Technology
Arts Sciences
TH Köln

Medieninformatik Master
Multiperspective Product Development

Projekt II – Entwicklung SS22

Projectowner: Herr Prof. Dr. Rer. Nat Christian Kohls

Hazal Nuray Karanci | 11113840

Öznur Karadayi | 11103304

Inhalt

Abbildungsverzeichnis

1 Einführung

Frau Mileva Maric-Einstein (80) befindet sich wie jeden Monat einmal in der Engel Apotheke, um ihre Bluthochdruck-Tabletten zu besorgen. Da sie sehr vergesslich geworden ist, speichert sie für das nächste Mal den Standort mit ihrer Medikamentenliste in die Remind Me There App ab. Dazu wählt sie eine Entfernung von 70 Metern aus. Schließlich sucht sie sich noch einen passenden Klingelton. Somit wird Frau Mileva Maric-Einstein beim nächsten Stadtbesuch rechtzeitig von ihrer App auf ihre Medikamentenliste erinnert.

Smartphones dienen mit der Entwicklung und Bereicherung unseres Jahrzehnts viel mehr als nur ein Kommunikationsmittel, es ist auch unsere Assistenz, Ratgeber oder sogar Rückzugsort. Wenn wir die Smartphones nutzen, fühlen wir uns frei und sicher. Sie sind zu einem elektronischen Teil von uns geworden. Im Rahmen des Praxisprojekts wurde eine Applikation: “Remind Me There App” entwickelt, die ortsgebundene Erinnerungsnachrichten an den Benutzer sendet. In unserem Projekt II – Entwicklung haben wir dieses Projekt weiterhin verwendet, um die fehlenden Funktionen zu implementieren. Dabei haben wir auf unsere Schwerpunktbezogene Module geachtet um unsere App nicht nur praktisch, sondern auch theoretisch mit korrekten Inhalten und Recherchen zu erstellen. Wir haben insgesamt zwei große Funktionen implementiert, die unsere App für uns vervollständigt hat. Bevor die Implementation erfolgt ist, haben wir Recherchen durchgeführt, die auf die Inhalte von unserem Schwerpunkt basieren.

2 Vorbereitung

Da wir direkt mit Projekt 2 gestartet sind, haben wir zur Beginn nochmals eine Vorbereitung für die App durchgeführt. Hierbei haben wir uns auf die Methoden des Moduls Qualitätsmanagement und Qualitätssicherung fokussiert.

2.1 AHP und Kano

Die Methode AHP und Kano (Analytic Hierarchy Process) dienen dazu, Anforderungen zu klassifizieren und ebenso zu präferieren. Die Anforderungsermittlung ist die „Stimme“ des Kunden und wird in drei Kategorien klassifiziert. Diese sind Basisfaktoren, Leistungsfaktoren und Begeisterungsfaktoren. Durch diese Klassifizierung kann geprüft werden, welche Anforderungen für das Minimum Viable Product (MVP) erwünscht oder nicht erwünscht sind. Wir haben zwei Anforderungen nach Kano klassifiziert. Anschließend haben wir die Anforderungspriorisierung nach AHP durchgeführt. Hierbei haben wir die funktionalen Anforderungen des Projektes “RemindMeThere” nach Kano herausgesucht. Das Ziel ist es die Klassifikation von Anforderungen einzuteilen, um herauszufinden, welche Anforderungen für das MVP benötigt werden. Wir haben zuerst die Willkommensseite ohne Login Funktion und die gewünschten Funktionen untersucht, dies sind die Login Funktionen und die Teilfunktion für unser App “RemindMeThere”. Die Probanden waren andere Kommilitonen aus unserem Modul Qualitätssicherung und Management. Zuerst wurden der Probanden 3 funktionale und 3 dysfunktionale Fragen gestellt;

Funktionale Fragen:

- Was würden Sie empfinden, wenn Sie bei der Anmeldung auf der Willkommensseite Ihren Namen sehen?
- Was würden Sie empfinden, wenn die persistente Speicherung automatisch erfolgt?
- Was würden Sie empfinden, wenn die App nur funktioniert, wenn Sie angemeldet sind?

Dysfunktionale Fragen:

- Was würden Sie empfinden, wenn Sie nach der Anmeldung direkt die Hauptfunktionen sehen können?
- Was würden Sie empfinden, wenn Sie die persistente Abspeicherung durch eine manuelle Bestätigung erfolgen muss?
- Was würden Sie empfinden, wenn die App auch voll funktionsfähig als Gast nutzbar ist?

Anforderung	Priorität	Willkommensseite	Standort bearbeiten	Erinnerung bearbeiten	Benutzerinformationen bearbeiten
Willkommensseite	0,05	1	1/9	1/9	1/3
Standort bearbeiten	0,41	9	1	1	3
Erinnerung bearbeiten	0,41	9	1	1	3
Benutzerinformationen bearbeite	0,14	3	1/3	1/3	1
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
	-	-	-	-	-
Total		22	2 4/9	2 4/9	7 1/3

Abbildung 1: Priorisierung mit Analytic Hierarchy Process (AHP)

Viel wichtiger	9
Wichtiger	3
Gleich wichtig	1
Unwichtiger	1/3
Viel unwichtiger	1/9

Abbildung 2: Bewertungstabelle (AHP)

Anhand der Methode AHP und Kano lässt sich erschließen, dass die Funktionalitäten “Standort bearbeiten” und “Erinnerung bearbeiten” für die Probanden viel wichtiger sind als die Funktionalitäten “Benutzerinformationen bearbeiten” und “Willkommensseite”.

2.2 Normen und Standards

Bevor wir unsere Entwicklung fortgesetzt haben, haben wir unsere Normen und Standards recherchiert. Wir können noch nicht alles vollständig umsetzen, da unser App noch erweitert wird und wir auch die alten Funktionen zwischenzeitlich ändern müssen. Wir geben drauf acht, dass wir bei der Implementierung bereits die UI und UX-Design die Normen und Standards einhalten. Die aktuell eingehaltenen Normen und Standards, die für uns einen hohen Stellenwert haben sind:

- **EN ISO 9241 –110:** Regelt die Interaktionprinzipien zwischen Benutzer und System
- **ISO/IEC 13817:** für Programmiersprachen (Programminglanguages, theirenvironmentsandsystem, softwareinterfaces)
  Gestaltung von grafischen Symbolen zur Anwendung in der technischen Produktdokumentation
- **ISO/IEC/IEEE 29119:** Software testing Standard
- **DIN EN ISO 9001:2015:** Qualitätsmanagement
- **DIN EN ISO 1934 / DIN ISO 1932:** Geoinformationen- Standortbezogene Dienste: Ortsbezogene Dienste
- DIN EN ISO/IEC 27701: Sicherheitstechniken – Erweiterung zu **ISO/IEC 27001** und
  ISO/IEC 27002 für das Management von Informationen zum Datenschutz
  Erweiterung von Datenschutzkriterien
- **DIN-EN-ISO14638 (2015-12-00):** GPS Geometrische Produktspezifikation (GPS). Das Framework soll Benutzern von ISO-GPS-Standards von Nutzen sein, indem es den Umfang des Anwendungsbereichs der verschiedenen Standards veranschaulicht und ihre Beziehung zueinander aufzeigt.

2.3 Datenbankarchitektur

Zur persistenten Speicherung der Daten der App wird eine SQLite-Datenbank benutzt. Dabei haben wir uns für Room entschieden, eine Bibliothek zur persistenten

Speicherung von Datenobjekten, die eine Abstraktionsschicht über SQLite, bestehend aus Data Access Objects und Annotationen, bereitstellt. Über einen Cache wird eine konsistente Kopie der wichtigen Informationen im App angezeigt unabhängig von einem Internetzugriff.

Benötigte Typen für den Datenbankzugriff mit Room:
In diesem Abschnitt wird erläutert, welche Klassen und Schnittstellen für den Zugriff auf eine Datenbankdatei mit Room benötigt werden.

Als **Entität** bezeichnet man eine persistente Klasse, deren Objekte in der Datenbank gespeichert werden sollen. Sie wird auf eine Datenbanktabelle abgebildet, wobei jedem Objekt der Entität eine Zeile der Tabelle entspricht. Die Werte eines Objekts für die Instanzvariablen der persistenten Klasse, werden auf die Tabellenspalten abgebildet.

Room legt eine Datenbanktabelle und deren Schema (Namen und Typen der Attribute bzw. Spalten) nötigenfalls passend zur Definition der zugehörigen Entitätsklasse automatisch an.

Für die Login Funktion haben wir die Google Firebase genutzt. Dies ist nützlich um die Registrierungs- und Anmeldedaten sowie die Passwörter zu speichern. Firebase ist ein Echtzeit-Datenbank. Wir können über die Authentifizierung in Firebase Konsole die angemeldeten bzw. Die registrierten Emails sehen. Wenn wir über die Konsole auf die Database gehen, sehen wir die Profilinformationen der Nutzer. Wir können die Passwörter nicht sehen, sowie die Handlungen die Nutzer in dem App unternehmen, sondern nur die Profil Daten wie Name, Alter, E-Mail etc.

2.4 Event-Sequence-Graph

Bevor wir unsere Funktion implementiert haben, haben wir uns dazu entschieden einen Test mithilfe eines Event-Sequence-Graph (kurz: ESG) durchzuführen. Dies ist wichtig, um zuverlässige mobile Apps zu entwickeln. Für uns war es beim Testen wichtig, die Fehlerblindbereiche zu berücksichtigen und den Vorgang des Testens abstrakt anzugehen.

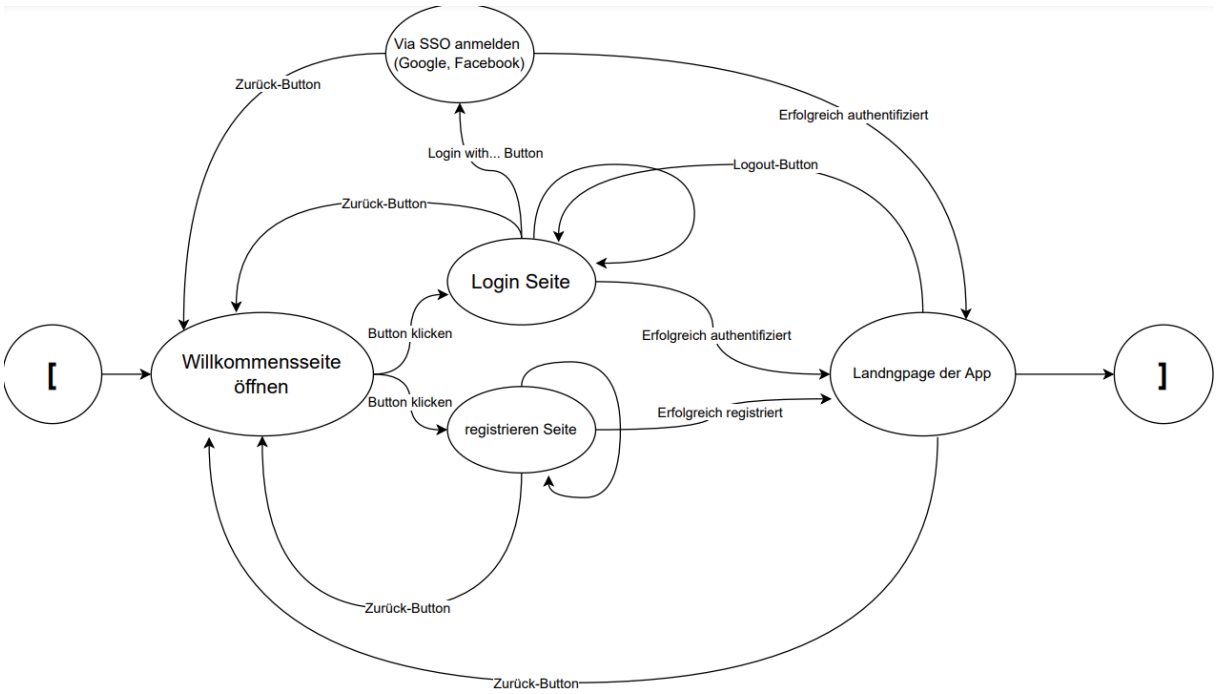

Abbildung 3: Event-Sequence-Graph für die Login Funktion

2.5 Ablaufdiagramm

Dieser Ablaufdiagramm ist eine Ergänzung unseres Praxisprojekts. Es stellt die neu implementierten Funktionen dar. Die Login Funktion erscheint auf der Willkommensseite. Wenn die Nutzer zum ersten Mal die App nutzen, dann müssen sie sich mit ihrer E-Mail-Adresse und einem Passwort registrieren. Die Registrierung muss über die Mail-Server bestätigt werden. Wenn dieser Schritt erfolgreich getätigt ist, sind die Nutzer im System gespeichert und können sich in unsere App anmelden und somit in die Startseite für die weitere Funktionen der App gelangen. Wenn die E-Mail-Adresse falsch ist, wird die Anmeldung nicht funktionieren und die Nutzer bleiben weiterhin auf der Willkommensseite. Wir geben unseren Nutzern die Möglichkeit, deren Passwörter zurückzusetzen, für den Fall, falls sie es vergessen. Um Standort zu teilen, müssen die Nutzer auf der Startseite den “Standort anlegen” wählen, daraufhin ist am nächsten Fenster die drei Möglichkeiten zu sehen und der dritte ist die “Standort teilen”. Wählt ein Nutzer diesen Button, dann gelingt diese automatisch in die Google Maps auf den aktuellen Standort. Auf der Map ist ein kleiner Button, die mit einem Klick einen Toolbar öffnet und Kommunikationsmöglichkeiten anbietet. Wählt Nutzer den “Teil-Button”, so wird der

Standort per SMS mit anderen Personen geteilt. Die App öffnet somit die Messages im Smartphone und selbst bleibt sie im Hintergrund geöffnet. Geht der Nutzer zurück zu RemindMeThere App, befindet diese weiterhin auf der Google Maps.

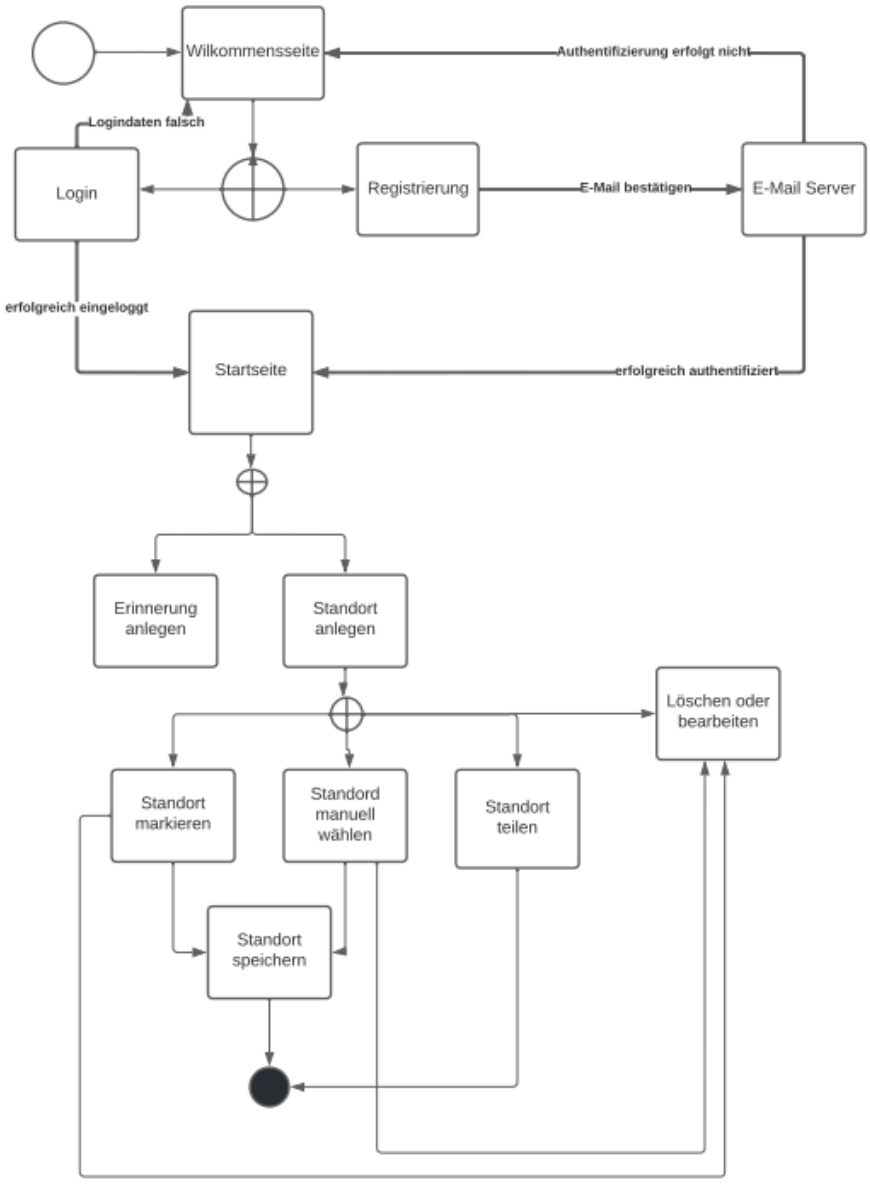

Abbildung 4: Ablauf Diagramm für die Login- und Teil-Funktion

3 Umsetzung

Als wir das Projekt RemindMeThere erneut geöffnet haben, sind wir auf sehr viele Probleme gestoßen. Vor alle war der Softwarestand sehr veraltet. Dies hat enorm

viel Zeit genommen, da wir jedes Update einzeln ausführen müssten und bei viele Updates hatten wir keine Meldung von Android Studio erhalten. Dies haben wir selbst herausgefunden, indem wir unsere App immer auf die Lauffähigkeit getestet haben. Unsere AVD-Manager war ebenfalls veraltet, sodass wir die HAXM neu installieren müsste. Da Android Studio etwas problematisch geworden ist, hatten wir Schwierigkeiten bei der Installation von HAXM. Wir müsste auch die APIs aktualisieren, dafür haben wir unsere Google Account aktualisiert und neue Zugangsdaten für die API beantragt. Dies hat ebenfalls einige Tage in Anspruch genommen und ohne die erfolgreiche Aktualisierung unsere APIs könnten wir die App nicht ausführen. Nachdem wir geschafft haben unsere App erfolgreich zu aktualisieren, wurde zunächst eine Teilfunktion und die Login-Funktion implementiert. Um in unser Projekt nicht durcheinander zu kommen, haben wir für die einzelnen Funktionen eigene Projekte in Android Studio erstellt und dies als eigene Apps ausgeführt. Wenn die Funktionen vollständig funktioniert haben, haben wir sie einzeln mit unserem Projekt RemindMeThere App zusammengeführt. Die Zusammensetzung ist nicht komplex, aber zeitaufwendig, da wir die Activities und Layouts(xml) anpassen mussten. Ebenfalls mussten wir die Berechtigungen und API‘s aktualisieren.

3.1 Implementation der Login-Funktion

Um eine klare Ansicht zu verschaffen welche Funktionen und Buttons wir benötigen, haben wir die Login Seite in XML optisch dargestellt. Wir haben die Schaltfläche für Anmeldung und eine Textansicht für das vergessene Passwort sowie Registrierung erstellt. Um die Nutzer während der Anmelde-Prozedur nicht zu verunsichern, haben wir einen Fortschrittsbalken erstellt, dies zeigt, dass die App geladen wird, wenn sie sich anmelden oder registrieren. Nachdem wir unsere Optik für die Login-Seite erstellt haben, öffneten wir einen neuen Activity, um unseren Code zu beginnen. Für die Verbindung mit dem Firebase, haben wir über Tools in Android Studio die Firebase E-Mail und Passwort Authentication gewählt und den Schritten gefolgt. Nach dem die Verbindung erfolgreich hergestellt worden ist, haben wir die restlichen Variablen initialisiert, sodass die Buttons zugehörige Funktionen bekamen.

```
private TextView register, forgotpassword;
private EditText editTextMail, editTextPassword;
private Button signIn;

private FirebaseAuth mAuth;
private ProgressBar progressBar;

@Override
protected void onCreate(Bundle savedInstanceState){
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_registeru1);

    register = (TextView) findViewById(R.id.register);
    register.setOnClickListener(this);

    signIn = (Button) findViewById(R.id.signIn);
    signIn.setOnClickListener(this);

    editTextMail = (EditText) findViewById(R.id.email);
    editTextPassword = (EditText) findViewById(R.id.password);

    progressBar = (ProgressBar) findViewById(R.id.progressBar);

    mAuth = FirebaseAuth.getInstance();

    forgotpassword = (TextView) findViewById(R.id.forgotPassword);
    forgotpassword.setOnClickListener(this);


@Override
public void onClick(View v) {
    switch (v.getId()){
        case R.id.register:
            startActivity(new Intent(this, registeru.class));
```

```
            break;

        case R.id.signIn:
            userLogin();
            break;

        case R.id.forgotPassword:
            startActivity(new Intent(this, forgotpassword.class));
            break;

    }
```

3.1.1 Registrierung

Wir haben Validierungen erstellt damit der Benutzer kein leeres Formular sendet. Dafür müssten wir die jede Eingabe in String konvertieren. Mit wenigen If-Anweisungen könnten wir dann die Eingaben Validieren. Ist der Namensfeld leer, wird die Registrierung nicht durchgeführt, bis in diesem Feld ein Name geschrieben wird. Dies Gilt für die gesamten Profilinformationen bei der Registrierung. Für die E-Mail-Adresse ist eine weitere Validierung erforderlich. Durch einen Import von Android Utility Paket kann man ein E-Mail-Muster abgleichen z.B. ob die angegebene E-Mail ein “@” oder ein “.de” enthält. Wenn die E-Mail-Adresse ungültig ist, kommt eine Meldung mit dem Text “Bitte geben Sie eine gültige Mail Adresse ein”.

```
if (fullName.isEmpty()) {
    editTextFullName.setError("Name erforderlich!");
    editTextFullName.requestFocus();
    return;

}

if (email.isEmpty()) {
```

```
    editTextMail.setError("Email erforderlich!");
    editTextMail.requestFocus();
    return;
}

if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
    editTextMail.setError("Bitte geben Sie eine gültige E-Mail-
Adresse an!");
    editTextMail.requestFocus();
    return;


}

if (password.isEmpty()) {
    editTextPassword.setError("Passwort erforderlich");
    editTextPassword.requestFocus();
    return;
}

if (password.length() < 6) {
    editTextPassword.setError("Passwort muss mindestens 6 Zeichen
besitzen");
    editTextPassword.requestFocus();
    return;
}
```

Um unsere Bedingungen in Firebase zu speichern, haben wir ein neues Java Klasse für das Benutzerobjekt erstellt und die Benutzerinformationen und das Objekt gespeichert. Dies haben wir dann in Firebase gesendet. Die Java-Klasse wurde nach “User” benannt. Hier haben wir String Variablen für die Benutzerinformationen erstellt, also für Name und E-Mail. Wir haben zwei Konstruktoren erstellt, ein öffentlicher Konstruktor und ein Konstruktor für zwei Argumente: Name und E-Mail.

```
public class User {

    public String fullName, email;

    public User() {

    }
    public User(String fullName, String email){
        this.fullName = fullName;

        this.email = email;

    }
```

Wir haben in unsere Main Klasse für Registrierung ein Progressbar erstellt, um die Fortschritte der Registrierung sichtbar zu machen. Die “mAuth” ist ein Objekt für die Firebase Authentifizierung, bei dieser Methode geben wir die Variablen E-Mail und Passwort an. Um zu prüfen, ob der Nutzer vollständig registriert ist, nutzen wir die CompleteListener und implementieren onCompleteListener<AuthResult>. Mit einer neuen IF-Bedindung können wir überprüfen, ob die Registrierung erfolgreich war. Dann haben wir ein Benutzerobjekt erstellt, damit die Benutzerinformationen und die Echtzeitdatenbank aufgelöst werden.

```
progressBar.setVisibility(View.VISIBLE);
mAuth.createUserWithEmailAndPassword(email,password)
        .addOnCompleteListener(new OnCompleteListener<AuthResult>()
{
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                if(task.isSuccessful()) {
                    User user = new User(fullName, email);
```

```
FirebaseDatabase.getInstance().getReference("Users")

.child(FirebaseAuth.getInstance().getCurrentUser().getUid())

.setValue(user).addOnCompleteListener(new OnCompleteListener<Void>()
{
                                @Override
                                public void onComplete(@NonNull
Task<Void> task) {

                                    if (task.isSuccessful()) {

Toast.makeText(registeru.this, "Registrierung Erfolgreich!",
Toast.LENGTH_LONG).show();

progressBar.setVisibility(View.VISIBLE);

                                    } else {

Toast.makeText(registeru.this, "Registrierung Fehlgeschlagen!
Versuchen Sie es erneut!", Toast.LENGTH_LONG).show();

progressBar.setVisibility(View.GONE);
                                    }
                                }
                            });
                }else{
                    Toast.makeText(registeru.this, "Registrierung
Fehlgeschlagen! Versuchen Sie es erneut!",
Toast.LENGTH_LONG).show();
                    progressBar.setVisibility(View.GONE);
                }
            }
```

3.1.2 Anmeldung

Bisher haben wir die Schaltfläche Registrierung programmiert. Danach haben wir die Texteingabeflächen für die Anmeldung validiert. Dazu haben wir eine Private Variable für die Bearbeitung des Textes, für die E-Mail und Passwort erstellt. Auch für die Anmeldung haben wir ein Progressbar erstellt. Sobald der Nutzer auf Login klickt, fängt dieser Balken an sich zu drehen. Wir haben eine Methode erstellt, um den Benutzer anzumelden, sodass wenn der Benutzer auf Login klickt, öffnet ein neues Activity. Dies ist die Activity der Startseite. In diese Methode werden die Anmeldeinformationen abgerufen. Sie werden in einer Zeichenfolge konvertiert, sobald der Nutzer die beiden Eingaben bereitgestellt hat. Diese wurden ebenfalls in String konvertiert. Dafür haben wir zuerst eine E-Mail-Stringvariable erstellt. Für die müssen wir überprüfen, dass der Benutzer keine leere Anmeldedaten angibt, mit (!Patterns.EMAIL_ADRESS.matcher(email)matches()){ überprüfen wir, ob die E-Mail Adresse korrekt eingegeben worden ist. Für das Passwort haben wir ebenfalls eine IF-Bedingung verwendet, außerdem müssen die Passwörter länger als 6 Zeichen sein, dies haben wir auch in einer IF-Bedingung zugewiesen. Auch hier haben wir die “mAuth” Objekt für eine erfolgreiche Firebase Authentifizierung verwendet. Letztendlich haben wir die Startseite nach einer erfolgreichen Anmeldung festgelegt.

```
progressBar.setVisibility(View.VISIBLE);

mAuth.signInWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>()
{
    @Override
    public void onComplete(@NonNull Task<AuthResult> task) {

        if(task.isSuccessful()){
            FirebaseUser user =
FirebaseAuth.getInstance().getCurrentUser();
```

```
            if(user.isEmailVerified()) { //Email Vertifikation
                startActivity(new Intent(registeru1.this,
Profile.class)); //evtl. hier anstatt profile->logo

            }else{
                user.sendEmailVerification();
                Toast.makeText(registeru1.this, "Überprüfen Sie Ihre
E-Mail, um Ihr Konto zu bestätigen!", Toast.LENGTH_SHORT).show();
            }
        }else{
            Toast.makeText(registeru1.this, "Login Fehlgeschlagen!
Überprüfen Sie Ihre Eingaben!", Toast.LENGTH_SHORT).show();
        }
    }
```

3.1.3 Reset Passwort

Wenn die Nutzer deren Passwort vergessen, gibt es eine Möglichkeit dies zurückzusetzen. Hierzu muss man die E-Mail angeben, zu der wir einen Link zum zurücksetzen schicken, damit das Passwort mit Firebase-Authentifizierung zurücksetzen kann. Zuallererst haben wir eine leere Activity geöffnet, danach haben wir ein setOnClickListener auf das Reset Passwort angelegt. Damit dieser Button uns auf die neue Activity führt. Wir haben ein Layout erstellt, die eine E-Mail-Texteingabe Feld beinhaltet und und den Button “Reset Passwort”, dieses Layout wurden dann mit unserem neuen Activity verbunden. Diese Prozedur erfolgt über Firebase Authentifizierung, deshalb nutzen wir auth = FirebaseAuth.getInstance();. Auch hier müssen wir sicher gehen, dass der Benutzer keine leeren Formulare schickt, daher werden diese validiert mit IF(Patterns.EMAIL_ADRESS.matcher(email).matches()). Wenn die E-Mail-Adresse inkorrekt ist, wird eine Meldung hervorgerufen, danach kann der Nutzer seine E-Mail korrigieren. Wenn die E-Mail-Adresse korrekt ist, bekommt der Nutzer eine Bestätigung zum Überprüfen der Posteingang.

```
private void resetPassword() {
    String email = emailEditText.getText().toString().trim();


    if(email.isEmpty()){
        emailEditText.setError("Email ist Erforderlich");
        emailEditText.requestFocus();
        return;
    }

    if(!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        emailEditText.setError("Bitte geben Sie eine gültige E-Mail
an");
        emailEditText.requestFocus();
        return;

    }

    progressBar.setVisibility(View.VISIBLE);
    auth.sendPasswordResetEmail(email).addOnCompleteListener(new
OnCompleteListener<Void>() {
        @Override
        public void onComplete(@NonNull Task<Void> task) {

            if(task.isSuccessful()){
                Toast.makeText(forgotpassword.this,"Überprüfen Sie
Ihre E-Mail, um Ihr Passwort
zurückzusetzen!",Toast.LENGTH_LONG).show();
            }else{
                Toast.makeText(forgotpassword.this,"\n" +
                        "Versuchen Sie es erneut. Es ist ein Fehler
aufgetreten!",Toast.LENGTH_LONG).show();
            }
```

3.1.4 Firebase Authentication

Damit die App die Identität des Benutzers wiedererkennt wurde die Firebase Authentication aktiviert. Hiermit kann man Benutzerdaten sicher in einem Cloud speichern und diese jederzeit wiederverwenden. Hierbei unterstützt die Firebase Authentication die Authentifizierung mit den E-Mail-Adressen und den dazugehörigen Passwörtern. Somit kann der User sich in die App einloggen.

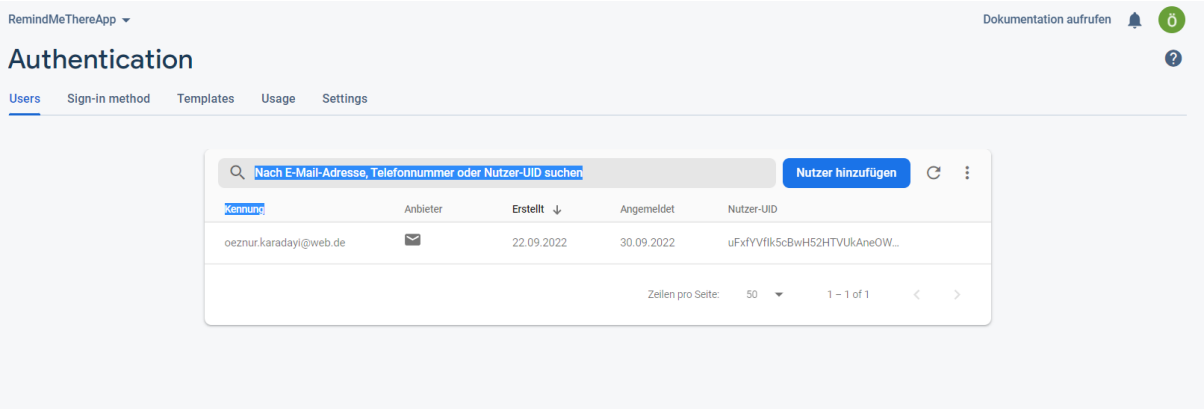

Abbildung 5: Firebase Authentication

3.1.5 Firebase Realtime Database

Da wir die Daten (Name, E-Mail und Passwort) speichern möchten und auch diese jederzeit offline nutzen möchten, haben wir die NoSQL-Cloud-Datenbank genutzt. Hierbei werden die Daten als JSON gespeichert und in Echtzeit synchronisiert. Auch wenn die Daten sich verändern (Beispiel: Passwort Reset) werden diese wieder direkt aktualisiert.

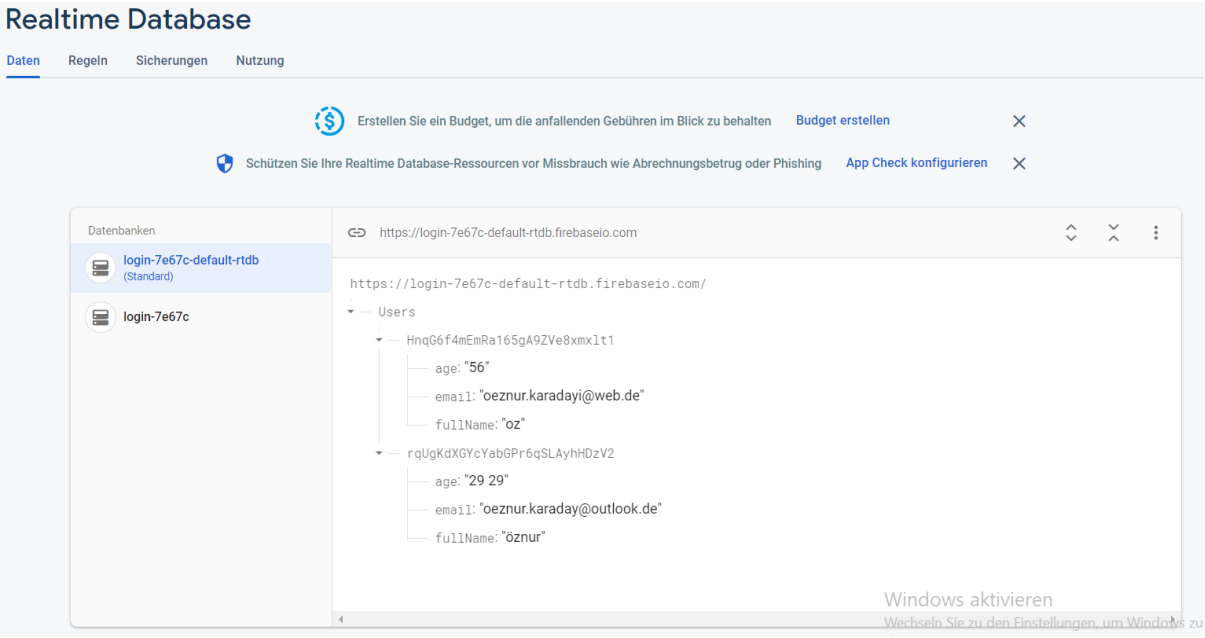

Abbildung 6: Firebase Realtime Database

3.2 Implementation der Teil-Funktion

Ziel dieser Funktion ist, den markierten Standorten mit anderen Kontakten zu teilen. Wenn ein Ort zu empfehlen ist oder für die Kontaktperson zu nutzen ist, dann können sie ebenfalls über den geteilten Link auf den Standort zugreifen. Falls die Internetverbindung der Kontaktperson nicht in der ist, den Link zu öffnen, können die Nutzer einfach den Koordinaten aus dem Link ablesen.

Um diese Funktion zu implementieren haben wir zwei Activities genutzt. Eine Activity ist für die Button “Markiere ein Standort” und die andere ist um öffnet die Google Maps mit dem Button zum Teilen dieser Standort.

Hierfür haben wir eine API-Key erstellt, um die Karte von Google Maps in die App einzubinden. Durch diese kann nun die App mit Google kommunizieren und die Karte wird angezeigt. Außerdem wurde die API-Key auf 4 Bibliotheken eingeschränkt: **Geocoding** API, **Gelocation** API, **Maps SDK for Android** und **Places** API.

Beim **Geocoding** werden Adressen (z. B. eine Straßenadresse) in geografische Koordinaten (z. B. Längen- und Breitengrad) umgewandelt, hierdurch wird der Marker auf die Karte positioniert.

Die **Geolocation-API** gibt einen Standort und einen Genauigkeitsradius basierend auf Informationen über Mobilfunkmasten und WiFi-Knoten zurück, die der mobile Client erkennen kann. Dieses Dokument beschreibt das Protokoll, das verwendet wird, um diese Daten an den Server zu senden und eine Antwort an den Client zurückzusenden.

Mit dem **Maps SDK for Android** kann man der App nun die Karte hinzufügen. Hierüber hat man die standardisierte Google Kartendarstellung, welche auch auf bestimmte Touch-Gesten reagiert. Außerdem lassen sich Markierungen und weiteres in die Karte aufnehmen, um zusätzliche Informationen zu Standorten anzuzeigen und eine Interaktion mit der Karte zu ermöglichen.

Mit der **Places API** kann man Informationen zu Orten über verschiedene Kategorien suchen, darunter Einrichtungen und geografische Orte. Die Orte können anhand der Entfernung oder anhand eines Textstrings gesucht werden. Eine „Ortssuche“ gibt eine Liste von Orten sowie zusammenfassende Informationen zu jedem Ort zurück.

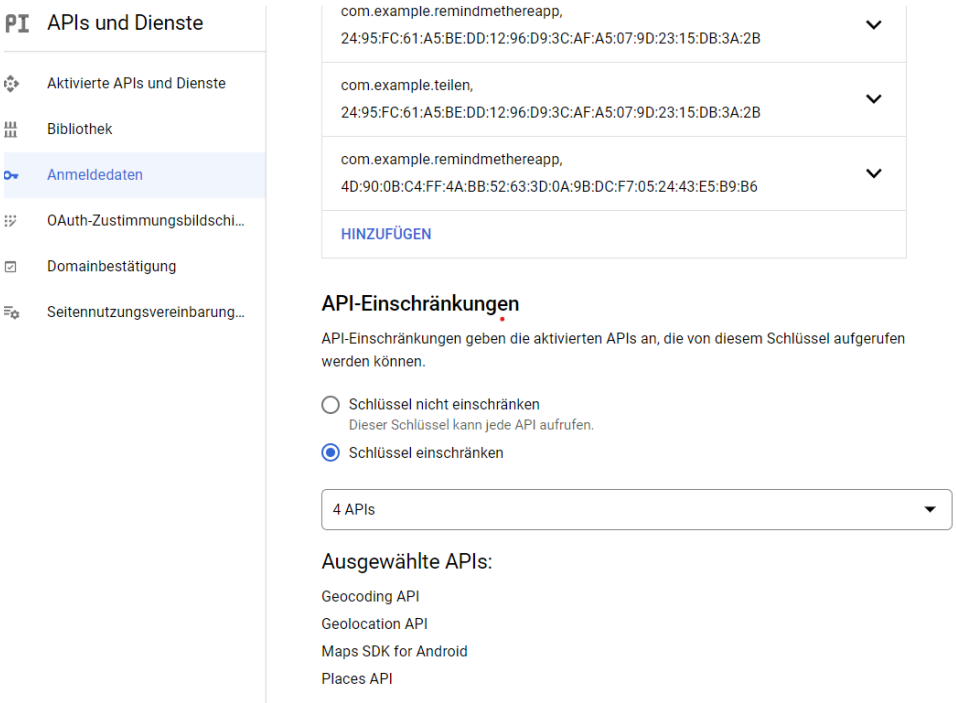

Abbildung 7: API und Dienste

3.3 Bibliotheken

Unsere Bibliotheken sind unten wie folgt geschildert;

```
dependencies {

        implementation 'androidx.appcompat:appcompat:1.5.0'
        implementation 'com.google.android.material:material:1.6.1'
        implementation
'androidx.constraintlayout:constraintlayout:2.1.4'
        implementation
'androidx.coordinatorlayout:coordinatorlayout:1.2.0'
        implementation 'androidx.lifecycle:lifecycle-livedata-
ktx:2.5.1'
        implementation 'androidx.lifecycle:lifecycle-viewmodel-
ktx:2.5.1'

//    implementation "org.jetbrains.kotlin:kotlin-
stdlib:$kotlin_version"
        implementation 'androidx.navigation:navigation-fragment-
ktx:2.5.1'
        implementation 'androidx.navigation:navigation-ui-ktx:2.5.1'
        testImplementation 'junit:junit:4.13.2'
        implementation 'androidx.cardview:cardview:1.0.0'
        def nav_version = "2.3.2"
        implementation "androidx.navigation:navigation-fragment-
ktx:$nav_version"
        implementation "androidx.navigation:navigation-ui-
ktx:$nav_version"
        implementation 'com.mapbox.mapboxsdk:mapbox-android-
sdk:9.2.0'
        //noinspection GradleCompatible
        implementation 'com.google.android.gms:play-services-
```

```
maps:18.1.0'

        //RecyclerView
        implementation 'androidx.recyclerview:recyclerview:1.2.1'
        implementation "com.google.android.material:material:1.6.1"

        // design für Toolbar
        implementation "com.google.android.material:material:1.8.0-
alpha01"
        implementation 'androidx.cardview:cardview:1.0.0'
        implementation 'com.android.support:multidex:1.0.3'



        // room
        def room_version = "2.2.6"

        //noinspection GradleDependency
        implementation "androidx.room:room-runtime:$room_version"
        annotationProcessor "androidx.room:room-
compiler:$room_version"

        androidTestImplementation 'androidx.test.ext:junit:1.1.3'
        androidTestImplementation 'androidx.test.espresso:espresso-
core:3.4.0'
    }
    buildFeatures {
        viewBinding true
    }
}
repositories {
    mavenCentral()
}
```

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.5.0'
    implementation
'androidx.constraintlayout:constraintlayout:2.1.4'

    implementation 'com.android.support:multidex:1.0.3'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.navigation:navigation-fragment:2.5.1'
    implementation 'androidx.navigation:navigation-ui:2.5.1'

    implementation 'com.google.android.material:material:1.6.1'
    implementation 'android.arch.persistence.room:runtime:1.1.1'
    implementation 'com.google.firebase:firebase-auth:21.0.8'
    implementation 'com.google.firebase:firebase-database:20.0.6'
    annotationProcessor
'android.arch.persistence.room:compiler:1.1.1'

    //noinspection UseOfBundledGooglePlayServices
    implementation 'com.google.android.gms:play-services:12.0.1'
```

4 Problematik

Unsere Schwierigkeiten lagen daran, die bestehende RemindMeThere App auf den aktuellen Stand zu bringen. Da die Softwareupdates sehr veraltet waren, müssten wir alles einzeln aktualisieren und die Codezeilen erneuern. Die Bibliotheken müssten ebenfalls aktualisiert werden, viele Fehler konnten wir nicht entdecken und damit unser App wieder funktioniert, müssten wir alle Fehler selbst aufdecken. Als nächstes hat die Implementierung von Login und Registrierungsfunktion und der Teil Funktion viel Zeit in Anspruch genommen, da wir vorhin nicht derart programmiert haben bzw. die Methoden nicht kannten. Bis die Funktionen erfolgreich funktioniert haben, haben wir viele Wege probiert und getestet. Anschließend hat die

Zusammenführung der Projekte viel Zeit in Anspruch genommen, wie oben auch erwähnt haben wir unsere Funktionen zuerst in getrennten Projekten aufgeführt, damit wir sicher sein könnten, dass die einzelnen Funktionen richtig waren.

5 Ziel

Da wir unsere gewünschten Anforderungen erreicht haben, werden wir uns als nächstes mit User Interfaces und User Experiences beschäftigen. Da die nicht funktionalen Anforderungen sich immer erweitern werden bleibt das als ein bestehendes Ziel. Der Grund hierfür ist, dass die Digitalisierung sich immer fortsetzt, außerdem ändern sich die Kundenwünsche aus verschiedenen Gründen wie z.B. Jahreszeit, Umweltveränderungen oder die aktuellen Trends immer wieder neu. Wir werden uns mit den Inhalten Nutzerpsychology, Informationsarchitektur, Usability, Ästhetik und der Aha-Effekt beschäftigen. Uns ist es wichtig unser App nach Erfahrungen der Nutzer anzupassen, die Nutzung der App zu vereinfachen und das Design anzupassen. Wir werden beobachten und führen auch Aufgabenanalysen durch, um zu sehen, wie Benutzer Aufgaben in einem Benutzerfluss tatsächlich erledigen, wie z.B. Wie einfach ist der Prozess beim Standortteilen? Oder Wie leicht fällt es dem Nutzer sich zu registrieren? Bei UI-Design geht es für uns darum sicherzustellen, dass die Benutzeroberfläche unseres Produkts so intuitiv wie möglich ist und das bedeutet, dass jedes einzelne visuelle interaktive Element auf, das der Benutzer stoßen konnte, sorgfältig zu prüfen ist. Wir werden über Symbole und Schaltflächen, Typgrafie und Farbschemata, Abstände, Bilder und weiteres fortentwickeln.

6 Fazit

Im Rahmen des Projektes haben wir unser bereits bestehende App auf neue Funktionen erfolgreich erweitert. Unsere App verfügt jetzt über eine Registrierungs- und Login Funktion, die Nutzer können deren aktuellen Standort mit anderen teilen. Die Inhalte aus dem Modul Qualitätssicherung- und Management wurden

berücksichtigt und in dem Projekt verwendet. Da wir im Sommer Semester begonnen haben zu studieren, haben wir die anderen Module aus unserem Schwerpunkt noch nicht kennengelernt. In Github befindet sich unsere Repositories, unter dem folgenden Link sind sie erreichbar; https://github.com/hazal-01/remindmethere.

Literaturverzeichnis

T. Künneth: Android 8. Das Praxisbuch für Java-Entwickler. Rheinwerk Verlag, Bonn 2018, 5., aktualisierte Auflage 2018, ISBN 978-3-8362-6058-9.

BISWAS, Nabendu, 2022. *Beginning React and Firebase: Create Four Beginner-Friendly Projects Using React and Firebase* [online]. 1st ed. 2022. Berkeley, CA: Apress (2022) ; Imprint: Apress. ISBN 978-1-4842-7812-3. Verfügbar unter: https://doi.org/10.1007/978-1-4842-7812-3

Darbyshire, Paul; Darbyshire, Adam (2010):Google Maps. In: Paul Darbyshire und Adam Darbyshire (Hg.): Getting StartED with Google Apps.

Google LLC: Google Keep – Notizen & Listen. Online verfügbar unter: google.com

Zhekov N. (2016) Google Maps and Google Local Search. In: Lee N. (eds) Google It. Springer, New York, NY. https://doi.org/10.1007/978-1-4939-6415-4_8