# Motion Bootcamp – Exam 2021

# The Recrutment Tool

You must create the **REST API** for a recruitment tool, which purpose is to help a HR company with managing a number of available jobs, the recruiters that work for the company, the available candidates for the jobs and their skills.

You need to take care of how to store this information in your database and to provide all the endpoints listed below, working as expected.

Follow the **best practices** for **Object Oriented design** and **high-quality code.**

* **Functionalities**
  + **Create candidate**

Each candidate should have **skills** on the basis of which he then looks for a job, **first name, last name, email and bio**.

* + **Create recruiter**

Each candidate should have **ONE recruiter**, which is created **AUTOMATICALLY** by creating a candidate. Each recruiter should have **5 free interview slots** for conducting interviews and an **experience level**, starting from 1. When adding a new candidate, we need to check if his recruiter **exists**. If it exists, we must **increment** its **level** with one, otherwise we must **create** a new one.

* + **Create** **job**

Each job has **skills** on the basis of which we can look for a candidate. When we create a job, an **interview with a suitable candidate must be created AUTOMATICALLY**. A suitable candidate is one who has **at least one skill required by the job**. If there is more than one suitable candidate, then **create interviews for everyone**. Keep in mind that every candidate **has only one recruiter and that recruiter has only five free slots**. When we create an interview, we must **find** the **recruiter** who is responsible for the candidate and **increase his experience level by one**.

* + **Remove**  **job**

When we remove a particular job, we must **delete all interviews for this job**, as well as **free up the slots of the recruiters responsible for this**.

* **Endpoints**
  + **Candidates**
    - **POST:** **http://localhost:8080/candidates/**
    - **GET**: [**http://localhost:8080/candidates/{id}**](http://localhost:8080/candidates/%7bid%7d)
    - **PUT**: **http://localhost:8080/candidates/{id}**
    - **DELETE**: [**http://localhost:8080/candidates/{id}**](http://localhost:8080/candidates/%7bid%7d)

**Note**: Look at the **example below** to see the **required structure** of the candidates JSON objects

* + ***Skills***
    - **GET: http://localhost:8080/skills/{id}**
    - **GET** (only skills for which we have candidates at this moment): [**http://localhost:8080/skills/active**](http://localhost:8080/skills/active)
  + ***Recruiters***
    - **GET** (only recruiter with available candidates): **http://localhost:8080/recruiters**
    - **GET**: [**http://localhost:8080/recruiters?level={level}**](http://localhost:8080/recruiters?level=%7blevel%7d)
  + ***Jobs***
    - **POST**:  **http://localhost:8080/jobs**
    - **GET**: [**http://localhost:8080/jobs?skill={skillName}**](http://localhost:8080/jobs?skillName=%7bskillName%7d)
    - **DELETE**: [**http://localhost:8080/jobs/{id}**](http://localhost:8080/jobs/%7bid%7d)

**Note**: Look at the **example bellow** to see the **required structure** of the **jobs** JSON objects



* + ***Interviews***
    - **GET**: **http://localhost:8080/interviews**
* **Bonus Task**
  + Think about how to **record each action related to creating a new candidate** to your database and what information you need to save for this action.
* **Constraints**
  + Make sure that you store only **valid** **data** in your database.
  + You can use whichever **relational database** you want.
  + Recruiter and skill are created only if they **DO NOT exist**.
  + If the recruiter does **NOT have free slots**, an **interview should NOT be created**.