DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Architekturentwurf

Android Eingabesimulation zur Testautomatisierung

von Dennis Meier

[1]

Stand: 15.06.2020

Kurs: TINF18B5

Version	Autor	Änderungsvermerk
1.0	D. Meier	Initiale Fassung

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ADB	Android Debug Bridge
DAO	Data Acess Object
GUI	Graphical User Inteface
IDE	Integrated Development Environment
UML	Unified Modeling Language

1 Einleitung

Bei der Entwicklung von Software muss der Quellcode stetig angepasst und erweitert werden. Gründe hierfür sind beispielsweise bug-fixing oder Updates. Nachdem der Code umgeschrieben wurde, muss allerdings auch jedes Mal die Anwendung erneut getestet werden. Ansonsten könnten neue Bugs übersehen werden oder die Funktion von Erweiterungen nicht gewährleistet werden. Dieser Prozess findet auch bei der Entwicklung für Android Applikationen statt. Ständiges Testen der korrekten Funktionsfähigkeit ist allerdings sehr zeitaufwändig.

Dieses Problem soll durch die Automatisierung von Benutzereingaben gelöst werden. Es soll über eine Desktop Anwendung möglich sein die Benutzereingaben von einem Smartphone aufzuzeichnen und wieder abzuspielen. Des Weiteren soll es möglich sein die aufgenommenen Eingaben zu editieren und abzuspeichern.

Dadurch können wiederkehrende Tests einfach und zeitsparend durchgeführt werden. Des Weiteren können die Eingabeabläufe ständig erweitert werden und so an neue Programmfunktionen angepasst werden.

Es muss eine Grafische Oberfläche für Desktop PCs entwickelt werden, welche auch ohne ausführliche Einweisung bedienbar sein muss. Es muss eine Verbindung zwischen Rechner und Smartphone hergestellt werden, um Eingaben auszulesen und auszuführen. Die Aufgezeichneten Benutzereingaben müssen auf dem Rechner gespeichert werden.

In diesem Dokument wird eine Architektur beschrieben, welche zur schnellen und strukturierten Implementierung der geforderten Funktionen dient.

2 Architekturstrategie

In diesem Kapitel wird genauer auf die Architekturstrategie eingegangen. Das Hauptziel stellt hierbei eine hohe Testabdeckung dar, welche über automatisierte Tests erzielt wird.

2.1 Programmatische Architekturstrategie

Die Testbarkeit sowie die Erweiterbarkeit stehen an vorderster Stelle bei der Planung des Projekts. Daher werden teile des Projekts an passende Entwurfsmuster angepasst. Die Implementierung der einzelnen Entwurfsmuster wird in einem UML-Klassendiagramm dargestellt.

2.2 Humble Object Pattern

Das Humble Objct Pattern kommt dann zum Einsatz, wenn Programmmodule nur schwer testbar sind. Das nachfolgende Projekt besteht hauptsächlich aus einer Grafischen Oberfläche und dem dazugehörigen Backend. Zusätzlich wird noch die ADB Shell verwendet, welche aber nicht getestet wird. Da Tests auf dem Frontend nur schwer durchführbar sind, wird das Humble Object Pattern verwendet. Hierbei wird die Logik des schwer zu testenden Moduls in das so genannte „Humble Object“ ausgelagert. Dadurch wird die Testbarkeit des Frontends und so des Projekts erhöht.

2.3 Data Access Object Pattern

Im Projekt wird eine Datenbank verwendet um die Daten außerhalb der Laufzeit zu speichern. Um die Erweiterbarkeit der Datenquelle zu gewährleisten, wird das Data Acess Object Pattern verwendet. Dies ist ein Entwurfsmuster, welches den Zugriff auf Datenquellen kapselt und diese so austauschbar macht. In der Praxis wird dabei eine Schnittstelle zwischen Programm und Datenquelle geschrieben, die unabhängig vom eigentlichen Quellcode angepasst werden kann. Dadurch wird die Erweiterbarkeit des Projekts erhöht.

3 Statische Sicht – zu persistierende Daten

Damit die Daten auch außerhalb der Laufzeit erhalten bleiben wird im Projekt eine Datenbank verwendet. Im folgenden Kapitel wird die Wahl des Datenbankmodells genauer erläutert.

3.1 Multi-Touch-Protokoll

Ausschlaggebend für den Aufbau des Entity-Relationship-Modells ist das Multi-Touch-Protokoll, welches zunächst erläutert wird. Das Android Betriebssystem, verwendet den Linux Kernel, welcher sehr Identisch zur Desktop Variante ist. Sie verarbeiten unter anderem auch die Benutzereingaben identisch, bei denen das Multi-Touch-Protokoll verwendet wird. Das Protokoll erfordert einen Eingabebericht der bis zu sieben Zeilen besteht. Jede Zeile besteht aus einem Code, der vier Parameter besitzt. Zusammen können so alle Benutzereingaben eines Smartphones gespeichert und wiedergegeben werden.

3.2 Entscheidung Datenbankmodell

Es gibt unterschiedliche Datenbankmodelle, welche unterschiedliche Funktionen bieten, aber doch das gleiche Ziel haben: Daten speichern. Allerdings hat jedes Datenbankmodell seine eigenen Vor- und Nachteile, weshalb im folgenden die Wahl des Datenbankmodells anhand einer Entscheidungsmatrix erläutert wird.

Tabelle 1 zeigt die einzelnen Kriterien und deren Gewichtungen. Die Kompatibilität wird als K.O.-Kriterium festgelegt, da es eine Grundlegende Eigenschaft ist die Datenstruktur in geeigneter Weise abzubilden. Der Einarbeitungsaufwand wird mit *zwei* gewichtet, da für die Implementierung nur ein begrenzter Zeitrahmen vorhanden ist. Ein sehr komplexes Datenbankmodell kann trotz möglicher anderer Vorteile nicht in entsprechender Zeit implementiert werden. Die Performance und der Speicherbedarf werden mit *eins* entsprechend niedrig Gewichtet, da nur kleine Datenmengen erwartet werden. Die Datenintegrität ist eine weitere Wichtige

Anforderung an ein Datenbankmodell, weshalb diese mit *zwei* gewichtet wird.

Kriterium-ID	Kriterium	Bewertungsskala	Gewichtung
KR-1	Kompatibilität	Intervallskala (subjektiv) gering, mittel, hoch	K.O.
KR-2	Einarbeitungsaufwand	Intervallskala (subjektiv) gering, mittel, hoch	2
KR-3	Performance	Intervallskala (subjektiv) gering, mittel, hoch	1
KR-4	Speicherbedarf	Intervallskala (subjektiv) gering, mittel, hoch	1
KR-5	Datenintegrität gewährleistet	Nominalskala, Ja oder Nein	2

Tabelle 1: Kriterien Entscheidungsmatrix Datenbankmodell

Aufgrund der Entscheidungsmatrix fällt die Wahl des Datenbankmodells ein klassisches relationales Modell. In Tabelle 2 ist erkennbar dass dieses Modell durchweg die Kriterien erfüllt und so die meisten Punkte erreicht.

	Relational	Objektorientiert	Hierarchisch	Graphen
KR-1	Hoch	Hoch	Hoch	Gering
KR-2	Gering	mittel	Hoch	Hoch
KR-3	Mittel	Gering	Hoch	Hoch
KR-4	Mittel	Mittel	Mittel	Mittel
KR-5	Nein	Nein	Ja	Ja
	2*3+1*2+1*2	2*2+1*1+1*2	1*2+3*1+2*1+2*1	-
Wertung	10	7	9	-

Tabelle 2: Ergebnis Entscheidungsmatrix Datenbankmodell

3.3 Entity-Relationship-Modell

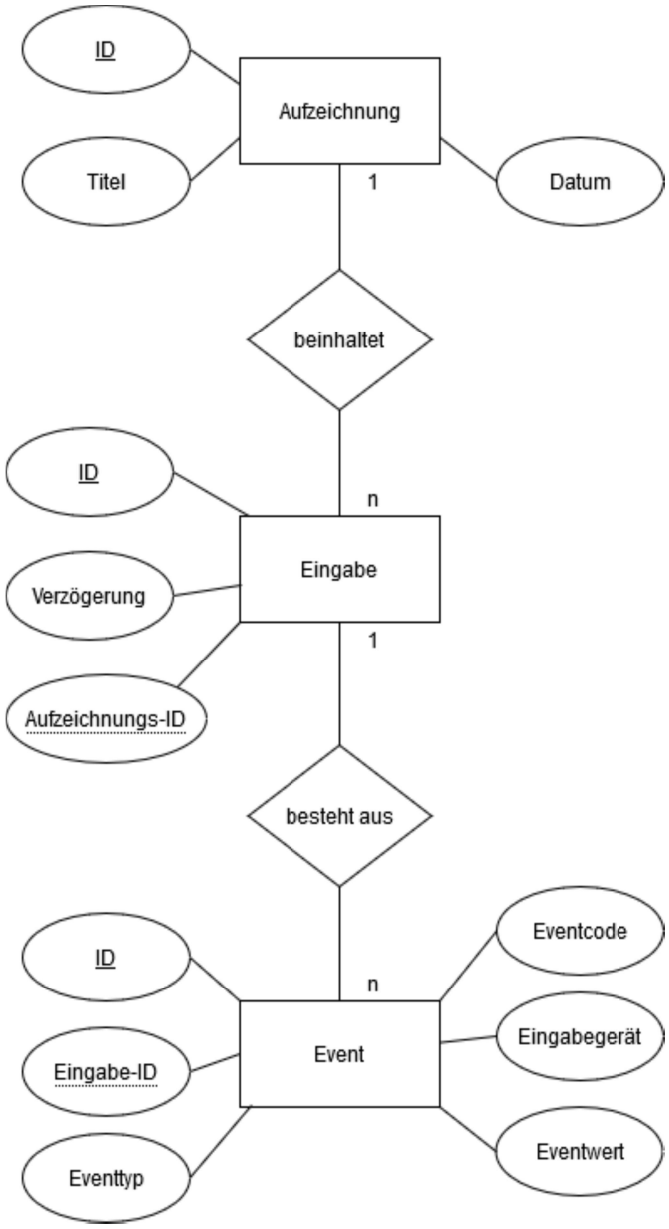

Abbildung 1: ERM

Die Wahl der Entitäten wird wie bereits in Kapitel 2.1 beschrieben auf das Multi-Touch-Protokoll des Linux Kernels zurückgeführt. Für das Eingabesimulationssystem sind die Entitäten Aufzeichnung, Eingabe und Event erforderlich. Eine Aufzeichnung besitzt einen Titel und kann beliebig viele Eingaben beinhalten, welche über einen Fremdschlüssel aufeinander verweisen. Eine Eingabe kann ein einzelne Berührung

auf dem Touchscreen des Smartphones sein. Die Eingaben bestehen jeweils aus mehreren Codes, die ebenfalls über einen Fremdschlüssel gekoppelt sind. Alle Entitäten verwenden IDs als Primärschlüssel.

3.3.1 Beschreibung der Attribute

Entität	Attribut	Beschreibung
Aufzeichnung	Titel	Titel, welcher individuell vergeben werden kann und die Funktion der Aufzeichnung beschreiben soll
Aufzeichnung	Datum	Datum und Uhrzeit an dem die Aufzeichnung getätigt oder zuletzt bearbeitet wurde
Eingabe	Verzögerung	Zeitverzögerung in Millisekunde, bevor die Eingabe erfolgt
Eingab	Aufzeichnungs-ID	Fremdschlüssel um auf die dazugehörige Aufzeichnung zu verweisen
Code	Eingabegerät	Die Bezeichnung für das Android-interne Eingabegerät, z.B. Touchscreen, Laut/Leise-Button, Home-Button
Code	Eventtyp	Typ des Events anhand eines Zahlenwerts, z.B. Änderungen der Axenwerte auf dem touchscreen, Änderung der Schalterstellung
Code	Eventcode	Zuweisung des Wertes anhand eines Zahlenwertes, z.B. Tracking-ID, X-Koordinate, Y-Koordinate
Code	Wert	Beliebiger zahlenwert, der abhängig vom Eventcode

		ist, z.B. 300 als X-Koordinate
Code	Eingabe-ID	Fremdschlüssel um auf die dazugehörige Eingabe zu verweisen

Tabelle 3: Beschreibung der Attribute

3.3.2 Beschreibung der Relationen

Aus dem Entity-Relationship-Modell werden unter Berücksichtigung der Transformationsregeln die Relationen gebildet. Diese werden im folgenden tabellarisch dargestellt. Für die IDs werden durchgehend Integer-Werte verwendet. Spaltenbezeichnungen die fett markiert wurden, dürfen keine null-Werte enthalten.

Aufzeichnung		
Int	varchar(100)	datetime
ID	Titel	**Datum**
1	Aufzeichnung_1	2020-01-01 12:30:00
2	Aufzeichnung_2	2020-01-03 10:00:00

Tabelle 4: Relation - Aufzeichnung

Für den Titel wird ein varchar mit der maximalen Länge von 100 verwendet, da der Titel aus Gründen der Übersichtlichkeit nicht länger sein sollte. Der Titel muss nicht zwingend vorhanden sein, da auch Duplikate erlaubt sind und dadurch keine eindeutige Unterscheidung gegeben ist. Das Datum wird vom System vergeben und darf daher keinen Null-Wert annehmen.

Eingabe		
Int	Int	Int
ID	**Aufzeichnungs-ID**	Verzögerung
1	1	1000
2	1	2000
3	1	3000

Tabelle 5: Relation - Eingabe

Die Verzögerung kann einen beliebigen positiven Ganzzahlwert annehmen. Wenn ein Null-Wert enthalten ist, wird von einer Verzögerung von 0 millisekunden ausgegangen.

Event					
Int	Int	varchar(7)	Int	Int	Int
ID	**Eingabe-ID**	Eingabegerät	Eventtyp	Eventcode	Eventwert
1	1	event0	3	57	0
2	1	event0	3	53	300
3	1	event0	3	54	400
4	1	event0	3	48	5
5	1	event0	3	58	50
6	1	event0	0	2	0
7	1	event0	0	0	0
8	2	event1	3	57	0

Tabelle 6: Relation - Event

Das Eingabegerät ist vom Typ varchar und kann eine maximale Länge von sieben Ziffern haben. Dies wird aus dem Aufbau von „event“ und einer maximal zweistelligen Ziffer abgeleitet. Eventtyp, Eventcode und Eventwert können alle positive Ganzzahlwerte annehmen.

4 Statische Sicht: Objekttypen zur Laufzeit

Im folgenden werden alle alle Objekte zur Laufzeit in einem UML-Klassendiagramm dargestellt

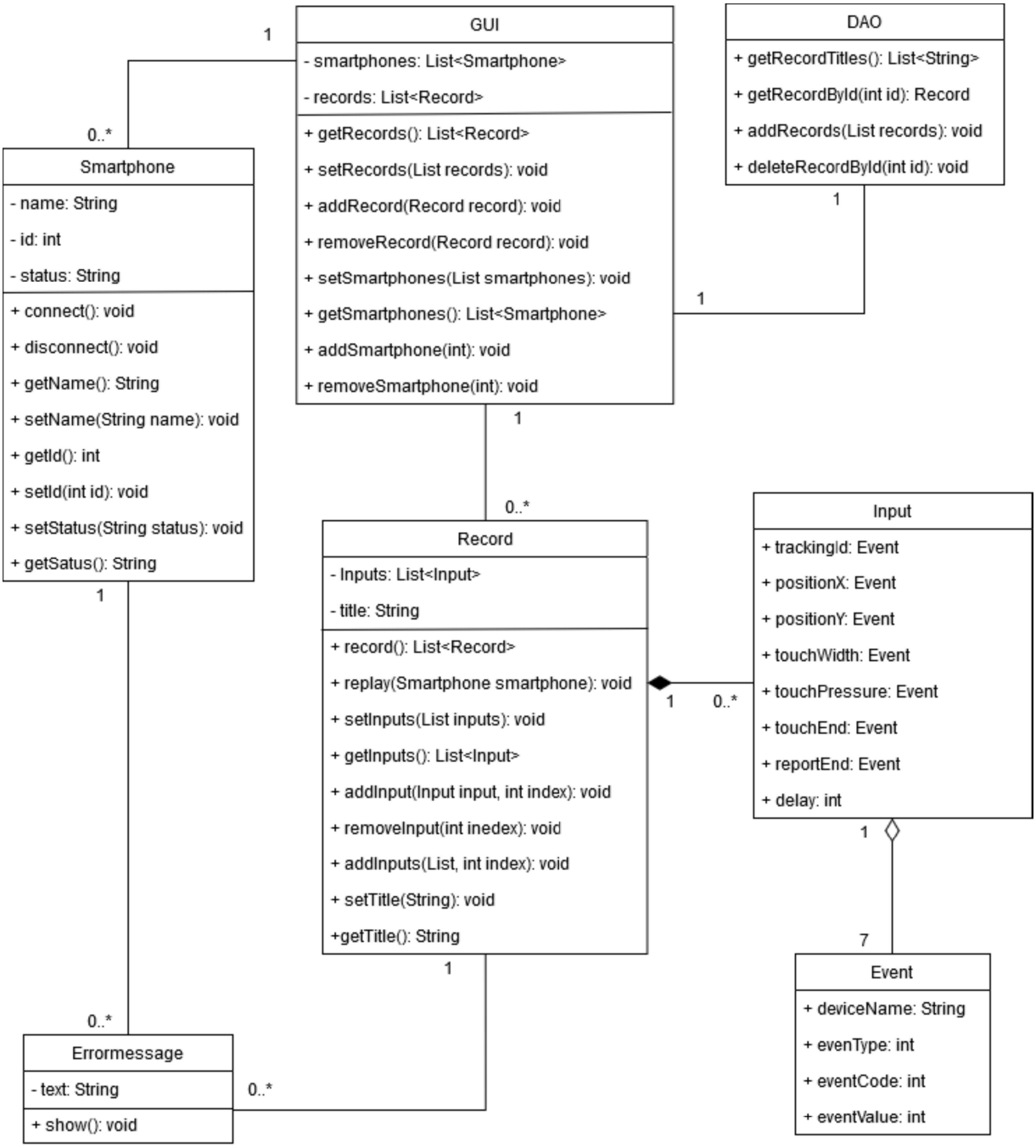

Abbildung 2: UML-Klassendiagramm

Die Hauptkomponente bildet die Grafische Oberfläche (GUI) und das dazugehörige Backend, welche für die Verwaltung des Funktionsumfang zuständig ist. Die GUI besitzt neben einem Titel jeweils eine Liste mit Smartphone- und Aufzeichnungs-Objekten. Die Smartphone-Klasse besitzt Eine Variable für die Bezeichnung, z.B. Samsung Galaxy S9. Die IDs werden automatische beim Verbinden vergeben und verhindern Komplikationen wenn zwei identische Smartphones verbunden werden. Der Status gibt Auskunft ob ein Smartphone verbunden ist, getrennt wurde oder Verbindungsprobleme vorhanden sind. Für das Verbinden und Trennen sind die gleichnamigen Funktionen zuständig. Die Aufzeichnungs-Klasse verfügt über einen Titel um diese voneinander zu unterscheiden und einer Liste mit Eingabe-Objekten. Die Variablen des Eingabe-Objekts werden in Tabelle 7 erläutert.

Variable	Beschreibung
trackingId	ID der Toucheingabe (wichtig, wenn mehrere Eingaben gleichzeitig getätigt werden)
positionX	X Koordinate
positionY	Y Koordinate
touchWidth	breite der toucheingabe in Pixeln
touchPressure	Druck der Toucheingabe
touchEnd	Ende der Toucheingabe
reportEnd	Ende des Berichts
delay	Zeitverzögerung der Eingabe in ms

Tabelle 7: Eingabevariablen

Bis auf die Verzögerung bestehen alle Instanzvariablen aus Event-Objekten. Dieses besitzt Variablen für die Gerätebezeichnung, dem Eventtyp, dem Eventcode und dem Eventwert. (siehe Tabelle 3)

5 Dynamische Sicht

5.1 Funktionseinheiten

Geräteverwaltung

- Verbinden (R1, R2, R3, R5)
- Anzeigen(R4)
  - Trennen (R24)

Aufzeichnen

- Starten (R6, R8)
  - Informationen anzeigen (R7)
  - Stoppen (R9, R10)

Aufzeichnungsverwaltung

- Anzeigen (R11)
  - Löschen (R12)
  - Bearbeiten (R13, R14, R15, R16, R17, R18, R19)

Eingabesimulation

- Starten (R20, R21, R22, R23)

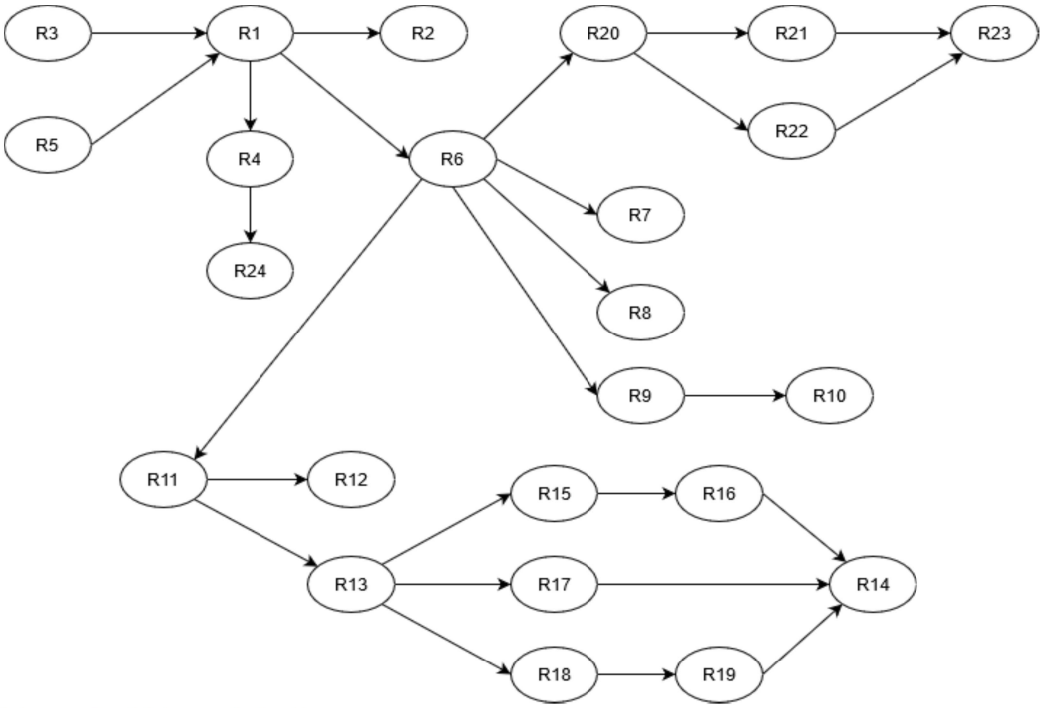

Abbildung 3: Abhängigkeiten

5.2 Prozessbeschreibung für Funktionseinheit Geräteverwaltung

Die Geräteverwaltung kümmert sich um die Verbindung von Smartphone und der Benutzerschnittstelle. Die Verbindung und Kommunikation erfolgt über die Android Debug Bridge (ADB). ADB ist ein Kommandozeilentool, welches nach dem Client-Server-Prinzip aufgebaut ist. Dadurch wird der Dateitransfer und auch der Zugriff auf die interne Kommandozeile ermöglicht. Die Geräteverwaltung listet alle verbundenen Geräte mit Bezeichnung und Status auf. Kommt es zu Fehlern werden entsprechende Fehlermeldungen ausgegeben.

5.3 Prozessbeschreibung für Funktionseinheit Aufzeichnen

Beim Aufzeichnen wird zunächst das Gerät ausgewählt von welchem die Benutzereingaben aufgezeichnet werden sollen. Nachdem die Aufzeichnung gestartet ist, werden alle Benutzereingaben auf der GUI zeitgleich dargestellt. Nachdem die Aufzeichnung beendet wurde muss der Benutzer einen Titel vergeben. Anschießend wird diese in der Datenbank abgespeichert und steht so dem Benutzer auch außerhalb der Laufzeit zur Verfügung.

5.4 Prozessbeschreibung für Funktionseinheit Aufzeichnungsverwaltung

Die Aufzeichnungsverwaltung kümmert sich um die Verwaltung aller bisher abgespeicherten Aufzeichnungen. Diese werden zunächst aus der Datenbank geladen und aufgelistet. Der Benutzer kann diese dort Löschen und Bearbeiten. Beim Bearbeiten stehen mehrere Funktionen zur Auswahl. Zum einen kann die Verzögerung jeder Eingabe verändert werden. Des Weiteren können komplette Aufzeichnungen ineinander integriert und auch einzelne Eingaben entfernt werden. Der Prozess ist anhand eines UML-Aktivitätsdiagramm in Abbildung 4 dargestellt.

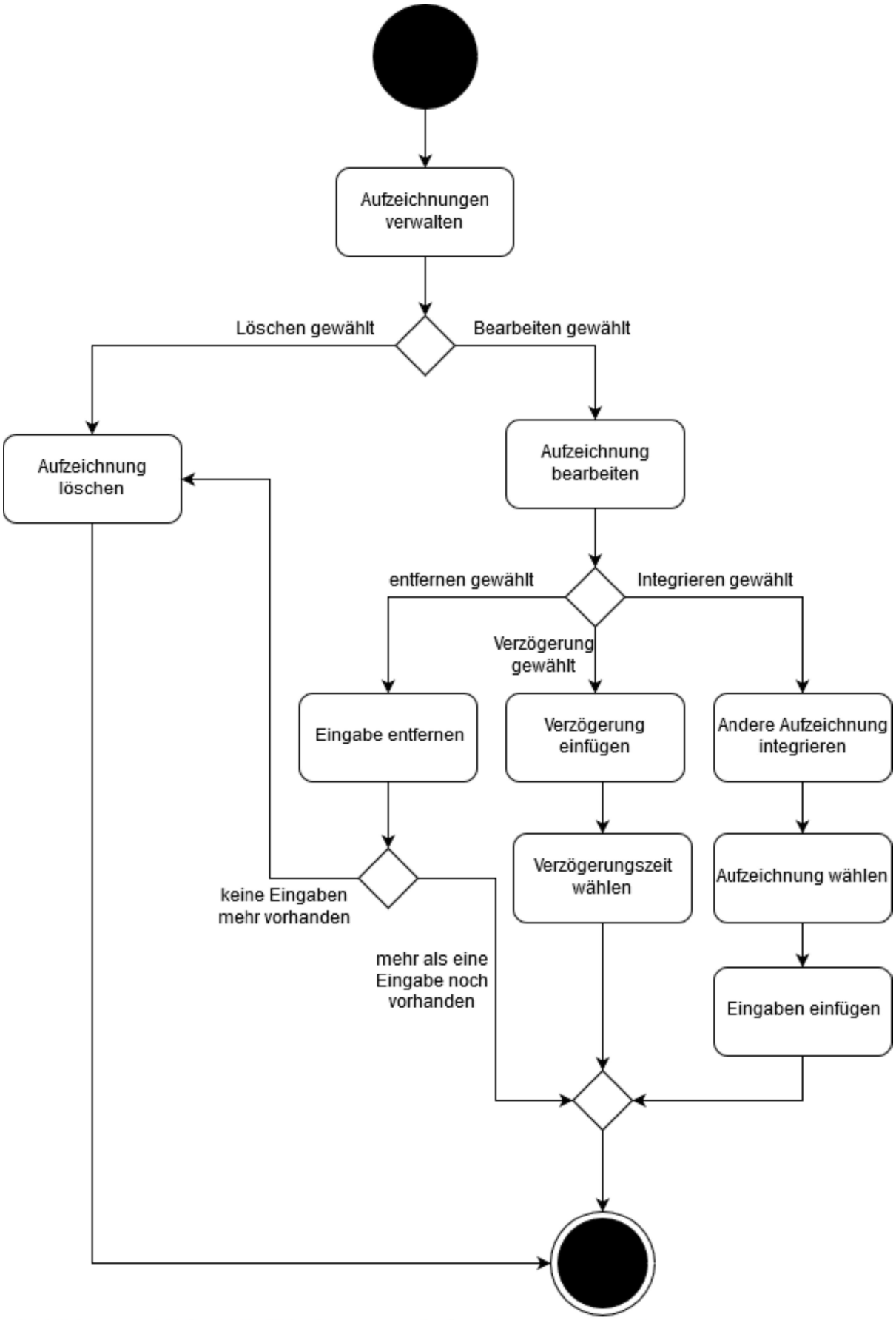

Abbildung 4: UML-Aktivitätsdiagramm Aufzeichnungsverwaltung

5.5 Prozessbeschreibung für Funktionseinheit Eingabesimulation

Bei der Eingabesimulation wird zunächst eine Aufzeichnung aus der Liste der verfügbaren Aufzeichnungen gewählt. Falls mehrere Smartphones mit dem System verbunden sind, wird eine Liste zum auswählen des Zielgerätes angezeigt. Anschießend kann der Benutzer wählen wie oft die Aufzeichnung abgespielt werden soll. Während der Wiedergabe kann der Benutzer den Prozess auch abbrechen. Bei Beendigung wird eine entsprechende Nachricht angezeigt.

6 Auswahl des Technologiestacks

6.1 Programmiersprache

Da es eine Vielzahl an möglichen Programmiersprachen gibt, wird die Wahl mit Hilfe einer Entscheidungsmatrix getroffen. Im folgenden sind die Entscheidungskriterien aufgelistet und gewichtet.

Kriterium-ID	Kriterium	Bewertungsskala	Gewichtung
KR-1	Einarbeitungsaufwand	Intervallskala (subjektiv) gering, mittel, hoch	3
KR-2	GUI-Bibliothek vorhanden	Nominalskala, Ja oder Nein	K.O.
KR-3	ADB-Bibliothek vorhanden	Nominalskala, Ja oder Nein	2
KR-4	Lesbarkeit	Intervallskala (subjektiv) gering, mittel, hoch	1

Tabelle 8: Kriterien - Programmiersprache

Der Einarbeitungsaufwand wird mit *drei* gewichtet, da es im Rahmen des Projekts nicht möglich ist, sich in eine komplett neue Programmiersprache einzuarbeiten. Das Vorhandensein einer geeigneten GUI-Bibliothek, bzw. allgemein die Möglichkeit eine grafische Oberfläche zu generieren wird als K.O.-Kriterium gewertet, da dies eine essenzielle Anforderung an das Projekt ist. Eine geeignete Bibliothek, welche an die ADB-Shell anknüpft wäre wünschenswert, weshalb diese mit *zwei* gewichtet wird. Da die Lesbarkeit einer Programmiersprache nicht im Vordergrund steht, wird eine Gewichtung von *eins* gewählt.

	Python	Java	C#
KR-1	Gering	Gering	Mittel
KR-2	Ja	Ja	Ja
KR-3	Ja	Ja	Nein
KR-4	hoch	Mittel	Mittel
	3*3+1*2+3*1	3*3+1*2+2*1	2*3+2*1
Wertung	14	13	8

Tabelle 9: Ergebnis - Programmiersprache

Bei der Wahl der Programmiersprachen wird die Rangliste der populärsten Programmiersprachen von RedMonk hinzugezogen. Die Liste basiert auf den Daten von GitHub und Stack Overflow vom ersten Quartal 2020[2] . Neben den gewählten Programmiersprachen sind noch JavaScript und PHP vertreten, welche allerdings nicht in für Desktop Anwendungen ausgelegt sind und so nicht mit in die Entscheidungsmatrix mit aufgenommen wurden. Da die Programmiersprache Python alle Kriterien erfüllt hat und so die meisten Punkte erreicht hat, wurde diese für dieses Projekt ausgewählt. Da es zwei Versionen von Python gibt, wird die neuere und zukunftssichere Version Python 3 verwendet.

6.2 Entwicklungsumgebung

Es gibt unterschiedliche Entwicklungsumgebung die alle verschiedene Anforderungen erfüllen. Um die passendste Entwicklungsumgebung für das Projekt zu finden, wird die Wahl im folgenden mit einer Entscheidungsmatrix getroffen.

Kiterium-ID	Kriterium	Bewertungsskala	Gewichtung
KR-1	Kostenlos	Nominalskala, Ja oder Nein	3
KR-2	Syntax Highlighting	Intervallskala (subjektiv) gering, mittel, hoch	1
KR-3	Debugger -funktionsumfang	Intervallskala (subjektiv) gering, mittel, hoch	2
KR-4	Kompilieren in Anwendung möglich	Nominalskala, Ja oder Nein	2
KR-5	Syntaxprüfung	Nominalskala, Ja oder Nein	2
KR-6	Einarbeitungsaufwand	Intervallskala (subjektiv) gering, mittel, hoch	1

Tabelle 10: Kriterien - Entwicklungsumgebung

Besonders Wichtig bei der Wahl der Entwicklungsumgebung ist, dass diese kostenlos zur Verfügung steht. Deshalb wird dieser Aspekt mit *drei* gewichtet. Da Syntax Highlighting nicht zwingend notwendig ist wird dieses Kriterium nur mit *eins* Gewichtet. Da das Vorhandensein eines Debuggers nicht für eine Bewertung ausreicht, wird hierbei eine Intervallskala verwendet, die Auskunft über den Funktionsumfang und Bedienung gibt. Das Kompilieren aus der Anwendung heraus und die Syntaxprüfung sind Grundlegende Funktionen, weshalb diese mit *zwei* gewichtet werden. Der Einarbeitungsaufwand wird lediglich mit *eins* gewichtet, da allgemeine Kenntnisse bereits vorhanden sind.

	IDLE	PyCarm	Eclipse
KR-1	Ja	Ja	Ja
KR-2	Mittel	Hoch	Hoch
KR-3	Mittel	Hoch	Hoch
KR-4	Ja	Ja	Ja
KR-5	Ja	Ja	Ja
KR-6	Mittel	Gering	Hoch
	1*3+2*1+2*2+1*2+1*2+2*1	1*3+3*1+3*2+1*2+1*2+3*1	1*3+3*1+3*2+1*2+1*2+1*1
Wertung	15	19	17

Tabelle 11: Ergebnis - Entwicklungsumgebung

Bei den möglichen Entwicklungsumgebungen fiel die Wahl zum einen auf IDLE, einer leichtgewichtigen Anwendung, die bereits im Python Installer enthalten ist. Des Weiteren wurde PyCharm gewählt, die als populärste Python Entwicklungsumgebung gilt. Eclipse ist eine populäre IDE die ursprünglich für Java genutzt wurde, welche allerdings mit Plugins auch auf weitere Programmiersprachen erweitert werden kann. Auf Grundlage der Entscheidungsmatrix fällt die Wahl der geeignetsten Entwicklungsumgebung auf PyCharm. Dank der Sonderstellung als Student kann hier eine Lizenz für die Vollversion kostenlos für ein Jahr erworben werden. Alternativ kann aber auch die Community-Version verwendet werden, welche auf diverse Funktionen verzichtet, die aber nicht zwingend für das Projekt benötigt werden.

6.3 Datenbankmanagementsystem

Als Datenbankmanagementsystem wird MySQL verwendet. Der MySQL-Datenbankserver ist quelloffen und bestens für relationale Datenbankmodelle geeignet. Weitere Vorteile sind die Schnelligkeit und Zuverlässigkeit der Anwendung. Zusätzlich sind bereits Kenntnisse über MySQL vorhanden, wodurch ein leichter Einstieg gewährleistet wird.

6.4 Testtool

Die im Projekt die Programmiersprache Python verwendet wird, wird das „Unit testing framework“ verwendet. Dabei handelt es sich um eine frei zur Verfügung stehenden Python Bibliothek. Die Erweiterung hat sich vom bekannten Framework „JUnit“ inspirieren lassen und ist daher ähnlich aufgebaut. Unterstützt wird die Testautomatisierung und weitere nützliche Funktionen.

7 Verteilungsdiagramm

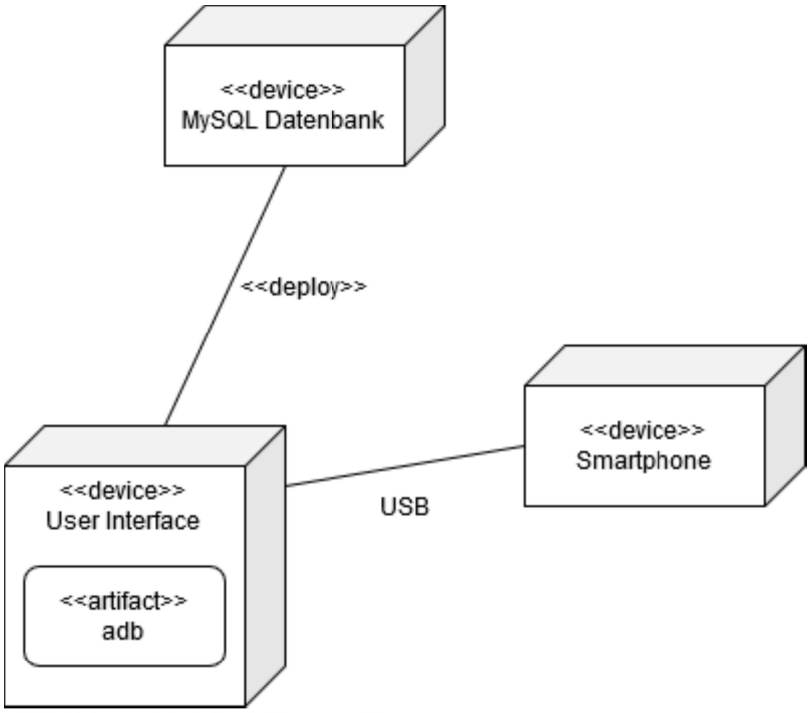

Abbildung 5: Verteilungsdiagramm

Es ist zu sehen, dass das Smartphone über eine USB-Verbindung mit dem User Interface und dem dazugehörigen Backend verbunden ist. Der Zugriff auf die Kommandozeile des Smartphones erfolgt über die Android Debug Bridge. Zusätzlich steht eine MySQL Datenbank lokal zur Verfügung um die anfallenden Daten zu speicern.

8 Fazit

In diesem Architekturdokument wurde eine mögliche Architektur entwickelt und aufgezeigt, welche die Anforderungen des dazugehörigen Anforderungsdokument umsetzt. Wichtig bei der Entwicklung der Architektur war der Aspekt der Erweiterbarkeit. Diese sollte jederzeit und auch für dritte gegeben sein. Des weiteren wurden ausschließlich quelloffene oder kostenfreie Technologien ausgewählt um die Abhängigkeit von Drittanbietern zu minimieren.

Literaturverzeichnis

[1] https://de.wikipedia.org/wiki/Datei:Android_robot.svg

[2] https://redmonk.com/sogrady/2020/02/28/language-rankings-1-20/