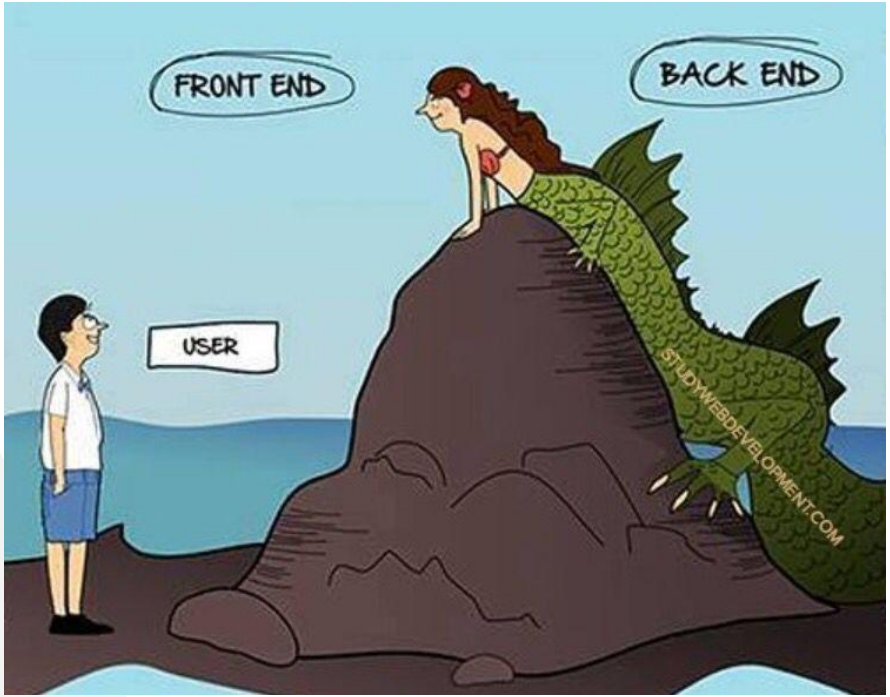

iTj
{...}
BOOTCAMP
Backend intro &
ExpressJS

Backend

Backend

Frontend:

The part of a computer system or application **that is directly accessed by the user**.



Backend:

The part of a computer system or application **that is not directly accessed by the user**, typically responsible for storing and manipulating **data**.

REST API Concepts

What is a REST API?

What is a REST API?

REST + API

REpresentational
State
Transfer

Application
Programming
Interface

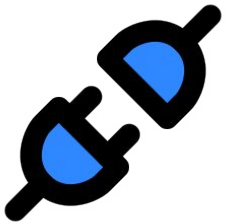
What is a REST API?

REST

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

API

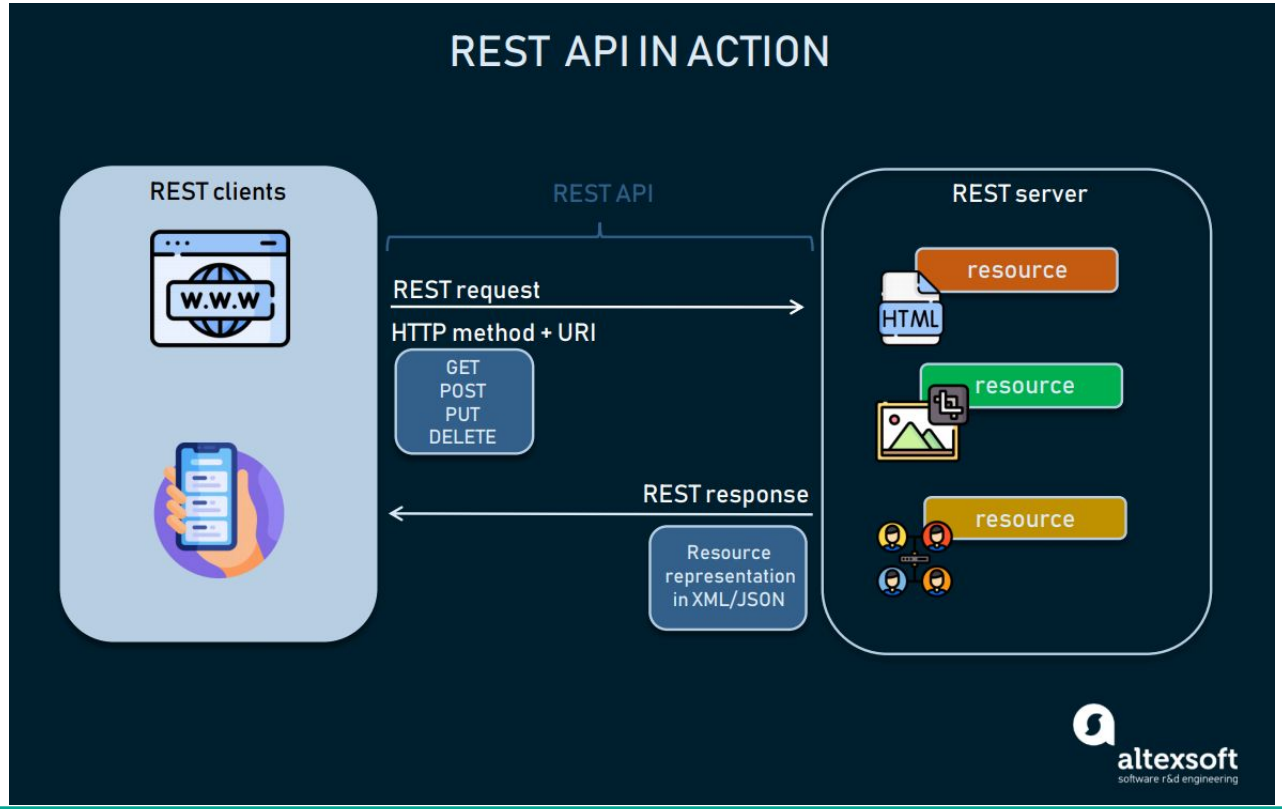
- Enable two software components to communicate with each other using a set of **definitions / methods / operations**



RESTful 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

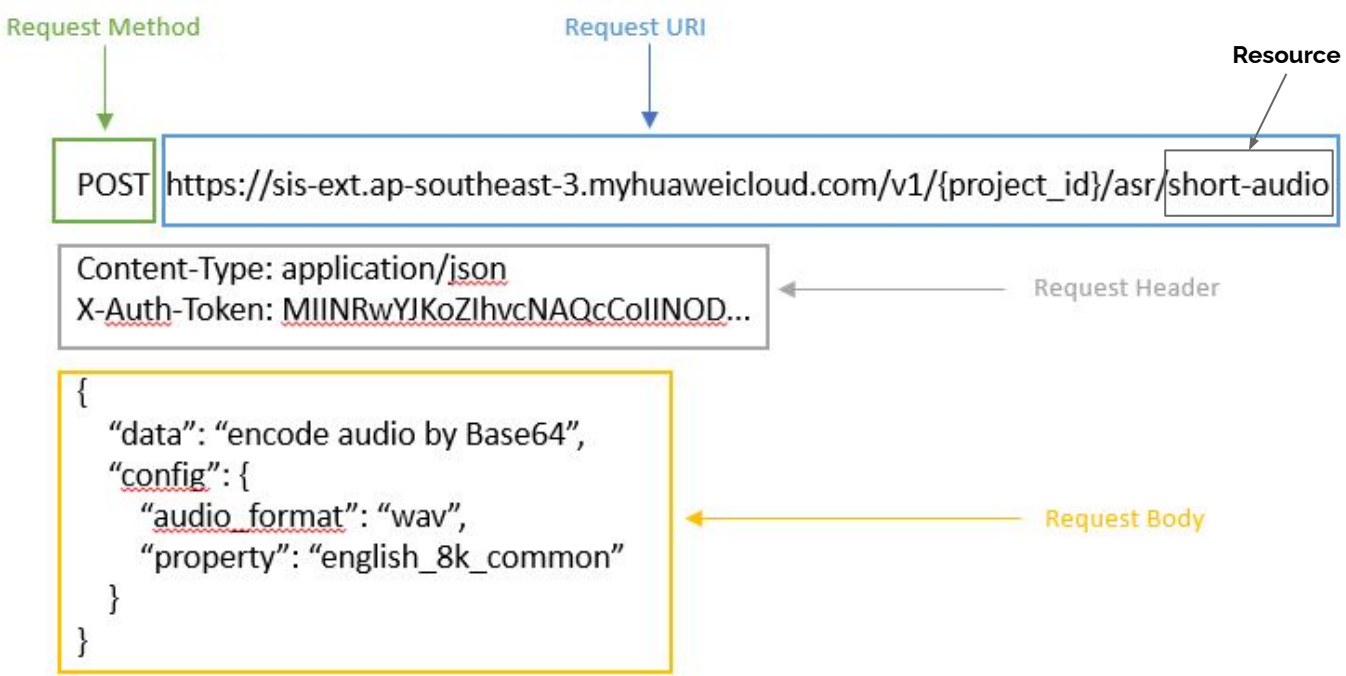
- **URI:** To specify the **resource** to work with.

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 create or update.

HTTP Request



HTTP Response

A response includes three essential parts:

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

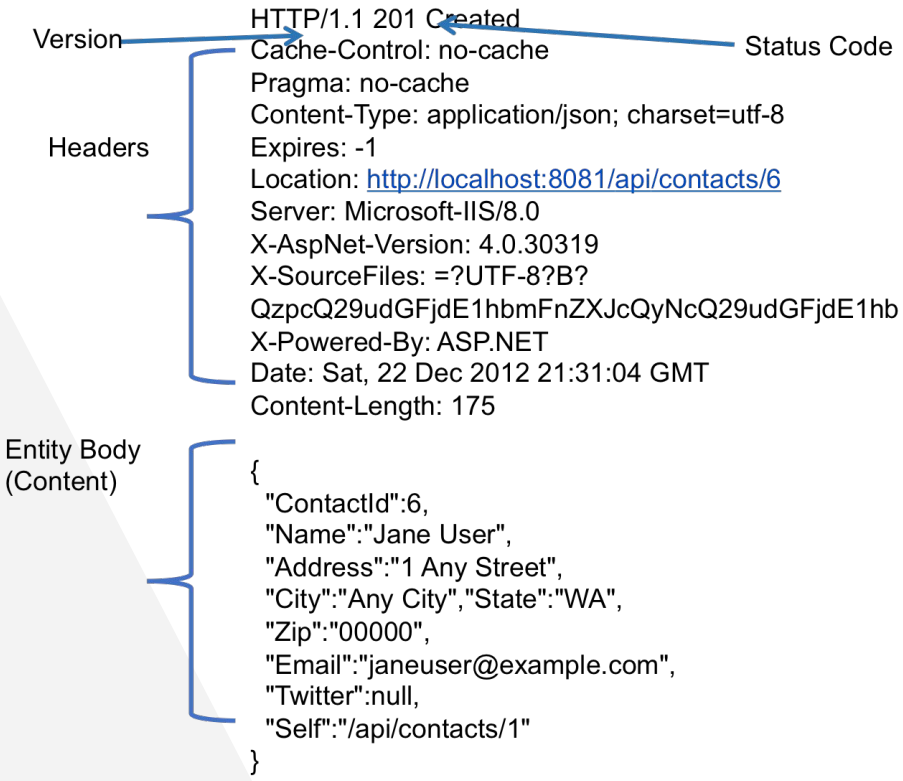
Status code	Meaning
200 (OK)	Response for successful HTTP requests.
201 (Created)	Response for an HTTP request that resulted in a resource being successfully created.
204 (No Content)	Response for successful HTTP requests, where nothing is returned in the response body.
400 (Bad Request)	Response when 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



Node

Executing Javascript on the server

Why we want to use Node?

- Easy to learn
- Fullstack Javascript
- Large community
- Extended support
- Flexible
- You can easily connect to common databases

Why we want to use Node?

- Default tool for manage projects.
- Configure your project
- Install dependencies on your project.
- We'll use the npm CLI

Let's code!

Why we want to use Express?

- Web development fast and easy
- Easy to configure and customize
- Large community
- Many resources /libraries
- You can easily create a REST API.

Why we want to use Express?

Express

```
const server = http.createServer(async (req, res) => {
    if (req.url === "/api" && req.method === "GET") {
        res.writeHead(200, { "Content-Type": "application/json" });
        res.write('Hello World!');
        res.end();
    }

    else {
        res.writeHead(404, { "Content-Type": "application/json" });
        res.end(JSON.stringify({ message: "Route not found" }));
    }
});

server.listen(3000, () => {
  console.log(`server started on port: ${3000}`);
});
```

```
const app = require('express')()

app.get('/', (req, res) => {
  res.send('Hello World!')
})
app.get('*', function(req, res){
  res.status(404).send("Route not found");
});
app.listen(3000, () => {
  console.log(`Example app listening on port ${3000}`)
})
```

Let's code!

BOOTCAMP

THANKS!

Any questions?