Projektarbeit
E-Charge App

NUTZERZENTRIERTE SOFTWAREENTWICKLUNG
WS21/22
–
DOKUMENTATION

---

von

Herr Paul Bartelt (745280)

Herr Samuel Koob (769070)

Herr Sebastian Zill (769544)

Datum: 11.02.2022

INHALT

Sonder-Inhaltsverzeichnisse

Anhang:

Darstellungsverzeichnis:

1. AUFGABENBESCHREIBUNG

Gegenstand der Praktikumsaufgabe ist die Entwicklung einer App zur E-Mobilität. Nutzer der App sollen Zugang zu Informationen von Ladestationen für Elektrofahrzeuge erhalten. Dafür steht eine Datei der Bundesnetzagentur mit allen registrierten Ladesäulen und Informationen zu diesen (Steckertypen etc.) zur Verfügung. Darüber hinaus sollen nicht nur „normale“ Nutzer, sondern auch Servicemitarbeiter, die sich um die Instandhaltung der Ladesäulen kümmern, einen Zugang erhalten. Am Ende soll ein funktionsfähiger Prototyp entstehen, der möglichst intuitiv für Nutzer gestaltet ist. Dazu werden verschiedene Rahmenbedingungen durch das Praktikumsaufgabenblatt und Rücksprachen mit Prof. Dr. Wiedling gestellt. Diese werden in den folgenden Abschnitten aufgelistet und anhand von Erläuterungen, Codebeispielen und Bildern weiter beleuchtet.

2. PROJEKTANFORDERUNGEN

2.1 USER STORIES

Im 1. Praktikumstermin wurden User Stories formuliert, welche einen Teil der Rahmenbedingungen bilden und den Verlauf der App-Entwicklung maßgeblich beeinflussen. Diese sind folglich zusammengefasst, wobei User Stories wie zum Beispiel „Als Nutzer möchte ich eine Route planen können.“ nicht behandelt werden, da dies eine andere Aufgabenstellung verlangt:

User Stories Nutzer:

1. Als Nutzer möchte ich Ladesäulen in Umgebung und Ferne angezeigt bekommen,
   um meine Route besser planen zu können.
2. Als Nutzer möchte ich diverse Informationen zu Ladesäulen angezeigt bekommen,
   um meine Auswahl treffend wählen zu können.
3. Als Nutzer möchte ich defekte Ladestationen melden können,
   damit diese schnellstmöglich repariert werden.
4. Als Nutzer möchte ich Favoriten hinzufügen können,
   um meine Lieblingsstandorte schnell wieder finden zu können.
5. Als Nutzer möchte ich nach bestimmten Kriterien filtern können,
   um passende Ladesäulen zu finden.

User Stories Service:

1. Als Service Mitarbeiter möchte ich defekte Ladesäulen angezeigt bekommen,
   um diese schnell bearbeiten zu können.
2. Als Service Mitarbeiter möchte ich reparierte Ladesäulen wieder freigeben können,
   um die Nutzer und meine Kollegen über die getätigte Arbeit zu informieren.
3. Als Service Mitarbeiter möchte ich Angaben zu den defekten Ladesäulen erhalten,
   um meine Auswahl anzufahrender Ladesäulen passend treffen zu können.
4. Als Service Mitarbeiter möchte ich nach Ladesäulen in meiner Nähe filtern können,
   um meine Arbeitsrouten besser planen zu können.

2.2 RAHMENBEDINGUNGEN

Aus den User Stories, dem Aufgabenblatt und Rücksprachen mit Prof. Dr. Wiedling lassen sich schließlich folgende Rahmenbedingungen zur Projektarbeit formulieren:

- Es soll ein funktionsfähiger Prototyp entstehen, den man als APK Datei exportieren und auf mobilen Geräten installieren und ausführen kann.

- In der Applikation sollen Bedienungen nach Nutzern und Service Mitarbeitern unterschieden werden können.

- Der Nutzer soll auf dem individuell verwendeten Endgerät über persistente und persönliche Daten, wie Favoriten und Standort, lokal verfügen.

- Für die große Anzahl an Ladesäulen soll die Funktionalität eines RecyclerViews verwendet werden, um nicht alle Ladesäulen auf einmal zu laden und somit die Auslastung des Gerätes und Netzwerkes zu verringern.

- Eine Umkreissuche, um nur Ladesäulen in bestimmten Entfernungen zum Nutzer anzuzeigen, soll implementiert werden.

- Diverse weitere Filterfunktionen nach den lokalen Präferenzen des Nutzers sollen für die Ladesäulen möglich sein. Unter diese fallen unter anderem favorisierte und als defekt markierte Standorte.

- Der Nutzer soll für Ladesäulen Defekte melden, und *nur* der Service Mitarbeiter diese werden entfernen können.

- Die App soll eine Funktionalität zur Mehrsprachigkeit anhand der eingestellten Systemsprache des Endgerätes haben.

- Die App soll in mindestens einem Screen im Landscape Modus benutzbar sein.

3. PROJEKTUMFELD

Das Projektumfeld wird vorgegeben. Die verwendete Programmiersprache ist **Java** und die verwendete IDE **Android Studio**. Diese beinhaltet auch gleichzeitig einen **Emulator** zum Testen des Prototypen. Zusätzlich haben wir die Möglichkeit die App auch nativ auf einem **Android Gerät** zu testen. Zur besseren Teamarbeit wird **GitLab** zur Versionsverwaltung und **Discord** zur Kommunikation verwendet. Um ein übersichtliches Design zu erstellen, wird das Programm **InVision Studio** (ähnlich AdobeXD) benutzt.

4. ABLAUF UND TERMINPLANUNG

Die Terminplanung orientiert sich vor allem an dem Praktikumsblatt und den im vorherigen Abschnitt erwähnten User Stories und Anforderungen. In Tabelle 1 wird dies noch einmal zusammengefasst.

Termin	Datum	Inhalt
1	04.11.2021	Erstellen von UserStories, Diskussion der Ergebnisse
2	18.11.2021	Erste Schritte mit Android Studio, Screens erstellen, BottomNavigationBar
3	02.12.2021	Abläufe konkretisieren, Views ergänzen, Adapterklasse
4	16.12.2021	RecyclerView, persistente Funktionalität, Vervollständigung
5	20.01.2022	Usability Test, Apps anderer Gruppen betrachten, Abnahme der PVL

Tabelle 1: Praktikumstermine und -inhalte

Somit ist der Zeitraum festgelegt und die einzelnen Schritte können nacheinander abgearbeitet werden, wobei sich teilweise Features überschneiden oder voneinander abhängig sind, sodass andere Funktionen früher gebraucht werden als gedacht oder Klassen und deren Funktionen durch spätere Stände ergänzt wurden.

In der folgenden Abbildung ist unser Arbeitsablauf grob skizziert:

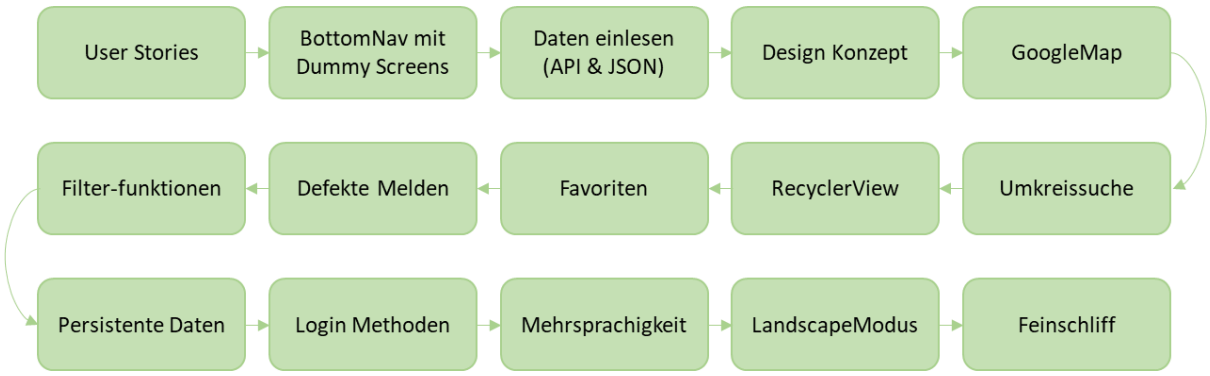


Abbildung 1: Arbeitsablauf

5. FUNKTIONALE UMSETZUNG DER PROJEKTANFORDERUNGEN

In diesem Abschnitt wird die Umsetzung der bereits in den vorherigen Abschnitten gestellten Projektanforderungen dargestellt, um in der funktionierenden Applikation anhand der User Stories zu resultieren. Es wird anhand des Arbeitsablaufes aus Abbildung 1 strukturiert, wobei einige Punkte parallel oder in Abhängigkeiten zueinander umgesetzt und ergänzt wurden.

5.1 SCREEN NAVIGATION

Nach der Entwicklung der User Stories und dem Kennenlernen der IDE Android Studio ging es zunächst darum die ersten Screens zu erstellen. Dies beinhaltet auch die Navigation zwischen den einzelnen Screens. Es gibt verschiedene Möglichkeiten zwischen zwei Screens hin und her zu wechseln. Die zwei am häufigsten verwendeten sind Tabs und Bottom Navigation. Tabs sind meist am oberen Bildschirmrand angesiedelt und vielen Usern bereits durch Verwendung eines Internet-Browsers bekannt. Die Bottom Navigation ist hingegen am unteren Bildschirmrand angesiedelt und vielen Usern durch die Navigation innerhalb eines Android oder IOS Betriebssystems bekannt.

Unsere Entscheidung fiel auf die Bottom Navigation, da diese aus der Sicht des Nutzers einfacher und intuitiver zu bedienen ist. Sie ist am unteren Bildschirmrand angesiedelt, wodurch es dem Nutzer einfacher fällt sie zu erreichen und hierdurch auch die Barrierefreiheit verbessert wird, da die Navigation auf jedem Screen gleich immer direkt sichtbar ist und die einzelnen Seiten durch ihre Icons erläutert sind. Da es sich außerdem um eine reine Android-App handelt und keine Web-App oder ähnliches vorgesehen ist, ist die Bedienung am unteren Bildschirmrand analog zur Bedienung des Smartphones und vielen anderen Apps.

Erstellt wurde die Bottom Navigation Bar mit Hilfe eines YouTube-Tutorials [1] und der Android Studio Dokumentation [2]. Auch hier muss wieder eine Entscheidung getroffen werden: Verwendet man Activities oder Fragments für die Screens?

Wir haben uns schließlich für Fragments entschieden, um eine übersichtlichere Projektstruktur zu gewährleisten. Eine Gegenüberstellung der beiden Möglichkeiten ist in Abbildung 2 zu sehen.

Activity	Fragment
Activity is an application component that gives a user interface where the user can interact.	The fragment is only part of an activity, it basically contributes its UI to that activity.
Activity is not dependent on fragment	Fragment is dependent on activity. It can't exist independently.
we need to mention all activity it in the manifest.xml file	Fragment is not required to mention in the manifest file
We can't create multi-screen UI without using fragment in an activity,	After using multiple fragments in a single activity, we can create a multi-screen UI.
Activity can exist without a Fragment	Fragment cannot be used without an Activity.
Creating a project using only Activity then it's difficult to manage	While Using fragments in the project, the project structure will be good and we can handle it easily.
Lifecycle methods are hosted by OS. The activity has its own life cycle.	Lifecycle methods in fragments are hosted by hosting the activity.
Activity is not lite weight.	The fragment is the lite weight.

Abbildung 2: Gegenüberstellung Activity und Fragment [3]

5.2 DATEN EINLESEN

Nach dem Erstellen der Dummy Screens und der Bottom Navigation Bar ging es um das Einbinden der gegebenen Daten. Diese liegen in Form einer CSV-Datei und dank Herrn Quick auch als API vor. Wir haben uns dazu entschieden die API zu nutzen, da sich der offizielle Link der Bundesnetzagentur als externe Quelle [4] oder dessen interne Daten jederzeit verändern könnten und zudem dadurch das Dateiformat JSON vorliegt. Wir hatte ebenso bereits in letzten Semestern persönlichen Kontakt und Zusammenarbeit mit Herrn Quick als Lerngruppe und uns daher vor Projekt Kick Off bereits über die Vor- und Nachteile einer JSON gegenüber einer CSV ausgetauscht. Ein solcher Vorteil von JSON wäre zum Beispiel, dass sich die Dateistruktur generell und über Bibliotheken wie GSON besser eignet, um Objekte aus dessen Inhalten zu serialisieren und zu generieren.

Die Daten werden mittels der API von Herrn Quick [5] und dem Zusammenspiel mehrerer Klassen und Threads eingelesen:

DownloadThread.java: In einem weiteren Thread wird über die Implementation einer runnable die auf der API liegende JSON-Datei [5] heruntergeladen. Wir haben uns für die Verwendung eines parallelen Threads entschieden, da wir in ersten Tests festgestellt haben, dass der Download einige Zeit benötigt und in dieser Zeit die gesamte App stillsteht, was den Nutzer fälschlicherweise in die Vermutung eines Absturzes verleiten und generell irritieren könnte. Der Download und die Verarbeitung der JSON Datei wird in der Funktion getJson() abgewickelt. Anschließend wird der JSON-String über die Google - GSON Library mit der Funktion gson.fromJson() in Objekte konvertiert.

Diese Liste des DownloadThreads wird dann über die initialize() der **StationAPI.java** anhand der Funktion getStations() in den **GlobalStorage.java** übergeben, sodass die weiteren Fragmente auf diese Liste zugreifen und die Stationen verwenden können.

```
1.  public class DownloadThread implements Runnable
2.  {
3.      private volatile ArrayList<Ladestation> stations;
4.
5.      /**
6.       * Overrides the Runnable's run method
7.       */
8.      @Override
9.      public void run()
10.     {
11.         Gson gson = new Gson(); // get gson object (generate objects from json format)
            // set type of Ladestation object for gson return
12.         Type return_type = new TypeToken<ArrayList<Ladestation>>(){}.getType();
13.         final String json = getJson(); // Load json from API url and save into var
            // generate and return ArrayList from parsed json string with gson
14.         this.stations = gson.fromJson(json, return_type);
15.     }
16.
17.     /**
18.      * Fetches the stations as a JSON from the given URL
19.      * @return JSON-Formatted string
20.      */
21.     private String getJson()
22.     {
23.         BufferedReader reader = null;
24.         try
25.         {
26.             try
27.             {
28.                 URL url = new URL("https://api.aurora-theogenia.de/chargingstations/charg-
    ing_stations.json"); // read into buffer from url
29.                 // Create buffer to load giant string by chunks
30.                 reader = new BufferedReader(new InputStreamReader(read(url)));
31.                 StringBuffer buffer = new StringBuffer();
32.
33.                 int read;
34.                 char[] chars = new char[1024]; // Read every 1024 chars into char array
```

```
35.                 while ((read = reader.read(chars)) != -1)
36.                 {
37.                     buffer.append(chars, 0, read); // Append those chars onto buffer
38.                 }
39.                 return buffer.toString(); // Convert buffer to string and return it
40.             }
41.
42.             finally // once done close the reader again
43.             {
44.                 if (reader != null)
45.                 {
46.                     reader.close();
47.                 }
48.             }
49.         }
50.         catch (Exception e)
51.         {
52.             e.printStackTrace();
53.             return null;
54.         }
55.     }
56.
57.     /**
58.      * Reads a URL connection's stream and returns it
59.      * @param url The URL to get the stream from
60.      * @return The fetched InputStream
61.      * @throws IOException
62.      */
63.     private InputStream read(URL url) throws IOException
64.     {
65.         HttpURLConnection httpCon = (HttpURLConnection) url.openConnection();
66.         httpCon.setRequestProperty("User-Agent", "Mozilla/5.0 (Windows NT 6.1) AppleWeb-
    Kit/537.36 (KHTML, like Gecko) Chrome/41.0.2228.0 Safari/537.36");
67.         httpCon.setConnectTimeout(10000);
68.         httpCon.setReadTimeout(10000);
69.         return httpCon.getInputStream();
70.     }
71.
72.     // return the stations ArrayList to save into ArrayList in StationsAPI
73.
74.     /**
75.      * Getter for all stations
76.      * @return the stations ArrayList
77.      */
78.     public ArrayList<Ladestation> getStations(){return stations;}
79. }
```

Codebeispiel 1: DownloadThread.java

```
1.  (…)
2.      /**
3.       * Initializes the Station API with default values
4.       * @param context
5.       */
6.      public static void initialize(Context context)
7.      {
8.          // generate downloadthread object
9.          DownloadThread dl = new DownloadThread();
10.         Thread thread = new Thread(dl);
11.         thread.start();
12.         try
13.         {
14.             thread.join();
15.         }
16.         catch (InterruptedException e)
17.         {
18.             e.printStackTrace();
19.         }
20.         // return and save downloaded ArrayList into api static
21.         GlobalStorage.setAllStations(dl.getStations());
22.
23.         // fill fav array
24.         sqLiteHelper = new SQLiteHelper(context);
25.         GlobalStorage.setFavStations(sqLiteHelper.readFavouritesFromDB());
26.
27.         // fill defect array
28.         GlobalStorage.setDefStations(sqLiteHelper.readFlagsFromDB());
29.         GlobalStorage.setSaveCheckedState(new SaveCheckedState());
30.     }
31. (…)
```

Codebeispiel 2: StationAPI.java

Die Struktur der Objekte sind identisch zur Klasse **Ladestation.java** und deren Daten können über Getter und Setter dieser Klasse abgefragt und bearbeitet werden.

5.3 DESIGN KONZEPT

Nachdem wir nun die Daten in unserer App zur Verfügung und bereit zur Darstellung hatten, wurde von uns als nächsten Schritt ein erstes Design Konzept mit Hilfe des UI/UX Tools InVision Studio entwickelt.

Wir entschieden uns für diesen Zwischenschritt, um uns vor Implementation einen Roten Faden zur Nutzerzentrierung zu bilden, um ein einheitliches und schlankes Design für unsere App zu erhalten.

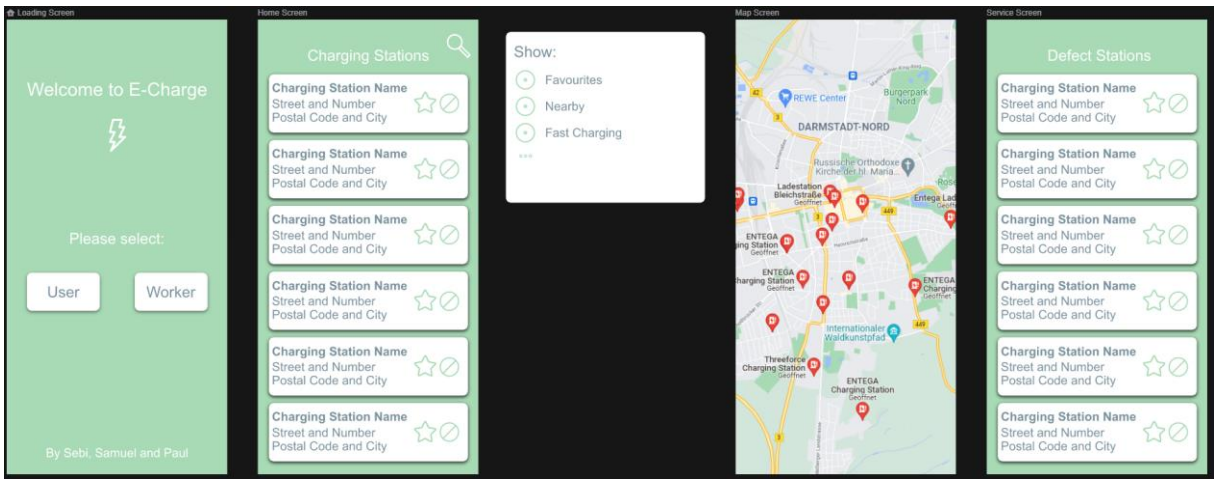


Abbildung 3: Erstes Design Konzept

Beim Erstellen des ersten Designs kam die Idee auf dem User Zugang zu zwei Screens zu ermöglichen, dem Home Screen und der GoogleMap. Der Service-Techniker erhält die Zugriffsrechte auf einen zusätzlichen Service Screen, um *nur* die defekten Ladestationen angezeigt zu bekommen und bearbeiten zu können.

Die entsprechende Bottom-Navigation des allgemeinen Layouts ergänzt diese Designs am unteren Bildschirmrand und wurde zur Visualisierung der restlichen Inhalte hier weggelassen.

5.4 GOOGLEMAP

Auch wenn es unter den Projektanforderungen nicht gelistet ist, haben wir uns dazu entschieden, dem Nutzer eine Karte zur Verfügung zu stellen, um neben dem RecyclerView noch eine weitere bekannte Anzeigemöglichkeit für Ladestationen im Umkreis zu haben. Der Nutzer kann dadurch selbst entscheiden welche Darstellungsweise ihm besser zusagt und für seine Zwecke passender ist. Zusätzlich bietet die GoogleMap weitere Möglichkeiten für zukünftige Implementierungen (siehe Kapitel 5).

Die GoogleMap wurde in die App mittels eines eigenen Fragments implementiert. Hierfür wurde für die Seite ein eigenes Layout-Fragment **fragment_map.xml** erstellt, welches sich über die gesamte Höhe und Breite erstreckt und über die Fragmentklasse **MapFragment.java** die Karte per inflate reingeladen und asynchron betrieben wird.

Bevor die Karte allerdings geladen und dieser Screen aufgerufen wird, muss eine Funktionalität zur Bestätigung der Standortabnahme des Gerätes hinzugefügt werden. Diese Bestätigung wird bei jedem Start der App in der MainActivity von den Android-Permissions im Manifest angefragt. Insofern diese vorhanden ist, muss der Nutzer nichts weiter tun und die App startet ordnungsgemäß. Sollte aber noch keine Erlaubnis vorliegen, würden Map und Umkreissuche nicht funktionieren und die Bestätigung wird von der App vor dem eigentlichen Start der restlichen Fragmente in der **MainActivity.java** angefragt. Der User erhält dann ein Pop-Up mit der Aufforderung den Standort freizugeben.

```
1.      /**
2.        * Function triggers on create
3.        * @param savedInstanceState
4.        */
5.       @Override
6.       protected void onCreate(Bundle savedInstanceState) {
7.           super.onCreate(savedInstanceState);
8.           setContentView(R.layout.activity_main);
9.
10.          if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCA
                     TION) != PackageManager.PERMISSION_GRANTED && ActivityCompat.checkSelfPermis
                     sion(this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PER
                     MISSION_GRANTED){
11.              ActivityCompat.requestPermissions(this, new String[]{Manifest.permission.AC
                     CESS_FINE_LOCATION, Manifest.permission.ACCESS_COARSE_LOCATION}, request
                     Code);
12.              if(ActivityCompat.checkSelfPermission(this, Manifest.permission.AC
                                 CESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED && Activi
                                 tyCompat.checkSelfPermission(this, Manifest.permission.AC
                                 CESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED){
13.                              Toast.makeText(this , this.getResources().getString(R.string.err
                                 Location), Toast.LENGTH_LONG).show();
14.              }
15.          }
16.
17.          bottomNav = findViewById(R.id.bottom_navigation);
18.          bottomNav.setOnItemSelectedListener(navListener);
19.
20.          StationAPI.initialize(getApplicationContext());
21.          Toast.makeText(this, Integer.toString(GlobalStorage.getAllStations().size()),
                     Toast.LENGTH_SHORT).show();
22.
23.          getSupportFragmentManager().beginTransaction().replace(R.id.fragment_container,
24.                   new LoginFragment()).commit();
25.
26.          bottomNav.setVisibility(View.INVISIBLE);
27.      }
```

Codebeispiel 3: MainActivity.java

Wenn der Standort also bestätigt ist, lädt die Karte per Inflater in dessen View hinein und weitere Funktionalitäten konnten ergänzt werden.

So wurde die im nächsten Punkt beschriebene Umkreissuche für die GoogleMap in das **MapFragment.java** hinzugefügt. Diese Umkreissuche fungiert mit einem so genannten setOnMapClickListener per Klick auf einen bestimmten Punkt der Karte. Für diesen Punkt werden dann alle Standorte der Ladestationen innerhalb einer bestimmten Entfernung (später in den Filtereinstellungen von Punkt 4.9 vom Nutzer einstellbar) angezeigt, wobei Favoriten auch außerhalb dieser Entfernungen als Sonderstandorte angezeigt werden.

Zusätzlich wurden für die Standorte nach deren Status eigene Markierungen auf der Karte ergänzt:

- Rot: Defekt
- Gelb: Favorit
- Blau: Normal funktionstüchtig

Zu guter Letzt wurde noch für jede Station ein Pop-Up hinzugefügt, sodass detailliertere Informationen zu den angewählten Stationen nach Klick auf deren Markierungen angezeigt werden.

Abbildung 4: GoogleMap mit Markierungen

5.5 UMKREISSUCHE

Um sich nicht alle Ladestationen, sondern nur die relevanten in der Nähe anzeigen zu lassen, wird eine Umkreissuche implementiert. Eine Funktion zur Berechnung des Abstands zwischen zwei Geokoordinaten wird im Praktikumsblatt bereits gegeben:

```
1.  public static ArrayList<Ladestation> getProximityStations(ArrayList<Ladestation> station-
    sList, double lat1, double lon1, int dist){
2.          ArrayList<Ladestation> filtered = new ArrayList<>();
3.          stationsList.forEach(ladestation -> {
4.              double lat2 = ladestation.getLat();
5.              double lon2 = ladestation.getLon();
6.              double R = 6371; // Erdradius in km
7.              double dLat = (lat1 - lat2) * Math.PI / 180.0;
8.              double dLon = (lon1 - lon2) * Math.PI / 180.0;
9.              double a = Math.sin(dLat / 2.) * Math.sin(dLat / 2.)
10.                     + Math.cos(lat1 * Math.PI / 180.0)
11.                     * Math.cos(lat2 * Math.PI / 180.0)
12.                     * Math.sin(dLon / 2.) * Math.sin(dLon / 2.);
13.             double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1. - a));
14.             double d = R * c;
15.             // Compare distance
16.             if(d < dist)
17.             {
18.                 filtered.add(ladestation);
19.             }
20.         });
21.         return  filtered;
22. }
```

Codebeispiel 4: Berechnung Abstand zweier Geokoordinaten [5]

Mit Hilfe der Funktion lässt sich das Array aller Ladestationen filtern und ein neues gefiltertes Array mit den Ladestationen in einem bestimmten Umkreis zurückgeben. Die Funktion wurde anschließend für den RecyclerView verwendet und nachträglich als eigene Filterfunktion in die GoogleMap ergänzt.

5.6 RECYCLERVIEW

Der RecyclerView ist explizit gefordert. Am einfachsten lässt er sich mit einer dynamischen Liste beschreiben, die nur die angezeigten Elemente kreiert und somit auch für große Datenmengen gut einsetzbar ist. [7]

Für die Implementierung wird eine Adapter Klasse **StationAdapter.java** benötigt, der die Stationen in das RecyclerView fragment der Liste ergänzt:

```
1.  public class StationAdapter extends RecyclerView.Adapter<StationAdapter.ViewHolder>{
2.      private ArrayList<Ladestation> stationList;
3.      private SQLiteHelper sqLiteHelper;
4.
5.      public StationAdapter(ArrayList<Ladestation> stationList, SQLiteHelper sqLiteHelper)
6.      {
7.          this.stationList = stationList;
8.          this.sqLiteHelper = sqLiteHelper;
9.      }
10.
11.     @NonNull
12.     @Override
13.     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
14.         Context context = parent.getContext();
15.         LayoutInflater inflater = LayoutInflater.from(context);
16.         View stationView = inflater.inflate(R.layout.list_item, parent, false);
17.         return new ViewHolder(stationView);
18.     }
19.
20.     @SuppressLint("SetTextI18n")
21.     @Override
22.     public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
23.         Ladestation ladestation = stationList.get(position);
24.         holder.favouriteButton.setImageResource(R.drawable.ic_baseline_star_25);
25.
26.         // Set top text
27.         TextView stationTextTop = holder.stationTextTop;
28.         stationTextTop.setText(ladestation.getOperator());
29.
30.         // Set middle text
31.         TextView stationTextCenter = holder.stationTextCenter;
32.         stationTextCenter.setText(ladestation.getStreet() +" "+ ladestation.getNumber());
33.
34.         // Set bottom text
35.         TextView stationTextBottom = holder.stationTextBottom;
36.         stationTextBottom.setText(ladestation.getPostalCode() + " " + ladestation.getLoca-
    tion());
37.
38.         // Set station id
39.         holder.stationID = ladestation.id;
40.
41.         // Buttons
42.         ImageButton button = holder.favouriteButton;
43.         if(GlobalStorage.getFavStations().contains(ladestation.id)){
44.             button.setImageResource(R.drawable.ic_baseline_star_24);
45.         }
46.         button.setOnClickListener(v -> toggleButtonFav(ladestation, button));
47.
48.         ImageButton buttonRep = holder.reportButton;
49.         if(GlobalStorage.getDefStations().contains(ladestation.id)){
50.             buttonRep.setImageResource(R.drawable.ic_baseline_flag_25);
51.         }
52.
53.         if(MainActivity.onServiceFragment){
54.             buttonRep.setOnClickListener(v -> toggleButtonRepWorker(ladestation, buttonRep));
55.         }
56.         else{
57.             buttonRep.setOnClickListener(v -> toggleButtonRep(ladestation, buttonRep));
58.         }
59.     }
60.     @Override
61.     public int getItemCount() {
```

```
62.         return stationList.size();
63.     }
64.
65.     public class ViewHolder extends RecyclerView.ViewHolder {
66.         public TextView stationTextTop;
67.         public TextView stationTextCenter;
68.         public TextView stationTextBottom;
69.         public ImageButton favouriteButton;
70.         public ImageButton reportButton;
71.         public int stationID;
72.
73.         public ViewHolder(View itemView) {
74.             super(itemView);
75.             stationTextTop = itemView.findViewById(R.id.text_stationTop);
76.             stationTextCenter = itemView.findViewById(R.id.text_stationCenter);
77.             stationTextBottom = itemView.findViewById(R.id.text_stationBottom);
78.             favouriteButton = itemView.findViewById(R.id.FavouriteButton);
79.             reportButton = itemView.findViewById(R.id.ReportButton);
80.         }
81.     }
82.     @SuppressLint("NotifyDataSetChanged")
83.     public void setNewList(ArrayList<Ladestation> newList){
84.         stationList = newList;
85.         notifyDataSetChanged();
86.     }
87. }
```

Codebeispiel 5: RecyclerView.java

Zunächst werden im Konstruktor des StationAdapters sowohl die in den vorherigen Schritten im GlobalStorage abgelegten Liste aller Stationen als auch die Anbindung an die persistenten Daten der SQLite (folgt in Kapitel 4.10) übergeben (siehe **RecyclerView.java** Zeile 2 - 9).

Anschließend wird im Abschnitt des onCreateViewHolder() das erstellte XML - Layout in den Screen geladen.

Der ViewHolder() bildet wiederrum ein Skelett für einen unbefüllten Listview, welcher dann anschließend mit Daten befüllt und eingeblendet wird.

Die eigentlich beschriebene Funktionalität des RecyclerViews, die Datensätze nach und nach anhand deren Position das Skelett zu befüllen und in den View zu laden, wird anschließend mit der Funktion onBindViewHolder() umgesetzt. Hier wird anhand der Scrollposition der zugehörige Datensatz in der Ladestationen-Liste mit .get(position) geladen und per Holder als Listenitem in der Anzeige angehängt.

5.7 FAVORITEN

Eine weitere User Story ist die Möglichkeit Favoriten hinzuzufügen. Dadurch erkennt der Nutzer nicht nur seine persönlichen Lieblingsladestationen auf einen Blick, sondern hat zusätzlich noch die Möglichkeit die App zu seiner persönlichen zu machen. Jeder Nutzer kann seine eigenen Favoriten setzen, wodurch jeder auch seine App für sich gestaltet.

Der Button für das Hinzufügen der Favoriten wird bereits als „favouriteButton“ aus Codebeispiel 2 im RecyclerView (Zeile 42 - 46) implementiert. Gekennzeichnet ist dieser mit einem gelben Stern, um dem Nutzer auch ohne Beschreibung eine eindeutige Zuweisung zu geben.

Diese Einstellungen werden dann wie in Kapitel 4.10 beschrieben in einer App-internen SQLite Datenbank direkt auf dem Gerät gespeichert, sodass sie zu einem späteren App Start wieder zur Verfügung stehen.

5.8 DEFEKTE MELDEN

Damit kein Nutzer eine Ladestation anfährt, die gerade außer Betrieb ist, wird eine weitere Funktion implementiert, um defekte Ladestationen zu melden. Zusätzlich bietet dies die Möglichkeit, dass Service Techniker defekt gemeldete Stationen anfahren und reparieren können.

Es entsteht ein Netzwerk an Nutzern, von dem jeder Nutzer anhand der Aktivität anderer profitiert.

Wie auch der Favoriten Button, wird der Button für das Melden als defekt bereits als „reportButton“ aus Codebeispiel 2 im RecyclerView (Zeile 48 - 51) implementiert. Gekennzeichnet ist dieser mit einer roten Flagge, um den Nutzer auch ohne Beschreibung eine eindeutige Zuweisung zu geben.

Wird der Button gedrückt, erscheint ein Pop Up, welches den Nutzer auffordert einen Grund für den Defekt anzugeben. Diesen Grund kann der Service Mitarbeiter für seine Problemanalyse angezeigt bekommen und nutzen.

5.9 FILTER FUNKTIONEN

Eine ungeordnete Liste aller Ladestationen bringt dem individuellen Nutzer nicht viel. Er möchte nur bestimmte für den eigenen Zweck angezeigt bekommen. Die wichtigsten Kriterien hierbei sind:

- Seine persönlichen Favoriten angezeigt zu bekommen.
- Nur Ladestationen in einem bestimmten Umkreis darstellen.

Der RecyclerView soll also gefiltert werden. Mit einem Klick auf die Lupe am oberen rechten Bildschirmrand erhält der Nutzer ein Pop Up mit den verschiedenen Filtermöglichkeiten (siehe Abbildung 5).

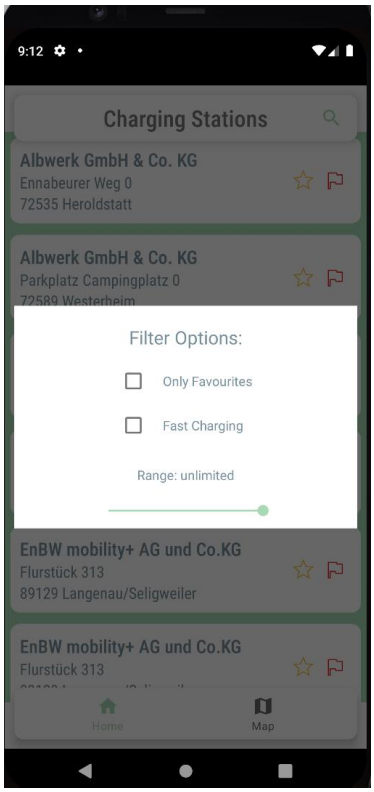


Abbildung 5: Filter Funktionen Pop Up

Die Favoriten und Schnelladesäulen lassen sich mit einer Checkbox anzeigen und der Umkreis lässt sich mit Hilfe eines Sliders (Seekbar) auf die gewünschten Kilometer einstellen. Dazu wird die bereits implementierte Umkreissuche aus Abschnitt 4.5 benutzt. Diese Filterfunktionen können auf die Liste beliebig kombiniert angewendet werden.

Um die Filterfunktionen als Pop-Up über dem eigentlichen Screen zu verwenden, wurde eine eigenes Layout und eine eigene Klasse **PopUpService.java** erstellt, die das Design dort drüber aufbaut. Zudem wurde im Zusammenspiel mit dieser Klasse eine weitere Klasse **SaveCheckedState.java** verwendet, um die getätigten Einstellungen in den Checkboxen und dem Slider vom Nutzer im Pop-Up auch nach schließen und öffnen dessen zu speichern.

Demnach sind in der **SaveCheckedState.java** lediglich drei Variablen (showFavourites, showFastCharging, searchRange) mit Gettern und Settern verwaltet. Diese Variablen speichern entsprechende Einstellungen im Pop-Up zwischen und werden im PopUpService bei jedem Aufruf entsprechend wieder hergestellt.

Die Klasse **PopUpService.java** schaut und verwaltet entsprechend die Filtereinstellungen und gleicht diese mit der SaveCheckedState Klasse in der Funktion showPopup() ab. Anschließend übergibt der Service an den RecyclerView eine neue Liste, die anhand der Filtereinstellungen neu gefiltert wurden. Dies passiert in der Funktion calculateThings().

```java
public class PopUpService {
    private final Context context;
    private StationAdapter stationAdapter;
    private AlertDialog dialog;
    private SaveCheckedState saveCheckedState;

    // set necessary vars within constructor

    /**
     * Constructor
     * @param context The context
     * @param stationAdapter the current station adapter
     * @param saveCheckedState the savestate
     */
    public PopUpService(Context context, StationAdapter stationAdapter, SaveCheckedState saveCheckedState) {
        this.context = context;
        this.stationAdapter = stationAdapter;
        this.saveCheckedState = saveCheckedState;
    }


    // calculate filtered stations from popup to display in recyclerview list

    /**
     * calculate filtered stations from popup to display in recyclerview list
     */
    private void calculateThings(){
        String favourites = this.context.getResources().getString(R.string.favourites);
        String fastCharging = this.context.getResources().getString(R.string.fastCharging);
        String range = this.context.getResources().getString(R.string.range);

        // Ini list of filtered stations and deleted stations (those will be removed from full list later on, because java does not allow remove in for)
        ArrayList<Ladestation> finalStations = new ArrayList<>();
        ArrayList<Ladestation> toDel = new ArrayList<>();

        // first add all stations to list
        finalStations.addAll(GlobalStorage.getAllStations());

        // check favourites filter
        if(saveCheckedState.isShowFavourites()){
            for(Ladestation ladestation : finalStations){
                if(!GlobalStorage.getFavStations().contains(ladestation.id)){
                    toDel.add(ladestation);
                }
            }
        }
        // check fastcharging filter
        if(saveCheckedState.isShowFastCharging()){
            for(Ladestation ladestation : finalStations){
                if(ladestation.getModuleType() != Ladestation.ModuleType.FAST_CHARGING){
                    if(!toDel.contains(ladestation)){
                        toDel.add(ladestation);
                    }
                }
            }
        }
        // check seekbar range filter
        if(saveCheckedState.getSearchRange() != 100){
            ArrayList<Ladestation> rangeStations = StationAPI.getProximityStations(final Stations, MainActivity.userLocation.latitude, MainActivity.userLocation.longitude, saveCheckedState.getSearchRange());
            for(Ladestation ladestation : finalStations){
                if(!rangeStations.contains(ladestation)){
                    toDel.add(ladestation);
                }
            }
        }

        // delete stations outside of filters in delarray from finalarry
        for(Ladestation ladestation : toDel){
            finalStations.remove(ladestation);
        }

        // clear delarray for next filtering
        toDel.clear();

        // set filtered finalarray as new list in listview
        stationAdapter.setNewList(finalStations);
    }


    /**
     * function for popup toggle and listeners
     */
    void showPopup() {
        String range = this.context.getResources().getString(R.string.range);
        String unlimited = this.context.getResources().getString(R.string.unlimited);

```

```
87.         // inflate popup above current listview
88.         AlertDialog.Builder builder = new AlertDialog.Builder(context);
89.         View view = LayoutInflater.from(context).inflate(R.layout.options_popup, null);
90.
91.         // check favourites checkbox
92.         CheckBox checkBoxFav = view.findViewById(R.id.FavToggleButton);
93.         if(saveCheckedState.isShowFavourites()){
94.             checkBoxFav.setChecked(true);
95.         }
96.         else {
97.             checkBoxFav.setChecked(false);
98.         }
99.
100.        // check fastcharging checkbox
101.        CheckBox checkBoxFastCharging = view.findViewById(R.id.FastChargingToggleButton);
102.        if(saveCheckedState.isShowFastCharging()){
103.            checkBoxFastCharging.setChecked(true);
104.        }
105.        else {
106.            checkBoxFastCharging.setChecked(false);
107.        }
108.
109.        // check and handle seekbar range filters
110.        SeekBar seekBar = view.findViewById(R.id.seekBar);
111.        TextView tvProgressLabel = view.findViewById(R.id.seekBarText);
112.        seekBar.setProgress(saveCheckedState.getSearchRange());
113.        if(seekBar.getProgress() == 100){
114.            tvProgressLabel.setText(range + ": " + unlimited);
115.        }
116.        else{
117.            tvProgressLabel.setText(range + ": " + seekBar.getProgress() + "km");
118.        }
119.
120.
121.        // set listeners in according sequence
122.        checkBoxFav.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
123.            @SuppressLint("NotifyDataSetChanged")
124.            @Override
125.            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
126.                saveCheckedState.setShowFavourites(isChecked);
127.                if(checkBoxFastCharging.isChecked()){
128.                    saveCheckedState.setShowFastCharging(true);
129.                }
130.                calculateThings();
131.            }
132.        });
133.        checkBoxFastCharging.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
134.            @Override
135.            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
136.                saveCheckedState.setShowFastCharging(isChecked);
137.                if(checkBoxFav.isChecked()){
138.                    saveCheckedState.setShowFavourites(true);
139.                }
140.                calculateThings();
141.            }
142.        });
143.        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
144.            @Override
145.            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
146.                if(seekBar.getProgress() == 100){
147.                    tvProgressLabel.setText(range + ": " + unlimited);
148.                }
149.                else{
150.                    tvProgressLabel.setText(range + ": " + seekBar.getProgress() + "km");
151.                }
152.            }
153.
154.            @Override
155.            public void onStartTrackingTouch(SeekBar seekBar) {}
156.
157.            @Override
158.            public void onStopTrackingTouch(SeekBar seekBar) {
159.                saveCheckedState.setSearchRange(seekBar.getProgress());
160.                calculateThings();
161.            }
162.        });
163.
164.        builder.setView(view);
165.        builder.setCancelable(true);
166.
167.        dialog = builder.show();
168.        dialog.getWindow().setBackgroundDrawableResource(R.color.transparent);
169.    }
170. }
```

Codebeispiel 6: PopUpService.java

5.10 PERSISTENTE DATEN

Bisher ist nach dem Beenden der App alles wieder auf Anfang gesetzt. Um auch nach erneutem Starten der App die Einstellungen zu Favoriten und Defekten Stationen zu erhalten, wird eine SQLite Datenbank über die Klasse **SQLiteHelper.java** implementiert. Dabei wird sich an den Codebeispielen aus dem Moodle-Kurs von Prof. Dr. Wiedling und der Android Studio Dokumentation [8] orientiert.

Die Datenbank speichert die ID's der Favoriten und defekten Stationen in jeweils eigenen Tabellen und lädt diese zu App Start aus. Hierfür wurde sowohl für Favoriten, als auch die Defekte ein trio an Funktionen implementiert:

- saveSingleFavoriteToDB(), saveSingleFlagToDB(): Speichert die ID der entsprechend als Favorit oder Defekt markierten Ladestation

```
1.      public void saveSingleFavoriteToDB(int id) // save single value
2.      {
3.          SQLiteDatabase database = this.getWritableDatabase();
4.          ContentValues values = new ContentValues();
5.          values.put(ATTRIBUTE_ID, id); // Write id to value set
6.          database.insert(FAVOURITE_TABLE_NAME, null, values); // Write the valueset to DB
7.          database.close();
8.      }
```

```
1.      public void saveSingleFlagToDB(int id) // save single value
2.      {
3.          SQLiteDatabase database = this.getWritableDatabase();
4.          ContentValues values = new ContentValues();
5.          values.put(ATTRIBUTE_ID, id); // Write id to value set
6.          database.insert(REPORTS_TABLE_NAME, null, values); // Write the valueset to DB
7.          database.close();
8.      }
```

Codebeispiel 7: Favoriten oder Defekte in Datenbank schreiben

- removeSingleFavoriteFromDB(), removeSingleReportFromDB(): Löscht die entsprechende ID der als Favorit oder Defekt entfernten Ladestation

```
1.      public void removeSingleFavoriteFromDB(int id)
2.      {
3.          SQLiteDatabase database = this.getWritableDatabase();
4.          database.delete(SQLiteHelper.FAVOURITE_TABLE_NAME, "id = ?",new String[]
          {String.valueOf(id)});
5.          database.close();
6.      }
```

```
1.      public void removeSingleReportFromDB(int id)
2.      {
3.          SQLiteDatabase database = this.getWritableDatabase();
4.          database.delete(SQLiteHelper.REPORTS_TABLE_NAME, "id = ?",new String[]
                  {String.valueOf(id)});
5.          database.close();
6.      }
```

Codebeispiel 8: Favoriten oder Defekte aus Datenbank entfernen

- readFavouritesFromDB(), readFlagsFromDB(): Liest alle IDs von entsprechenden Favoriten oder als Defekt markierten Ladestationen in der lokalen SQLite Datenbank aus und gibt diese als Array zur Verarbeitung der Instanz zurück.

```
1.        public ArrayList<Integer> readFavouritesFromDB() {
2.          SQLiteDatabase database = this.getReadableDatabase(); // Get reference to db
3.          String query = "SELECT * FROM "+SQLiteHelper.FAVOURITE_TABLE_NAME;
4.          Cursor res = database.rawQuery(query,null);
5.          ArrayList<Integer> resultList = new ArrayList<Integer>();
6.
7.          if(res.moveToFirst())
8.          {
9.              do // Loop over all results
10.             {
11.                 resultList.add(Integer.parseInt(res.getString(0)));
12.             } while(res.moveToNext());
13.         }
14.         res.close();
15.         database.close();
16.         for(Integer i : resultList)
17.         {
18.             System.out.println(i);
19.         }
20.         return resultList;
21.     }
```

```
1.          public ArrayList<Integer> readFlagsFromDB() {
2.          SQLiteDatabase database = this.getReadableDatabase(); // Get reference to db
3.          String query = "SELECT * FROM "+SQLiteHelper.REPORTS_TABLE_NAME;
4.          Cursor res = database.rawQuery(query,null);
5.          ArrayList<Integer> resultList = new ArrayList<Integer>();
6.
7.          if(res.moveToFirst())
8.          {
9.              do // Loop over all results
10.             {
11.                 resultList.add(Integer.parseInt(res.getString(0)));
12.             }while(res.moveToNext());
13.         }
14.         res.close();
15.         database.close();
16.         for(Integer i : resultList)
17.         {
18.             System.out.println(i);
19.         }
20.         return resultList;
21.     }
```

Codebeispiel 9: Favoriten oder Defekte aus Datenbank lesen

5.11 LOGIN METHODEN

Nicht nur „normale“ Nutzer, sondern auch Service Techniker sollen Zugang zu der App haben. Da Service Techniker jedoch andere Rechte haben als Nutzer, benötigt man verschiedene Logins. Der Service Techniker erhält einen zusätzlichen Screen, auf dem *nur* defekte Ladestationen angezeigt werden. Zusätzlich hat er die Möglichkeit, sich den Grund für den Defekt anzuschauen und nach erfolgreicher Reparatur die Ladestation wieder freizugeben.

Umgesetzt wurde die Benutzerunterscheidung über ein eigenes Loginfragment, welches beim Appstart vor allen anderen Screens aufgerufen und angezeigt wird.

Je nach gewähltem Nutzer wird über die Klasse **LoginFragment.java** das Layout der Bottom-Navigation um den Punkt des Service Screens ergänzt oder dieser ausgeblendet, sodass er vom normalen Nutzer nicht angesteuert wird. Hierfür wurden im Menü zwei Layouts der Bottom-Navigation angelegt: **bottom_navigation.xml** und **bottom_navigation_worker.xml**.

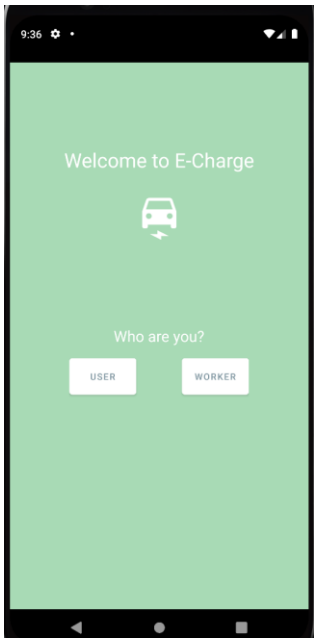


Abbildung 6: Login Screen

```java
public class LoginFragment extends Fragment {

    /**
     * @param inflater
     * @param container
     * @param savedInstanceState
     * @return view
     */
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_login, container, false);

        // USER
        Button userButton = view.findViewById(R.id.UserButton);
        userButton.setOnClickListener(v -> {
            MainActivity.getBottomNav().inflateMenu(R.menu.bottom_navigation);
            getParentFragmentManager().beginTransaction().replace(R.id.fragment_container, new MapFragment()).commit();
            MainActivity.getBottomNav().setSelectedItemId(R.id.nav_map);
            MainActivity.getBottomNav().setVisibility(View.VISIBLE);
        });

        // WORKER
        Button workerButton = view.findViewById(R.id.WorkerButton);
        workerButton.setOnClickListener(v -> {
            MainActivity.getBottomNav().inflateMenu(R.menu.bottom_navigation_worker);
            getParentFragmentManager().beginTransaction().replace(R.id.fragment_container, new MapFragment()).commit();

            MainActivity.getBottomNav().setSelectedItemId(R.id.nav_map);
            MainActivity.getBottomNav().setVisibility(View.VISIBLE);
        });

        return view;
    }
}
```

Codebeispiel 10: LoginFragment.java

5.12 MEHRSPRACHIGKEIT

Eine weitere gestellte Projektanforderung ist die Mehrsprachigkeit. Somit haben Nutzer die Möglichkeit die App in ihrer eingestellten Systemsprache zu verwenden. Realisiert wird dies mit **string.xml** Dateien für die jeweilige Sprache. Dazu wird zunächst jeder Text, der auf dem Bildschirm angezeigt werden kann als String Variable deklariert. Anschließend wird jeder Begriff übersetzt und in einer Tabelle abgespeichert. Bei App Start wird die Tabelle der eingestellten Systemsprache verwendet und jeder Text auf dem Bildschirm in der Systemsprache angezeigt. Ein Auszug davon ist in Abbildung 7 zu sehen.

Key	Resource Folder	Untranslatable	Default Value	German (de) in Germany (DE)
filter_options	app/src/main/res	☐	Filter Options:	Filteroptionen:
charging_stations	app/src/main/res	☐	Charging Stations	Ladestationen
welcome_to_e_charge	app/src/main/res	☐	Welcome to E-Charge	Willkommen bei E-Charge
please_select	app/src/main/res	☐	Who are you?	Wer bist du?
user	app/src/main/res	☐	User	Benutzer
worker	app/src/main/res	☐	Worker	Techniker
only_favourites	app/src/main/res	☐	Only Favourites	Nur Favoriten
fast_charging	app/src/main/res	☐	Fast Charging	Nur Schnellladen
range_unlimited	app/src/main/res	☐	Range: unlimited	Umkreis: unbegrenzt
home	app/src/main/res	☐	Home	Start
map	app/src/main/res	☐	Map	Karte
service	app/src/main/res	☐	Service	Service
errLocation	app/src/main/res	☐	Please allow location	Bitte Standort zulassen!
broken	app/src/main/res	☐	Out of order	Defekt
favouriteDefect	app/src/main/res	☐	Favourite (defect)	Favorit (defekt)
favourite	app/src/main/res	☐	Favourite	Favorit
station	app/src/main/res	☐	station	Station
favourites	app/src/main/res	☐	Favourites	Favoriten
fastCharging	app/src/main/res	☐	Fast Charging	Schnelladen
range	app/src/main/res	☐	Range	Distanz
unlimited	app/src/main/res	☐	unlimited	unbegrenzt

Abbildung 7: Tabelle Übersetzungen

Ein Beispiel zur Verwendung der **string.xml** zur Generierung von Texten der entsprechenden Sprache können in der PopUpService Klasse eingesehen werden:

```
1.  String range = this.context.getResources().getString(R.string.range);
2.  String unlimited = this.context.getResources().getString(R.string.unlimited);
```

```
1.  if(seekBar.getProgress() == 100){
2.      tvProgressLabel.setText(range + ": " + unlimited);
3.  }
4.  else{
5.      tvProgressLabel.setText(range + ": " + seekBar.getProgress() + "km");
6.  }
```

Codebeispiel 11: Multilanguage Support Strings

Das Hinzufügen von weiteren Sprachen ist somit kein großer Mehraufwand. Es wird lediglich eine weitere Spalte mit der jeweiligen Sprache angelegt und die Übersetzungen in die Zeilen eingetragen.

5.13 LANDSCAPE MODUS

Der Landscape Modus ist ein Feature, das dem Nutzer mehr Möglichkeiten und Freiraum bietet, die App nach seinen persönlichen Präferenzen zu nutzen. Beim Drehen des Endgerätes schaltet die App automatisch in den Landscape Modus, sodass auch beim seitlichen Halten des Smartphones alle Elemente weiterhin auf dem Screen angezeigt werden und die App voll benutzbar bleibt.

Für die Implementierung wird ein Parameter in **AndroidManifest.xml** benötigt (Codebeispiel 3 Zeile 27):

```
1.   <?xml version="1.0" encoding="utf-8"?>
2.  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.      package="com.example.myapplication">
4.
5.      <uses-permission android:name="android.permission.INTERNET"/>
6.      <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
7.      <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
8.      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
9.      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
10.
11.     <application
12.         android:allowBackup="true"
13.         android:icon="@mipmap/ic_launcher"
14.         android:label="E-Charge"
15.         android:roundIcon="@mipmap/ic_launcher_round"
16.         android:supportsRtl="true"
17.         android:theme="@style/Theme.MyApplication"
18.         >
19.
20.         <meta-data
21.             android:name="com.google.android.geo.API_KEY"
22.             android:value="AIzaSyBtXnHz1RzepQ0B6tg4bbBP27UIl5aEgaw"/>
23.
24.         <activity
25.             android:name=".MainActivity"
26.             android:exported="true"
27.             android:screenOrientation="sensor">
28.             <intent-filter>
29.                 <action android:name="android.intent.action.MAIN" />
30.
31.                 <category android:name="android.intent.category.LAUNCHER" />
32.             </intent-filter>
33.         </activity>
34.         <service
35.             android:name=".RequestService"
36.             android:exported="false" />
37.     </application>
38. </manifest>
```

Codebeispiel 12: AndroidManifest.xml Screen Orientation

Zusätzlich müssen für die verschiedenen Screens Landscape Varianten erstellt werden (siehe Abbildung 8).

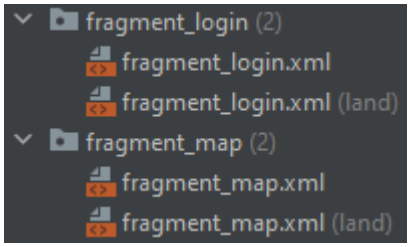


Abbildung 8: Landscape Screens

Anschließend wählt die App automatisch das passende Layout je Drehrichtung des Smartphones.

5.14 FEINSCHLIFF

Nach Implementierung der geforderten und festgelegten Features, wird die App mehreren Personen, die nicht an der Entwicklung beteiligt sind, zum Testen gegeben. Dafür werden den Personen die folgenden Aufgaben gegeben (als Nutzer):

- Logge dich als Nutzer ein.
- Finde Ladestationen in einem beliebigen Umkreis zu dir.
- Finde Ladestationen auf der Karte in einem Umkreis zu dir.
- Füge einen oder mehrere Favoriten hinzu.
- Melde eine Ladestation als defekt und gebe einen Grund an.

Und als Service Techniker:

- Logge dich als Service Techniker ein.
- Finde Ladestationen, die als defekt gemeldet wurden.
- Lasse dir nur defekte Ladestationen in deinem Umkreis anzeigen.
- Sehe dir den Grund für einen Defekt an.
- Gebe die Ladestation wieder frei.

Die Nutzer konnten zu großem Teil alle ihnen gestellten Aufgaben erledigen. Speziell der Login und das Filter Pop-Up über Lupen Icon wurden über die Aussagekräftigen Icons von allen Personen schnell gefunden.

Weiterhin konnten Nutzer im RecyclerView gut nachvollziehen, welche Icons für Favoriten- und Defekt-Markierungen zuständig sind und diese Einstellungen entsprechend setzen und filtern.

Ein Problem, welches sich im RecyclerView stellte war die ebenfalls von uns implementierte „Neu-Laden“-Funktion, wenn man über den obersten Eintrag der Liste hinausscrollt. Das Neu-Laden Icon, welches auftaucht, steht nämlich ohne Animation still, was dem Nutzer den Eindruck eines Absturzes vermittelt. Dieses Problem lässt sich beheben, indem man die „Neu-Laden“-Funktion in einen eigenen asynchronen Thread verlagert.

Ansonsten war der RecyclerView für alle Nutzer einleuchtend und gut zu bedienen.

Eine weitere Problematik ergab sich aus der Bedienung der Google Karte, da ein Zoom zu nah an die Karte auf den eigenen Standort eventuell darin resultiert, dass nach Klick zum Anzeigen von Ladestationen im eigenen Umfeld keine Ladestation nah genug ist, als dass sich auf der Karte etwas verändert und Stationen angezeigt werden. Hierfür könnte man als Lösung eine Funktion zum Rauszoomen implementieren, sollten keine Stationen im eigenen View liegen.

Zudem sollte es möglich sein über Anwählen einer Ladestationen diese direkt auf der Karte als Favorit oder Defekt markieren zu können.

Ein weiterer Feature-Wunsch, der sich implementieren lässt, war, dass Nutzer eine Verlinkung zur Routenführung über Google Maps von der ausgewählten Station erhalten.

Anhand der Weitergabe unserer App hatten wir uns also in der Orientierung und Umsetzung der User Stories bestätigt gefühlt, jedoch auch festgestellt, dass durch die Individualität der Nutzer immer neue Featureanfragen entstehen, sowie Weiterentwicklungen und Verbesserungen möglich sind.

6. AUSBLICK

Beim Testen der App mit den Probanten wurde deutlich, dass die GoogleMap noch intensiver in die App eingebunden werden sollte. Insbesondere das Feature „Favoriten setzen“ auf der GoogleMap wurde vermisst.

Wie bereits in Abschnitt 4.14 aufgegriffen, sollte der Zoom automatisch angepasst werden, sobald der Nutzer zu nah herangezoomt hat, dass keine Ladestationen im Umkreis mehr sichtbar sind.

Die „Neu-Laden“-Funktion des RecyclerViews sollte künftig in einem asynchronen Thread verlagert werden, um nicht den Anschein zu erwecken, dass die App abgestürzt ist.

Es könnten weitere Filtermöglichkeiten, wie zum Beispiel Steckertyp, Hersteller etc. hinzugefügt werden, um die Erfahrung des Nutzers weiter zu verbessern. Durch die bereits vorhandenen Daten lässt sich dieses Feature leicht integrieren.

Des Weiteren lässt sich mit wenigen Handgriffen der Login verbessern. Da die Datenbank bereits angebunden wurde lassen sich Nutzerdaten verschlüsselt darin speichern, sodass jeder Nutzer einen eigenen Zugang hat und Service Techniker noch besser von normalen Nutzern getrennt sind.

7. FAZIT

Der Umgang mit Java und Android Studio war für uns alle zunächst neu. Daher wurden die ersten Wochen vor allem für die Grundlagen in der IDE genutzt. Das Verwenden von XML-Dateien, die Implementierung von Adaptern, Listenern, Layouts etc. war ungewohnt, wurde aber mit Hilfe des Skiptes von Prof. Dr. Wiedling, Internetquellen und YouTube-Videos verstanden. Sobald die Grundfunktionen verstanden wurden, ist der Workflow deutlich angenehmer, wodurch auch Umsetzung und Planung der gewünschten Funktionen besser werden.

Sowohl beim Design als auch bei Umsetzung der Projektanforderungen standen immer die User Stories im Vordergrund. So haben wir auf ein aufwendiges, verschachteltes Design verzichtet, um auch ohne Beschreibungen dem Nutzer die Möglichkeit zu geben alle Funktionen einfach zu finden und keine Fragen offenbleiben. Alle Buttons wurden entsprechend so gestaltet, dass sie ohne Erklärung aussagekräftig und eindeutig sind.

Alles in allem war es ein spannendes Projekt, bei dem wir viel gelernt haben. Generell die Gelegenheit an einem größeren Projekt, das über das gesamte Semester geht, zu arbeiten, gefiel uns sehr gut. Wir konnten uns in der Gruppe selbst organisieren und mit verschiedenen Tools wie Discord und GitLab super verständigen.

8. QUELLEN

[1] https://www.youtube.com/watch?v=tPV8xA7m-iw

[2] https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView

[3] https://www.geeksforgeeks.org/difference-between-a-fragment-and-an-activity-in-android/

[4] https://www.bundesnetzagentur.de/SharedDocs/Downloads/DE/Sachgebiete/Energie/Unternehmen_Institutionen/E_Mobilitaet/Ladesaeulenregister_CSV.csv?__blob=publicationFile&v=27

[5] https://api.aurora-theogenia.de/chargingstations/charging_stations.json

[6] http://www.movable-type.co.uk/scripts/latlong.html

[7] https://developer.android.com/guide/topics/ui/layout/recyclerview

[8] https://developer.android.com/training/data-storage/sqlite

[9] Absprachen mit Prof. Dr. Wiedling und Herr Quick

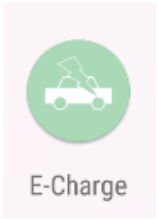
A.1 Eigenständigkeitserklärung:

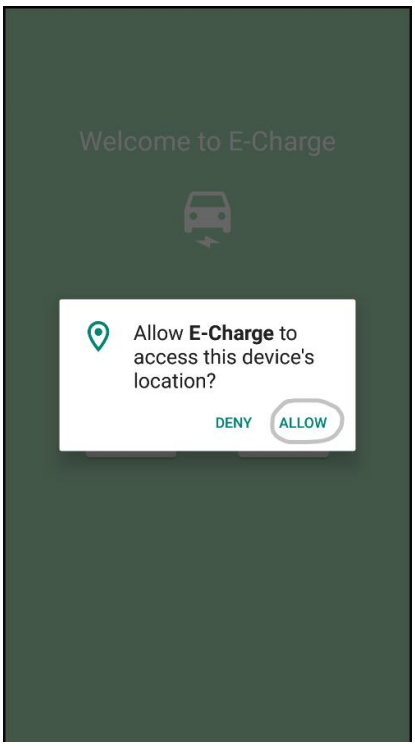
„Wir versichern hiermit, dass wir die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt haben.

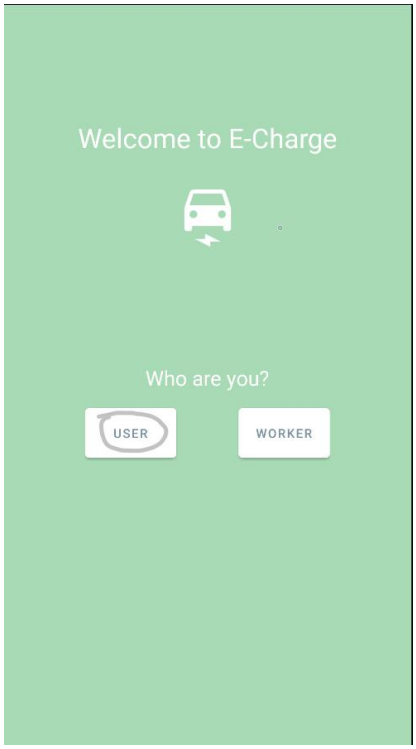
Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht.

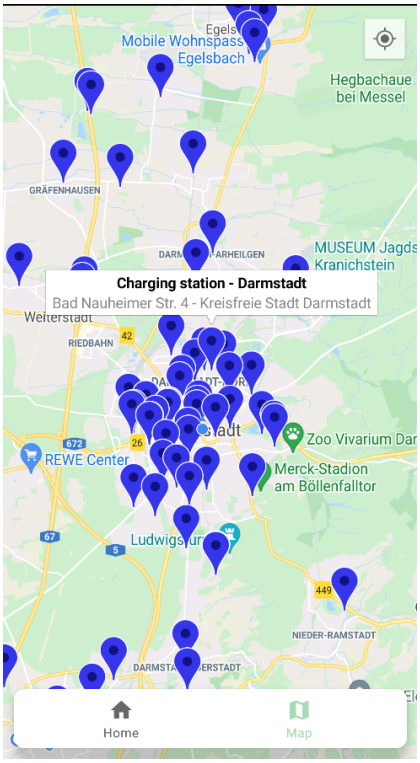
Die Zeichnungen oder Abbildungen in dieser Arbeit sind von uns selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.“

A.2 App Durchlauf anhand Bilderreihe



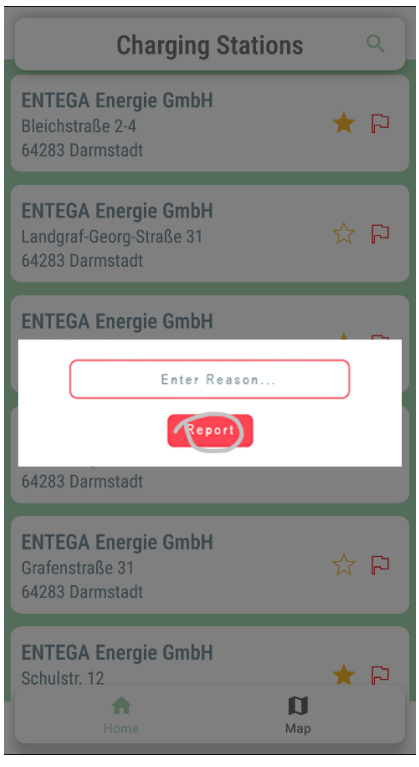


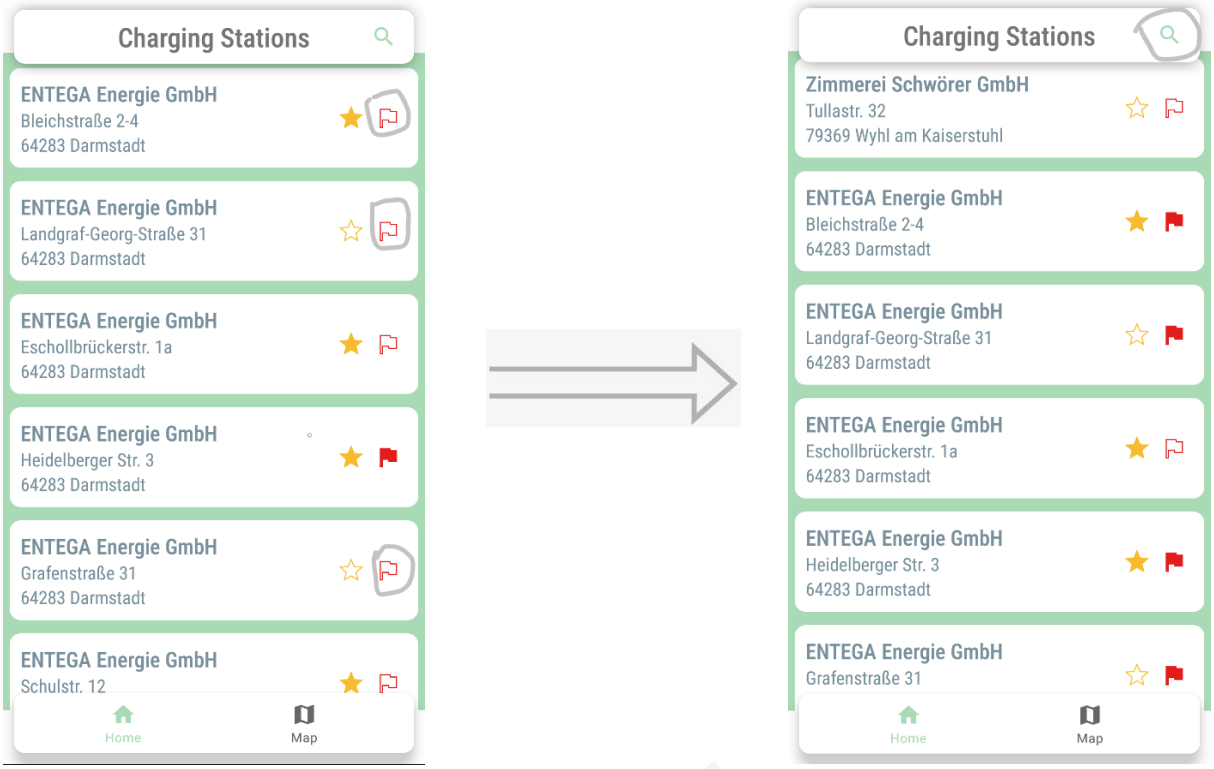


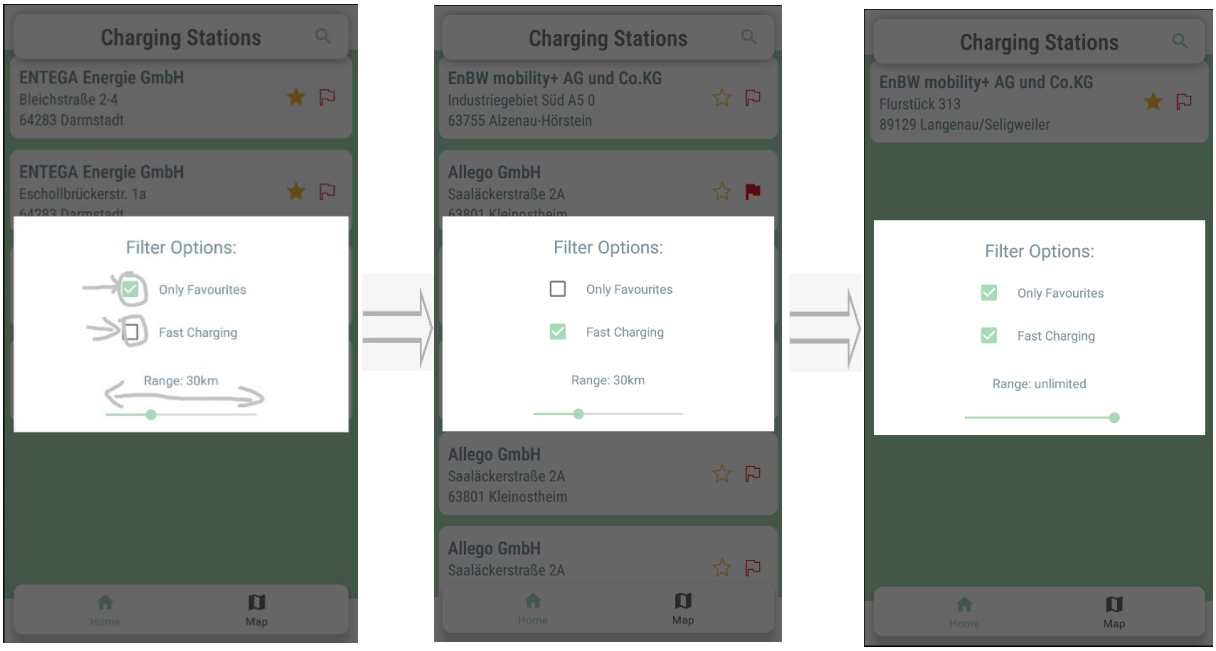

Mobile Wohnspass Egelsbach
Hegbachaue bei Messel
GRÄFENHAUSEN
MUSEUM Jagds Kranichstein
Charging station - Darmstadt
Bad Nauheimer Str. 4 - Kreisfreie Stadt Darmstadt
RIEDBAHN
Zoo Vivarium Dar
REWE Center
Merck-Stadion am Böllenfalltor
NIEDER-RAMSTADT
Home
Map

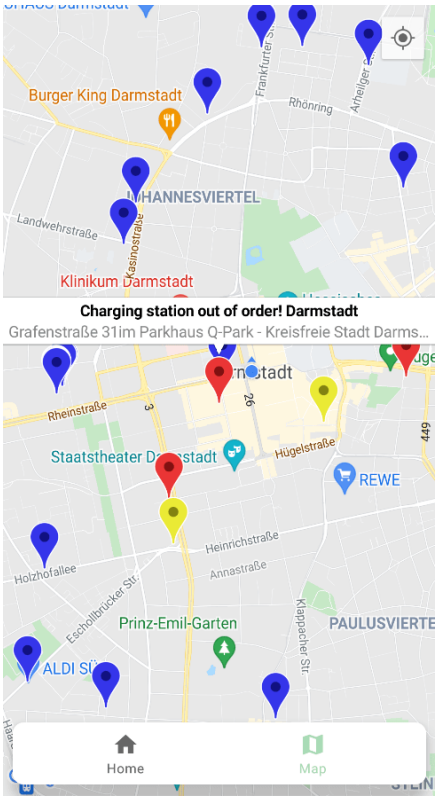

Charging Stations
ENTEGA Energie GmbH
Bleichstraße 2-4
64283 Darmstadt
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Eschollbrückerstr. 1a
64283 Darmstadt
ENTEGA Energie GmbH
Heidelberger Str. 3
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
64283 Darmstadt
ENTEGA Energie GmbH
Schulstr. 12
Home
Map

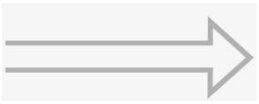

Charging Stations
ENTEGA Energie GmbH
Bleichstraße 2-4
64283 Darmstadt
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Eschollbrückerstr. 1a
64283 Darmstadt
ENTEGA Energie GmbH
Heidelberger Str. 3
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
64283 Darmstadt
ENTEGA Energie GmbH
Schulstr. 12
Home
Map

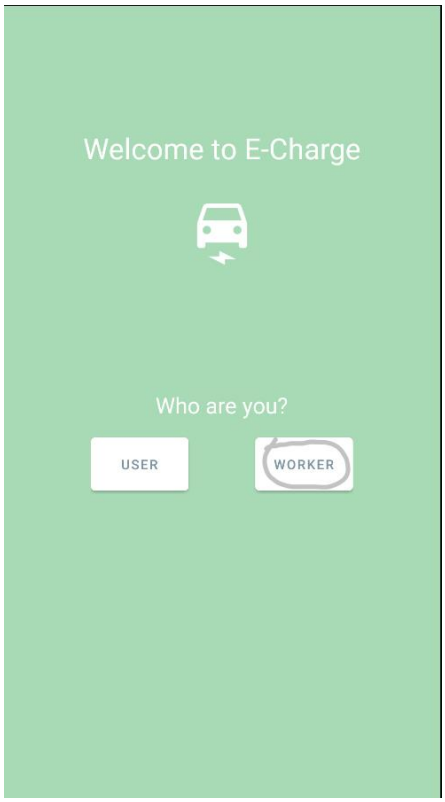

Charging Stations
ENTEGA Energie GmbH
Bleichstraße 2-4
64283 Darmstadt
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Enter Reason...
Report
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
64283 Darmstadt
ENTEGA Energie GmbH
Schulstr. 12
Home
Map

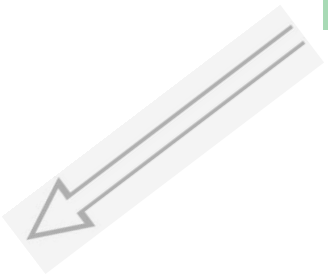

Charging Stations
ENTEGA Energie GmbH
Bleichstraße 2-4
64283 Darmstadt
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Eschollbrückerstr. 1a
64283 Darmstadt
ENTEGA Energie GmbH
Heidelberger Str. 3
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
64283 Darmstadt
ENTEGA Energie GmbH
Schulstr. 12
Home
Map
Charging Stations
Zimmerei Schwörer GmbH
Tullastr. 32
79369 Wyhl am Kaiserstuhl
ENTEGA Energie GmbH
Bleichstraße 2-4
64283 Darmstadt
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Eschollbrückerstr. 1a
64283 Darmstadt
ENTEGA Energie GmbH
Heidelberger Str. 3
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
Home
Map


Charging Stations
ENTEGA Energie GmbH
Bleichstraße 2-4
64283 Darmstadt
ENTEGA Energie GmbH
Eschollbrückerstr. 1a
Filter Options:
Only Favourites
Fast Charging
Range: 30km
Home
Map
Charging Stations
EnBW mobility+ AG und Co.KG
Industriegebiet Süd A5 0
63755 Alzenau-Hörstein
Allego GmbH
Saaläckerstraße 2A
Filter Options:
Only Favourites
Fast Charging
Range: 30km
Allego GmbH
Saaläckerstraße 2A
63801 Kleinostheim
Allego GmbH
Saaläckerstraße 2A
Home
Map
Charging Stations
EnBW mobility+ AG und Co.KG
Flurstück 313
89129 Langenau/Seligweiler
Filter Options:
Only Favourites
Fast Charging
Range: unlimited
Home
Map

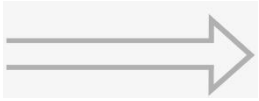

Burger King Darmstadt
JOHANNESVIERTEL
Klinikum Darmstadt
Charging station out of order! Darmstadt
Grafenstraße 31im Parkhaus Q-Park - Kreisfreie Stadt Darms...
Staatstheater Darmstadt
REWE
Prinz-Emil-Garten
PAULUSVIERTE
Home
Map

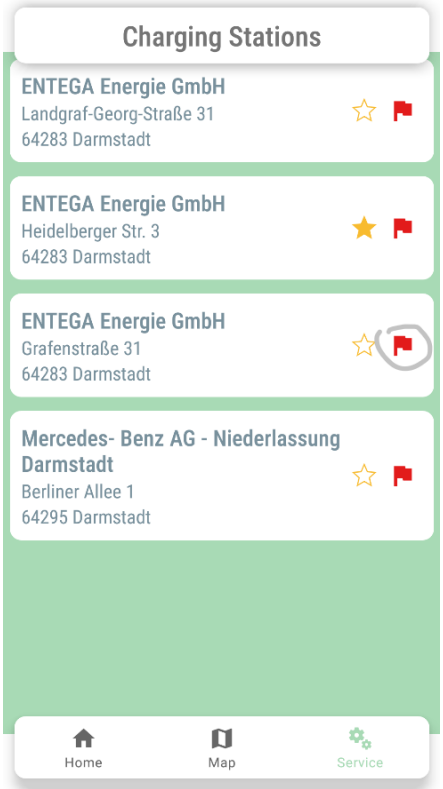


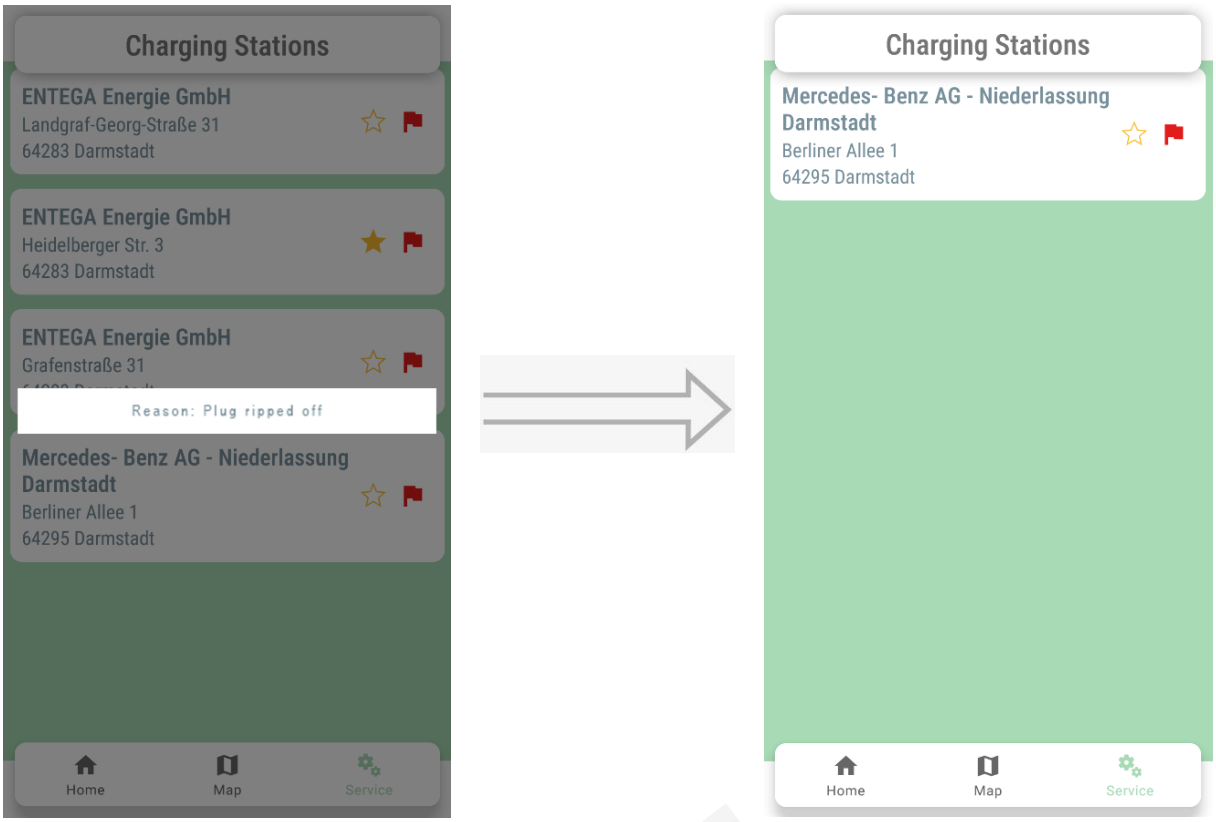

Welcome to E-Charge
Who are you?
USER
WORKER

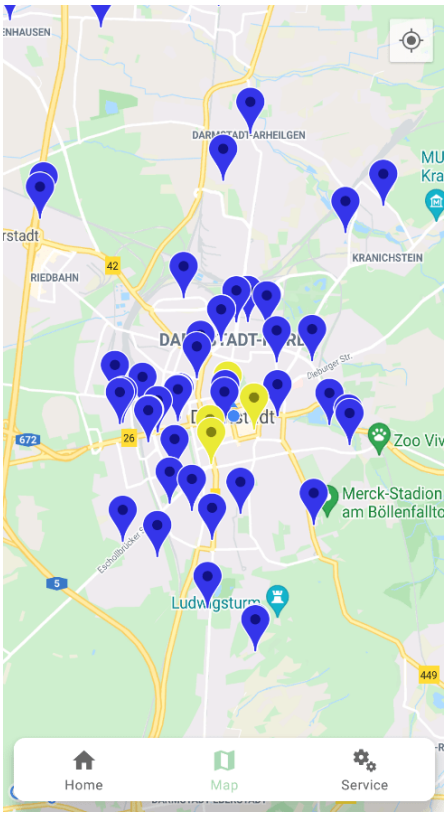



JOHANNESVIERTEL
Herrngarten
Landgericht Darmstadt
Darmstadt
TK Maxx
SATURN
Staatstheater Darmstadt
REWE
Home
Map
Service




Charging Stations
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Heidelberger Str. 3
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
64283 Darmstadt
Mercedes- Benz AG - Niederlassung Darmstadt
Berliner Allee 1
64295 Darmstadt
Home
Map
Service


Charging Stations
ENTEGA Energie GmbH
Landgraf-Georg-Straße 31
64283 Darmstadt
ENTEGA Energie GmbH
Heidelberger Str. 3
64283 Darmstadt
ENTEGA Energie GmbH
Grafenstraße 31
Reason: Plug ripped off
Mercedes- Benz AG - Niederlassung Darmstadt
Berliner Allee 1
64295 Darmstadt
Home
Map
Service
Charging Stations
Mercedes- Benz AG - Niederlassung Darmstadt
Berliner Allee 1
64295 Darmstadt
Home
Map
Service


DARMSTADT-ARHEILGEN
KRANICHSTEIN
RIEDBAHN
Zoo Viv
Merck-Stadion am Böllenfalltc
Ludwigsturm
Home
Map
Service

A.3 Doxygen Java Dokumenation

E Charge

Generated by Doxygen 1.9.3

Chapter 1

Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 Package com.example.myapplication

Classes

- class BuildConfig
- class DownloadThread
- class ExampleUnitTest
- class GlobalStorage
- class HomeFragment
- class Ladestation
- class LoginFragment
- class MainActivity
- class MapFragment
- class PopUpService
- class SaveCheckedState
- class ServiceFragment
- class SQLiteHelper
- class StationAdapter
- class StationAPI

5.1.1 Detailed Description

Automatically generated file. DO NOT MODIFY

Chapter 6

Class Documentation

6.1 com.example.myapplication.BuildConfig Class Reference

Static Public Attributes

- static final boolean DEBUG = Boolean.parseBoolean("true")
- static final String APPLICATION_ID = "com.example.myapplication"
- static final String BUILD_TYPE = "debug"
- static final int VERSION_CODE = 1
- static final String VERSION_NAME = "1.0"

6.1.1 Member Data Documentation

6.1.1.1 APPLICATION_ID

```
static final String com.example.myapplication.BuildConfig.APPLICATION_ID = "com.example.↩
myapplication"  [static]
```

6.1.1.2 BUILD_TYPE

```
static final String com.example.myapplication.BuildConfig.BUILD_TYPE = "debug"  [static]
```

6.1.1.3 DEBUG

```
static final boolean com.example.myapplication.BuildConfig.DEBUG = Boolean.parseBoolean("true")
[static]
```

6.1.1.4 VERSION_CODE

```
static final int com.example.myapplication.BuildConfig.VERSION_CODE = 1  [static]
```

6.1.1.5 VERSION_NAME

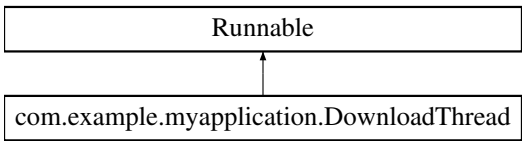
```
static final String com.example.myapplication.BuildConfig.VERSION_NAME = "1.0"  [static]
```

The documentation for this class was generated from the following files:

- app/build/generated/source/buildConfig/debug/com/example/myapplication/BuildConfig.java
- app/build/generated/source/buildConfig/release/com/example/myapplication/BuildConfig.java

6.2 com.example.myapplication.DownloadThread Class Reference

Inheritance diagram for com.example.myapplication.DownloadThread:



Public Member Functions

- void run ()
- ArrayList< Ladestation > getStations ()

6.2.1 Member Function Documentation

6.2.1.1 getStations()

```
ArrayList< Ladestation > com.example.myapplication.DownloadThread.getStations ( )
```

Getter for all stations

Returns

the stations ArrayList

6.2.1.2 run()

```
void com.example.myapplication.DownloadThread.run ( )
```

Overrides the Runnable's run method

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/DownloadThread.java

6.3 com.example.myapplication.ExampleUnitTest Class Reference

Public Member Functions

- void addition_isCorrect ()

6.3.1 Detailed Description

Example local unit test, which will execute on the development machine (host).

See also

Testing documentation

6.3.2 Member Function Documentation

6.3.2.1 addition_isCorrect()

```
void com.example.myapplication.ExampleUnitTest.addition_isCorrect ( )
```

The documentation for this class was generated from the following file:

- app/src/test/java/com/example/myapplication/ExampleUnitTest.java

6.4 com.example.myapplication.GlobalStorage Class Reference

Static Public Member Functions

- static ArrayList< Ladestation > getAllStations ()
- static void setAllStations (ArrayList< Ladestation > allStations)
- static ArrayList< Integer > getFavStations ()
- static void setFavStations (ArrayList< Integer > favStations)
- static ArrayList< Integer > getDefStations ()
- static void setDefStations (ArrayList< Integer > defStations)
- static SaveCheckedState getSaveCheckedState ()
- static void setSaveCheckedState (SaveCheckedState saveCheckedState)

6.4.1 Member Function Documentation

6.4.1.1 getAllStations()

```
static ArrayList< Ladestation > com.example.myapplication.GlobalStorage.getAllStations ( )
[static]
```

Returns allStations

Returns

The allStations Array

6.4.1.2 getDefStations()

```
static ArrayList< Integer > com.example.myapplication.GlobalStorage.getDefStations ( )  [static]
```

Returns the defect stations as ArrayList

Returns

the defectStations ArrayList

6.4.1.3 getFavStations()

```
static ArrayList< Integer > com.example.myapplication.GlobalStorage.getFavStations ( )  [static]
```

Returns all favorite stations

Returns

The favoriteStation ArrayList

6.4.1.4 getSaveCheckedState()

```
static SaveCheckedState com.example.myapplication.GlobalStorage.getSaveCheckedState ( )  [static]
```

Returns the SaveCheckedState

Returns

saveCheckedState

6.4.1.5 setAllStations()

```
static void com.example.myapplication.GlobalStorage.setAllStations (
            ArrayList< Ladestation > allStations )  [static]
```

Set all stations in global storage

Parameters

allStations	The ArrayList to set the stations in Global Storage to

6.4.1.6 setDefStations()

```
static void com.example.myapplication.GlobalStorage.setDefStations (
            ArrayList< Integer > defStations )  [static]
```

6.4.1.7 setFavStations()

```
static void com.example.myapplication.GlobalStorage.setFavStations (
            ArrayList< Integer > favStations )  [static]
```

Sets the favorite stations in Global Storage to the passed ArrayList

Parameters

favStations	the ArrayList to set the Global Storage's array list to.

6.4.1.8 setSaveCheckedState()

```
static void com.example.myapplication.GlobalStorage.setSaveCheckedState (
            SaveCheckedState saveCheckedState )  [static]
```

Sets the GlobalStorage's SaveCheckedState

Parameters

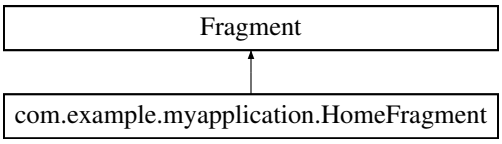
saveCheckedState	the state to set the checkedState to

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/GlobalStorage.java

6.5 com.example.myapplication.HomeFragment Class Reference

Inheritance diagram for com.example.myapplication.HomeFragment:



Public Member Functions

- View onCreateView (@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)

6.5.1 Member Function Documentation

6.5.1.1 onCreateView()

```
View com.example.myapplication.HomeFragment.onCreateView (
            @NonNull LayoutInflater inflater,
            @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState )
```

Parameters

inflater	The inflater inflates
container	The container contains
savedInstanceState	The savedInstanceState saves the Instance state

Returns

Returns the View

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/HomeFragment.java

6.6 com.example.myapplication.Ladestation Class Reference

Classes

- enum ModuleType
- enum PlugType

Public Member Functions

- String getOperator ()
- String getStreet ()
- String getNumber ()
- String getAdditional ()
- int getPostalCode ()
- String getLocation ()
- String getState ()
- String getArea ()
- float getLat ()
- float getLon ()
- String getInstallationDate ()
- float getConnPower ()
- ModuleType getModuleType ()
- int getNumberOfConnections ()
- List< PlugType > getPlugTypes_1 ()
- float getPower_1 ()
- String getPublicKey_1 ()
- List< PlugType > getPlugTypes_2 ()
- float getPower_2 ()
- String getPublicKey_2 ()
- List< PlugType > getPlugTypes_3 ()
- float getPower_3 ()
- String getPublicKey_3 ()
- List< PlugType > getPlugTypes_4 ()
- float getPower_4 ()
- String getPublicKey_4 ()

Public Attributes

- int id

6.6.1 Member Function Documentation

6.6.1.1 getAdditional()

```
String com.example.myapplication.Ladestation.getAdditional ( )
```

Getter for additional

Returns

additional

6.6.1.2 getArea()

```
String com.example.myapplication.Ladestation.getArea ( )
```

Getter for area

Returns

area

6.6.1.3 getConnPower()

```
float com.example.myapplication.Ladestation.getConnPower ( )
```

Getter for conn_power

Returns

conn_power

6.6.1.4 getInstallationDate()

```
String com.example.myapplication.Ladestation.getInstallationDate ( )
```

Getter for installation_date

Returns

installation_date

6.6.1.5 getLat()

```
float com.example.myapplication.Ladestation.getLat ( )
```

Getter for lat

Returns

lat

6.6.1.6 getLocation()

```
String com.example.myapplication.Ladestation.getLocation ( )
```

Getter for location

Returns

location

6.6.1.7 getLon()

```
float com.example.myapplication.Ladestation.getLon ( )
```

Getter for lon

Returns

lon

6.6.1.8 getModuleType()

```
ModuleType com.example.myapplication.Ladestation.getModuleType ( )
```

Getter for module_type

Returns

module_type

6.6.1.9 getNumber()

```
String com.example.myapplication.Ladestation.getNumber ( )
```

Getter for number

Returns

number

6.6.1.10 getNumberOfConnections()

```
int com.example.myapplication.Ladestation.getNumberOfConnections ( )
```

Getter for number_of_connections

Returns

number_of_connections

6.6.1.11 getOperator()

```
String com.example.myapplication.Ladestation.getOperator ( )
```

Getter for operator

Returns

operator

6.6.1.12 getPlugTypes_1()

```
List< PlugType > com.example.myapplication.Ladestation.getPlugTypes_1 ( )
```

Getter for plug_types_1

Returns

plug_types_1

6.6.1.13 getPlugTypes_2()

```
List< PlugType > com.example.myapplication.Ladestation.getPlugTypes_2 ( )
```

Getter for plug_types_2

Returns

plug_types_2

6.6.1.14 getPlugTypes_3()

```
List< PlugType > com.example.myapplication.Ladestation.getPlugTypes_3 ( )
```

Getter for plug_types_3

Returns

plug_types_3

6.6.1.15 getPlugTypes_4()

```
List< PlugType > com.example.myapplication.Ladestation.getPlugTypes_4 ( )
```

Getter for plug_types_4

Returns

plug_types_4

6.6.1.16 getPostalCode()

```
int com.example.myapplication.Ladestation.getPostalCode ( )
```

Getter for postal_code

Returns

postal_code

6.6.1.17 getPower_1()

```
float com.example.myapplication.Ladestation.getPower_1 ( )
```

Getter for power_1

Returns

power_1

6.6.1.18 getPower_2()

```
float com.example.myapplication.Ladestation.getPower_2 ( )
```

Getter for power_2

Returns

power_2

6.6.1.19 getPower_3()

```
float com.example.myapplication.Ladestation.getPower_3 ( )
```

Getter for power_3

Returns

power_3

6.6.1.20 getPower_4()

```
float com.example.myapplication.Ladestation.getPower_4 ( )
```

Getter for power_4

Returns

power_4

6.6.1.21 getPublicKey_1()

```
String com.example.myapplication.Ladestation.getPublicKey_1 ( )
```

Getter for public_key_1

Returns

public_key_1

6.6.1.22 getPublicKey_2()

```
String com.example.myapplication.Ladestation.getPublicKey_2 ( )
```

Getter for public_key_2

Returns

public_key_2

6.6.1.23 getPublicKey_3()

```
String com.example.myapplication.Ladestation.getPublicKey_3 ( )
```

Getter for public_key_3

Returns

public_key_3

6.6.1.24 getPublicKey_4()

```
String com.example.myapplication.Ladestation.getPublicKey_4 ( )
```

Getter for public_key_4

Returns

public_key_4

6.6.1.25 getState()

```
String com.example.myapplication.Ladestation.getState ( )
```

Getter for location

Returns

location

6.6.1.26 getStreet()

```
String com.example.myapplication.Ladestation.getStreet ( )
```

Getter for Street

Returns

street

6.6.2 Member Data Documentation

6.6.2.1 id

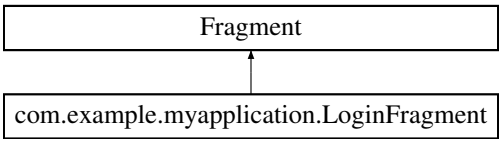
```
int com.example.myapplication.Ladestation.id
```

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/Ladestation.java

6.7 com.example.myapplication.LoginFragment Class Reference

Inheritance diagram for com.example.myapplication.LoginFragment:



Public Member Functions

- View onCreateView (@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)

6.7.1 Member Function Documentation

6.7.1.1 onCreateView()

```
View com.example.myapplication.LoginFragment.onCreateView (
            @NonNull LayoutInflater inflater,
            @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState )
```

Parameters

inflater	
container	
savedInstanceState	

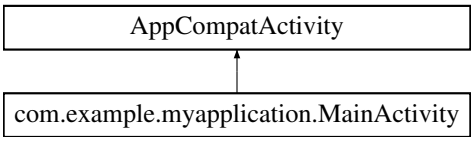
Returns

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/LoginFragment.java

6.8 com.example.myapplication.MainActivity Class Reference

Inheritance diagram for com.example.myapplication.MainActivity:



Static Public Member Functions

- static NavigationBarView getBottomNav ()

Static Public Attributes

- static boolean onServiceFragment = false
- static LatLng userLocation = new LatLng(49.872768, 8.651180)

Protected Member Functions

- void onCreate (Bundle savedInstanceState)

6.8.1 Member Function Documentation

6.8.1.1 getBottomNav()

```
static NavigationBarView com.example.myapplication.MainActivity.getBottomNav ( )  [static]
```

6.8.1.2 onCreate()

```
void com.example.myapplication.MainActivity.onCreate (
            Bundle savedInstanceState )  [protected]
```

Function triggers on create

Parameters

savedInstanceState	

6.8.2 Member Data Documentation

6.8.2.1 onServiceFragment

```
boolean com.example.myapplication.MainActivity.onServiceFragment = false  [static]
```

6.8.2.2 userLocation

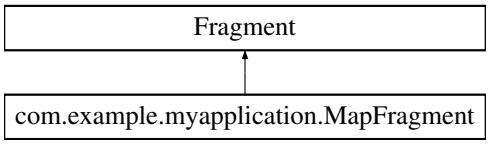
```
LatLng com.example.myapplication.MainActivity.userLocation = new LatLng(49.872768, 8.651180)
[static]
```

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/MainActivity.java

6.9 com.example.myapplication.MapFragment Class Reference

Inheritance diagram for com.example.myapplication.MapFragment:



Public Member Functions

- View onCreateView (@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)
- void onConfigurationChanged (@NonNull Configuration newConfig)

6.9.1 Member Function Documentation

6.9.1.1 onConfigurationChanged()

```
void com.example.myapplication.MapFragment.onConfigurationChanged (
            @NonNull Configuration newConfig )
```

6.9.1.2 onCreateView()

```
View com.example.myapplication.MapFragment.onCreateView (
            @NonNull LayoutInflater inflater,
            @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState )
```

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/MapFragment.java

6.10 com.example.myapplication.Ladestation.ModuleType Enum Reference

Public Attributes

- SerializedName =("Normalladeeinrichtung") STANDARD

6.10.1 Member Data Documentation

6.10.1.1 SerializedName

```
com.example.myapplication.Ladestation.ModuleType.SerializedName =("Normalladeeinrichtung")
STANDARD
```

The documentation for this enum was generated from the following file:

- app/src/main/java/com/example/myapplication/Ladestation.java

6.11 com.example.myapplication.Ladestation.PlugType Enum Reference

Public Attributes

- SerializedName =("AC Steckdose Typ 2") AC_PLUG_TYPE_2

6.11.1 Member Data Documentation

6.11.1.1 SerializedName

```
com.example.myapplication.Ladestation.PlugType.SerializedName =("AC Steckdose Typ 2") AC_↩
PLUG_TYPE_2
```

The documentation for this enum was generated from the following file:

- app/src/main/java/com/example/myapplication/Ladestation.java

6.12 com.example.myapplication.PopUpService Class Reference

Public Member Functions

- PopUpService (Context context, StationAdapter stationAdapter, SaveCheckedState saveCheckedState)

6.12.1 Constructor & Destructor Documentation

6.12.1.1 PopUpService()

```
com.example.myapplication.PopUpService.PopUpService (
            Context context,
            StationAdapter stationAdapter,
            SaveCheckedState saveCheckedState )
```

Constructor

Parameters

context	The context
stationAdapter	the current station adapter
saveCheckedState	the savestate

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/PopUpService.java

6.13 com.example.myapplication.SaveCheckedState Class Reference

Public Member Functions

- SaveCheckedState ()
- boolean isShowFavourites ()
- void setShowFavourites (boolean showFavourites)
- boolean isShowFastCharging ()
- void setShowFastCharging (boolean showFastCharging)
- int getSearchRange ()
- void setSearchRange (int searchRange)

6.13.1 Constructor & Destructor Documentation

6.13.1.1 SaveCheckedState()

```
com.example.myapplication.SaveCheckedState.SaveCheckedState ( )
```

Constructor

6.13.2 Member Function Documentation

6.13.2.1 getSearchRange()

```
int com.example.myapplication.SaveCheckedState.getSearchRange ( )
```

Returns showFastCharging

Returns

showFastCharging

6.13.2.2 isShowFastCharging()

```
boolean com.example.myapplication.SaveCheckedState.isShowFastCharging ( )
```

Returns showFastCharging

Returns

showFastCharging

6.13.2.3 isShowFavourites()

```
boolean com.example.myapplication.SaveCheckedState.isShowFavourites ( )
```

Returns showFavourites

Returns

showFavourites

6.13.2.4 setSearchRange()

```
void com.example.myapplication.SaveCheckedState.setSearchRange (
            int searchRange )
```

Returns showFavourites

Returns

showFavourites

6.13.2.5 setShowFastCharging()

```
void com.example.myapplication.SaveCheckedState.setShowFastCharging (
            boolean showFastCharging )
```

Returns showFastCharging

Returns

showFastCharging

6.13.2.6 setShowFavourites()

```
void com.example.myapplication.SaveCheckedState.setShowFavourites (
            boolean showFavourites )
```

Returns showFavourites

Returns

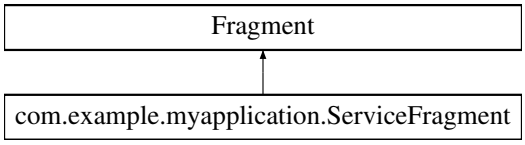
showFavourites

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/SaveCheckedState.java

6.14 com.example.myapplication.ServiceFragment Class Reference

Inheritance diagram for com.example.myapplication.ServiceFragment:



Public Member Functions

- View onCreateView (@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState)

6.14.1 Member Function Documentation

6.14.1.1 onCreateView()

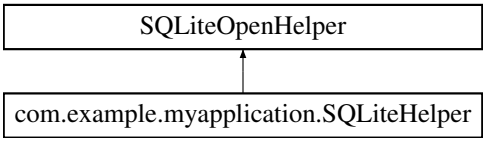
```
View com.example.myapplication.ServiceFragment.onCreateView (
            @NonNull LayoutInflater inflater,
            @Nullable ViewGroup container,
            @Nullable Bundle savedInstanceState )
```

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/ServiceFragment.java

6.15 com.example.myapplication.SQLiteHelper Class Reference

Inheritance diagram for com.example.myapplication.SQLiteHelper:



Public Member Functions

- SQLiteHelper (Context context)
- void onCreate (SQLiteDatabase sqLiteDatabase)
- void onUpgrade (SQLiteDatabase sqLiteDatabase, int i, int i1)
- void saveSingleFavoriteToDB (int id)
- void removeSingleFavoriteFromDB (int id)
- ArrayList< Integer > readFavouritesFromDB ()
- void saveSingleFlagToDB (int id)
- ArrayList< Integer > readFlagsFromDB ()
- void removeSingleReportFromDB (int id)

Static Public Attributes

- static final String DATABASE_NAME = "database.db"
- static final String FAVOURITE_TABLE_NAME = "favourites"
- static final String REPORTS_TABLE_NAME = "reports"
- static final String ATTRIBUTE_ID = "ID"

6.15.1 Constructor & Destructor Documentation

6.15.1.1 SQLiteHelper()

```
com.example.myapplication.SQLiteHelper.SQLiteHelper (
            Context context )
```

Constructor

Parameters

context	

6.15.2 Member Function Documentation

6.15.2.1 onCreate()

```
void com.example.myapplication.SQLiteHelper.onCreate (
            SQLiteDatabase sqLiteDatabase )
```

Creates the basic DB

Parameters

sqLiteDatabase	the SQLiteDatabase

6.15.2.2 onUpgrade()

```
void com.example.myapplication.SQLiteHelper.onUpgrade (
            SQLiteDatabase sqLiteDatabase,
            int i,
            int i1 )
```

6.15.2.3 readFavouritesFromDB()

```
ArrayList< Integer > com.example.myapplication.SQLiteHelper.readFavouritesFromDB ( )
```

Returns the favorites as ArrayList

Returns

The favorites as ArrayList

6.15.2.4 readFlagsFromDB()

```
ArrayList< Integer > com.example.myapplication.SQLiteHelper.readFlagsFromDB ( )
```

Reads all reported stations from DB and returns as arrayList

Returns

the arrayList

6.15.2.5 removeSingleFavoriteFromDB()

```
void com.example.myapplication.SQLiteHelper.removeSingleFavoriteFromDB (
            int id )
```

Removes a single favorite from db

Parameters

id	the ladestation's ID

6.15.2.6 removeSingleReportFromDB()

```
void com.example.myapplication.SQLiteHelper.removeSingleReportFromDB (
            int id )
```

Remove a single Report from DB

Parameters

id	The ladestations ID to remove the report from

6.15.2.7 saveSingleFavoriteToDB()

```
void com.example.myapplication.SQLiteHelper.saveSingleFavoriteToDB (
            int id )
```

Saves a single favorite to the database

Parameters

id	the ID of the ladestation to save as fav

6.15.2.8 saveSingleFlagToDB()

```
void com.example.myapplication.SQLiteHelper.saveSingleFlagToDB (
            int id )
```

Saves a single reported ladestations id to db

Parameters

id	The ID of the ladestation

6.15.3 Member Data Documentation

6.15.3.1 ATTRIBUTE_ID

```
final String com.example.myapplication.SQLiteHelper.ATTRIBUTE_ID = "ID"  [static]
```

6.15.3.2 DATABASE_NAME

```
final String com.example.myapplication.SQLiteHelper.DATABASE_NAME = "database.db"  [static]
```

6.15.3.3 FAVOURITE_TABLE_NAME

```
final String com.example.myapplication.SQLiteHelper.FAVOURITE_TABLE_NAME = "favourites"  [static]
```

6.15.3.4 REPORTS_TABLE_NAME

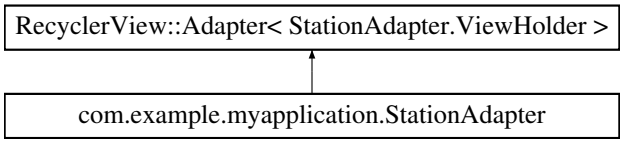
```
final String com.example.myapplication.SQLiteHelper.REPORTS_TABLE_NAME = "reports"  [static]
```

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/SQLiteHelper.java

6.16 com.example.myapplication.StationAdapter Class Reference

Inheritance diagram for com.example.myapplication.StationAdapter:



Classes

- class ViewHolder

Public Member Functions

- StationAdapter (ArrayList< Ladestation > stationList, SQLiteHelper sqLiteHelper)
- ViewHolder onCreateViewHolder (@NonNull ViewGroup parent, int viewType)
- void onBindViewHolder (@NonNull ViewHolder holder, int position)
- int getItemCount ()
- void toggleButtonRepWorker (Ladestation ladestation, ImageButton button)
- void setNewList (ArrayList< Ladestation > newList)

6.16.1 Constructor & Destructor Documentation

6.16.1.1 StationAdapter()

```
com.example.myapplication.StationAdapter.StationAdapter (
            ArrayList< Ladestation > stationList,
            SQLiteHelper sqLiteHelper )
```

6.16.2 Member Function Documentation

6.16.2.1 getItemCount()

```
int com.example.myapplication.StationAdapter.getItemCount ( )
```

6.16.2.2 onBindViewHolder()

```
void com.example.myapplication.StationAdapter.onBindViewHolder (
            @NonNull ViewHolder holder,
            int position )
```

6.16.2.3 onCreateViewHolder()

```
ViewHolder com.example.myapplication.StationAdapter.onCreateViewHolder (
            @NonNull ViewGroup parent,
            int viewType )
```

6.16.2.4 setNewList()

```
void com.example.myapplication.StationAdapter.setNewList (
            ArrayList< Ladestation > newList )
```

Sets the stationList to new list

Parameters

newList	The new station list

6.16.2.5 toggleButtonRepWorker()

```
void com.example.myapplication.StationAdapter.toggleButtonRepWorker (
            Ladestation ladestation,
            ImageButton button )
```

Does the same as toggleButtonRep, but allows the worker to "repair" the station

Parameters

ladestation	The ladestation to repair
button	The button (the button, button)

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/StationAdapter.java

6.17 com.example.myapplication.StationAPI Class Reference

Static Public Member Functions

- static SQLiteHelper getSqLiteHelper ()
- static void initialize (Context context)
- static ArrayList< Ladestation > sort (ArrayList< Ladestation > stationsList, double lat, final double lon, final int range)
- static ArrayList< Ladestation > getProximityStations (ArrayList< Ladestation > stationsList, double lat1, double lon1, int dist)

6.17.1 Member Function Documentation

6.17.1.1 getProximityStations()

```
static ArrayList< Ladestation > com.example.myapplication.StationAPI.getProximityStations (
            ArrayList< Ladestation > stationsList,
            double lat1,
            double lon1,
            int dist )  [static]
```

Returns only the nearby stations, given 2 coordinates and distance

Parameters

stationsList	The station list
lat1	the coordinate to use as center
lon1	the coordinate to use as center
dist	the distance

Returns

the array of Ladestationen in range

6.17.1.2 getSqLiteHelper()

```
static SQLiteHelper com.example.myapplication.StationAPI.getSqLiteHelper ( )  [static]
```

6.17.1.3 initialize()

```
static void com.example.myapplication.StationAPI.initialize (
            Context context )  [static]
```

Initializes the Station API with default values

Parameters

context	

6.17.1.4 sort()

```
static ArrayList< Ladestation > com.example.myapplication.StationAPI.sort (
            ArrayList< Ladestation > stationsList,
            double lat,
            final double lon,
            final int range )  [static]
```

Sorts the array list by distance

Parameters

stationsList	
lat	
lon	
range	

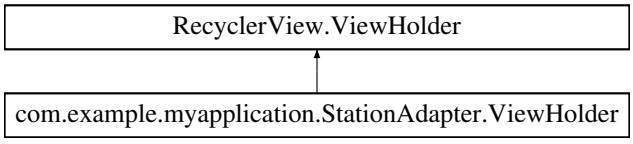
Returns

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/StationAPI.java

6.18 com.example.myapplication.StationAdapter.ViewHolder Class Reference

Inheritance diagram for com.example.myapplication.StationAdapter.ViewHolder:



Public Member Functions

- ViewHolder (View itemView)

Public Attributes

- TextView stationTextTop
- TextView stationTextCenter
- TextView stationTextBottom
- ImageButton favouriteButton
- ImageButton reportButton
- int stationID

6.18.1 Constructor & Destructor Documentation

6.18.1.1 ViewHolder()

```
com.example.myapplication.StationAdapter.ViewHolder.ViewHolder (
            View itemView )
```

6.18.2 Member Data Documentation

6.18.2.1 favouriteButton

```
ImageButton com.example.myapplication.StationAdapter.ViewHolder.favouriteButton
```

6.18.2.2 reportButton

```
ImageButton com.example.myapplication.StationAdapter.ViewHolder.reportButton
```

6.18.2.3 stationID

```
int com.example.myapplication.StationAdapter.ViewHolder.stationID
```

6.18.2.4 stationTextBottom

```
TextView com.example.myapplication.StationAdapter.ViewHolder.stationTextBottom
```

6.18.2.5 stationTextCenter

```
TextView com.example.myapplication.StationAdapter.ViewHolder.stationTextCenter
```

6.18.2.6 stationTextTop

```
TextView com.example.myapplication.StationAdapter.ViewHolder.stationTextTop
```

The documentation for this class was generated from the following file:

- app/src/main/java/com/example/myapplication/StationAdapter.java

Chapter 7

File Documentation

7.1 app/build/generated/source/build↩ Config/debug/com/example/myapplication/BuildConfig.java File Reference

Classes

- class com.example.myapplication.BuildConfig

Packages

- package com.example.myapplication

7.2 app/build/generated/source/build↩ Config/release/com/example/myapplication/BuildConfig.java File Reference

Classes

- class com.example.myapplication.BuildConfig

Packages

- package com.example.myapplication

7.3 app/src/main/java/com/example/myapplication/DownloadThread.java File Reference

Classes

- class com.example.myapplication.DownloadThread

Packages

- package com.example.myapplication

7.4 app/src/main/java/com/example/myapplication/GlobalStorage.java File Reference

Classes

- class com.example.myapplication.GlobalStorage

Packages

- package com.example.myapplication

7.5 app/src/main/java/com/example/myapplication/HomeFragment.java File Reference

Classes

- class com.example.myapplication.HomeFragment

Packages

- package com.example.myapplication

7.6 app/src/main/java/com/example/myapplication/Ladestation.java File Reference

Classes

- class com.example.myapplication.Ladestation
- enum com.example.myapplication.Ladestation.PlugType
- enum com.example.myapplication.Ladestation.ModuleType

Packages

- package com.example.myapplication

7.7 app/src/main/java/com/example/myapplication/LoginFragment.java File Reference

Classes

- class com.example.myapplication.LoginFragment

Packages

- package com.example.myapplication

7.8 app/src/main/java/com/example/myapplication/MainActivity.java File Reference

Classes

- class com.example.myapplication.MainActivity

Packages

- package com.example.myapplication

7.9 app/src/main/java/com/example/myapplication/MapFragment.java File Reference

Classes

- class com.example.myapplication.MapFragment

Packages

- package com.example.myapplication

7.10 app/src/main/java/com/example/myapplication/PopUpService.java File Reference

Classes

- class com.example.myapplication.PopUpService

Packages

- package com.example.myapplication

7.11 app/src/main/java/com/example/myapplication/SaveCheckedState.java File Reference

Classes

- class com.example.myapplication.SaveCheckedState

Packages

- package com.example.myapplication

7.12 app/src/main/java/com/example/myapplication/ServiceFragment.java File Reference

Classes

- class com.example.myapplication.ServiceFragment

Packages

- package com.example.myapplication

7.13 app/src/main/java/com/example/myapplication/SQLiteHelper.java File Reference

Classes

- class com.example.myapplication.SQLiteHelper

Packages

- package com.example.myapplication

7.14 app/src/main/java/com/example/myapplication/StationAdapter.java File Reference

Classes

- class com.example.myapplication.StationAdapter
- class com.example.myapplication.StationAdapter.ViewHolder

Packages

- package com.example.myapplication

7.15 app/src/main/java/com/example/myapplication/StationAPI.java File Reference

Classes

- class com.example.myapplication.StationAPI

Packages

- package com.example.myapplication

7.16 app/src/test/java/com/example/myapplication/ExampleUnitTest.java File Reference

Classes

- class com.example.myapplication.ExampleUnitTest

Packages

- package com.example.myapplication

Index