

# Miniprojekt – Teil Android

Neben den Vorlesungen und Übungsaufgaben ist auch ein Miniprojekt Teil von Mobile and GUI Engineering. Das Projekt gliedert sich – wie auch die Vorlesung – in einen Android-Teil und einen WPF-Teil. Ziel des Projekts ist es, das Gelernte selbstständig auf ein neues Szenario anwenden zu können.

Miniprojekt bedeutet, dass das Projekt über mehrere Wochen läuft und von Ihnen selbstständig, in den Übungslektionen oder im Selbststudium, bearbeitet wird. Sie dürfen das Projekt auch mit einem Partner oder in einer Gruppe von maximal drei Personen durchführen (Teams mit Mitgliedern aus mehreren Studiengängen wären toll). Tragen Sie die Gruppen bitte hier ein: <https://goo.gl/0K8i5G>.

Die beiden Teile des Miniprojekts werden bewertet, aber nicht benotet. Nur wenn Sie beide bestanden haben sind Sie zur Prüfung zugelassen! Achten Sie also darauf, die Bewertungskriterien einzuhalten und sprechen Sie bei Unklarheiten mit Ihrem Übungsbetreuer.

Sie finden dieses Dokument auch unter <https://goo.gl/vVPVB6>.

## Bewertungskriterien

Die folgenden Bewertungskriterien dienen als Richtlinien für die Bewertung. Um das Projekt zu bestehen, müssen nicht alle Punkte vollumfänglich erfüllt sein, bei zu grossen Mängeln wird die Prüfungszulassung aber nicht erteilt:

Szenarios	Können die vorgegebenen Szenarios durchgespielt werden?
Design	Entspricht die App den Material Design Kriterien?
Interaktion	Behandelt die App Fehlerfälle und gibt dem Anwender Feedback dazu? Ist die Navigation nachvollziehbar (Back-Button)?
Code	Ist der Code sauber strukturiert (einheitlicher Stil, keine TODOs, Dokumentation nicht-trivialer Stellen)? Ist der Code in einem Repository eingchecked? Sind Tests vorhanden?

Die Bewertung erfolgt während einer kurzen Demonstration der App in Ihrer Übungslektion in der Semesterwoche 7 (also der letzten Android-Woche).

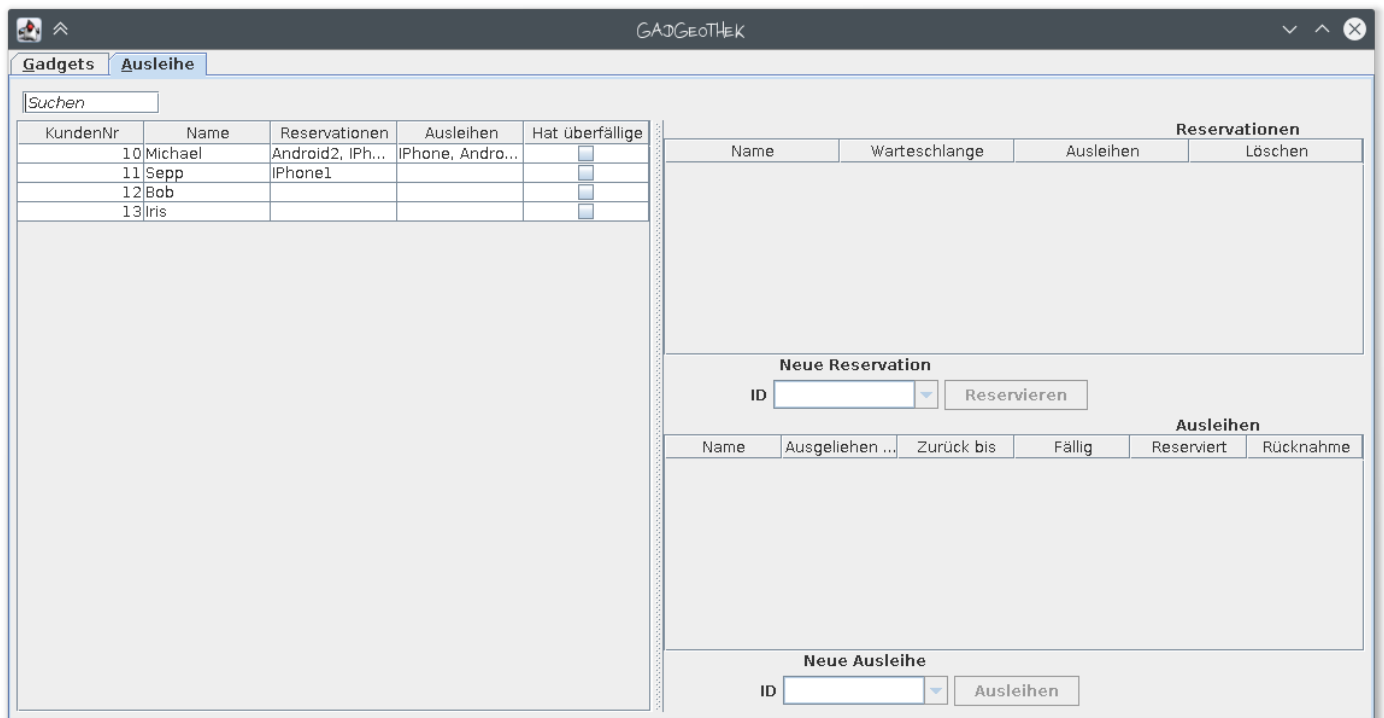
Im nächsten Kapitel folgt die Aufgabenstellung für die Gadgeotheek-App die wir bauen wollen. Alternativ dürfen Sie auch gerne ein eigenes Thema bearbeiten! Schreiben Sie dazu eine eigene Aufgabenstellung, definieren Sie Szenarios die Sie umsetzen werden und senden Sie sie an [mirko.stocker@hsr.ch](mailto:mirko.stocker@hsr.ch). Sie werden innert zwei Tagen einen go/no go Entscheid erhalten.

Im Umfang und Komplexität sollte die eigene Aufgabe in etwa der Gadgeotheek-App entsprechen (Domainmodell mit 3 Klassen, Kommunikation mit einem Server) und muss *zwingend* in Java (oder Kotlin) für Android entwickelt werden.

## Aufgabenstellung Gadgeotheek

Sie kennen bestimmt die [Gadgeotheek der HSR-Bibliothek](#) (intern). Im Miniprojekt geht es darum, eine App für die (fiktiven – wir haben leider keinen Zugriff auf den echten Bibliotheksserver) Bibliotheksbenutzer zu entwickeln.

Auf [GitHub](#) finden Sie das Administrationstool der Gadgeotheek, eine Swing-Applikation die im Vorgängermodul User Interfaces 1 entwickelt wurde und uns freundlicherweise von den Autoren Samuel Jost und Philipp Schilter zur Verfügung gestellt wird.



The screenshot shows the GADGEOtheK application window. The 'Ausleihe' tab is active. On the left, there is a search bar labeled 'Suchen' and a table with columns: KundenNr, Name, Reservationen, Ausleihen, and Hat überfällige. The table contains four rows of data. On the right, there are two main sections: 'Reservierungen' and 'Ausleihen'. The 'Reservierungen' section has a table with columns: Name, Warteschlange, Ausleihen, and Löschen. Below this is a 'Neue Reservation' section with an ID input field and a 'Reservieren' button. The 'Ausleihen' section has a table with columns: Name, Ausgeliehen ..., Zurück bis, Fällig, Reserviert, and Rücknahme. Below this is a 'Neue Ausleihe' section with an ID input field and an 'Ausleihen' button.

## Szenarios

Szenarios helfen Ihnen zu verstehen, in welchem Kontext die Software verwendet wird. Die untenstehenden Szenarien beschreiben, was ihre Applikation am Ende des Miniprojektes unterstützen sollte. Verwenden Sie sie zum Testen ihrer Applikation und schauen sie darauf, dass Sie möglichst alle davon optimal unterstützen können.

### Szenario Registrieren

Dario hat soeben von der App erfahren und hat diese in einer langweiligen MGE-Vorlesungssequenz bereits installiert. Da Dario noch keinen Account hat, muss er sich zuerst mit Angabe seiner E-Mail Adresse, Name, Passwort und Matrikelnummer registrieren.

### Szenario Ausleihe Prüfen

Nach dem Abschluss des Miniprojekts im Modul Mobile and GUI Engineering ist Luca nicht mehr sicher, ob er das Android-Smartphone ausgeliehen hatte oder ob es Dario war. Er loggt sich deshalb in der Gadgeotheek-App ein und schaut in seinen aktuellen Ausleihen nach, ob es an ihm ist das Smartphone zu retournieren.

## Szenario Reservation Erstellen

Anton ist ein grosser Fan von Apples Produkten, möchte aber herausfinden, ob Android wirklich so gut ist wie ihm Dario und Luca die ganze Zeit vorschwärmen. Leider sind im Moment alle Android- Smartphones ausgeliehen, deshalb trägt er sich in die Reservationsliste für das neue *Samsung Galaxy S6 Edge* ein.

## Szenario Reservation Löschen

In der Nacht auf heute hat Apple neue iPhones vorgestellt! Anton hat sich schon eines vorbestellt und will sich nicht doch von Android verführen lassen. Deshalb löscht er seine Reservation des *Samsung Galaxy S6 Edge* wieder.

## Szenario Bibliothek Wechseln

Dario macht ein Austauschsemester an der ZHAW. Seit neustem gibt es auch dort eine Gadgeotheek, und sie verwendet sogar dieselbe Software. Dario braucht nur den Server anzupassen und schon kann er wieder loslegen und neue Gadgets reservieren.

## Installation

Zum Projektstart ist ein grosser Teil der Domain und der Businesslogik der Gadgeotheek-App vorgegeben. Setzen sie bei ihrer Lösung diesen Code ein! Die nachfolgenden Klassendiagramme bietet ihnen eine Übersicht über den vorgegebenen Code. Sollten sie das Bedürfnis haben diesen Code grundlegend zu modifizieren, so dürfen sie dies gerne tun. Den Code finden Sie auf GitHub unter [github.com/HSR-MGE/Miniprojekt-Vorlage-Android](https://github.com/HSR-MGE/Miniprojekt-Vorlage-Android) (Achtung, dies ist keine Projektvorlage zum importieren sondern nur ein paar Klassen).

Der Server besteht aus einer einfachen Node.js-Anwendung die eine REST-Schnittstelle anbietet. Sie finden den Code dazu unter [github.com/HSR-MGE/Miniprojekt-Server](https://github.com/HSR-MGE/Miniprojekt-Server) . Sie können den Server entweder lokal starten, oder noch besser, Sie verwenden eine unserer zehn Instanzen `mge1.dev.ifs.hsr.ch`, `mge2 ... mge10.dev.ifs.hsr.ch`.

## Server lokal starten (nur bei Interesse)

Um den Server lokal zu starten müssen Sie [Node.js](https://nodejs.org/) installiert haben. Kopieren Sie dann die `package.json` und `server.js` Dateien in ein Verzeichnis (bzw. klonen Sie einfach das Repository). In diesem Verzeichnis installieren Sie dann mit npm (das ist der Node-Packagemanager) die benötigten Libraries und starten den Server:

```
$ npm install
...
$ node server.js
```

Der Server hört auf Port 8080 (HTTP) und gibt Debug-Output aus. Alternativ steht auch ein Docker-Container bereit den Sie stattdessen einsetzen können. Zum Beispiel so:

```
$ docker run -t -i -p 8080:8080 misto/miniprojekt-server
```

Die Administrationsoberfläche [github.com/HSR-MGE/Miniprojekt-Admin](https://github.com/HSR-MGE/Miniprojekt-Admin) starten Sie mit folgendem Befehl und unter der Angabe der Serveradresse und Port:

```
java -Dserver=localhost:8080 -jar gadgeotheek-admin.jar
```

oder

```
java -Dserver=mge7.dev.ifs.hsr.ch -jar gadgeotheek-admin.jar
```

Sobald sie später mit dem Android-Client auf denselben Server verbinden werden Sie Updates in der Administrationsoberfläche sehen.

## Vorlagen

Die Kommunikation mit dem Server erfolgt über die (auch vorgegebene) `LibraryService`-Klasse. Im Lauf der Vorlesung werden Sie erkennen, dass diese Klasse nicht unbedingt gerade Android Best-Practices entspricht, der Einfachheit halber setzen wir sie aber trotzdem ein (Android-Services sind erst in Woche 6 ein Thema – zu spät fürs Miniprojekt. Fortgeschnittene Teilnehmer die eine zusätzliche Herausforderung wünschen können diese Klasse z.B. mit [square.github.io/retrofit](https://square.github.io/retrofit) ersetzen. Ein umfangreiches Tutorial inklusive Node.js-Backend [finden Sie hier](#).).

Achtung: Ihre App benötigt für den Netzwerkzugriff natürlich die entsprechende Berechtigung im Manifest!

```
<uses-permission android:name="android.permission.INTERNET" />
```

Die Klassen haben externe Dependencies, zusätzlich müssen im `build.gradle` des app-Moduls folgende Dependencies ergänzt werden:

```
compile 'com.squareup.okhttp3:okhttp:3.4.1'
compile 'com.google.code.gson:gson:2.4'
```

Als erstes müssen Sie der `LibraryService`-Klasse mitteilen, wo der Server zu finden ist. Wenn Sie eine unserer Serverinstanzen verwenden, lautet der erste Aufruf:

```
LibraryService.setServerAddress("http://mgeX.dev.ifs.hsr.ch/public");
```

Beachten Sie das `/public`, welches der Serverteil (auch der lokal gestartete) erwartet. Wenn Sie die App auf dem Emulator starten und den Server auf ihrem Rechner am laufen haben können Sie mit der Adresse `"http://10.0.2.2:8080/public"` darauf verbinden.

Vor der ersten Anfrage (ausgenommen der Registrierung) an den Server müssen Sie sich einloggen, dies geschieht mit der `login`-Methode:

```
public static void login(String mail, String password, Callback<Boolean> callback)
```

Die Methode hat keinen Rückgabewert, aber nimmt ein `Callback`-Objekt entgegen, mit welchem der Rückgabewert des Aufrufs (hier ein `Boolean`) mitgeteilt wird. `Callback` ist ein Interface mit zwei Methoden:

```
public interface Callback<T> {
    void onCompletion(T input);
    void onError(String message);
}
```

```
}
```

Ist der Request erfolgreich (die Verbindung zum Server könnte ja z.B. auch unterbrochen werden) wird die `onCompletion`-Methode aufgerufen mit dem entsprechenden Resultat (also `boolean` beim login). Im Fehlerfall wird `onError` mit der Fehlermeldung aufgerufen. Die Login-Methode wird also folgendermasse benutzt:

```
LibraryService.login(email, password, new Callback<Boolean>() {  
    @Override  
    public void onCompletion(Boolean success) {  
        if(success) {  
            // Jetzt sind wir eingeloggt  
        } else {  
            // Passwort war falsch oder User unbekannt.  
        }  
    }  
  
    @Override  
    public void onError(String message) {  
        // Fehler z.B. in einem Toast/Snackbar darstellen  
    }  
});
```

Warum so kompliziert? Wir können das Resultat nicht einfach als Rückgabewert der Login-Methode zurückgeben, da sonst die App an dieser Stelle einfach blockiert ist und auf das Resultat des Servers wartet. Stattdessen geben wir einen Callback mit – zeigen dem User in der Zwischenzeit eine Progressbar oder ähnliches – und aktualisieren das UI dann wenn der Callback – `onCompletion` oder `onError` – eingetroffen ist.

In der Vorlage finden Sie zudem das Domain-Package, welches die drei Klassen Gadget (entspricht einem konkreten Geräte), Loan (eine Ausleihe eines Gerätes) und Reservation enthält. Die Klassen sollten hoffentlich selbsterklärend sein, bei Fragen zögern Sie aber bitte nicht diese in den Übungen zu stellen.

Auf den restlichen Seiten finden Sie noch die Klassendiagramme der Vorlagen. Sie werden wohl nicht alle Attribute der Klassen auch tatsächlich verwenden, da die Klassen aber auch in der Admin-Software eingesetzt werden sind diese dennoch enthalten.