FH MÜNSTER
University of Applied Sciences

Testergebnisse

F&E-Projekt

Victor Corbet (Matr-Nr. 1066911)

13.12.2023

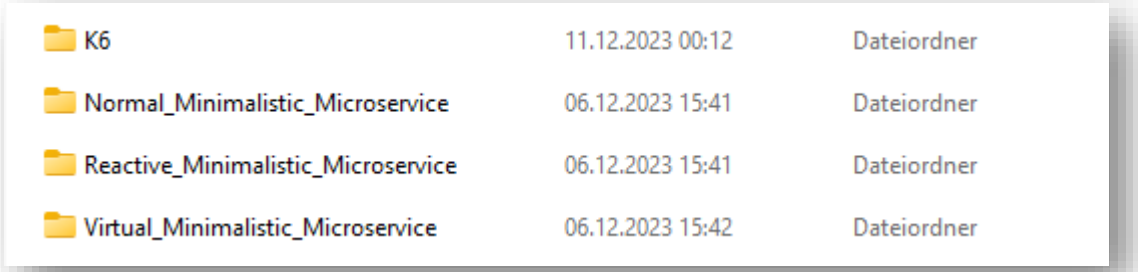
Inhalte

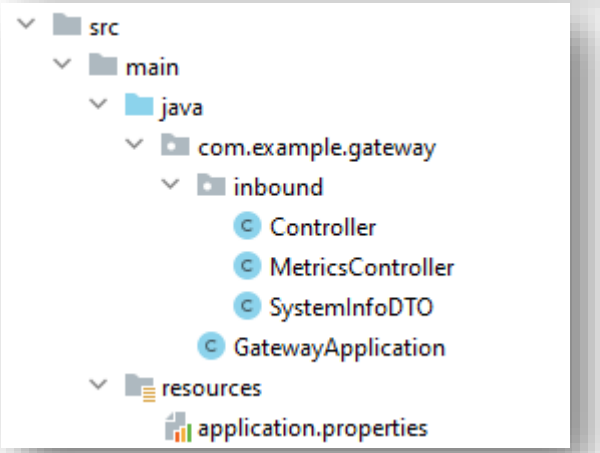
FH MÜNSTER
University of Applied Sciences

- Einleitung
  - Aufbau
  - Beispiel Reactive
  - Identisches Setup
- Testing
  - Testautomatisierung mittels Batch-Datei
  - Container Ressourcen
  - Dynamisches K6 Test Setup

- Analyse
  - Grouping, Filtering, Mapping, Statistics
  - Visualisierung

Einleitung
Aufbau





Einleitung
Beispiel Reactive

FH MÜNSTER
University of Applied Sciences

```java
@RestController
@RequestMapping("/minimalistic")
public class Controller {


    @Autowired
    public Controller() {    }

    @GetMapping
    public Mono<String> get() {
        // Simulate an asynchronous operation with delay
        return Mono.delay(java.time.Duration.ofSeconds(4))
                .map(delay -> "Reactive - Minimalistic");
    }
}
```

```java
public class SystemInfoDTO {
    private double cpuUsage;
    private long totalMemory;
    private long freeMemory;
    private double percentOfCpuUsed;
    private double percentOfRamUsed;
    private String currentTime;
```

```xml
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

```java
@RestController
@RequestMapping("/custom-metrics")
public class MetricsController {

    @GetMapping("/system-info")
    public Mono<SystemInfoDTO> getSystemInfo() {
        return Mono.fromCallable(() -> {
            OperatingSystemMXBean osBean = (OperatingSystemMXBean) ManagementFactory.getOperatingSystemMXBean();
            double cpuUsage = osBean.getProcessCpuLoad() * 100;

            // Convert total memory and free memory to megabytes
            long totalMemoryInBytes = osBean.getTotalMemorySize();
            long freeMemoryInBytes = osBean.getFreeMemorySize();
            long totalMemoryInMB = totalMemoryInBytes / (1024 * 1024);  // Convert bytes to megabytes
            long freeMemoryInMB = freeMemoryInBytes / (1024 * 1024);    // Convert bytes to megabytes

            double percentOfCpuUsed = (cpuUsage / osBean.getAvailableProcessors());
            double percentOfRamUsed = (1 - ((double) freeMemoryInBytes / totalMemoryInBytes)) * 100;

            LocalDateTime currentTime = LocalDateTime.now();
            String formattedTime = currentTime.format(DateTimeFormatter.ofPattern("HH:mm:ss:SSS"));

            return new SystemInfoDTO(cpuUsage, totalMemoryInMB, freeMemoryInMB, percentOfCpuUsed, percentOfRamUsed, formattedTime);
        });
    }
}
```

```json
{
    "cpuUsage": 0.625,
    "totalMemory": 1024,
    "freeMemory": 748,
    "percentOfCpuUsed": 0.078125,
    "percentOfRamUsed": 26.873016357421875,
    "currentTime": "23:20:00:636"
}
```

Einleitung
Identisches Setup

FH MÜNSTER
University of Applied Sciences

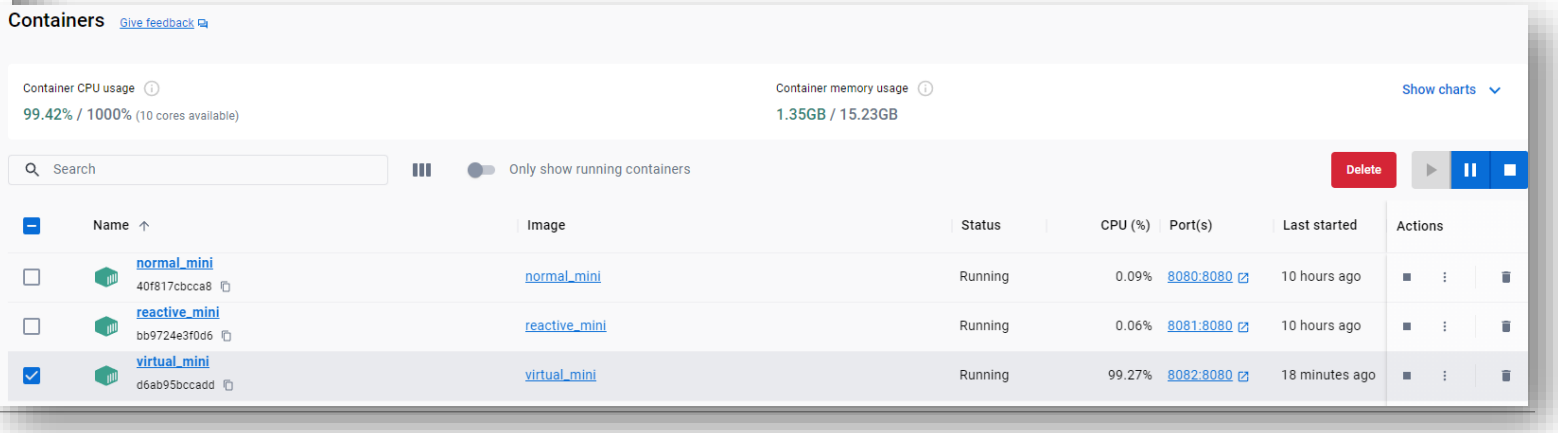
- Dockerfile für alle 3 -> Generieren eines Image „docker build -t reactive_mini .”
- Starten eines Containers mit 1 GB RAM und 8 Cores (CPU) “docker run -p 8081:8080 --cpus=8 --memory=1g --name reactive_mini reactive_mini”

```
# Verwenden Sie das offizielle OpenJDK-Image als Basis
FROM openjdk:22-ea-21-slim-bullseye
# Setzen Sie das Arbeitsverzeichnis im Container
WORKDIR /app

# Kopieren Sie die JAR-Datei in das Arbeitsverzeichnis
COPY target/gateway-0.0.1-SNAPSHOT.jar /app/app.jar

# Festlegen des Befehls zum Ausführen der Anwendung
CMD ["java", "-jar", "app.jar"]
```

```
C:\Users\User>docker images
REPOSITORY      TAG       IMAGE ID       CREATED         SIZE
virtual_mini    latest    6927a9406bcb   4 days ago      463MB
reactive_mini   latest    636cdc547fe1   4 days ago      465MB
```



Testing

Testautomatisierung mittels Batch-Datei

FH MÜNSTER
University of Applied Sciences

run.bat

```
@echo off
setlocal enabledelayedexpansion

REM Set the path to k6 executable (adjust this path based on your k6 installation)

REM Initialize iterations variable
set iterations=0

for /L %%V in (2400, 100, 3000) do (
    REM Increment the iterations counter
    set /a iterations+=1

    REM Update the config.json with the new "iterations" value
    echo { "info": "Port: Normal=8080 ; Reactive=8081 ; Virtual=8082", "port": "8082", "vus": %%V, "iterations": !iterations! } > config.json

    REM Start fetching RAM and CPU
    start node monitor.js

    REM Run k6
    k6 run test.js

    REM Stop fetching RAM and CPU
    taskkill /F /IM node.exe

    REM Sleep
    timeout /t 3
)

REM Analysis
cd .\output\analysis\
python analysis.py

REM Sleep
timeout /t 3

python visualize.py

REM Sleep
timeout /t 100

endlocal
exit /b 0
```

config.json

```
{
  "info": "Port: Normal=8080 ; Reactive=8081 ; Virtual=8082",
  "port": "8082",
  "vus": 2500,
  "iterations": 2
}
```

Testing
Container Ressourcen

FH MÜNSTER
University of Applied Sciences

Monitor.js

```
const config = require('./config.json');
const port = config.port;
const apiUrl = `http://localhost:${port}/custom-metrics/system-info`;

// Read vus value from config.json
const vus = config.vus;

// Construct the CSV file path with vus as a suffix
const csvFilePath = `./output/ressources/metric_output_${vus}.csv`;

function sendMonitorRequest() {
  http.get(apiUrl, (response) => {
    let data = '';

    // A chunk of data has been received.
    response.on('data', (chunk) => {
      data += chunk;
    });

    // The whole response has been received.
    response.on('end', () => {
      try {
        const jsonData = JSON.parse(data);
        saveToCsv(jsonData);
      } catch (error) {
        console.error('Error parsing JSON:', error.message);
      }
    });
  }).on('error', (error) => {
    console.error('Error making request:', error.message);
  });
}

function saveToCsv(data) {
  const timestamp = new Date().toISOString();
  const csvRow = `${timestamp},${data.cpuUsage},${data.totalMemory},${

  fs.appendFile(csvFilePath, csvRow, (err) => {
    if (err) {
      console.error('Error writing to CSV:', err.message);
    } else {
      console.log('Data saved to CSV successfully.');
    }
  });
}

// Start monitoring every second.
setInterval(sendMonitorRequest, 1000);
```

- Es wird jede Sekunde ein HTTP get an den Actuator-Endpoint geschickt
- Es wird das ergebnis als eine Zeile in einer CSV-Datei geschrieben
- (es wird node.js benötig)

Ergebnis Custom-Actuator-Endpoint


```
{
  "cpuUsage": 0.625,
  "totalMemory": 1024,
  "freeMemory": 748,
  "percentOfCpuUsed": 0.078125,
  "percentOfRamUsed": 26.873016357421875,
  "currentTime": "23:20:00:636"
}
```


output


```
2023-12-10T17:41:43.173Z,0.0992063492063492,1024,865,0.01240079365079365,15.497970581054688,17:41:43:171
2023-12-10T17:41:44.167Z,0,1024,865,0,15.498733520507812,17:41:44:166
2023-12-10T17:41:45.168Z,0.625,1024,865,0.078125,15.498733520507812,17:41:45:167
2023-12-10T17:41:46.182Z,0.4166666666666667,1024,865,0.052083333333333336,15.498733520507812,17:41:46:181
2023-12-10T17:41:47.194Z,0.5,1024,865,0.0625,15.498733520507812,17:41:47:193
2023-12-10T17:41:48.206Z,0.1388888888888889,1024,865,0.017361111111111112,15.526199340820312,17:41:48:205
2023-12-10T17:41:49.221Z,0.375,1024,865,0.046875,15.523147583007812,17:41:49:220
2023-12-10T17:41:50.232Z,0.375,1024,863,0.046875,15.708160400390625,17:41:50:231
2023-12-10T17:41:51.233Z,0.375,1024,863,0.046875,15.720367431640625,17:41:51:231
2023-12-10T17:41:52.234Z,0.45454545454545453,1024,862,0.056818181818181816,15.75164794921875,17:41:52:232
```

Testing
Dynamisches K6 Test Setup

FH MÜNSTER
University of Applied Sciences

```
const configFile = 'config.json';
const configData = JSON.parse(open(configFile));
const port = parseInt(configData.port); // Use the "port" value from the JSON
const vus = parseInt(configData.vus); // Use the "vus" value from the JSON

const csvFile = 'results.csv';

export const options = {
    stages: [
        { duration: '5s', target: 0 },
        { duration: '20s', target: vus }, // Use the "vus" value here
        { duration: '20s', target: vus }, // Use the "vus" value here
        { duration: '5s', target: 0 },
    ],
};

export default function () {

    const startTime = new Date().getTime();
    const response = http.get(`http://localhost:${port}/minimalistic`);
    const endTime = new Date().getTime();
    const responseTime = endTime - startTime;

    // Log the response time
    console.log(`[localhost:${port}] Response time: ${responseTime} ms`);



    // Introduce a sleep to simulate user think time
    sleep(1);
}

export function handleSummary(data) {
    // Return the default data object
    const my_string = "C:\\Users\\User\\fep\\minimalistic_gateway\\K6\\output/k6/summary_" + vus + ".json";
    const result = {};
    result[my_string] = JSON.stringify(data);
    return result;
}
```

Analyse

Grouping, Filtering, Mapping, Statistics

FH MÜNSTER
University of Applied Sciences

- Die Resultate der K6-Tests analysieren und wesentliche Kennzahlen extrahieren.

```python
def read_json(file_path):
    with open(file_path, 'r') as file:
        data = json.load(file)

    # Extract relevant values from the JSON
    avg_duration = data['metrics']['http_req_duration']['values']['avg']
    avg_waiting = data['metrics']['http_req_waiting']['values']['avg']
    error_rate = data['metrics']['http_req_failed']['values']['rate']

    # Create a DataFrame
    df = pd.DataFrame({
        'avg_duration': [avg_duration],
        'avg_waiting': [avg_waiting],
        'error_rate': [error_rate]
    })
```

K6-Summary-file (test.js)


```json
{
  "metrics": {
    "http_req_duration{expected_response:true}": {
      "contains": "time",
      "values": {
        "min": 4001.4211,
        "med": 4002.6477,
        "max": 4039.8309,
        "p(90)": 4003.69048,
        "p(95)": 4003.99983,
        "avg": 4002.9387010014284
      },
      "type": "trend"
    },
    "http_req_waiting": {
      "type": "trend",
      "contains": "time",
      "values": {
        "p(95)": 4003.96511,
        "avg": 4002.9158612303277,
        "min": 4001.4211,
        "med": 4002.6406,
        "max": 4039.2992,
        "p(90)": 4003.6475
      }
    },
    "data_received": {…
```

Analyse

Grouping, Filtering, Mapping, Statistics …

FH MÜNSTER
University of Applied Sciences

- Ergebnisse der Ressourcen-Abfragen auslesen und relevante werte extrahieren

```python
def read_csv(file_path):
    # Read CSV into a DataFrame
    df_csv = pd.read_csv(file_path, header=None, names=['timestamp', 'cpuUsage', 'total

    # Calculate the average for relevant columns
    avg_cpuUsage = df_csv['cpuUsage'].mean()
    avg_totalMemory = df_csv['totalMemory'].mean()
    avg_freeMemory = df_csv['freeMemory'].mean()
    avg_percentOfCpuUsed = df_csv['percentOfCpuUsed'].mean()
    avg_percentOfRamUsed = df_csv['percentOfRamUsed'].mean() / 100

    # Create a DataFrame with average values
    df_avg = pd.DataFrame({
        'avg_cpuUsage': [avg_cpuUsage],
        'avg_totalMemory': [avg_totalMemory],
        'avg_freeMemory': [avg_freeMemory],
        'avg_percentOfCpuUsed': [avg_percentOfCpuUsed],
        'avg_percentOfRamUsed': [avg_percentOfRamUsed]
    })

    return df_avg
```

```
7:41:43.173Z,0.0992063492063492,1024,865,0.01240079365079365,15.497970581054688,17:41:43:171
7:41:44.167Z,0,1024,865,0,15.498733520507812,17:41:44:166
7:41:45.168Z,0.625,1024,865,0.078125,15.498733520507812,17:41:45:167
7:41:46.182Z,0.4166666666666667,1024,865,0.052083333333333336,15.498733520507812,17:41:46:181
7:41:47.194Z,0.5,1024,865,0.0625,15.498733520507812,17:41:47:193
7:41:48.206Z,0.1388888888888889,1024,865,0.017361111111111112,15.526199340820312,17:41:48:205
2023-12-10T17:41:49.221Z,0.375,1024,865,0.046875,15.523147583007812,17:41:49:220
2023-12-10T17:41:50.232Z,0.375,1024,863,0.046875,15.708160400390625,17:41:50:231
2023-12-10T17:41:51.233Z,0.375,1024,863,0.046875,15.720367431640625,17:41:51:231
2023-12-10T17:41:52.234Z,0.4545454545454545,1024,862,0.056818181818181816,15.75164794921875,17:41:52:232
```

Analyse

Grouping, Filtering, Mapping, Statistics …

FH MÜNSTER
University of Applied Sciences

```python
def main():
    # Provide the path to the config.json file
    config_file_path = os.path.abspath('../../config.json')

    # Read values from config.json
    with open(config_file_path, 'r') as config_file:
        config_data = json.load(config_file)

    # Extract values from config.json
    end_vus = config_data['vus']
    start_vus = end_vus - ((config_data['iterations'] - 1) * 100)

    # Create an empty DataFrame to store the results
    final_df = pd.DataFrame()

    for vus in range(start_vus, end_vus + 1, 100):
        # Construct the paths based on the loop variable
        json_file_path = os.path.abspath(f'./../k6/summary_{vus}.json')
        csv_file_path = os.path.abspath(f'./../ressources/metric_output_{vus}.csv')

        # Read JSON and create DataFrame
        df_json = read_json(json_file_path)

        # Read CSV, calculate average, and create DataFrame
        df_avg = read_csv(csv_file_path)

        # Merge DataFrames on a common column (e.g., timestamp)
        merged_df = pd.merge(df_json, df_avg, left_index=True, right_index=True)

        # Set the index
        merged_df.index = [str(vus)]

        # Add the row to the final DataFrame
        final_df = pd.concat([final_df, merged_df])

    # Save the final DataFrame to a CSV file in the current folder
    output_csv_path = os.path.abspath('./final_metrics.csv')
    final_df.to_csv(output_csv_path, index=True, header=True)

    # Display the final DataFrame
    print(final_df)

    # Display the final DataFrame
    print(final_df)
```

- Für jeden Testdurchlauf (z.B. vus=100) wird eine Test-Datei (summary_100.json) und eine Ressourcen-Datei (metric_output_100.csv) angelegt
- Hier werden für beide bestimmte Operationen durchgefürt (z.B. Mittelwerte für CPU und RAM verbrauch)
- Zum Schluss: „Merging“ zu einer neuen Spalte in einem neunen Output-File (final_metrics.csv).
- Index/Schlüssel (links) sind die Anzahl vus

```
,avg_duration,avg_waiting,error_rate,avg_cpuUsage,avg_totalMemory,avg_freeMemory,avg_percentOfCpuUsed,avg_percentOfRamUsed
100,4002.9387010014284,4002.9158612303277,0,0.7126255945700392,1024.0,851.6111111111111,0.08907819932125488,0.16786977979871964
200,4002.0377112464244,4001.997961819489,0,0.7329044412377747,1024.0,824.5370370370371,0.09161305515472182,0.1943056318495009
300,4001.928797709928,4001.904204580157,0,0.7423991903158568,1024.0,806.1111111111111,0.0927998987894821,0.21233346727159286
```

Analyse
Visualisierung

FH MÜNSTER
University of Applied Sciences

```python
def read_and_visualize_csv(csv_file_path):
    # Read CSV file into a DataFrame
    df = pd.read_csv(csv_file_path, index_col=0)

    # Extract relevant columns for visualization
    columns_to_visualize = ['avg_duration', 'error_rate', 'avg_cpuUsage', '

    # Set up the figure and axes
    fig, ax1 = plt.subplots(figsize=(12, 8))  # Adjust the figure size as n

    # Set the width of the bars
    bar_width = 6  # Adjust the width of the bars for all columns

    # Plot bars for avg_duration on the left axis with a wider bar
    color = 'tab:blue'
    ax1.bar(df.index, df['avg_duration'], width=bar_width, color=color, lab
    ax1.set_xlabel('Virtual Users')
    ax1.set_ylabel('Average Duration (ms)', color=color)
    ax1.tick_params(axis='y', labelcolor=color)

    # Create a second y-axis for error_rate, avg_percentOfCpuUsed, and avg_
    ax2 = ax1.twinx()

    # Plot error_rate on the right axis
    color = 'tab:red'
    ax2.plot(df.index, df['error_rate'] * 100, color=color, linestyle='dash
    ax2.tick_params(axis='y', labelcolor=color)

    # Plot avg_percentOfCpuUsed on the right axis
    color = 'tab:orange'
    ax2.plot(df.index, df['avg_percentOfCpuUsed'] * 100, color=color, lines
    ax2.tick_params(axis='y', labelcolor=color)

    # Plot avg_percentOfRamUsed on the right axis
    color = 'tab:green'
    ax2.set_ylabel('Prozent', color=color)
    ax2.plot(df.index, df['avg_percentOfRamUsed'] * 100, color=color, lines
    ax2.tick_params(axis='y', labelcolor=color)

    # Set labels and title
    plt.title('Performance Metrics for Different Virtual Users')

    # Move the legend outside the chart
    ax1.legend(loc='upper left', bbox_to_anchor=(1, 1))
    ax2.legend(loc='upper left', bbox_to_anchor=(1, 0.9))

    # Show the chart
    plt.show()
```