Anhang - MS6

1 Implementationsabgabe

Die Implementierung des TeamDrive Systems basiert auf verschiedene Module. Damit Sie ein besseres Verständnis bekommen, wird die Aufführung des Codes erst Clientseitig und dann Serverseitig gezeigt.

1.1 Client - Android

Grundsätzlich lässt sich der Client mit den verschiedenen Klassen und Activitys beschreiben. Um darzustellen, wie der Client funktioniert, wird hier an dieser Stelle nur auf die relevanten Codeteile eingegangen.

1.1.1 GPS Tracker

Bei dem GPS Tracker handelt es sich um die Lokalisation der Koordinaten eines Benutzers. Die GPS Daten werden durch einen Thread in der HomeActivity aufgerufen. Dieser Vorgang wird nach dem Eintragen als Fahrer oder Mitfahrer automatisch ausgeführt. Die Klasse “GPS Tracker” übernimmt für die Applikation das Abrufen der GPS Koordinaten der Benutzer. Mittels “getLocation” werden die Daten durch das GPS Modul aufgerufen. Zuvor muss jedoch abgefragt werden, ob das GPS Modul aktiviert ist. Ist das Modul aktiviert, werden die Koordinaten minütlich oder bei Änderung der Position abgerufen.

```
private static final long MIN_DISTANCE_CHANGE_FOR_UPDATES = 10;
    private static final long MIN_TIME_BW_UPDATES = 1000 * 60 * 1;

    protected LocationManager locationManager;

    public GPSTracker(Context context) {
        this.context = context;
        getLocation();
    }

    public Location getLocation() {
        try {
            locationManager = (LocationManager) context.getSystemService(LOCATION_SERVICE);
            isGPSEnabled = locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER);
            isNetworkEnabled = locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
            if(!isGPSEnabled && !isNetworkEnabled) {
            } else {
                this.canGetLocation = true;
                if (isNetworkEnabled) {
                    locationManager.requestLocationUpdates(
                            LocationManager.NETWORK_PROVIDER,
                            MIN_TIME_BW_UPDATES,
                            MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                    if (locationManager != null) {
                        location = locationManager
                                .getLastKnownLocation(LocationManager.NETWORK_PROVIDER);
                        if (location != null) {

                            latitude = location.getLatitude();
                            longitude = location.getLongitude();
                        }
                    }
                }
                if(isGPSEnabled) {
                    if(location == null) {
                        locationManager.requestLocationUpdates(
                                LocationManager.GPS_PROVIDER,
                                MIN_TIME_BW_UPDATES,
                                MIN_DISTANCE_CHANGE_FOR_UPDATES, this);
                        if(locationManager != null) {
                            location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);
                            if(location != null) {
                                latitude = location.getLatitude();
                                longitude = location.getLongitude();
                            }
                        }
                    }
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return location;
    }
```

Abbildung 1: GPS Modul Android Client

1.1.2 Anzeigen der Mitfahrer

Damit der Fahrer sich die Mitfahrer in einer Karte anzeigen lassen kann, wurde die Google Map implementiert. Zusätzlich ist eine Zielnavigation in der Google Map integriert. Damit die Google Map auf dem Smartphone angezeigt wird, musste man die Applikation mit einem Google Konto unter "https://console.developers.google.com/project" signieren lassen.

Schlüssel für Android-Anwendungen

API-Schlüssel	AIzaSyCDyrht0i3I5eDa6jrk5GsT8zu6uqMOALk
Android-Anwendungen	C1:8F:06:80:58:5C:0A:3D:1F:D0:4F:D0:03:FF:F5:DA:7E:08:30:91;com.teamdrive.example
Aktivierungsdatum	24.06.2015, 12:15:00
Aktiviert durch	vskueppers@gmail.com (ich)

Abbildung 2: Signierte APK

Der API-Schlüssel musste dann in die Datei "AndroidManifest.xml" eingetragen werden.

```
<meta-data
    android:name="com.google.android.maps.v2.API_KEY"
    android:value="AIzaSyCDyrht0i3I5eDa6jrk5GsT8zu6uqMOALk"
/>
```

Abbildung 3: API-Schlüssel im "AndroidManifest.xml"

Damit die Google Map nun in der Applikation angezeigt wird, muss diese in der "FahrerActivity.java" initialisiert werden. Dies geschieht durch den folgenden Code.

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_fahrer);
    context = this.getApplicationContext();
    sharedPref = context.getSharedPreferences(
            "prefs", Context.MODE_PRIVATE);
    db = new DatabaseHelper(context);
    try {
        initilizeMap();

    } catch (Exception e) {
        e.printStackTrace();
    }
    googleMap.setMyLocationEnabled(true);
    googleMap.getUiSettings().setMyLocationButtonEnabled(true);
}
 private void initilizeMap() {
        if (googleMap == null) {
            googleMap = ((MapFragment) getFragmentManager().findFragmentById(
                    R.id.map)).getMap();

            // check if map is created successfully or not
            if (googleMap == null) {
                Toast.makeText(getApplicationContext(),
                        "Sorry! unable to create maps", Toast.LENGTH_SHORT)
                        .show();
            }
        }
    }
```

Abbildung 3: Initialisierung der Google Map

Um nun die Mitfahrer auf der Karte anzeigen zu lassen, müssen die Koordinaten zu jedem Mitfahrer vom TeamDrive Server abgerufen werden. Dies geschieht mit einem Aufruf “holeMitfahrer()”. Nachdem die Koordinaten abgerufen wurden, werden diese in Variablen gespeichert, damit sie dann als Pins auf der Karte angezeigt werden können.

```
public void holeMitfahrer(){
    Ion.with(getApplicationContext())
        .load(serverUrl + "/getMitfahrer")
        .setBodyParameter("f_id", user_id)
        .setBodyParameter("e_id", aktuelleventid)
        .asJsonObject()
        .setCallback(new FutureCallback<JsonObject>() {
            @Override
            public void onCompleted(Exception e, JsonObject result) {
                int anzahlalt = 0;
                int state = result.get("state").getAsInt();
                if (state == 1)//successfull, do stuff
                {
                    JsonArray mitfahrer = result.getAsJsonArray("mitf_position");
                    for(int i = 0; i<mitfahrer.size(); i++){
                        double longitude = 0;
                        double latitude = 0;
                        if (mitfahrer.get(i).getAsJsonObject().has("longitude"))
                         longitude= mitfahrer.get(i).getAsJsonObject().get("longitude").getAsDouble();
                        if (mitfahrer.get(i).getAsJsonObject().has("latitude"))
                         latitude= mitfahrer.get(i).getAsJsonObject().get("latitude").getAsDouble();
                        String fahrername = mitfahrer.get(i).getAsJsonObject().get("mname").getAsString();
                        MarkerOptions marker = new MarkerOptions().position(new LatLng(latitude, longitude)).title(fahrername);
                         if (latitude > 0)
                        googleMap.addMarker(marker);
                    }
                }
            }
        });
}
```

Abbildung 4: Anzeigen der Mitfahrer

1.1.3 Nachrichten an Fahrer

Diese Funktion ist für den Mitfahrer gedacht. Mit dieser Funktion bekommt der Mitfahrer automatisch mitgeteilt wer sein Fahrer ist und kann ihm so direkt eine Nachricht schicken ohne dass er sich erstmal informieren muss, wer ihn abholt. Um den Fahrer abzurufen wird folgender Code benötigt:

```
String fvorname = "", fnachname = "", f_id = "";
   boolean cansend = false;
   public void getFahrer(){
       Log.d("p_id", user_id);
       Log.d("e_id", aktuelleventid);
       Ion.with(getApplicationContext())
               .load(serverUrl + "/getDriver")
               .setBodyParameter("p_id", user_id)
               .setBodyParameter("e_id", aktuelleventid)
               .asJsonObject()
               .setCallback(new FutureCallback<JsonObject>() {



                   @Override
                   public void onCompleted(Exception e, JsonObject result) {


                       int state = result.get("state").getAsInt();
                       if (state == 1)//successfull, do stuff
                       {
                           fvorname = result.getAsJsonObject("driver").get("vorname").getAsString();
                           fnachname= result.getAsJsonObject("driver").get("nachname").getAsString();
                           f_id= result.getAsJsonObject("driver").get("f_id").getAsString();
                           txtName.setText(fvorname + " " + fnachname);
                           cansend = true;
                       }

                       initializeDataFromSharedPref();
                   }
               });
   }
```

Abbildung 5: Abrufen des Fahrers

Damit der Mitfahrer an den Fahrer eine Nachricht kann, sendet dieser einen Request an den TeamDrive Server.

```
public void sendMessage(){
        message = edtnachricht.getText().toString();
        Ion.with(getApplicationContext())
                .load(serverUrl + "/sendMessage")
                .setBodyParameter("p_id", user_id)
                .setBodyParameter("f_id", f_id)
                .setBodyParameter("e_id", aktuelleventid)
                .setBodyParameter("msg", message)
                .asJsonObject()
                .setCallback(new FutureCallback<JsonObject>() {
                    @Override
                    public void onCompleted(Exception e, JsonObject result) {

                        int state = result.get("state").getAsInt();
                        if (state == 1)//successfull, do stuff
                        {
                            edtnachricht.setText("");
                            Nachricht nachricht = new Nachricht(0, aktuelleventid, fvorname,"", fnachname, "ja", message);
                            db.createnachricht(nachricht);
                            //update listview
                        }
                        setEintragList();
                        cansend = true;
                    }
                });
    }
```

Abbildung 6: Nachricht senden

2.1 Server - NodeJS und MongoDB

2.1.1 REST-API Schnittstelle

Wie in der Systemarchitektur beschrieben, wurden NodeJS und MongoDB als Server- und Datenbankinstanz gewählt.

In Abbildung 7 befindet sich der Code für die REST-API Schnittstelle. Der aufgeführte API-Aufruf wird durch einen GET-Request getriggert, sodass auf die Ressource “/home/:id/players” ausgeführt wird. Mit dem Aufruf liefert die Ressource die Nutzerdaten und die Spielerdaten.

```
app.get('/home/:id/players', function(req, res){
    store.getPerson(req.params.id, function(err, person){
        if (!person){
            res.status(400).send(err);
        }else{
            store.getAllPlayers(person.per_mannschaft, function(err, players){
                if(err) {
                    res.writeHead(500, "Es ist ein Fehler aufgetreten");
                }else{
                    res.render('players', {person:person, players:players});
                }
            });
        }
    });
});
```

Abbildung 7: REST-API Schnittstelle

2.1.2 Ereignissteuerung

Die Abbildung 8 zeigt die Ereignissteuerung des Servers bei einem Aufruf eines Clients. Der Server reagiert aber nicht nur auf Ereignisse eines Clients sondern kann sich auch selbst initiieren. In dem hier vorliegenden Beispiel werden die überlieferten Daten (Mitfahrer IDs) dafür benutzt, um aus der Mitfahrer Datenbank die Koordinaten zu den IDs zu zuordnen. Als Antwort bekommt der Client ein Mitfahrer Array im JSON Format mit der Mitfahrer ID, dem Mitfahrer Namen und den GPS Koordinaten (Latitude und Longitude).

```
getPosition: function(data, callback) {
    var mitfahrer = [];
    var count = 0;
    var anzahl = data.length;
    if (data == null){
           callback('Es wurden keine Daten gefunden!');
    }else{
        data.forEach(function(person){
            Mitfahrer.find({p_id:person.m_id}, function(err, user){
                user.forEach(function(position){
                    count++;
                    mitfahrer.push({
                        m_id: person.m_id,
                        mname: person.mname,
                        latitude: position.latitude,
                        longitude: position.longitude});
                    if(anzahl == count) callback(null, mitfahrer);
                });
            });
        });
    }
}
```

Abbildung 8: JSON-Request an den Client