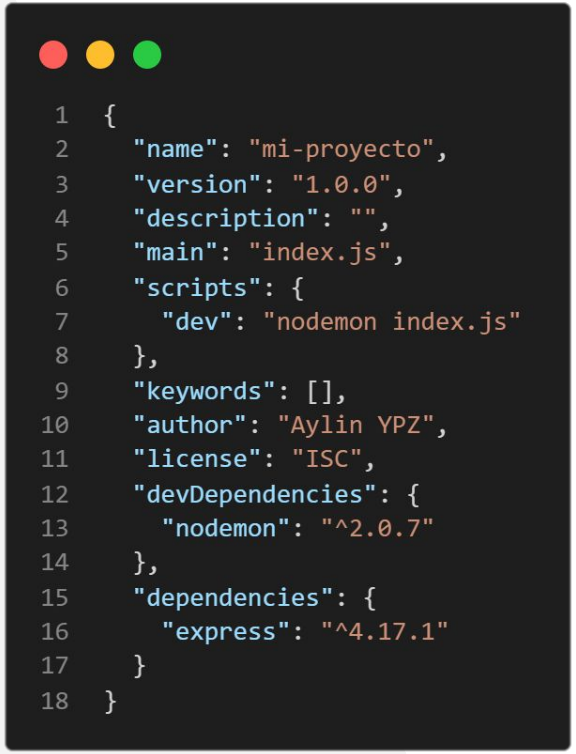
In a folder we execute the next command:

npm init

Or use the shortest version

npm init -y

STEP 1



```
{
  "name": "mi-proyecto",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon index.js"
  },
  "keywords": [],
  "author": "Aylin YPZ",
  "license": "ISC",
  "devDependencies": {
    "nodemon": "^2.0.7"
  },
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

package.json

File that contains information about the project

“ main”

Route to the main file

“ scripts”

Commands associated with a keyword, are executed with... "npm run pClave"

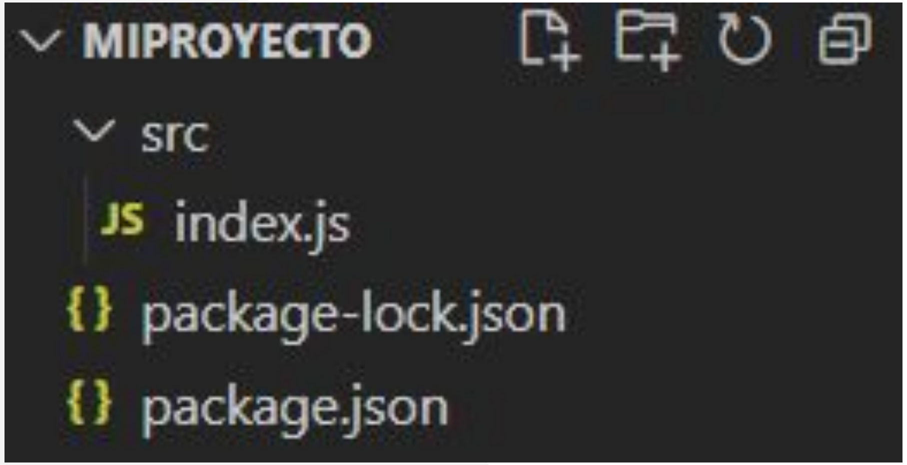
“ dependencies”

Production dependencies

“ devDependencies”

Development dependencies

STEP 1

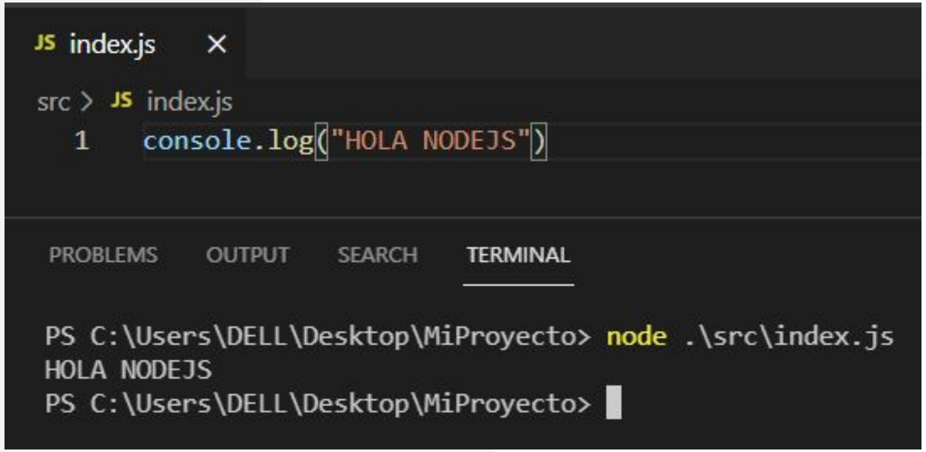


```
MIPROYECTO
  src
    index.js
  package-lock.json
  package.json
```

STEP 2

We créate the structure for our Project, " index.js" will be our main file

STEP 1



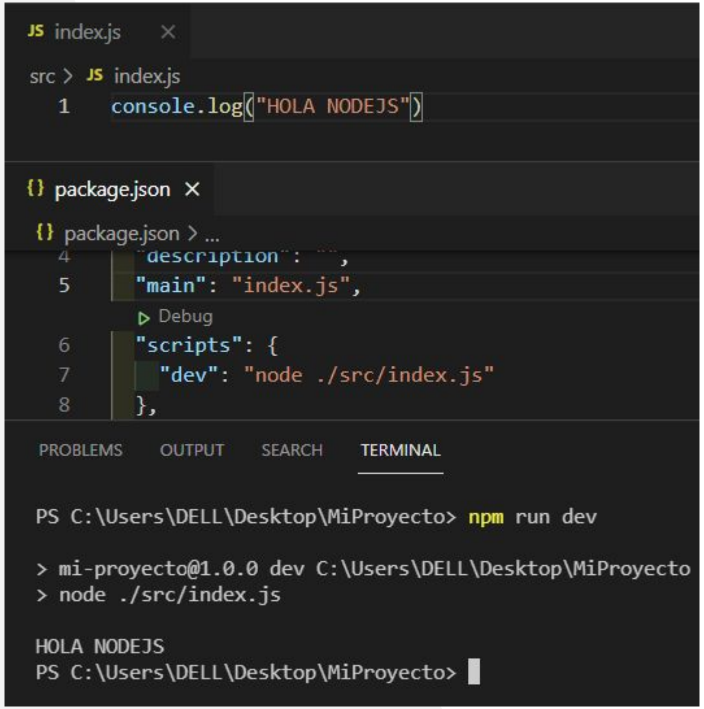
```
console.log("HOLA NODEJS")
```

```
PS C:\Users\DELL\Desktop\MiProyecto> node .\src\index.js
HOLA NODEJS
PS C:\Users\DELL\Desktop\MiProyecto>
```

STEP 3

We write code into " index.js" file and we execute it with...
node ..\src\index.js

STEP 1



index.js

src > index.js

```
console.log("HOLA NODEJS")
```

package.json

package.json > ...

```
  "description": "",
  "main": "index.js",
  Debug
  "scripts": {
    "dev": "node ./src/index.js"
  },
```

PROBLEMS OUTPUT SEARCH TERMINAL

```
PS C:\Users\DELL\Desktop\MiProyecto> npm run dev

> mi-proyecto@1.0.0 dev C:\Users\DELL\Desktop\MiProyecto
> node ./src/index.js

HOLA NODEJS
PS C:\Users\DELL\Desktop\MiProyecto>
```

STEP 4

Remember, in package.json we can define commands to use it from console, let's do it iTj {...} BOOTCAMP

THANKS!

Any questions?