Project B



Julian Janda, Markus Pulverer, Mergime Neziri, Bernhard Rohr, Anita Topalovic

# Teammitglieder

**Julian Janda**

Matrikelnummer: 1160028

**Projektmanagement**

**Markus Pulverer**

Matrikelnummer: 1261544

**Programmierung**

**Mergime Neziri**

Matrikelnummer: 1360032

**Testen**

**Bernhard Rohr**

Matrikelnummer: 1460215

**Programmierung**

**Anita Topalovic**

Matrikelnummer: 1460217

**Projektmappe**

# Kriterien, Metriken und Methoden

## Kriterien:

Leserliche, Klare Dokumentation.

Code muss automatisch/manuell getestet werden.

Code muss einen Mindestwert an Qualität erfüllen (Code Metriken).

## Metriken:

"MetricsReloaded" (Android Studio Plugin)

## Methoden:

JUnit tests

## Testplan:

1) Entwickler erstellen vor Beginn einer user story einen neuen Branch.

2) Entwickler schreiben Code und zugehörige Tests.

3) Entwickler generieren für fertig getesteten Code ein "pull request".

4) Kollegen prüfen besagten Code -> falls Qualitätskriterien erfüllt, wird dem request stattgegeben.

# Product Backlog

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Beschreibung** | **Story Points** | **Priorität** | **Notizen** | **Verantwortlichkeit** |
| 1 | Als Spieler möchte ich Partien hosten können | 20 | 60 | Eine Partie beginnen und andere kommen hinzu | Bernhard Rohr, Markus Pulverer |
| 2 | Als Spieler möchte ich einer Partie beitreten können | 13 | 60 |  | Bernhard Rohr |
| 3 | Als Spieler möchte ich eine Möglichkeit haben zu Cheaten | 8 | 60 |  | Mergime Neziri, Anita Topalovic |
| 4 | Als Spieler möchte ich das Spielfeld sehen | 5 | 100 |  | Julian Janda |
| 5 | Als Spieler möchte ich über Mauern springen | 5 | 80 |  | Anita Topalovic |
| 6 | Als Spieler möchte ich meine Gegner umwerfen | 2 | 70 | Kegel umwerfen | Anita Topalovic |
| 7 | Als Spieler möchte ich würfeln | 5 | 80 |  | Markus Pulverer |
| 8 | Als Spieler möchte ich mein Bild als Spielstein verwenden | 1 | 10 |  | Mergime Neziri |
| 9 | Als Spieler möchte ich konfigurieren | 3 | 30 | Nutzername; Spiel gespeichert | Julian Janda |
| 10 | Als Spieler möchte ich die anderen Mitspieler sehen | 2 | 40 | Nutzername der Mitspieler | Mergime Neziri |
| 11 | Als Spieler möchte ich eine einfache Bedienung haben | 2 | 50 | z.B. bei den Einstellungen nicht zu kompliziert | Alle |

# Burn Down Charts







# Fragenkatalog

* Mit welcher Android-Version soll das Spiel kompatibel sein?
* Soll man das Spiel auf einem Gerät spielen oder auf mehreren?
* Soll es für Handys oder Tablets kompatibel sein?
* Wie lange soll es die Leute beschäftigen?
* Muss man ein Konto erstellen, um es zu spielen?
* Braucht man eine Internetverbindung um zu spielen? Oder geht es auch offline?
* Soll es einen Highscore geben?
* Für wie viele Spieler soll es sein?
* Welche Sprache?
* Welche Zielgruppe?
* Welche Daten sollen gespeichert werden?

Project B

Universitätsstraße 65-67

9020 Klagenfurt

Herr Dr. Martin Pinzger

Universitätsstraße 65-67

9020 Klagenfurt

Datum: 19.04.2016

Angebotsnr: A1111

**Angebot**

Sehr geehrter Herr Dr. Pinzger!

Vielen Dank für das Interesse an unserem Produkt! Das Brettspiel „**Mauerhüpfer**“ ist eine neue Version des beliebten Klassikers „Mensch Ärgere dich nicht“. Es können sich bis zu vier Geräte (**Android-Version 4.4 oder höher**) über WLAN miteinander verbinden und die App spielen. Neben den inherenten Features wie das Hüpfen über Mauern oder dem Spielen mit zwei Würfeln, bietet unsere Ausführung noch ein paar weitere **Besonderheiten** an:

* Kreative Möglichkeiten zum Schummeln
* Intuitives Würfeln durch Bewegungssteuerung
* Personalisierbare Spielfiguren

Der Preis für die App setzt sich zusammen aus:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Bezeichnung** | **USt** | **Menge** | **Preis** | **Gesamt** |
| Arbeit | 20% | 165 Stunden | 40,00 € | 6.600,00 € |
| Zusätzliche Kosten | 20% |  |  | 1.000,00 € |
|  |  |  | **Netto:** | 7.600,00 € |
|  |  |  | **USt:** | 1.520,00 € |
|  |  |  | **Brutto:** | **9.120,00 €** |

Der Lieferumfang umfasst:

* Qualitätsgeprüfte App
* Installationsanleitung
* Spielanleitung

Wir haben für die Fertigstellung Ihrer App folgende Meilensteine festgelegt:

* **05.05.2016** Singleplayer Funktionalitäten
* **25.05.2016** Multiplayer Funktionalitäten
* **Ende Juni** Auslieferung der fertigen Software

Laut unseres Zeitplanes wäre die vorgeschlagene Software Ende Juni 2016 verfügbar. Der genaue Liefertermin würde dann noch rechtzeitig bekanntgegeben. Sollten Sie noch Fragen oder weitere Wünsche haben, können Sie uns gerne jederzeit kontaktieren.

Mit freundlichen Grüßen

Project B

# Stundenlisten

## Julian Janda

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Beschreibung | Datum | Stunden |
| 1 | Teambesprechung | 07.03.2016 | 1 |
| 2 | Installation von Android Studio | 08.03.2016 | 0,5 |
| 3 | Einrichten von Repository | 03.04.2016 | 0,5 |
| 4 | Kundengespräch | 04.04.2016 | 0,5 |
| 5 | Product Backlog | 11.04.2016 | 2 |
| 6 | Festlegen der Metriken/Methoden/Kriterien | 12.04.2016 | 2 |
| 7 | Android Studio Projektinitialisierung | 16.04.2016 | 2 |
| 8 | Ideensammlung für Angebot | 18.04.2016 | 1 |
| 9 | Erstellen von Assets | 26.04.2016 | 4,5 |
| 10 | Weitere Assets und erste Implementierung | 02.05.2016 | 2 |
| 11 | Spielplan V1.0 + Code Aufräumen | 03.05.2016 | 3 |
| 12 | Programmstruktur - Planung | 04.05.2016 | 1 |
| 13 | Block Asset Generator | 10.05.2016 | 3 |
| 14 | Refactoring, Recovery & HD Block Assets | 24.05.2016 | 5 |
| 15 | Methods for handling blocks | 25.05.2016 | 2,5 |
| 16 | Block Display Prototype | 27.05.2016 | 4 |
| 17 | Blocktypen | 29.05.2016 | 2 |
| 18 | Spieler | 30.05.2016 | 1 |
| 19 | Views Dynamic 🡪 XML | 01.06.2016 | 1,5 |
| 20 | Moving Figures 1 | 02.06.2016 | 3 |
| 21 | Moving Figures 1.5 | 05.06.2016 | 3 |
| 22 | Moving Figures 2 | 07.06.2016 | 4 |
| 23 | Moving Figures 3 & 4 | 09.06.2016 | 4 |
| 24 | Moving Figures 5 | 10.06.2016 | 4 |
| 25 | Moving Figures 6 & Refactoring + Tidying up | 12.06.2016 | 3 |
| 26 | Merging Branches | 15.06.2016 | 4 |
| 27 | Multiplayer 1 & 2 | 16.06.2016 | 4 |
| 28 | Multiplayer, Game Mechanics, Info alerts, Layout | 17.06.2016 | 6 |
| 29 | Refactoring, Bug Fixes, Clean-Up | 18.06.2016 | 6,5 |
|  |  |  |  |
|  |  |  |  |

## Markus Pulverer

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Beschreibung | Datum | Stunden |
| 1 | Teambesprechung | 07.03.2016 | 1 |
| 2 | Installation von Android Studio | 14.03.2016 | 2 |
| 3 | Internet Recherche: Android Programmierung | 18.03.2016 | 2 |
| 4 | Kundengespräch | 04.04.2016 | 0,5 |
| 5 | Product Backlog | 11.04.2016 | 2 |
| 6 | Würfelfunktion | 16.04.2016 | 3 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Bernhard Rohr

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Beschreibung | Datum | Stunden |
| 0 | Diverse Meetings und Präsentationen | --- | 5 |
| 1 | Teambesprechung | 07.03.2016 | 1 |
| 2 | Installation von Android Studio | 12.03.2016 | 1 |
| 3 | Kundengespräch | 04.04.2016 | 0,5 |
| 4 | Product Backlog | 11.04.2016 | 2 |
| 5 | Netzwerk | 18.04.2016 | 2 |
| 6 | Angebot | 18.04.2016 | 1 |
| 7 | Jenkins | 18.04.2016 | 2 |
| 8 | Host/Join 1.0 | 23.04.2016 | 5 |
| 9 | Host/Join Update | 14.05.2016 | 5 |
| 10 | Netzwerk + Options | 27.05.2016 | 3 |
| 11 | Refactor, change and optimize Host/Join | 29.05.2016 | 8 |
| 12 | Sync | 30.05.2016 | 3 |
| 13 | Merging | 15.06.2016 | 4 |
| 14 | Jenkins Probleme | 15.06.2016 | 1,5 |
| 15 | Refactor, fix Technical Debt, Jenkins | 16.06.2016 | 5 |
| 16 | Diverse CodeProbleme, refactor | 17.06.2016 | 6,5 |
| 17 | Jenkins... Build Probleme / Code Coverage | 17.06.2016 | 3,5 |
| 18 | Auf Cheaten Reagieren und Debt fix | 20.06.2016 | 5 |
|  |  |  |  |
|  |  |  |  |

## Mergime Neziri

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Beschreibung | Datum | Stunden |
| 1 | Teambesprechung | 07.03.2016 | 1 |
| 2 | Installation von Android Studio | 16.03.2016 | 0,5 |
| 3 | Kundengespräch | 04.04.2016 | 0,5 |
| 4 | Ausarbeitung von Aufgabenblatt 2, Aufgabe 2 | 05.04.2016 | 2,5 |
| 5 | Product Backlog | 11.04.2016 | 2 |
| 6 | Scrum-Meeting | 13.04.2016 | 0,15 |
| 7 | Angebot erstellen | 18.04.2016 | 2 |
| 8 | Angebot erstellen | 19.04.2016 | 1 |
| 9 | Scrum Meeting | 26.04.2016 | 0,15 |
| 10 | Spielfiguren | 02.05.2016 | 2 |
| 11 | Präsentation erstellen | 03.05.2016 | 1 |
| 12 | Userbild einfügen | 09.05.2016 | 2 |
| 13 | Userbild einfügen | 11.05.2016 | 2 |
| 14 | Userbild einfügen | 14.05.2016 | 1,5 |
| 15 | Userbild einfügen | 17.05.2016 | 2 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Anita Topalovic

|  |  |  |  |
| --- | --- | --- | --- |
| Nr. | Beschreibung | Datum | Stunden |
| 1 | Teambesprechung | 07.03.2016 | 1 |
| 2 | Installation von Android Studio | 17.03.2016 | 1 |
| 3 | Kundengespräch | 04.04.2016 | 0,5 |
| 4 | Ausarbeitung von Aufgabenblatt 2, Aufgabe 1 | 05.04.2016 | 3 |
| 5 | Product Backlog | 11.04.2016 | 2 |
| 6 | Burn Down Charts erstellen | 12.04.2016 | 0,15 |
| 7 | Scrum Meeting | 13.04.2016 | 0,15 |
| 8 | Kundenangebot erstellen | 18.04.2016 | 2 |
| 9 | Kundenangebot fertigstellen | 19.04.2016 | 1,5 |
| 10 | Funktion über die Mauer zu springen | 25.04.2016 | 1 |
| 11 | Scrum Meeting | 26.04.2016 | 0,15 |
| 12 | Spielerbewegung | 02.05.2016 | 2 |
| 13 | Spielerbewegung | 09.05.2016 | 1 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Glossar

## Scrum

### Was ist Scrum?

Scrum ist ein agiler Prozess, mit dem man es schafft, in einem kurzen Zeitraum eine lauffähige Software zu entwickeln. Es ist ein einfacher Prozess, welcher sich aus Meetings, Artefakten und bestimmten Rollen zusammensetzt.

### Rollen

**Product Owner**

Der Product Owner repräsentiert den Kunden und achtet darauf, dass die Anforderungen im Entwicklungsprozess mit einfließen. Er entscheidet welche Funktionalitäten entwickelt werden und wann das Projekt beendet sein soll. Der Product Owner entscheidet am Ende der Arbeit, ob das Produkt Marktfähig ist oder ob es noch Verbessert gehört.

**Scrum-Team**

Das Team besteht meist aus fünf bis zehn Leuten. Es organisiert sich selbst und jeder macht alles, d.h. es sind alle beteiligt bei der Programmierung, der Projektmappe etc. Die Software wird eigenverantwortlich entwickelt, daher sollten die Mitglieder während des ganzen Projekts in der Gruppe bleiben. Nach jedem Sprint muss das Team ein lauffähiges System präsentieren, welches dem Endprodukt immer näher kommt.

**Scrum-Master**

Der Scrum-Master hat die Aufgabe, dass die Scrum-Werte eingehalten werden. Entstehen Hindernisse, die das Team aufhält, werden diese von ihm/ihr beseitigt. Er/Sie ist dafür verantwortlich, dass das Team nicht von außen gehemmt wird und gewährleistet, dass die Gruppe funktioniert und effektiv ist.

### Sprint

Der Sprint liegt im Mittelpunkt von Scrum. Die Dauer zwischen den „Sprints“ sollte unverändert und regelmäßig sein. Während der Sprints wird nichts verändert, welches das Ziel gefährdet.

Ein Sprint besteht aus den Meetings,

* Sprint-Planung,
* Daily Scrum Meetings,
* Sprint Review und
* Sprint Retrospektive.

Nach jedem Durchlauf wird vom Team ein lauffähiges System präsentiert, welches dem Endprodukt immer näher kommt.

### Meetings

**Sprint-Planung**

Zu Beginn jedes Sprints findet ein Sprint Planning statt, bei dem alle Projektrollen teilnehmen. Dabei entscheiden das Team und der Product Owner die Inhalte für den nächsten Sprint.

**Daily Scrum Meetings**

Die täglich statt findenden Treffen des Teams und dem Scrum Master nennt man Daily Scrum Meeting. Es können auch noch andere Personen daran teilnehmen, doch nur diese beiden Rollen haben das Recht was zu sagen. Jedes Meeting dauert maximal 15 Minuten und da es eher informell ist, wird es oft im Stehen abgehalten. Das Stehen hilft den Teilnehmern, das Treffen kurz zu halten und nur die wesentlichsten Punkte zu besprechen.

Jedes Teammitglied hat die Aufgabe, drei Fragen zu beantworten:

* Was habe ich getan, um meinen Teammitgliedern zu helfen?
* Was werde ich tun, um meinen Teammitgliedern zu helfen?
* Ist etwas dazwischengekommen, was mich und mein Team hemmt?

Die Aufgabe des Scrum Masters ist es, sicherzustellen, dass das Team während des Meetings entscheidet, was an diesem Tag noch zu erledigen ist, um das Ziel des Sprints zu erreichen. Außerdem ist er dafür verantwortlich, Hindernisse zu beseitigen. Doch er ist nicht zuständig, dass Team zu steuern. Sie sind selbst dafür verantwortlich.

**Sprint Review**

Ein Sprint Review erfolgt am Ende eine Sprints. Das Team präsentiert dem Product Owner, dem Management und den anderen Teilnehmern, was es während des ganzen Sprints geschafft hat.

**Sprint Retrospektive**

Ebenfalls wie der Sprint Review, findet auch die Sprint Retrospektive, am Ende eines Sprints statt. Doch bei diesem Meeting darf nur das Scrum Team teilnehmen und es wird besprochen, was beim nächsten Sprint besser laufen soll bzw. was beim letzten Sprint gut gelaufen ist und weiter fortgesetzt wird.

### Artefakte

**Product Backlog**

Das Product Backlog ist eine Liste mit Anforderungen/Projektarbeiten, die immer laufend aktualisiert werden. Je nach Reihenfolge der Einträge, liegt deren Priorität fest. Gibt es während des Prozesses Änderungen oder neue Anforderungen, werden diese in Backlog eingetragen. Da das Product Backlog so flexibel ist, kann es schnell auf Veränderungen reagieren, ohne dass das Projekt gefährdet wird.

**Sprint Backlog**

Die Einträge, die im Product Backlog als höchst priorisierend eingestuft worden sind, werden auf deren Aufwand geschätzt und diese ins Sprint Backlog überführt. Sind das Team und der Product Owner einverstanden mit der Realisierung des Sprint Backlogs, dann wird das Ziel des Sprints festgelegt. Das Team fügt, wenn erforderlich, weitere Arbeiten hinzu bzw. unnütze Komponenten werden entfernt. Die ganze Arbeit wird mit dem Sprint Backlog sichtbar. Während des Sprints wird der Sprint Backlog weiter entwickelt.

**Impediment List**

Der Scrum Master hat die Aufgabe, „Blocker“(Hindernis) zu beseitigen. Ergibt sich so ein Blocker, wird dieser in der Impediment List festgehalten und beim nächsten Daily Scrum Meeting den anderen Teammitgliedern mitgeteilt.

**Burn Down Charts**

Der Burn Down Chart zeigt, wie viel Arbeit noch zu erledigen ist. Es wird täglich von jedem Teammitglied aktualisiert, dadurch wird die Grafik immer genauer. Von Tag zu Tag sehen sie, welche Aufgaben noch zu erledigen sind und erkennen Risiken, wenn welche entstehen. Durch diese Selbstverantwortung und Eigenorganisation steigt die Motivation des Teams.

### Definition of Done

Mit der Definition of Done stellt das Team fest, wann die Arbeit an den Product Backlog-Einträgen eines Sprints abgeschlossen ist. Dies gilt auch für die Anzahl der Einträge. Die Definition of Done wird vom Team immer erweitert, um so strengere Kriterien für eine bessere Qualität zu sichern. Es sollte jedes Produkt eine Definition of Done haben.

vgl. <http://www.scrum-kompakt.de/einfuehrung-in-scrum/>

vgl. <http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-DE.pdf>

## Android App Programmierung

### Entwicklungsprogramme

Für die Entwicklung eines Android Apps, ist das Herunterladen von Android Studio und den SDK Tools sowie Plattformen, welche den SDK Manager verwenden, notwendig. Kenntnis der Programmiersprache Java ist eine wichtige Voraussetzung.

### Projekt erzeugen

1. Durch das Anklicken von New Project kann ein Projekt gestartet werden. Es können mehrere Projekte erzeugt werden.
2. Es erscheint nach dem Erzeugen des Projektes ein Fenster. Die Felder sollen ausgefüllt und Next angeklickt werden.

**Applikation Name:** das ist der App Name, welcher den Usern erscheint

z.B.: „My First App“

**Company Domain:** Domain bietet eine Kennzeichnung welche dem Paketnamen angehängt wird.

**Package name:** ist der vollständige Name für das Projekt. Der Paketname muss einzigartig sein. Dieser kann abgeändert werden.

**Project location:** Verzeichnis auf dem System, welches die Projektdateien enthält.

1. Unter **Select the form factors your app will run on**, kann die Box für das Smartphone oder Tablet gecheckt werden.
2. Für Minimum SDK, sollte der API 8: Android 2.2 ausgewählt werden.

Minimum SDK ist die letzte Version von Android welche die App unterstützt, angegeben in der API level. Um die Unterstützung von mehreren Geräten zu erhalten, sollte die niedrigste verfügbare Version heruntergeladen werden, welche der App erlaubt sein feature set zur Verfügung zu stellen.

1. Alle anderen Optionen wie TV usw. sollten nicht ausgewählt werden und auf Next klicken.
2. Danach unter Add an activity to <template>, sollte Blank Activity ausgewählt werden und Next anklicken.

**Blank Activity**

Erstellt ein leeres Activity mit einer Aktionsleiste (action bar).

**Blank Activity with Fragment**

Leeres Activity mit einer Aktionsleiste und einem Fragment im Layout.

**Empty Activity**

Ein Leeres Activity. Hierbei handelt es sich um eine Minimalversion einer App ohne action bar.

**Fullscreen Activity**

Zeigt den Wechsel in und aus dem Vollbild-Modus.

**Master/Detail Flow**

Erzeugt zwei Activities sowie zwei Fragmente und zeigt, wie auf Tablets und Handies unterschiedliche Benutzerschnittstellen umgesetzt werden können.

**Navigation Drawer Activity**

Ähnelt einem Tabbed Activity mit vertikal angeordneten Tabs.

**Tabbed Activity**

Tab basierendes Activity.

1. Unter Customize the Activity, kann der Activity Name geändert werden.
2. Finish anklicken um das Projekt zu erzeugen.

### Einrichten des Geräts

Das Gerät muss mittels USB Kabel an den PC oder Laptop angeschlossen werden. Bei Anwendung von Windows muss ein geeigneter USB Driver für das Gerät installiert werden. Das USB Debugging muss aktiviert werden.

Die App kann durch das Anklicken von Run ausgeführt werden. Im Fenster Choose Device, kann das Gerät auf dem die App angezeigt werden soll, ausgewählt werden. Danach installiert das Android Studio die App auf das Gerät und startet sie.

### Kernkomponenten einer Android App

Eine Android App besteht aus verschiedenen Komponenten wie Activities, Fragments, Services, Content Provider, Broadcast Receiver, der Manifest-Datei und Ressourcen.

1. **Activities:** stellt eine einzelne Bildschirmseite der App dar z.B.: Verwalten von Kontaktdaten.

Merkmale:

* in einer App sind mehrere Activities vorhanden,
* werden auf einem Stack verwaltet,
* wenn ein neues Activity gestartet wird, wird es oben auf den Stapel gelegt,
* bei drücken der Zurück-Taste wird das zuvor aktive Activity wieder aktiv,
* bei Ressourcenknappheit werden inaktive Activities von der Virtual Maschine beendet,
* Userinterface kann in XML oder im Java Programmcode erstellt werden,
* Activities verfügen über Methoden, die in den unterschiedlichen Phasen des Lebenszyklus aufgerufen werden.

1. **Fragmente:** wiederverwendbarer Teil des Userinterface. Activities verwenden Fragmente abhängig von der Bildschirmgröße um auf Tablets und Handys unterschiedliche Informationen anzuzeigen. Fragmente besitzen einen eigenen Lebenszyklus.
2. **Services (Dienste):** Werden in einem Android Projekt zum Ausführen von lange dauernde Operationen im Hintergrund verwendet. Enthalten keine Benutzerschnittstelle und laufen unabhängig von den Komponenten, die sie erzeugt haben. Services können von mehreren Apps verwendet werden.
3. **Content Provider:** Dient zum Speichern und Abrufen von Daten und stellen die Daten Apps zur Verfügung. Ermöglicht den Datenaustausch zwischen Apps durch das Bereitstellen einer URL zur Identifikation eines Datensatzes. Daten werden über eine Tabelle eines Datenbankmodells bereitgestellt.
4. **Broadcast Receivers:** Reagieren auf Systembenachrichtigungen. Die Apps können Broadcasts versenden, auf die andere Apps reagieren.

### Klasse Intent

Die Intents aktivieren Komponenten der Apps. Wir unterscheiden zwischen zwei Arten von Intents:

1. **Implizite Intents:** diese führen eine Aktion aus, ohne die dazu verwendete Komponente zu kennen. Implizite Intents haben kein Wissen darüber, welche Komponenten sich um die Ausführung der Aktion kümmern.
2. **Explizite Intents:** diese aktivieren eine bekannte Komponente.

### Res und Assets

1. **Res:** die abgelegten Ressourcen werden über eine ID angesprochen. Die Ordnerstruktur innerhalb des res Ordners ist von Android vorgegeben.
2. **Assets:** die Dateien im assets Ordner können beliebig abgelegt werden. Die Dateien werden nicht direkt über eine ID angesprochen.

### Android Berechtigungen und Debugging

Verwendung von Berechtigungen unter Android, das Fehlverhalten im Fall einer fehlenden Berechtigung und die Möglichkeiten zur Fehleranalyse nachdem eine App abgestürzt ist.

Berechtigungen erlauben einer App den Zugriff auf verschiedene Systemkomponenten z.B.: Internetzugriff, SMS. In der Datei AndroidManifest.xml werden die von der App benötigten Berechtigungen angegeben. Der Benutzer wird vor der Installation einer App über die von der App verlangten Berechtigungen gewarnt. Es sollen nur die Berechtigungen angegeben werden, die auch tatsächlich benötigt werden.

Wenn die App eine notwendige Berechtigung nicht gesetzt hat, führt das zu einer Fehlermeldung, häufig zu einer Exception, gefolgt von einem Absturz der App.

# Die Android Layout Klassen

Die Android Layout Klassen dienen zur Anordnung von Steuerelementen. Es können die verschiedenen Layouts beliebig geschachtelt werden. Beim Erzeugen der Benutzerschnittstelle wird zur Laufzeit aus der Layout XML Datei eine Java Klasse erstellt. Da dieser Vorgang sehr rechen- und speicherintensiv ist, sollte die Benutzerschnittstelle immer so flach wie möglich gehalten werden.

Wir unterscheiden zwischen LinearLayout, RelativeLayout, TableLayout, GridLayout und FrameLayout.

1. **LinearLayout** ordnet die eingebetteten Steuerelemente entweder nebeneinander oder untereinander an, Es kann über die Parameter android:layout\_width="0dp" und android:layout\_weight="1" die Breite festgelegt werden.
2. **RelativeLayout** positioniert die eingebetteten Steuerelemente relativ zueinander.
3. **TableLayout** gruppiert die eingebetteten Steuerelemente innerhalb der Elemente, auch hier kann über android:layout\_width und android:layout\_weight die Größe der Elemente in einer TableRow festgelegt werden.
4. **GridLayout** positioniert die Elemente auf einem Raster. Ein Steuerelement erstreckt sich über mehrere Zeilen und Spalten.
5. **FrameLayout** zeigt die eingebetteten Elemente übereinander an. Im Programmcode können dann verschiedene Steuerelemente abhängig von der Benutzereingabe angezeigt oder ausgeblendet werden.

# Layout-Parameter

Folgende Parameter können verwendet werden:

1. Größenangabe von Layouts und deren Steuerelementen:

**android:layout\_width und android:layout\_height**

1. XML Parameter konfiguriert die Ausrichtung der eingebetteten Steuerelemente:

**android:gravity="bottom|right"**

1. Ein Zeilenumbruch innerhalb eines Steuerelements wird verhindert:

**android:singleLine="true"**

# Größenangaben

Festlegen der Breite**: layout\_width**

Festlegen der Höhe: **layout\_height**

Höhe und Breite kann auch in Pixel angegeben werden.

Beispiel: android:layout\_width="128dp" (Density-independent Pixel = Geräte-unabhängige Pixel, das Layout erscheint auf Geräten mit unterschiedlichen Bildschirmauflösungen und Pixeldichten)

# Buttons

Verschiedene Arten von Schaltflächen stehen zur Verfügung

Arten:

1. **Button** gewöhnliche Schaltfläche mit Beschriftung.
2. **ImageButton** zeigt ein Bild anstelle der Beschriftung. .
3. **ToggleButton** ähnelt einer CheckBox, kann zwei Zustände annehmen.
4. **Switch** stellt einen Button dar, der zwei Zustände annehmen kann.
5. **RadioButton** wird zum Anzeigen von sich gegenseitig ausschließenden Optionen verwendet.
6. **CheckBox** bzw. Kontrollkästchen, das entweder angekreuzt oder leer sein kann.

# Die Klasse TextView

Diese Klasse wird zum Anzeigen von Text verwendet.