

Aufgabe 1: namegen Service

1. *a* Backend:

```
1 FROM mcr.microsoft.com/dotnet/core/sdk:3.0
2
3 WORKDIR /root
4
5 COPY namegen namegen
6
7 RUN dotnet build namegen
8
9 EXPOSE 5000
10
11 CMD dotnet run --urls "http://0.0.0.0:5000"
```

2. *b* Frontend:

```
1 FROM node:latest
2
3 WORKDIR /root
4
5 COPY web web
6
7 WORKDIR web
8
9 RUN npm i
10
11 EXPOSE 3000
12
13 CMD node app.js
```

Aufgabe 2: Webshop

1. Product.java:

```
1  package ex3.bgd;
2
3  import java.util.HashMap;
4  import java.util.Map;
5
6  public class Product {
7      private String hash;
8      private String name;
9      private String category;
10     private String description;
11     private double price;
12
13     public Product() {
14
15     }
16
17     public String getHash() {
18         return hash;
19     }
20
21     public void setHash(String hash) {
22         this.hash = hash;
23     }
24
25     public String getName() {
26         return name;
27     }
28
29     public void setName(String name) {
30         this.name = name;
31     }
32
33     public String getCategory() {
34         return category;
35     }
36
37     public void setCategory(String category) {
38         this.category = category;
39     }
40
41     public String getDescription() {
```

```
42         return description;
43     }
44
45     public void setDescription(String description) {
46         this.description = description;
47     }
48
49     public double getPrice() {
50         return price;
51     }
52
53     public void setPrice(double price) {
54         this.price = price;
55     }
56
57     public Map<String, String> toMap() {
58         Map<String, String> result = new HashMap<>();
59         result.put("hash", hash);
60         result.put("name", name);
61         result.put("category", category);
62         result.put("description", description);
63         result.put("price", Double.toString(price));
64
65         return result;
66     }
67
68     public static Product fromMap(Map<String, String> map) {
69         Product result = new Product();
70
71         result.setHash(map.get("hash"));
72         result.setName(map.get("name"));
73         result.setCategory(map.get("category"));
74         result.setDescription(map.get("description"));
75         result.setPrice(Double.parseDouble(map.get("price")));
76
77         return result;
78     }
79
80 }
```

2. ProductController.java:

```
1  package ex3.bgd;
2
3  import java.util.ArrayList;
4  import java.util.List;
5  import java.util.Set;
6
7  import org.springframework.web.bind.annotation.RequestBody;
8  import org.springframework.web.bind.annotation.RequestMapping;
9  import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.bind.annotation.RestController;
11
12 import redis.clients.jedis.Jedis;
13
14 @RestController
15 public class ProductController {
16     private static Jedis jedis = new Jedis("localhost");
17     private static String REDIS_PRODUCTDATA = "pd";
18     private static String SORTED_KEY = "sorted";
19
20     // Speichern/Hochladen von Dateien
21     @RequestMapping(value = "/product", method = RequestMethod.POST)
22     public void file(@RequestBody Product product) {
23         product.setHash(Integer.toString(product.getName().hashCode()));
24
25         jedis.sadd(REDIS_PRODUCTDATA, product.getHash());
26         jedis.sadd(product.getCategory(), product.getHash());
27         jedis.hset(product.getHash(), product.toMap());
28
29         jedis.zadd(SORTED_KEY, product.getPrice(), product.getHash());
30     }
31
32     @RequestMapping(value = "/product", method = RequestMethod.GET, params = { "name" })
33     public Product product(String name) {
34         return Product.fromMap(jedis.hgetAll(Integer.toString(name.hashCode())));
35     }
36
37     @RequestMapping(value = "/products", method = RequestMethod.GET)
38     public List<Product> products() {
39         List<Product> products = new ArrayList<>();
40
41         Set<String> hashes = jedis.smembers(REDIS_PRODUCTDATA);
42
43         for (String hash : hashes) {
```

```
44         Product product = Product.fromMap(jedis.hgetAll(hash));
45         products.add(product);
46     }
47
48     return products;
49 }
50
51 @RequestMapping(value = "/category", method = RequestMethod.GET, params = { "category" })
52 public List<Product> category(String category) {
53     List<Product> products = new ArrayList<>();
54
55     Set<String> hashes = jedis.smembers(category);
56
57     for (String hash : hashes) {
58         Product product = Product.fromMap(jedis.hgetAll(hash));
59         products.add(product);
60     }
61
62     return products;
63 }
64
65 @RequestMapping(value = "/cheapest", method = RequestMethod.GET)
66 public List<Product> filesPrio() {
67     List<Product> products = new ArrayList<>();
68
69     Set<String> hashes = jedis.zrange(SORTED_KEY, 0, 4);
70
71     for (String hash : hashes) {
72         Product product = Product.fromMap(jedis.hgetAll(hash));
73         products.add(product);
74     }
75
76     return products;
77 }
78
79 }
```
