Exercise 3

In the third exercise, we will

1. add two more features to Blockstarter version 1

- backer invests in project (`makeInvestment`)
- get funding status of a project (`getFunding`)

2. build an initial version of Blockstarter 2.0

- exchange Postgresql with MongoDB and refactor the "Project" model and DAO.

Blockstarter 1.1 API

Please add two more features to Blockstarter 1.1 (with Postgresql database). For this purpose, please extend the Blockstarter 1.0 application. You can

- use the source code from the solution of exercise 2 or
- continue with your own implementation for exercise 2.

Blockstarter 1.1 API: makeInvestment

Implement a transaction that adds `amount` to `Investment` and decreases the `Backer`'s balance accordingly. Here is a suggestion how you might start out, extending *controllers/backers.js*.

```
// A backer makes an investment transaction
let makeInvestment = (req, res, next) => {
  let tR = global.db.transaction((t) => {
    // TODO
  }).then((result) => { // Transaction committed
    res.status(200).send({
      "data": result
    });
  }).catch((err) => { // Transaction rolled back
  callback
    res.status(400).send(err);
  });
};

router.post('/makeInvestment', makeInvestment);
```

Blockstarter 1.1 API: makeInvestment request/response example

Request body for `POST /backers/makeInvestment`:

```
{
        "backer_id" : 1,
        "project_id": 1,
        "amount": 10
}
```

Blockstarter 1.1 API: makeInvestment request/response example

Response body:


```
{
  "data": {
    "backer": {
      "id": 1,
      "name": "Bob the Backer",
      "balance": 0
    },
    "project": {
      "id": 1,
      "category": "tech",
      "title": "Cool project",
      "description": "A Minecraft block-chain.",
      "creator_id": 1
    }
  }
}
```

Blockstarter 1.1 API: getFunding

Implement a function for viewing the total sum of investments of a given project. Here is a suggestion how you might start out, extending *controllers/projects.js*.

```
let getFunding = (req, res, next) => {
  Investment.findAll({
    where: {
      'project_id': req.params.id
    },
    attributes: ['amount']
  }).then((investments) => {
    // TODO calculate sum and return it in the response
  });
};

router.get('/:id/funding', getFunding);
```

Blockstarter 1.1 API: getFunding request/response example

Request: `GET /projects/1/funding`:

Response body:

```
{
  "funding": 10
}
```

Blockstarter 2.0

Refactor the application source code by replacing Postgresql with MongoDB. If you continue with the project setup as given with the exercise 2 solution, this would require mainly two changes:

- replace the *models/project.js* code with a suitable MongoDB schema (you can, for example, use the Mongoose ODM)
- replace the *dao/postgresql.js* with a suitable *dao/mongodb.js*

You do not need to refactor the other models (backer, creator, etc.) and relations for this exercise.

Blockstarter 2.0: *models/project.js*

```
const Schema = require('mongoose').Schema;

const modelSchema = new Schema({
  // TODO
});

// global db
module.exports = db.model('Project', modelSchema);
```

Exercise 3 submission

Please create a zip archive that contains the two directories *blockstarter1* and *blockstarter2* with your solutions of the respective parts of the programming exercise.

- *blockstarter1* should contain Blockstarter 1.1
- *blockstarter2* should contain Blockstarter 2.0

(source code, docker-compose file etc.)