Schnaps2gether - Produktmappe

APPSOLUTE GAMES | EBERHARD, EDER, MAIER, PACHATZ, PLATTER

MEhrspieler schnapsen android app

eberhard, eder, MAIER, PACHATZ, PLATTER

2015

Inhalt

[1. Einleitung 3](#_Toc422852546)

[1.1. Zweck dieses Dokuments 3](#_Toc422852547)

[1.2. Projektrahmen 3](#_Toc422852548)

[1.3. Projektübersicht 3](#_Toc422852549)

[2. Allgemeine Produktbeschreibung 3](#_Toc422852550)

[2.1. Produktfunktionen 3](#_Toc422852551)

[2.1.2. Zusatzfeature 1 – Steuerung durch Bewegung 3](#_Toc422852552)

[2.1.3. Zusatzfeature 2 – Steuerung durch Sprache (ab Version 2.0) 4](#_Toc422852553)

[2.1.4. Zusatzfeature 3 – Statistik aller Spiele 4](#_Toc422852554)

[2.1.5. Information zu ähnlichen Systemen 4](#_Toc422852555)

[2.1.6. Zielgruppe 4](#_Toc422852556)

[2.1.7. Rahmenbedingungen 4](#_Toc422852557)

[3. Nicht funktionale Anforderungen 4](#_Toc422852558)

[3.1.1. Zuverlässigkeit 4](#_Toc422852559)

[3.1.2. Performanz 4](#_Toc422852560)

[4. Terminplan 5](#_Toc422852561)

[4.1.1. User Stories 5](#_Toc422852562)

[5. Vorläufiger Budgetplan 6](#_Toc422852564)

[6. Qualitätssicherung 6](#_Toc422852565)

[6.1.1. Verifikation der Anforderung 6](#_Toc422852566)

[6.1.2. Umfangsbestimmung von Quellcode 6](#_Toc422852567)

[6.1.3. Automatisierung 6](#_Toc422852568)

[6.1.4. Testfall 6](#_Toc422852569)

[6.1.5. Verwendete Testmethoden 6](#_Toc422852570)

[6.1.6. Metriken 7](#_Toc422852571)

[6.1.11. Weighted Methods per Class (Summe der Komplexität) 9](#_Toc422852572)

[6.1.12. Metriken zu Spiel3.cs 9](#_Toc422852574)

[6.1.13. McCabe Metrik zu Spiel3.cs 10](#_Toc422852575)

[6.1.14. Method Lines of Code Metrik zu Spiel3.cs 11](#_Toc422852576)

[6.1.15. Leistung CPU Metrik - Menu 11](#_Toc422852577)

[6.1.16. Leistung – Schnelles Spiel (1 Runde) 11](#_Toc422852578)

[6.1.17. Memory – Schnelles Spiel (1 Runde) 12](#_Toc422852579)

[6.1.18. Datenverbrauch 12](#_Toc422852580)

[7. Anhang 12](#_Toc422852581)

[7.1.1. Protokoll Kick-off Meeting 12](#_Toc422852582)

[7.1.2. Wichtigste Konzepte von Scrum 13](#_Toc422852583)

[7.1.3. Übersicht zur Android Programmierung 13](#_Toc422852584)

[7.1.4. Handhabung der Zeiterfassung 13](#_Toc422852585)

[7.1.5. Zeiterfassungseinträge von Appsolute Games 14](#_Toc422852586)

[7.1.6. Testberichte 14](#_Toc422852587)

[7.1.7. Burndowncharts 14](#_Toc422852588)

[7.1.8. Sprintbacklogs und Product-Backlog 14](#_Toc422852589)

# Einleitung

### Zweck dieses Dokuments

Dieses Dokument dient dazu die Anforderungen und Pläne für das Projekt Schnaps2gether zu sammeln. Des Weitern ist ein Kapitel über die Qualitätssicherung der App enthalten. Im Anhang finden Sie eine vollständige Stundenliste aller Team-Mitglieder.

### Projektrahmen

#### Projektteam

|  |  |  |
| --- | --- | --- |
| Name | Matrikelnummer | Aufgaben |
| Eberhard Alice | 1360554 | Design |
| Eder Norbert | 1360123 | Testen / Qualitätssicherung / Metriken |
| Maier Kerstin | 1360152 | Implementierung |
| Pachatz Veronika | 1360191 | Implementierung |
| Platter Frederik | 1360281 | Dokumentation |

### Projektübersicht

Das Ziel dieses Projekts ist die Erstellung und Auslieferung eines Softwareprodukts. Dieses Produkt setzt sich aus der Android-Applikation Schnaps2gether, der Produktmappe und der zugehörigen Dokumentation (Wartungshandbuch, Benutzerhandbuch) zusammen.

# Allgemeine Produktbeschreibung

### Produktfunktionen

#### Basisfeatures

* Schnaps2gether ermöglicht 2, 3 oder 4 Spielern gegeneinander an einem Ort Schnapsen zu spielen.
* Ein Spieler kann ein neues Spiel starten oder einem bereits erstellten Spiel beitreten.
* Der Spieler kann ein Spiel für 2, 3 oder 4 Spieler anlegen.
* Jeder Spieler kann sich einen Nicknamen geben.
* Jedem Spieler werden die aktuellen Punktelisten des Spiels angezeigt.
* Spieler können im 2er Spiel Schummeln und beim Schummeln ertappt werden (durch Betätigung von einer Schaltfläche).
* Möglichkeiten zum Schummeln sind Karten austauschen und in die Karten von Gegnern schauen.
* Wenn ein Spieler das Spiel verlässt wird die Spielrunde automatisch beendet.
* Die Spielregeln sind als Dokumentation in der App aufrufbar.

### Zusatzfeature 1 – Steuerung durch Bewegung

* Als Spieler kann ich durch eine Schüttelbewegung des Android-Geräts das Schummeln eines gegnerischen Spielers unterbinden.

### Zusatzfeature 2 – Steuerung durch Sprache (ab Version 2.0)

* Spieler können 20er und 40er ansagen, wenn Sie mit dem nächsten Spielzug an der Reihe sind.
* Ein Spieler kann beim 3er und 4er Schnapsen ansagen, welche Spielart (Land, Schnapser, etc.) er spielen möchte und das Spiel eines anderen „flecken“.

### Zusatzfeature 3 – Statistik aller Spiele

* Jeder Spieler kann sich seine persönliche Statistik über all seine bisherigen Spiele anzeigen lassen. Die Statistik enthält:
  + die Anzahl der gewonnen und verlorenen Spiele
  + die Gesamtanzahl der erreichten Bummerl
  + die höchste erreichte Punkteanzahl in einem Spiel

### Information zu ähnlichen Systemen

Kartenspielapplikationen sind keine Besonderheit, allerdings verwenden sie im Allgemeinen international bekannte Kartenspiele. Mehrspielerfähige Applikationen gibt es auch in großer Menge, jedoch verwenden diese fast immer eine Internetverbindung, sodass Spieler aus allen Winkeln der Welt miteinander spielen können. Mehrspieler-Kartenspielapplikationen sind meist simple Implementierungen ihrer realen Vorbilder und brillieren durch ein intuitives User Interface.

### Zielgruppe

Als Zielgruppe werden Jugendliche im Alter von 14 bis 25 Jahren betrachtet, die in Cafés und Gasthäuser gehen und sich durch die Applikation Schnaps2gether zusammensetzen können, um Karten zu spielen.

### Rahmenbedingungen

Die Obergrenze des Budgets des Kunden liegt bei 100.000 €. Acht Konkurrenten bewarben sich um die Ausschreibung und der Kunde legt bei der Wahl des Angebots vor allem Wert auf das Preis-Leistungs-Verhältnis. Das vollständige Produkt soll dem Kunden am 24.06.2015 präsentiert werden.

# Nicht funktionale Anforderungen

### Zuverlässigkeit

Die App soll zuverlässig genug sein um den Spielspaß nicht zu beeinträchtigen.

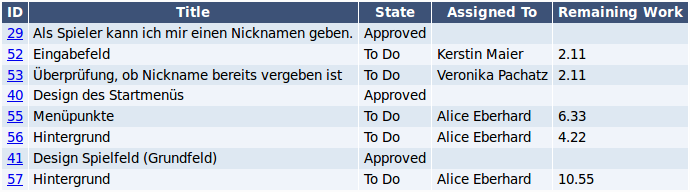
### Performanz

Die App soll flüssig genug laufen um den Spielspaß nicht zu beeinträchtigen.

# Terminplan

### User Stories

# 



# Vorläufiger Budgetplan

|  |  |  |
| --- | --- | --- |
| Pos. | Beschreibung | EUR |
| 1 | Basisfeatures | 72 345,00 € |
| 2 | Zusatzfeature 1 – Steuerung durch Bewegung | 2 205,00 € |
| 3 | Zusatzfeature 2 – Steuerung durch Sprache | 15 750,00 € |
| 4 | Zusatzfeature 3 – Statistik aller Spiele | 4 410,00 € |
|  | **SUMME** | **94 710,00 €** |

# Qualitätssicherung

Die Qualität des Produkts wurde durch eine genaue Anforderungsanalyse, Beschreibung einzelner Tasks, Durchführung von Reviews und Produkttests gewährleistet.

### Verifikation der Anforderung

Durch das Framework Scrum wurden die Anforderungen an die App in User-Storys zerlegt und mit den Teammitgliedern besprochen.

### Umfangsbestimmung von Quellcode

* + - Lines of Code
    - Halstead
    - McCabe

### Automatisierung

* + - Regressionstests
    - Für Continuous Integration wurde Jenkins verwendet. Damit konnten täglich Builds erstellt werden; zusätzlich wurden Junit-Tests ausgeführt.

### Testfall

Test-Spezifikationen dienten als Basis bei der Erstellung von Testfällen.

### Verwendete Testmethoden

* + - 1. Black-Box-Test

Das Produkt wurde durch spezifizierte Use-Cases getestet und der Input sowie der Output wurden genau dokumentiert. Fehler wurden behoben, bis das Ergebnis so ausfiel, wie erwartet.

* + - 1. White-Box-Test

Testfälle wurden direkt auf Basis des Codes erzeugt. Für diese Tests wurde Junit verwendet.

* + - 1. Reviews

Reviews wurden wöchentlich durchgeführt. Dadurch wurde die Qualität erhöht und Fehler frühzeitig erkannt.

### Metriken

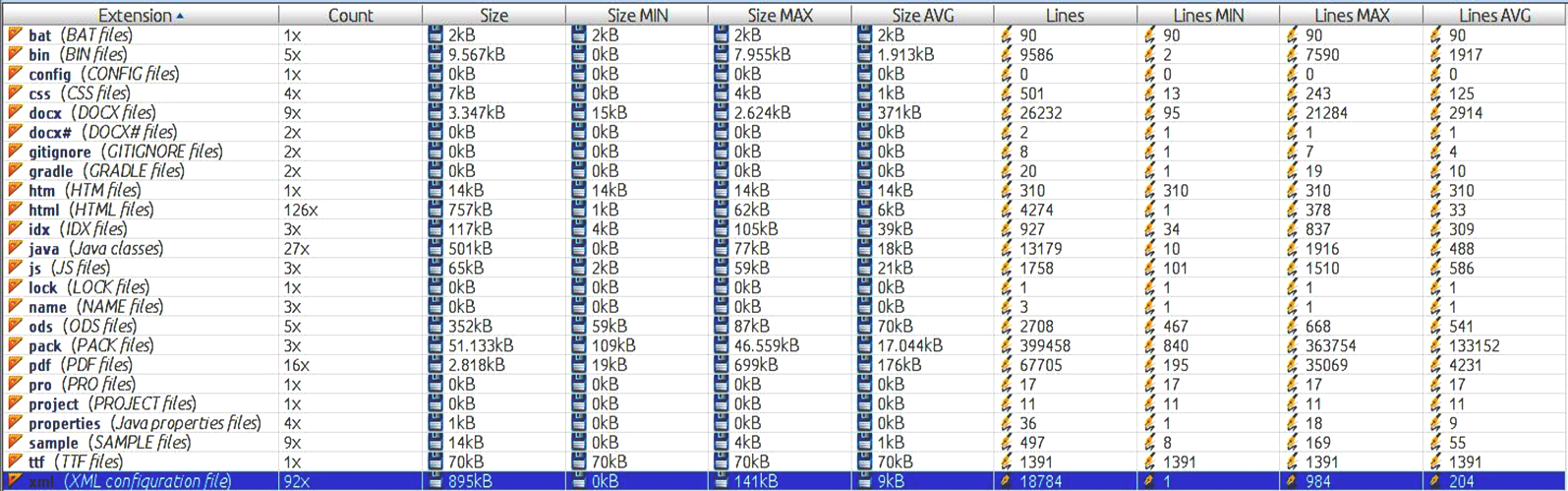
* + - 1. Lines of Code

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Source File** | **Total Lines** | **Source Code Lines** | **Source Code Lines**  **%** | **Comment Lines** | **Comment Lines**  **%** | **Blank Lines** | **Blank Lines %** |
| Application Test | 13 | 8 | 62% | 3 | 23% | 2 | 15% |
| Bummerl2 | 59 | 46 | 78% | 0 | 0% | 13 | 22% |
| Bummerl3 | 67 | 55 | 82% | 0 | 0% | 12 | 18% |
| Bummerl4 | 70 | 56 | 80% | 0 | 0% | 14 | 22% |
| ChangeNickname | 59 | 43 | 73% | 9 | 15% | 7 | 12% |
| GameEnd | 71 | 50 | 70% | 14 | 20% | 7 | 10% |
| Karte | 194 | 163 | 84% | 0 | 0% | 31 | 16% |
| KartenKomparator | 33 | 29 | 88% | 0 | 0% | 4 | 12% |
| Lobby | 503 | 358 | 71% | 63 | 13% | 82 | 16% |
| NeuesSpiel | 76 | 57 | 75% | 5 | 7% | 14 | 18% |
| Rufspiel | 43 | 35 | 81% | 3 | 7% | 5 | 12% |
| Spiel2 | 1083 | 793 | 73% | 90 | 8% | 200 | 18% |
| Spiel3 | 1041 | 858 | 82% | 32 | 3% | 151 | 15% |
| Spiel4 | 846 | 695 | 82% | 44 | 5% | 107 | 15% |
| Spieler | 83 | 64 | 77% | 0 | 0% | 19 | 23% |
| Spielfeld2 | 803 | 639 | 80% | 26 | 3% | 138 | 17% |
| Spielfeld2Client | 1094 | 895 | 82% | 26 | 2% | 173 | 16% |
| Spielfeld2Host | 1242 | 1000 | 81% | 39 | 3% | 203 | 16% |
| Spielfeld3Client | 1518 | 1267 | 83% | 28 | 2% | 223 | 15% |
| Spielfeld3Host | 1916 | 1575 | 82% | 44 | 2% | 297 | 16% |
| Spielfeld4Client | 793 | 660 | 83% | 40 | 5% | 93 | 12% |
| Spielfeld4Host | 913 | 768 | 84% | 54 | 6% | 91 | 10% |
| Spielfeld4Logik | 402 | 348 | 87% | 6 | 1% | 48 | 12% |
| Spielregeln | 54 | 38 | 70% | 5 | 9% | 11 | 20% |
| Startmenue | 118 | 91 | 77% | 5 | 4% | 22 | 19% |
| Statistik | 75 | 53 | 71% | 5 | 7% | 17 | 23% |
| WrongGameException | 10 | 6 | 60% | 3 | 30% | 1 | 10% |
| Total: | 13179 | 10650 | 81% | 544 | 4% | 1985 | 15% |

* + 1. Projekt – Lines of Code and Command lines
* Lines of Code: 13.179 Anzahl alle Programmzeilen (Projektumfang)
* Source Code Lines: 10.650 Algorithmen-Implementierungsgröße
* Comment Lines: 544 Kommentaranteil des Projektes
  + 1. Halstead-Metrik
* Klasse Spieler.java
  + Operatoren n1: 19 Operanden n2: 41
  + Operatoren N1: 200 Operanden N2: 296
  + Programmlänge: N1+N2 = 60
  + Vokabulargröße: n1+n2 = 496
  + Schwierigkeitsgrad (D): 13,85
  + Programmniveau (L): 4909,10
    1. Testabdeckung

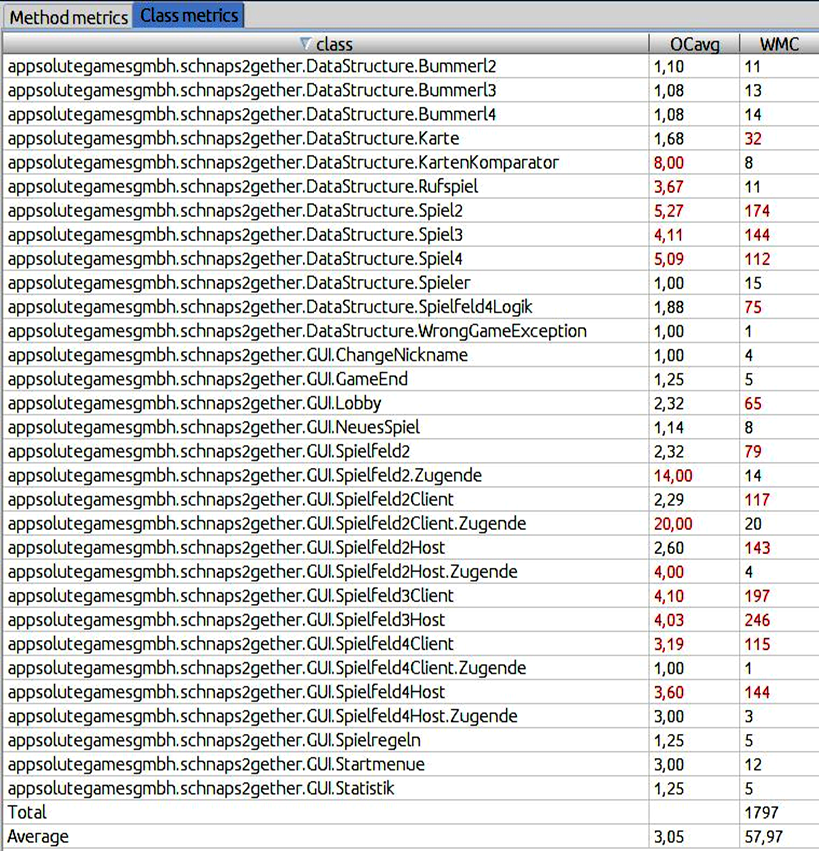


* + 1. Größe und Anzahl der Dateien



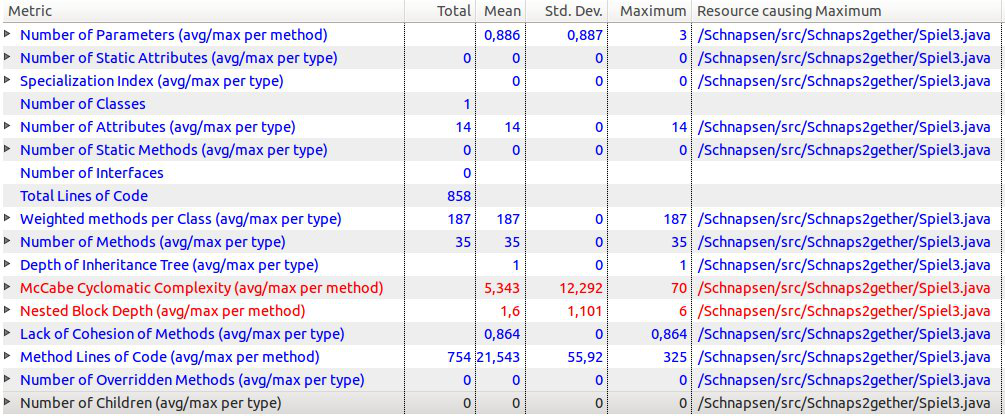
### Weighted Methods per Class (Summe der Komplexität)

### Rote/Hohe WMC Werte lassen auf einen höheren Wartungsaufwand schließen und die Klassen sind schwerer wiederverwendbar.



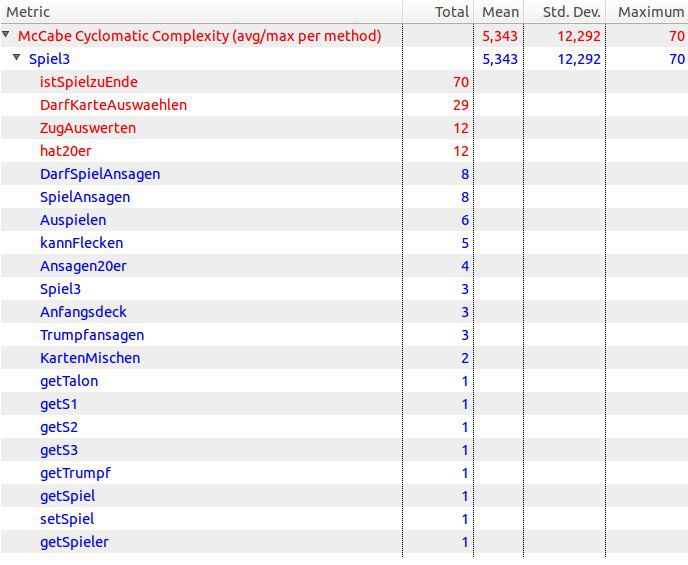
### Metriken zu Spiel3.cs

Die Klasse Spiel3.java enthält im Durschnitt 0,886 Parameter für die 35 Methoden. Die Methoden „ZugAuswerten“ und „TalonAustauschen“ haben 3 Parameter. Es sind keine statischen Attribute enthalten, aber 14 mit private gekennzeichnete. SLOC beträgt 858. Je höher die Verschatelungstiefe desto schwerer ist der Code zu verstehen (Verschachtelungstiefe >5 = erschwertes Programmverständnis).



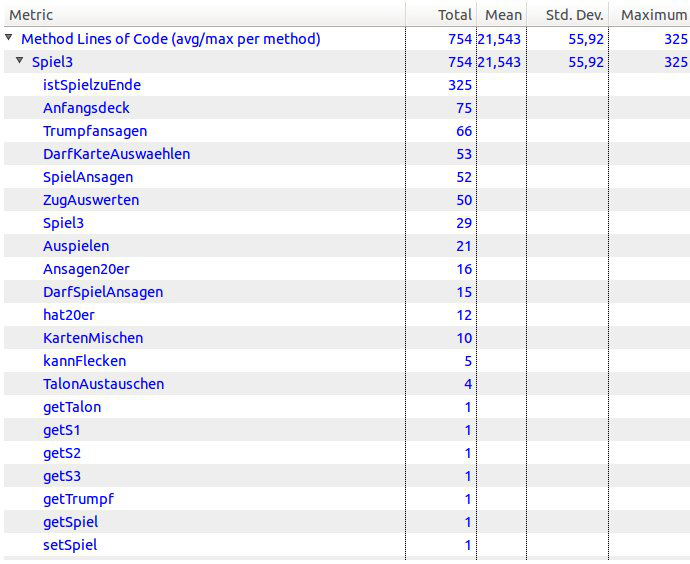
### McCabe Metrik zu Spiel3.cs

McCabe-Zahl >10: schlechte Wartungseigenschaften

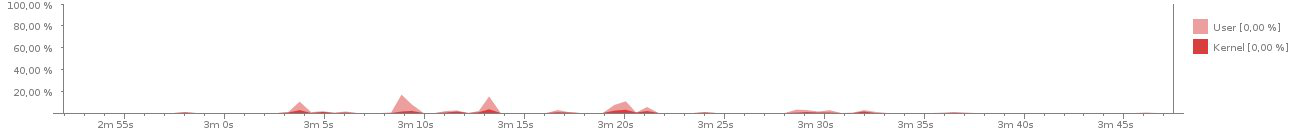


### Method Lines of Code Metrik zu Spiel3.cs

McCabe-Zahl >10: schlechte Wartungseigenschaften

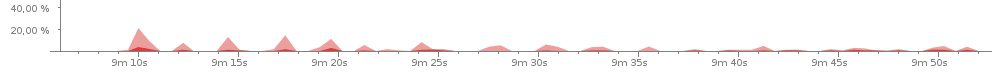


### Leistung CPU Metrik - Menu

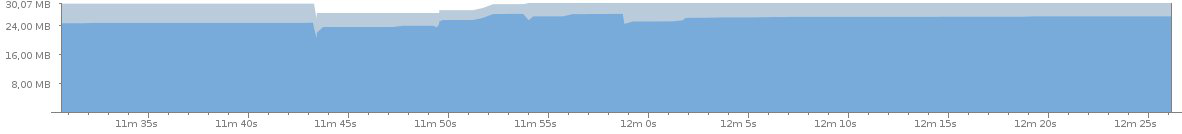


* Peak1: „Name ändern“
* Peak2: „Schnelles Spiel“
* Peak3: „Spielregeln“
* Peak4: „Spielen“
* Peak5: „Statistik“
* Peak6: „Beenden“

### Leistung – Schnelles Spiel (1 Runde)



### Memory – Schnelles Spiel (1 Runde)



### Datenverbrauch

* Der Datenverbrauch pro Runde beträgt für 2 Spieler ca. 23 KB.

# Anhang

### Protokoll Kick-off Meeting

* Besprochene Punkte:
  + Aus welchem Grund wollen sie diese App erstellen?
  + Wie sieht die Zielgruppe aus?
  + Welches Spiel sollte genau implementiert werden?
  + Welche Spielregeln sollten dem Spiel zugrunde liegen?
  + Wie hoch sollte die maximale Spieleranzahl? Bei Schnapsen z.B. maximal 4 Spieler
  + In der Kurzbeschreibung wurde erwähnt, dass alle Spieler vor Ort sein sollen. Kann Implementierung via Bluetooth erfolgen.
  + Wie sollte es möglich sein zu Schummeln, Schummeln zu unterbinden und in die Karten eines anderen zu schauen?
  + Wie sollten Benutzer Gesten, Sprachsteuerungen, Bewegungen zur Spielbeeinflussung nutzen können?
  + Genügt die Unterstützung ab Android Version 4.0
  + Welche Produktfunktionen sind Musskriterien?
  + Sind Punktlisten ein Muss-/Wunschkriterium?
  + Gibt es einen Zeitrahmen / Budgetrahmen?
  + Sind Wartungstätigkeiten zu erwarten? Wenn ja, werden dafür nach Projektfertigstellung eigene Wartungsverträge vereinbart oder soll dies gleich in die Aufwandsschätzung mit einfließen?
  + Gibt es für Sie noch wesentliche Punkte die Musskriterien sind und in der bisherigen Diskussion nicht besprochen wurden?

### Wichtigste Konzepte von Scrum

* + **Daily Scrum Meeting:** Tägliche maximal 15 minütige Meetings des Entwicklungsteams und des Scrum Masters. Besprochen werden die zuletzt erledigten und die als nächstes anfallenden User Stories.
  + **Product Owner:** Er repräsentiert den Kunden und vertritt die Interessen des Kunden.
  + **Product-Backlog:** Geordnete Auflistung aller User Stories eines Softwareproduktes.
  + **Scrum Master:** Person die das Scrum-Team vom Management abschirmt, aber nicht Teil des Scrum-Teams oder des Managements ist. Sorgt für einen fortlaufenden Entwicklungsprozess.
  + **Scrum Team:** Entwicklungsteam ohne fixe Zuteilung von Aufgaben über alle Sprints hinweg.
  + **Scrum:** agiles Softwareentwicklungs-Framework.
  + **Sprint Review Meeting:** Scrum-Team präsentiert Ergebnisse des letzten Sprints.
  + **Sprint:** kurze Entwicklungszyklen in der Länge von 2 bis 6 Wochen.
  + **Sprint-Backlog:** Geordnete Auflistung aller User Stories die innerhalb eines Sprints abgearbeitet werden.
  + **Story Point:** Einheit zur Größenschätzung einer User Story.
  + **User Story:** Beschreibung eines konkreten Anwendungsfalls aus der Sicht des Nutzers.

### Übersicht zur Android Programmierung

Die Entwicklung dieses Software-Produkts wird mit der Hilfe von Android Studio erfolgen. Dies ist ein IDE für das Java-Android-Framework und wurde von Google entwickelt. Grundsätzlich gibt es einen Source-Ordner mit ein oder mehreren \*.xml-Dateien und darin befinden sich Informationen über die GUI-Elemente. Außerdem kommen auch verwendete Bilder in diesen Ordner. Des Weiteren gibt es eine Main-Datei, von der aus die App gestartet wird. Die wichtigsten Funktionen der App befinden sich in dieser Main-Datei. Beim Erstellen eines neuen Projektes kann die Android-Version ausgewählt werden und es wird automatisch angezeigt zu viel Prozent die App mit verschiedenen Android-Geräten kompatibel ist. Man kann innerhalb des Android Studios einen Emulator starten um die App zu testen. Leider kann mit dem Emulator keine Mehrspieler-App die via WLAN funktioniert getestet werden. Zu Beginn des Projektes hatte keines der Teammitglieder praktische Erfahrung mit der Android-App Programmierung.

Die Programmierung benötigt keine speziellen Kenntnisse. Grundlegende Java-Kenntnisse reichen aus um die Implementierung vorzunehmen. Beim Design ist darauf zu achten, dass alle GUI Elemente und Activities für unterschiedliche Bildschirmgrößen (Tablet, Smartphones) angepasst werden.

Die Mehrspielerverbindung wurde mit Hilfe von der Google Nearby Connection implementiert.

### Handhabung der Zeiterfassung

Für die Zeiterfassung wurde die kostenlose Online-Plattform toggl verwendet. Die Beschreibung des Zeiterfassungseintrags hat folgende Syntax: „Tätigkeit – User Story Titel“. Tätigkeit kann Analyse, Implementierung, QA, Dokumentation oder Meeting sein. Die Zeiterfassungseinträge werden maximal auf 15 Minuten genau verbucht.

### Zeiterfassungseinträge von Appsolute Games

* Zeiterfassungen gesammelt: [Zeiterfassung\Zeiterfassung\_Gesamtbericht.pdf](Zeiterfassung/Zeiterfassung_Gesamtbericht.pdf)
* Zeiterfassung Eberhard: [Zeiterfassung\Zeiterfassung\_Eberhard.pdf](Zeiterfassung/Zeiterfassung_Eberhard.pdf)
* Zeiterfassung Eder: [Zeiterfassung\Zeiterfassung\_Eder.pdf](Zeiterfassung/Zeiterfassung_Eder.pdf)
* Zeiterfassung Maier: [Zeiterfassung\Zeiterfassung\_Maier.pdf](Zeiterfassung/Zeiterfassung_Maier.pdf)
* Zeiterfassung Pachatz: [Zeiterfassung\Zeiterfassung\_Pachatz.pdf](Zeiterfassung/Zeiterfassung_Pachatz.pdf)
* Zeiterfassung Platter: [Zeiterfassung\Zeiterfassung\_Platter.pdf](file:///D:\Uni\Semester%204_Informatik%20&%20Infomgmt\SE2\UE\Schnaps2gether\Produktmappe\Zeiterfassung\Zeiterfassung_Platter.pdf)

### Testberichte

* Sammlung der Testberichte: <Testberichte>

### Burndowncharts

* Sammlung der Burndowncharts: <Burndowncharts>

### Sprintbacklogs und Product-Backlog

* Sammlung der Backlogs: <Backlogs>