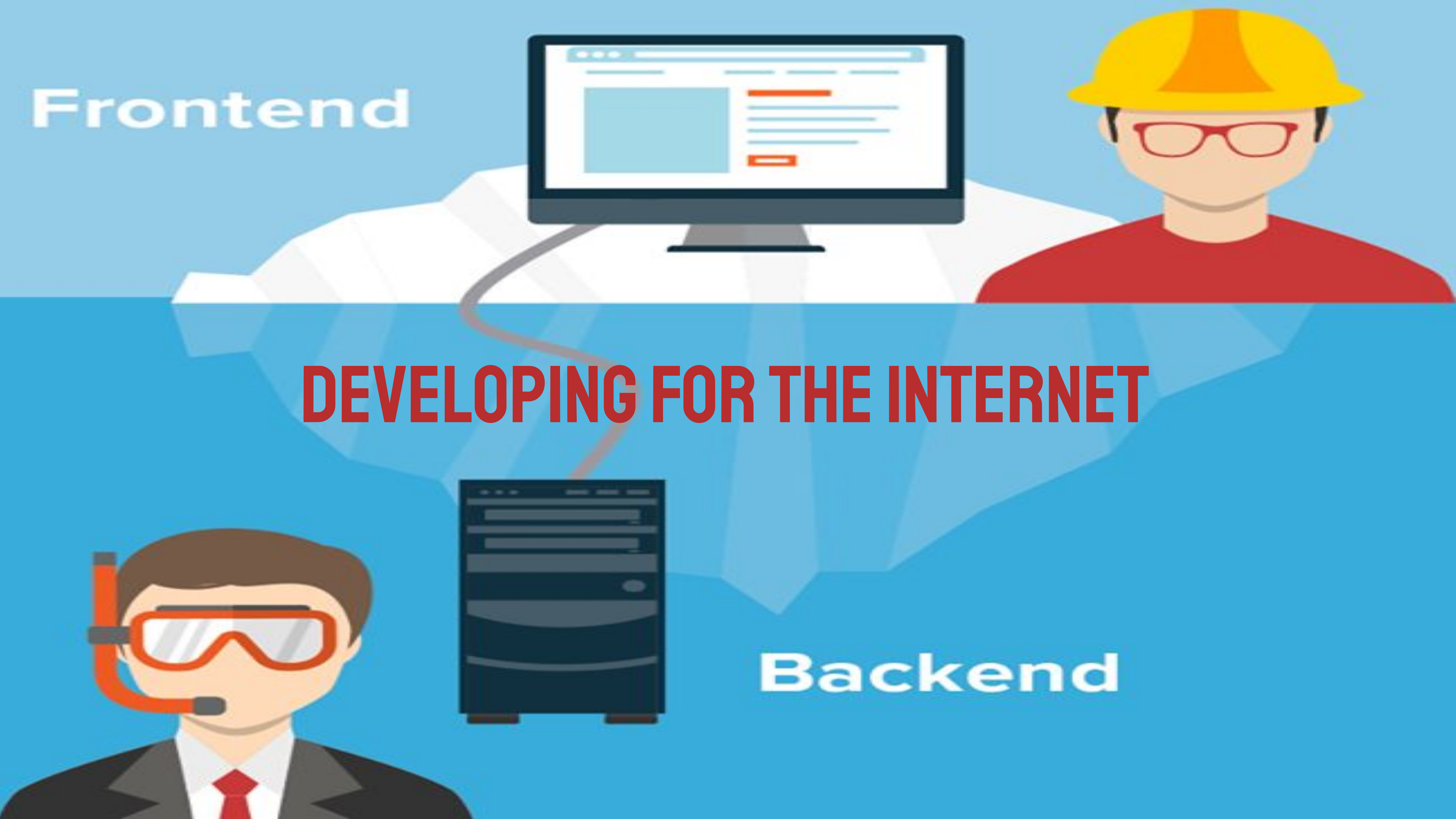
DEVELOPING FOR THE INTERNET



WHAT WE WILL LEARN

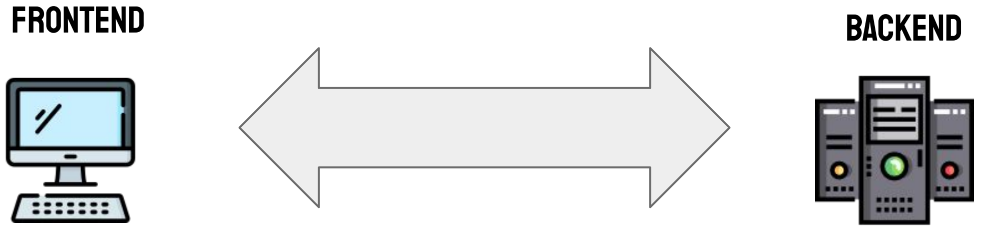
FRONTEND AND BACKEND

REST APIS AND HTTP

SECURITY

MODERN INTERNET APPLICATIONS: FRONTEND AND BACKEND SOFTWARE

Modern Internet Applications have several components. One or more client Interface applications that we all know and love (The frontends) and one or more backend servers that provide resources to the clients frontend. Both front and back ends interacting together via Http.

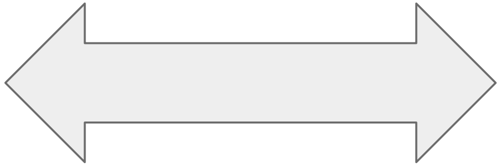


What the user sees and interacts with
Browser Mobile App

Processes requested by client
Storage, computations, transactions
Behind the scenes, not visible to users

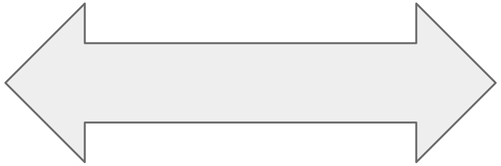
MODERN INTERNET APPLICATIONS: FRONTEND AND BACKEND SOFTWARE

Customer representative communicates with customer, and forwards any requests to workers to be completed.



People behind the scene perform the actual job and let the representative know the result.

Newsreader presents to audience the news, supported by others in the backoffice preparing the content.

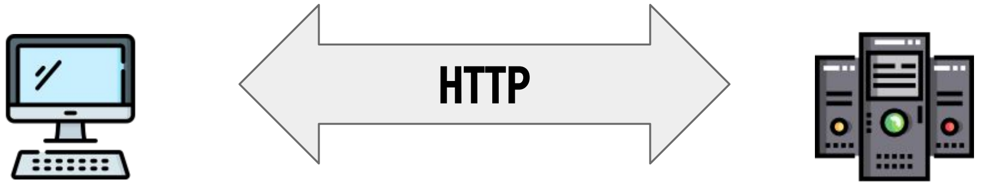


People in the backoffice prepare the news content and guide the newsreader about the program.

FRONTEND AND BACKEND: TOOLS

FRONTEND

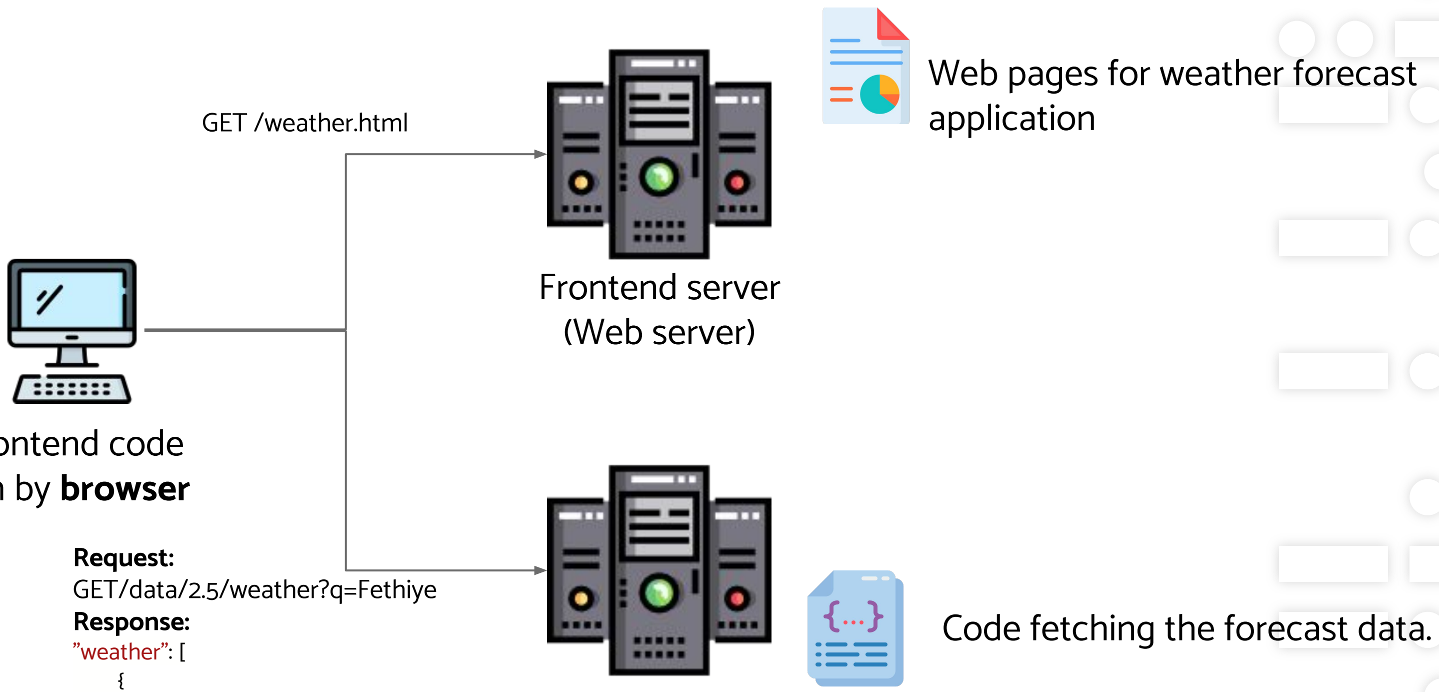
BACKEND



What the user sees and interacts with
Graphics, animation, forms

Processes requested by client
Storage, computations, transactions
Behind the scene, not visible to user

HTTP FOR WEB APPS


GET /weather.html
Web pages for weather forecast application
Frontend server (Web server)
Frontend code run by **browser**
Backend server (Application server)
Code fetching the forecast data.

Request:
GET/data/2.5/weather?q=Fethiye
Response:

```
"weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01n"
    }
  ],
```

API: Application Programming Interface

APIs are today part and parcel of software development for the internet and have become a standard way to offer services across the internet. They live in the ether, that is on servers in the cloud.

An API is at its basic level is a set of what are called ‘Endpoints’ (an endpoint is equivalent to a specific service request on an API) that allows other computers/software clients to request information or access a service. APIs are stateless, they do not maintain a state, that is, there is no concept of doing something based on the changing condition of some entity over time.

The fundamental pattern of how APIs work is based on the http request/response workflow using something called REST (representational state transfer), an architectural style for designing APIs. REST APIs are sometimes called RESTFUL APIs.

According to a set of predefined rules, a client requests some information from a specific API-endpoint and if the request is granted (a legitimate request, both syntactically and semantically) the API responds accordingly.

Some APIs are totally open APIs they do not require authentication (passwords and or other credentials) to request a service from an endpoint. Others have fully protected endpoints using various authentication methods such as oauth and JWT (JSON WEB TOKENS), amongst others, whilst other APIs have both open and protected endpoints.

Many APIs are paid for services, whilst others are free. Google, Amazon, Twitter, Facebook… amongst a zillion other companies all offer API’s for access to various services.

More on APIs

API: AN EXAMPLE API CALL TO GET WEATHER INFORMATION

$$\underbrace{\text{https://}}_{\text{Protocol}}\underbrace{\text{api.openweathermap.org}}_{\text{Host address}}\underbrace{\text{/data/2.5/weather}}_{\text{Path}}\text{?}\underbrace{\text{q=Fethiye\&appid=7426fbdad185c2d61ec8d50ccf936400}}_{\text{URL parameters}}$$

Enter this URL to your browser and press enter.

This is a GET request to the Weather API.

As response, you will get a JSON response with weather forecast of Fethiye, like this:

{”coord”:{”lon”:29.1164,”lat”:36.6217},”weather”:[{”id”:803,”main”:”Clouds”,”description”:”broken clouds”,”icon”:”04d”}],”base”:”stations”,”main”:{”temp”:289.08,”feels_like”:288.11,”temp_min”:289.08,”temp_max”:289.08,”pressure”:1026,”humidity”:53,”sea_level”:1026,”grnd_level”:1024},”visibility”:10000,”wind”:{”speed”:2.05,”deg”:235,”gust”:1.46},”clouds”:{”all”:65},”dt”:1644665053,”sys”:{”type”:1,”id”:6984,”country”:”TR”,”sunrise”:1644641724,”sunset”:1644680419},”timezone”:10800,”id”:314967,”name”:”Fethiye”,”cod”:200}

Each API has a specification describing:

1) How to construct a request and 2) What kind of response to expect
Examine the spec for this API at https://openweathermap.org/current

For example try &lat=&long= params and &mode=html and &mode=xml to see different responses.

Next, let’s understand what it is and how we can use it.

fathat.org

API: Practically every modern app on the internet has an API

- Facebook Graph API: https://developers.facebook.com/docs/graph-api/
- Instagram API: https://developers.facebook.com/docs/instagram-api/
- Google Maps API: https://developers.google.com/maps/documentation
- Twitter API: https://developer.twitter.com/en/docs/twitter-api
- Borsa/Doviz API:
  https://collectapi.com/tr/api/economy/altin-doviz-ve-borsa-api

... And many more

API: Open API for Writing API Specifications

OpenAPI is an open source standard for defining and writing API specifications. An API specification is like a document that defines the details of an APi. For example

What requests can be made.
The paths of the requests - path to the endpoint
The type of request
The structure of the request
The data required for the request
The security required for requests

The response provided to each request
The data contained in the response

And more...

```
openapi: 3.0.0
info:
  version: 1.0.0
  title: Sample API
  description: A sample API to illustrate OpenAPI concepts
paths:
  /list:
    get:
      description: Returns a list of stuff
      responses:
        '200':
           description: Successful response
```

Study the following openAPI tutorial - you'll need it for the project.

API: DATA EXCHANGE USING JSON (JAVASCRIPT OBJECT NOTATION)

Json has become a defacto standard for many developers for sending and receiving structured data. It is similar to many data structures in programming languages, data objects in Javascript, Dictionaries in Python, Maps in Java. It is widely used in APIs for incoming data in requests and outgoing in responses.

Example: Weather forecast

Request: GET/data/2.5/weather?q=Fethiye
Response:

```
{
   ”timezone”: 10800,
   ”id”: 314967,
   ”name”: ”Fethiye”,
   ”weather”: [
      {
         ”id”: 800,
         ”main”: ”Clear”,
         ”description”: ”clear sky”,
         ”icon”: ”01n”
      }
   ],
}
```

Each object is a list of key-value pairs.
Each key is a string.
Each value is either

- String
- Number (decimal)
- Boolean (true, false)
- Object (nested): {...}
- List of values: [...]

API: REST

There are a number of http methods used for accessing REST APIs.

GET: Read/retrieve information.

Weather forecast, finance data, bank account balance

POST: Create/save new data

Open bank account, new exam, make bank transaction.

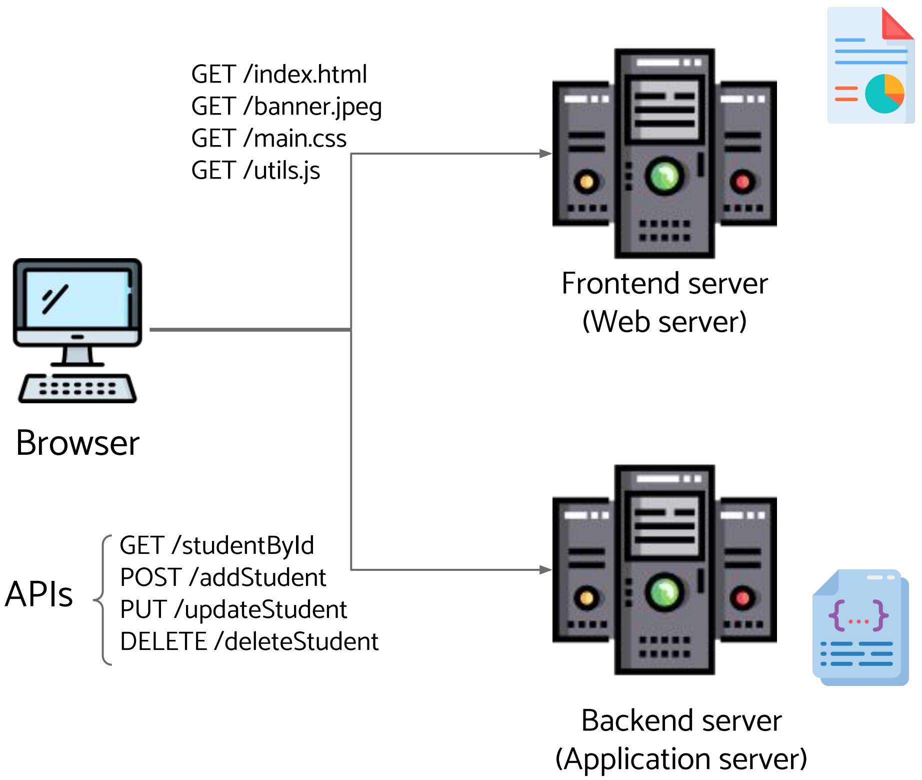
PUT: Update existing data

Update phone number, change student grade

DELETE: Delete data

Unregister student, close bank account

API: HTTP FOR WEB APPS: SEPARATION OF FRONTEND AND BACKEND SERVICES



HTML/CSS/JavaScript and others (images, vides, PDFs, Word docs): Static in server; runs on client.

Backend code: Runs on server

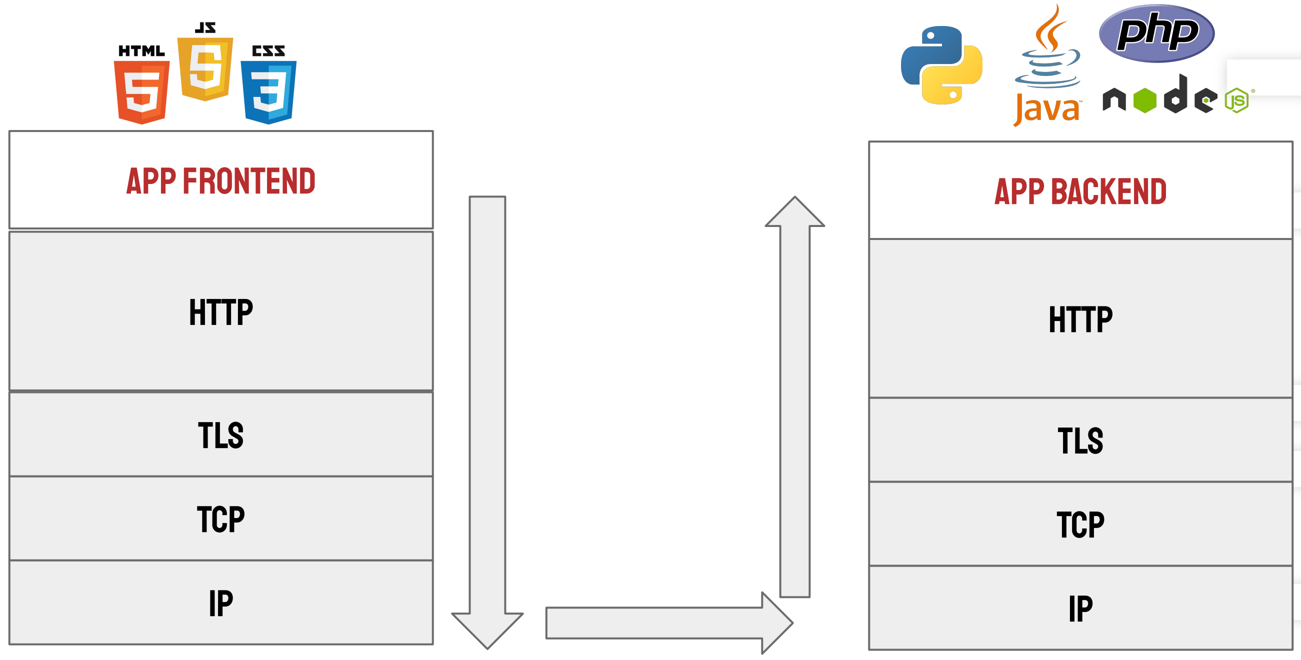
fathat.org

API: REST EXAMPLES BY USE CASES

USE CASE	GET (READ)	POST (READ)	PUT (UPDATE)	DELETE
School app	Get student info by school id	Register new student	Change contact info of student	Unregister a student
Bank app	Get account balance Get credit card statement	Open new account Send new transaction	Set withdrawal limit for account or credit card	Delete bank account Cancel a transaction
Chat app	Search in messages	Send new message	Edit previous message	Delete a message

FRONTENDE AND BACKEND COMMUNITATE WITH HTTP

HTTP is an application layer protocol, uses TCP/IP protocols to send data to other side in reliable way.



FRONTEND LANGUAGES: HTML / CSS / JAVASCRIPT

HTML - Hypertext Markup Language
Content and Structure
Paragraphs, tables, lists

CSS - Cascading Style Sheets
Presentation
Layout, Font, color, border, size

JavaScript
Behavior
Dynamic view, user interactions, clicks, server calls

FRONTEND LANGUAGES: HTML / CSS / JAVASCRIPT

```html
<html>
  <button id="btn" class="buttons">0</button>
</html>
```

```css
.buttons {
  color: blue;
  font-size: 12px;
}
```

```javascript
document.getElementById("btn").addEventListener("click", function() {
  var current = document.getElementById("btn").innerText;
  document.getElementById("btn").innerText = parseInt(current) + 1;
});
```

AUTHENTICATION AND AUTHORISATION

Authentication and Authorisation are extensive subjects in their own right. We will go over the main aspects so that you have a general understanding about some of the methods used.

The main reason for their existence is clear:

MANAGING ACCESS TO RESOURCES

Not all resources in the Internet are free.

- Only YOU should see your child's e-Devlet information.
- Only YOU or BANK AGENT should make a bank transaction.
- Only YOUR TEACHER should add enter exam grade.

Two problems to solve:

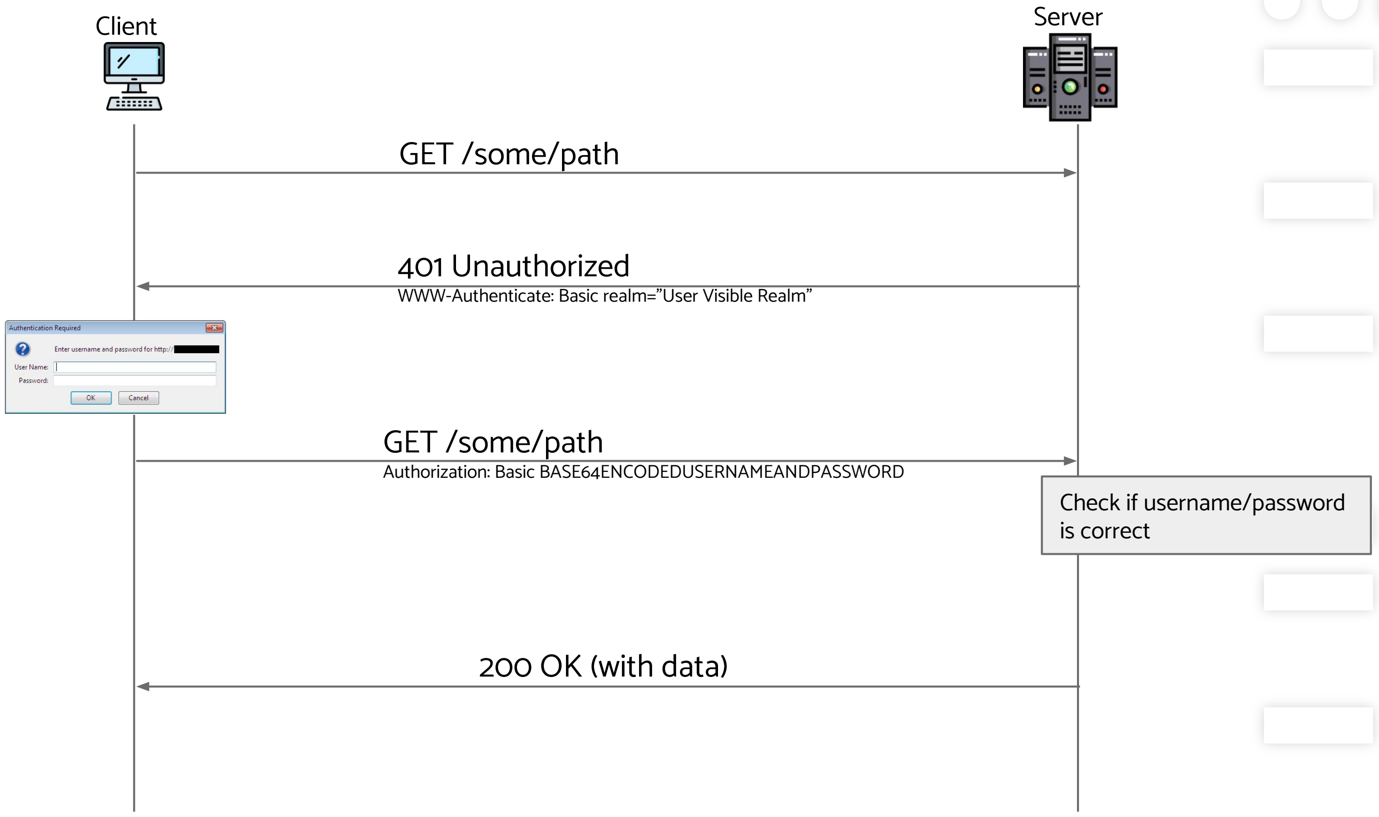
1. How to identify that YOU are the user
    a. Authentication: Identify users
2. How to ensure you get and modify ONLY YOUR data
    a. Authorisation: Check permissions

Authentication: Identify Users

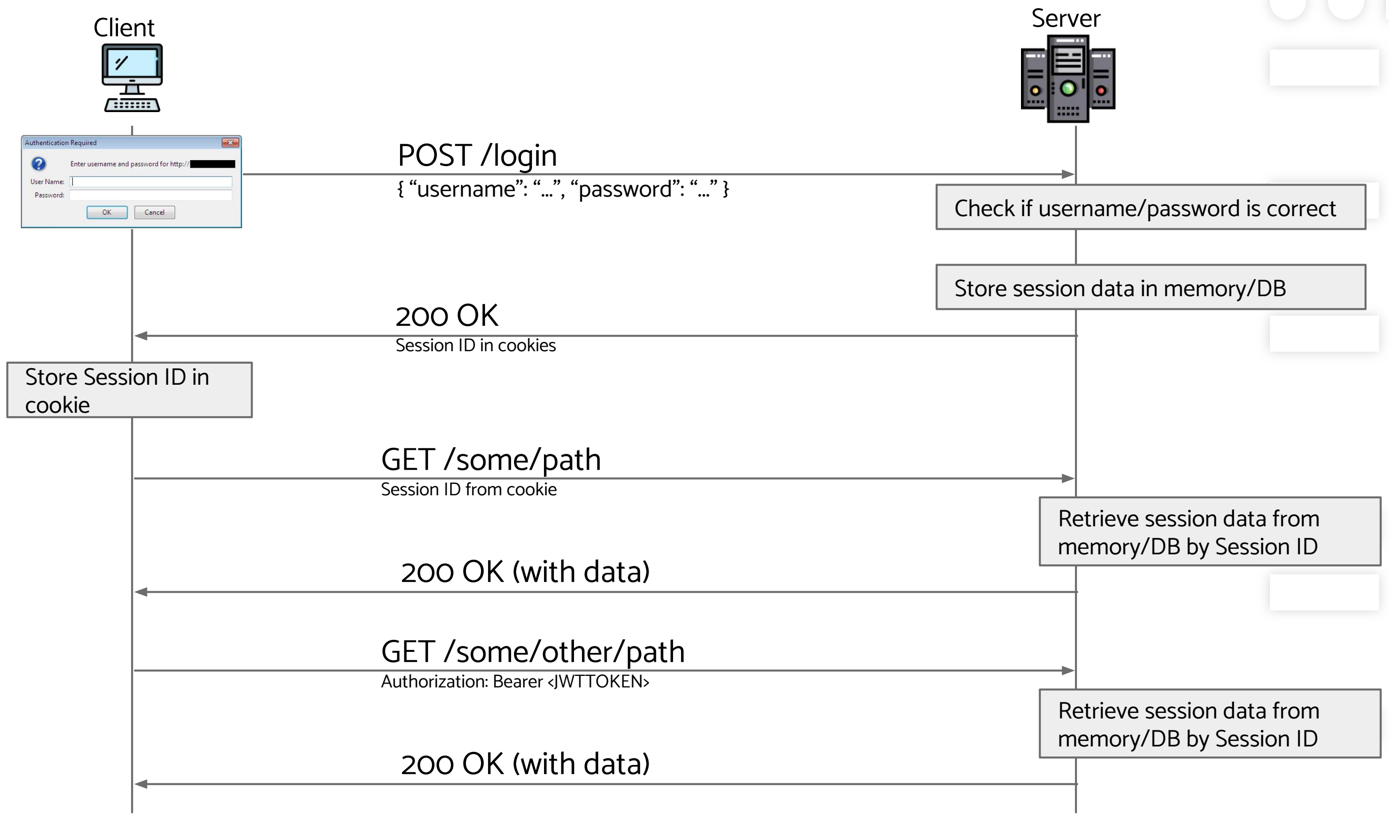
Types of authentication:

- Password based
  - Include username+password at every request
- Multi-factor
  - Username+password, plus a code sent to email/phone
- Certificate-based
  - Send public key to to prove the right identity
- Biometric
  - Fingerprint/eye scan
- Token-based
  - First send username+password, and then use token

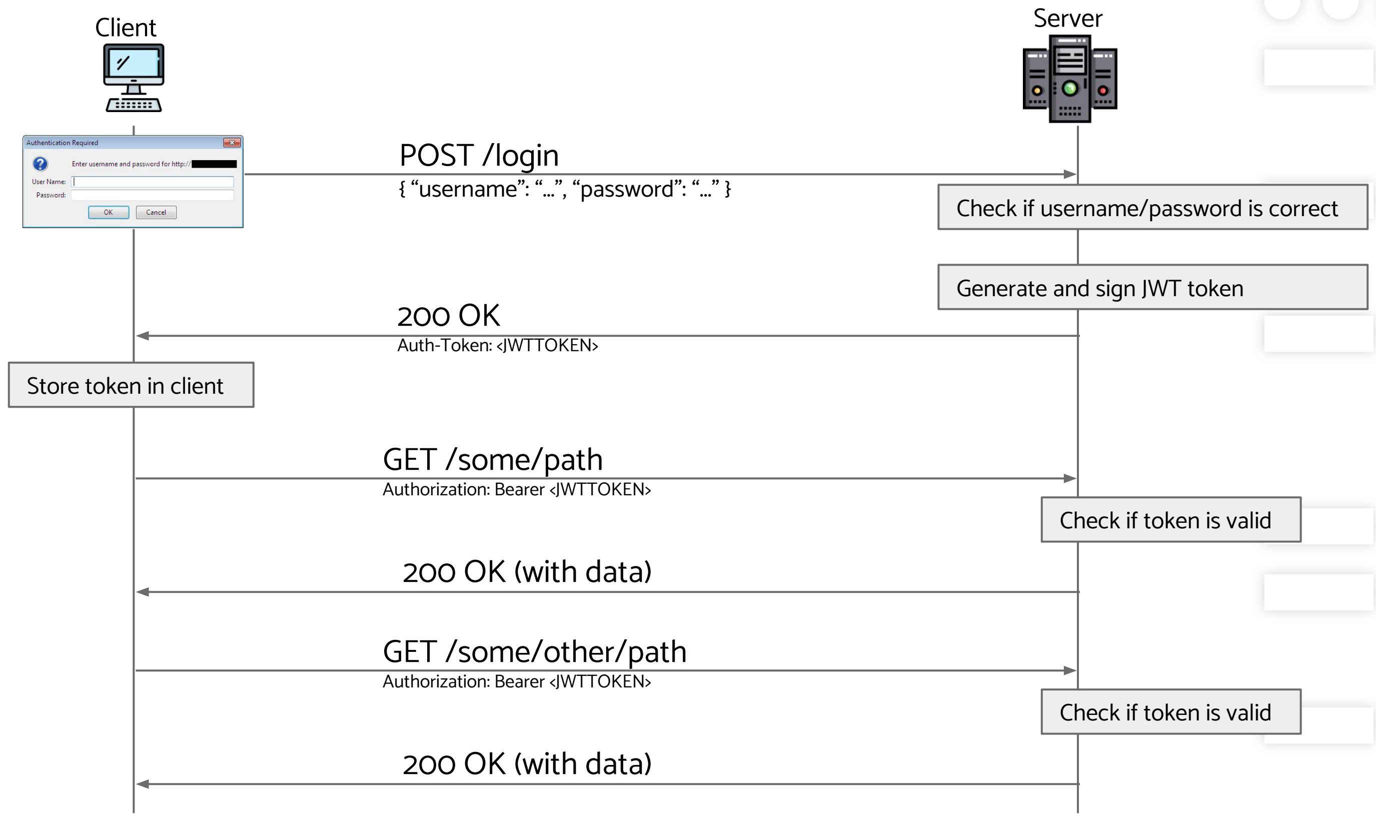
AUTHENTICATION: BASIC


Client
Server
GET /some/path
401 Unauthorized
WWW-Authenticate: Basic realm="User Visible Realm"
Authentication Required
Enter username and password for http://[redacted]
User Name:
Password:
OK
Cancel
GET /some/path
Authorization: Basic BASE64ENCODEDUSERNAMEANDPASSWORD
Check if username/password is correct
200 OK (with data)

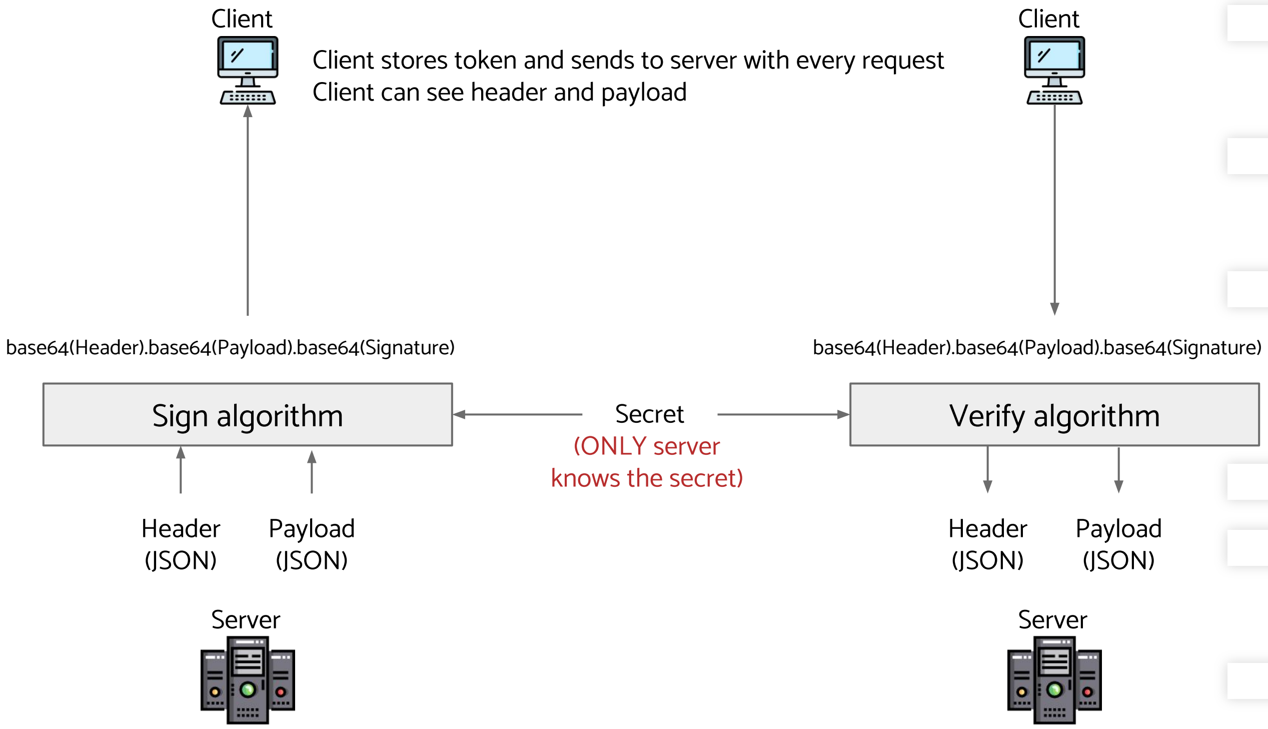
AUTHENTICATION: SESSION BASED


Client
Server
Authentication Required
Enter username and password for http:// [redacted]
User Name:
Password:
OK
Cancel
POST /login
{ "username": "...", "password": "..." }
Check if username/password is correct
Store session data in memory/DB
200 OK
Session ID in cookies
Store Session ID in cookie
GET /some/path
Session ID from cookie
Retrieve session data from memory/DB by Session ID
200 OK (with data)
GET /some/other/path
Authorization: Bearer <JWTTOKEN>
Retrieve session data from memory/DB by Session ID
200 OK (with data)

AUTHENTICATION: TOKEN BASED

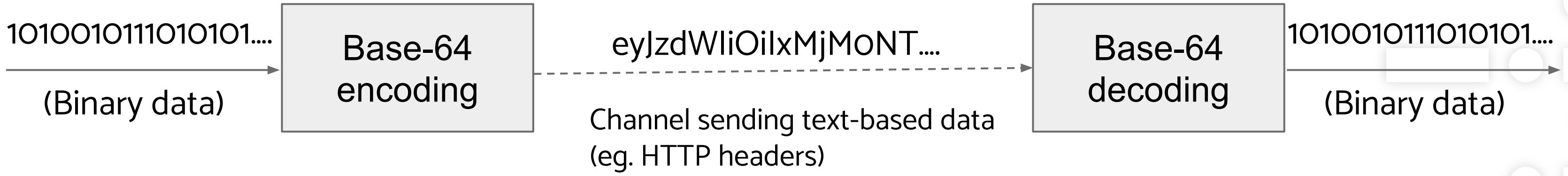


AUTHENTICATION: JWT - JSON WEB TOKENS


Client
Client stores token and sends to server with every request
Client can see header and payload
Client
base64(Header).base64(Payload).base64(Signature)
base64(Header).base64(Payload).base64(Signature)
Sign algorithm
Secret
(ONLY server knows the secret)
Verify algorithm
Header (JSON)
Payload (JSON)
Header (JSON)
Payload (JSON)
Server
Server

BASE-64 ENCODING AND DECODING

Base-64 is a group of algorithms to represent raw binary data as textual data, for sending this data in a channel that only supports text-based data.
All data is encoded into 64 characters.



fathat.org

AUTHORISATION: CHECKING PERMISSIONS

- User roles:
  - Basic user, Admin, Power User etc.
  - Each has access to certain operations

- Scopes:
  - Indicates which operations allowed by a token
  - Token are generated to contains scopes, indicating which operations can be performed when that token is provided