HSR
HOCHSCHULE FÜR TECHNIK
RAPPERSWIL
FHO Fachhochschule Ostschweiz

IFS INSTITUTE FOR SOFTWARE

Hochschule für Technik Rapperswil

Studienarbeit

Abteilung Informatik

Methode 635 als Cross Plattform App mit Xamarin

Authoren:
Elias Brunner & Oliver Dias

Betreuer:
Prof. Dr. Olaf Zimmermann

20. Dezember 2018

Aufgabenstellung

Die komplette Aufgabenstellung wurde von Prof. Dr. Zimmermann erstellt und kann dem Anhang A entnommen werden.

Ausgangslage

Gerade in der IT-Welt ist das Finden von Lösungen für neu auftretende, aber auch für bekannte Probleme ein wichtiger Bestandteil des Jobs. Durch den Einsatz von Innovationsmethoden kann der Ideenfindung auf die Sprünge geholfen werden. In diesem Bereich finden sich die verschiedensten Ansätze und Methoden; in dieser Studienarbeit soll die Methode 635 [1] verwendet werden. Methode 635 ist eine Kreativitäts- und Brainwriting-Technik, welche die Entwicklung von neuen, ungewöhnlichen Ideen für Problemlösungen in der Gruppe fördert. Die Methode wird im PQM-Modul an der HSR vorgestellt.

Es gibt nach heutigem Kenntnisstand noch keine mobile App, die die Methode 635 unterstützt. Die Motivation für die Studienarbeit besteht darin, eine Cross-Plattform App zu konzipieren und zu implementieren. Dabei sollen moderne Technologien zum Einsatz kommen, welche es den Anwendern ermöglichen, schneller und einfacher eine Lösung für ein Problem zu erarbeiten.

Ziele der Arbeit und Liefergegenstände

In der Studienarbeit soll die Methode 635 als SmartPhone App umgesetzt werden. Android- und iOS- Support soll durch Verwendung von Xamarin erreicht werden. Es wird erwartet, dass bis zum Ende des Projektes eine lauffähige und getestete Cross-Plattform Applikation umgesetzt wird, welche es Benutzern ermöglicht, die Methode 635 effektiv und effizient auf ihre Probleme anzuwenden.

Damit die App einen Mehrwert gegenüber der Papierversion bietet, soll es z.B. möglich sein, die Anzahl der Teilnehmer variabel zu bestimmen oder verschiedene Medien (Text, Video, Bilder, etc.) zu verwenden bzw. einzubinden. Die Persistierung der bearbeiteten Problemstellungen soll aus Sicht des Kunden einfacher möglich sein als dies mit Papier möglich ist. Ein weiterer Vorteil einer mobilen Anwendung ist, dass Anwender die Methode 635 nutzen können auch wenn sie nicht am selben Ort sind oder die Lösungsvorschläge nicht zur selben Zeit bearbeiten.

Die Vision der Arbeit ist also, die Papierversion für diese Methodik zu funktional und qualitativ zu überbieten. Dabei spielen Erfolgsfaktoren wie einfache und intuitive Bedienung der App und ein unkompliziertes Reporting sowie Robustheit und Stabilität (Bsp. keine Zeit- und Datenverluste) eine wichtige Rolle.

Weitere kritische Erfolgsfaktoren sind:

- Konfigurierbarkeit (z.B. Anzahl Teilnehmer und Schritte) und Erweiterbarkeit (im Hinblick auf Folgearbeiten, die u.U. auch andere Brainstorming

Methoden unterstützen)

- sinnvolle Ausnutzung der Smartphone-Fähigkeiten, um einen Mehrwert im Vergleich zur traditionellen, papiergestützten Methode zu erreichen
- Validierung der Konzepte und ihrer Implementierung mit Hilfe von User Tests in mindestens einem Anwendungsbereich (Bsp. Architekturentscheidungen und -optionen).

Abstract

Mobile Applikationen für das Smartphone haben in den letzten Jahren immer mehr an Bedeutung gewonnen. Das liegt unter anderem daran, dass sie einfach zu bedienen sind und man sein eigenes Smartphone immer und überall dabei hat. Dieser Umstand hat uns in der vorliegenden Arbeit dazu bewogen, die Methode 635 als Cross-Plattform Applikation für iOS und Android zu implementieren. Die Methode 635 ist eine Kreativitäts- und Brainwriting-Technik, welche die Entwicklung von neuen, ungewöhnlichen Ideen für Problemlösungen in der Gruppe fördert. Die Methode wird im Modul "Projekt und Qualitätsmanagement" an der HSR vorgestellt.

Dafür haben wir eine Vorstudie angefertigt, in der wir uns zunächst genauer mit der Methode 635 auseinander gesetzt haben. Ebenfalls beschäftigten wir uns während der Vorstudie mit der Frage, ob sich Xamarin für unser Projekt überhaupt anbietet und welches der beiden Umsetzungsarten (Xamarin.Forms oder Xamarin native) für unser Projekt besser geeignet ist.

Aus diesen und weiteren Analysen und Konzepten entwickelten wir einen einfachen Prototypen. Zweck des Prototypen war es primär zu klären, ob unsere theoretischen Überlegungen auch praktisch funktionierten und dieser in der Lage war, die definierten NFAs abzudecken. Da dies der Fall war, entwickelten wir basierend auf unserem Prototypen das gesamte System weiter.

Aus der Arbeit ist eine lauffähige Cross-Plattform Applikation für iOS und Android hervorgegangen, welche es Benutzern ermöglicht, die Methode 635 auf ihre Probleme anzuwenden. Weiter dokumentierten wir Erkenntnisse über Herausforderungen, die während der Umsetzung aufgetreten sind. So können unsere Erfahrungen anderen Entwicklern bei ähnlichen Projekten helfen oder einzelne Risiken gar komplett verringern.

Inhaltsverzeichnis

1 Management Summary

1.1 Ausgangslage

In einer Zeit, in der der technische Fortschritt immer mehr an Bedeutung gewinnt, hat man das Gefühl alles könne automatisiert und durch die Technik schneller und präziser erledigt werden. Doch bei der Aufgabe eine Lösung für ein bestimmtes Problem zu finden, ist der menschliche Ideenreichtum unumgänglich. Durch den Einsatz von verschiedensten Innovationsmethoden kann der Ideenfindung häufig etwas nachgeholfen werden. Dazu zählt auch die Methode 635. Die Methode 635 ist eine Kreativitäts- und Brainwriting-Technik, welche die Entwicklung von neuen, ungewöhnlichen Ideen für Problemlösungen in der Gruppe fördert.

Nach heutigem Wissenstand hat sich dafür noch keine mobile Applikation in den App Stores von Google oder Apple durchgesetzt. Daher ist es das Ziel dieser Arbeit, die Methode 635 als mobile Cross-Plattform Applikation zu konzipieren und zu implementieren. Die Möglichkeiten der Technik sollen dazu genutzt werden, eine Lösung für ein Problem schneller und einfacher zu erarbeiten.

1.2 Vorgehen

Das Vorgehen für unser Projekt lässt sich in zwei Phasen einteilen. In der ersten Phase konzentrierten wir uns primär auf Recherchearbeiten und konzeptionelle Arbeiten. Dabei führten wir eine Vorstudie durch, in der wir grundsätzliche Analysen durchführten und Fragen klärten. Des Weiteren konzipierten wir zunächst theoretisch wie wir die Methode 635 als Cross-Plattform Applikation umsetzen sollten. Eine weitere Aufgabe war das Verfassen der funktionalen sowie nicht-funktionalen Anforderungen. Um unsere Vorüberlegungen abzusichern, programmierten wir zu Ende dieser Phase einen einfach Prototypen.

In der zweiten Phase ging es dann darum, alle unsere Überlegungen und Designs komplett umzusetzen. Da unser Prototyp wie gewünscht funktionierte, entwickelten wir basierend darauf alle weiteren Funktionen. Damit wir sicherstellen konnten, dass die Applikation wie gewünscht funktionierte, testeten wir während der gesamten Dauer manuell unser System.

1.3 Ergebnisse

Aus der Arbeit ist eine lauffähige Cross-Plattform Applikation für iOS und Android hervorgegangen, welche es Benutzern ermöglicht, die Methode 635 auf ihre Probleme anzuwenden. Die Applikation ist sehr variabel gestaltet. So kann die Anzahl an Teilnehmer pro Gruppe sowie die Anzahl an Ideen pro Runde und die Startdauer pro Runde individuell bestimmt werden.

Durch das Persistieren in einer zentralen Datenbank ist es dem Endnutzer zudem möglich, seine erarbeiteten Ideen jederzeit abzurufen und weiter zu verwenden.

In einer weiteren Version ist es zudem denkbar, weitere Features wie das Erfassen von Bildern und Weblinks zu implementieren.

2 Technischer Bericht

2.1 Einleitung und Übersicht

Der technische Bericht besteht neben diesem Kapitel noch aus zehn weiteren Unterkapiteln. Die Kapitel "Was ist Xamarin?" und "Was ist die Methode 635?" stellen eine Einführung in die Thematik dar und erklären, um was es in unserer Arbeit geht. Bevor wir uns mit der eigentlichen Umsetzung beschäftigten, führten wir eine Vorstudie durch, in der wir grundsätzliche Analysen durchführten und Fragen klärten. All dies ist im Kapitel "Vorstudie" niedergeschrieben. Im Kapitel der "Anforderungsspezifikation" listen wir alle funktionalen sowie nichtfunktionalen Anforderungen als Use-Cases auf. Das Domain-Modell findet sich im Kapitel "Domainanalyse". Die "Architekturdokumentation" ist das nächste Kapitel und dokumentiert die logische Architektur, sowie das Deployment. Jegliche Entscheidungen, welche wir in Bezug auf die Architektur getroffen haben, werden im Kapitel "Architekturentscheide" begründet. Während der gesamten Umsetzung der Applikation sind natürlich auch Probleme aufgetreten. Diese haben wir im Kapitel "Herausforderungen" zusammengetragen. Die eigentliche Implementation haben wir im Kapitel "Ergebnisse" dokumentiert. Zum Schluss blicken wir im Kapitel "Schlussfolgerungen" nochmals kritisch auf unser Projekt zurück und bewerten unsere Arbeit und geben einen Ausblick, wie man die Applikation noch erweitern könnte.

2.2 Was ist Xamarin?

Im Mai 2011 gründete Miguel De Icaza die Firma Xamarin mit dem Ziel, die Entwicklung für Cross-Platform-Applikationen für Smartphones zu vereinfachen und zu beschleunigen [2].

Die Idee für die Entwicklung von Cross-Platform-Applikationen hatte Miguel De Icaza allerdings schon 10 Jahre zuvor, als er 2001 das Projekt Mono ins Leben gerufen hatte. Beim Projekt Mono handelt es sich um eine quelloffene Implementation von Microsofts .Net Framework. Die Entwicklung plattformunabhängiger Applikationen ist das Ziel des Projektes [3]. Die Entwickler achten bei der Implementation auf die Einhaltung der Standards für die Common Language Infrastrucute (CLI) und Common Language Specification (auch .Net Framework genannt) [4].

Mono wird stetig weiterentwickelt und ist fester Bestandteil der Xamarin Plattform. Softwareentwickler, welche mit der Xamarin Plattform arbeiten, können Apps für iOS, Android und WindowsPhone in C# schreiben. Der geschriebene Quellcode kann durchschnittlich zu 75 Prozent für alle Plattformen benutzt werden. Die in C# geschriebenen Applikationen werden von der Xamarin Plattform in die jeweilige native Sprache übersetzt, was gewährleistet, dass am Ende des Entwicklungspro-

zesses eine native Applikation für das jeweilige Betriebsystem zur Verfügung steht. Neben dem mobilen Betriebssystemen ist auch die Entwicklung für Mac und Windows möglich [3].

Im Februar 2016 wurde Xamarin von Microsoft aufgekauft und ist seither eine Tochtergesellschaft von Microsoft mit Sitz in San Francisco [2].

2.3 Was ist die Methode 635?

Die gesamte Beschreibung der Methode 635 wurde von kreativitästechniken.info [1] übernommen.

Die Methode 635 (auch Methode 6-3-5 geschrieben) ist eine Brainwriting-Kreativitätstechnik. Der Name leitet sich aus den drei wesentlichen Eigenschaften der Methode ab: jeweils 6 Teilnehmer erhalten ein Blatt Papier, auf dem sie je 3 Ideen notieren und die Blätter dann insgesamt 5 mal weiterreichen.

Anwendungsgebiete der Methode 635

Die Methode 635 ist eine Variante des Brainwriting. Sie eignet sich besonders für die erste Phase im kreativen Prozess. Dabei werden zunächst Ideen gesammelt ohne dass eine Bewertung stattfindet. Im Idealfall können so in kurzer Zeit 108 Ideen entstehen. Die Aufforderung, bestehende Ideen aufzugreifen und weiterzuentwickeln macht die Methode 635 zu einer konstruktiven Kreativitätstechnik. Gleichzeitig kann die Kreativität durch die strukturierte Form aber auch gebremst werden. In der Praxis entstehen daher oft etwas weniger Ideen.

Vorgehen bei der Methode 635

Zunächst erklärt der Moderator die Regeln der Methode 635, führt die Teilnehmer in das Ausgangsproblem ein und ist im Folgenden verantwortlich für die Zeitmessung. Sobald die Teilnehmer über die Ausgangsfrage oder -problem aufgeklärt sind, startet der Moderator die erste von sechs Runden. In jeder Runde werden die Teilnehmer aufgerufen, die oberste noch freie Zeile, bestehend aus 3 Kästchen, mit ihren Ideen zu füllen. Dabei sollten die Teilnehmer die Ideen der Vorgänger aufgreifen, erweitern und/oder weiterentwickeln ohne die Ideen zunächst zu bewerten. Nach einer festgelegten Zeit von beispielsweise 5 Minuten beendet der Moderator die Runde. Die Teilnehmer reichen ihr Arbeitsblatt im Uhrzeigersinn an ihren Sitznachbarn weiter und eine neue Runde beginnt. Im Idealfall sind nach 6 Runden genau $6 \cdot 18 = 108$ Ideen entstanden. In Realität ist die Anzahl aufgrund von doppelten oder leeren Einträgen wahrscheinlich etwas geringer. Dennoch sollten nun eine Vielzahl von Ideen vorliegen.

Nun kann eine Diskussion, Analyse und Bewertung der Ideen erfolgen.

2.4 Vorstudie

In der Vorstudie beschäftigten wir uns zuerst mit der Methode 635 an sich, um ein Gefühl dafür zu bekommen, wie es ist, diese selbst an einem konkreten Problem zu nutzen.

Weiter beschäftigten wir uns mit Xamarin. Hierbei war vor allem der Entscheid zwischen Xamarin.Forms und Xamarin native von grosser Bedeutung, da dieser im späteren Projektverlauf kaum mehr rückgängig zu machen ist.

Ein weiterer Teil der Vorstudie bestand darin eine passende Umgebung für das automatische Deployen der Xamarin Applikation zu finden.

Zum Schluss der Vorstudie beschäftigten wir uns mit der Frage, welches Backend wir für unser Projekt verwenden sollten.

Einige wichtige Punkte und Entscheide waren stark von der Vorstudie abhängig. Es war daher entscheidend, bei der Vorstudie sorgfältig zu arbeiten.

2.4.1 Erste Erfahrungen mit der Methode 635

Bevor wir uns mit den technischen Details der Umsetzung zur Cross-Plattform App auseinander gesetzt haben, spielte jeder der beiden Projektmitgliedern die Methode 635, wie in Kapitel 2.3 beschrieben, mit seiner Familie, Bekannten oder Freunden einmal durch.

Dabei wurden folgende persönliche Erfahrungen und Beobachtungen gemacht:

Diskussion gestartet

Schon nach 2-3 Runden konnte beobachtet werden, dass sich spannende Diskussionen zum gestellten Problem entwickelten. Die Teilnehmer mussten sogar angehalten werden die Diskussionen auf dem Papier weiterzuführen, da sonst die Aussagen verloren gehen.

Interessante Methode

Die Teilnehmer empfanden die Methode 635 als spannend und interessant. Durch die einzelnen Teilnehmer waren auch verschiedenste Blickwinkel auf das Problem vertreten, was wiederum zu unterschiedlichsten neuen Ideen führe.

Wertung einführen

Was allerdings als kleiner Nachteil empfunden wurde, war der Umstand, dass die Methode 635 in der originalen Version keine Möglichkeit für Wertungen bietet.

Als Mensch bildet man sich während des Lesens der einzelnen Ideen automatisch eine Meinung darüber und wird dadurch stark dazu verleitet,

eine wertende Bemerkung statt einer neuen oder angepassten Idee zu schreiben.

Eine Verbesserung könnte darin bestehen, eine Möglichkeit für Wertungen der Ideen einzuführen. Dies könnte so aussehen, dass man neben dem Text auch ein Label (z.B. Pro oder Contra) und die Idee, welche man bewerten will, auswählen kann.

Zielstrebige Lösungsfindung Da die zwei Durchführungen nacheinander stattgefunden haben, konnten im zweiten Versuch die Teilnehmer darauf hingewiesen werden, dass explizit keine Wertung, sondern eine Erweiterung der vorhergehenden Idee erstellt werden soll. Durch diesen Hinweis konnten am Ende des Brainstormings sehr interessante und brauchbare Ergebnisse erzielt werden.

2.4.2 Backend-Technologie

Das PlayFramework [5] ist ein, unter der Apache 2 Lizenz stehendes, Web Application Framework. Es folgt dem MVC-Pattern und erleichtert das Erstellen von Webanwendungen. Durch den Aufbau ermöglicht es dem Entwickler unter anderem auf einfache Art und Weise eine RESTful API zu schreiben.

Play basiert auf einer zustandslosen und schlanken Architektur. Da es auf Akka aufgebaut ist, nützt es eine vollständig asynchrones I/O. Ausserdem bietet Play minimalen Ressourcenverbrauch.

Sobald am Code eine Änderung vorgenommen wurde, wird der Server automatisch neugestartet, was die Produktivität des Entwicklers weiter erhöht.

Jeglicher Code, welcher nicht kompiliert werden konnte, wird zudem im Browser angezeigt und man weiss als Entwickler sofort auf welcher Zeile man nachbessern muss.

Seit der Version 2.0 des PlayFrameworks ist der Framework Core in Scala geschrieben und als Build Tool wird SBT verwendet. Für das Testen stehen verschiedene Test Frameworks sowohl für Java als auch für Scala zur Verfügung. Mittels Scalatest oder JUnit können Unit Tests für beide Sprachen geschrieben werden. Es ist auch möglich Tools wie scoverage für die Code Coverage zu verwenden.

2.5 Anforderungsspezifikation

Bei der Analyse der verschiedenen Anforderungen an unser Projekt haben wir die entscheidenen funktionalen und nicht-funktionalen Anforderungen definiert und in diesem Kapitel spezifiziert.

2.5.1 Funktionale Anforderungen

In den folgenden Kapiteln werden jegliche funktionalen Anforderungen an die Applikation als sogenannte Use-Cases beschrieben. Als Übersicht dient das Use-Case-Diagramm, wie in Abbildung 1 dargestellt.

2.5.1.1 Brief Use-Cases

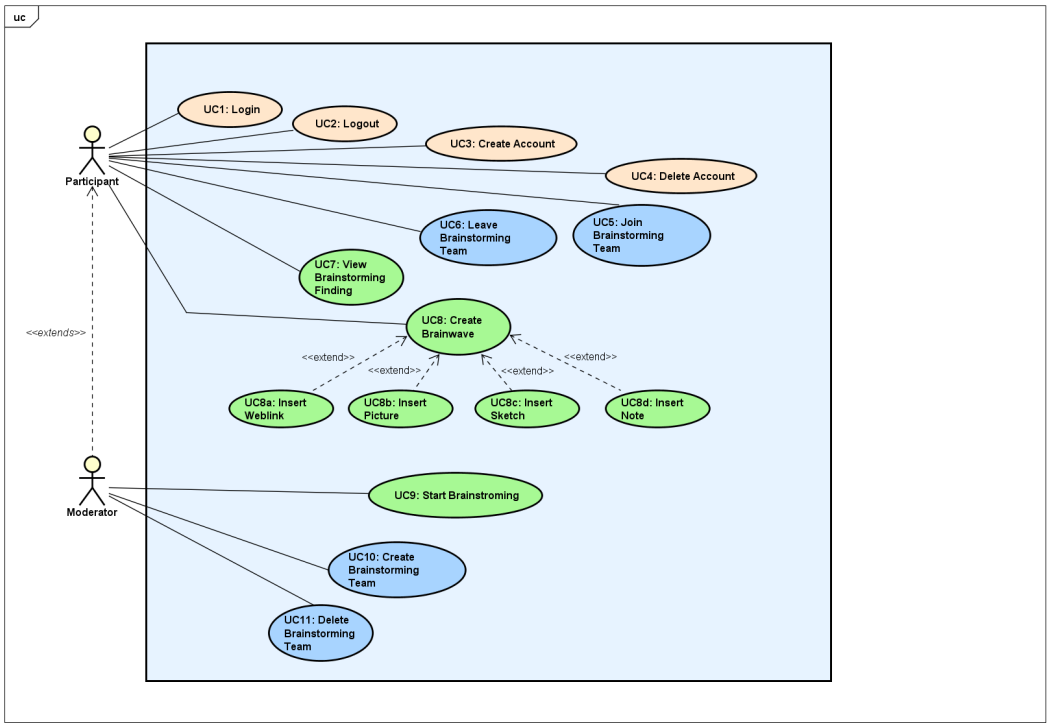

Abbildung 1: Use-Case-Diagramm

Jeder Use-Case hat den nachfolgend kurz (*briefly*) beschriebenen Funktionsumfang. Die Hauptaktivitäten UC7-9 sind am Schluss *fully-dressed* beschrieben.

***UC1:* Login** — Als Participant möchte ich mich mit Benutzernamen und Passwort in das System einloggen können.

***UC2:* Logout**	Als eingeloggter Participant will ich mich ausloggen, sodass das Startfenster wieder erscheint.
***UC3:* Create Account**	Als Participant möchte ich mich registrieren können.
***UC4:* Delete Account**	Als Participant will ich meinen erstellten Account wieder löschen können.
***UC5:* Join Brainstorming Team**	Als Participant will ich einem bereits existierenden Team beitreten können.
***UC6:* Leave Brainstorming Team**	Als Participant will ich ein beigetretenes Team verlassen können.
***UC7:* View Brainstorming Finding**	Als Participant will ich, nachdem eine Brainstorming Session durchgeführt wurde, das Resultat (*Finding*) meiner Gruppe einsehen können.
***UC8:* Create Brainwave**	Als Participant will ich während einer Brainstorming Session ein Brainwave (bestehend aus mehreren Ideen) erstellen und einreichen können.
***UC8a:* Insert Weblink**	Als Participant will ich einen Weblink in mein aktuelles Sheet einfügen können.
***UC8b:* Insert Picture**	Als Participant will ich ein Bild in mein aktuelles Sheet einfügen können.
***UC8c:* Insert Sketch**	Als Participant will ich in mein aktuelles Sheet zeichnen können.

***UC8d:* Insert Note**	Als Participant will ich normalen Text in mein aktuelles Sheet einfügen können (Defaultoption).
***UC9:* Start Brainstorming**	Als Moderator will ich eine Brainstorming Session starten.
***UC10:* Create Brainstorming Team**	Als Moderator will ich ein Brainstorming Team erstellen können.
***UC11:* Delete Brainstorming Team**	Als Moderator will ich ein Brainstorming Team löschen können.

2.5.1.2 Fully-Dressed Use-Cases

Die fully-dressed Use-Cases folgen den im Modul *Software Engineering 1* empfohlenen Punkten.

Use-Case 7: View Brainstorming Finding	
Primary Actor	Participant
Stakeholders & Interests	Ein Participant wünscht sich eine Übersicht von allen Ideen der Gruppenmitglieder. Er will das Gesamtresultat (*Brainstorming Finding*) einsehen und daraus etwas lernen.
Preconditions	Participant existiert im System (UC3), ist eingeloggt (UC1) und einer Gruppe beigetreten (UC5). Zudem hat die Gruppe des Participants alle Runden durchgemacht.
Post Conditions/Success Guarantee	Die Notizen aller Participants werden übersichtlich dargestellt. Das heisst, jede Ausgangsidee ist mit deren Ergänzungen von den verschiedenen Participants ersichtlich.

Main Success Scenario/Basic Flow	1. Der Participant schliesst die abschliessende Runde als Letzter ab. 2. Das System speichert seine Notizen auf dem Server. 3. Das System zeigt dem Participant eine Meldung, dass das Finding einsehbar ist. 4. Der Participant clickt auf "Resultate einsehen"(o.Ä.). 5. Das System zeigt dem Participant eine Übersicht mit allen Notizen der Teilnehmer.

Alternative Flows	Alternative Success Scenario 1: 1.a Der Participant schliesst die Runde nicht als Letzter ab. 1.b Das System speichert seine Notizen auf dem Server und zeigt dem Benutzer die verbleibende Zeit an. 1.c Der Participant aktualisiert den gezeigten Screen nach dem Ablauf der Zeit. Alternative Success Scenario 2: 4.a Der Participant clickt nicht auf "Resultate einsehen", sondern navigiert zurück auf den Homescreen. 4.b Das System zeigt dem Participant den Homescreen an. 4.c Der Participant navigiert auf seine Gruppe. 4.d Das System zeigt dem User die Gruppe an. 4.e Der Participant will sich die Resulate dieser Gruppe anzeigen lassen und clickt entsprechenden Button. 4.f Das System zeigt dem Benutzer die Übersicht an.
Frequency of Occurrence	Oft, Kernfunktionalität.

Use-Case 8: Create Brainwave	
Primary Actor	Participant
Stakeholders & Interests	Ein Participant will, dass seine Brainwave erfasst wird.
Preconditions	Der Participant muss existent (UC3) sowie eingeloggt (UC1) sein. Des Weiteren muss ein Brainstorming Team existieren (UC10), zu welchem er gehört. Die Brainstorming Runde muss ebenfalls gestartet worden sein (UC9).

Post Conditions/Success Guarantee	Vom Participant erfasste Notizen werden erfolgreich auf dem System gespeichert.
Main Success Scenario/Basic Flow	1. Der Participant erfasst während der aktuellen Rundenzeit Notizen. 2. Das System zeigt diese Notizen an. 3. Der Participant beendet frühzeitig die Runde und clickt auf "Brainwave abgeben"(o.Ä). 4. Das System persistiert die Notizen und zeigt dem Participant eine Bestätigung an. 5. Der Benutzer kann aktualisieren, um den nächsten Rundenstart nicht zu verpassen.
Alternative Flows	Alternative Success Scenario 1: 3.a Der Participant beendet die Runde nicht manuell. 3.b Das System zeigt dem Participant eine Meldung an, dass die Zeit der Runde abgelaufen ist. Es persistiert die bis zu dem Zeitpunkt erfassten Notizen.
Frequency of Occurrence	Sehr oft, passiert pro Brainstorming-Session mehrmals.

Use-Case 9: Start Brainstorming	
Primary Actor	Moderator
Stakeholders & Interests	Ein Moderator will eine neue Brainstorming-Session starten können.

Preconditions	Der Moderator muss existent (UC3) sowie eingeloggt (UC1) sein. Des Weiteren muss ein Brainstorming Team existieren (UC10), das er erstellt hat. Zudem muss die Gruppe komplett sein (konfigurierte Anzahl an Participants sind dem Team beigetreten).
Post Conditions/Success Guarantee	Alle Participants des Teams können aktualisieren und sehen die gestartete Runde.
Main Success Scenario/Basic Flow	1. Der Moderator clickt auf "Brainstorming starten" (o.Ä) auf der Gruppe, die er erstellt hat. 2. Das System überprüft, dass alle Einstellungen korrekt sind und die korrekte Anzahl an Participants in der Gruppe sind. 3. Das System zeigt dem Moderator an, dass die Session gestartet ist. 4. Der Moderator kann wie die normalen Participants eine Brainwave (bestehend aus mehreren Ideen) erfassen.
Alternative Flows	-
Frequency of Occurrence	Oft, Kernfunktionalität.

2.5.1.3 Abuse-Cases

Um einem Missbrauch der Applikation entgegenzuwirken, sind neben den Use-Cases auch Abuse-Cases definiert. Diese helfen, mit unangebrachtem Inhalt und unangebrachter Verwendung umzugehen.

***AC1:* Unangebrachte Brainwaves** Ein Participant könnte unangebrachte [1] Inhalte in einer Gruppe hinzufügen. Um dies zu verhindern, könnte eine die Applikation um eine Funktion erweitert werden, die es dem Moderator erlaubt, Benutzer auszuschliessen.

[1] Als unangebrachte Inhalte werden Brainwaves mit rassistischen, pornographischen, sexistischen sowie unethischen Inhalten verstanden.

***AC2:* Missbrauch der Vertraulichkeit** Ein Participant könnte das Brainstorming Finding an die Konkurrenz leaken.

2.5.1.4 Sequenzdiagramm

Der Ablauf der Kernlogik ist der Abbildung 2 zu entnehmen. Darin ist der Prozess vom Erstellen des BrainstormingTeams (UC10) bis zum Abschliessen der Brainwave modelliert.

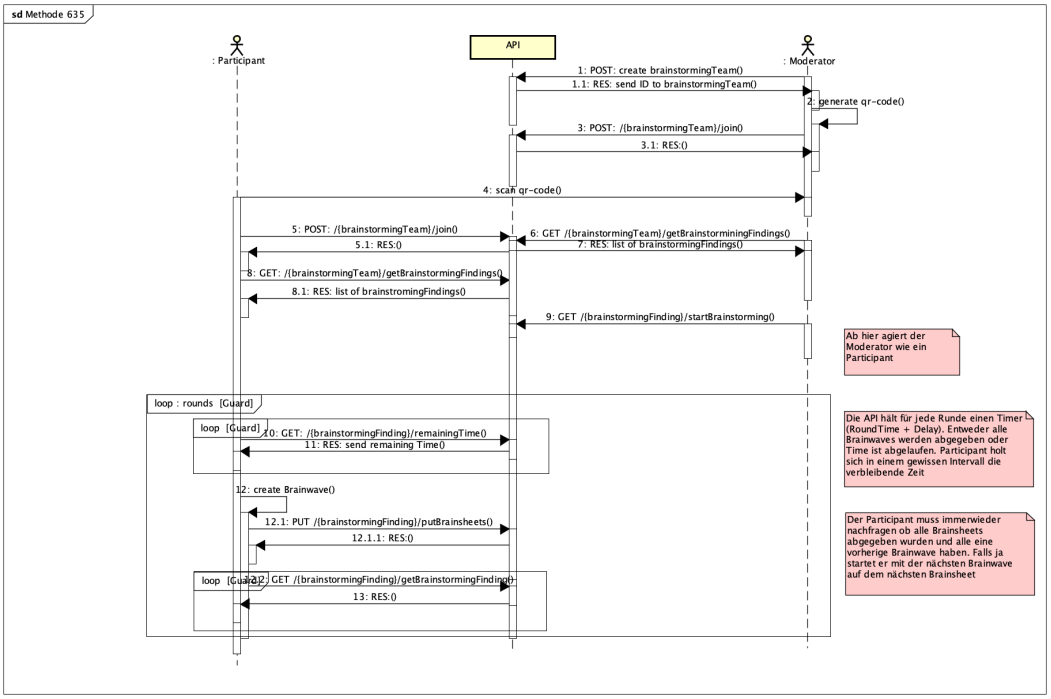

Abbildung 2: Ablauf der Kernlogik

2.5.2 Nicht-Funktionale Anforderungen

Beim Thema Nicht-Funktionale Anforderungen halten wir uns an die Standards ISO 9126[6] bzw. dessen Nachfolger ISO 25010[7]. Beide ISO-Normen sind sich sehr ähnlich und liefern eine gute Checkliste für jegliche Art von Systemanforderungen.

Diese Normen sind sehr umfangreich gestaltet. Wir werden uns daher auf die, für uns, wichtigsten Anforderungen konzentrieren. Um genaue und erfüllbare nicht-funktionale Anforderungen zu definieren, müssen die SMART-Kriterien [9] erfüllt sein.

Ressourcennutzung Die internen Ressourcen Kamera, Dateisystem dürfen nur bei effektivem Bedarf benützt werden. Die CPU-

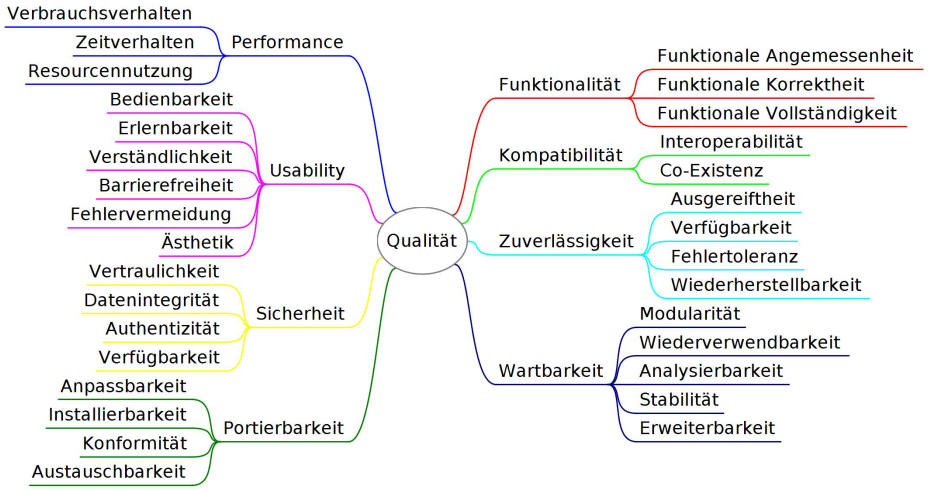

Abbildung 3: Anforderungskategorien nach ISO 25010
[8]

	Ressourcennutzung darf pro Minute im Bereich von bis zu 40% in Anspruch genommen werden.[2][3]
Bedienbarkeit	Wenn eine Aktion länger als 1-2s geht, soll dem User ein Wartesymbol angezeigt werden.
Ästhetik	Die Benutzeroberflächen der Applikation sind so gestaltet, dass die Elemente wiedererkennbar sind (Buttons haben gleichen Stil, leere Textfelder haben Platzhalter).
Vertraulichkeit	Die Daten einer Brainstorming Session können nur von der zugehörigen Gruppe eingesehen werden.
Anpassbarkeit	Die Anpassung bestehender oder Integration neuer Brainstorming-Methoden muss gewährleistet sein.
Installierbarkeit	Die Installation der Applikation auf einem Endgerät erfolgt durch das Ausführen eines *.apk oder *.ipa.[4] Dieser

[2] Referenzsystem Android: Huawei P10 mit Android Version 8.0.0 mit Hisilicon Kirin 960 CPU und 4GB RAM

[3] Referenzsystem iOS: iPhone 6 mit iOS Version 12 mit Dual-core 1.4 GHz Typhoon CPU und 1GB RAM

[4] Eine Applikation auf einem Android Smartphone hat üblicherweise die Endung *.apk. Beim iPhone bzw. beim iOS haben die einzelnen Applikationen die Endung *.ipa.

	Prozess soll unter 1 Minute geschehen.
Co-Existenz	Sollte zu einem späteren Zeitpunkt entschieden werden ein Web-Frontend zu programmieren, muss dieses co-existent mit der Xamarin Applikation existieren können.
Wiederherstellbarkeit	Im Falle eines fehlerhaften Features, muss es innerhalb eines Werktages möglich sein, die Applikation wieder auf den letzten funktionierenden Stand zurück zu holen und erneut zu deployen.
Analysierbarkeit	Das Ausführen eines Use-Cases muss durch Analyse von Logfiles erkennbar sein.

2.6 Domainanalyse

Das Domain-Modell besteht grob aus zwei Teilen: den Benutzern und der Brainstorming Methodik.

Dabei bilden mehrere Participants ein *Brainstorming Team*. Diese wird von einem der Participants, dem *Moderator*, gegründet. Das Team hat die Möglichkeit, ein oder mehrere *Brainstorming Findings* zu erarbeiten. Dies entspricht einer gesamten Durchgang der Methode. Der Moderator erstellt diese und hat die Möglichkeit, die Anzahl von Ideen sowie die erste Rundenzeit zu konfigurieren. Jede weitere Runde wird um eine Minute verlängert.

Das *Brainsheet* entspricht einem physikalischem Blatt, das herumgegeben wird. In der Standardkonfiguration 635 existieren also 6 Sheets (weil 6 Teilnehmer dabei sind).

Eine *Brainwave* ist das Produkt jedes Participants am Ende einer Runde. Es gehört in ein Brainsheet, das jede Runde an den nächsten Participant weitergegeben wird. In der Standardkonfiguration besteht eine Brainwave aus 3 Ideen (6**3**5).

Die *Idea* ist ein effektiv erarbeiteter Teil einer Brainwave. Im Normalfall ist eine Idee simpler Text (*NoteIdea*), wobei weitere Typen von Ideen (Bild, Weblink und Zeichnung) durch das verwendete Design erdenklich sind.

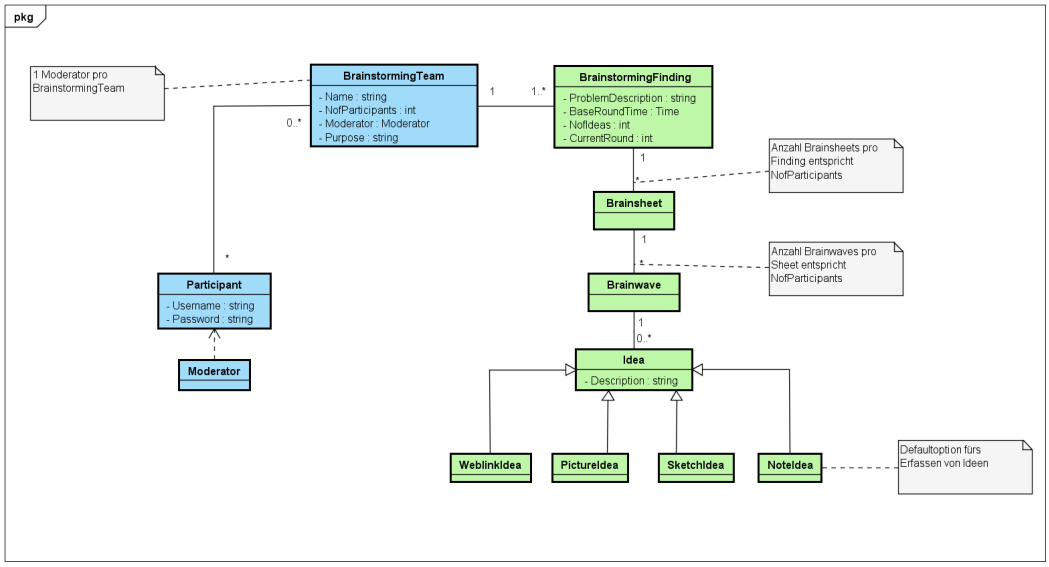

Abbildung 4: Domain Modell BrainingOutOfBox

2.7 Architekturdokumentation

In diesem Kapitel gehen wir detailliert in die Architektur und das Deployment unseres Projektes ein.

2.7.1 Logische Architektur

Wir teilen die Architektur des gesamten Systems in drei Schichten auf. In der Abbildung 5 sind diese als Presentation-, Businesslogic- und Persistence-Schicht zu erkennen.

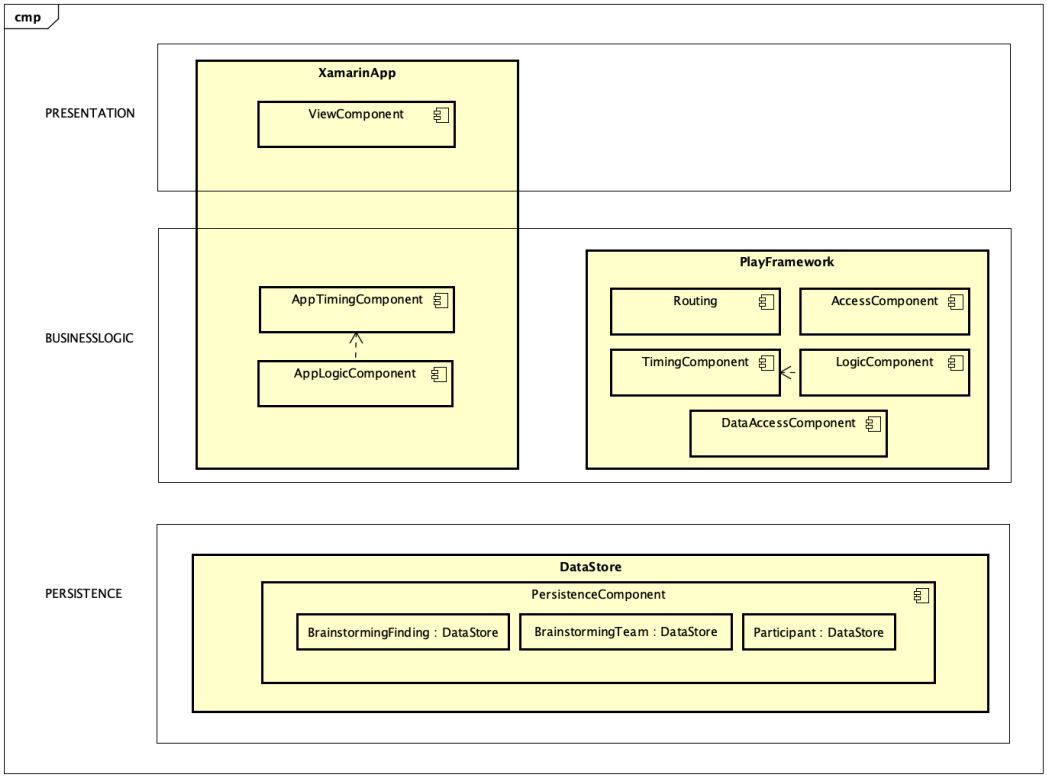

Abbildung 5: Logische Architektur BrainingOutOfBox

Die Präsentationsschicht ist die Schicht über die der Benutzer mit der Xamarin App kommuniziert. Konkreter gesagt, umfasst diese die verschiedenen View-Komponenten, welche für das Aussehen der App verantwortlich sind. Jegliche Interaktionen über die Oberfläche werden anschliessend in der Schicht der Businesslogic weiter verarbeitet. In dieser Schicht haben wir zum einen wieder unsere Xamarin App, welche selbst Logik-Komponenten wie die Timing-Komponente oder weitere App spezifische Logik-Komponenten enthält.

Die AppLogic-Komponente ist für das korrekte Verarbeiten und Weiterleiten der Eingaben an das PlayFramework verantwortlich.

Die AppTiming-Komponente ist zuständig für das Zeitmanagement während den einzelnen Runden. Diese Komponente überwacht daher die noch verbleibende Zeit.

Auf der anderen Seite haben wir das PlayFramework, welches wiederum Access-Komponenten, Routing-Komponenten, eine Timing-Komponente, Logik-Komponenten und eine DataAccess-Komponente enthält.

Die Timing-Komponente und die Logic-Komponente haben die selben Aufgaben wie ihre Gegenstücke in der Xamarin App. Auch hier verwalten diese das Zeitmanagement während den einzelnen Runden und stellen sicher, dass nach Ablauf der Zeit oder sobald alle *BrainSheets* abgegeben wurden, eine neue Runde beginnt. Des Weitern ist die Logic-Komponente zum Beispiel verantwortlich, dass ein *Participant* einer Gruppe nicht zweimal beitreten kann oder diese verlassen kann, wenn er sie schon einmal verlassen hat.

Die DataAccess-Komponente stellt sicher, dass jegliche Daten korrekt geladen oder gespeichert werden.

Mit der Schicht der Datenhaltung (Persistence) haben wir eine Schicht zur Verfügung, welche eine Persistence-Komponente hält.

Konkret steht uns je ein DataStore für die *BrainstormingFindings*, für die *BrainstormingTeams* und für die *Participants* zur Verfügung.

Komponenten

Nachfolgend sind nochmals alle Komponenten aufgelistet und kurz beschrieben. Für eine ausführlichere Beschreibung ist der Text oberhalb zu lesen.

ViewComponent Die View-Komponenten der Xamarin App sind für das korrekte Anzeigen der Informationen verantwortlich. Sie definieren das Aussehen der Applikation.

AppTimingComponent Die Xamarin App hält in der logischen Schicht eine Timing-Komponente, welche dafür sorgt, dass ein *BrainstormingFinding* nach Ablauf der Zeit abgesendet wird.

AppLogicComponent Die Logik-Komponente der Xamarin App regelt weitere Logik, wie z.B. den Zugriff auf das PlayFramework.

AccessComponent Die Access-Komponente auf dem PlayFramework regelt den Zugriff mittels JWT-Token[11]. JWT-Tokens werden bei erfolgreichem Login an den Benutzer der App gesendet. So kann sichergestellt werden, dass nur registrierte Benutzer mit dem PlayFramework interagieren können.

Routing Die Routing-Komponente sorgt anhand der URL für das Aufrufen der korrekten Funktion.

TimingComponent Wie die Xamarin App hält auch das PlayFramework eine Timing-Komponente, um den Zustand der Zeit verwalten zu können.

LogicComponent In der Logik-Komponente werden die eigentlichen Funktionen geschrieben. Hier ist auch die Logik für den Austausch der Blätter untergebracht.

DataAccessComponent Die DataAccess-Komponente stellt das Bindeglied zwischen dem PlayFramework und dem DataStore dar. Es ermöglicht erst den Zugriff auf die gespeicherten Daten.

PersistenceComponent Die Persistence-Komponente regelt das korrekte und dauerhafte Speichern in die einzelnen DataStores.

2.7.2 Deployment

Wie in der Abbildung 6 zu sehen ist, besteht unser System aus zwei physikalischen Geräten. Das ist zum einen der Client und zum anderen der BackendNode. Diese beinhalten jeweils sogenannte *DeploymentUnits* (DU).

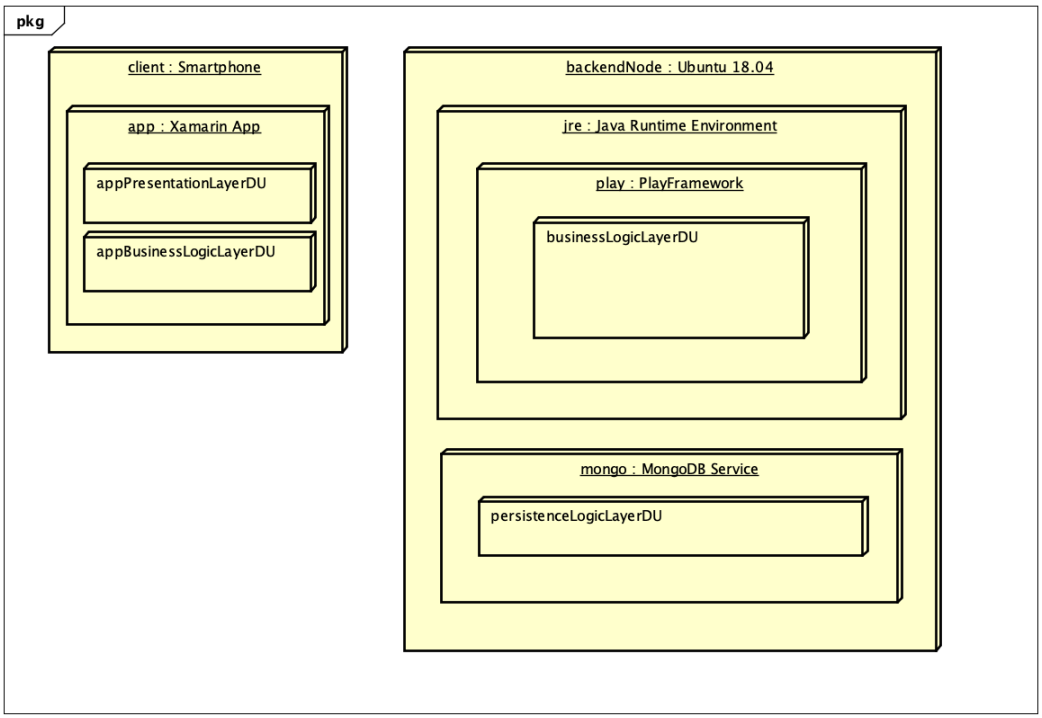

Abbildung 6: Deploymentdiagramm BrainingOutOfBox

Beim Client handelt es sich um das Smartphone des jeweiligen Benutzers. Auf seinem Smartphone läuft die Xamarin App, welche wiederum die appPresentationLayerDU und die appBusinessLogicLayerDU hält.

Der BackendNode ist ein Ubuntu 18.04 auf dem ein Java Runtime Environment (JRE) installiert ist. Innerhalb der JRE läuft das PlayFramework, in dem wiederum die businessLogicLayerDU läuft.

Zudem ist auf dem BackendNode ein MongoDB Service installiert, welche die persistenceLogicLayerDU beinhaltet.

Komponenten

Nachfolgend sind nochmals alle DeploymentUnits aufgelistet und kurz beschrieben.

businessLogicLayerDU Die businessLogicLayerDU enthält alle Komponenten, welche in Abbildung 5 in der Businesslogic-Schicht im PlayFramework eingezeichnet sind.

persistenceLogicLayerDU Die persistenceLogicLayerDU beinhaltet alle Komponenten der Persistence-Schicht, welche in Abbildung 5 zu sehen ist.

appPresentationLayerDU Die appPresentationLayerDU enthält alle Komponenten, welche in Abbildung 5 in der Präsentationsschicht liegen.

appBusinessLogicLayerDU Die appBusinessLogicLayerDU enthält alle Komponenten, welche in Abbildung 5 in der Businesslogic-Schicht in der Xamarin App gezeichnet sind.

2.8 Architekturentscheide

Wesentliche Entscheide, welche wir während dem Projekt getroffen haben, sind hier detailliert begründet. Auch Gedanken oder Ideen, welche wir während der Analysephase hatten, dann aber verworfen haben, sind hier aufgeschrieben.

2.8.1 Erste Erfahrungen mit der Methode 635

Wie in Kapitel 2.4.1 beschrieben, tendiert der Mensch dazu, die Ideen der anderen Teilnehmer automatisch zu bewerten. Als mögliche Lösung wurde die Integration einer Bewertungsmöglichkeit beschrieben.

Wir haben uns allerdings darauf geeinigt, dass wir die originale Version, also ohne die Möglichkeit für eine Wertung, als Vorlage nehmen und diese auch so in unserer Cross-Plattform Applikation umsetzen.

Die Integration einer Bewertungsmöglichkeit wird als optionales Feature angesehen und lediglich bei genügend Restzeit im Projekt umgesetzt.

2.8.2 Xamarin.Forms oder Xamarin native

Für diesen Entscheid galt es zu evaluieren, welche User Controls für unsere Applikation die exotischsten sind. Dies, weil Xamarin.Forms eine Menge an Standard-Controls anbietet, die vom Framework selber in das jeweilige Betriebssystem konvertiert werden. Sind alle vorgesehenen Benutzerelemente in Forms enthalten, sparen wir uns die Zeit, betriebssystemspezifische Elemente zu entwickeln.

Für unser Projekt haben wir folgende Benutzerelemente als exotisch oder kritisch definiert:

- Canvas Control für Zeichnen einer Idee
- Camera Funktion für das Erkennen von Quick Response-Codes (QR-Codes)
- Verarbeitung und Generierung von QR-Codes

Nach einer Recherche stellte sich heraus, dass sich ein Canvas View von Google namens SkiaSharp [12] eignet. Darauf lässt sich gemäss der Dokumentation zeichnen sowie definierte Formen einfügen. Dies könnte auch für eine Erweiterung spannend sein, in der Patterns in UML als Vorlage angeboten werden können.

Für die Kamera-Funktionalität steht ein NuGet-Packet (Xam.Media.Plugin [13]) bereit, das uns diese Arbeit abnehmen wird.

Das Generieren und Lesen der QR-Codes ist an sich kein Problem von Xamarin.Forms, denn grundsätzlich müssen die von der Kamera generierten Files eingelesen und ins entsprechende QR-Code-Tool eingefügt werden. Hierfür eignet sich das NuGet ZXing.Net [14].

Es stellte sich relativ rasch heraus, dass die gewünschten Funktionalitäten in Xamarin.Forms in ausreichender Qualität enthalten sind und uns das individuelle Entwickeln dadurch abgenommen wird.

2.8.3 Backend-Technologie

Neben den Vorteilen, wie der schlanken und zustandslosen Architektur des Play-Frameworks, dem asynchronen und nicht-blockierenden Verhalten und den vielen unterstützten Bibliotheken, haben wir uns hauptsächlich dafür entschieden, weil wir in anderen Projekten schon sehr gute Erfahrungen mit dem PlayFramework gemacht haben.

Ein weiterer Grund bestand darin, dass das PlayFramework nicht nur in Scala sondern auch in Java geschrieben ist. Mit Java kennen wir uns beide gut aus und mussten uns so keine neue Programmiersprache aneignen.

Da wir uns schon relativ früh für eine MongoDB als Datenbanksystem entschieden hatten, viel die Wahl für das PlayFramework erst recht, als wir einen asynchronen MongoDB-Treiber für Java gefunden hatten.

2.8.4 MongoDB als Datenbanksystem

MongoDB (abgeleitet von humongous) ist eine dokumentorientierte, einfache, dynamische und skalierbare NoSQL Datenbank , welche von der MongoDB Inc. entwickelt wird [15] [16] [17].

Die Basis für das Speichern von Informationen bilden die sogenannten Documents. Die Datenobjekte werden in separaten Documents innerhalb einer Collection (anders als bei traditionellen relationalen Datenbanken in Spalten und Zeilen) gespeichert [16]. Mit den Documents können zudem hierarchische Strukturen und Relationen sehr einfach gespeichert werden.

Der Vorteil einer MongoDB Datenbank liegt darin, dass zusammengehörige Informationen gemeinsam in einem Document gespeichert werden. Dies ermöglicht einen schnellen Zugriff auf die Daten mittels der MongoDB Query Language. Da MongoDB zudem ohne Schemas auskommt, ist es nicht nötig, die Datenbank offline zu nehmen, wenn man ein neues Feld einfügen möchte [17].

Weitere Vorteile sind laut DZone.com die hohe Performance, Verfügbarkeit (durch Replikas) und Skalierung (durch Sharding). Da alle Informationen zusammen in einem Document gespeichert sind, sind auch keine Joins zu anderen Tabellen notwendig. Auch unterstützt MongoDB Funktionen für die Speicherung von Geoinformationen [16].

Der Hauptgrund warum wir uns für MongoDB als Datenbanksystem entschieden haben, war die Tatsache, dass alle zusammengehörigen Informationen in einem Document abgelegt werden können. Auch die Möglichkeit hierarchische Strukturen (bei uns die *Brainsheets*, *Brainwaves* und *Ideas*) einfach zu Speichern, hat uns viel Zeit erspart.

Die Mächtigkeit von relationalen Datenbanken im Bereich einer Auswertung, ist in unserem Projekt nicht notwendig. Daher haben wir auch von Beginn an auf das Paradigma der dokumentorientieren Datenbanken gesetzt. Grund warum wir uns schlussendlich für MongoDB und nicht für eine andere dokumentorientierte

Datenbank entschieden haben ist, dass MongoDB Thema während dem Studium ist und wir schon einige Erfahrung damit hatten.

2.8.5 Methode 635 als Peer-to-Peer-System

Wir haben uns auch überlegt, die Cross-Plattform Applikation d.h. vor allem die Kommunikation zwischen den einzelnen Teilnehmer, als Peer-to-Peer System [18] zu konzipieren.

Prof. Thomas Bocek, Professor für verteilte Systeme an der Hochschule Rapperswil, hat uns allerdings davon abgeraten. Ein verteiltes System sei immer komplexer und komplizierter als ein Server/Client System. Für diese geringe Anzahl von Teilnehmern, welche prinzipiell nur Messages austauschen, lohnt es sich nicht ein verteiltes System zu bauen.

Daher haben wir uns für eine klassische Server/Client Architektur entschieden. Andere Varianten der Kommunikation, wie Webhooks oder Websockets werden daher nicht weiter verfolgt.

2.8.6 Kommunikation zwischen Server und App

Die Kommunikation zwischen dem zentralen Server und den Clients, also den Cross-Plattform Applikationen in unserem Fall, ist von grosser Bedeutung. Wegen der Problematik der Network Address Translation kurz NAT [19] kann diese prinzipiell auf zwei Arten erfolgen: Entweder man verwendet Websockets [20], welche eine permanente Verbindung zwischen Server und Client öffnen oder die Kommunikation beginnt ausschliesslich beim Client.

Wir haben uns für das stetige Abfragen von Informationen (Polling) entschieden, da es eine einfache Variante darstellt. Zwar werden dadurch vermeidbare Requests an den Server gesendet, da aber die Anzahl an Teilnehmer bzw. die Ressourcennutzung des Backends in einem vertretbaren Rahmen liegt, ist diese Polling-Variante völlig in Ordnung.

Zwei Beispiele für Polling in der Applikation: Wenn ein Teilnehmer seine Ideen aufschreibt, fragt die Applikation den Server im Hintergrund in regelmässigen Abständen nach der verbleibenden Zeit für diese Runde ab.

Auch nach der Abgabe der aufgeschriebenen Ideen an den Server, muss der Teilnehmer warten, bis er die Ideen bzw. das Blatt seines Nachbarn bekommt. Dafür muss die Applikation immer wieder den Server fragen, ob der Nachbar überhaupt sein Blatt bzw. seine Ideen abgegeben hat. Bevor dies nicht geschehen ist, kann der Teilnehmer auch nicht weitermachen.

Begründung in Zahlen: Bei diesen beiden Szenarien sendet die Applikation pro Sekunde einen Request an den Server. Wenn wir bei der Standardmethode von 6 Teilnehmern bleiben, wären das 6 Request pro Sekunde. Wenn wir nun noch annehmen, dass 50 gleichzeitige Ausführungen stattfinden, ergibt das 300 Requests pro Sekunde für eines der beiden Szenarien.

Laut den Release-Informationen zur Version 2.5 von Play [21] ist das PlayFramework in der Lage 60'000 Requests pro Sekunde zu verarbeiten.

Nehmen wir nun diese 60'000 Requests als Basis für unsere Berechnung, entsprechen unsere 300 Requests gerade einmal 0.5% der möglichen 60'000 Requests. Für beide Szenarien entsprechend das Doppelte, also 1%.

$$\frac{300 Requests}{1s} = \frac{1 Request}{1s} \cdot 6 Participants \cdot 50 Brainstormings$$

$$0.5\% = \frac{\frac{300 Requests}{1s} \cdot 100}{\frac{60000 Requests}{1s}} = \frac{300 \cdot 100}{60000}$$

Fazit: Diese kleine Rechnung verdeutlich schon ziemlich stark, dass die Variante mit dem Polling das PlayFramework in keinster Weise an dessen Limit bringt. Selbst wenn die Anzahl an Teilnehmern oder gleichzeitigen Ausführungen erhöht wird, ist das PlayFramework im Stande die eingehenden Requests noch zu verarbeiten. Auch andere gleichzeitige Aufrufe an das PlayFramework können dann noch ausgeführt werden.

Sollte es dennoch jemals zu einer zu starken Ressourcennutzung durch das Polling kommen, könnte der Algorithmus so angepasst werden, dass dieser nicht jede Sekunde das Backend z.B. nach der verbleibenden Zeit abfragt sondern bei viel verbleibender Zeit nur alle 30 Sekunden und bei wenig verbleibender Zeit jede Sekunde.

Auf diese Weise kann die Anzahl an Requests drastisch reduziert werden.

2.9 Herausforderungen

Hier sind besonders erwähnenswerte Herausforderungen und Hürden beschrieben, die im Verlaufe des Projektes aufkamen. Dies soll anderen Software Ingenieuren oder Interessierten helfen, aus unseren Schwierigkeiten zu lernen.

2.9.1 HTTPS REST Schnittstelle in CI/CD aufsetzen

Während der Entwicklung des Prototyps in der Evaluation stellten wir fest, dass das Abfragen unserer Backend-Schnittstelle auf Android wie gewünscht funktionierte. Auf iOS jedoch warf die Applikation beim Ausführen eine Exception. Der Grund dafür war, dass iOS seit Version 9 [22] keine Verbindungen zu unverschlüsselten Webseiten mehr zulässt. Eine verschlüsselte Website war daher unumgänglich.

Nach kurzen Recherchen haben wir für das Erstellen und Validieren des Zertifikates auf Let's Encrypt [23] gewählt, gerade deshalb weil eine ausführliche Dokumentation und eine rasche Generierung möglich ist.

Auch mit dem verwendeten PlayFramework sollte das Aufsetzen des HTTPS Endpunktes kein Problem darstellen, mit einem Parameter beim Start der Applikation verschlüsselt Play die Seite automatisch.

Nachdem das Zertifikat installiert wurde und die Applikation lokal erfolgreich verschlüsselt lief, galt es, das Gesamte noch in den Build-Prozess einzubauen. Dabei kam das Problem auf, dass der Pfad zum Keystore, der das Let's Encrypt-Zertifikat beinhaltet, nicht gültig war. Nach mehrmaligem, gründlichem Überprüfen des Pfades und neu Builden, wurden immer noch Exceptions geworfen.

Wir haben keinen Anhaltspunkt, was der Fehler sein könnte, denn werden die genau gleichen Befehle auf dem Backend-Server direkt ausgeführt, funktioniert die Applikation einwandfrei auf HTTPS. Sobald es aber über den Build-Server läuft, findet er den Pfad zum Keystore nicht mehr.

Als Work-Around haben wir eine `CustomSslEngineProvider` geschrieben, der den Pfad zum Keystore direkt im Code beinhaltet. Darauf hat Puppet keinen Einfluss und die Applikation funktioniert wie gewünscht.

2.9.2 Komplexes Design der Brainstorming Logik im Frontend

Im Verlaufe der Entwicklungsphasen hat sich insbesondere die Klasse `BrainstormingPageViewModel` stetig vergrössert und wurde komplexer. Die gesamte Logik für den Blattwechsel, das Holen der verbleibenden Zeit, das Ein- und Ausblenden einiger Controls sowie das Überprüfen, ob eine Runde abgeschlossen wurde, ist in dieser Klasse implementiert.

Wie wir bald feststellten, macht eine solche Klasse die Fehlersuche sehr schwierig. Es wurden zwar einige kleinere Bugs behoben, aber es wurden dabei auch neue aufgedeckt beziehungsweise eingeschleust. Wenn zwei Timers aktiv sind und somit Parallelität im Spiel ist, ist das Verhalten der App nicht deterministisch. So haben

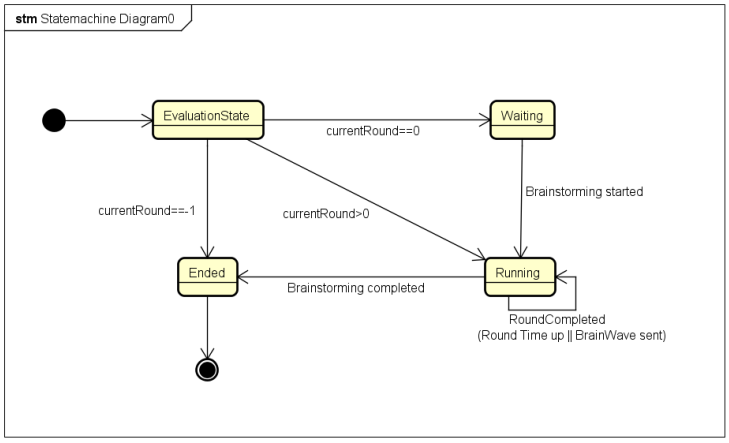

Abbildung 7: Zustandsmaschine für Brainstorminglogik

wir die Situation, dass einmal eine Runde einwandfrei funktioniert, und ein andermal mit derselben Konfiguration die Controls nicht richtig aktualisiert werden.

Um die Komplexität zu verringern, wurde ein Konzept erarbeitet, welches in einer allfälligen Folgearbeit umgesetzt werden sollte. Abbildung 7 erklärt dieses Konzept anhand eines Zustandsdiagramms. Dabei geht es darum, eine State-Machine einzuführen, welche die drei Zustände **Waiting**, **Running** und **Ended** enthält. Bei jedem Aufruf der ViewModel-Klasse wird zu Beginn überprüft, in welchem Zustand das aktuelle Finding ist (**Evaluation State**) und die Maschine wird dann in den entsprechenden Zustand versetzt. Darauf sind alle für die Controls benötigten Properties und Timers implementiert. Zustandswechsel treten bei Rundenwechsel auf, zum Beispiel wechselt beim Start des Brainstormings die Rundenzahl von null auf eins, was die Zustandsmaschine vom Waiting- in den Running-Zustand versetzt. Analog funktioniert sie in der letzten Runde, wobei die Rundenzahl in dieser Situation auf -1 gesetzt wird.

Der grosse Vorteil bei dieser Lösung ist, dass der Überblick zurückgewonnen werden kann und somit eine sauberere Fehlerbehandlung möglich ist. Auch Erweiterungen sind somit einfacher möglich (zum Beispiel Hinzufügen eines "Review-States).

Workaround

Als aktuelle Lösung bieten wir einen Sync-Button an, der beim Betätigen die Brainstorming Page aktualisiert. Dies hat zur Folge, dass alle Daten vom Backend geholt und wieder gesetzt werden. Wenn zum Beispiel ein Rundenwechsel nicht funktioniert hat, eignet sich dieses Feature, um den korrekten Zustand wiederherzustellen und mit dem Backend zu synchronisieren. Der Button ist im Icon hinterlegt, wie in Abbildung 8 sichtbar.

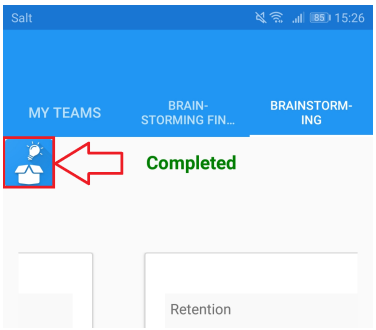

Abbildung 8: Sync Button

2.10 Ergebnisse

Das Kapitel der Ergebnisse befasst sich mit der konkreten Umsetzung der Xamarin Applikation und dem PlayFramework. Wir haben uns dabei ganz konkret für die nachfolgenden Code-Beispiele entschieden. Der Grund dafür ist, dass diese entweder den typischen Aufbau einer Methode zeigen oder wie das Listing 4, eine Kernlogik darstellen. Bei den Ergebnissen der Xamarin App zeigen wir zusätzlich noch eine kurze Retrospektive auf geplantes Design und effektiver Umsetzung des GUIs. Das Kapitel wird durch eine Gegenüberstellung der erreichten und geplanten Arbeit abgeschlossen.

Zur besseren Übersicht wurde der Code vereinzelt gekürzt. Dies ist durch 3 Punkte (...) gekennzeichnet.

2.10.1 Implementierung des PlayFrameworks

Die Umsetzung des Backends basiert auf dem PlayFramework. Die Gründe für diesen Entscheid können in Kapitel 2.4 nachgelesen werden.

Um die Daten permanent zu speichern, besitzt das PlayFramework eine Anbindung an eine MongoDB Instanz. Die Anbindung wurde mit dem MongoDB Async Driver für Java [24] realisiert. Die Implementation ist daher auch in Java geschrieben. Für die Dokumentation des Backends verwendeten wir Swagger [25].

ParticipantController.java

Die nachfolgenden Listings zeigen einen Ausschnitt aus der `ParticipantController.java` Klasse. Diese befindet sich in der LogicComponent (Abbildung 5) vom PlayFramework.

```
1 public Result createParticipant(){
2
3     JsonNode body = request().body().asJson();
```

```
4
5       if (body == null ) {
6           return forbidden(Json.toJson(new ErrorMessage
                ("Error", "json body is null")));
7       } else if(  body.hasNonNull("username") &&
8                   body.hasNonNull("password") &&
9                   body.hasNonNull("firstname") &&
10                  body.hasNonNull("lastname")) {
11
12      Participant participant = new Participant(body.
            get("username").asText(), body.get("password")
            .asText(), body.get("firstname").asText(),
            body.get("lastname").asText());
13
14      participantCollection.insertOne(participant, new
            SingleResultCallback<Void>() {
15          @Override
16          public void onResult(Void result, Throwable t
                ) {
17              Logger.info("Inserted Participant!");
18          }
19      });
20
21      return ok(Json.toJson(new SuccessMessage("Success
            ", "Participant successfully inserted")));
22      }
23
24      return forbidden(Json.toJson(new ErrorMessage("
            Error", "json body not as expected")));
25  }
```

Listing 1: Participant erstellen

Bei der Methode für das Erstellen von Participants, prüft das Framework zuerst, ob ein HTML-Body existiert. Ist dies nicht der Fall, sendet es ein HTTP-Response Status-Code (Zeile 24) zurück.

Existiert ein HTML-Body, wird dieser auf die Existenz der Felder *username*, *password*, *firstname* und *lastname* geprüft (Zeile 7-10). Daraus erstellt das Framework als nächstes ein `participant` (Zeile 12), welcher mittels `insertOne` in die `participantCollection` gespeichert wird (Zeile 14).

Zuletzt sendet das PlayFramework eine Antwort mit dem HTTP-Status Code 200 und einer Nachricht an den Absender.

```
1  public Result login() throws
       UnsupportedEncodingException, ExecutionException,
       InterruptedException {
2  ...
3  if (body.hasNonNull("username") && body.hasNonNull("
       password")) {
4      CompletableFuture<Participant> future = new
           CompletableFuture<>();
5
6      participantCollection.find(and(
7      eq("username", body.get("username").asText()),
8      eq("password", body.get("password").asText()))).
           first(new SingleResultCallback<Participant>()
           {
9          @Override
10         public void onResult(Participant participant,
               Throwable t) {
11             if (participant != null) {
12                 Logger.info("Found participant");
13                 future.complete(participant);
14             } else {
15                 future.complete(null);
16             }
17         }
18     });
19
20     if (future.get()!= null){
21         ObjectNode result = Json.newObject();
22         result.putPOJO("participant", future.get());
23         result.put("access_token", getSignedToken(71)
               );
24         return ok(result);
25     } else {...}
26 } else {...}
27 }
```

Listing 2: Login

Für das Login eines Participants schaut auch hier zuerst das Framework im HTML-Body nach der Existenz der Felder *username* und *password* (Zeile 3). Im nächsten Schritt durchsucht es die Datenbank nach einem Participant mit den angegebenen Werten (Zeile 6-8).

Da wir einen asynchronen Treiber verwenden, benötigen wir ein `CompletableFuture`, um das Resultat der Abfrage darin abzuspeichern und um mittels `future.get()` darauf zugreifen zu können (Zeile 20).

Am Ende wird dem Resultat neben dem gefundenen *participant* noch ein *JWT-Token* angefügt (Zeile 21-24).

TeamController.java

Das Listing 3 zeigt einen Ausschnitt aus der `TeamController.java` Klasse. Auch diese befindet sich in der LogicComponent (Abbildung 5) vom PlayFramework.

```
1  public Result joinBrainstormingTeam(String
       teamIdentifier) throws ExecutionException,
       InterruptedException {
2  ...
3  if (brainstormingTeam!= null && brainstormingTeam.
       getNrOfParticipants() > brainstormingTeam.
       getCurrentNrOfParticipants() && brainstormingTeam.
       joinTeam(participant)) {
4
5      teamCollection.updateOne(eq("identifier",
           teamIdentifier),combine(set("participants",
           brainstormingTeam.getParticipants()), inc("
           currentNrOfParticipants", 1)), new
           SingleResultCallback<UpdateResult>() {
6          @Override
7          public void onResult(final UpdateResult
               result, final Throwable t) {
8              Logger.info(result.getModifiedCount() + "
                   Team successfully updated");
9          }
10     });
11     return ok(Json.toJson(new SuccessMessage("Success
           ", "Participant successfully added to the
           brainstormingTeam")));
12
13 } else {
14     ...
15 }
16 ...
17 }
```

Listing 3: Einem Team beitreten

Dieses Beispiel soll zeigen, wie mit dem MongoDB Async Driver ein Eintrag mittels `updateOne` aktualisiert werden kann.

Wie auch schon bei der `find` Methode, kennzeichnet der erste Parameter das Dokument, welches man aktualisieren möchte. Der zweite Parameter steht für die Felder und deren neuen Werte und der dritte und letzte Parameter ist wieder der `SingleResultCallback` und beschreibt wie das Resultat weiter prozessiert wird. All dies ist auf Zeile 5-8 zu finden.

FindingController.java

Das Listing 4 zeigt einen Ausschnitt aus der `FindingController.java` Klasse. Wie auch schon die vorherigen Klassen, befindet sich auch diese in der LogicComponent (Abbildung 5) vom PlayFramework.

```
1  private void startWatcherForBrainstormingFinding(
       String identifier){
2
3  ScheduledExecutorService executor = Executors.
       newSingleThreadScheduledExecutor();
4
5  TimerTask task = new TimerTask() {
6      @Override
7      public void run() {
8          try {
9        ...
10   if (finding.getCurrentRound() > finding.
       getBrainsheets().size()){
11       lastRound(identifier);
12       executor.shutdown();
13   }
14
15   if (endDateTime.plusSeconds(30).isBeforeNow() ||
16   finding.getDeliveredBrainsheetsInCurrentRound() >=
       finding.getBrainsheets().size()){
17   nextRound(identifier);
18   }
19               cancel();
20
21           } catch (ExecutionException e) {
22               e.printStackTrace();
23           } catch (InterruptedException e) {
24               e.printStackTrace();
25           }
```

```
26      }
27  };
28
29  executor.scheduleAtFixedRate(task, 1000L, 5000L,
        TimeUnit.MILLISECONDS);
30  }
```

Listing 4: Watcher für BrainstormingFinding

Um den Zustand über die verbleibende Zeit oder die bereits eingereichten *Brainsheets* überwachen zu können, setzen wir einen `ScheduledExecutorService` [26] ein. Dieser erlaubt uns alle 5000ms bzw. alle 5s den `TimerTask` auf Zeile 5 auszuführen.

Der `TimerTask` prüft zuerst, ob die aktuelle Runde schon die letzte Runde ist (Zeile 10). Ist dies der Fall, führt er die Methode *lastRound(identifier)* aus und beendet den executor, sodass keine neuen `TimerTask` Objekte gestartet werden. In diesem Zustand ist das gesamte *BrainstormingFinding* ausgefüllt.

Ist dies nicht der Fall, prüft er als nächstes, ob die Endzeit der aktuellen Runde plus 30s noch vor der aktuellen Uhrzeit liegt (Zeile 15). Die Bedingung auf Zeile 16 prüft, ob alle *Brainsheets*, welche für die Abgabe erwartet werden schon abgegeben wurden. Unabhängig welche dieser zwei Bedingungen (Zeile 15 oder 16) zuerst eintrifft, es wird anschliessend immer die Methode *nextRound(identifier)* ausgeführt.

Sollte keine der Bedingungen (Zeile 10, 15 oder 16) zutreffen, so beendet sich der `TimerTask` mittels *cancel* selbst.

Da der `executor` aber nach 5s den nächsten `TimerTask` startet, ist so für die gesamte Dauer, für die das *BrainstormingFinding* läuft, ein "Watcher" für den korrekten Ablauf zuständig. Die Methode *startWatcherForBrainstormingFinding(String identifier)* wird beim Start eines *BrainstormingFinding* ausgeführt.

```
1  public Result startBrainstorming(String
       findingIdentifer) throws ExecutionException,
       InterruptedException {
2          startWatcherForBrainstormingFinding(
               findingIdentifer);
3          return nextRound(findingIdentifer);
4  }
```

2.10.2 Verwendete Bibliotheken im Backend

Tabelle 4 zeigt die Bibliotheken auf, die im Backend verwendet werden.

Bibliothek	Version	Repository	Lizenz
swagger-play2	1.6.0	mvnrepository.com	Apache 2.0
java-jwt	3.2.0	mvnrepository.com	MIT
mongodb-driver-async	3.8.0	mvnrepository.com	MIT

Tabelle 4: Verwendete Bibliotheken Backend

2.10.3 Implementierung der Xamarin App

In vielen Microsoft UI Technologien ist das MVVM-Pattern Standard. Dabei geht es um ein Design-Pattern, welches die Objekte in **M**odel, **V**iew, **V**iew**M**odel unterteilt. Man kann so die View sauber von den darunterliegenden Schichten trennen. Auch in diesem Projekt steht dieses Pattern im Einsatz.

Um ein sauberes Design des Frontends zu ermöglichen, ist das MVVM-Framework Prism.Forms [27] im Einsatz. Dies vereinfacht die Verwendung des MVVM-Patterns, das auch bei Xamarin üblich ist. Die ViewModels werden mit diesem Framework entweder per Konvention oder per Konfiguration miteinander verknüpft, sodass der `BindingContext` einer View auf das entsprechende ViewModel gesetzt ist. Dieser wird benötigt, um die berechneten Werte der Variablen im User Interface darzustellen. In Listing 5 verknüpfen wir das MainPageViewModel mit der MainPageView, sodass wir darauf navigieren können.

```
1 containerRegistry.RegisterForNavigation<MainPage,
    MainPageViewModel>();
```

Listing 5: Verknüpfung von View mit ViewModel in Prism.Forms

Das Framework bietet weiter *Dependency Injection*[5] an. Zum Beispiel kann so ein Navigation Service jeder Komponente im Konstruktor mitgegeben werden, dass diesem das Navigieren ermöglicht wird.

Als Ausgangslage für die Umsetzung dienten die Mockups. Aufgrund diesen recherchierten wir in der Dokumentation von Xamarin.Forms nach möglichen User Controls und Layouts, die eingesetzt werden sollten. Dabei stellte sich heraus, dass das erarbeitete Navigationskonzept in den Mockups sich nicht implementieren liess. Problem dabei war, dass das Mockup mit den von Xamarin.Forms dargebotenen Layouts nur sehr umständlich programmiert werden konnte. Dabei hätten wir eine `MasterDetailPage` mit Swipe Gestures erweitern müssen, welche aber schon vom Framework für das Auf- und Zuklappen der Seitennavigation verwendet werden. Deshalb haben wir uns für eine `TabbedPage` entschieden, die die relevanten Daten ebenso nützlich darstellt. In Abbildung 9 sind die Unterschiede zwischen der App und dem Mockup ersichtlich.

[5] Siehe Application Architecture Vorlesung Woche 5

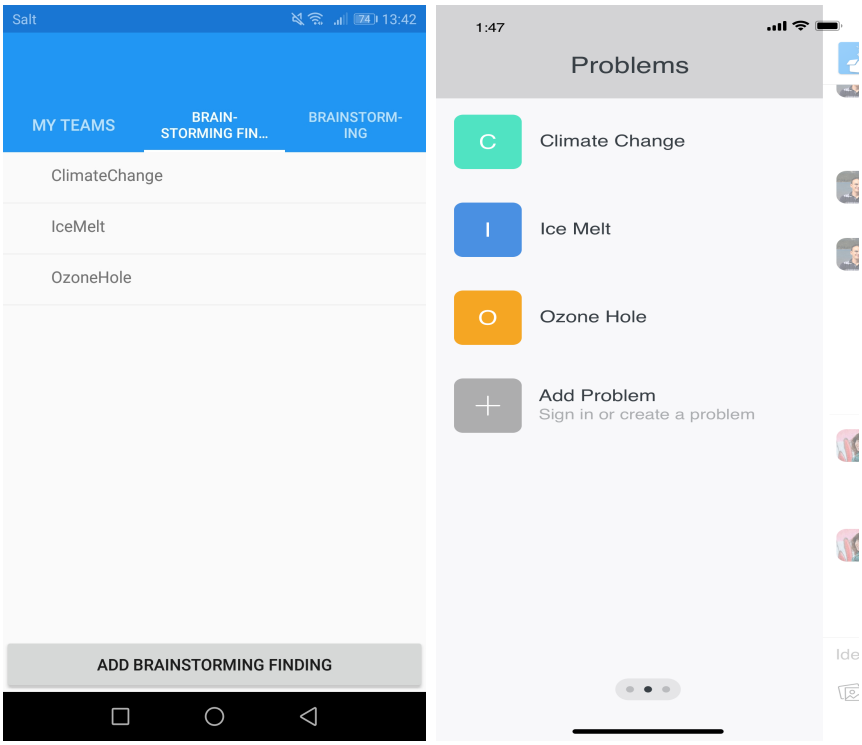

Abbildung 9: Vergleich App und Mockup

Umsetzung Brainstorming

Die Brainstorming Page besteht aus drei visuellen Bereichen, Header, Footer und Center. Je nach Zustand des BrainstormingFindings werden Elemente davon angezeigt oder nicht. Es gibt 3 Zustände, indem ein Brainstorming sein kann:

1. **Waiting**. Das Brainstorming ist erstellt und es wird darauf gewartet, dass der Moderator den Prozess startet.
2. **Running**. Das Brainstorming läuft und die Runden werden herauf- und die verbleibende Zeit pro Runde heruntergezählt.
3. **Ended**. Das Brainstorming ist beendet.

Tabelle 5 zeigt die Sichtbarkeit einiger relevanten Controls im Brainstorming.

Controls	Waiting	Running	Ended
Header			
Send Button	✓(disabled)	✓	x
RemainingTime	✓(00m:00s)	✓	x

'Completed'	x	x	✓
Center			
BrainSheets (Darstellung der Blätter)	x	✓	✓
Footer			
Commit Button	✓(disabled)	✓	x
Text Entry	✓	✓	x

Tabelle 5: Sichtbarkeit nach Zustände im Brainstorming

Einige der erwähnten Komponenten können der Abbildung 10 entnommen werden.

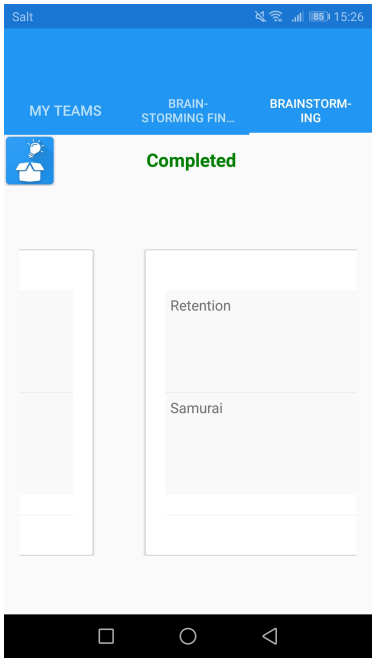

Abbildung 10: Brainstorming Finding Übersicht während des Swipens von einem ins nächste BrainSheet

Für das Abfragen der verbleibenden Zeit in der aktuellen Runde ist ein `System.Timers.Timer` implementiert, der im Sekundentakt das Backend abfragt. Sinkt diese Zahl unter eine Sekunde, werden die vorhandenen Ideen der aktuellen BrainWave ans Backend gesendet. Gleichzeitig startet ein zusätzlicher Timer, der das BrainstormingFinding im Takt von 2.5s abfragt und prüft, ob sich die Runde geändert hat. Ist dies der Fall, müssen die betroffenen UI-Elemente aktualisiert werden. Die Logik, die vom zusätzlichen Timer aufgerufen wird, ist in der Methode *UpdateRound()* im Listing 6 zu finden.

```
1  private void UpdateRound()
2  {
3   var backendFinding =
       _brainstormingFindingRestResolver.GetFinding(
       _context.CurrentFinding);
4   if (backendFinding.CurrentRound != _context.
       CurrentFinding.CurrentRound)
5   {
6    _context.CurrentFinding = backendFinding;
7    Console.WriteLine("Round has changed, proceeding to
         next round");
8    NextRound();
9   }
10 }
```

Listing 6: Poll-Mechanismus um zu prüfen ob Runde gewechselt hat

Create/Join Group mit QR-Code

Wie in UC 10: Create Brainstorming Team und UC 5: Join Brainstorming Team spezifiziert, muss unser System einen Mechanismus für das Erstellen von Teams und beitreten in diese Teams bieten.

Wir kreierten eine Lösung, die beim Erstellen des Teams einen QR-Code generiert, welcher von anderen Teammitgliedern mittels QR-Scanner einlesen werden kann. Durch den Scan-Vorgang treten die Mitglieder automatisch der erstellten Gruppe bei. Dies bedingt, dass alle Mitglieder einmal für das Erstellen des Teams beieinander sein müssen.

Um diesen Mechanismus umzusetzen, wurde die ZXing [14] Library verwendet, die wir im Rahmen der Recherche (Kapitel 2.8.2) entdeckten. Diese erlaubt es, ein Xamarin Control für das Scannen und das Generieren zu verwenden. Mit etwas plattform-spezifischem Konfigurationsaufwand funktioniert diese Library wie gewünscht.

Listing 7 demonstriert, wie das Generieren und Darstellen des QR-Codes in XAML funktioniert.

```
1 <forms:ZXingBarcodeImageView BarcodeValue="{Binding
    Text}"  BarcodeFormat="QR_CODE">
```

Listing 7: Verwendung der ZXing Library fürs Generieren des QR-Codes

Listing 8 zeigt das Gegenstück zum oben genannten Generieren, nämlich den XAML-Code, der für das Scannen verwendet wird. Der interessante Teil dabei ist das Property `ScanResultCommand`, was es mittels MVVM erlaubt, über einen Command eine parametrisierte Methode aufzurufen. Auf dem Parameter ist das Scan-Resultat als Text gespeichert. In unserem Fall entspricht dieser der Team-Id, aufgrund welcher der QR-Code generiert wurde. Die Team-Id wird danach verwendet, um mittels RESTful HTTP auf dem Backend den JoinTeam Endpoint aufzurufen, der wiederum den entsprechenden Eintrag in der Datenbank durchführt und somit den User ins Team einfügt.

```
1 <zxing:ZXingScannerView IsScanning="true" Options="{
    Binding BarcodeOptions}" WidthRequest="200"
    HeightRequest="200" ScanResultCommand="{Binding
    FoundTeamIdCommand}" />
2 <zxing:ZXingDefaultOverlay TopText="Place the code
    inside the area" Opacity="0.9" BottomText="{
    Binding BottomOverlayText}" />
```

Listing 8: XAML fürs Scannen von QR-Codes

Die Abbildung 11 zeigt die beiden Controls zur Generierung und Scannen von QR-Codes auf einem Android Device in Aktion.

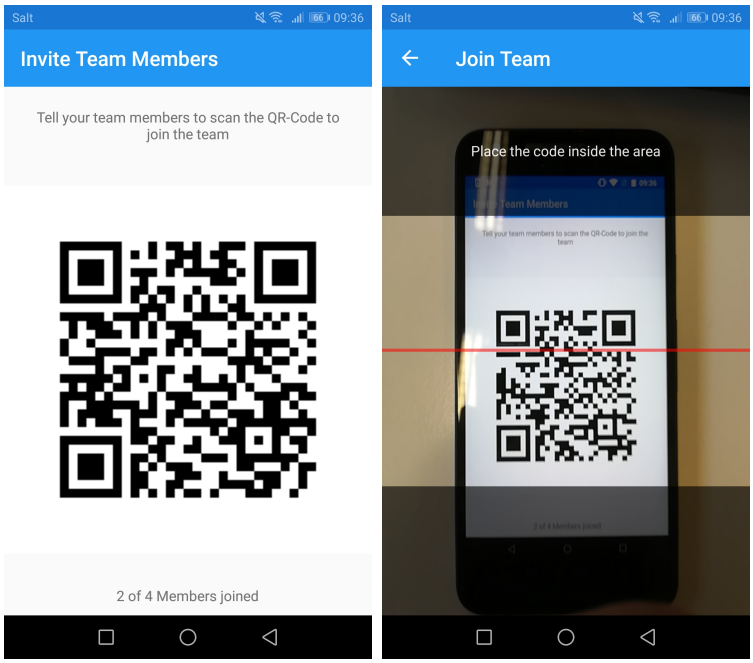

Abbildung 11: Generierter QR-Code links und QR-Code Scan rechts

2.10.4 Verwendete Bibliotheken im Frontend

In Tabelle 6 sind die verwendeten Libraries und Frameworks des Frontends aufgelistet.

Bibliothek	Version	Repository	Lizenz
CarouselView.FormsPlugin	5.2.0	github.com	MIT
Microsoft.AppCenter	1.10.0	AppCenter	MIT
NUnit	3.11.0	nunit.org	MIT
NUnit3TestAdapter	3.11.2	github.com	MIT
Prism.Forms	7.0.0.396	github.com	MIT
Xamarin.Forms	3.3.0.912540	Microsoft Docs	MIT
XamlStyler.Console	3.0.0	github.com	Apache 2.0
ZXing.Net.Mobile	2.4.1	github.com	Apache 2.0
ZXing.Net.Mobile.Forms	2.4.1	github.com	Apache 2.0

Tabelle 6: Direkt verwendete Bibliotheken Frontend

2.10.5 Vergleich Soll/Ist

Die im Anhang A formulierten Ziele und Liefergegenstände wurden wie folgt erreicht. Die Konfigurierbarkeit ist durch die Applikation gegeben. Es ist daher möglich, Bestandteil von verschiedenen Teams zu sein und darin mehrere BrainstormingFindings mit unterschiedlichen Anzahl an Runden bzw. Zeit pro Runde zu erstellen. Des Weiteren werden Smartphone Fähigkeiten wie Kamera verwendet, um dem Team beizutreten. Hier sehen wir vor allem in Kombination mit der Erweiterbarkeit noch Potenzial, die Kamera als Mittel fürs Brainstorming selbst einzusetzen (gemäss UC8b: Insert Picture). Dass dies nicht dazu kam liegt an der begrenzten Zeit von 14 Wochen. Aus gleichem Grund wurden die beiden anderen Sub-Use-Cases UC8a: Insert Weblink und UC8c: Insert Sketch nicht umgesetzt.

Das Persistieren von Ideen in unserem System bietet der Papierversion gegenüber einen Vorteil, da die einzige Information für das spätere Abholen der Ideen die Login-Daten sind. Mit diesen können alle jemals erfassten BrainstormingFindings wieder gelesen werden, während in der Papierversion die Unterlagen physisch hervorgeholt werden müssten, falls sie überhaupt archiviert wurden.

Ein knapp erfülltes Ziel ist die Robustheit und Bedienbarkeit. So ist es ohne Weiteres möglich, die App gemäss ihrem Sinne auszuführen und zu verwenden, jedoch ist die Benutzerführung nicht überall intuitiv. Dies ergab unser User Test am 06. Dezember 2018. Auch kann es vorkommen, dass ab und an ein Refresh benötigt wird. Zum Beispiel kann es sein, dass beim Starten des Brainstormings nicht bei allen Participants das entsprechende Sheet vorliegt, sondern die View leer bleibt. Dies hängt mit den in Kapitel 2.9.2 erläuterten Problemen zusammen und kann durch das Betätigen der Sync-Funktionlität auf dem Icon behoben werden.

Weitere nicht implementierte Features sind UC 2: Logout auf Back- und Frontend sowie UC4: Delete Account und UC6: Leave Brainstorming Team auf dem Frontend. Da diese keine kritischen Erfolgsfaktoren sind und zur Kernfunktionlität nicht wesentlich beitragen, wurden diese nach iterativer Planung und Abwägen der Ressourcen nicht implementiert.

Ansonsten konnten alle spezifizierten Use-Cases umgesetzt werden.

2.11 Schlussfolgerungen

Im Kapitel der Schlussfolgerungen wollen wir nochmals auf unser Projekt und dessen Verlauf schauen und unsere Ergebnisse kritisch bewerten. Dabei wollen wir aufzeigen, was wir in dieser Zeit erreicht haben, aber auch an welchen Stellen es noch Verbesserungspotenzial gibt. Ausserdem soll im Kapitel 2.11.3 aufgezeigt werden, was das weitere Vorgehen für dieses Projekt sein könnte.

2.11.1 Ergebnisbewertung

Kapitel 2.10.5 zeigt auf, dass viele Ziele erreicht werden konnten. Voraussetzung dafür waren einige Faktoren, unter anderem sehr gute Kommunikation untereinander, die richtig gewählten Management-Methoden und Techniken sowie das Verwenden von einem unterstützenden Toolset.

Besonders in der Elaboration Phase verhalf das Herunterbrechen der Methode 635 in verschiedene Komponenten dem Verständnis und gab Aufschluss über die Herangehensweise für die technische Umsetzung.

Diese verlief dann ziemlich effizient, wobei aber immer wieder kleinere Probleme auftauchten. Durch Absprache untereinander und mit dem Betreuer wurde aber immer eine Lösung gefunden. Am Schluss der Entwicklung gab es einige Schwierigkeiten, da der Code mit dem Implementieren weiterer Features immer umfangreicher wurde und somit auch unübersichtlicher.

Alle erwähnten Punkte kombiniert hatten zur Folge, dass eine lauffähige Applikation zustande kam, die für das Brainstorming verwendbar ist. Der durchgeführte Test am 06. Dezember ergab, dass die Benutzung der Applikation für den Endbenutzer nicht immer ganz klar und intuitiv ist. Dies schränkt den Benutzer aber nicht in der Kernfunktionalität, der kreativen Ideenfindung, ein. Dies konnte nur geschehen, weil wir während der Projektzeit ein starker Fokus auf die Funktionalität legten. Die Usability haben wir also gegenüber der Funktionalität abgewogen und darin verhältnismässig weniger Aufwand investiert.

Das Projekt ist in unseren Augen erfolgreich verlaufen. In einer zukünftigen Arbeit würden beide Autoren ein ähnliches, iteratives Vorgehen wählen, den Fokus aber auch stärker auf die Usability legen.

2.11.2 Bekannte Probleme

Dieses Kapitel dient dazu, Fehlverhalten in unserem System zu dokumentieren.

Absturz bei Join Team Beim erstmaligem Ausführen der Join Team Funktionalität fragt Android nach der Berechtigung für den Zugriff auf die Kamera. Sobald diese gewährleistet wird, stürzt die Applikation aus ungeklärten Gründen ab. Wurde die Berechtigung einmal vergeben, funktioniert das Beitreten des Teams wie gewünscht.

Neue Runde/Overview wird nicht richtig dargestellt Wenn das Brainstorming gestartet wird oder eine neue Runde beginnt, kommt es vor, dass die Sheets nicht wie gewünscht aktualisiert werden. Abhilfe verschafft die Sync-Funktionlität, die hinter dem Icon versteckt ist.

Kein Feedback beim Team erstellen Wird zuerst ein Team erstellt, darauf ein BrainstormingFinding erstellt und gestartet und im Anschluss darauf versucht, erneut ein Team zu erstellen, wird man nicht auf dessen Liste von BrainstormingFindings weitergeleitet. Das Team wird zwar auf dem Backend erstellt, jedoch auf dem Frontend nicht aktualisiert.

2.11.3 Ausblick

Da die Vision dieses Projektes vielversprechend ist, empfinden wir es als sehr lohnenswert diese Applikation zu erweitern. Dabei kommen vor allem weitere Features für das Erfassen von Ideen in Frage, wie das Verwenden der Kamera für Fotos sowie Einfügen von Weblinks und Skizzen.

Grundbaustein für diese Erweiterungen sind aber sauber entworfene Software Architekturen auf Front- und Backend. Dazu empfehlen wir stark, das Konzept gemäss unserer Analyse in Kapitel 2.9.2 umzusetzen, um eine ideale Plattform für die erwähnten Erweiterungen zu haben.

Des Weiteren sehen wir vor allem einer Erweiterung mit der Skizzier-Funktion viel Potenzial, da mit Skizzen die Grenzen der Kreativität loser sind als nur mit Text oder Fotos. Um dieses sinnvoll anbieten zu können, muss der Fokus auf die Usability erheblich steigen. Die Applikation sollte auf Tablets gut unterstützt werden, denn dort sehen wir die meisten Benutzer, welche die Skizzier-Funktion verwenden könnten.

Die Usability sollte nicht nur im Bezug auf die zusätzlichen Funktionen, sondern generell verbessert werden. Xamarin.Forms bietet viele interessante Konzepte für Animationen und Visualisierungen, welche die Benutzerführung stark verbessern könnten. Durch das Analysieren der Dokumentation von Xamarin.Forms könnten viele angebrachte Konzepte und Elemente entdeckt werden, die der Usability zugute kämen.

Als zusätzliche Erweiterung wäre eine webbasierte Applikation denkbar. Dazu müsste allerdings die Logik rund um das Team etwas angepasst werden, denn aktuell kann man nur über das Scannen eines QR-Codes einem Team beitreten.

Wie aus diesen Punkten ersichtlich ist, stecken in dieser Applikation viele spannende Erweiterungsmöglichkeiten. Aus diesen Gründen sehen wir ein grosses Potenzial in einer allfälligen Folgearbeit.

Literatur

[1] Kreativitaetstechniken, "635-methode." https://kreativitaetstechniken.info/6-3-5-methode/, Oktober 2011. Accessed on 2018-09-18.

[2] Wikipedia, "Xamarin." https://de.wikipedia.org/wiki/Xamarin, November 2018. Accessed on 2018-11-22.

[3] CCVossel, "Xamarin - was ist das eigentlich?." https://ccvossel.de/2016/07/xamarin/, November 2018. Accessed on 2018-11-22.

[4] S. Clark, "What is xamarin? what is its nedd?." https://www.quora.com/What-is-Xamarin-What-is-its-need, November 2018. Accessed on 2018-11-22.

[5] Lightbend, "Playframework." https://www.playframework.com/, August 2017. Accessed on 2018-10-09.

[6] Wikipedia, "Iso 9126." https://de.wikipedia.org/wiki/ISO/IEC9126, Mai 2018. Accessed on 2018-09-25.

[7] Johner-Institut, "Iso 25010 und iso 9126." https://www.johner-institut.de/blog/iec-62304-medizinische-software/iso-9126-und-iso-25010/, August 2015. Accessed on 2018-09-25.

[8] M. Kops, "Qualitaet, funktionale und nichtfunktionale anforderungen in der software-entwicklung." https://blog.seibert-media.net/blog/2018/05/14/qualitaet-funktionale-und-nichtfunktionale-anforderungen-in-der-software-entwicklung/, Mai 2018. Accessed on 2018-09-25.

[9] Wikipedia, "Smart criteria." https://en.wikipedia.org/wiki/SMART_criteria, August 2018. Accessed on 2018-09-27.

[10] SolidSourceIt, "Does source code duplication matter?." https://solidsourceit.wordpress.com/2012/08/03/does-source-code-duplication-matter/, August 2012. Accessed on 2018-10-06.

[11] Auth0, "Jwt." https://jwt.io, November 2018. Accessed on 2018-11-01.

[12] mono, "mono/skiasharp." https://github.com/mono/SkiaSharp, Oktober 2018. Accessed on 2018-10-25.

[13] jamesmontemagno, "Xam-media-plugin." https://www.nuget.org/packages/Xam.Plugin.Media/, Oktober 2018. Accessed on 2018-10-25.

[14] M. Jahn, "Zxing.net." https://www.nuget.org/packages/ZXing.Net/, October 2018. Accessed on 2018-10-25.

[15] Wikipedia, "Mongodb." https://de.wikipedia.org/wiki/MongoDB, November 2018. Accessed on 2018-11-23.

[16] DZone, "When to use (and not to use) mongodb." https://dzone.com/articles/why-mongodb, November 2018. Accessed on 2018-11-23.

[17] M. Inc., "What is mongodb?." https://www.mongodb.com/compare/mongodb-mysql, November 2018. Accessed on 2018-11-23.

[18] ITWissen, "Peer-to-peer-netz." https://www.itwissen.info/Peer-to-Peer-Netz-peer-to-peer-network-P2P.html, Januar 2014. Accessed on 2018-10-16.

[19] A. Donner, "Was ist nat (network address translation)?." https://www.ip-insider.de/was-ist-nat-network-address-translation-a-663954/, August 2017. Accessed on 2018-10-06.

[20] Wikipedia, "Websocket." https://de.wikipedia.org/wiki/WebSocket, Juli 2018. Accessed on 2018-10-16.

[21] Play, "What's new in play 2.5." https://www.playframework.com/documentation/2.6.x/Highlights25, November 2018. Accessed on 2018-11-15.

[22] Apple, "What's new in ios 9." https://developer.apple.com, June 2017. Accessed on 2018-11-14.

[23] ISRG, "Let's encrypt." https://letsencrypt.org/, Oktober 2018. Accessed on 2018-10-30.

[24] MongoDB, "Mongodb async java driver." http://mongodb.github.io/mongo-java-driver/3.8/driver-async/, November 2018. Accessed on 2018-11-13.

[25] Swagger, "Swagger-play2." https://github.com/swagger-api/swagger-play/tree/master/play-2.6/swagger-play2, November 2018. Accessed on 2018-11-13.

[26] E. Paraschiv, "Java timer." https://www.baeldung.com/java-timer-and-timertask, November 2018. Accessed on 2018-11-13.

[27] Prism, "Prism library." https://github.com/PrismLibrary/Prism, December 2018. Accessed on 2018-12-06.

[28] Microsoft, "Framework design guidelines." https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/, Dezember 2017. Accessed on 2018-09-18.

[29] M. Kuhrmann, "Rational unified process." http://www.enzyklopaedie-der-wirtschaftsinformatik.de/lexikon/is-management/Systementwicklung/Vorgehensmodell/Rational-Unified-Process-RUP/index.html, Juli 2018. Accessed on 2018-10-16.

[30] S. Master, "Scrum - auf einer seite erklaert." https://scrum-master.de/WasistScrum/ScrumaufeinerSeiteerklaert. Accessed on 2018-10-16.

Abbildungsverzeichnis

A Aufgabenstellung

Aufgabenstellung Studienarbeit
Elias Brunner, Oliver Dias-Lalcaca

Methode 365 als Cross-Plattform App

1. Betreuer

Diese Studienarbeit wird in Zusammenarbeit mit dem Institut für Software (IFS) durchgeführt.

Betreuer HSR:

Prof. Dr. Olaf Zimmermann, Institut für Software, ozimmerm@hsr.ch

2. Ausgangslage

Gerade in der IT-Welt ist das Finden von Lösungen für neu auftretende, aber auch für bekannte Probleme ein wichtiger Bestandteil des Jobs. Durch den Einsatz von Innovationsmethoden kann der Ideenfindung auf die Sprünge geholfen werden. In diesem Bereich finden sich die verschiedensten Ansätze und Methoden; in dieser Studienarbeit soll die Methode 635 verwendet werden. Methode 635 ist eine Kreativitäts- und Brainwriting-Technik, welche die Entwicklung von neuen, ungewöhnlichen Ideen für Problemlösungen in Gruppe fördert. Die Methode wird im PQM-Modul an der HSR vorgestellt.

Es gibt nach heutigem Kenntnisstand noch keine mobile App, die die Methode 635 unterstützt. Die Motivation für die Studienarbeit besteht darin, eine Cross-Plattform App zu konzipieren und zu implementieren. Dabei sollen moderne Technologien zum Einsatz kommen, welche es den Anwendern ermöglichen, schneller und einfacher eine Lösung für ein Problem zu erarbeiten.

3. Ziele der Arbeit und Liefergegenstände

In der Studienarbeit soll die Methode 635 als SmartPhone App umgesetzt werden. Android- und iOS-Support soll durch Verwendung von Xamarin erreicht werden.

Es wird erwartet, dass bis zum Ende des Projektes eine lauffähige und getestete Cross-Plattform Applikation umgesetzt wird, welche es Benutzern ermöglicht, die Methode 635 effektiv und effizient auf ihre Probleme anzuwenden.

Damit die App einen Mehrwert gegenüber der Papierversion bietet, soll es z.B. möglich sein, die Anzahl der Teilnehmer variabel zu bestimmen oder verschiedene Medien (Text, Video, Bilder, etc.) zu verwenden bzw. einzubinden. Die persistente Speicherung der bearbeiteten Problemstellungen soll aus Sicht des Kunden einfacher möglich sein als dies mit Papier möglich ist. Ein weiterer Vorteil einer mobilen Anwendung ist, dass Anwender die Methode 635 nutzen können auch wenn sie nicht am selben Ort sind oder die Lösungsvorschläge nicht zur selben Zeit bearbeiten.

Die Vision der Arbeit ist also, die Papierversion für diese Methodik zu funktional und qualitativ zu überbieten. Dabei spielen Erfolgsfaktoren wie einfache und intuitive Bedienung der App und ein

unkompliziertes Reporting sowie Robustheit und Stabilität (Bsp. keine Zeit- und Datenverluste) eine wichtige Rolle.

Weitere kritische Erfolgsfaktoren sind:

- Konfigurierbarkeit (z.B. Anzahl Teilnehmer und Schritte) und Erweiterbarkeit (im Hinblick auf Folgearbeiten, die u.U. auch andere Brainstorming Methoden unterstützen)
- sinnvolle Ausnutzung der Smartphone-Fähigkeiten, um einen Mehrwert im Vergleich zur traditionellen, papiergestützten Methode zu erreichen
- Validierung der Konzepte und ihrer Implementierung mit Hilfe von User Tests in mindestens einem Anwendungsbereich (Bsp. Architekturentscheidungen und -optionen).

4. Unterstützung

Die erwartete und effektiv erhaltene Unterstützung wird durch die Studenten in Sitzungsprotokollen definiert und im SA-Bericht dokumentiert.

5. Zur Durchführung

Mit dem HSR-Betreuer finden in der Regel wöchentlich Besprechungen statt. Zusätzliche Besprechungen sind nach Bedarf zu veranlassen. Besprechungen mit dem Auftraggeber werden nach Bedarf durchgeführt.

Alle Besprechungen, bei denen eine Vorbereitung durch den Betreuer nötig ist, sind von den Studenten mit einer Traktandenliste vorzubereiten. Beschlüsse sind in einem Protokoll zu dokumentieren.

Für die Durchführung der Arbeit ist ein Projektplan zu erstellen. Dabei ist auf einen kontinuierlichen und sichtbaren Arbeitsfortschritt zu achten. Arbeitszeiten sind zu dokumentieren.

Die Spezifikation der Anforderungen geschieht durch die Studenten in Absprache mit dem Betreuer. Bei Disputen entscheidet der Betreuer in Rücksprache mit den Studenten über die definitiv für die Studienarbeit relevanten Anforderungen.

Vorstudie, Anforderungsdokumentation und Architekturdokumentation sollten im Laufe des Projektes mittels Milestone mit dem Auftraggebern und dem Betreuer in einem stabilen Zustand abgenommen werden. Zu den abgegebenen Arbeitsresultaten wird ein vorläufiges Feedback abgegeben. Eine definitive Beurteilung erfolgt auf Grund der am Abgabetermin abgelieferten Dokumentation.

Die Rechte an den Ergebnissen der Studienarbeit werden in einer separaten Vereinbarung definiert (gemäss Vorlage Studiengang: die Ergebnisse der Arbeit dürfen sowohl von der Studentin / dem Student wie von der HSR und Prof. Zimmermann nach Abschluss der Arbeit verwendet und weiter entwickelt werden).

6. Dokumentation

Über diese Arbeit ist eine Dokumentation gemäss den Richtlinien der Abteilung Informatik zu verfassen. Die zu erstellenden Dokumente sind im Projektplan festzuhalten. Alle Dokumente sind nachzuführen, d.h. sie sollten den Stand der Arbeit bei der Abgabe in konsistenter Form dokumentieren. Bei der Projektdokumentation und der Abgabe sind die Anleitungen des Studienganges inklusive Anhängen zu beachten. Zudem ist eine kurze Projektresultatdokumentation für das Wiki von Prof. Zimmermann erwünscht (ggfs. kurzes Video).

7. Termine

Siehe HSR-Webseiten.Allfällige weitere Termine sind am Sekretariat der Abteilung Informatik zu erfragen und sollten entsprechend in einem Sitzungsprotokoll dokumentiert werden.

8. Beurteilung

Eine erfolgreiche Studienarbeit zählt 8 ECTS-Punkte pro Studierenden. Für 1 ECTS-Punkt ist eine Arbeitsleistung von ca. 25 bis 30 Stunden budgetiert.

Siehe http://studien.hsr.ch/allModules/23498_M_SAI.html für die Modulbeschreibung der Studienarbeiten.

Für die Beurteilung sind die HSR-Betreuer verantwortlich unter Einbezug des Feedbacks der Projektpartner.

Gesichtspunkt	Gewicht
1. Organisation, Durchführung	1/5
2. Berichte (Abstract, Management Summary, technische u. persönliche Berichte) sowie Gliederung, Darstellung und Sprache der gesamten Dokumentation.	1/5
3. Inhalt*)	3/5

*) Die Unterteilung und Gewichtung von 3. Inhalt wird im Laufe dieser Arbeit festgelegt.

Im Übrigen gelten die Bestimmungen der Abt. Informatik zur Durchführung von Studienarbeiten.

Rapperswil, den 20.09.2018

Prof. Dr. Olaf Zimmermann
Institut für Software
Hochschule für Technik Rapperswil

B Installationsanleitung

B.1 Zweck dieses Dokuments

Die Installationsanleitung bietet interessierten Personen eine Schritt-für-Schritt Anleitung, wie das gesamte System installiert wird.

B.2 Anforderungen

Für die Installation der Cross-Plattform Applikation auf dem eigenen Smartphone wird Folgendes vorausgesetzt:

- Visual Studio IDE Community Edition (7.6.11 (build 9))
- Android + Xamarin.Forms
- iOS + Xamarin.Forms
- .NET Core + ASP.Net Core

Die Visual Studio IDE kann unter www.visualstudio.microsoft.com heruntergeladen werden.

Wem es nur darum geht, die Xamarin Applikation auszuprobieren, kann bei Kapitel B.3 weiterlesen. Wer allerdings zusätzlich noch das PlayFramework laufen lassen möchte, benötigt zusätzlich noch Folgendes:

- Java SE 1.8 oder höher
- Ein Build-Tool wie SBT oder Gradle
- Docker Community Edition

Weitere Informationen zur Installation von Java oder den Build-Tools sind unter www.playframework.com zu finden.

B.3 Installation

Bevor die eigentliche Installation jedoch beginnen kann, muss man noch über den Quellcode verfügen. Dieser kann von unseren Github Repositories heruntergeladen werden.

Installation der Cross-Plattform Applikation

Für die Installation der Xamarin Applikation für ein **Android Smartphone** muss folgendermassen vorgegangen werden.

1. Öffne das Projekt mit Visual Studio
2. Wähle das "Method635.App.Forms.Android "als Startprojekt
3. Schliesse dein Smartphone mittels USB-Kabel an deine Entwicklungshardware an
4. Mittels F5 oder dem Play Button kann die Applikation auf dem Smartphone gestartet werden

Für die Installation der Xamarin Applikation für ein **iOS Smartphone** muss etwas anders vorgegangen werden. Dabei sind allerdings ein Entwickler Account von Apple sowie ein Apple Mac notwendig.

1. Öffne zuerst Xcode und erstelle ein leeres Projekt
2. Schliesse dein Smartphone mittels USB-Kabel an deine Entwicklungshardware an
3. Signiere das leere Projekt und stelle es auf deinem iOS Gerät bereit
4. Öffne nun das Method635-Projekt mit Visual Studio
5. Wähle das "Method635.App.Forms.iOS "als Startprojekt
6. Mittels F5 oder dem Play Button kann die Applikation auf dem Smartphone gestartet werden

Bemerkung: Sollte dabei der Fehler "resource fork, Finder information, or similar detritus not allowed" auftreten, kann dieser wie auf www.stackoverflow.com beschrieben, vermieden werden. Danach muss das Projekt aber zuerst bereinigt werden.

Installation des PlayFrameworks

Für die Installation des PlayFrameworks muss zuerst eine Datenbank mit den entsprechenden Collections bereitgestellt werden. Dazu nutzen wir eine MongoDB-Instanz, welche in einem Docker-Container läuft.

1. docker run -name methode635 -p 40002:27017 -d mongo:latest
2. docker exec -it methode635 bash

3. use Methode635
4. db.createCollection("BrainstormingFinding")
5. db.createCollection("BrainstormingTeam")
6. db.createCollection("Participant")

Weitere Informationen zu MongoDB als Docker-Container können unter hub.docker.com nachgelesen werden. Wer mehr über die createCollection Befehle erfahren möchte, findet unter docs.mongodb.com mehr dazu.

Um das PlayFramework zu builden und laufen zu lassen genügt es, im **API Projekt-Ordner** folgenden Befehl auszuführen.

1. sbt run "40000"

Weitere Hilfestellungen im Umgang mit SBT und dem PlayFramework finden sich auf playframework.com.

Bemerkung: Allenfalls muss für die Interaktion zwischen API und der Datenbank der Username, das Passwort sowie der Datenbankname geändert werden. Dies kann in allen Controllerklassen angepasst werden (siehe Zeile 3 und 5).

```
1 public ParticipantController(){
2 ...
3 mongoClient = MongoClients.create(new
     ConnectionString("mongodb://user:
     password@localhost:40002/?authSource=admin&
     authMechanism=SCRAM-SHA-1"));
4
5 database = mongoClient.getDatabase("database");
6 ...
7 participantCollection = database.getCollection("
     Participant", Participant.class);
8
9 }
```

Listing 9: Verbindung zur MongoDB

Auch muss im Quellcode der Cross-Plattform Applikation der URL auf die IP deines PC gewechselt werden. Dies kann in der RestResolverBase Klasse geändert werden (Zeile 3).

```
1 public abstract class RestResolverBase
2 {
3 private const string URL = "https://sinv-56079.edu.
     hsr.ch";
4 private const int PORT = 40000;
5 }
```

Listing 10: Verbindung zum API

C Testprotokoll

C.1 Zweck dieses Dokuments

Um sicher zu stellen, dass unsere Cross-Plattform Applikation in ihrer Kernfunktionalität richtig funktioniert, wurde ein systematischer Test anhand des nachfolgenden Testprotokolls durchgeführt.

C.2 Verweise

Die Testfälle leiten sich von den Fully-Dressed Use Cases aus Kapitel 2.5.1.2 ab.

C.3 Testaufbau

Der Test wurde am 06. Dezember 2018 von Oliver Dias, Elias Brunner und Prof. Dr. Zimmermann einmal komplett anhand des vorliegenden Protokolls durchgeführt. Dabei wurde ein iPhone 6S mit der Version iOS 12.1 und zwei Android Smartphones (Nexus 5X und Huawei P10) mit der Android-Version 8.1.0 verwendet. Die Testdurchführung erfolgte manuell. Die einzelnen Testfälle wurden allerdings auch während der Entwicklung mehrfach individuell getestet.

C.4 Testfälle

Use Case 7: View Brainstorming Finding			
Test Nr	Beschreibung	Erwartetes Resultat	Beobachtetes Verhalten (Bestanden?)
1	Als Participant möchte ich als Letzter die letzte Runde abschliessen.	Das System speichert meine Notizen auf dem Server und zeigt mir eine Meldung an, dass das Finding einsehbar ist.	Ja, Führung für User könnte besser sein.
2	Als Participant möchte ich die Resultate der Gruppe ansehen.	Das System zeigt mir eine Übersicht mit allen Notizen der Teilnehmer.	Ja.
3	Als Participant möchte ich nicht als Letzter die letzte Runde abschliessen.	Das System speichert meine Notizen auf dem Server und zeigt mir die verbleibende Zeit an.	Ja, Führung für User könnte besser sein.

4	Nach Ablauf der Zeit meldet mir das System, dass das Finding einsehbar ist.	Das System zeigt mir eine Übersicht mit allen Notizen der Teilnehmer.	Ja.
5	Als Participant möchte ich nicht direkt zu den Resultaten der Gruppe navigieren sondern zurück zum Homescreen gelangen.	Das System zeigt mir den Homescreen an. Von dort aus gelange ich aber auch zu den Notizen der Teilnehmer.	Ja.

Use Case 8: Create Brainwave			
Test Nr	Beschreibung	Erwartetes Resultat	Beobachtetes Verhalten (Bestanden?)
1	Als Participant möchte ich Notizen während der laufenden Rundenzeit zum aktuellen Problem erfassen.	Das System zeigt meine Notizen an.	Ja, nicht immer klar wo die Idee eingefügt wurde.
2	Als Participant möchte ich meine Notizen frühzeitig abgeben.	Das System persistiert meine Notizen und zeigt mir eine Bestätigung an.	Nein, keine Rückmeldung für den User.
3	Als Participant möchte ich meine Notizen nicht frühzeitig abgeben.	Das System zeigt mir eine Meldung an, dass die Zeit der aktuellen Runde abgelaufen ist. Es persistiert die bis zu dem Zeitpunkt erfassten Notizen.	Nein, keine Rückmeldung für den User.

Use Case 9: Start Brainstorming			
Test Nr	Beschreibung	Erwartetes Resultat	Beobachtetes Verhalten (Bestanden?)

1	Als Moderator möchte ich ein Brainstorming-Session starten.	Das System überprüft zunächst, ob alle Einstellungen stimmen und startet dann die Brainstorming-Session.	Ja.
2	Als Participant möchte ich ein Brainstorming-Session starten.	Als Participant ist es nicht möglich eine Brainstorming-Session zu starten	Ja.

C.5 Testauswertung

Nachdem wir die Testfälle mehrmals mit einem iPhone und zwei Android-Smartphones durchgeführt haben, haben sich folgende Punkte für Verbesserungen herauskristallisiert:

1. In der gesamten Applikation gibt es fast keine Führung für den User. Damit ist die Orientierung für den Enduser gerade zu Beginn noch sehr schwierig und nicht immer intuitiv. Hier würde es sich anbieten, diese beim ersten Start in einem Tutorial zu erklären und dabei Fragen wie "Wo bin ich?" oder "Was ist hier zu tun?" zu erläutern.

2. Auch gibt es keine Rückmeldungen vom System an den User (Erfolgreiche Eingabe etc.), was den Umstand der schwierigen Orientierung von verstärkt.

3. Ein weiterer Punkt, welcher für die schlechte Userführung verantwortlich ist, war der Umstand, dass bei der Eingabe der einzelnen Ideen nicht klar war, dass man nach links und rechts wischen kann, um die anderen Ideen und Blätter anzusehen. Hier würde dem User ein Label mit der aktuellen Blattnummer (Sheet $1/3$) viel Klarheit schaffen.

4. Weiter würde eine Umbenennung einzelner Buttons ("Login" zu "Login to Methode 635" oder "Add Brainstorming Finding" zu "Add Brainstorming Session") zur Verbesserung der Benutzerführung beitragen.

5. Als zusätzliches Feature wurde die Möglichkeit gesehen, die Ideen einer abgeschlossenen Brainstorming Session als PDF abzuspeichern oder gar einer anderen App zur Verfügung zu stellen. Auch wäre hier eine "Copy-To-Clipboard" Funktion vorstellbar.

Während der Testdurchführung hat sich herausgestellt, dass der automatische Rundenwechsel von der Applikation nicht immer korrekt erkannt und durchgeführt

wird. Dies führt dann zu irreführenden Zuständen für den Benutzer. Dafür konnte aber ein Workaround mittels einer Sync-Funktion entwickelt werden. Sollte der Rundenwechsel nicht automatisch funktionieren, kann auf das Logo geklickt werden, um den Zustand neu vom Server zu laden.