

Conference Paper Title*

*Note: Sub-titles are not captured in Xplore and should not be used

1st Justin Frommberger
Interaktionstechnik und Design
Hochschule Hamm-Lippstadt
City, Country
email address or ORCID

2nd Jonas Gerken
Interaktionstechnik und Design
Hochschule Hamm-Lippstadt
City, Country
email address or ORCID

3rd Benedikt Lipinski
Interaktionstechnik und Design
Hochschule Hamm-Lippstadt
Soest, Deutschland
benedikt.lipinski@stud.hshl.de

4th Phillip Wagner
Interaktionstechnik und Design
Hochschule Hamm-Lippstadt
City, Country
email address or ORCID

Abstract—This document is a model and instructions for \LaTeX . This and the `IEEEtran.cls` file define the components of your paper [title, text, heads, etc.]. ***CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract.**

Index Terms—component, formatting, style, styling, insert

I. PARTS OF INTEGRATION

II. SERVER

In diesem Kontext spielt der Server eine sehr wichtige rolle in der Kommunikation zwischen den ausführenden parts des Projektes. Durch den Server und seine strukturen wird letztendlich erst eine plattform geschaffen, die Allen fahrzeuge und den Kunden (Usern) eine Möglichkeit bietet eine verbindung unter einander zu schaffen und weitere Aufgaben zu erledigen. Konkret waren die aufgaben des Servers:

Anmeldung von clients
Das nächstgelegene Fahrzeug finden
Jedem client eine eindeutige ID zuordnen
Übermittelt fahrzeugdaten an den Kunden
Interne verarbeitung einer fahrzeug bestellung

A. Anmedlung von Clients

Aufgabe des Servers ist es eine Anmeldung von clients zu ermöglichen um einerseits nur aus dem pool der aktuell aktiven, freie fahrzeuge auszuwählen und andererseits einer unbekannten menge an Fahrzeugen und kunden die Möglichkeit zu bieten am angebot teilzuhaben. Zu den Clients gehören sowohl die Kunden (User), wie auch alle Fahrzeug typen , damit sind alle Servicefahrzeuge mit den unterkat-egorien: Police, Firefighter, ambulance gemeint und zuletzt auch Fahrzeuge der Kategorie Taxi.

Um auf eine Unbekannte Anzahl an sich zu registrierenden clients reagieren zu können, muss der initiative schritt durch den Client erfolgen. Zur Registrierung sendet der Client eine nachricht mittels mqtt Protokoll über die adressen des Users oder der Fahrzeug typen mit dem ersten teil "adresse = hshl/mqtt_exercise/" und der endung des jeweiligen fahrzeug typen, also: "user,taxi,police,firefighter,ambulance" an den Server. Dieser verarbeitet die nachricht bei erhalt in der Funktion

```
def receive()  
    ...  
    messageprocessing(temp)  
    ...
```

wobei die aufgabe der Funktion *receive()* eher die generelle verarbeitung der mqtt nachricht darstellt und nicht die zuordnung der nachricht zu einem bestimmten anwendungszweck. Dieser wird anschließend durch den aufruf der Funktion *messageprocessing* und einer teilung des übergebenen Arrays in die informationen zu inhalt und Adresse der nachricht erreicht. um ein auslesen der nachricht überhaupt zu ermöglichen, ist es notwendig die Empfangenen nachricht erst in das Json format zu codieren um anschließen die vorteile einer verarbeitung mit Json datasets zu nutzen. dies wird mit der textzeile

```
def messageprocessing(msg)  
    json.loads(str(msg[1]))
```

erreicht.

Zur besseren Identifizierung und klassifizierung als Registrierungsnachricht, wird in dieser die ID des clients durch das wort "register" ersetzt. Dies dient wie schon erwähnt einerseits der besseren einordnung, anderer seits half das menschen lesbarhalten von nachrichten bei der entwicklung ungemein. Einen negativen einfluss auf den erfolgreichen ablauf der registrierung des clients hat dies nicht, da eine ID erst mit antwort des servers an den client vergeben wird. Die nachricht, die ein Client zur Registrierung/Anmeldung senden muss sieht wie folgt aus:

```
data={  
    "id": "register",  
    "name": name,  
    "coordinates": coor  
}
```

Zudem wird in der Internen verarbeitung ein neuer kanal für die weitere kommunikation mit dem Client geschaffen, so dass eine direkte kommunikation mit diesem möglich ist ohne das andere clients hierdurch beeinträchtigt werden. Die adresse des neuen kanal wird unter zuhulfe nahme der druch den server in den funktionen *registrationUser(data)* und *registrationCar(data, type)* vergebenen ID geöffnet und mittels einer nachricht auf dem kanal "hshl/mqtt_exercise/user/back" an diesen zurück gesendet.

Anhand des Quellscodes für das registrieren des users wird gezeigt, wie die vergabe einer Neuen ID und das einspeichern des clients in den server funktioniert, dies ist ganz ähnlich für das vorgehen bei der registrierung von fahrzeugen, mit dem einzigen unterschied, das in diesem fall mittels einer separierung durch den übergabe wert "type" die einzelnen fahrzeuge unterschieden werden können. Desweiteren wird bei der registrierung der fahrzeuge noch der status "free" vergeben.

zuerst wird durch den aufruf der Funktion *findid(user)* die kleinste noch freie ID aus der liste der angemeldeten fahrzeuge gesucht, indem die höchste vergebene ID gesucht wird und um einen zähler höher zurück gegeben wird. Anschliessend wird in der methode *registrationUser(data)* durch ein weiteres

durchlaufen der liste geprüft ob bereits ein user mit dem selben namen vorhanden ist und bei negativem ergebnis in die liste aller user eingetragen.

der user bekommt abschliessend auf dem rückkanal eine nachricht mit seiner eindeutigen ID.

B. Bestellen eines Fahrzeugs

REFERENCES

[1]