Purpose

Defining the REST API specifications and the database Schema.

System main participants

Managers, Employees.

Main Activities

1. Managers/Employees can (update, list) projects
2. Managers/Employees can create a project that its owner is a manager
3. Managers can assign participants, working in his department, to a project
4. Employees can assign participants, working in the same department as the owner, to a project

Non-functional requirements

1. Application easily deployable
2. Application is robust and extendible
3. Only managers can be the owner of a project.
4. Participants to a project must be a part of the same department as the owner

1. API Definition

Projects Resource

	Action	Method	Resource URL	Success Code	Error Code
1	Get Projects	GET	/ projects	200	500
3	Add Project	POST	/ projects	201	500, 400
4	Update Project	PUT	/ projects/:id	200	500, 400
5	Add participant	POST	/ projects /:id/participants	201	500, 400

Status codes:

2xx (success category)

1- 200 ok: representing success for GET method.
2- 201 created: representing success for a new resource created for a POST method.

4xx (Client Error Category)

1- 400 Bad Request: The client send a request with a missing param or a request that doesn't fulfill the API exposed by the server.
2- 403 Forbidden: the request is valid but he does not have the rights to access this resource.
3- 404 Not Found: indicates that the requested resource does not exist.

5XX(Server Error Category)

1- 500 internal server error: the request is valid from the client but the error occurred when the server tried to process the request.

2. Database

Project Schema

```
{
  id: String($uuid) PK
  name*: String,
  state: ENUM('active', 'planned', 'done', 'failed'),
  progress: String,
  department*: String,
  owner*: String($uuid) FK→participant
}
```

Participant Schema

```
{
  id*: String($uuid) pk
  role*: ENUM('manager', 'employee'),
  department*: String,
  project_id*: String($uuid) FK→project
}
```

Relationship cardinality: project 1 →* participant.

N.b: for data integrity between two microservices (employee, project) as there are duplicated data of employee in Participant table. The project microservice should subscribe on a topic when an employee data, role or department, is updated.