Cloud Application Development

Technologies

- Java
- Spring Boot
- Maven
- Swagger (UI)
- TypeScript
- Angular
- NPM
- PostgreSQL DB
- Docker (-compose)

Backend (Spring Boot)

The backend is developed in Java with the Spring Boot Framework, built with Maven and represents a REST-Service.

Getting Started

To start the Backend run

```
mvn spring-boot:run
```

or execute the jar in the target directory.

Spring Profiles

There are two profiles configured

`dev` which is active by default and uses useful features for development such like

- Using a H2 in Memory DB
- Dropping and creating the DB after start up
- Inserting test data

`prod` which has to be activated manually e.g. like this:

```
java -jar -Dspring.profiles.active=prod backend.jar
```

In production the backend uses a PostgreSQL DB.

Schema creation is done by a SQL script which has to be placed inside the PostgreSQL directory.

REST Endpoints

The Rest Endpoints are documented via Swagger which is available under Swagger UI after start.

Get Media

Post Media

POST /media saveFile

Response Class (Status 200)

int64

Response Content Type */*

Parameters

Parameter	Value	Description	Parameter Type	Data Type
mediaDto	(required)	**mediaDto**	body	Model | Example Value

Parameter content type: application/json

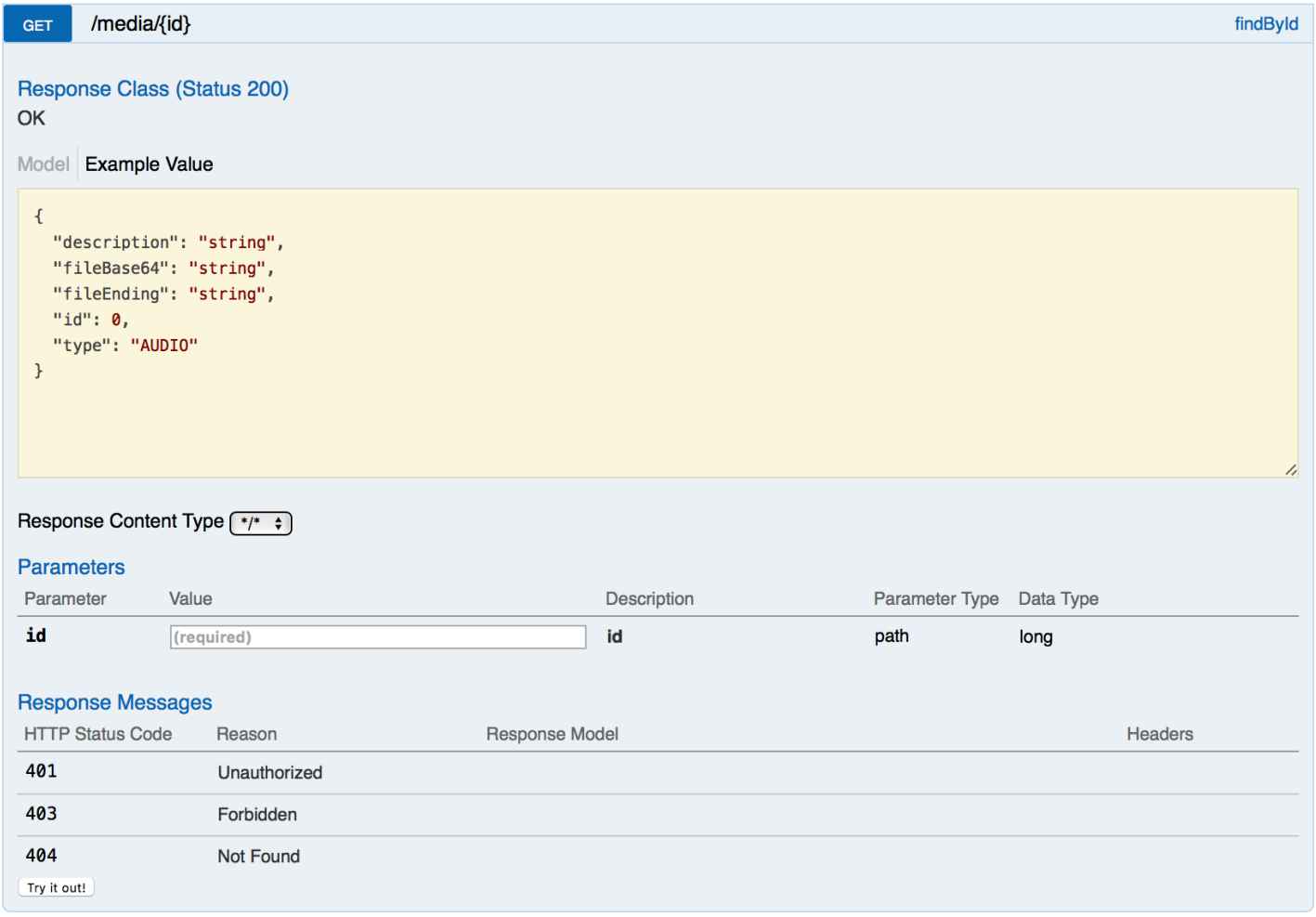
```
{
  "base64": "string",
  "description": "string",
  "fileEnding": "string",
  "type": "string"
}
```

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out!

Find Media By Id



Frontend (Angular)

The frontend is developed using TypeScript and the Framework Angular.

For building the project run:

```
npm install
```

To start the frontend in development mode run:

```
npm start
```

The frontend is also a Maven project and can be built with:

```
mvn install
```

The artifact is a dependency in the backen project and get copied in the spring `static` folder and gets therefore shipped with the backend jar.

Deployment (Docker)

For deployment use:

```
docker-compose up -d
```

The backend and the PostgreSQL DB get wired together via docker-compose and run in single containers.

Each container consists of a single Docker-Image:

- PostgreSQL DB Image (pulled from the Docker Hub)
- App-Image (built from the projects Dockerfile inheriting a Maven image)

App-Image:

```
FROM maven:3.5.3-jdk-8
EXPOSE 8080
COPY ./backend /backend
COPY ./frontend /frontend
RUN mvn -f /frontend clean install
RUN mvn -f /backend clean install

CMD java -jar -Dspring.profiles.active=prod /backend/target/backend*.jar
```

Docker runs the Maven `install` goal while building the image and executes the jar when starting the container.

Therefore the only thing which has to be installed on the Server is Docker.

The hole application is deployed on an AWS EC2-Instance available here.

"Continuous deployment"

A cron job is installed on the EC2 instance which runs the `update.sh` script every 10 minutes.

The script performs a check if there are any new changes in the git repository. If a new commit is available the `buildAndDeploy.sh` script is called which deploys the newer version.