Aufgabe 1: namegen Service

Namegen Service:

```
FROM mcr.microsoft.com/dotnet/core/sdk:3.0

WORKDIR /root

COPY namegen namegen

RUN dotnet build namegen

EXPOSE 5000

CMD dotnet run --urls "http://0.0.0.0:5000"
```

Node App:

```
FROM node:latest

WORKDIR /root

COPY web web

WORKDIR web

RUN npm i

EXPOSE 3000

CMD node app.js
```

Compose file:

```yaml
version: '3'

services:
  web:
    container_name: 'nodeapp_c1'
    image: nodeapp:0.0
    command: node app.js
    links:
      - "namegenserver:namegen"
    ports:
      - "3000:3000"
    networks:
      - webnet
    depends_on:
      - namegenserver
  namegenserver:
    container_name: 'namegen_c1'
    image: namegen:0.0
    command: dotnet run --project namegen --urls "http://0.0.0.0:5000"
    expose:
      - 5000
    networks:
      - webnet

networks:
  webnet:
```

Aufgabe 2: Webshop

Product.java:

```java
package ex3.bgd;

import java.util.HashMap;
import java.util.Map;

public class Product {
    private String hash;
    private String name;
    private String category;
    private String description;
    private double price;

    public Product() {

    }

    public String getHash() {
        return hash;
    }

    public void setHash(String hash) {
        this.hash = hash;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getCategory() {
        return category;
    }

    public void setCategory(String category) {
        this.category = category;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public Map<String, String> toMap() {
        Map<String, String> result = new HashMap<>();
        result.put("hash", hash);
        result.put("name", name);
        result.put("category", category);
        result.put("description", description);
        result.put("price", Double.toString(price));

        return result;
    }

    public static Product fromMap(Map<String, String> map) {
        Product result = new Product();

        result.setHash(map.get("hash"));
        result.setName(map.get("name"));
        result.setCategory(map.get("category"));
        result.setDescription(map.get("description"));
        result.setPrice(Double.parseDouble(map.get("price")));

        return result;
    }

}
```

ProductController.java:

```java
package ex3.bgd;

import java.util.ArrayList;
import java.util.List;
import java.util.Set;

import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;

import redis.clients.jedis.Jedis;

@RestController
public class ProductController {
    private static Jedis jedis = new Jedis("localhost");
    private static String REDIS_PRODUCTDATA = "pd";
    private static String SORTED_KEY = "sorted";

    // Speichern/Hochladen von Dateien
    @RequestMapping(value = "/product", method = RequestMethod.POST)
    public void file(@RequestBody Product product) {
        product.setHash(Integer.toString(product.getName().hashCode()));

        jedis.sadd(REDIS_PRODUCTDATA, product.getHash());
        jedis.sadd(product.getCategory(), product.getHash());
        jedis.hset(product.getHash(), product.toMap());

        jedis.zadd(SORTED_KEY, product.getPrice(), product.getHash());
    }

    @RequestMapping(value = "/product", method = RequestMethod.GET, params = { "name" })
    public Product product(String name) {
        return Product.fromMap(jedis.hgetAll(Integer.toString(name.hashCode())));
    }

    @RequestMapping(value = "/products", method = RequestMethod.GET)
    public List<Product> products() {
        List<Product> products = new ArrayList<>();

        Set<String> hashes = jedis.smembers(REDIS_PRODUCTDATA);

        for (String hash : hashes) {
            Product product = Product.fromMap(jedis.hgetAll(hash));
            products.add(product);
        }

        return products;
    }

    @RequestMapping(value = "/category", method = RequestMethod.GET, params = { "category" })
    public List<Product> category(String category) {
        List<Product> products = new ArrayList<>();

        Set<String> hashes = jedis.smembers(category);

        for (String hash : hashes) {
            Product product = Product.fromMap(jedis.hgetAll(hash));
            products.add(product);
        }

        return products;
    }

    @RequestMapping(value = "/cheapest", method = RequestMethod.GET)
    public List<Product> filesPrio() {
        List<Product> products = new ArrayList<>();

        Set<String> hashes = jedis.zrange(SORTED_KEY, 0, 4);

        for (String hash : hashes) {
            Product product = Product.fromMap(jedis.hgetAll(hash));
            products.add(product);
        }

        return products;
    }

}
```