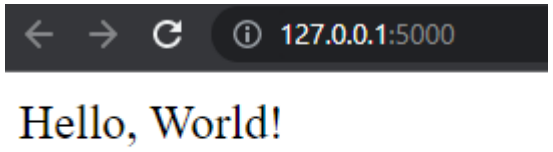
Flask

Flask

- **`Web Framework`**
  - Kreirao ga je Amin Ronacher
  - **`Werkzeug WSGI toolkit + Jinja2 template engine`**
- Često ga nazivaju **`micro framework`**
  - Ne sadrži module koji su zaduženi za komunikaciju sa bazom podataka, validaciju podataka i sl., ali se oni mogu dodati

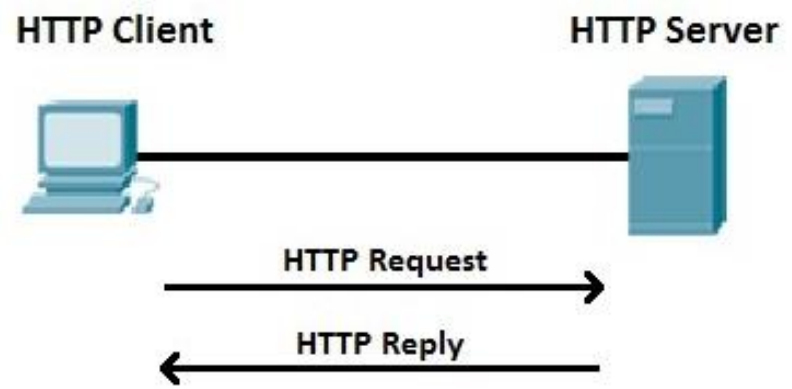
Flask – minimalna aplikacija

```
from flask import Flask
app = Flask ( __name__ )

@app.route ( '/' )
def hello_world ( ):
    return 'Hello, World!'

if __name__ == '__main__':
    app.run ( )
```




```
 * Serving Flask app 'script'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a
production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [28/Feb/2023 21:38:28] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/Feb/2023 21:38:29] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/Feb/2023 21:38:29] "GET /favicon.ico HTTP/1.1" 404 -
```

```
python –m venv ./venv
./venv/Scripts/activate
pip install flask
python script.py
```



Flask

- Objekat klase **`Flask`** predstavlja WSGI aplikaciju
- Metoda **`app.route(rule, options)`** je dekorater koji vezuje funkciju za određeni URL:
  - **`rule`** – URL adresa za koju se vezuje funkcija
  - **`options`** – dodatni parametri (npr. HTTP metoda)
- Metoda **`app.run(host, port, debug, options)`** pokreće aplikaciju:
  - **`host`** – adresa na kojoj se može pristupiti aplikaciji (podrazumevano 127.0.0.1, ukoliko je vrednost 0.0.0.0 aplikaciji se može pristupiti spolja)
  - **`port`** – port na kojem se može pristupiti aplikaciji (podrazumevano 5000)
  - **`debug`** – da li se aplikacija izvršava u **`debug`** režimu (u ovom režimu svaka promena koda je vidljiva odmah)
  - **`options`** – dodatni parametri koji se prosleđuju **`Werkzeug`** serveru

Flask - rutiranje

```
@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
    return 'Hello, World'
```

```
@app.route('/projects/')
def projects():
    return 'The project page'

@app.route('/about')
def about():
    return 'The about page'
```

- Pristup adresama **`localhost/hello/`** i **`localhost/about/`** će proizvesti grešku **`404 Not Found`**
- Pristup adresama **`localhost/projects`** i **`localhost/projects/`** vodi do funkcije **`projects`**

Flask - rutiranje

- Moguće je postići bolju organizaciju ruta korišćenjem **`Blueprint`** objekata

```
from flask import Blueprint

bp = Blueprint ( "foo", __name__, url_prefix = "foo" )

@bp.route ( "/bar" )
def bar ( ):
   return "bar"
```

```
from flask import Flask, request
from . import blueprint

app = Flask ( __name__ )

app.register_blueprint ( blueprint.bp )

if __name__ == '__main__':
   app.run ( debug = True )
```

Flask - rutiranje

- Dozvoljeno je ugneždavanje **`Blueprint`** objekata

```
parent = Blueprint ( 'parent', __name__, url_prefix='/parent' )
child = Blueprint ( 'child', __name__, url_prefix='/child' )
parent.register_blueprint ( child )
app.register_blueprint ( parent )
```

```
url_for ( 'parent.child.create' ) => /parent/child/create
```

Flask - rutiranje

```
from markupsafe import escape

@app.route('/user/<forname>/<surname>')
def show_user_profile(forename, surname):
    return f'User {forename} {surname}'

@app.route('/post/<int:post_id>')
def show_post(post_id):
    return f'Post {post_id}'

@app.route('/path/<path:subpath>')
def show_subpath(subpath):
    return f'Subpath {escape(subpath)}'
```

- Moguće je proslediti parametre kroz putanju
- Dodatno, moguće je navesti način konverzije za prosleđene parametre:

string	(default) accepts any text without a slash
int	accepts positive integers
float	accepts positive floating point values
path	like string but also accepts slashes
uuid	accepts UUID strings

Flask – `request` objekat

- Globalni objekat u kojem su smešteni podaci vezani za tekući zahtev, sadrži sledeće atribute:
  - **`args`** – parametri prosleđeni putem **`query`** stringa (deo URL)
  - **`json`** – telo zahteva
  - **`files`** – sadržaj prosleđenih datoteka
  - **`method`** – HTTP metoda (GET, POST, PUT, …)

```
from flask import request
…
@app.route('/login', methods=['POST', 'GET'])
def login():
    if request.method == 'POST':
            …
    else:
            …

    return "SUCCESS"
```

Flask – `request` objekat

- **`files`** polje **`request`** objekta predstavlja rečnik prosleđenih datoteka, svaka datoteka sadrži **`stream`** objekat pomoću kojeg je moguće pročitati sadržaj

```
@app.route ( "/upload", methods=["POST"] )
def upload ( ):
    content = request.files["file"].stream.read ( )
    return content.decode ( )
```

- Datoteke je moguće sačuvati na serveru, ali je potrebno dodatno konfigurisati aplikaciju

app.config['UPLOAD_FOLDER']	Defines path for upload folder
app.config['MAX_CONTENT_PATH']	Specifies maximum size of file to be uploaded – in bytes

Flask – povratne vrednosti

- Povratne vrednosti su predstavljene objektom klase **`Response`** koja je deo **`flask`** modula
- Objekat se ne mora kreirati ručno, dovoljno je pozvati **`make_response`** funkciju

```
class flask.Response (
        response = None,
        status = None,
        headers = None,
        mimetype = None,
        content_type = None,
        direct_passthrough = False
)
```

```
from flask import make_response

@app.route ( ... )
def foo ( ):
    r = make_response ( ... )
    r.headers['X-Something'] = 'A value'
    return r
```

Flask – povratne vrednosti

- Povratna vrednost funkcije se automatski pretvara u objekat klase **`Response`**. Logika koju **`Flask`** primenjuje za konverziju povratnih vrednosti u objekte klase **`Response`** je sledeća:
  - Ako je povratna vrednost objekat klase **`Response`**, on se direktno vraća klijentskoj strani
  - Ako je povratna vrednost string, kreira se objekat klase **`Response`** sa statusnim kodom 200
  - Ako je u pitanju iterator ili generator koji vraća stringove ili bajtove, tretira se kao tok podataka
  - Ako je povratna vrednost rečnik ili lista, objekat klase **`Response`** se kreira pomoću funkcije **`jsonify`**
  - Ako je povratna vrednost torka, ona mora biti u obliku **`(response, status)`**, **`(response, headers)`** ili **`(response, status, headers)`**, zaglavlja mogu biti predstavljena listom ili rečnikom

Flask – konfiguracija aplikacije

- Postoji predefinisan skup promenljivih okružena koji utiče na rad aplikacije (pogledati dokumentaciju)
- Vrednost ovih promenljivih je moguće definisati na više načina:
  - Direktnim upisom u **`config`** atribut objekta klase **`Flask`**
  - Dodelom vrednosti na nivou OS
  - Dodelom vrednosti u okviru neke datoteke i kasnijim učitavanjem iste
  - Definisanjem vrednosti u okviru **`python`** klasa

```
app.config.from_object ( yourapplication.default_settings )
```

Flask – kontekst aplikacije i zahteva

- Prilikom obrade zahteva kreiraju se kontekst aplikacije i kontekst zahteva koji se po završetku obrade brišu
  - Mehanizam pomoću kojeg je eliminisan problem kružne zavisnosti između modula
  - Moguće im je pristupiti preko globalnih promenljivih **`current_app`** i **`request`**
  - Ukoliko je potrebno odraditi nešto van obrade zahteva, potrebno je ručno kreirati kontekst

```
RuntimeError: Working outside of application context.

This typically means that you attempted to use functionality that
needed to interface with the current application object in some way.
To solve this, set up an application context with app.app_context().
```

```
app = Flask ( __name__ )

with app.app_context ( ):
        init_db ( )
```

Flask – logovi

- **`Flask`** koristi **`logging`** modul
  - Podrazumevana konfiguracija ne ispisuje poruke ispod nivoa **`warning`**

```
@app.route('/login', methods=['POST'])
def login():
    user = get_user(request.form['username'])

    if user.check_password(request.form['password']):
        login_user(user)
        app.logger.info('%s logged in successfully', user.username)
        return redirect(url_for('index'))
    else:
        app.logger.info('%s failed to log in', user.username)
        abort(401)
```

Flask – logovi

```
from logging.config import dictConfig

dictConfig({
    'version': 1,
    'formatters': {'default': {
        'format': '[%(asctime)s] %(levelname)s in %(module)s: %(message)s',
    }},
    'handlers': {'wsgi': {
        'class': 'logging.StreamHandler',
        'stream': 'ext://flask.logging.wsgi_errors_stream',
        'formatter': 'default'
    }},
    'root': {
        'level': 'INFO',
        'handlers': ['wsgi']
    }
})

app = Flask(__name__)
```

Flask - sesije

- HTTP je *stateless* protokol => svaki zahtev se obrađuje nezavisno od prethodnih
- Informacije bitne za nekog korisnika se mogu čuvati u okviru sesije
  - Sesija se pristupa preko globalne promenljive **`session`**

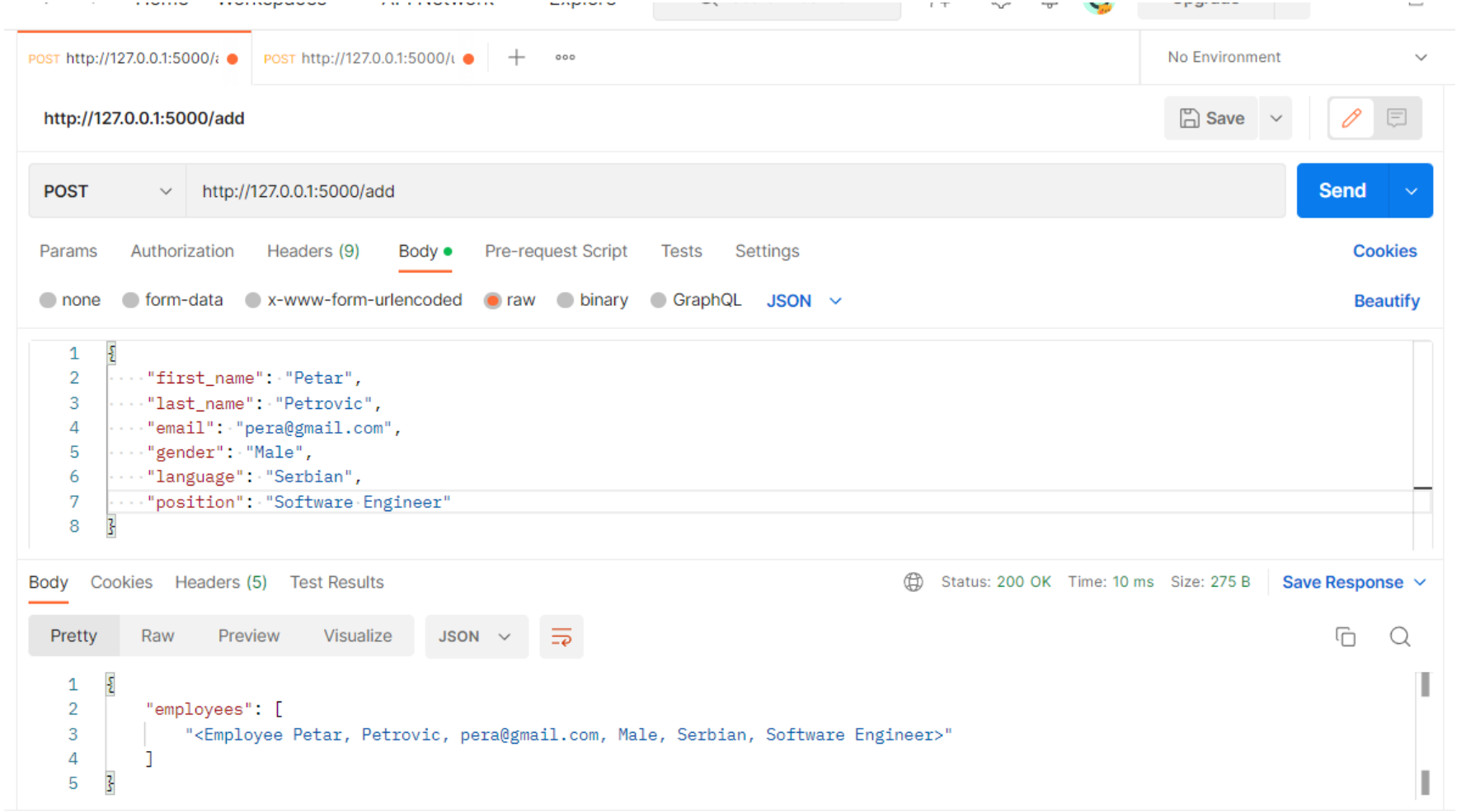
```
from flask import session

# Set the secret key to some random bytes
app.secret_key = b'_5#y2L"F4Q8z\n\xec]/'

@app.route('/')
def index():
    if 'username' in session:
        return f'Logged in as {session["username"]}'
    return 'You are not logged in'

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        session['username'] = request.form['username']
        return redirect(url_for('index'))
    return '''
        <form method="post">
            <p><input type=text name=username>
            <p><input type=submit value=Login>
        </form>
    '''

@app.route('/logout')
def logout():
    # remove the username from the session if it's there
    session.pop('username', None)
    return redirect(url_for('index'))
```

Zadatak

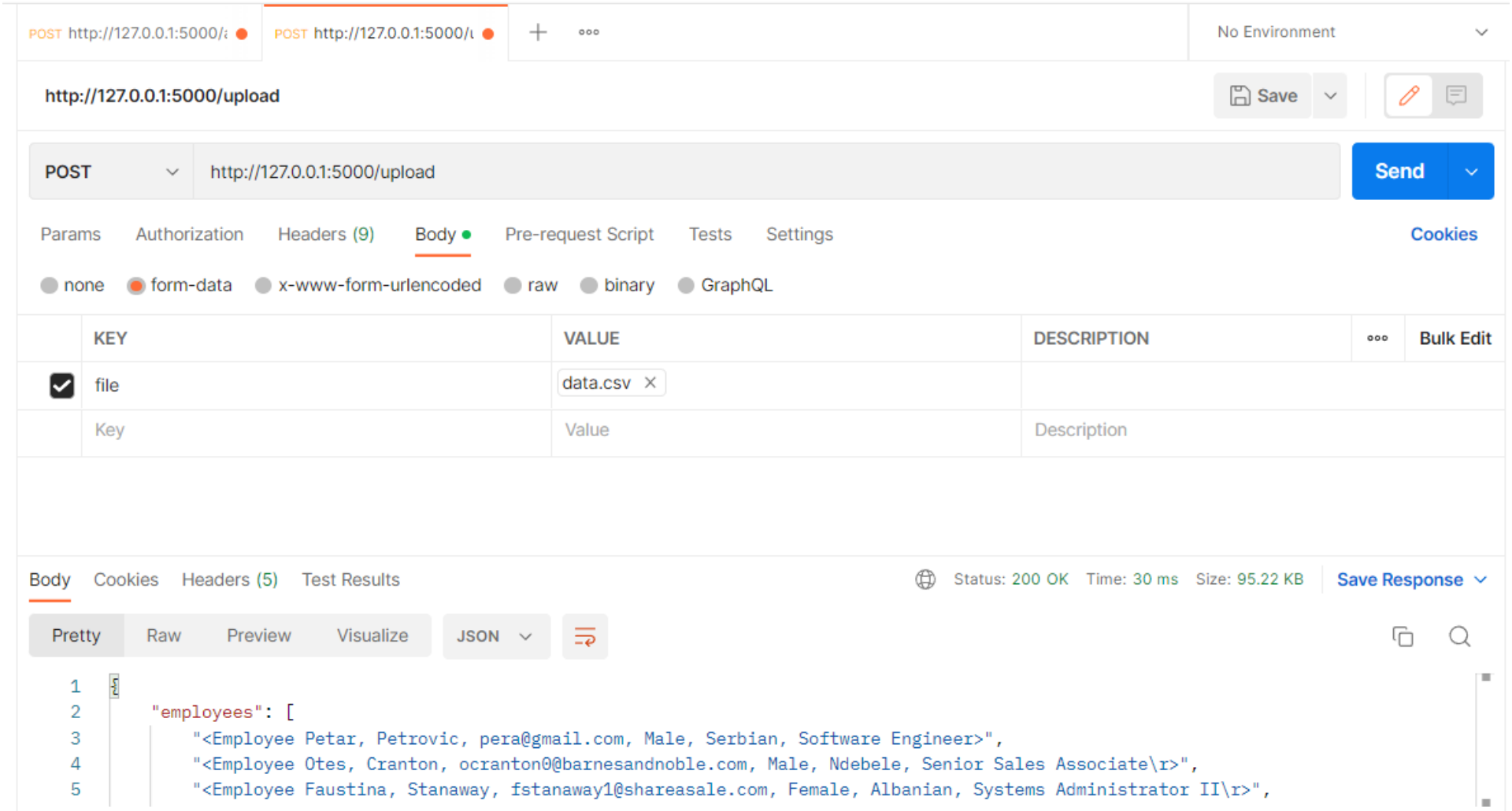
- Napisati jednostavnu veb aplikaciju koja korisnicima omogućava dodavanje podataka o zaposlenima i pretragu nad istim podacima:
  - Za svakog zaposlenog se čuva ime, prezime, email adresa, pol, jezik kojim govore i njihova pozicija
  - Omogućiti dodavanje pojedinačnih zaposlenih, kao i dodavanje putem CSV datoteke
  - Omogućiti pretragu po svi atributima

Zadatak

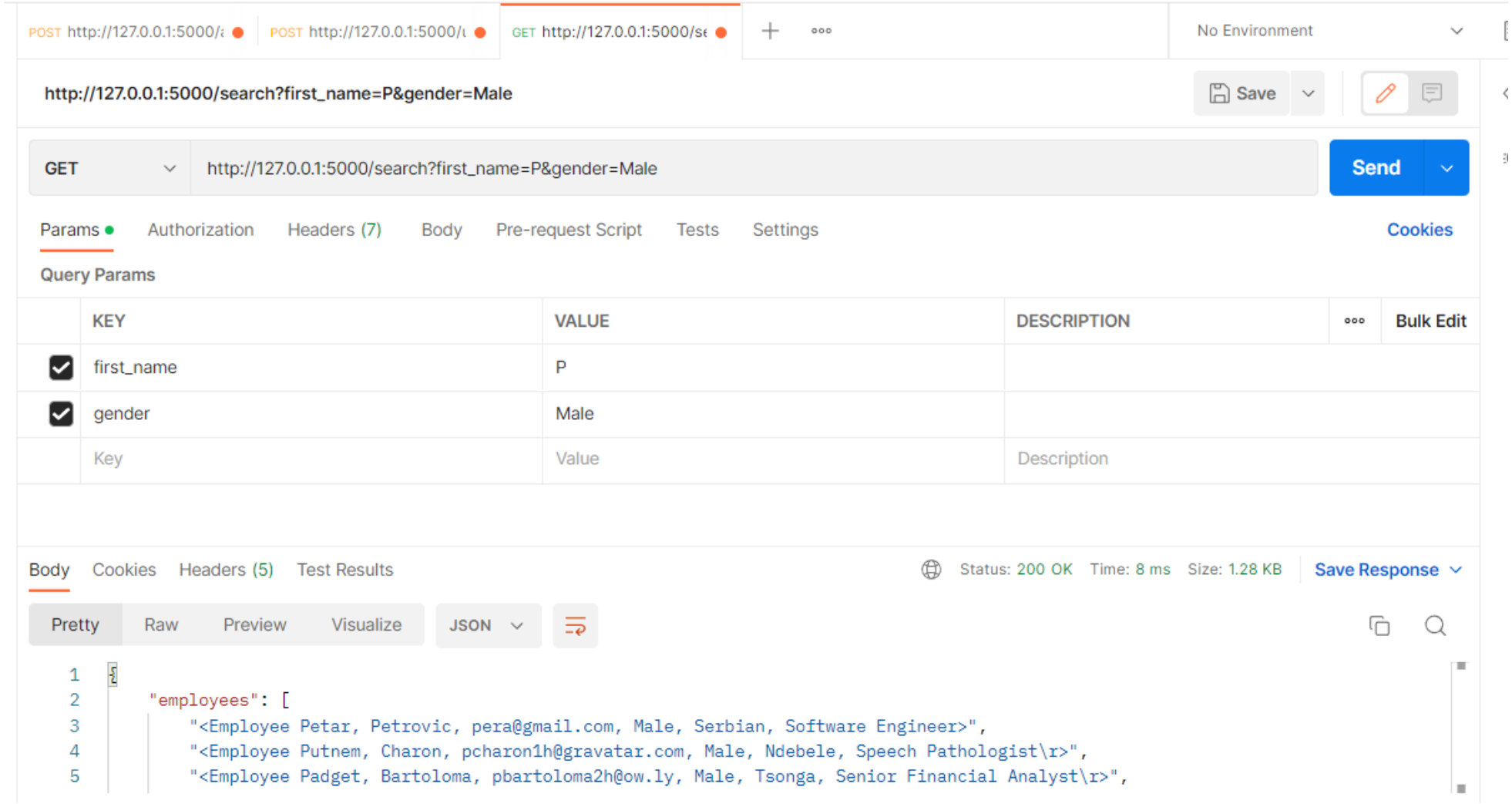
- Aplikacija se može testirati korišćenjem Postman alata



Zadatak


POST http://127.0.0.1:5000/
POST http://127.0.0.1:5000/
No Environment
http://127.0.0.1:5000/upload
Save
POST
http://127.0.0.1:5000/upload
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
KEY
VALUE
DESCRIPTION
Bulk Edit
file
data.csv
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 30 ms
Size: 95.22 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 {
2 "employees": [
3 "<Employee Petar, Petrovic, pera@gmail.com, Male, Serbian, Software Engineer>",
4 "<Employee Otes, Cranton, ocranton0@barnesandnoble.com, Male, Ndebele, Senior Sales Associate\r>",
5 "<Employee Faustina, Stanaway, fstanaway1@shareasale.com, Female, Albanian, Systems Administrator II\r>",

Zadatak


POST http://127.0.0.1:5000/
POST http://127.0.0.1:5000/
GET http://127.0.0.1:5000/se
No Environment
http://127.0.0.1:5000/search?first_name=P&gender=Male
Save
GET
http://127.0.0.1:5000/search?first_name=P&gender=Male
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
first_name
P
gender
Male
Key
Value
Description
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 8 ms
Size: 1.28 KB
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"employees": [
"<Employee Petar, Petrovic, pera@gmail.com, Male, Serbian, Software Engineer>",
"<Employee Putnem, Charon, pcharon1h@gravatar.com, Male, Ndebele, Speech Pathologist\r>",
"<Employee Padget, Bartoloma, pbartoloma2h@ow.ly, Male, Tsonga, Senior Financial Analyst\r>",