```
# this is app.py

import os
import random
import argparse
import urlparse

import requests
import psycopg2
import bottle
from bottle import route, run, template, static_file, request, post, response

bottle.BaseRequest.MEMFILE_MAX = 10000000 #(10M)

app = bottle.Bottle()

urlparse.uses_netloc.append("postgres")
# The Database URl is loaded as an Environment variable
url = urlparse.urlparse(os.environ["DATABASE_URL"])

conn = psycopg2.connect(
    database=url.path[1:],
    user=url.username,
    password=url.password,
    host=url.hostname,
    port=url.port
)


def setup_database():
  print("[INFO]: Setting up database")
  cursor = conn.cursor()
  cursor.execute("""
    CREATE TABLE IF NOT EXISTS visualisations (
      id serial PRIMARY KEY,
      svg text
    )
  """)



@app.route('/')
def home():
  return static_file("./index.html", root="./Session2/")
```

```
@app.route('/hello/<name>')
def index(name):
    return template('<b>Hello {{name}}</b>!', name=name)

@app.route('/api/random-test')
def random_test():
  return {'value': random.random()}

@app.post('/api/anon-search')
def anon_search():
  url_to_be_fetched = request.forms.get('url')
  reply = requests.get(url_to_be_fetched)
  return {
    'status': 'OK',
    'url': url_to_be_fetched,
    'content': reply.text
  }

@app.route('/static/<pathname>')
def home(pathname):
  return static_file(pathname, root="./Session2/static")

@app.post('/api/save/visualisation')
def savevisu():
  print("Saving visualisation...")
  svg_data = request.body.getvalue()
  print("    Received SVG data: %d bytes"%len(svg_data) )
  cursor = conn.cursor()
  cursor.execute("INSERT INTO visualisations (svg) VALUES (%s) ", (svg_data,) )
  conn.commit()
  print("    Transaction committed." )
  return {'status': 'OK'}

@app.route('/api/vis-gallery')
def showvizs():
  cursor = conn.cursor()
  cursor.execute("SELECT * FROM visualisations;")
  visualisations = [r[1] for r in  cursor.fetchmany(30)]
  return {
    "status": "OK",
    "visualisations": visualisations,
    "count": len(visualisations)
  }

def gen_results(num):
  return [{'weight':random.random(), 'personalisation':random.random() } for x i
```

```
def item_weight(item):
  return item['weight']

def item_personalisation(item):
  return item['personalisation']

@app.route('/api/search')
def search():
  query = request.query.get('query')
  return {
    'query': query,
    'sorted_items': sorted(gen_results(100), key=item_personalisation)
  }

if __name__ == "__main__":
  parser = argparse.ArgumentParser(description='Process some integers.')
  parser.add_argument('--port', metavar='PORT', type=int, help='Port to serve on
  parser.add_argument('--setup', help="Setup database")
  args = parser.parse_args()

  if args.setup and os.environ.get("DATABASE_URL"):
    setup_database()

  port = None
  if os.environ.get('PORT'):
    port = os.environ.get('PORT')
  elif args.port:
    port = args.port
  else:
    raise Exception("Port not configured!")

  app.run(host='0.0.0.0', port=port)
```