AGENCY 04

Spring Boot Summer School 2020 Hackathon

HEAD OFFICE ADDRESS: AG04 Innovative Solutions | Srebrnjak 36 | Zagreb, Croatia
MAIL ADDRESS: AG04 Innovative Solutions | OIB 99895959831 | Antuna Bauera 29 | Zagreb, Croatia
info@ag04.com | +385 99 292 81 10

ToC

ToC 1

About 2

Assignment 2

Grading 2

Assumptions 3

Requirements 3

User stories 3

SBSS-01: As a heist organiser, I want to add a potential heist member. 3

SBSS-02: As a heist organiser, I want to update a member's skills. 5

SBSS-03: As a heist organiser, I want to remove a member's skill. 6

SBSS-04: As a heist organiser, I want to add a new heist. 7

SBSS-05: As a heist organiser, I want to update required skills on a heist. 9

SBSS-06: As a heist organiser, I want to view members eligible to participate in a heist. 10

When the heist does not exist. 11

SBSS-07: As a heist organiser, I want to confirm members that should participate in a heist. 12

SBSS-08: As a heist organiser, I want to start a heist manually. 13

SBSS-09: As a heist organiser, I want to have a richer API available. 14

1. GET /member/<member_id> 14
2. GET /member/<member_id>/skills 15
3. GET /heist/<heist_id> 16
4. GET /heist/<heist_id>/members 17
5. GET /heist/<heist_id>/skills 18
6. GET /heist/<heist_id>/status 18

SBSS-10: As a heist organiser, I want a heist to be automatically started and finished. 19

SBSS-11: As a heist organiser, I want to view the outcome of a heist. 20

SBSS-12: As a heist organiser, I want to notify heist members upon important events. 21

SBSS-13: As a heist member, I want my skill improvements to be reflected on my member profile. 21

SBSS-14 (optional): As a heist organiser, I want to view everything visually. 21

About

This is an assignment created by Agency04 for the Spring Boot Summer School 2020 Hackathon. The assignment is intended to make you use the knowledge gained during the 5-week long course.

Assignment

Watching the crew from La Casa de Papel (Money Heist) plan and execute two consecutive heists, in a ~50 minutes episode format from the comfort of your couch was easy right?

Behind the scenes, El Profesor had to find a crew with various speciality skills to make sure the heists would be a success.

The application **you** will build is all about automating the process of finding the right people for the right heist, improving people skills before the heist, starting heists and seeing their outcome and finally for us to become the new El Profesor.

Read the assignment carefully and approach it as you would a regular project in the future.

How to tackle the user stories is up to you, but because some of them are linked it would be smart to do them in order.

Grading

Completing all user stories is not required, but the more you manage to do, the more points you will get.

Half the points will be awarded by leveraging automated tests, so make sure you implement the API specification carefully. The other half will be awarded by manually reviewing the code and the usage of best practices.

Assumptions

- There is no security layer (for now), everybody can do anything, let's keep it simple.
- All request fields are required (unless indicated otherwise).
- If a required request field is missing, the HTTP Response Code `400` (Bad Request) should be returned.
- The standard HTTP Response Code is `200` (OK) (unless indicated otherwise).

Requirements

The requirements for the application are specified using user stories below. As previously mentioned, not all the stories are required to be completed, but a higher number of **correctly** implemented stories means more points.

User stories

SBSS-01: As a heist organiser, I want to add a potential heist member.

The purpose of this story is to enable the heist organiser to add potential heist members. The request contains a couple of fields, most of them are self-explanatory.

- The `name` field.
- The `sex` field, allowed values are `F` or `M`.
- The `email` field is unique, hence two robbers having the same email can't exist.
- The `skills` field is an array of objects, containing two fields:
  - The `name` field which is unique in the system (case insensitive).
  - The `level` field is optional and is a string made of asterisk characters (*). It's default value is 1 asterisk character with a maximum value of 10 asterisk characters.
- The `mainSkill` field is optional and references one of the skills from the `skills` array.
- The `status` field indicates the status of the robber, one of the following values is allowed:
  - `AVAILABLE`
  - `EXPIRED`
  - `INCARCERATED`
  - `RETIRED`

The response body should be empty, while the Location header should be set to the resource URI of the newly added member.

Request: `POST /member`

```
{
  "name": "Helsinki",
  "sex": "M",
  "email": "helsinki@ag04.com",
  "skills": [
    {
      "name": "combat",
      "level": "**********"
    },
    {
      "name": "driving",
      "level": "****"
    }
  ],
  "mainSkill": "combat",
  "status": "AVAILABLE"
}
```

Response:

`201` *(Created)*

```
Header: Location /member/<member_id>
```

`400` *(Bad Request)*

When a member with the same email already exists or multiple skills having the same name were provided.

SBSS-02: As a heist organiser, I want to update a member's skills.

The purpose of this story is to enable the heist organiser to update skills of a particular member. Skills can be added or updated using this endpoint.

The `skills` array is optional if the `mainSkill` field is provided and vice versa.

Request: `PUT /member/<member_id>/skills`

```
{
  "skills": [
    {
      "name": "combat",
      "level": "***"
    },
    {
      "name": "money-laundering",
      "level": "*"
    },
    {
      "name": "lock-breaking",
      "level": "****"
    }
  ],
  "mainSkill": "lock-breaking"
}
```

Response:

204 (No Content)

```
Header: Content-Location /member/<member_id>/skills
```

400 (Bad Request)

When the main skill was changed, but the skill is not part of the member's previous or updated skill array or multiple skills having the same name were provided.

404 (Not Found)

When the member does not exist.

SBSS-03: As a heist organiser, I want to remove a member's skill.

The purpose of this story is to enable the heist organiser to remove a skill from a member.

Request: `DELETE /member/<member_id>/skills/<skill_name>`

Response:

`204` *(No Content)*

`404` *(Not Found)*

When the member or the member's skill does not exist.

SBSS-04: As a heist organiser, I want to add a new heist.

The purpose of this story is to enable the heist organiser to add heists. The request contains a couple of fields, most of them are self-explanatory.

- The `name` field which is unique.
- The `location` field.
- The `startTime` - start date and time of the heist, in ISO8601 format
- The `endTime` - end date and time of the heist, in ISO8601 format
- The `skills` field is an array of objects, containing three fields:
  - The `name` field.
  - The `level` field which indicates the required level of the skill.
  - The `members` field which indicates how many members are required to have this particular skill.

The response body should be empty while the Location header should be set to the resource URI of the newly added heist.

Request: `POST /heist`

```
{
  "name": "Fábrica Nacional de Moneda y Timbre",
  "location": "Spain",
  "startTime": "2020-09-05T22:00:00.000Z",
  "endTime": "2020-09-10T18:00:00.000Z",
  "skills": [
    {
      "name": "combat",
      "level": "*****",
      "members": 1
    },
    {
      "name": "combat",
      "level": "*",
      "members": 3
    }
  ]
}
```

The skills array can contain multiple skills with the same name, but with a different level.

Response:

`201` *(Created)*

```
Header: Location /heist/<heist_id>
```

`400` *(Bad Request)*

When a heist with the same name already exists, the `startTime` is after the `endTime`, the `endTime` is in the past or multiple skills with the same name and level were provided.

SBSS-05: As a heist organiser, I want to update required skills on a heist.

The purpose of this story is to enable the heist organiser to update the required skills for the heist. Skills can be added or updated using this endpoint.

Request: `PATCH /heist/<heist_id>/skills`

```
{
  "skills": [
    {
      "name": "driving",
      "level": "***",
      "members": 1
    }
  ]
}
```

Response:

`204` *(No Content)*

```
Header: Content-Location /heist/<heist_id>/skills
```

`400` *(Bad Request)*

When multiple skills with the same name and level were provided.

`404` *(Not Found)*

When the heist does not exist.

`405` *(Method Not Allowed)*

When the heist has already started (SBSS-08).

SBSS-06: As a heist organiser, I want to view members eligible to participate in a heist.

The purpose of this story is to enable the heist organiser to view members eligible to participate in a heist.

Members returned as part of this response should conform to the following rules:

- Their `status` field should be either `AVAILABLE` or `RETIRED`.
- At least one of their skills should match the required skill of the heist and should have a level equal or higher than the required skill level.
- They are not confirmed members of another heist happening in the same time window (SBSS-08).

Request: `GET /heist/<heist_id>/eligible_members`

Response:

200 (OK)

```
{
  "skills": [
    {
      "name": "leadership",
      "level": "*****",
      "members": 1
    }
  ],
  "members": [
    {
      "name": "Berlin",
      "skills": [
        {
          "name": "leadership",
          "level": "**********"
        }
       ]
    }
  ]
}
```

Some of the `member` fields have been omitted from the response example, they are defined as part of the *Request* in the SBSS-01 user story.

`404` *(Not Found)*

When the heist does not exist.

`405` *(Method Not Allowed)*

When the heist members have already been confirmed (SBSS-07).

SBSS-07: As a heist organiser, I want to confirm members that should participate in a heist.

The purpose of this story is to enable the heist organiser to confirm members of a heist.

Expand the `heist` object with a `status` field, the initial value of the field should be `PLANNING`. Once the members are confirmed the status field should be updated to `READY`.

Request: `PUT /heist/<heist_id>/members`

```
{
  "members": ["Berlin", "Denver", "Helsinki", "Moscow", "Nairobi", "Oslo", "Rio", "Tokyo"]
}
```

Response:

`204` *(No Content)*

```
Header: Content-Location /heist/<heist_id>/members
```

`400` *(Bad Request)*

When a member does not exist, at least one of their skills does not match the required skill of the heist, their `status` is not `AVAILABLE` or `RETIRED`, or is already a confirmed member of another heist happening at the same time.

`404` *(Not Found)*

When the heist does not exist.

`405` *(Method Not Allowed)*

When the heist `status` is not `PLANNING`.

SBSS-08: As a heist organiser, I want to start a heist manually.

The purpose of this story is to enable the heist organiser to start a heist manually.

Once the heist is started the status field should be updated to `IN_PROGRESS`.

Request: `PUT /heist/<heist_id>/start`

Response:

`200` *(OK)*

```
Header: Location /heist/<heist_id>/status
```

`404` *(Not Found)*

When the heist does not exist.

`405` *(Method Not Allowed)*

When the heist `status` is not `READY`.

SBSS-09: As a heist organiser, I want to have a richer API available.

The purpose of this story is to enable the heist organiser to view different objects by leveraging the API.

1. GET /member/<member_id>

Request: `GET /member/<member_id>`

Response:

`200` *(OK)*

```
{
  "name": "Oslo",
  "sex": "M",
  "email": "oslo@ag04.com",
  "skills": [
    {
      "name": "combat",
      "level": "**********"
    },
    {
      "name": "driving",
      "level": "****"
    }
  ],
  "mainSkill": "combat",
  "status": "EXPIRED"
}
```

`404` *(Not Found)*

When the member does not exist.

2. GET /member/<member_id>/skills

Request: `GET /member/<member_id>/skills`

Response:

200 (OK)

```
{
  "skills": [
    {
      "name": "combat",
      "level": "***"
    },
    {
      "name": "money-laundering",
      "level": "*"
    },
    {
      "name": "lock-breaking",
      "level": "****"
    }
  ],
  "mainSkill": "lock-breaking"
}
```

404 (Not Found)

When the member does not exist.

3. GET /heist/<heist_id>

Request: `GET /heist/<heist_id>`

Response:

200 (OK)


```
{
  "name": "Fábrica Nacional de Moneda y Timbre",
  "location": "Spain",
  "startTime": "2020-09-05T22:00:00.000Z",
  "endTime": "2020-09-10T18:00:00.000Z",
  "skills": [
    {
      "name": "combat",
      "level": "*****",
      "members": 1
    },
    {
      "name": "combat",
      "level": "*",
      "members": 3
    }
  ],
  "status": "IN_PROGRESS"
}
```


404 (Not Found)

When the heist does not exist.

4. GET /heist/<heist_id>/members

Request: `GET /heist/<heist_id>/members`

Response:

200 (OK)

```
[
  {
    "name": "Berlin",
    "skills": [
      {
        "name": "leadership",
        "level": "**********"
      }
    ]
  },
  {
    "name": "Denver",
    "skills": [
      {
        "name": "combat",
        "level": "**********"
      }
    ]
  }
]
```

404 (Not Found)

When the heist does not exist.

405 (Method Not Allowed)

When the heist `status` is `PLANNING`.

5. GET /heist/<heist_id>/skills

Request: `GET /heist/<heist_id>/skills`

Response:

`200` *(OK)*

```
[
  {
    "name": "driving",
    "level": "***",
    "members": 1
  }
]
```

`404` *(Not Found)*

When the heist does not exist.

6. GET /heist/<heist_id>/status

Request: `GET /heist/<heist_id>/status`

Response:

`200` *(OK)*

```
{
  "status": "IN_PROGRESS"
}
```

`404` *(Not Found)*

When the heist does not exist.

SBSS-10: As a heist organiser, I want a heist to be automatically started and finished.

The purpose of this story is to start and finish a heist automatically.

The heist should be started on `startTime`, and the heist status field updated to `IN_PROGRESS`.

The heist should be finished on `endTime`, and the heist status field updated to `FINISHED`.

SBSS-11: As a heist organiser, I want to view the outcome of a heist.

The purpose of this story is to enable the heist organiser to view the outcome of a heist.

The possible outcomes and repercussions of a heist are determined by the following table:

required members	possible outcome	repercussion
< 50%	`FAILED`	all members `EXPIRED` or `INCARCERATED`
≥ 50% < 75%	`FAILED`	⅔ of the members `EXPIRED` or `INCARCERATED`
≥ 50% < 75%	`SUCCEEDED`	⅓ of the members `EXPIRED` or `INCARCERATED`
≥ 75% < 100%	`SUCCEEDED`	⅓ of the members `INCARCERATED`
100%	`SUCCEEDED`	-

There is a 50:50 chance for the heist member to end up `EXPIRED` or `INCARCERATED`. Their status field should be updated with the new status.

Request: `GET /heist/<heist_id>/outcome`

Response:

`200` *(OK)*

```
{
  "outcome": "FAILED"
}
```

`404` *(Not Found)*

When the heist does not exist.

`405` *(Method Not Allowed)*

When the heist `status` is not `FINISHED`.

SBSS-12: As a heist organiser, I want to notify heist members upon important events.

The purpose of this story is to notify heist members via email upon important events:

- They have been added as members (SBSS-01).
- They have been confirmed to participate in a heist (SBSS-07).
- The heist has started or finished (SBSS-08, SBSS-10).

You can use the following SMTP credentials:

- Server Name: **email-smtp.eu-west-1.amazonaws.com**
- Port: **25**, **465** or **587**
- TLS: **Yes**
- Authentication:
    - SMTP Username: **AKIA3QRJDSTT4P7LI2NJ**
    - SMTP Password: **BDVKBQLtJH5DFJ3isJMP80afrFXxjylOKlMNrdyHw7aD**

SBSS-13: As a heist member, I want my skill improvements to be reflected on my member profile.

So you arrived at the last (non-optional) user story, or are just having a quick read through? Either way, the purpose of the last user story is to reflect the skill improvements a heist member can get while actively doing heists.

For every 86400 seconds (24h) spent on a heist the member's skills (that were required for the heist) increase by 1 asterisk, up to a maximum of 10 asterisks.

Make the 86400 seconds (24h) configurable inside the `application.properties` file in a property named `levelUpTime`.

SBSS-14 (optional): As a heist organiser, I want to view everything visually.

In your favourite frontend technology, implement a simple UI on top of the previously built API.