



universität
wien

MICROSERVICE PLACES STATIC

Verfasser

Antonio Gasparevic (01506462), Nemanja Srdanovic (01576891),

Kevin Çobaj (01635087)

Wien, 2021

11.06.2021

Studienkennzahl lt. Studienblatt:

UA 033 526

Fachrichtung:

Informatik - Wirtschaftsinformatik

Betreuerin / Betreuer:

o. Univ.-Prof. Dr. techn. Dimitris Karagiannis; Dipl. Ing.,
Mag. Dr. Wilfrid Utz

Abstract

Wir haben es uns zum Ziel gesetzt eine API zu entwickeln die mit dem Microservice Framework OLIVE auf dem DKE Server für alle Mitstudierenden und Kollegen vom DKE zur Verfügung gestellt werden. Der Microservice soll einen Ort als Input bekommen und die Kategorien der in der Umgebung liegenden Orte zurückliefern. Danach kann nach der Kategorie gesucht werden, die dann die genauen Orte ausgibt und dem User bereitstellt. Außerdem kann der User eine Abfrage senden, die ihm die Unterkategorien der Hauptkategorie zurückliefert. Ein Endpoint für Abfragestatistiken ist ebenfalls vorhanden.

Inhaltsverzeichnis

1	Einleitung.....	4
2	Potenzielle Anwendungen	4
2.1	Touristen-App	4
2.2	Gastronomie-App.....	4
2.3	Statistik-App.....	4
2.4	Taxi-App.....	5
2.5	Freizeitpark-App.....	5
3	Potenzielle Kombinationen mit andere Microservices	5
3.1	People (Gruppe 10)	5
3.2	Geolocation Service (Gruppe 00)	5
3.3	Charging Infrastructure Service (Gruppe 01)	5
3.4	Weather (Gruppe 03).....	6
4	Realisierung der Services	6
4.1	places/byAdress/{streetName}/{houseNumber}/{radius}	7
4.2	places/byCategory/{category}/{searchIdentifier}	7
4.3	places/getSubcategories/{mainCategory}/{searchIdentifier}	8
4.4	places/searchStatistic/{sortingOrder}.....	8
5	Frontend Beschreibung und Illustration	10
5.1	getContentFromAdress() Methode	11
5.2	getElementsOfCategory() Methode	12
6	Wie starte ich das Frontend lokal auf meinem Computer?	12
7	Arbeitsaufteilung	14
8	Conclusio	14
9	Literaturverzeichnis	15
10	Abbildungsverzeichnis	15

1 Einleitung

Mit einer exponentiell wachsenden Smartphone-Technologie werden viele Dienste angeboten, an die vor 10 Jahren noch nicht einmal gedacht wurde. Viele dieser Dienste enthalten auch standortbezogene Optionen, z. B. die Freigabe Ihres aktuellen Standorts auf WhatsApp. Unser Microservice sollte „Points of Interest“ basierend auf Ihrer aktuellen Adresse und einer ausgewählten Kategorie zurückgeben. Das Potenzial für eine Kombination mit anderen Diensten wie Geolokalisierung oder Wettervorhersage ist enorm. Dieser Service kann auch als Plugin für verschiedene Anwendungen angeboten werden.

2 Potenzielle Anwendungen

2.1 Touristen-App

Dieser Service kann einer Touristen-App angeboten werden. Finden Sie je nach aktuellem Standort die für Touristen am wichtigsten benötigten Interessenpunkte. Ein Beispiel könnten Touristeninformationszentren, Sehenswürdigkeiten, öffentliche Verkehrsmittel sein.

2.2 Gastronomie-App

Dieser Service kann Restaurants und Bars angeboten werden. Ein aktuelles Beispiel wäre eine Person, die sich in der Innenstadt befindet und ein Gastronomieunternehmen besuchen möchte. Sie konnten sehen, wie viele Personen zu einem bestimmten Zeitpunkt auf jedes Gastronomieunternehmen geklickt haben oder wer das am häufigsten angeklickten Restaurant ist.

2.3 Statistik-App

Dieser Service kann der österreichischen Statistikbehörde angeboten werden. Zu jedem bestimmten Zeitpunkt konnten Sie sehen, welche Kategorien am häufigsten gesucht wurden. Beispiel: Ein Benutzer sucht zur Mittagszeit nach Restaurants, abends jedoch nach Bars.

2.4 Taxi-App

Dieser Service kann einer Taxi-App angeboten werden. Eine Kategorie im Service könnte darin bestehen, Taxis oder private Transportmittel wie Bolt in der Nähe zu finden. Dies kann auch hervorragend mit dem Geolocation Service kombiniert werden.

2.5 Freizeitpark-App

Dieser Service kann einem Spielpark angeboten werden. In großen Themenparks wie Praterpark (es gibt viele Parks, die viel größer sind) konnte der Benutzer seinen aktuellen Standort und die verschiedenen Spiele in seiner Nähe und im gesamten Park durch interaktives Zoomen sehen.

3 Potenzielle Kombinationen mit andere Microservices

3.1 People (Gruppe 10)

Andere Services könnten auf unser Service zugreifen, indem sie herausfinden, wieso die Menschenmasse gerade in diesem Bereich sich bewegen. Zuerst können wir nach der Adresse und dann nach der Kategorie wie z.B. Restaurants oder Bars filtern.

3.2 Geolocation Service (Gruppe 00)

Wir könnten ein Geolocation Microservice verwenden, um einen genaueren Standort festlegen zu können, mit Hilfe der erfragten Koordinaten und diese an den Benutzer weitersenden.

3.3 Charging Infrastructure Service (Gruppe 01)

Mit Enterprise Ressource Planung könnten wir nach E-Tankstellen kategorisieren, indem wir diese API verwenden und die Ergebnisse dem User ausgeben.

3.4 Weather (Gruppe 03)

Eine Kombination mit ein Wettervorhersage Service wäre auch möglich, wo der Service dem User zeigt, wie das Wetter in den angegebenen Standort sein wird. Es könnte sein, dass es am Standort eine Veranstaltung in einem offenen Bereich gibt, aber während der Veranstaltung wird es zum Regnen anfangen. Alert: Regenschirm mitbringen!

4 Realisierung der Services

Um unser Projekt zu realisieren, musste zuerst eine gewisse Planung stattfinden, in der wir besprochen haben, welche APIs wir nutzen werden. Hierbei haben sich die APIs Geoapify und Openstreetmap, als Favoriten herauskristallisiert. Diese beiden APIs werden hierfür benötigt, um Abfragen an diese zu senden und so einen Strom an Daten zu erlangen, den wir danach mittels unseres selbst erstellten Programms bearbeiten.

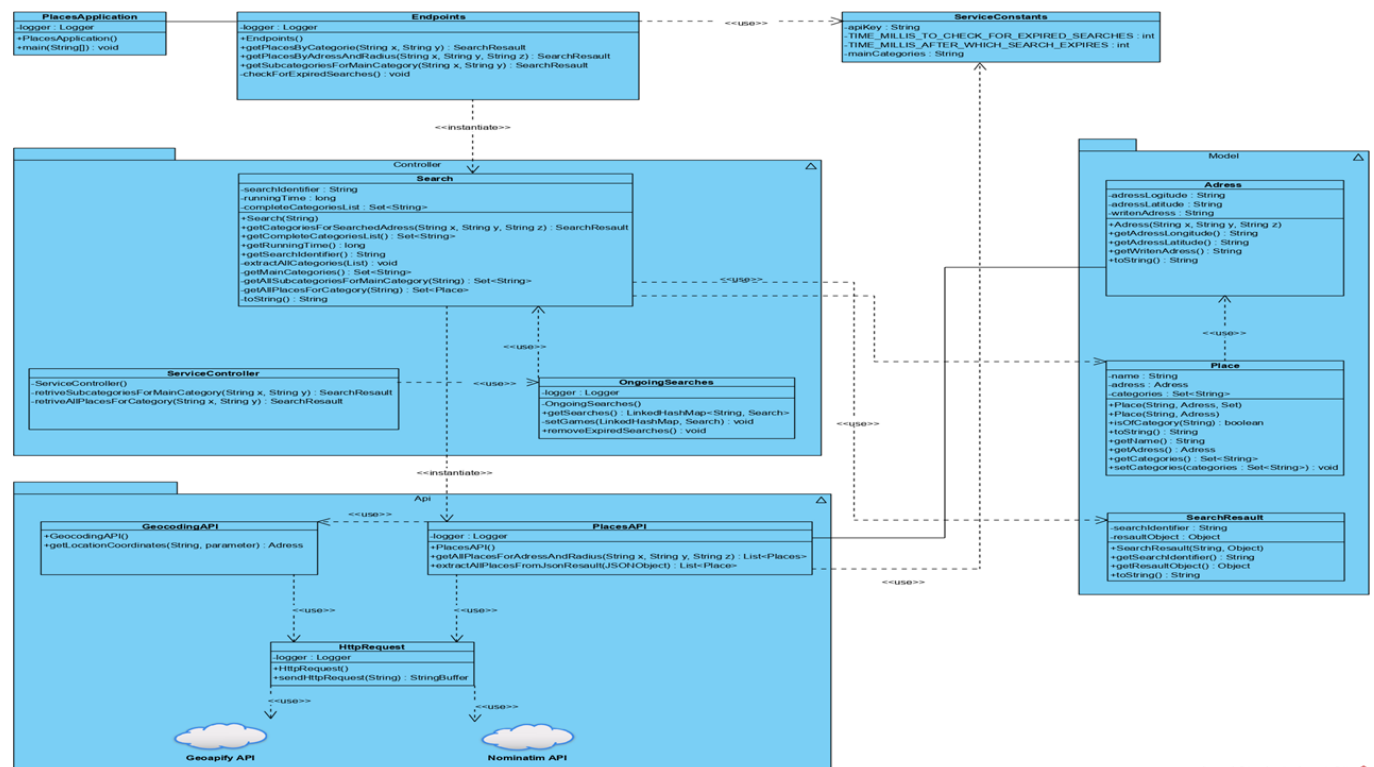


Abb. 1 UML-Places-Klassendiagramm

4.1 places/byAdress/{streetName}/{houseNumber}/{radius}

Der Nutzer gibt den Straßennamen, Hausnummer und den Radius in dem gesucht werden soll an. Dabei übernimmt der Microservice die angegebenen Variablen und sendet sie an die Openstreetmap API, welche die genauen Koordinaten zurücksendet. Danach werden die Koordinaten verarbeitet und an die Geoapify API weitergesendet, welche uns dann eine Liste an Daten generiert mit der wir unsere benötigten Informationen erhalten. Hierbei werden dann die Kategorien, der in der Umgebung befindlichen Orte bereitgestellt.

```
@CrossOrigin
@RequestMapping(value = "/byAdress/{streetName}/{houseNumber}/{radius}")
public SearchResult getPlacesByAddressAndRadius(@PathVariable String streetName, @PathVariable String houseNumber,
    @PathVariable String radius) throws IOException, JSONException {

    String searchIdentifier = UUID.randomUUID().toString().substring(0, 5);

    Search newSearch = new Search(searchIdentifier);

    OngoingSearches.getSearches().put(searchIdentifier, newSearch);

    Logger.info("New search created. ID : {}", searchIdentifier);

    return newSearch.getCategoriesForSearchedAddresses(streetName, houseNumber, radius);
}
```

Abb. 2 Endpoint Integration byAdress im Backend

4.2 places/byCategory/{category}/{searchIdentifier}

Der Benutzer sendet eine, der bereits erhaltenen Kategorien von der vorherigen API, Kategorie an diesen Endpoint mit den vorherigen Daten und erhält alle Orte, die dieser Kategorie entsprechen in einer Liste zurück.

```
@CrossOrigin
@RequestMapping(value = "/byCategory/{category}/{searchIdentifier}")
public SearchResult getPlacesByCategory(@PathVariable String category, @PathVariable String searchIdentifier) {

    SearchStatistic.addToCategorySearchStatistic(category);

    return ServiceController.retrieveAllPlacesForCategory(category, searchIdentifier);
}
```

Abb. 3 Endpoint Integration byCategorie im Backend

4.3 places/getSubcategories/{mainCategory}/{searchIdentifier}

Der Benutzer sendet eine, der bereits erhaltenen Kategorien vom ersten Endpoint, Kategorie an diesen Endpoint mit den vorherigen Daten und erhält alle Orte, die dieser Kategorie entsprechen in einer Liste zurück.

```
@CrossOrigin
@RequestMapping(value = "/getSubcategories/{mainCategory}/{searchIdentifier}")
public SearchResault getSubcategoriesForMainCategory(@PathVariable String mainCategory,
    @PathVariable String searchIdentifier) {
    return ServiceController.retrieveSubcategoriesForMainCategory(mainCategory, searchIdentifier);
}
```

Abb. 4 Endpoint Integration getSubcategories im Backend

4.4 places/searchStatistic/{sortingOrder}

Für die Benutzer könnte es interessant sein zu erfahren, welche Kategorien am öftesten gesucht worden sind. Hierfür wurde dieser Endpoint bereitgestellt, welcher hierbei noch einen Extra Ascending oder Descending Parameter benötigt, um die Liste auf- oder absteigend zu sortieren.

```
@CrossOrigin
@RequestMapping(value = "/searchStatistic/{sortingOrder}")
public LinkedHashMap<String, Integer> searchStatistic(@PathVariable String sortingOrder) {
    return SearchStatistic.getCategorySearches(sortingOrder);
}
```

Abb. 5 Endpoint Integration searchStatistic im Backend

Um die technische Realisierung des Microservices zu beschreiben wurde ein auskommentierter Code auf dem Git Repository bereitgestellt. (<https://gitlab.dke.univie.ac.at/eis/2021s/group9/-/tree/master/src/main/java/com/group9/places>)

Abstract

1 Architecture

2 Technology

3 Deployment

4 Microservice Places Static

Microservice: Places (static)

Unsere Operationen hier einfach zum Benutzen.

•
•
•
•
•
•

we are gonna go Places

byAddress



byCategory



bySubCategory



searchStatistic



Abb. 6 Microservices auf der Webseite

https://eis.dke.univie.ac.at/eis2021/details/?id=479&path=documentation/4_Microservice_Places_Static.md

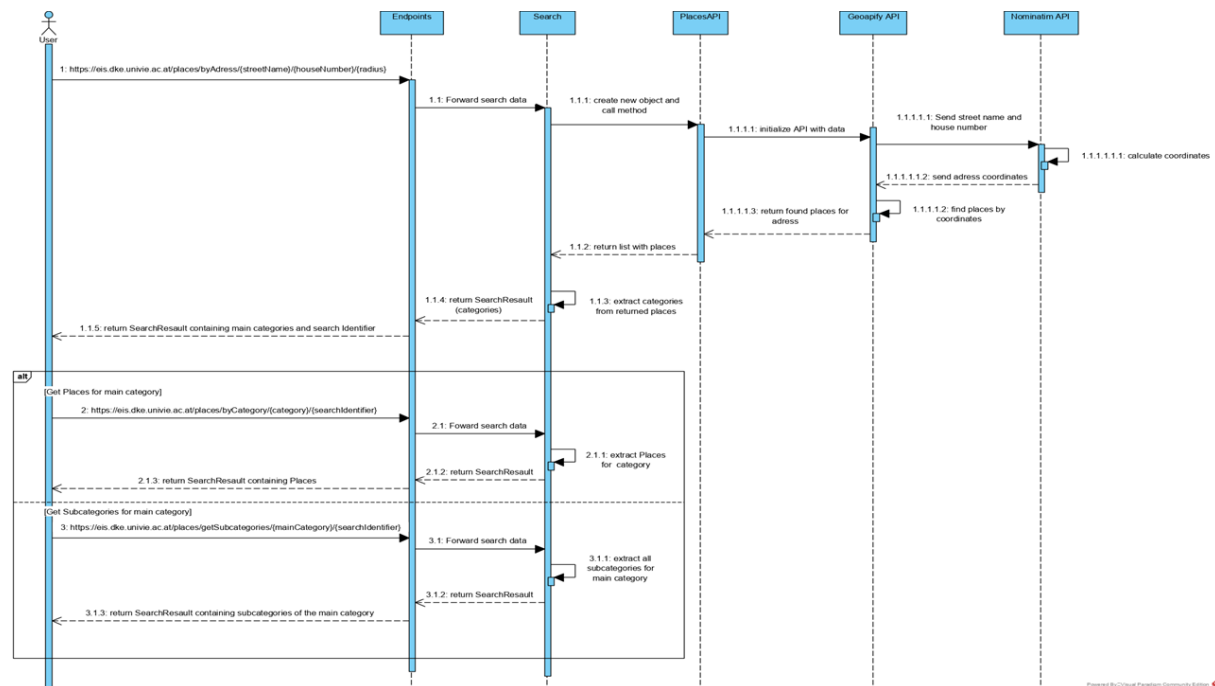
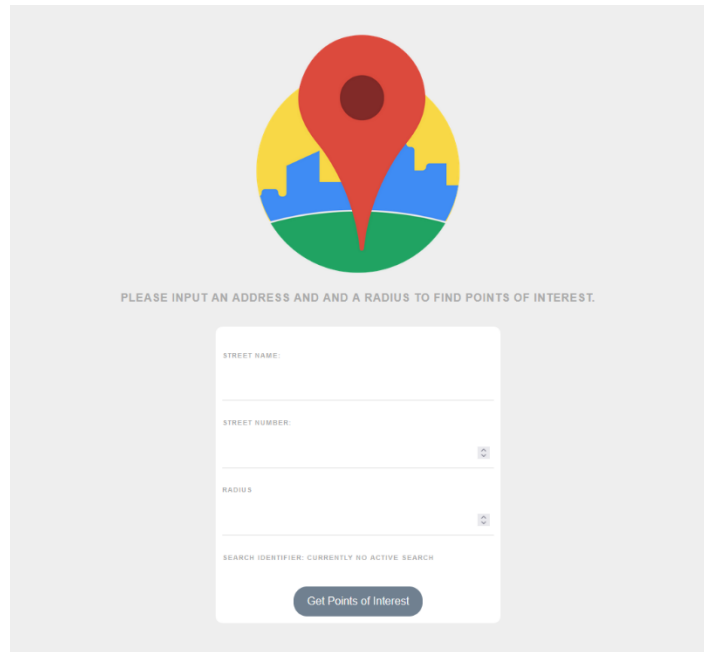


Abb. 7 UML-Places-Sequenzdiagramm

5 Frontend Beschreibung und Illustration

Wir haben versucht, dass das Frontend so simple wie möglich gehalten wird. Im Endeffekt sieht der User nur Felder, wo er 3 Inputs hinzufügen kann.

- Die Straße
- Die Gebäudenummer
- Radius



The image shows a web interface for finding points of interest. At the top, there is a large red location pin icon with a yellow and blue city skyline and a green field in the background. Below the icon, the text reads: "PLEASE INPUT AN ADDRESS AND AND A RADIUS TO FIND POINTS OF INTEREST." The input area contains three text boxes: "STREET NAME:", "STREET NUMBER:", and "RADIUS:". Each text box has a small "x" icon to its right. Below the "RADIUS:" box, there is a text label "SEARCH IDENTIFIER: CURRENTLY NO ACTIVE SEARCH". At the bottom of the input area is a button labeled "Get Points of Interest".

Abb. 8 Frontend Ansicht der Startseite auf localhost:8080

Nachdem der User den Button „Get Points of Interest“ klickt, kommuniziert das Frontend mit unserem Olive Microservice, die uns der DKE Server online zur Verfügung stellt.

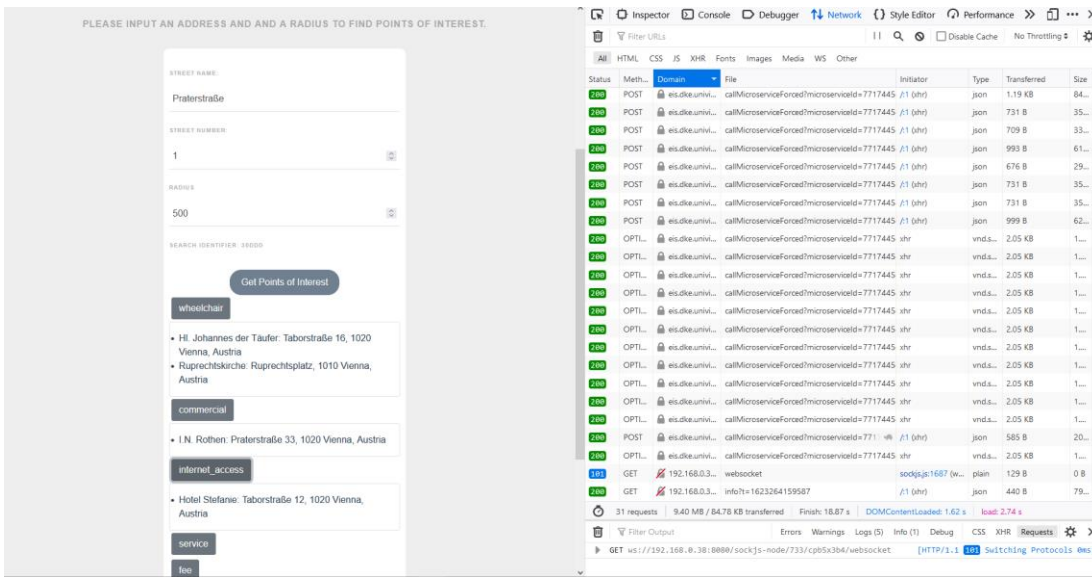


Abb. 9 Frontend Ansicht mit Network Entwicklerkonsole

Das Frontend wurde in Vue.js und JavaScript geschrieben und für die Drop Down Komponenten wurde Bootstrap verwendet.

Insgesamt gibt es 2 Methoden, die es möglich machen, Ergebnisse von der DKE Server zu bekommen.

5.1 getContentFromAdress() Methode

Diese Methode entspricht den Endpoint byAdress in unsere Dokumentation (https://eis.dke.univie.ac.at/eis2021/details/?id=479&path=documentation/4_Microservice_Places_Static.md). Ein SearchIdentifier und die Kategorien werden als JSON Objekte zurückgeliefert und die Informationen werden lokal gespeichert.

Endpoint on DKE Server, wo POST Request geschickt wird:

<https://eis.dke.univie.ac.at/micro-service-controller-rest/rest/msc/callMicroserviceForced?microserviceId=7717445d-958c-4957-ab73-05ff9b157740&operationId=byAdress>

5.2 getElementsOfCategory() Methode

Nachdem alle Kategorien lokal gespeichert sind, schickt der Frontend ein POST Request an den Microservice für jede Kategorie und die Ergebnisse, die als JSON zurückgeliefert werden, werden nach Kategorie entsprechend im Screen gezeigt. Diese Methode entspricht den Endpoint `byCategory` in unsere Dokumentation

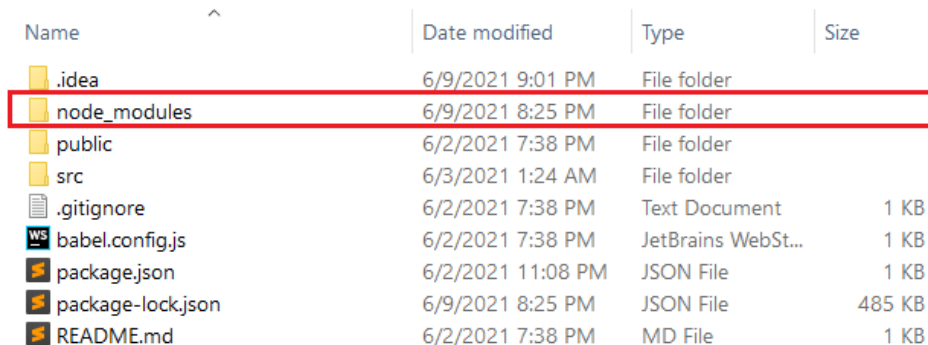
(https://eis.dke.univie.ac.at/eis2021/details/?id=479&path=documentation/4_Microservice_Places_Static.md)

Endpoint on DKE Server wo POST Request geschickt wird:

`https://eis.dke.univie.ac.at/micro-service-controller-rest/rest/msc/callMicroserviceForced?microserviceId=7717445d-958c-4957-ab73-05ff9b157740&operationId=byCategory`

6 Wie starte ich das Frontend lokal auf meinem Computer?

1) Im Frontend Ordner wechseln.



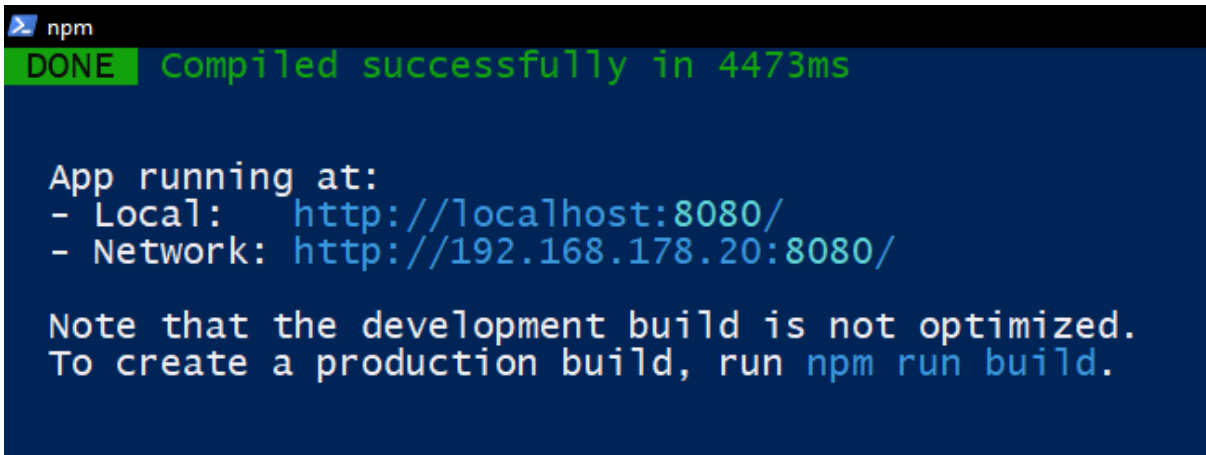
Name	Date modified	Type	Size
.idea	6/9/2021 9:01 PM	File folder	
node_modules	6/9/2021 8:25 PM	File folder	
public	6/2/2021 7:38 PM	File folder	
src	6/3/2021 1:24 AM	File folder	
.gitignore	6/2/2021 7:38 PM	Text Document	1 KB
babel.config.js	6/2/2021 7:38 PM	JetBrains WebSt...	1 KB
package.json	6/2/2021 11:08 PM	JSON File	1 KB
package-lock.json	6/9/2021 8:25 PM	JSON File	485 KB
README.md	6/2/2021 7:38 PM	MD File	1 KB

Abb. 10 Frontend Ordner mit dem neu eingefügten `node_modules` Ordner

Sie werden bemerken, dass der Ordner „`node_modules`“ fehlt. Der nächste Schritt wäre diese zu installieren und das Frontend zu starten. Das wäre mit folgenden Schritten möglich:

- 2) Command Prompt (CMD) im Order starten.
- 3) in CMD: `npm install`
- 4) in CMD: `npm run serve`

- 5) Das Frontend sollte auf localhost:8080 laufen.

A terminal window with a dark blue background. The title bar shows a terminal icon and the text 'npm'. The first line of output is 'DONE' in white text on a green background, followed by 'Compiled successfully in 4473ms' in green text. Below this, the text 'App running at:' is shown in white. This is followed by two lines of information: '- Local: http://localhost:8080/' and '- Network: http://192.168.178.20:8080/'. The final line of text is a note in white: 'Note that the development build is not optimized. To create a production build, run npm run build.'

```
npm
DONE Compiled successfully in 4473ms

App running at:
- Local: http://localhost:8080/
- Network: http://192.168.178.20:8080/

Note that the development build is not optimized.
To create a production build, run npm run build.
```

Abb. 11 Konsole mit dem laufenden Frontend

7 Arbeitsaufteilung

Antonio Gasparevic (01506462)	4,5,6,7,8
Nemanja Srdanovic (01576891)	4,5,6,7,8
Kevin Çobaj (01635087)	1,2,3,5,6

8 Conclusio

Der Microservice wurde auf dem DKE Server deployed und ist einsatzbereit. User können auf die von OLIVE bereitgestellten Endpoints zugreifen und diese verwenden. Das Frontend wurde nur für den Lokalen Gebrauch entwickelt und muss deshalb auch lokal installiert werden, um verwendet werden zu können. Abschließend haben wir ein Video als Tutorial hochgeladen, dass unseren Microservice erklärt und demonstriert. Video ist abrufbar unter dem Link:

<https://www.youtube.com/watch?v=NUuNDO-o7KE>

9 Literaturverzeichnis

<https://nominatim.openstreetmap.org/>

<https://www.geoapify.com/>

10 Abbildungsverzeichnis

Abb. 1 UML-Places-Klassendiagramm	6
Abb. 2 Endpoint Integration byAdress im Backend.....	7
Abb. 3 Endpoint Integration byCategorie im Backend	7
Abb. 4 Endpoint Integration getSubcategories im Backend.....	8
Abb. 5 Endpoint Integration searchStatistic im Backend	8
Abb. 6 Microservices auf der Webseite	9
Abb. 7 UML-Places-Sequenzdiagramm	9
Abb. 8 Frontend Ansicht der Startseite auf localhost:8080	10
Abb. 9 Frontend Ansicht mit Network Entwicklerkonsole	11
Abb. 10 Frontend Ordner mit dem neu eingefügten node_modules Ordner	12
Abb. 11 Konsole mit dem laufenden Frontend	13