NodeJS
Part 3

jan.schulz@cileria.com

Agenda

1. Project Outline
2. Products and Product-Categories
3. Shop User-Management System

Project Outline

We want to build a backend for our shop.

Project Outline

Shop
Frontend
(Leandro + Valentin)

Shop
Backend
+ Administration
(Jan)

Project Outline

We will create an API using the CRUD principles.

C

R

U

D

Project Outline

We will create an API using the CRUD principles.

Create

Read

Update

Delete

Project Outline

We will create an API using the CRUD principles.

Create – HTTP method POST

Read – HTTP method GET

Update – HTTP method PUT

Delete – HTTP method DELETE

1. Products and Product-Categories

Task 1

- Create a new empty NodeJS-server with one route GET / that returns the JSON { “shop-api“: “1.0“ }

- Implement two GET routes /products and /categories.

  GET /products should return the contents of the table products like this JSON:

  [..., { “id“: “21“, “name“: “MacBook Pro 6“, “price“: 1650, “category_id“: 2 }, ...]

  GET /categories should return the contents of the table categories like this JSON:

  [..., { “id“: 2, “name“: “Mac“ }, ...]

2. User-Management

Customers need to be managed. Therefore, we want to apply CRUD methods on them.

Task 2a:

Create a GET route /customers that gives all information about the table customers as JSON.

2. User-Management

Task 2b:

Extend the table customers by one field “active“ which is either 0 (=inactive) or 1(=active). Set the default value to 0.

Create the PUT route /activate/:userid which expects a PUT body of {status: 1} or {status: 0}. Update the status accordingly.

2. User-Management

Task 2c:

Create the POST route /user which expects a
POST – body like this:

{ firstname: ..., lastname: ..., birthdate: ...,
phone: ..., city: ..., street: ..., email ... }

Insert a new user into the table customers then.
What happens to the fields that are not covered in the post-body? Verify that the email does not exist yet.

2. User-Management

Task 2c:

Create the PUT route /user/:userid which expects a
PUT – body like this:

{ firstname: ..., lastname: ..., birthdate: ...,
phone: ..., city: ..., street: ..., email: ... }

Change the table “customers“ accordingly where the id equals userid.

Note: What happens if certain fields are not specified?
I.e. Only firstname is given to be changed: { firstname: “Stefan“ }

2. User-Management

Task 2e:

Extend the user by a field “deleted“ which is a timestamp.

Create the DELETE route /user/:userid
which expects a parameter userid. Change the customer field “deleted“ to now()