Inhaltsverzeichnis

[Einleitung 2](#_Toc474778521)

[Hinweise zur Lesbarkeit 2](#_Toc474778522)

[Rechtliche Hinweise 2](#_Toc474778523)

[Abkürzungen 2](#_Toc474778524)

[Abbildungen 3](#_Toc474778525)

[Tabellen 3](#_Toc474778526)

[1. Projektbeschreibung 3](#_Toc474778527)

[1.1 Vorstellung 3](#_Toc474778528)

[1.2 Projektumfeld 4](#_Toc474778529)

[1.3 Projektziel 4](#_Toc474778530)

[1.4 IST-Zustand 5](#_Toc474778531)

[1.5 SOLL-Zustand 5](#_Toc474778532)

[2. Projektplanung 5](#_Toc474778533)

[2.1 Zeitplanung 5](#_Toc474778534)

[3. Projektdurchführung 6](#_Toc474778535)

[3.1 Erstellen der Level-Klasse 6](#_Toc474778536)

[3.2 Erstellen der Main Activity 6](#_Toc474778537)

[3.3 Erstellen der Grafiken 7](#_Toc474778538)

[3.4 Anzeigen der Grafiken im Spiel 8](#_Toc474778539)

[3.5 Erstellen der Datenbank-Klasse 8](#_Toc474778540)

[3.6 Erstellen des Logins und der Spielmodus-Auswahl 9](#_Toc474778541)

[3.7 Nutzen der Datenbank für Login 10](#_Toc474778542)

[3.8 Implementieren der Level-Auswahl 10](#_Toc474778543)

[3.9 Hinzufügen der Zeitmessung 11](#_Toc474778544)

[3.10 Anpassen des Layouts 11](#_Toc474778545)

[3.11 Erweiterung der Spielmodus-Auswahl auf 2 Spieler 11](#_Toc474778546)

[3.12 Implementieren der Highscore-Activity 12](#_Toc474778547)

[4. Projektergebnis 12](#_Toc474778548)

[4.1 Fehlerbehebung 12](#_Toc474778549)

[4.2 Abweichungen 13](#_Toc474778550)

[5. Anhang 14](#_Toc474778551)

[5.1 Screenshots 14](#_Toc474778552)

[5.2 Github-Accounts 15](#_Toc474778553)

Einleitung

Diese Dokumentation wurde im Rahmen einer schulischen Projektarbeit an der Berufsschule in Freising im Fach Anwendungsentwicklung und Programmierung zur Berufsausbildung zum Fachinformatiker für Anwendungsentwicklung erstellt. Sie beinhaltet die Erstellung einer Applikation für Smartphones mit den hierbei aufgekommenen Problemen und deren Lösungsansätzen. Ziel des Projektes ist es, die Lerninhalte des Unterrichts zu vertiefen und ein realitätsbezogenes Projekt umzusetzen.

Katharina Ambrosch, Robert Kais

# Hinweise zur Lesbarkeit

Zur besseren Lesbarkeit wurde der Quellcode durch eine gesonderte Schriftart hervorgehoben.

# Rechtliche Hinweise

Die Spiele-App sowie die dazugehörigen Dokumentations-, Präsentations- sowie alle weiteren Unterlagen sind geistiges Eigentum von Ambrosch Katharina und Kais Robert. Diese Dokumentation darf ohne Einverständnis der o. g. Personen weder vollständig noch auszugsweise kopiert, übertragen, vervielfältigt, veröffentlicht, geändert oder dupliziert werden. Ungeachtet der o. g. Bestimmungen sind Lehrer der Berufsschule Freising dazu berechtigt, zum schulischen Gebrauch die Dokumentation zu veröffentlichen, vorausgesetzt es bleibt ersichtlich, dass das Urheberrecht bei o. g. Personen liegt.

# Abkürzungen

Folgende Abkürzungen tauchen in der Dokumentation auf:

* AEP – Anwendungsentwicklung und Programmierung
* App – Applikation
* SP – Service Pack
* FMG – Flughafen München GmbH
* Abb. - Abbildung

# Abbildungen

Abb. 1 – Zeitplanung  
Abb. 2 – Level-Boden  
Abb. 3 – Spielfigur  
Abb. 4 – Crate Maze-Logo  
Abb. 5 – Button für Einspieler-Modus  
Abb. 6 – Code-Darstellung Level 5  
Abb. 7 – grafische Darstellung Level 5  
Abb. 8 – onCreate()-Methode (DatabaseManager)  
Abb. 9 – Skizze Levels  
Abb. 10 – Screenshot Login  
Abb. 11 – Skizze Spielmodus-Auswahl  
Abb. 12 – Screenshot Spielmodus-Auswahl  
Abb. 13 – Screenshot Level-Auswahl

# Tabellen

Tabelle 1 – Hardware  
Tabelle 2 – Notebook 1  
Tabelle 3 – Notebook 2  
Tabelle 4 – Software  
Tabelle 5 – Level-Definitionen

# 1. Projektbeschreibung

## 1.1 Vorstellung

Das Projekt wurde von Ambrosch Katharina und Kais Robert durchgeführt. Beide machen eine Ausbildung zum Fachinformatiker für Anwendungsentwicklung und sind im 3. Lehrjahr.

Ambrosch Katharina arbeitet bei der Flughafen München GmbH. Die FMG inkl. der Töchter umfasst in etwa 8.000 Mitarbeiter, davon arbeiten ca. 250 in der IT. Der Flughafen wurde am jetzigen Standort am 17.05.1992 eröffnet. Der Geschäftsführer ist Dr. Michael Kerkloh. 2015 wurde der Flughafen „Franz Josef Strauß“ zum ersten 5-Sterne-Airport Europas gekürt. In Deutschland zählt er zum zweitgrößten Flughafen. 2016 wurde ein neuer Passagierrekord mit 42 Mio. Fluggästen erreicht.

Kais Robert arbeitet bei der com2 Communications & Security GmbH. Die Firma besteht aus 14 Personen, wovon acht der IT angehören. Die com2 existiert seit 1992 und setzt auf maßgeschneiderte, sichere IT-Kommunikationslösungen. Die Geschäftsführer sind Boehlke Axel und Kleibömer Thomas.

## 1.2 Projektumfeld

Das Projekt wurde an der staatlichen Berufsschule in Freising umgesetzt.

Zur Realisierung des Projektes benötigte Mittel:

|  |  |
| --- | --- |
| Hardware | Beschreibung |
| Motorola XT1072 (Android 6.0) | Zum regelmäßigen Testen der App |
| Notebook 1 (siehe Tabelle 2) | Zur Implementierung bei Katharina |
| Notebook 2 (siehe Tabelle 3) | Zur Implementierung bei Robert |

*Tabelle 1*

|  |  |
| --- | --- |
| Notebook 1 | Beschreibung |
| Prozessor | Intel Core i5-4310M CPU 2,70 GHz |
| Arbeitsspeicher | 8 GB |
| Betriebssystem | Windows 7 Enterprise SP 1 (64 Bit) |

*Tabelle 2*

|  |  |
| --- | --- |
| Notebook 2 | Beschreibung |
| Prozessor | Intel Core i7 CPU 1,73 GHz |
| Arbeitsspeicher | 4 GB |
| Betriebssystem | Windows 8.1 Pro (64 Bit) |

*Tabelle 3*

|  |  |
| --- | --- |
| Software | Beschreibung |
| Android Studio | Programm zur Android-Implementierung |
| Paint.net | Programm zur Erstellung der Grafiken |

*Tabelle 4*

Ansprechperson bei Fragen zur Implementierung / zu Android-Studio:

Albrecht Pfleiderer, Lehrer im Fach AEP

## 1.3 Projektziel

Hauptziel des Projekts ist eine funktionierende Spiele-App für das Android-Betriebssystem. Das Spiel ist eine Art Labyrinth, in dem man Kisten geschickt aus dem Weg schieben muss, um an den Zielpunkt zu gelangen. Verwirklicht wird dies im Zweispieler-Modus. Mit einem Login kann man sich in der App anmelden und bei Bedarf über das Netzwerk (mit zwei Smartphones) gegeneinander spielen.

## 1.4 IST-Zustand

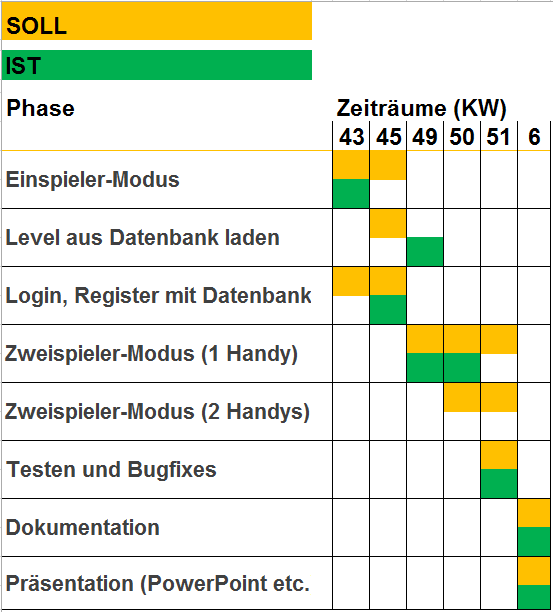
Derzeit steht kein Spiel als Android App zur Verfügung. Es sind bisher keine Programcodes dafür vorhanden.

## 1.5 SOLL-Zustand

Es soll eine funktionierende Spiele-App existieren. Diese soll mit einer Datenbank verknüpft sein, um einen Login bzw. eine Neuanmeldung zu ermöglichen und den Highscore zu speichern. Mit dem Zweispieler-Modus soll es möglich sein, gegeneinander über ein bzw. über zwei mobile Geräte spielen zu können.

# 2. Projektplanung

## 2.1 Zeitplanung



*Abb. 1*

# 3. Projektdurchführung

## 3.1 Erstellen der Level-Klasse

Jede Grafik im Level erhält eine Nummer, um das Layout richtig darstellen zu können. Folgende Definitionen wurden für die Darstellung eines Levels festgelegt.

|  |  |
| --- | --- |
| Nummer | Grafik |
| 0 | Boden |
| 1 | Mauer |
| 2 | Kiste |
| 3 | Startpunkt |
| 4 | Zielpunkt |

*Tabelle 5*

Diese Definition wird im Code mithilfe eines Enumerator namens TILES realisiert. Zum darstellen und Laden der Levels wird eine Klasse mit dem Namen Level erstellt. Um ein Level-Objekt zu initialisieren, muss die Methode LoadLevel() aufgerufen werden. Diese benötigt einen String, welcher 81 Zeichen lang ist und aus Nummern besteht, welche wie oben beschrieben, Grafiken repräsentieren. Die Nummern aus dem String werden in ein multidimensionales Array von der Größe 9 \* 9 gespeichert. Man kann mit der GetTile() -Methode einen Wert aus diesem Array von einer bestimmten Position bekommen, um beispielsweise zu überprüfen, ob es sich um eine Mauer handelt. Die Methode UpdateLevel() wird verwendet, um einen Wert im Array zu verändern, wenn zum Beispiel eine Kiste verschoben wird. Mit den Methoden getPlayerX() und getPlayerY() bekommt man die momentane Position des Spielers und mit SetPlayer() kann man diesen an eine andere Position setzen.

## 3.2 Erstellen der Main Activity

Zu Beginn wird in der Main Activity das Spiel realisiert, wofür erst einmal das Layout festgelegt wird. Hierfür werden in der activity\_main.xml mithilfe von Android Studio ein Text View oben in der Mitte erstellt. Text Views werden benutzt, um Text anzeigen zu lassen. Sie müssen eine ID bekommen, damit man sie im Code referenzieren kann. Zusätzlich werden vier Pfeile in Form von Image-Buttons für alle Richtungen an den Rändern platziert. Die Grafiken hierfür werden von Android Studio zur Verfügung gestellt. Auch diese benötigen IDs und können anders als Views gedrückt werden. Im Code werden zunächst in der vordefinierten onCreate() -Methode die Pfeile sowie die Text View initialisiert. Zusätzlich wird ein Level-Objekt erstellt und für den Anfang ein Test-Level geladen. Um auf das Drücken eines Pfeiles reagieren zu können, muss die Klasse View.OnClickListener implementiert werden. In der nun erforderlichen onClick() -Methode kann mithilfe der IDs zwischen den Pfeilen unterschieden werden. Jeder dieser Pfeile ruft die private Move() -Methode mit verschiedenen Parametern auf, welche einen Integer Wert von -1 bis 1 haben dürfen. Durch diese Parameter wird die Richtung festgelegt, in die der Spieler gehen wird. Innerhalb der Move() -Methode wird die Position des Spielers innerhalb des Levels abgefragt und überprüft, ob das Feld zu dem gegangen wird, eine Kiste hat, das Ziel oder frei ist. Im ersten Fall wird auch das danach folgende Feld überprüft und bewegt den Spieler sowie die Kiste nur, wenn es frei ist. Im zweiten Fall wird das Spiel beendet. Der Spieler wird auf das neue Feld gesetzt, falls der dritte Fall eintrifft. Falls keiner der Fälle zutrifft, also das nächste Feld eine Mauer ist, passiert nichts. Zum Schluss werden die Nummern jedes Feldes in einen String geschrieben und im Text View ausgegeben. Startet man das Spiel, kann man alle Nummern sehen, welche später von Grafiken repräsentiert werden sollen. Wird ein Pfeil betätigt, verändern sich die Zahlen entsprechend.

## 3.3 Erstellen der Grafiken

Alle Grafiken, die im Spiel zu sehen sind, werden selbst erstellt. Hierzu dient das Programm „paint.net“. Als Format eignet sich .PNG, da die Qualität besser ist, als beispielsweise bei .JPEG. In den Levels werden für die Mauern sowie für die zu verschiebenden Kisten Grafiken benötigt. Auch der Boden, der Zielpunkt und die Spielfigur müssen erstellt werden. Bei der Figur ist zu beachten, dass um sie herum die gleiche Farbe ist, die der Boden des Levels auch hat.



*Abb. 2 Abb. 3*

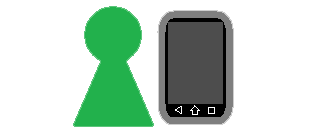
Die Richtungspfeile, welche seitlich bei den Levels zur Steuerung dienen, werden ohne weißer Hintergrundebene erstellt, damit keine farblichen Abweichungen mit dem Untergrund auftreten können.

Beim Login und bei der Auswahl des Spielmodus befindet sich oben ein Schriftzug, der den Namen des Spiels anzeigt. Dieser ist in der Schriftart „Showcard Gothic“ als Grafik abgespeichert mit weißem Hintergrund. Um Farbabweichungen zu vermeiden, wird bei der App-Oberfläche der Hintergrund auch auf weiß gestellt.

Für das App-Logo, das im Menü des Smartphones angezeigt wird, dienen die Anfangsbuchstaben „C“ und „M“ und die bereits erstellte Kiste. Hier darf es keine Hintergrundebene geben. So ist es möglich, dass das Bild überall platziert werden kann ohne farbliche Abweichungen. Auf folgender Abbildung wird dies erkennbar.



*Abb. 4*

Die Buttons zur Auswahl des Spielmodus sind ebenfalls Grafiken ohne eine Hintergrundebene. Es gibt jeweils Spielfiguren und ein bzw. zwei Smartphone-Abbildungen. Diese Bilder werden mithilfe von Formen in paint.net erstellt. Beispielsweise besteht eine Spielfigur aus einem Kreis und einem darüber gelegten Dreieck.

*Abb. 5*

## 3.4 Anzeigen der Grafiken im Spiel

Das Hinzufügen der Grafiken ins Spiel wird in der Main Activity implementiert. Dafür wird zuerst die zuvor erstellte Text View gelöscht und stattdessen 81 Image Views erstellt. Diese werden zentriert in einer 9 \* 9 Box angereiht. Die Pfeil-Grafiken, welche von Android Studio zur Verfügung gestellt werden, werden durch die selbst erstellten ersetzt. Image Views sind ähnlich zu Image-Buttons, indem sie Grafiken anzeigen. Allerdings können sie nicht gedrückt werden. Es müssen noch die neuen IDs und die Grafik für den Boden als Standard gesetzt werden. Im Code werden die neuen Image Views mithilfe eines 9 \* 9 Arrays initialisiert. Anstelle des Strings, der die Nummern aus dem Level bekommen hat, wird eine neue Methode namens DrawLevel() aufgerufen. In dieser werden die Nummern für jedes Feld im Level abgefragt und die Grafik des Image Views, der an derselben Stelle ist, zu der Grafik gesetzt, die der Nummer entspricht.

## 3.5 Erstellen der Datenbank-Klasse

In diesem Projekt wird mit der SQLite-Datenbank gearbeitet. Zunächst wird eine eigene Klasse erstellt, die DatabaseManager genannt wird. Diese erbt von SQLiteOpenHelper. Am Anfang wird die Datenbank selbst erzeugt mit folgendem Befehl.

private SQLiteDatabase sqldb;

Dann werden Klassenvariablen deklariert und initialisiert. Beispielsweise benötigt man den Datenbanknamen.

private static final String DB\_NAME = "Crate\_Maze\_db";

Dieser, sowie die Datenbank-Version und die beiden Tabellen „player“ und „level“ werden als private deklariert. Zudem werden zwei SELECT-Befehle auf die Tabellen initialisiert. Der Aufbau ist gleich den privaten Klassenvariablen von oben, nur dass diese beiden jetzt public sind.

Anschließend werden die Spalten für die beiden Tabellen definiert. Für die „player“-Tabelle gibt es Spalten für die ID, den Namen, des Passworts und dem aktuellen Level. Bei der „level“-Tabelle benötigt man die ID, den Level-Aufbau, die Zeit und den Highscore. Bei den ID-Namen ist zu beachten, dass ein Unterstrich vorangeführt ist, da einige Klassen der Android-API dies so erwarten.

In der Methode onCreate werden die Tabellen in der Datenbank erstellt. Es werden jeweils ein String Player und ein String Level initialisiert. Der Aufbau anschließend ist wie bei MySQL.

CREATE TABLE <tabellenname> (<spaltenname> <datentyp>);

Nach der Erstellung werden die Tabellen mit der Methode execSQL() abgeschickt. Für die vollständige Methode siehe Anhang, Abb. 7.

Beim Aufruf der Methode close() wird die Datenbank geschlossen. Der Konstruktor wird von der Mutterklasse übernommen mit dem super-Befehl. Die Methode updateRecord ist dazu da, um Daten in der Datenbank zu aktualisieren. Über insertRecord() werden neue Daten in die Datenbank geschrieben. Hier wird verglichen, ob es sich bei dem neuen Eintrag um die Level- oder die Player-Tabelle handelt.

## 3.6 Erstellen des Logins und der Spielmodus-Auswahl

Da eine der Voraussetzungen für dieses Projekt die Erstellung eines Logins erfordert, wird die aktuelle Main Activity zur Game Activity und eine neue Main Activity wird erstellt, in der ein Eingabefeld sowohl für den Benutzernamen als auch für ein Passwort erstellt wird. Beides sind Edit Texts wobei für Passwortfeld der inputType auf [textPassword] gestellt wird, damit dieses bei der Eingabe als \*-Symbole angezeigt wird. Außerdem werden zwei Text Views erstellt, damit der Benutzer weiß, in welches Feld welche Angabe gehört. Ein Image View für das Logo und drei Buttons zum Login, Registrieren und Beenden des Spiels werden ebenfalls hinzufügt. Auch hier werden die IDs für alle Elemente gesetzt. Die erstellten Elemente werden im Code initialisiert und wie zuvor auch wird für die Buttons der OnClickListener implementiert. Es werden zunächst nur ein Test-Benutzername und ein Passwort zum Überprüfen benutzt, anstelle die Datenbank anzuknüpfen. Das Vergleichen der eingegebenen Daten mit den Testdaten wird nur erfolgen, wenn der Login-Button gedrückt wird. Außerdem startet dieser die Menu Activity. Für den Button, der das Spiel beendet, wird nur die finish() -Methode aufgerufen. Der Registrierungs-Button bleibt zunächst noch ungenutzt.

Für die Spielmodus-Auswahl wird die Menu Activity erstellt, in welcher, wie auch schon bei der Main Activity, das Logo eingefügt wird. Zusätzlich werden zwei Buttons für den Highscore und den Logout erstellt. Zum Schluss werden noch drei Image-Buttons eingefügt, welche die Grafiken für die verschiedenen Spielmodi bekommen. Wie für jede andere Activity auch, werden die IDs gesetzt und alle Elemente im Code initialisiert. Der Logout-Button ruft nur die finish() -Methode auf, damit man zur Main Activity zurückkehrt. Der Image-Button für den Einspieler-Modus startet die Game Activity und die restlichen bleiben erst einmal ungenutzt.

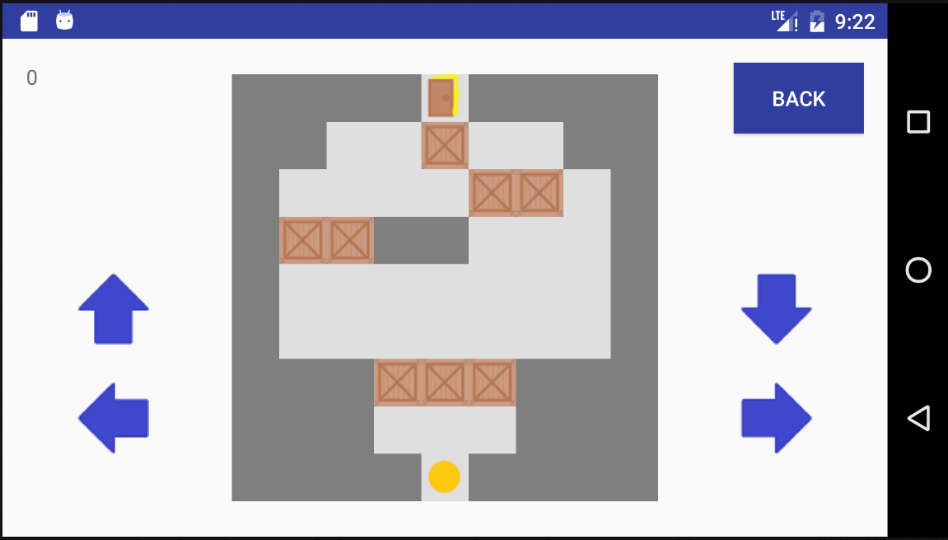
## 3.7 Nutzen der Datenbank für Login

Gleich zu Beginn des Spiels, also in der Main Activity, werden die Levels in die Datenbank geladen. Wie genau dies geschieht, wird im Punkt 3.8 genauer beschrieben. Außerdem wird das Überprüfen der Benutzerdaten so umgeschrieben, dass diese mit den Daten aus der Datenbank verglichen werden. Der Registrierungs-Button bekommt nun die Funktion, die Daten in den Eingabefeldern in die Datenbank zu schreiben. Damit die anderen Activities auch wissen, um welchen Spieler es sich handelt, wird die Spieler ID mithilfe eines Intents an die Menu Activity weitergegeben.

## 3.8 Implementieren der Level-Auswahl

Im ersten Schritt werden die Levels auf einem Blatt skizziert. Wichtig hierbei ist, dass darauf geachtet wird, ein Level so zu gestalten, dass es am Ende nicht unmöglich zu lösen ist. Im besten Fall steigert sich der Schwierigkeitsgrad allmählich.

Die fertigen Levels werden in die Main-Klasse implementiert nach dem Muster, das im Punkt 3.1 genauer erläutert ist. In folgenden Abbildungen sieht man beispielsweise Level 5 aus dem Programmcode und daneben die dazugehörige Spieldarstellung.

1 1 1 1 4 1 1 1 1   
1 1 0 0 2 0 0 1 1  
1 1 0 0 0 2 2 0 1  
1 2 2 1 1 0 0 0 1  
1 0 0 0 0 0 0 0 1  
1 0 0 0 0 0 0 0 1  
1 1 1 2 2 2 1 1 1  
1 1 1 0 0 0 1 1 1  
1 1 1 1 3 1 1 1 1  
 *Abb. 6*  *Abb. 7*

Für das Auswählen der Levels wird die Selection Activity erstellt, welche nun anstelle der Game Activity im Menu gestartet wird. Die Activity selbst bekommt nur elf Image-Buttons, einen um zurück zu kommen und die restlichen zehn für die Levels. Die Selection Activity bekommt von der Menu Activity die Spieler ID mithilfe eines Intents und gibt diese zusammen mit der ID des Levels an die Game Activity weiter. Dazu wird überprüft, welcher der zehn Buttons gedrückt wird. Dieser gibt dann die entsprechende ID weiter. Da aber nicht jedes Level von Anfang an spielbar sein soll, wird aus der Datenbank die ID des momentanen Levels für den Spieler gelesen und die nachfolgenden entsprechenden Buttons deaktiviert und grau eingefärbt. Um weitere Levels freizuschalten, wird die Methode onActivityResult() benutzt, welche aufgerufen wird sobald die Game Activity beendet wird. In dieser wird überprüft ob das gerade gespielte Level das letzte freigeschaltete war und anschließend wird das nächste freigeschaltet.

## 3.9 Hinzufügen der Zeitmessung

Das Spiel soll die Zeit anzeigen, die man für ein Level braucht und diese als Highscore nehmen. Dazu wird in der Game Activity ein TimerTask erstellt, welcher jede Sekunde von einem Timer aufgerufen wird, nachdem ein Button zum ersten Mal betätigt wird. Damit man die Zeit auch im Spiel sieht, wird oben links ein Text View erstellt, in dem die Zeit angezeigt wird. Der TimerTask zählt einen Integer-Wert von null aufwärts und setzt den Text in der Text View auf diesen Wert. Sobald das Level beendet wird, wird die benötigte Zeit mit dem aktuellen Highscore aus der Datenbank verglichen, falls es einen gibt. Falls es noch keinen gibt oder die neue Zeit besser ist, wird diese in die Datenbank zusammen mit der Spieler-ID eingetragen.

## 3.10 Anpassen des Layouts

Sowohl für den Login-Bildschirm als auch für die Spielmodus-Auswahl werden zuerst auf einem Blatt Skizzen gefertigt, um besser planen zu können, wie diese Menüs gestaltet werden sollen (siehe Abb. 11, 12). Die Darstellung wird dann in Android Studio so implementiert. Beim Layout wurde auf die Farben orange und blau wert gelegt, um ein besonderes Erscheinungsbild zu haben und an das Logo zu knüpfen, das ebenfalls aus den beiden Farben besteht. Beispielsweise sind alle Zurück- oder Ende-Buttons in blau gestaltet und die Buttons für die Levels orange, wenn sie frei zum Spielen sind. Die noch gesperrten Levels sind grau hinterlegt.

## 3.11 Erweiterung der Spielmodus-Auswahl auf 2 Spieler

Für das Projekt muss der Spielmodus auch auf zwei Spieler erweitert werden. Dafür wird in der Menu Activity zusätzlich zu der Spieler-ID eine weitere Variable namens Modus weitergegeben. Diese Variable ist abhängig von dem Spielmodus, der ausgewählt wird und kann die Werte 0 für Einzelspieler, 1 für zwei Spieler und 2 für Wifi Modus haben. Der Button für den Zweispieler-Modus startet ebenfalls die Selection Activity, aber der für den Wifi Modus bringt eine Meldung, dass dieser Modus noch nicht implementiert ist. Die Selection Activity macht mit der Modus-Variable nichts anderes, als sie weiter an die Game Activity zu geben, wo sie erst nach dem Beenden des Levels genutzt wird. Hier wird nun die neue Finish() -Methode aufgerufen, in der als Erstes der Modus abgefragt wird. Im Falle des Einspieler-Modus bleibt alles wie vorher ohne Änderung. Falls es sich um den Zweispieler-Modus handelt, wird die Zeit des ersten Spielers gespeichert und das Level neu geladen. Nachdem der zweite Spieler das Spiel beendet, wird die Zeit der Beiden verglichen und eine Gewinn-Nachricht für den entsprechenden Spieler ausgegeben. Der Highscore wird nicht in der Datenbank aktualisiert, da nicht bekannt ist wer der zweite Spieler ist.

## 3.12 Implementieren der Highscore-Activity

Damit man sehen kann, welcher Spieler die beste Zeit in den Levels hat, wird die Highscore Aktivity erstellt. In dieser wird nur ein Button unten in die Mitte gesetzt um zum eigentlichen Menü zurückzukehren. Ansonsten werden 30 Text Views gebraucht, wovon zehn die Levels 1-10 repräsentieren, zehn weitere die Namen der Spieler anzeigen und die letzten zehn die benötigte Zeit darstellen. Nachdem alle Elemente ihre IDs erhalten und sie initialisiert sind, wird aus der Datenbank der Name des Spielers mithilfe dessen Id, sowie des Highscores für jedes Level geladen und in die entsprechenden Text Views eingetragen. Beim betätigen des Zurück Buttons wird die finish() Methode aufgerufen.

# 4. Projektergebnis

## 4.1 Fehlerbehebung

### 4.1.1 Fortschritt wird nicht gespeichert

Ein großes Problem hat das Speichern des Fortschritts in die Datenbank gemacht. Da beim Lesen aus der Datenbank die Datensätze bei 0 anfangen, aber beim Lesen und Speichern bzw. Ändern bei 1 anfangen, hat dies zu viel Verwirrung geführt. Hinzu kommt, dass zu Testzwecken mehrere Spieler aus der Datenbank gelöscht wurden, aber der Auto-Increment in der Datenbank nicht zurückgesetzt wurde. Dies führte dazu, dass das Spiel den Highscore beispielsweise für den Spieler mit der ID 1 speichern mochte, aber dieser nicht existierte, da die IDs erst bei 4 anfingen. Dieses Problem wird gelöst, indem die Datenbank manuell zurückgesetzt wird.

### 4.1.2 Spiel stürzt nach Level 10 ab und verliert Fortschritt

Wie auch schon beim ersten Problem tritt hier ein ähnlicher Fehler auf. Aus der Datenbank wird der Highscore des nächsten Levels abgefragt, da davon ausgegangen wird, dass die Datenbank beim Auslesen bei 1 anfängt. Diese Annahme führt dazu, dass es nach dem letzten Level zu einem "Array: out of Bounds" -Fehler kommt und das Spiel abstürzt. Der Fehler wird dadurch gelöst, dass man beim Auslesen aus der Datenbank bei der Level ID eins abzieht.

### 4.1.3 Sonstige kleine Fehler

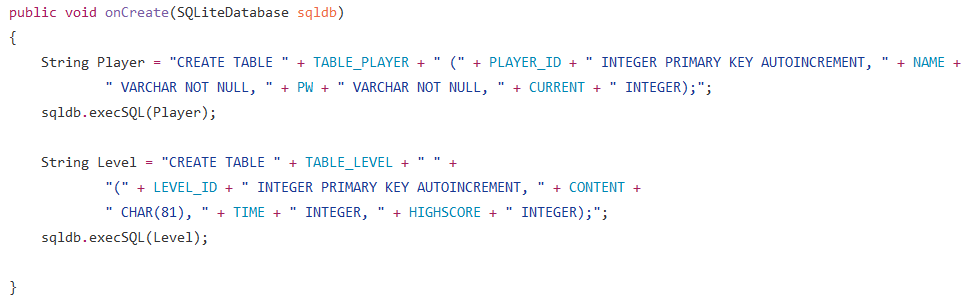
Wenn man einen Fehler in einem Level gemacht hat, kann man nicht neu anfangen, da kein „Zurück“-Button vorhanden ist. Dieser war bei der vorherigen Implementierung nicht vorgesehen. Der Button kann in dem Level-Layout hinzugefügt werden und muss anschließen in der Level Activity initialisiert werden und eine Funktion erhalten. Beim Drücken des Buttons kommt man nun zurück zur Level-Auswahl.

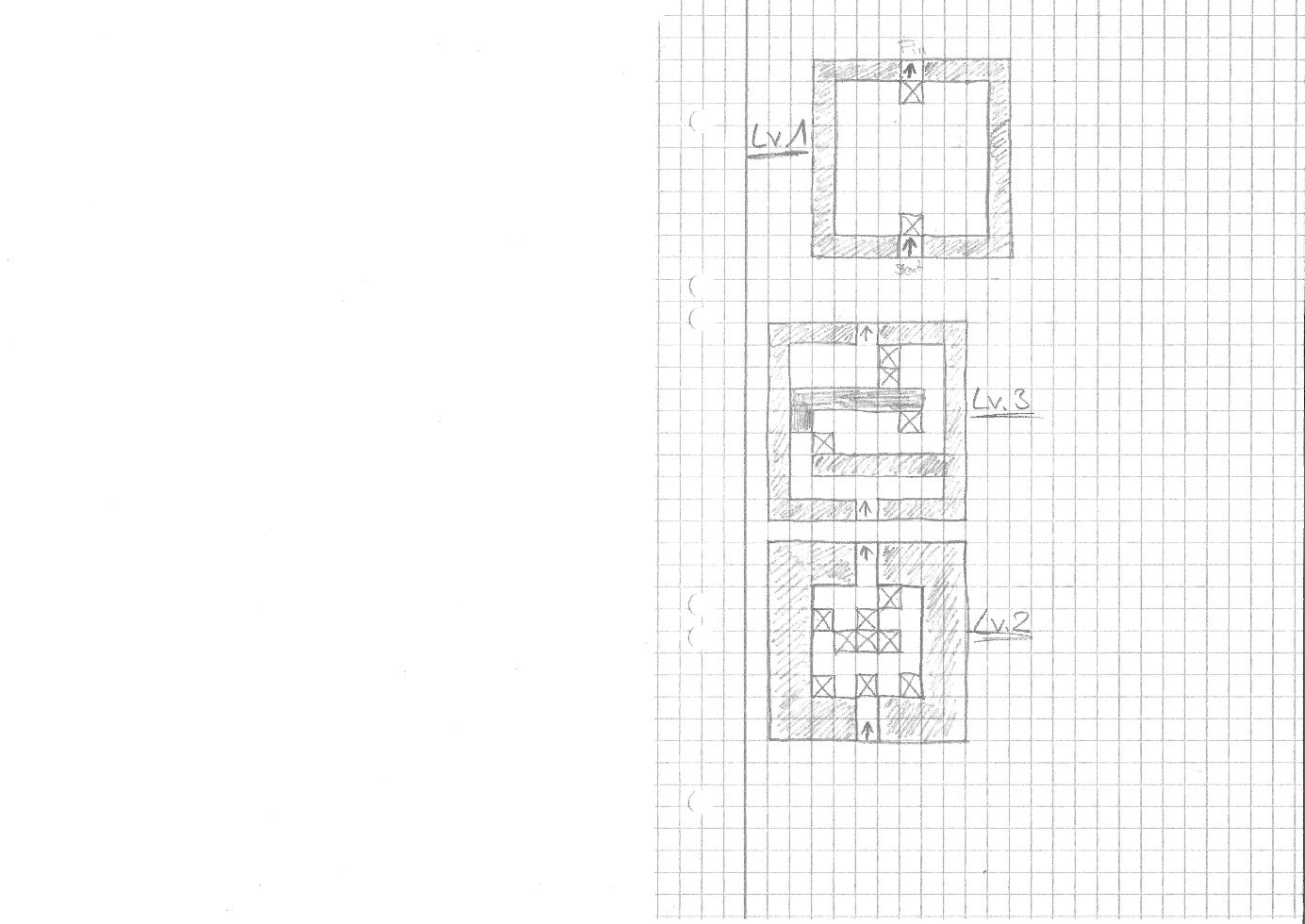
## 4.2 Abweichungen

Der Zweispieler-Modus über zwei unterschiedliche Smartphones konnte in der vorgegebenen Zeit nicht mehr verwirklicht werden. Dafür wurde ein Highscore implementiert, der ursprünglich nicht in der Planung vorgekommen ist. Engpässe traten vor allem in der Kalenderwoche 51 auf, da ein Krankheitsfall eingetreten ist. Im Vergleich der IST- und SOLL-Zeit wird ersichtlich, dass die Planung ansonsten im Großen und Ganzen korrekt durchgeführt wurde. Abweichungen sind vor allem zu sehen, weil mehrere Aufgaben lediglich über einen ganzen Berufsschulblock geplant wurden und dadurch eine Woche früher beendet bzw. eine Woche später angefangen wurden. Das Laden der Levels aus der Datenbank konnte erst etwas später umgesetzt werden, da die Datenbank-Klasse unerwartete Fehler aufwies.

# 5. Anhang

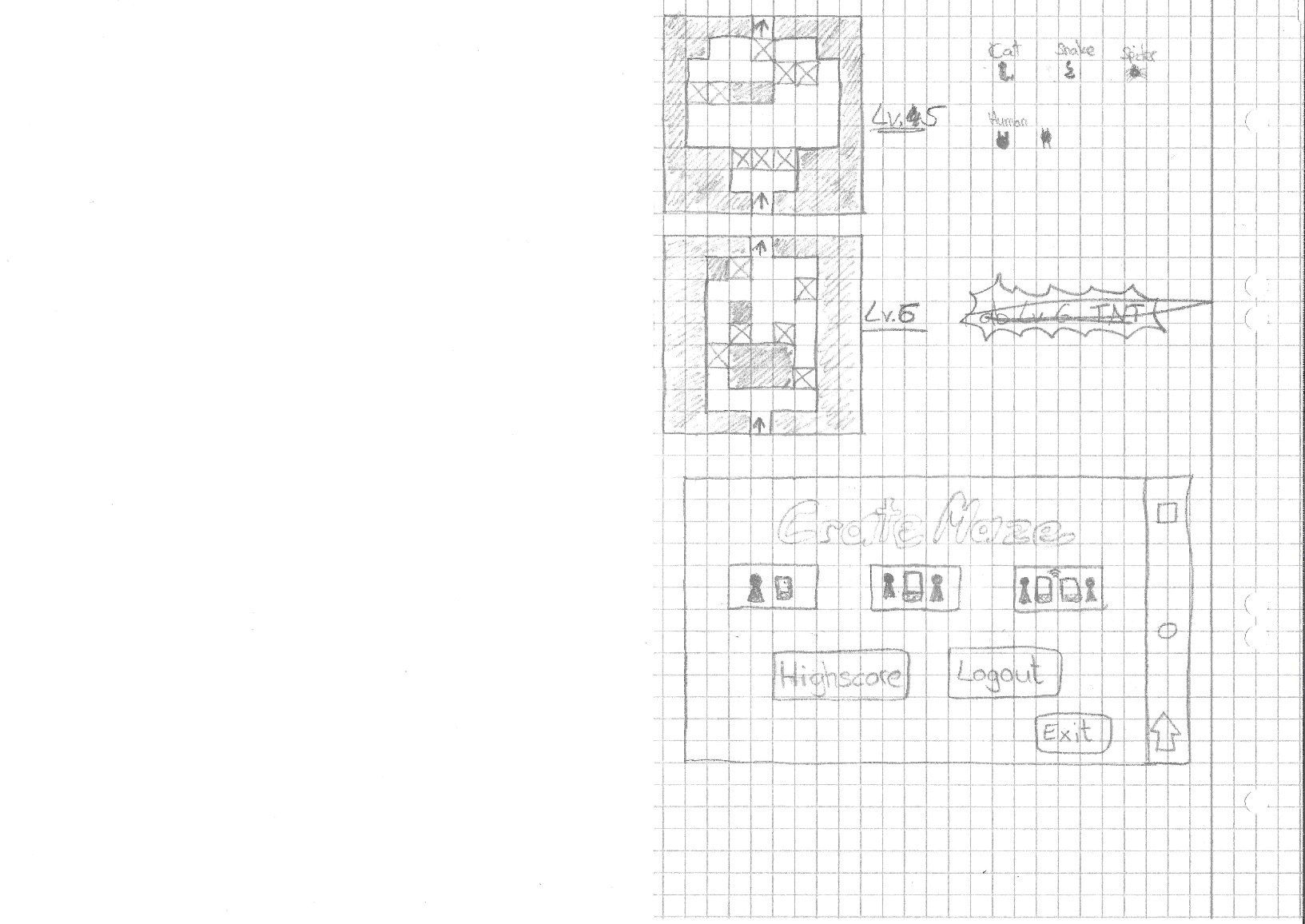
## 5.1 Screenshots

  
*Abb. 8*

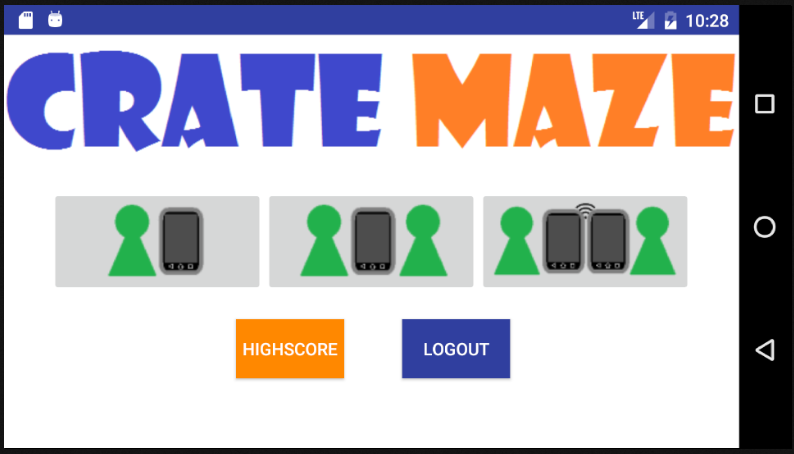


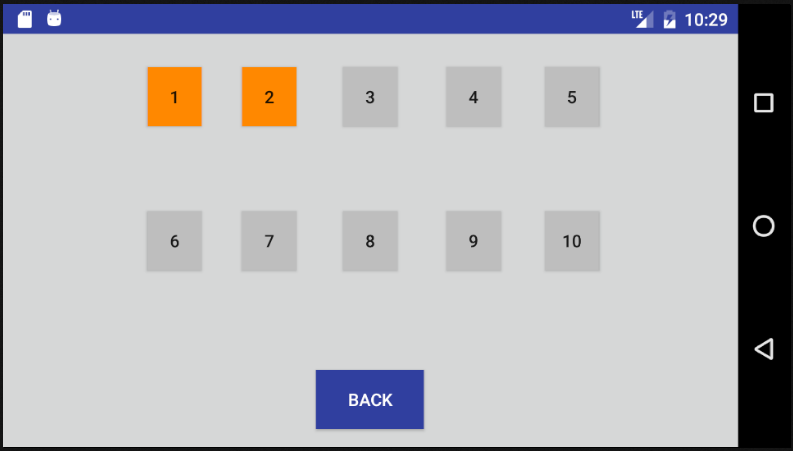
*Abb. 9*

  
*Abb. 10*



*Abb. 11*

  
*Abb. 12*

  
*Abb. 13*

## 5.2 Github-Accounts

Katharina Ambrosch: Unbreakable2112

Robert Kais: luxamy