

Jobsearch-assignment

REST API DOCUMENTATION

Szabó Dávid – T9D0K1

Get All Jobs

URL:

/jobs/all

Method:

GET

URL Params

None

Success Response

Code: 200

Content:

```
[
  {
    "title": "Web fejlesztő",
    "description": "Web fejlesztésére felelős akárki",
    "company": "Pók Kft",
    "category": "IT"
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/jobs/all"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.

Filter jobs by title

URL:

/jobs/bytitle/{title}

Method:

GET

URL Params:

Required:

{title}

Title of the job, string, path variable

Success Response:

Code: 200

Content:

```
[
  {
    "title": "Web fejlesztő",
    "description": "Web fejlesztésére felelős akárki",
    "company": "Pók Kft",
    "category": "IT"
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/jobs/bytitle/Web%20fejleszt%C5%91"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.

The passed variables must be url encoded according to RFC 3986.

Filter jobs by company

URL:

/jobs/bytitle/{company}

Method:

GET

URL Params:

Required:

{company}

The lister company's name. String, path variable.

Success Response:

Code: 200

Content:

```
[
  {
    "title": "Takarító néni",
    "description": "Igényes informatikai vállalat
keres kedves takarító nénit nappali munkarenddel",
    "company": "Clean-Office Kft",
    "category": "CLEANING"
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/jobs/bycompany/Clean-Office
%20Kft"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.
The passed variables must be url encoded according to RFC 3986.

Filter jobs by company

URL:

/jobs/bycategory/{category}

Method:

GET

URL Params:

Required:

{category}

The category of the job. Path variableThis parameter is an enumerable, acceptable

values are:

- CLEANING
- IT

Success Response:

Code: 200

Content:

```
[
  {
    "title": "Takarító néni",
    "description": "Igényes informatikai vállalat keres kedves takarító nénit nappali munkarenddel",
    "company": "Clean-Office Kft",
    "category": "CLEANING"
  }, ...
]
```

Error Response:

Code: 500

Content:

```
{
  "Description": "Failed to convert from type [java.lang.String] to type [@org.springframework.web.bind.annotation.PathVariable hu.me.iit.model.JobCategory] for value 'CLEANINGS'; nested exception is java.lang.IllegalArgumentException: No enum constant hu.me.iit.model.JobCategory.CLEANINGS",
  "Value": "CLEANINGS",
  "Error": "Conversion failed!"
}
```

Sample Call:

```
curl "http://localhost:8080/jobs/bycategory/CLEANING"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.
The passed variables must be url encoded according to RFC 3986.

Get All Applicants

URL:

`/applicants/all`

Method:

`GET`

URL Params

None

Success Response

Code: 200

Content:

```
[
  {
    "name": "Nagy Géza",
    "city": "Pilisborzasztó",
    "ekkr": 4,
    "born": 1980
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/applicants/all"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.

Filter applicants by city

URL:

/applicants/bycity/{city}

Method:

GET

URL Params:

Required:

{city}

The city of the current residency of the applicant. String, path value.

Success Response:

Code: 200

Content:

```
[
  {
    "name": "Nagy Géza",
    "city": "Pilisborzasztó",
    "ekkr": "4",
    "born": "1980"
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/applicants/bycity/Pilisborzasztó"
%3%B3"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.

The passed variables must be url encoded according to RFC 3986.

Filter applicants by name

URL:

/applicants/byname/{name}

Method:

GET

URL Params:

Required:

{name}

The name applicant. String, path value.

Success Response:

Code: 200

Content:

```
[
  {
    "name": "Pintér Csaba",
    "city": "Bivalybasznád",
    "ekkr": "6",
    "born": "1995"
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/applicants/byname/Pint%C3%A9r%20Csaba"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.

The passed variables must be url encoded according to RFC 3986.

Filter applicants by year of birth

URL:

/applicants/byborn/{born}

Method:

GET

URL Params:

Required:

{born}

The birth year of the applicant. Integer, path value. $2018 > x > 1990$

Success Response:

Code: 200

Content:

```
[
  {
    "name": "Pintér Csaba",
    "city": "Bivalybasznád",
    "ekkr": "6",
    "born": "1995"
  }, ...
]
```

Sample Call:

```
curl "http://localhost:8080/applicants/byborn/1995"
```

Notes:

In case of no record found, or empty database, the software will give you an empty array.

There won't be error response given for out of range birth years, but you can't input applicants with out of bound birth years, so the response will be empty.

The passed variables must be url encoded according to RFC 3986.

Filter applicants by year of birth

URL:

/applicants/byborn?after={after}&before={before}

Method:

GET

URL Params:

Required:

{after}

The birth year of the applicant. Integer, path value. 2018 >= x >= 1990. The search will only give applicants born this year or after. Optional. See notes.

{before}

The birth year of the applicant. Integer, path value. 2018 >= x >= 1990. The search will only give applicants born this year or before. Optional. See notes.

Success Response:

Code: 200

Content:

```
[
  {
    "name": "Pintér Csaba",
    "city": "Bivalybasznád",
    "ekkr": "6",
    "born": "1995"
  }, ...
]
```

Error Response:

Code: 400

Content:

""

Sample Call:

```
curl "http://localhost:8080/applicants/byborn?
after=1950&before=1990"
```

Notes:

You have to enter at least one of the parameters.

In case of no record found, or empty database, the software will give you an empty array.

There won't be error response given for out of range birth years, but you can't input applicants with out of bound birth years, so the response will be empty.

If you enter both parameters then the search will give applicants born between the two given years.

The passed variables must be url encoded according to RFC 3986.

Filter applicants by EKKR

URL:

/applicants/byekkr?

lowerLimit={lowerLimit}&higherLimit={higherLimit}

Method:

GET

URL Params:

Required:

{lowerLimit}

The lower limit of the required EKKR level. $8 \geq x \geq 0$. Optional. See notes.

{higherLimit}

The higher limit of the required EKKR level. $8 \geq x \geq 0$. Optional. See notes.

Success Response:

Code: 200

Content:

```
[
  {
    "name": "Nagy Géza",
    "city": "Pilisborzasztó",
    "ekkr": "4",
    "born": "1980"
  }, ...
]
```

Error Response:

Code: 400

Content:

""

Sample Call:

```
curl "http://localhost:8080/applicants/byekkr?
lowerLimit=2&higherLimit=5"
```

Notes:

You have to enter at least one of the parameters.

In case of no record found, or empty database, the software will give you an empty array.

There won't be error response given for out of range EKKR levels, but you can't input applicants with out of bound EKKR levels, so the response will be empty.

If you enter both parameters then the search will give applicants born between the two levels.

The passed variables must be url encoded according to RFC 3986.

You can read more about EKKR at

en.wikipedia.org/wiki/European_Qualifications_Framework