Week 08: REST API, OpenAPI

Agenda

```
> GET /info/agenda
> Host: tutor.pb138

< HTTP/1.1 200
< Content-Type: text/markdown;charset=utf-8
< Server: Tutor
<
< - Express.js
< - REST API
<   - Naming activity
<   - CRUD
< - OpenAPI
<    - REST Client
< - Jest
<   - Testing your API
< - Demo
<  - REST, Swagger, Jest
```

Express.js - Web Application Framework

- Framework that allows quickly building web applications and REST APIs
- Provides a very minimal, precise set of tools necessary for creating web applications
- Used by many other JS/TS frameworks as their backbone

Express - Install

```
npm i express
npm i -D @types/express
```

Adding express to the code

```
import express, { Express, Request, Response } from 'express';

const app: Express = express();
const port = 8080;

app.listen(port, () => {
  console.log(`Server is running at https://localhost:${port}`);
});
```

Express - routes and handlers

- The data is firstly processed by a pipeline of functions called middleware
- These functions can, for example, check privileges or handle things that need to happen to every request before it is processed individually
- The request then gets processed via a router
- Router then routes the requests it defines the flow of individual requests
- Each route has an assigned handler - a function that processes the request individually

```
// continuation of the code from the previous slide
app.get('/', (req: Request, res: Response) => {
  res.send('Express + TypeScript Server');
});
```

Let's learn by doing...

... by saving animals from shelter!

Background story:

Animal Crossing is mobile application that provides to potential pet owners simple and intuitive interface to adopt animal from shelter. After making an adoption request the new owner will receive guidelines for petting. Application also provides access for caretakers to add a pet for adoption or mark it as adopted.

Every shelter must include REST API interface for application which connect to Central API, to gather the data.

> *Note: All images are provided by OBČIANSKE ZDRUŽENIE TULÁČIK in Brezno utuloktulacik.sk, cute dog is named Tootsie*
> *Note: Application is inspired by Bc. Xuan Linh Phamová*

Let's get creative!

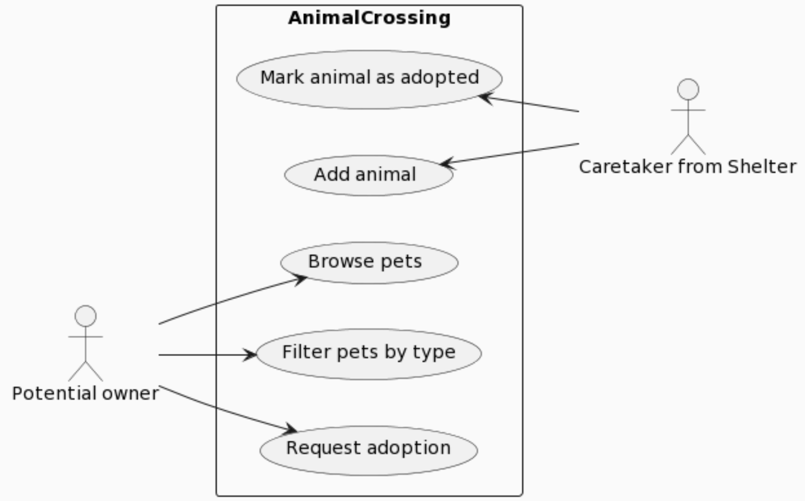
After a basic understanding of the project description, now find the exact functional requirements of our service that needs to be implemented.

- Think about animals...
- What about requests for adoption from users?

Let's get creative!

- Add animal
- Mark animal as adopted
- Browse animals
- Filter animals by type
- Request adoption

You probably thought of more than these, it's good, but we are only demoing.



From functional requirements to resources

What are the resources and endpoints here? What endpoints need to be implemented?

- Add animal
- Browse animals
- Filter animals by type
- Request adoption
- Mark animal as adopted

Think of previous functions and try to find resources.

We have two models, one for Animals and another one for adoption requests.

```
model Animal {
  name        String
  description String
  age         Float
  sex         Boolean
  type        String
  picture     String
  acceptedRequest   Request?
  requests          Request[]
  addedAt   DateTime
  adoptedAt DateTime?
}
```

```
 model Request {
    from       String
    isAccepted Boolean

    animal   Animal

    createdAt DateTime  @default(now())
    deletedAt DateTime?
}
```

From functional requirements to REST API

Add animal

From functional requirements to REST API

`POST /animals`

- JSON Body

From functional requirements to REST API

Browse animals

From functional requirements to REST API

```
GET /animals
```

From functional requirements to REST API

Filter animals by type and limit or other

From functional requirements to REST API

```
GET /animals?type=dog|cat&limit=2
```

From functional requirements to REST API

Request adoption

From functional requirements to REST API

`POST /requests`

- JSON Body

From functional requirements to REST API

Mark animal as adopted

From functional requirements to REST API

`PATCH /requests`

- JSON Body (isAccepted)

That's it let's implement it!

- API is written in Express.js
- Documentation written in OpenAPI
- Testing in Jest