Entwicklung einer Android Food Tracking App in der Programmiersprache Kotlin

Johannes Franz, Normen Krug

Wintersemester 2017/2018

Vorgelegt bei Prof. Dr. Michael Stepping

Inhaltsverzeichnis

1 Einleitung **1**

2 Motivation **2**

3 Kotlin als Programmiersprache **3**

4 Entwurfsphase **5**

5 Umsetzung **11**
5.1 Datenbank 11
5.2 Room 11

6 Aufgetretene Probleme **12**

7 Lessons learned **13**

8 Weitere Arbeiten **14**

1 Einleitung

Ziel ist es eine App zu entwickeln, die einen Rückschluss von sich zugenommener Nahrung auf Symptome zu ermöglichen. Dabei helfen die in der App eingetragenen Datenpunkte und Fotos diese mit einer z.B. allergischen Reaktion zu verknüpfen. Zielgruppe sind dabei Menschen, die wegen Erkrankungen aus ihren zu sich zugenommenen Speisen und Getränken Rückschlüsse auf ihr Wohlbefinden treffen wollen.

2 Motivation

Durch eigenen Bedarf motiviert entstand die Idee für diese App. Um aus wiederkehrenden Mahlzeiten eine Unverträglichkeit abzuleiten, stellt die App Funktionen bereit.

3 Kotlin als Programmiersprache

Für die App wurde die Programmiersprache Kotlin verwendet, welche auf der Google IO im Mai 2017 als offizielle Sprache für Android eingeführt wurde. Kotlin hat einen besser lesbaren Syntax als Java. Wie bei Java wird auch der Bytecode für die Java Virtual Machine übersetzt. Insgesamt ist Kotlin der Programmiersprache Swift 3 bzw 4 syntaktisch sehr nahe, was den Umgang als Entwickler Mobiler Anwendungen zusätzlich erleichtert.

```
package hello
import kotlin.collections.* // line comment

/**
 * Doc comment here for ‘SomeClass‘
 * @see Iterator#next()
 */
@Deprecated("Deprecated␣class")
private class MyClass<out T : Iterable<T>>(var prop1 : Int) {
    fun foo(nullable : String?, r : Runnable, f : () -> Int,
        fl : FunctionLike, dyn: dynamic) {
        println("length\nis␣${nullable?.length}␣\e")
        val ints = java.util.ArrayList<Int?>(2)
        ints[0] = 102 + f() + fl()
        val myFun = { -> "" };
        var ref = ints.size
        ints.lastIndex + globalCounter
        ints.forEach lit@ {
            if (it == null) return@lit
            println(it + ref)
        }
        dyn.dynamicCall()
        dyn.dynamicProp = 5
    }

    val test = """hello
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣world
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣kotlin"""
    override fun hashCode(): Int {
        return super.hashCode() * 31
    }
}
fun Int?.bar() {
    if (this != null) {
        println(message = toString())
```

```
    }
    else {
        println(this.toString())
    }
}
var globalCounter : Int = 5
    get = field
abstract class Abstract {
}
object Obj
enum class E { A, B }
interface FunctionLike {
    operator fun invoke() = 1
}
```

4 Entwurfsphase

In der Entwurfsphase wurde keine Zeit verschwendet und der Fokus auf Handskizzen gelegt. Da die Teammitglieder die zur Verfügung stehenden Steuerelemente in Android Der unnötige Schritt diese in einem professionellen Entwurf nachzubauen sparte Zeit, so dass sich direkt mit der Programmierung beschäftigt werden konnte.

Erster Screen Entwurf:

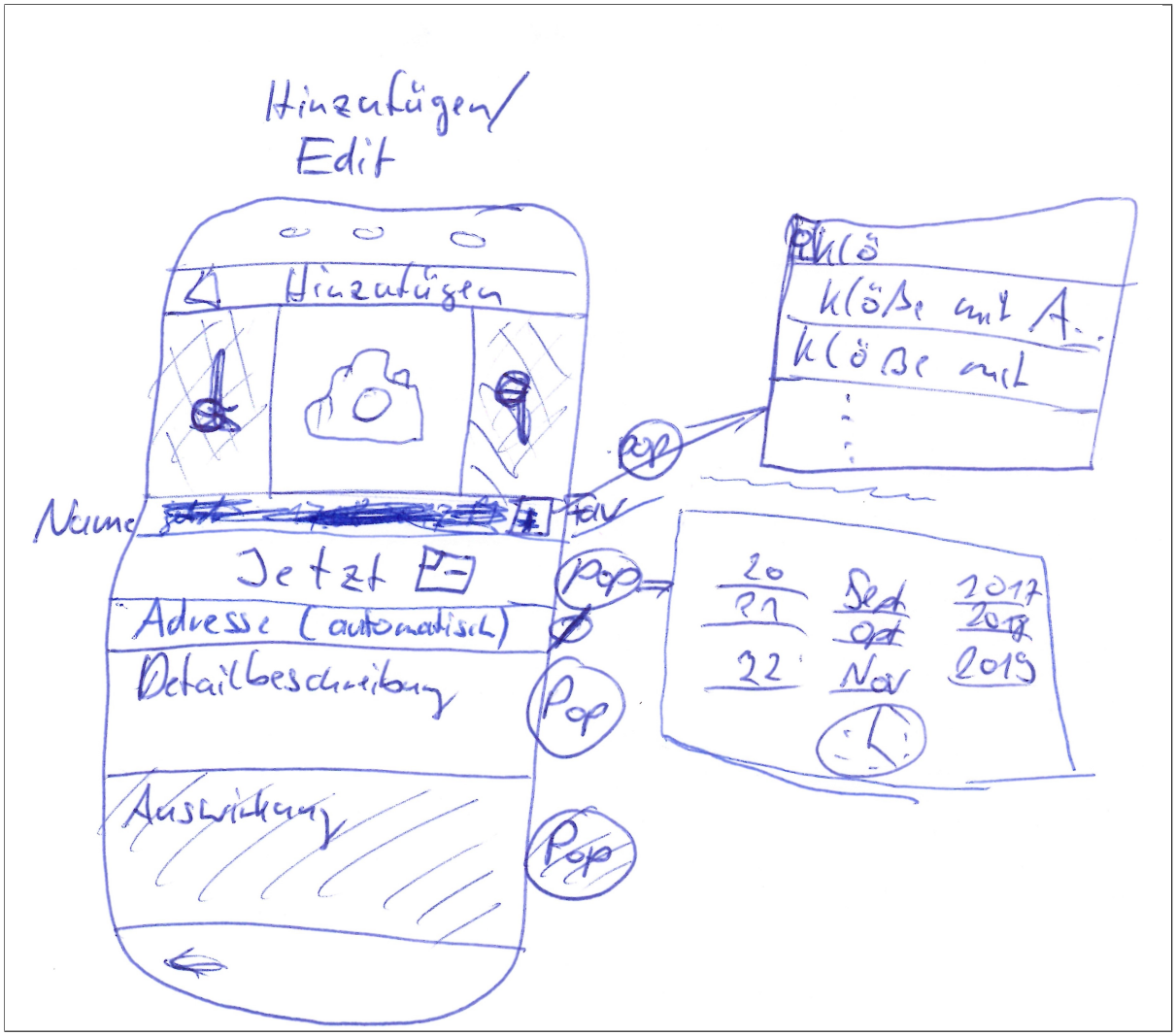

Abbildung 4.1: Hinzufügen eines Eintrags

Abbildung 4.2: Main Screen

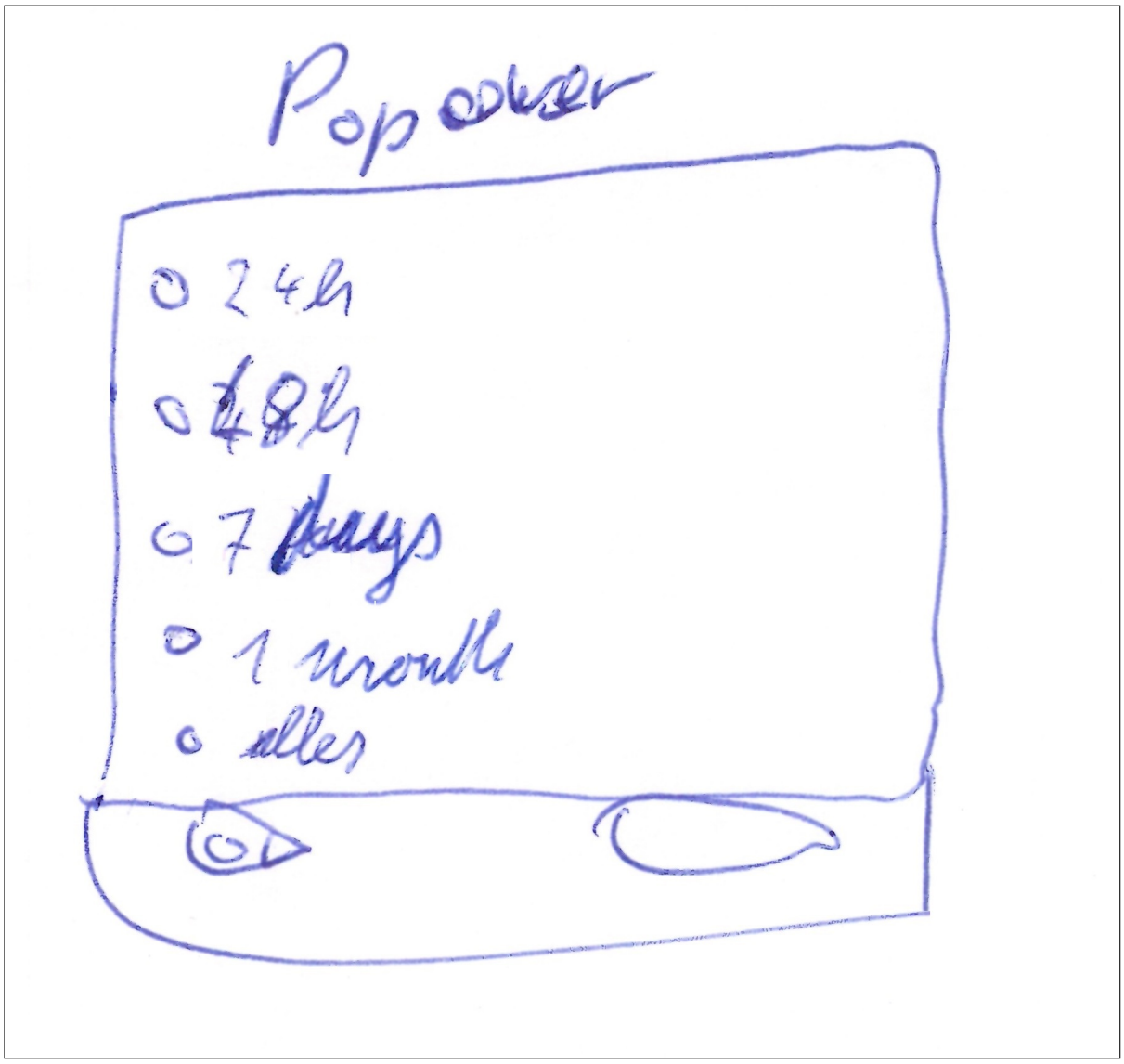

Abbildung 4.3: Popover Screen

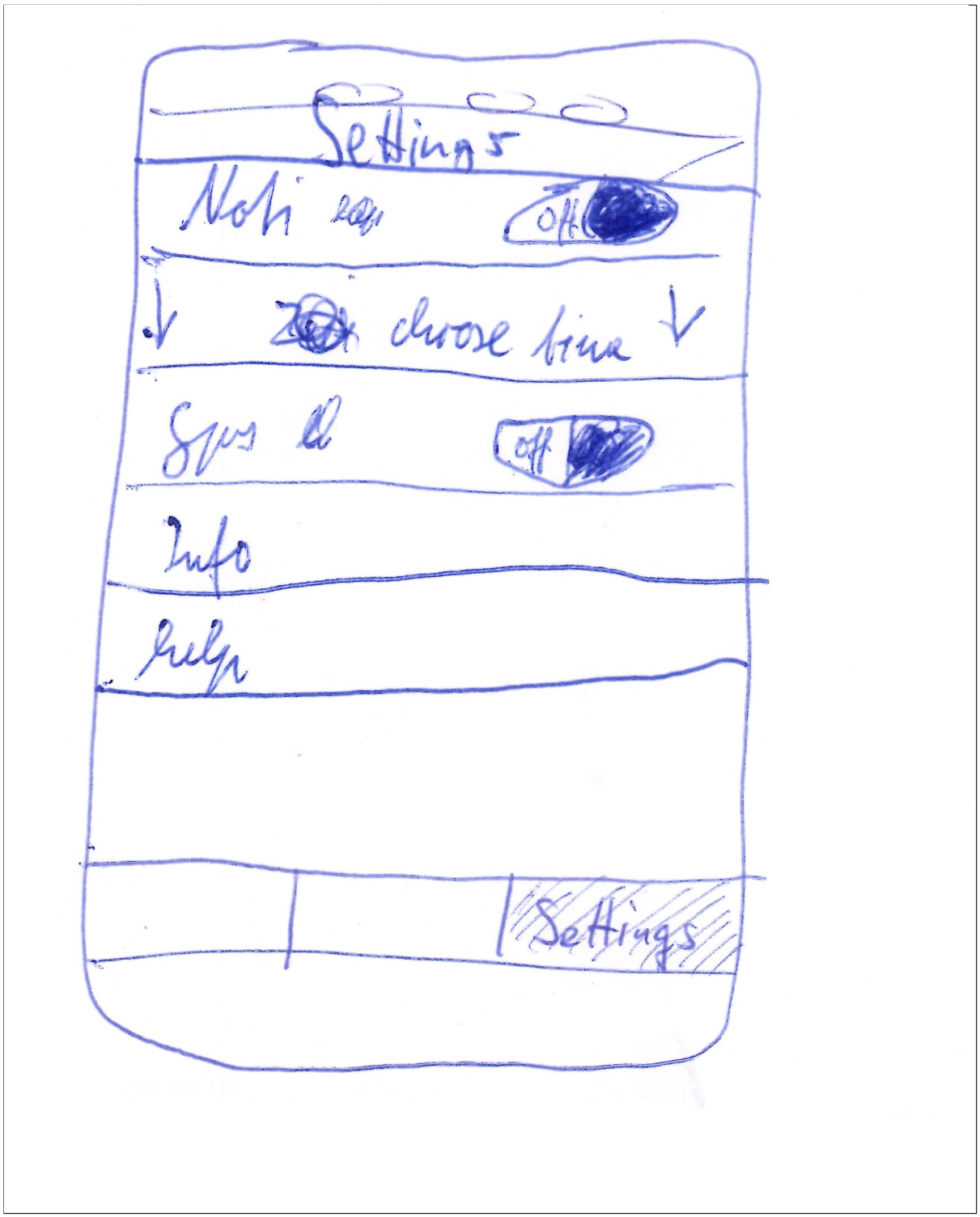

Abbildung 4.4: Settings Screen

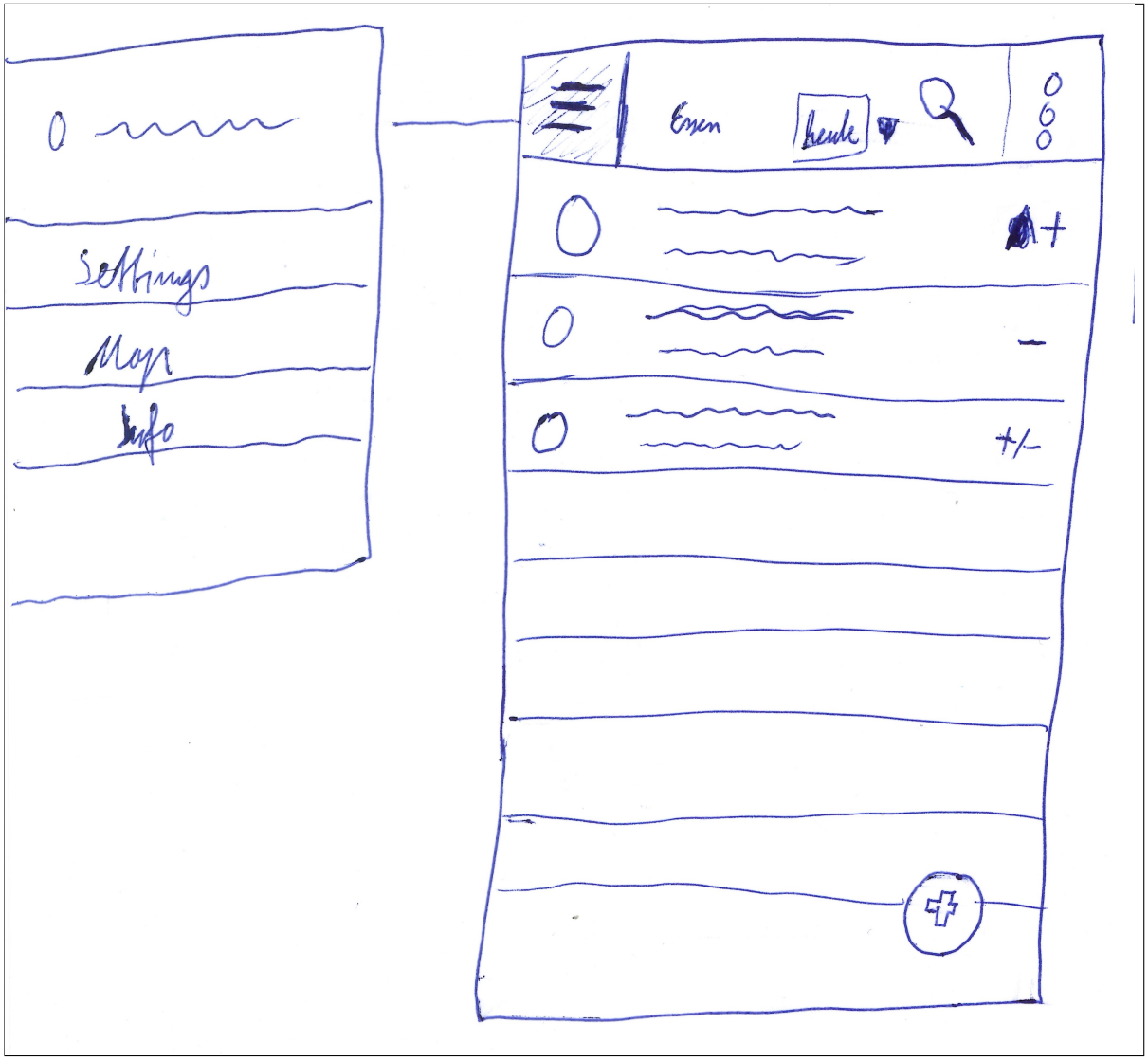

Abbildung 4.5: Main Hamburger Menü Screen

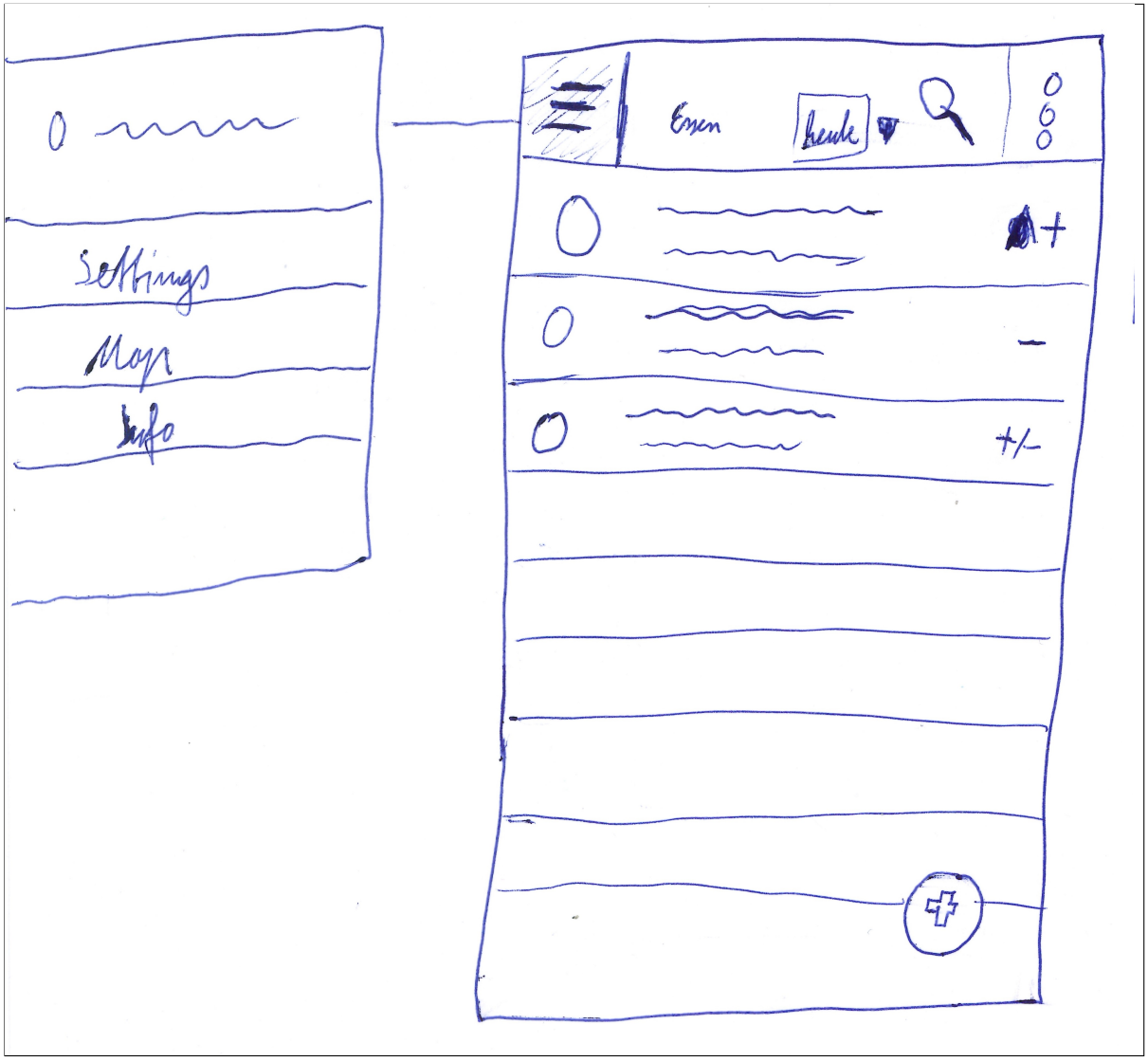

Abbildung 4.6: Karten Screen

5 Umsetzung

5.1 Datenbank

Für die Speicherung der Daten wurde sich für eine SQLite Datenbank entschieden. Das hat mehrere Vorteile. Zum einen lässt sich der Inhalt dieser Datenbank sehr einfach exportieren, um auf einem anderen Gerät wieder eingespielt werden zu können, zum anderen bietet eine Datenbankstruktur viele Möglichkeiten zur Datenanalyse mit schnellem Zugriff ohne wie bisher mit Files arbeiten zu müssen.

5.2 Room

Room wurde als Datenbank Abstraktionsschicht verwendet.

6 Aufgetretene Probleme

1. Aufwändige Integration von Dagger 2 die letztendlich verworfen werden musste.

7 Lessons learned

Dem Parkinsonschen Gesetz folgend wurde die Features während der vorhandenen Zeit umgesetzt. Da während der Appentwicklung die Stundenplan App der Hochschule betreut werden musste, entstand gelegentlich ein Interessen- und Ressourcenkonflikt. Letztendlich wurden die Änderungen an der Schnittstelle leicht gewichtet behandelt, da diese einen Mehrwert für viele Hundert Studenten auf Android und iOS Seite mit sich brachten.

8 Weitere Arbeiten

In diesem Projektabschnitt konnten alle gesteckten Ziele erreicht werden. Da solch ein junges Projekt noch viele Möglichkeiten beinhaltet Funktionen zu verbessern und neue Funktionen einzuführen werden hier mögliche Punkte zur Anregung aufgelistet.

Abbildungsverzeichnis

4.1 Hinzufügen eines Eintrags 5
4.2 Main Screen 6
4.3 Popover Screen 7
4.4 Settings Screen 8
4.5 Main Hamburger Menü Screen 9
4.6 Karten Screen 10