TU DRESDEN

Service and Cloud Computing

Praktikum

TimeTracker

Eine Microservice-Anwendung für das bessere Zeitmanagement

Autoren:
Sinthujan Thanabalasingam
Wieland Strauss

Dozentin:
Dr.-Ing. Iris Braun

20. Januar 2019

Inhaltsverzeichnis

1 Über TimeTracker

1.1 Inhalt der Anwendung

TimeTracker ist eine Anwendung um die Zeit im Überblick zu behalten. Wenn der Nutzer registriert ist, kann er sich jederzeit auf den Server einloggen und die Aufnahme einer Aktivität starten. Nach beenden seiner Aufgabe wir die Aufnahme gestoppt.

Dem Benutzer wird eine Übersicht über seine aufgenommene Zeit angeboten. Dadurch bekommt er einen besseren Überblick über seine aufgebrachte Zeit und kann sich in Zukunft besser einteilen.

Zusätzlich bietet die Plattform eine globale Statistik über alle Benutzer.

1.2 Einordnung der Arbeit

Die Anwendung wurde als Prüfungsvorleistung im Rahmen einer praktischen Übung der Lehrveranstaltung *Service and Cloud Computing* des Lehrstuhls Rechnernetze der TU Dresden entwickelt. Dozentin und Betreuerin ist Frau Dr.-Ing. Iris Braun.

2 Arbeitsablauf

2.1 Team

Das Entwicklerteam besteht aus den Studenten Sinthujanan Thanabalasingam (Master Informatik, 4. Semester)und Wieland Strauß (Diplom Informatik, 7. Semester).

2.2 Vorgehensweise

Nach der Ideenfindung wurde zuerst die Architektur entworfen. Die Entwicklung wurde nahezu vollständig nach dem Prinzip des TDD (Test Driven Development) betrieben. Die API wurde parallel mit *Swagger* erstellt. Auch die Verwendung von Docker und die Entwicklung des Frontends wurden zeitgleich umgesetzt. Zuletzt wurde die Anwendung auf einen Server von DigitalOcean ausgerollt

Der Arbeitsablauf organisierte sich aus mehreren Treffen. Bei diesem wurden die Planung durchgeführt, der Großteil der Anwendung geschrieben und Aufgaben für die selbstständige Durchführung festgelegt. Aufgrund der verschiedenen Expertise wurde zum Teil Pair Programming durchgeführt.

3 Technologien

3.1 Architektur

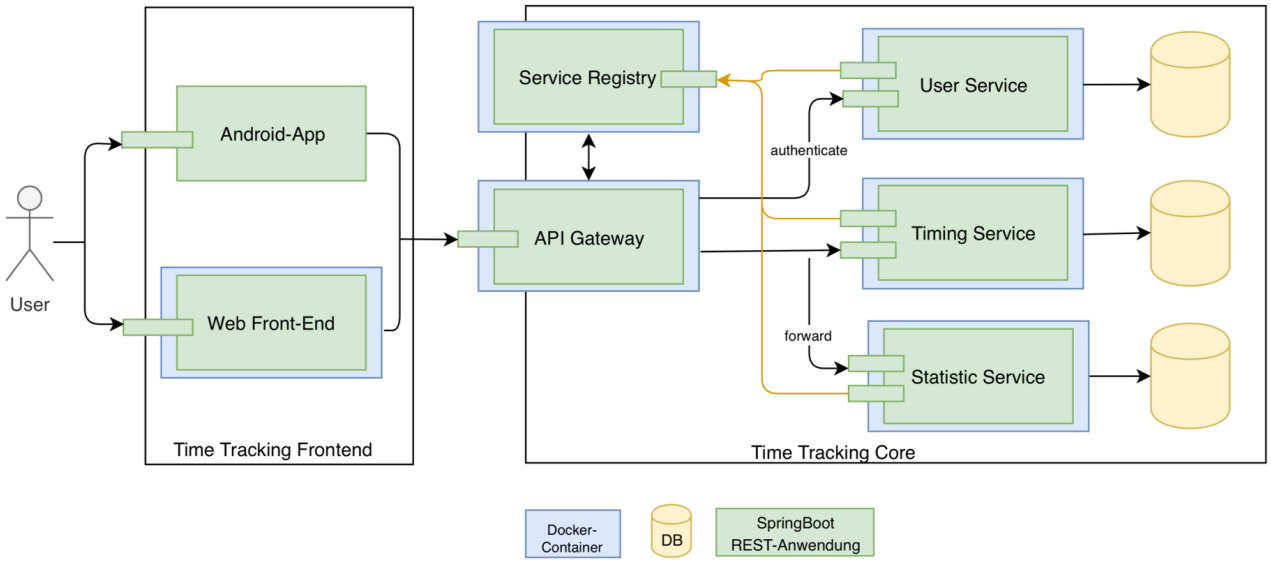

Abbildung 1: Architektur von TimeTracker

Anmerkung: Die Andoid-App wurde nicht im Rahmen der Lehrveranstaltung fertig gestellt und ist somit noch ausstehend.

3.2 Übersicht verwendeter Technologien

Tabelle 1: Technologien

Name	**Version**	**Verwendung**
Java	JDK 1.8.0	Verwendete Programmiersprache
Mavent	3.1.0	Build-Management-Tool
Spring boot	2.0.2	Java Framework für Web-Systeme
MySQL	5.7.0	Relationale Datenbank
Netflix Zuul	1.3.1	Edge Service für dynamisches Routen, Monitoring und Sicherheit
Netflix Eureka	1.9.2	REST-basierter Service, Zuordnen von Services
JSON	-	Dateiformat bei Übermittlung
REST	-	Programmierparadigma für Webservices
React	16.1.1	JavaScript Bibliothek für User Interfaces
single-spa	2.6.0	JavaScript Framework für Front-End Microservices
OpenAPI (Swagger)	3.0.	REST-Schnittstelle basiert auf Standard von OpenAPI, Zum Entwerfen und Dokumentieren der RESTfull-Schnittellen wurde das Framework Swagger verwendet
Postman	6.5.3	API Development Environment
Docker	18.09.0	Umgebung für Container

3.3 Schnittstellenbeschreibung

Die API wurde nach dem OpenAPI-Standard entwickelt und generierte automatisch ein Controller in Java. Eine Auflistung aller Methoden ist über die folgenden Links zu erreichen. Zusätzlich befindet sich im Anhang (Seite 7) ein Ausdruck der automatisch generierten Dokumentation.

- frontend-service:
- timing-service:
- user-service:

3.4 Sicherheit

Die Sicherheit wurde mittels **JSON Web Token (JWT)** und umgesetzt. Dabei erhält der Benutzer beim Login eine UUID, mittels derer er sich gegenüber den Diensten authentifizieren kann.

Des weiteren wird die Sicherheit der Verbindung mittels der Transportverschlüsselung **HTTPS** gewährleistet.

4 Bedienungsanleitung des Clients

Bei lokaler Installation: localhost:9123
TimeTracker bietet folgende Funktionen:

- Time Tracking
- Anlegen von Aktivitäten
- Tracking für verschiedene Aktivitäten
- Abruf von Statistiken (Benutzer/Global)

4.1 Fehlerbehandlung

Bei Fehlern im Root-Verzeichnisses des Projekts folgendes ausführen:

```
mvn clean install
docker-compose -f docker/docker-compose.yml up -d
```

5 Feedback und Kritik am Praktikum

Allgemein: Es machte uns sehr viel Spaß diese Anwendung zu entwickeln. Ein gutes Klima im Team trug dazu bei. Aufgrund des Unterschiedes in den Bereichen der praktischen Erfahrung konnte bei uns auch gut Wissen ausgetauscht werden, beziehungsweise entwickelte sich mit der Zeit eine Art Mentoring. Bei der Umsetzung wurde sehr deutlich, wie das Praktikum die Vorlesungsinhalte vertieft und im direkten Bezug dazu tritt. Das Wissen wurde dadurch anschaulicher und praktisch Umgesetzt.

Spezifikation der Anforderungen: Durch die in manchen Teilen unspezifische Aufgabenstellung lässt sich schwer Abschätzen, was es genau zu erreichen gilt. Positiv daran ist die Möglichkeit der freien Auswahl der Vorgehensweise, Umsetzung und Tools.

6 Anhang

- Generierte API-Dokumentation User Service
- Generierte API-Dokumentation Timing Service
- Generierte API-Dokumentation Frontend Service

User Service

No description provided (generated by Swagger Codegen https://github.com/swagger-api/swagger-codegen)
More information: https://helloreverb.com
Contact Info: hello@helloreverb.com
Version: 0.0.1
BasePath:/

http://apache.org/licenses/LICENSE-2.0.html

Access

1.

Methods

[Jump to Models]

Table of Contents

Default

GET /users/{uuid}

Up

(**getUserName**)

Path parameters

uuid (required)
Path Parameter —

Return type
UserData

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK UserData

POST /users/info

Up

(**getUserNamesFromUuidList**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

body **UuidData** **(required)**
Body Parameter —

Return type
array[UserData]

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK

GET /users/{name}/uuid

Up

(**getUserUuid**)

Path parameters

name **(required)**
Path Parameter —

Return type
UuidData

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK UuidData

POST /signup

Up

(**signup**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

body [UserData](#) (required)
Body Parameter —

Return type
[RegistrationStatus](#)

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK [RegistrationStatus](#)

Models

[Jump to Methods]

Table of Contents

`RegistrationStatus` Up

`UserData` Up

Data that is transferred to register a new or update an existing user.

username (optional)
String The user name.

password (optional)
String The hashed password of the user

role (optional)
String

`UuidData` Up

uuids (optional)
array[String]

operationResponse

Up

error (optional)
String Used to indicate error messages.

data (optional)
array[Object]

Timing Service

No description provided (generated by Swagger Codegen https://github.com/swagger-api/swagger-codegen)
More information: https://helloreverb.com
Contact Info: hello@helloreverb.com
Version: 0.0.1
BasePath:/

http://apache.org/licenses/LICENSE-2.0.html

Access

1.

Methods

[Jump to Models]

Table of Contents

Default

POST /activities/{activityId}/records

Up

(addActivityRecord)

Triggers start or end of a record for a given activityID

Path parameters

activityId (required)
Path Parameter —

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK operationResponse

POST /activities/

Up

(**createActivity**)

Consumes
This API call consumes the following media types via the Content-Type request header:

- `application/json`

Request body

body activity (required)
Body Parameter —

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

DELETE /activities/{activityId}

Up

(**deleteAcitity**)

Removes an activity with given id and all its associated records from the database.

Path parameters

activityId (required)
Path Parameter —

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

GET /activities/

Up

(**getActivities**)

Returns all activities

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

`GET /activities/{activityId}` Up

(**getActivity**)

Returns activity with given id

Path parameters

activityId (required)
Path Parameter —

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

`GET /activities/{activityId}/records` Up

(**getActivityRecords**)

Returns all records for an activity

Path parameters

activityId (required)
Path Parameter —

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200

OK operationResponse

GET /activities/stats

Up

(**getGlobalActivityStats**)

Returns statistics for this application

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

GET /activities/records

Up

(**getRecords**)

Returns all records for a user

Query parameters

tag (optional)
Query Parameter — The tag filter for user records

activityUuid (optional)
Query Parameter — The uuid of the activity of interest

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

PUT /activities/{activityId}

Up

(**updateActivity**)

Updates the activity meta data.

Path parameters

activityId (required)
Path Parameter —

Return type
operationResponse

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- `application/json`

Responses
200
OK operationResponse

Models

[Jump to Methods]

Table of Contents

activity Up

An activity that can be tracked

name (optional)
String The name of the activity

tag (optional)
String The tag of the activity

uuid (optional)
String The uuid of the activity

description (optional)
String The description of the activity

owneruuid (optional)
String The uuid of the user that owns this activity

activityRecord Up

A record of an activity that a user has submitted

activityuuid (optional)
String The uuid of the activity.

activityName (optional)
String The name of the activity.

time (optional)
Date The point in time a record update was issued (e.g. the request was sent to the server) format: date-time

startTime (optional)
Date The point in time this record was created format: date-time

endTime (optional)
Date The point in time this record ended format: date-time

uuid (optional)
String The uuid of this record

duration (optional)
Integer The duration of the record format: int32

tag (optional)
String The tag of the activity

state (optional)
String
Enum:
STARTED
ENDED

operationResponse

Up

error (optional)
String Used to indicate error messages. Null if no error occured

data (optional)
array[Object] Attached data of response

Platform Frontend Service

No description provided (generated by Swagger Codegen https://github.com/swagger-api/swagger-codegen)
More information: https://helloreverb.com
Contact Info: hello@helloreverb.com
Version: 0.1-oas3
BasePath:/

http://apache.org/licenses/LICENSE-2.0.html

Access

Methods

[Jump to Models]

Table of Contents

Default

GET /uiservices

Up

(**getUiServices**)

Return type
UiServices

Produces
This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses
200
OK UiServices

Models

[Jump to Methods]

Table of Contents

UiService

Up

Services with name and url.

serviceId (optional)
String

serviceName (optional)
String

serviceAddress (optional)
String

UiServices

Up

services (optional)
array[UiService]