PRÀCTICA 2: REST API

Víctor Sánchez Gassull
Joan Miquel Solé
Grup 20
Q1 2020-21

Entorn i Configuració:

Els dos companys de la pràctica tenim un sistema de Linux de 64 bits basat en ubuntu 20.04, de manera que no vam necessitar utilitzar un entorn virtual com una VM a virtualbox o una virtualització am docker.

Per a realitzar la pràctica necessitarem instalar node a través de les següents comandes:
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs

Hem utilitzat les llibreries morgan (un logger per les requests), cors (per fer les peticions a partir de postman web), fs (per llegir directoris i llegir i escriure fitxers), nodemon (per reiniciar el servidor cada cop que es guarda), express (per fer els endpoints), helmet (per afegir seguretat en les peticions).

Activació del Docker:
docker build -t <tag> .
docker run -p 49160:8080 -d <tag>

per a realitzar consultes s'haurà de fer al port que hi ha a la comanda del docker run, en aquest cas 49160.

Alternatives per resoldre-la:

Aquesta mateixa pràctica l'hauríem pogut resoldre fent servir una altra framework web com per exemple flask/python (més ràpid d'escriure, però més lent) o Golang (a partir de la lliberia net/http o utilitzant llibreries com gorilla/mux o gin).

Hauríem pogut utilitzar una base de dades com mongoDB o mariaDB per poder emmagatzemar les dades d'una forma més segura que no pas un fitxer tipus *JSON*.

Explicar la solució triada:

Tenim tot el codi en un fitxer *index.js*, ja que la solució no demana una escalabilitat òptima.

```
app.use(morgan('dev'));
app.use(helmet());
app.use(express.json());
app.use(cors());
```

Amb la funció app.use(. . .), el que fem és incloure les diferents funcions del backend a l'aplicació.

En aquest cas tindrem dos endpoints a la mateixa ruta (localhost:49160/car):

- GET : retorna tots els rental car creats fins el moment
- POST: Cal enviar un json al body amb els següents camps:
    - Maker
    - Model
    - Days
    - Units

Crear *'RentalCar'*:

- Creem un json amb el body del post, que contè el maker, el model, els dies i les unitats.
- Llegim el json que tenim guardat, que consta d'un array dels diferents *rental car*.
- Guardem el JSON a format string al fitxer i retornem un codi d'status 201 conforme s'ha creat correctament.

```
app.post('/car', (req, res) => {

    let rental = {
        maker: req.body.maker,
        model: req.body.model,
        days:  req.body.days,
        units: req.body.units
    };

    let cars = JSON.parse(fs.readFileSync(data_path));
    console.log(cars);
    cars['car'].push(rental)
    console.log(cars);

    fs.writeFileSync(data_path, JSON.stringify(cars), (err) => {
        if (err) return console.log(err);
        console.log('added new car');
    });
    res.status(201);
    res.send(rental);
    res.end();
});
```

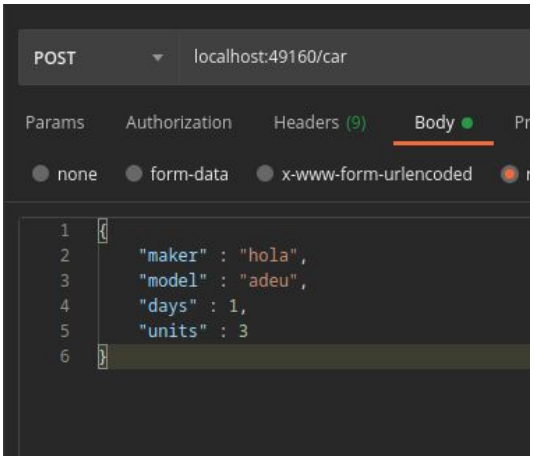
Retorn de tots els *'RentalCar'*:

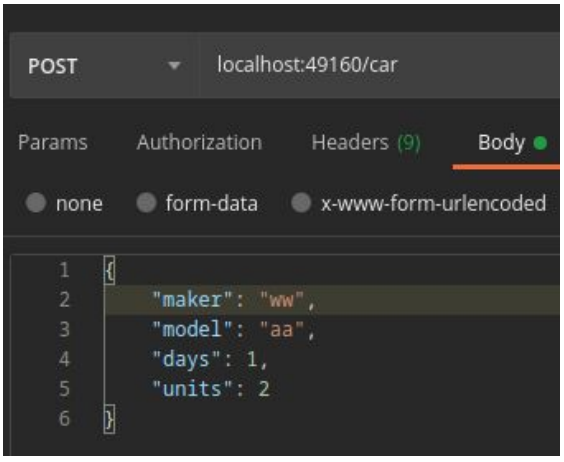
- Retorna el fitxer, parsejat a format JSON.

```
app.get('/car', (req, res) => {
    res.send(JSON.parse(fs.readFileSync(data_path)));
});
```

Avaluació de la solució:

Hem avaluat la correctesa de la solució a través de l'eina *Postman*, fent peticions provant diferents casos. En aquest cas hem fet dues peticions POST amb diferents car rental:





Ara hem fet una petició GET que hauria de retornar

Aspectes positius i negatius de la solució:

Dins el pretext del qual partim, la nostra solució la trobem correcte, ja que hem intentat fer el mínim nombre de cerques a fitxers (només mirem si el fitxer existeix, per crearlo amb el format desitjat quan iniciem el servidor).
Com hauríem pogut millorar la solució? Podríem haver utilitzat una base de dades com per exemple mongoDB o mariaDB (com bé hem dit al punt 2 de la pràctica) i juntar el docker de l'API amb el docker de mongoDB i juntar-ho en un docker-compose per a que ho gestioni.