

## **Project Overview:**

Create a RESTful Spring Boot API for a job service system, where users can add, retrieve, update, and delete job listings. This project will involve setting up a robust development environment, implementing the application with Spring Boot, handling data with JDBC, ensuring the quality with unit tests, and deploying the service using Docker and Kubernetes.

## **Application Purpose:**

The application is designed to provide a RESTful API for managing job listings. It allows users to add, retrieve, update, and delete job listings, facilitating efficient job management and access.

### **Key Components and Code Structure**

#### *Domain Model*

Job Entity: Represents the job listing with attributes such as

- jobid, title, description, companynamne, skill1, and skill2.

The entity is mapped to a database table using annotations.

#### *Repository Layer*

JobRepository: provides methods to perform CRUD operations on the database without explicit SQL queries.

#### *Service Layer*

JobService: Contains business logic to manage transactions and enforce business rules such as uniqueness of the jobid. Methods include adding a job, retrieving all jobs, updating a job, and deleting a job.

#### *Controller Layer*

JobController: Manages HTTP requests and responses. It uses Spring MVC annotations to define routes for various operations (GET, POST, PUT, DELETE) that interact with the JobService to serve job data.

#### *Exception Handling*

Global Exception Handler: Implemented using @ControllerAdvice to handle exceptions globally across the application. This ensures that all errors provide meaningful feedback to the client, avoiding any unhandled exceptions.

#### *Testing with JUnit 5*

Comprehensive tests for unit testing are written using JUnit 5. These tests validate the functionality of service methods and REST controllers under various conditions.

## *Security Scanning with Grype*

Grype is integrated to scan for vulnerabilities in the project dependencies or container images. The results of these scans are generated in SARIF format and uploaded to GitHub to be reviewed directly within the repository's Security tab.

## *CI/CD Pipeline using GitHub Actions*

A GitHub Actions workflow is set up for continuous integration and deployment. This workflow includes steps to check out the code, set up the environment, build the project with Gradle, run tests, perform a security scan with Grype, and handle the results.

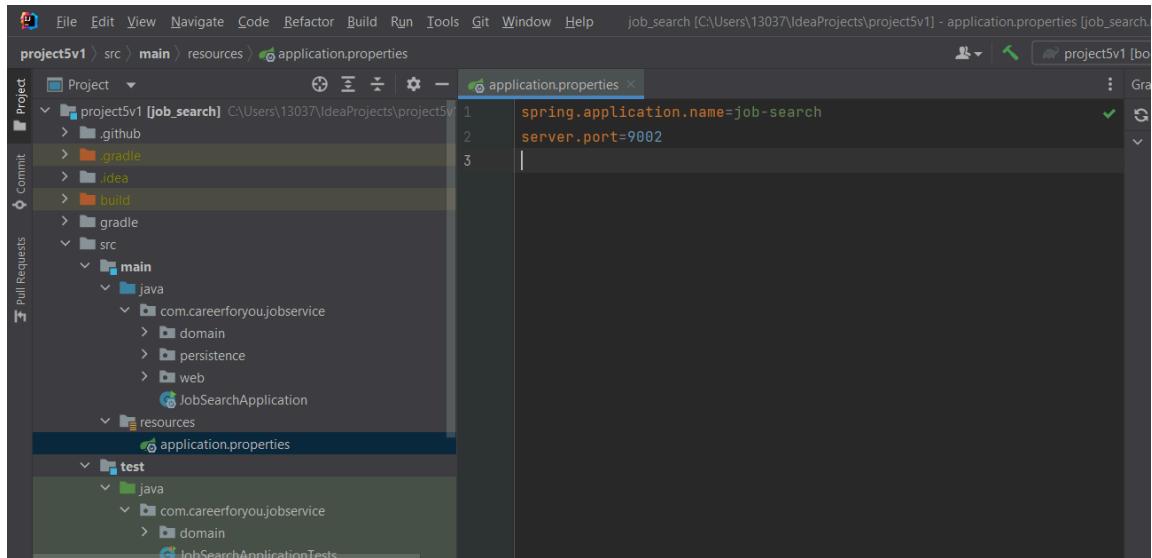
## **Project Configuration**

### API Port Configuration:

The application is configured to have its API endpoints listen on port 9002. This is specified in the application's properties file:

```
spring.application.name=job-search  
server.port=9002
```

As demonstrated above - you can determine the port within IntelliJ - this is one method but is then built into the JAR.



## **Domain Model**

### Job Entity:

The domain model for this application is a Job entity with the following attributes:

**jobid (String):** A unique identifier for the job. The choice between Long and String can depend on whether you anticipate needing alphanumeric IDs.

**title** (String): The title of the job position.

**description** (String): A detailed description of the job responsibilities and expectations.

**companynname** (String): The name of the company offering the job.

**skill1** and **skill2** (String): These are two skills.

Java records are a feature introduced as a way to model immutable data in a concise manner

```
package com.careerforyou.jobservice.domain;

import jakarta.validation.constraints.NotBlank;
import jakarta.validation.constraints.Pattern;

public record Job (

    @NotBlank(message = "The Job ID must be defined.")
    @Pattern(
        regexp = "^([0-9])\$",
        message = "The Job ID format must be valid"
    )
    String jobid,
    @NotBlank(message = "The Job title must be defined.")

    String title,
    @NotBlank(message = "The Job description must be defined.")

    String description,
    @NotBlank(message = "The Company Name must be defined")
    String companynname,

    String skill1,
    String skill2
) {}
```

**jobid**: A String that must be non-blank and match a regular expression. The regex `^([0-9])$` is intended to match a single digit

**title**, **description**, **companynname**: These are String fields marked with `@NotBlank`, ensuring they are not null

**skill1**, **skill2**: These fields are optional as they are not annotated with `@NotBlank`. They can be null or blank, providing flexibility in specifying job skills.

## REST API Specifications

### Endpoint Definitions:

#	Endpoint	HTTP Method	Request Body	Status Code	Response Body	Description
1	/jobs	GET	None	200	List<Job>	Retrieves all jobs in the database.
2	/jobs	POST	Job	201	Job	Adds a new job to the database.
3	/jobs	POST	Job	422	Error Message	Returns an error if a job with the same ID exists.
4	/jobs/{jobid}	GET	None	200	Job	Retrieves a job by its ID.
5	/jobs/{jobid}	GET	None	404	Error Message	Returns an error if no job with the given ID exists.
6	/jobs/{jobid}	PUT	Job	200	Job	Updates an existing job with the given ID.
7	/jobs/{jobid}	PUT	Job	201	Job	Creates a job with the given ID if it does not exist.
8	/jobs/{jobid}	DELETE	None	204	None	Deletes the job with the given ID.

### Application Integration

GET /jobs endpoint would call viewJobList().

GET /jobs/{jobid} endpoint would call viewJobDetails(jobid).

POST /jobs endpoint would call addJobToDatabase(job).

DELETE /jobs/{jobid} endpoint would call removeJobFromDatabase(jobid).

PUT /jobs/{jobid} endpoint would call editJobDetails(jobid, job).

The logic is implemented in the JobService.java class. The methods defined in the service class make it capable of handling various operations such as viewing, adding, editing, and deleting

job listings. The service class effectively abstracts the business logic from the web layer, focusing on operations that interact directly with the data layer.

```
package com.careerforyou.jobservice.domain;

import org.springframework.stereotype.Service;

@Service
public class JobService {
    private final JobRepository jobRepository; // Set up repository

    public JobService(JobRepository jobRepository) {
        this.jobRepository = jobRepository;
    }

    public Iterable<Job> viewJobList() {
        return jobRepository.findAll();
    }

    public Job viewJobDetails(String jobid) {
        return jobRepository.findByJobid(jobid)
            .orElseThrow(() -> new JobNotFoundException(jobid));
    }

    public Job addJobToDatabase(Job job) {
        if (jobRepository.existsByJobid(job.jobid())) {
            throw new JobAlreadyExistsException(job.jobid());
        }
        return jobRepository.save(job);
    }

    public void removeJobFromDatabase(String jobid) {
        jobRepository.deleteByJobid(jobid);
    }

    public Job editJobDetails(String jobid, Job job) {
        return jobRepository.findByJobid(jobid)
            .map(existingJob -> {
                var jobToUpdate = new Job(
                    existingJob.jobid(),
                    job.title(),
                    job.description(),
                    job.companyname(),
                    job.skill1(),
                    job.skill2());
                return jobRepository.save(jobToUpdate);
            })
            .orElseGet(() -> addJobToDatabase(job));
    }
}
```

```
}
```

## JobRepository Interface

The Java interface JobRepository.java outlined below is a custom repository for managing Job entities within the application

```
package com.careerforyou.jobservice.domain;

import java.util.Optional;

public interface JobRepository {
    Iterable<Job> findAll();

    Optional<Job> findByJobid(String jobid);
    boolean existsByJobid(String jobid);
    Job save(Job job);
    void deleteByJobid(String jobid);
}
```

## Exception Handling:

JobAlreadyExistsException.java class is an exception tailored for handling specific domain-related errors in the job management system, particularly when there's an attempt to add a duplicate job entry.

```
package com.careerforyou.jobservice.domain;

public class JobAlreadyExistsException extends RuntimeException {
    public JobAlreadyExistsException(String jobid) {
        super("A job with Jobid " + jobid + " already exists.");
    }
}
```

JobNotFoundException.java class is an exception that effectively handles scenarios where a job search operation fails because the specified job cannot be found.

```
package com.careerforyou.jobservice.domain;

public class JobNotFoundException extends RuntimeException {
    public JobNotFoundException(String jobid) {
        super("The job with Jobid " + jobid + " was not found.");
    }
}
```

## Global Exception Handler:

@ControllerAdvice to creates a global exception handler that catches JobNotFoundException and returns an appropriate HTTP status code

## Persistence:

The InMemoryJobRepository.java class is an implementation of the JobRepository interface that uses an in-memory store to manage Job entities.

Repository Annotation:

The @Repository annotation is used to denote that the class is a bean that interacts with the data layer.

The class uses a ConcurrentHashMap named jobs to store job data.

```
package com.careerforyou.jobservice.persistence;

import com.careerforyou.jobservice.domain.Job;
import com.careerforyou.jobservice.domain.JobRepository;
import org.springframework.stereotype.Repository;

import java.util.Map;
import java.util.Optional;
import java.util.concurrent.ConcurrentHashMap;

@Repository
public class InMemoryJobRepository implements JobRepository {
    private static final Map<String, Job> jobs = new ConcurrentHashMap<>();

    @Override
    public Iterable<Job> findAll() { return jobs.values(); }

    @Override
    public Optional<Job> findByJobid(String jobid) {
        return existsByJobid(jobid) ? Optional.of(jobs.get(jobid)) :
Optional.empty();
    }

    @Override
    public boolean existsByJobid(String jobid) {
        return jobs.get(jobid) != null;
    }

    @Override
    public Job save(Job job) {
        jobs.put(job.jobid(), job);
        return job;
    }

    @Override
    public void deleteByJobid(String jobid) {
        jobs.remove(jobid);
    }
}
```

```
}
```

The InMemoryJobRepository is for development and testing phases of the application.

## Web:

### JobController

Within the com.careerforyou.jobservice.web package the JobController, a central component of the web layer is responsible for handling HTTP requests and invoking the corresponding services related to job management. The controller leverages the JobService to perform CRUD operations and to ensure business logic is applied consistently.

```
package com.careerforyou.jobservice.web;

import com.careerforyou.jobservice.domain.Job;
import com.careerforyou.jobservice.domain.JobService;
import org.springframework.http.HttpStatus;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("job")
public class JobController {
    private final JobService jobService;
    public JobController(JobService jobService) {
        this.jobService = jobService;
    }

    @GetMapping                                         // base
    handler
    public Iterable<Job> get() {
        return jobService.viewJobList();
    }

    @GetMapping("{jobid}")
    public Job getByJobid(@PathVariable String jobid) {
        return jobService.viewJobDetails(jobid);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED) // return 201 if created successfully
    public Job post(@Validated @RequestBody Job job) { return
    jobService.addJobToDatabase(job); }

    @DeleteMapping("{jobid}")
    @ResponseStatus(HttpStatus.NO_CONTENT) // return 204 if a job is deleted
    successfully
    public void delete(@PathVariable String jobid) {
```

```

        jobService.removeJobFromDatabase(jobid) ;
    }

    @PutMapping("/{jobid}")
    public Job put(@PathVariable String jobid, @Validated @RequestBody Job job)
    {
        return jobService.editJobDetails(jobid, job);
    }
}

```

the JobController in the com.careerforyou.jobservice.web package, we have JobControllerAdvice. This class enhances the web layer's robustness by providing centralized exception handling across all controller classes. The @RestControllerAdvice annotation indicates that this class assists all controllers in handling exceptions thrown by request-handling methods.

```

package com.careerforyou.jobservice.web;

import java.util.HashMap;
import java.util.Map;
import com.careerforyou.jobservice.domain.JobAlreadyExistsException;
import com.careerforyou.jobservice.domain.JobNotFoundException;
import org.springframework.http.HttpStatus;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseStatus;
import org.springframework.web.bind.annotation.RestControllerAdvice;

@RestControllerAdvice
public class JobControllerAdvice {

    @ExceptionHandler(JobNotFoundException.class)
    @ResponseStatus(HttpStatus.NOT_FOUND)
    String jobNotFoundHandler(JobNotFoundException ex) {
        return ex.getMessage();
    }

    @ExceptionHandler(JobAlreadyExistsException.class)
    @ResponseStatus(HttpStatus.UNPROCESSABLE_ENTITY)
    String jobAlreadyExistHandler(JobAlreadyExistsException ex) {
        return ex.getMessage();
    }

    @ExceptionHandler(MethodArgumentNotValidException.class)
    @ResponseStatus(HttpStatus.BAD_REQUEST)
    public Map<String, String> handleValidationExceptions(
            MethodArgumentNotValidException ex
    ) {

```

```

        var errors = new HashMap<String, String>();
        ex.getBindingResult().getAllErrors().forEach(error -> {
            String fieldName = ((FieldError) error).getField();
            String errorMessage = error.getDefaultMessage();
            errors.put(fieldName, errorMessage);
        });
        return errors;
    }
}

```

### Testing:

test validation constraints on the Job entity using Jakarta Bean Validation. The class is structured to ensure that Job instances meet expected criteria before being persisted or processed within the application.

#### Test Cases:

```

package com.careerforyou.jobservice.domain;

import java.util.Set;

import jakarta.validation.ConstraintViolation;
import jakarta.validation.Validation;
import jakarta.validation.Validator;
import jakarta.validation.ValidatorFactory;

import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.Test;

import static org.assertj.core.api.Assertions.assertThat;

public class JobValidationTests {

    private static Validator validator;

    @BeforeAll
    static void setUp() {
        ValidatorFactory factory = Validation.buildDefaultValidatorFactory();
        validator = factory.getValidator();
    }

    @Test
    void whenAllFieldsCorrectThenValidationSucceeds() {

```

```

        var job = new Job("1", "Dev", "Dev", "Dev", "Dev", "Dev11", "Dev2");
        Set<ConstraintViolation<Job>> violations = validator.validate(job);
        assertThat(violations).isEmpty();
    }

    @Test
    void whenJobidDefinedButIncorrectThenValidationFails() {
        var job = new Job("one", "Dev", "Dev", "Dev", "Skill1", "Skill2");
        Set<ConstraintViolation<Job>> violations = validator.validate(job);
        assertThat(violations).hasSize(1);
        assertThat(violations.iterator().next().getMessage())
            .isEqualTo("The Job ID format must be valid");
    }
}

```

`whenAllFieldsCorrectThenValidationSucceeds`: This test verifies that a Job object with all fields correctly populated according to the validation constraints. The test creates a Job instance with valid inputs and asserts that the set of constraint violations is empty.

`whenJobidDefinedButIncorrectThenValidationFails`: This test checks the system's response to invalid input specifically in the jobid field, which is constrained by a pattern that requires numeric values. The test intentionally provides a non-numeric jobid to see if the system correctly identifies the violation and provides the expected error message.

The following shows the  
`build\tests\classes\com.careerforyou.jobservice.domain.JobValidationTests.html`

```

<thead>
<tr>
<th>Test</th>
<th>Duration</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
 >whenAllFieldsCorrectThenValidationSucceeds() | 0.201s | passed |
</tr>
<tr>
 >whenJobidDefinedButIncorrectThenValidationFails() | 0.022s | passed |
</tr>
</tbody>
</table>
</div>
<div id="tab1" class="tab">
<h2>Standard output</h2>
<span class="code">


```
15:03:12.196 [Test worker] INFO org.hibernate.validator.internal.util.Version -- HV000001: Hibernate Val
```


</span>
</div>

```

In the screenshot you can see that both unit tests ran and passed, you can see the time stamp for each test under the class ID. This shows that the unit tests are running successfully and are passing.

## GitHub Actions Workflow:

GitHub Actions workflow, "Commit Stage," is designed to handle several crucial operations as part of the CI/CD pipeline. It ensures that JobSearch application is built and tested using Gradle, scanned for vulnerabilities, and that those findings are documented and uploaded for further analysis.

To include the github workflow a .github folder is added to the project, within the .github folder, a folder name workflows is created, and the commit-stage.yml is added into the workflows folder below is the commit-stage.yml file.

Below is a screenshot of the commit-stage.yml file used to set up the github workflow

```
name: Commit Stage
on: push
jobs:
  build:
    name: Build and Test
    runs-on: ubuntu-22.04
    permissions:
      contents: read
      security-events: write
    steps:
      - name: Checkout source code
        uses: actions/checkout@v4
      - name: Setup JDK
        uses: actions/setup-java@v4
        with:
          distribution: temurin
          java-version: 17
          cache: gradle
      - name: Change wrapper permissions
        run: chmod +x
        ./gradlew
      - name: Build with Gradle (Build, unit tests and intergration tests)
        run: ./gradlew build
      - name: Document environment
        run: echo ${{ github.workspace }}
          ls -la ${{ github.workspace }}
      - name: Code vulnerability scanning
        uses: anchore/scan-action@v3
        id: scan
        with:
          path: "${{ github.workspace }}"
          fail-build: false
```

```

    severity-cutoff: high
- name: Examine grype report
  run: cat "./results.sarif"
- name: Upload vulnerability report
  uses: github/codeql-action/upload-sarif@v3
  if: success() || failure()
  with:
    sarif_file: ${{ steps.scan.outputs.sarif }}

```

## Overview of the GitHub Actions Workflow

This workflow is triggered on every push to the repository, and it executes several operations ranging from building the application to conducting security scans.

### Checkout Source Code:

- actions/checkout@v4: Checks out the repository allowing subsequent steps in the workflow to access or modify files within the checked-out repository.

### Setup JDK:

- actions/setup-java@v4: Configures the Java environment using Temurin distribution Java 17.

### Change Wrapper Permissions:

- The chmod +x ./gradlew command ensures the Gradle wrapper script is executable.

### Build with Gradle:

- ./gradlew build: Executes the Gradle build command which compiles the application and runs unit and integration tests.

### Document Environment:

- This step outputs the current workspace path and lists all files and directories within it.

### Code Vulnerability Scanning:

- anchore/scan-action@v3: This step uses Anchore's scan action to perform a security scan of your project's codebase or Docker images. It's configured not to fail the build, even if vulnerabilities are found, and it only reports vulnerabilities.

### Examine Grype Report:

- This command prints the content of the SARIF report generated by the previous step.

### Upload Vulnerability Report:

- github/codeql-action/upload-sarif@v3: This action uploads the SARIF report generated by the Anchore scan to GitHub. GitHub can then analyze and display the scan results, integrating them into the repository's security tab. The condition if: success() || failure() ensures that the upload happens regardless of the success or failure of previous steps.

### Permissions Setup

- Permissions:
  - contents: read allows actions to read the repository content.
  - security-events: write permits actions to write security events, which is necessary for uploading SARIF files.

The goal is to provide a comprehensive automated CI process that includes building, testing, and security scanning. Which we have implemented here with the commit-stage.yml further evidence of this is shown in the below screenshots

Triggered via push 3 days ago

Status: Success | Total duration: 1m 5s | Artifacts: -

**commit-stage.yml**  
on: push

**Build and Test** 57s

**Build and Test** succeeded now in 57s

- > ⚡ Setup Java
- > ✅ Change wrapper permissions
- > ✅ Build with Gradle (Build, unit tests and intergration tests)
- > ✅ Document environment
- > ✅ Code vulnerability scanning

```

1  Run anchore/scan-action@v3
13 /usr/bin/chmod +x /home/runn...
14 /home/runn...
15 [Info] checking github for release tag="v0.74.4"
16 [Info] fetching release script for tag="v0.74.4"
17 [Info] checking github for release tag="v0.74.4"
18 [Info] using release tag="v0.74.4" version="0.74.4" os="linux" arch="amd64"
19 [Info] installed /home/runn...
20 ▾ grype output...
21   Executing: grype -o sarif --fail-on high dir:/home/runn...
22
23   Examine grype report
24   Upload vulnerability report
25   Post Upload vulnerability report
26   Post SetupDK
27   Post Checkout source code
28   Complete job
  
```

Activate Windows Go to Settings to activate Windows

Breaking news Inflation keeps c...

5:35 PM 4/10/2024

The screenshot shows a GitHub Actions build log for a project named 'project5'. The log is titled 'Build and Test' and indicates it succeeded 3 days ago in 57s. It includes two sections: 'Code-vulnerability scanning' and 'Examine grype report'. The 'Code-vulnerability scanning' section shows the command 'Run anchore/scan-action:v3' and its execution, which failed with a high severity issue. The 'Examine grype report' section shows the command 'Run cat ./results.sarif' and its execution, which also failed with a high severity issue. The browser interface includes tabs for 'localhost:localdomain - VMware', 'ChatGPT', 'Home Community - ACC (01)', 'Fixed commit-stage.yml file...', 'Untitled document - Google Doc...', and 'All Bookmarks'. The bottom of the screen shows a Windows taskbar with various icons and the date/time '10:28 PM 4/13/2024'.

The above screenshot shows the grype output as fail on high for the directory:  
home/runner/work/project5/project5

We are also able to use snyk within IntelliJ - snyk is a vulnerability scanning tool that will show us potential vulnerabilities in the code.

```

19     implementation 'org.springframework.boot:spring-boot-starter-validation'
20     implementation 'org.springframework.boot:spring-boot-starter-web'
21     developmentOnly 'org.springframework.boot:spring-boot-devtools'
22     testImplementation 'org.springframework.boot:spring-boot-starter-test'
23     testImplementation 'org.springframework.boot:spring-boot-starter-webflux'
24 }
25
26 tasks.named('test') {
27     useJUnitPlatform()
28 }
29
30

```

**M Allocation of Resources Without Limits or Throttling**

Vulnerability | CWE-770 | CVE-2024-29025 | CVSS 5.3 | SNYK-JAVA-IONETTY-6483812

Vulnerable module: io.netty.netty-codec-http@4.1.107.Final: Allocation of Resources Without Limits or Throttling

Introduced through: org.springframework.boot:spring-boot-starter-webflux@3.2.4

Fixed in: io.netty.netty-codec-http@4.1.108.Final

Exploit maturity: Proof of Concept

Detailed paths

Introduced through: job\_search@0.0.1-SNAPSHOT > org.springframework.boot:spring-boot-starter-webflux@3.2.4 > org.springframework.boot:spring-boot-starter-reactor-netty@3.2.4 > io.projectreactor.netty.reactor-netty-http@1.1.17 >

**M Open Redirect**

Vulnerability | CWE-601 | CVE-2024-22262 | CVSS 5.4 | SNYK-JAVA-ORGSPRINGFRAMEWORK-6597980

Vulnerable module: org.springframework.web

Introduced through: org.springframework.boot:spring-boot-starter-web@3.2.4, org.springframework.boot:spring-boot-starter-webflux@3.2.4

Fixed in: org.springframework.web@5.3.34, @6.0.19, @6.1.6

Exploit maturity: Not Defined

Detailed paths

Introduced through: job\_search@0.0.1-SNAPSHOT > org.springframework.boot:spring-boot-web@3.2.4 > org.springframework.web@6.1.5

## Set up project to use external configuration of the Tomcat App Server port:

In the Spring Boot API project's deployment phase, execute a PowerShell command to set an environment variable SERVER\_PORT to "8888", configuring the application to listen on this port. Subsequently, the java -jar command launched the Jobs API, packaged as job\_search-0.0.1-SNAPSHOT.jar

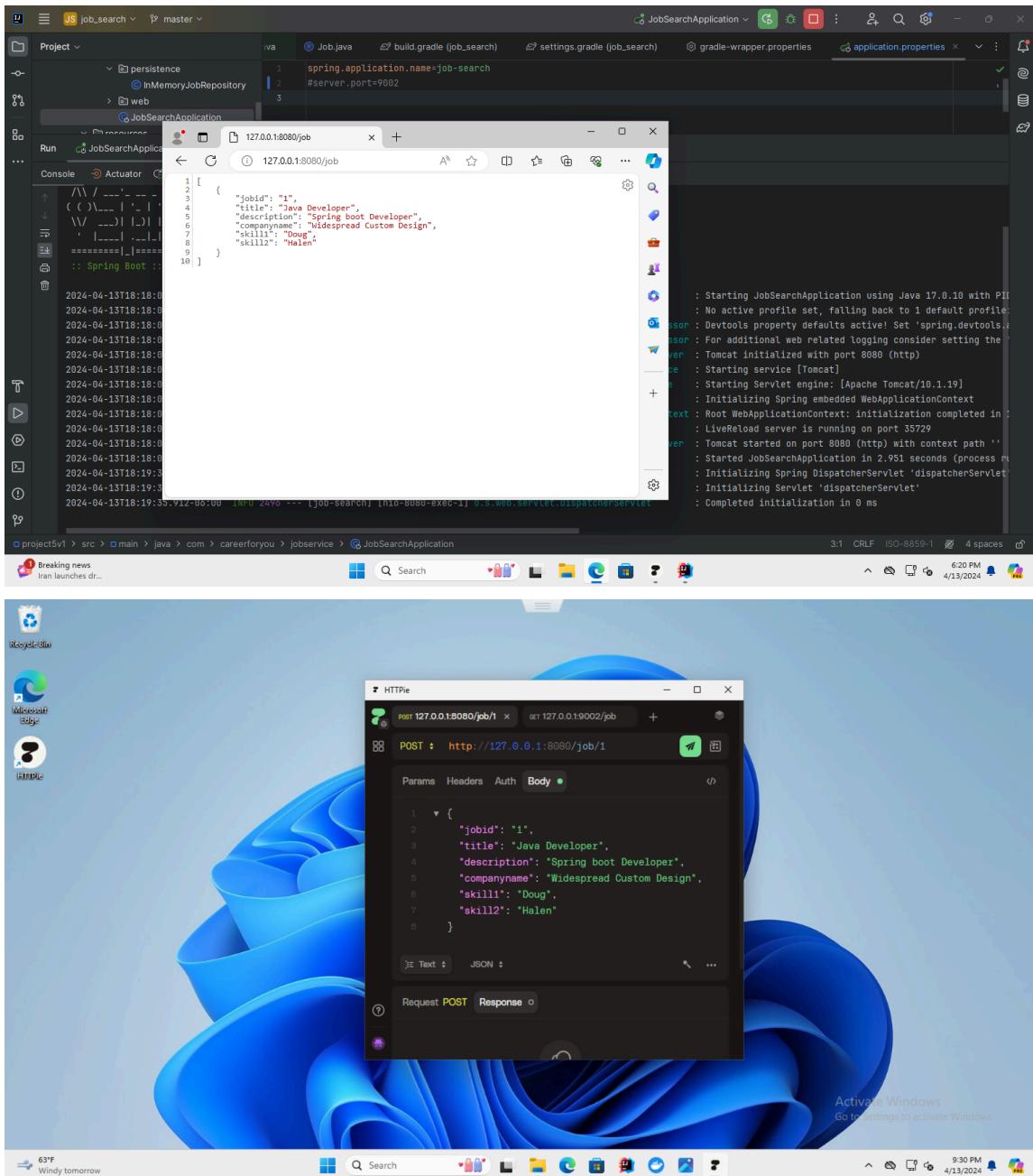
This is done by running the command:

```
$env:SERVER_PORT="8888";java -jar .\build\libs\job_search-0.0.1-SNAPSHOT.jar
```

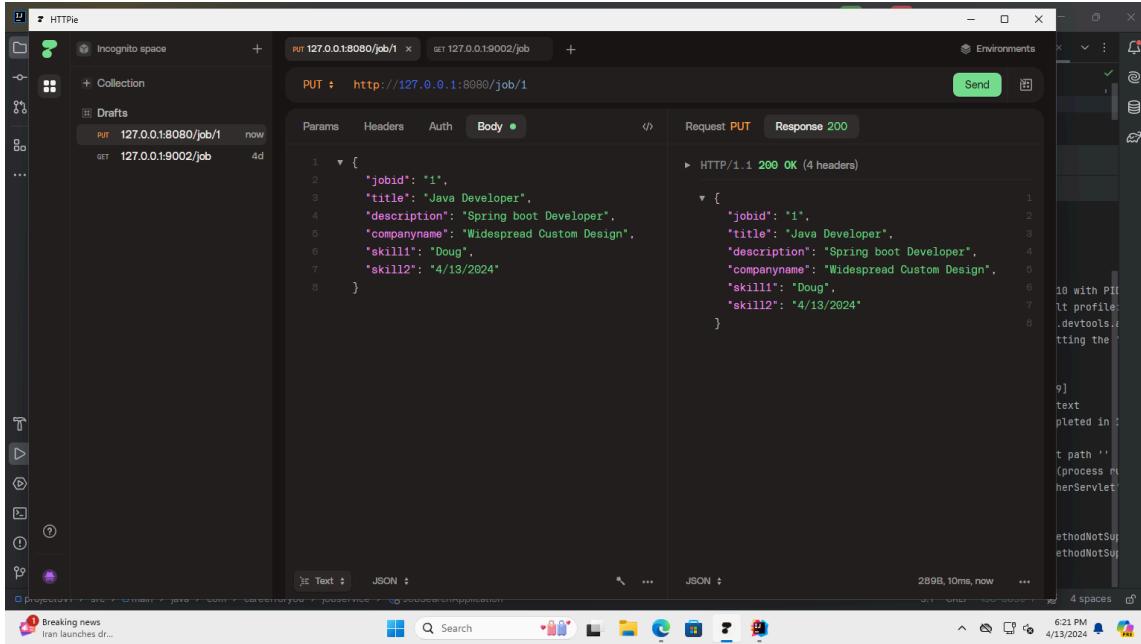
## **Application Testing:**

*IntelliJ (port:8080)*

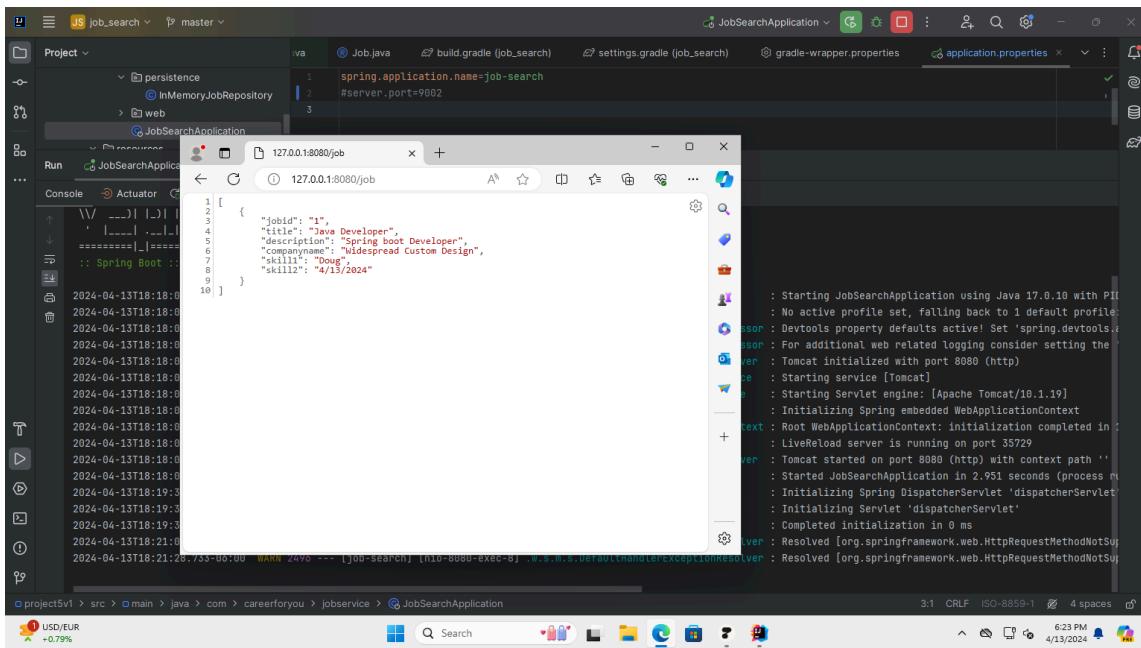
The above screenshot shows the JobSearchApplication running on port 8080 on 4/13/2024



The above screenshots show the JobSearchApplication running with data sent via HTTPie.



The above screenshot shows the PUT request functionality updating the field Skill2 to "Skill2: 4/13/2024" - below shows the updated webpage after the PUT request.



The below screenshot shows the exception for jobAlreadyExists enacted

```
POST : http://127.0.0.1:9002/job
Request POST Response 422
HTTP/1.1 422 Unprocessable Entity (4 headers)
A job with Jobid 1 already exists.
```

The below screenshot shows the jobNotFound exception being enacted

```
GET : http://127.0.0.1:9002/job/2
Request GET Response 404
HTTP/1.1 404 Not Found (4 headers)
The job with Jobid 2 was not found.
```

Below is a screenshot showing that the delete request works

```
DELETE : http://127.0.0.1:8080/job/1
Request DELETE Response 204
HTTP/1.1 204 No Content
Connection: close
Date: Sun, 14 Apr 2024 04:05:17 GMT
```

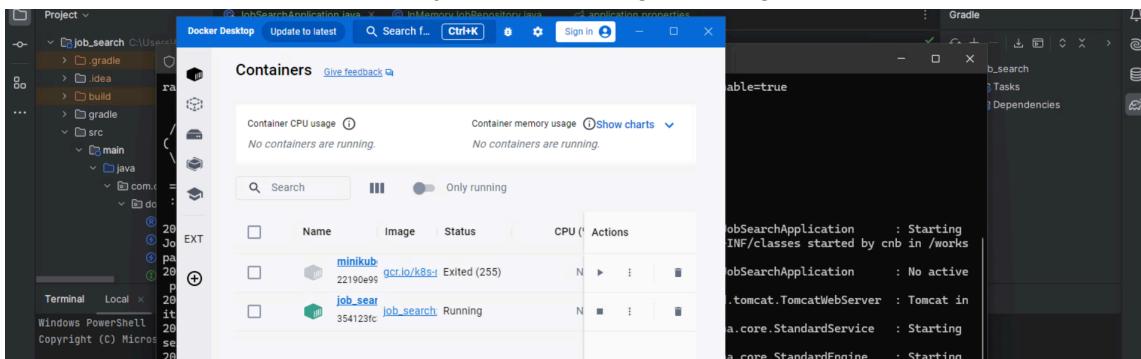
We can now deploy the application to docker using the command “ ./gradlew buildBootImage

Docker (port:9002)

Below we see the docker image we have just created

```
PS C:\Users\hjacobsen\IdeaProjects\JobSearch\job_search> docker images job_search:0.0.1-SNAPSHOT
REPOSITORY          TAG           IMAGE ID            CREATED             SIZE
job_search          0.0.1-SNAPSHOT   1184cd892947   44 years ago    296MB
PS C:\Users\hjacobsen\IdeaProjects\JobSearch\job_search>
```

The screenshot below shows the job\_search image running in docker



Now we can test the docker deployment for functionality

To do this we run the following command:

```
Docker run --rm --name jobsearch -p 9002:9002 job_search:0.0.1-SNAPSHOT
```

```
Administrator: Windows PowerShell > + ...
PS C:\Users\cyberlabadmin> docker run --rm --name jobsearch -p 9002:9002 job_search:0.0.1-SNAPSHOT
```

We can now test the docker deployment

HTTPie

Incognito space + POST 127.0.0.1:9002/job + http://127.0.0.1:9002/job

Params Headers Auth Body Request POST Response 201

```

1  [
2    "jobid": "1",
3    "title": "Dev",
4    "description": "developer",
5    "companyname": "1",
6    "skill1": "1",
7    "skill2": "1"
8  ]

```

HTTP/1.1 201 Created (4 headers)

```

1  [
2    "jobid": "1",
3    "title": "Dev",
4    "description": "developer",
5    "companyname": "1",
6    "skill1": "1",
7    "skill2": "1"
8

```

Activate Windows Go to Settings to activate Windows.

Administrator: Windows PowerShell

```

PS C:\Users\cyberlabadmin> docker run --rm --name jobsearch -p 9002:9002 job_search:0.0.1-SNAPSHOT
Setting Active Processor Count to 2
Calculating JVM memory based on 2762300K available memory
For more information on this calculation, see https://github.com/docker/docker/pull/23851#issuecomment-407903654
Calculated JVM Memory Configuration: -XX:MaxDirectMemorySize=1G -XX:ReservedCodeCacheSize=240M -Xss1M (Total Memory: 2762300K, Thread Count: 250, Loaded Class Count: 11474, Headroom: Enabling Java Native Memory Tracking
Adding 137 container CA certificates to JVM truststore
Spring Cloud Bindings Enabled
Picked up JAVA_TOOL_OPTIONS: -Djava.security.properties=security-properties/java-security.properties -XX:+ExitOnOutOfMemoryError -XX:ActiveProcessorCount=2 -XX:MaxDirectMemorySize=1G -XX:MemoryTracking=summary -XX:+PrintNMTStatistics
1 [
2   "jobid": "1",
3   "title": "Dev",
4   "description": "developer",
5   "companyname": "1",
6   "skill1": "1",
7   "skill2": "1"
8
9 ]

```

Java Security Properties

```

security-properties/java-security.properties -XX:+ExitOnOutOfMemoryError -XX:MaxDirectMemorySize=1G -XX:ReservedCodeCacheSize=240M -Xss1M -XX:+UnlockDiagnosticVMOptions -XX:+UseConcMarkSweepGC

```

Starting JobSearchApplication v0.0.1-SNAPSHOT using Java 17.0.1

No active profile set, falling back to 1 default profile: 'default'

Tomcat initialized with port 9002 (http)

Starting service [Tomcat]

Starting Servlet engine: [Apache Tomcat/10.1.19]

Initializing Spring embedded WebApplicationContext

Root WebApplicationContext: initialization completed in 3754 ms

Tomcat started on port 9002 (http) with context path ''

Started JobSearchApplication in 7.167 seconds (process running for 9.322)

Initializing Spring DispatcherServlet 'dispatcherServlet'

Initializing Servlet 'dispatcherServlet'

Completed initialization in 83 ms

Activate Windows Go to Settings to activate Windows.

S1°F Sunny 8:02 AM 4/14/2024

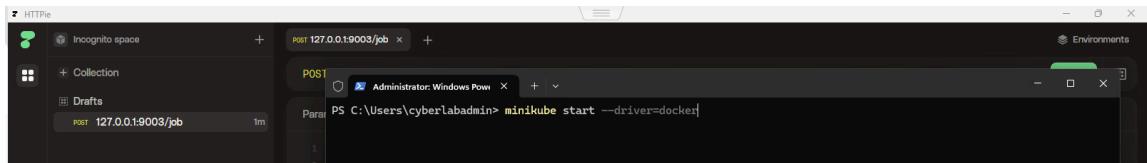
In the screenshots shown above it is shown that the docker deployment (port:9002) is functioning as it should and we can hit each endpoint using HTTPie

Now we can move on to kubernetes deployment

### Kubernetes (port:9003)

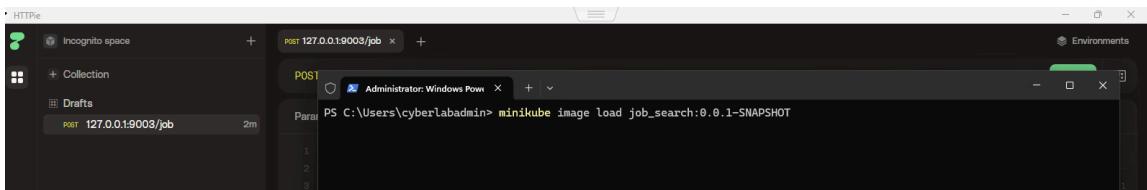
First start minikube using command:

Minikube start -driver=docker



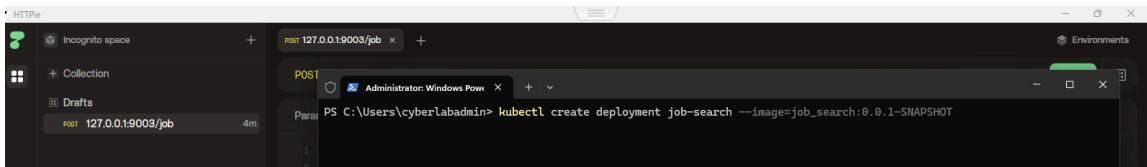
```
PS C:\Users\cyberlabadmin> minikube start --driver=docker
```

Then load the image (job\_search:0.0.1-SNAPSHOT)



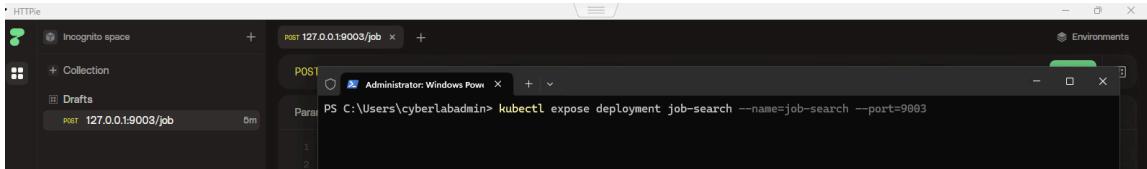
```
PS C:\Users\cyberlabadmin> minikube image load job_search:0.0.1-SNAPSHOT
```

Create deployment (job-search)



```
PS C:\Users\cyberlabadmin> kubectl create deployment job-search --image=job_search:0.0.1-SNAPSHOT
```

Expose deployment (job-search) on port 9003



```
PS C:\Users\cyberlabadmin> kubectl expose deployment job-search --name=job-search --port=9003
```

To initialize the kubernetes deployment we used the command:

`kubectl port-forward service/job-search 9003:9002`

(this command will direct traffic from the port specified (9003) to the port (9002))

The screenshot shows a POST request to `127.0.0.1:9003/job` in HTTPIE. The PowerShell window shows the command `kubectl port-forward service/job-search 9003:9003` being run, which results in an error because there is no service port 9003 defined.

```

PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9003
error: Service job-search does not have a service port 9003
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
1 Forwarding from 127.0.0.1:9003 -> 9002
2 Forwarding from [::1]:9003 -> 9002
3
4
5
6
7
8
9
10

```

The following screenshots show the functionality of the kubernetes (port:9003) deployment:

The screenshot shows a POST request to `127.0.0.1:9003/job` in HTTPIE. The PowerShell window shows the command `kubectl port-forward service/job-search 9003:9003` being run, which results in an error because there is no service port 9003 defined. This is similar to the previous screenshot but with a different URL.

```

PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9003
error: Service job-search does not have a service port 9003
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
Forwarding from 127.0.0.1:9003 -> 9002
Forwarding from [::1]:9003 -> 9002
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
Forwarding from 127.0.0.1:9003 -> 9002
Forwarding from [::1]:9003 -> 9002
Handling connection for 9003
Handling connection for 9003
|_

```

The screenshot shows a POST request to `127.0.0.1:9003/job` in HTTPIE. The PowerShell window shows the command `kubectl port-forward service/job-search 9003:9003` being run, which results in an error because there is no service port 9003 defined. This is similar to the previous screenshots but with a different URL and a slightly different log output.

```

PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9003
error: Service job-search does not have a service port 9003
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
Forwarding from 127.0.0.1:9003 -> 9002
Forwarding from [::1]:9003 -> 9002
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
Forwarding from 127.0.0.1:9003 -> 9002
Forwarding from [::1]:9003 -> 9002
Handling connection for 9003
|_

```

The screenshot shows a DELETE request to `127.0.0.1:9003/job/2` in HTTPIE. The PowerShell window shows the command `kubectl port-forward service/job-search 9003:9003` being run, which results in an error because there is no service port 9003 defined. This is similar to the previous screenshots but with a different URL and a different HTTP method.

```

DELETE : http://127.0.0.1:9003/job/2
Request DELETE Response 204
HTTP/1.1 204 No Content
Connection close
Date Sun, 14 Apr 2024 15:07:19 GMT
|_

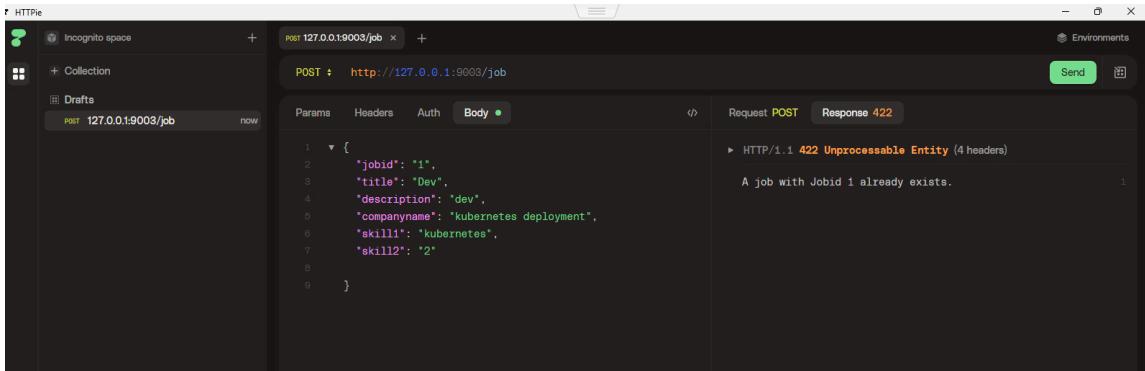
```

The screenshot shows a POST request to `127.0.0.1:9003/job` in HTTPIE. The PowerShell window shows the command `kubectl port-forward service/job-search 9003:9003` being run, which results in an error because there is no service port 9003 defined. This is similar to the previous screenshots but with a different URL.

```

PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9003
error: Service job-search does not have a service port 9003
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
Forwarding from 127.0.0.1:9003 -> 9002
Forwarding from [::1]:9003 -> 9002
PS C:\Users\cyberlabadmin> kubectl port-forward service/job-search 9003:9002
Forwarding from 127.0.0.1:9003 -> 9002
Forwarding from [::1]:9003 -> 9002
Handling connection for 9003
|_

```



HTTPie

Incognito space

+ Collection

Drafts

POST 127.0.0.1:9003/job now

POST : http://127.0.0.1:9003/job

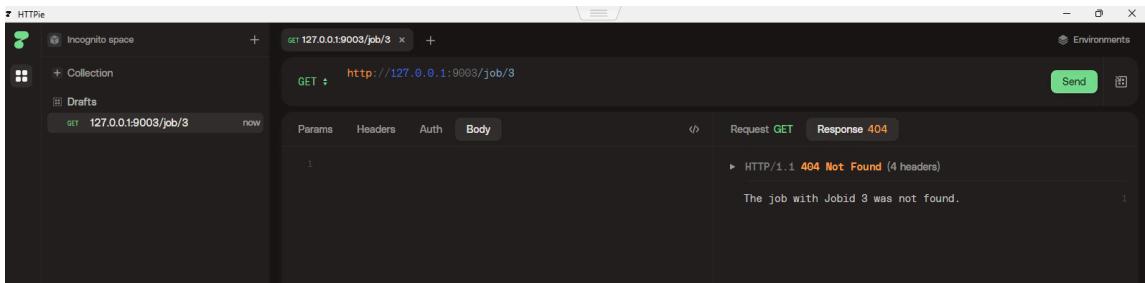
Params Headers Auth Body

```
1 v [
2   "jobid": "1",
3   "title": "Dev",
4   "description": "dev",
5   "companyname": "kubernetes deployment",
6   "skill1": "kubernetes",
7   "skill2": "2"
8 ]
9 }
```

Request POST Response 422

HTTP/1.1 422 Unprocessable Entity (4 headers)

A job with Jobid 1 already exists.



HTTPie

Incognito space

+ Collection

Drafts

GET 127.0.0.1:9003/job/3 now

GET : http://127.0.0.1:9003/job/3

Params Headers Auth Body

```
1
```

Request GET Response 404

HTTP/1.1 404 Not Found (4 headers)

The job with Jobid 3 was not found.