DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Konzeption und Implementierung eines Sachwarmverhaltens von mobilen Kleinrobotern anhand eines Verfolgungsszenarios

STUDIENARBEIT

für die Prüfung zum

Bachelor of Science

des Studiengangs Informatik
Studienrichtung Angewandte Informatik

an der

Dualen Hochschule Baden-Württemberg Karlsruhe

20. März 2017

Bearbeitungszeitraum	24 Wochen	
Name	Manuel Bothner	Simon Lang
Matrikelnummer	8359139	6794837
Kurs	TINF14B2	TINF14B2
Ausbildungsfirma	1&1 Internet SE	ifm ecomatic GmbH
	Brauerstr. 48	Im Heidach 18
	76135 Karlsruhe	88079 Kressbronn am Bodensee
Betreuer	Prof. Hans-Jörg Haubner	
Gutachter	Prof. Dr. Heinrich Braun	

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Erklärung

(gemäß §5(3) der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. 9. 2015)
Ich versichere hiermit, dass ich die Studienarbeit meiner Studienarbeit mit dem Thema: „Konzeption und Implementierung eines Sachwarmverhaltens von mobilen Kleinrobotern anhand eines Verfolgungsszenarios“ selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Ort, Datum Unterschrift

Ort, Datum Unterschrift

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Abstract

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Zusammenfassung

Inhaltsverzeichnis

Abkürzungsverzeichnis

API Application Programming Interface.

App Application.

Glossar

Application Eine Application ist ein ausführbares Programm für mobile Geräte, wie Smartphones oder Tablets.

Application Programming Interface Eine API ist eine Programmierschnittstelle, die die Anbindung von Software ermöglicht.

Abbildungsverzeichnis

DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

Tabellenverzeichnis

1 Einleitung

Heutzutage werden viele Arbeitsschritte in der Produktion, als auch Dienstleistungen von Maschinen verrichtet, da diese effizienter Arbeiten und weniger Kosten als Menschen verursachen. Da jede Maschine auf einen spezifischen Arbeitsschritt konfiguriert ist, müssen die verschiedenen Maschinen untereinander wie ein Schwarm agieren. Diese Verhaltensstrukturen kommen ursprünglich aus dem Tierreich, wie Fischschwärme, Ameisen oder Bienen. Hierbei erledigt jedes Individuum seine zugewiesenen Aufgaben und hält die anderen Parteien auf dem aktuellen Stand.
In diesem Projekt werden diese Verhaltensmuster aus dem Tierreich aufgegriffen und anhand eines Verhaltensszenarios mit Kleinrobotern verwirklicht, die autonom agieren und kommunizieren, um zusammen ihr Ziel zu erreichen. Dabei sollen Konzepte, sowie Algorithmen für Schwarmroboter entstehen, die auch auf andere Szenarien angewendet werden können.

1.1 Ausgangslage

1.2 Zielsetzung

1.3 Erwartetes Ergebnis

2 Technische Grundlagen

2.1 Robotik

2.1.1 Grundlagen

2.1.2 Mobile Roboter

2.1.3 Antriebsarten

2.1.4 Sensorik

2.1.5 LEGO Mindstorm

2.2 App Entwicklung

Eine App ist ein ausführbares Programm für mobile Geräte, wie Smartphones oder Tablets. Um eine App für ein mobiles Gerät zu entwickeln, müssen wie für andere Anwendungen im Voraus Anforderungen definiert werden, die diese erfüllen soll. Je nach festgelegten Anforderungen, die an das System gestellt werden, besteht eine bestimmte Anzahl von Möglichkeiten der Entwicklung. Allgemein kennt die App Entwicklung drei verschiedene Arten, die native, web und hybride Entwicklung, siehe (2.2.1), (2.2.2) und (2.2.3). Dabei werden verschiedene Frameworks verwendet, um mit unterschiedlichsten Programmiersprachen den Aufbau der Logik zu beschreiben. Eine App besteht immer aus zwei Teile, dem (UI), das meist mit einer (XML) ähnlichen Sprache beschrieben wird und dem Programmcode, der sich auf viele Klassen verteilt und die Funktionalitäten der App beschreiben.

Abbildung 1: App Entwicklung

2.2.1 Native Apps

In der Entwicklung von nativen Apps werden die direkten Ressourcen des Gerätes verwendet. Dazu gehört die Laufzeitumgebung des Betriebssystemes, Bibliotheken und Hardwareschnittstellen. Der Vorteil von einer nativen Entwicklung liegt hauptsächlich darin, dass diese für das Betriebssystem optimiert ist und die vorhandenen Schnittstellen genutzt werden können, um komplexe und rechenintensive Anwendungen zu ermöglichen.[1]
Vertreter diese Entwicklung finden sich für verschiedene Betriebssysteme. Der populärste unter ihnen ist bei weitem Android mit einer nativen Java Entwicklung über Android Studio von Google. Sie besitzt aktuellen den höchsten Marktanteil und eine entsprechende Popularität unter Entwickler und Nutzer.

2.2.2 Web Apps

Die Entwicklung von web Apps arbeitet mit systemübergreifenden Ressourcen und greift dabei auf gängige Webtechnologien, wie (HTML), (CSS) und JavaScript zurück. Die App wird hierbei nicht wie normale Anwendungen direkt auf dem System des Gerätes ausgeführt, sondern kommt in dessen Browser zur Ausführung. Der Vorteil hierbei ist vor allem, dass diese Art von App auf allen Betriebssystemen lauffähig ist und direkt über das Internet veröffentlicht und aktualisiert werden kann, jedoch wird eine stabile Internetverbindung vorausgesetzt.[1]

[1][vgl. 1, Unterschiede und Vergleich native Apps vs. Web Apps]

Von dieser Entwicklung finden sich viele Vertreter mit der Unterstützung diverser Frameworks. Das populärste unter ihnen ist aktuell AngularJS von Google, was auf JavaScript basiert. In Kombination mit anderen Webtechnologien, wie glshtml und CSS lassen sich perfomante web Apps entwickeln.

2.2.3 Hybride Apps

Die Entwicklung von hybride Apps vereinigt die beiden Entwicklungen von native und web. Sie besteht dabei aus einem nativen Rahmen, in der eine web App zur Ausführung kommt, diese besitzt entsprechende Zugriffsrechte auf Hardwareschnittstellen, um diese mit Application Programming Interfaces (APIs) anzusprechen.[2]
Diese Entwicklung ist aktuell noch sehr jung, jedoch stechen hier bereits verschiedene Vertreter hervor. Der populärste unter ihnen ist Ionic von Drifty, welches auf Apache Cordova als Basis zurückgreift. In Kombination mit AngularJS, TypeScript und anderen Webtechnologien lässt sich die web App entwickeln und auf einem beliebigen Gerät unter einem nativen Browser ausführen. Es unterstützt dabei verschiedenste Betriebssystem, wie Android, iOS und Windows. Diese Entwicklungen können dabei meist nicht nur mobil, sondern unter anderem auf weiteren Systemen, wie stationäre bereitgestellt werden.

2.2.4 Plattformübergreifende Entwicklung

Um die Entwicklung von Apps einfach zu halten, verwenden immer mehr Entwickler die Form der plattformübergreifenden Entwicklung. Dadurch lässt sich die App unabhängig des Betriebssystems entwickeln und kann somit eine größere Menge von Nutzern erreichen. Diese Entwicklung greift dabei meist auf plattformübergreifende Konzepte, wie eine native Laufzeitumgebung, oder Browser zurück, um darin die App auszuführen. Der große Vorteil in dieser Entwicklung, liegt in der Wiederverwendbarkeit des Quellcodes und der verbesserten Wartbarkeit, da hier lediglich ein Projekt gewartet werden muss und der Quellcode für viele Betriebssysteme übernommen werden kann. Zur plattformübergreifenden Entwicklung wurden die letzten Jahre viele Ansätze mit verschiedenen Frameworks entwickelt. Beispiele hierfür sind Ionic, Unity, Qt oder Xamarin.

[2][vgl. 2, Native App, Web App und Hybrid App im Überblick]

2.2.5 Xamarin

Xamarin ist ein Framework zur Entwicklung von nativen plattformübergreifenden Apps. Dabei baut Xamarin auf Mono, einer opensource Version des .NET Framework, welches auf den .NET ECMA Standards basiert.[3]
Um nativen Quellcode auf den verschiedenen Systemen auszuführen, setzt Xamarin auf verschiedene Softwarekomponenten. Für iOS Systeme verwendet Xamarin den AOT Compiler, um aus einem Xamarin.iOS Projekt binären Quellcode für ARM zu erzeugen. Bei Android nutzt Xamarin die IL, um JIT nativen Quellcode für die entsprechende Hardware zu compilieren und die App auszuführen.

Abbildung 2: Xamarin

2.2.6 Mono

Mono ist eine opensource Laufzeitumgebung für Linux Betriebssysteme, um Anwendungen auszuführen, die auf dem .NET Framework basieren. Dabei greift Mono auf Standards des CLI und ECMA von C# zurück. Gestartet wurde das Projekt durch die Firma Novell und aktuell weiterentwickelt von Microsoft und wird dadurch auf gleichem Stand wie .NET gehalten.

Abbildung 3: Mono

2.2.7 .NET Framework

Das .NET Framework ist eine Laufzeitumgebung für .NET Anwendungen, die verschiedene Dienste bereitstellt. Es besteht aus zwei Hauptkomponenten, der CLR, die eine Speicherverwaltung und verschiedene Systemdienste bereitstellt, sowie der .NET Bibliothek. Um Anwendungen für .NET zu entwickeln, wird die entsprechende Version von .NET Framework auf dem System benötigt. Als Programmiersprache ist der Entwickler weitgehend unabhängig, der Quellcode muss jedoch die CLI-Spezifikationen erfüllen. Dafür eignen sich unter anderem die Programmiersprachen von Microsoft, wie VisualBasic, C#, VisulF# und C++.

[3][vgl. **?** , Introduction to Mobile Development - Xamarin]

2.3 Java

2.3.1 Grundlagen

2.3.2 Java Runtime Environment

3 Theoretische Grundlagen

3.1 Schwarmverhalten

3.1.1 Allgemein

3.1.2 Vorbilder aus dem Tierreich

3.1.3 Szenarien

3.1.4 Algorithmen

3.2 Kommunikation

3.2.1 Grundlagen

3.2.2 TCP/IP

3.2.3 Wifi

3.2.4 Datenaustausch

4 Projektorganisation

4.1 Projektablaufplan

5 Konzeption

In diesem Kapitel werden die Anforderungsdefinitionen des Projektes, mit Spezialisierung auf die verschiedenen Use Cases beschrieben.

5.1 Anforderunsdefinitionen

In diesem Abschnitt wird auf die Funktionalitäten und Use Cases des Projektes eingegangen.

5.1.1 Softwarearchitektur

5.2 Use Cases

5.2.1 Connect

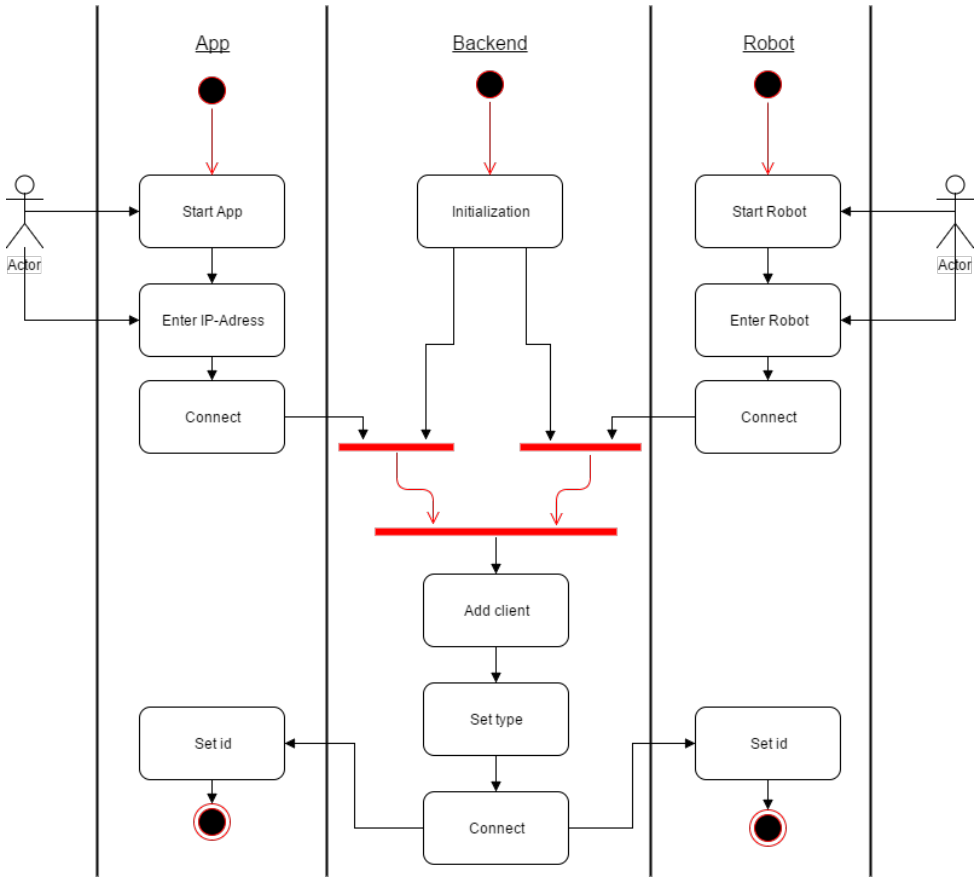
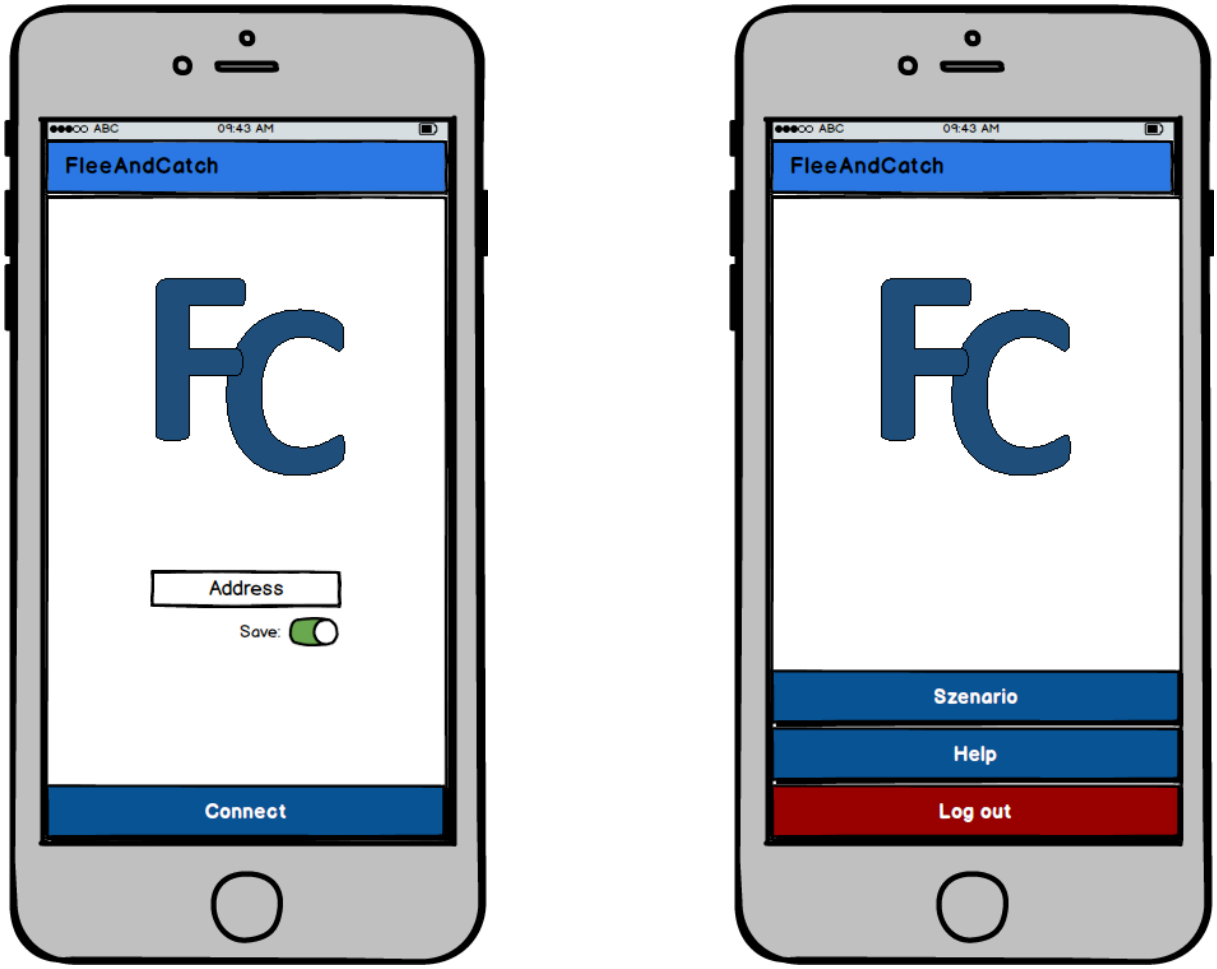


Abbildung 4: Connect

Im Use Case Connect wird eine erste Verbindung durch die Eingabe der IP-Adresse zum Backend aufgebaut. Dabei sendet die Komponente, ob Roboter oder App eine Abbildung seiner selbst als Objekt dem Backend. Daraufhin startet das Backend die Verbindung indem es der Komponente entsprechende Verbindungskommandos zusendet. Sobald eine



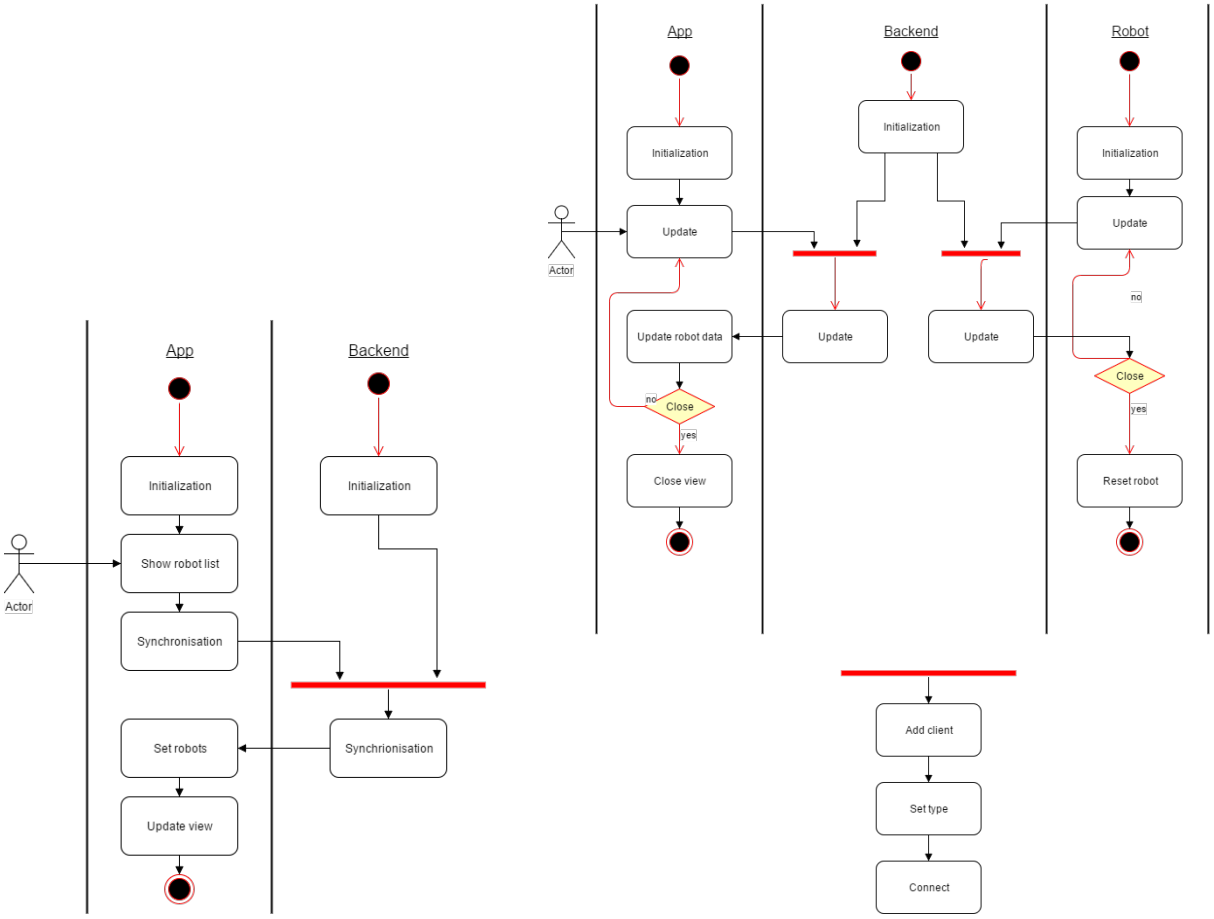
(a) Connect (b) Home

Abbildung 5: Connection

Reaktion in einer festgelegten Zeit erfolgt, akzeptiert das Backend die Verbindung und sendet die entsprechende Id für die Komponente. Ab diesem Moment ist die Komponente verbunden und ein Robot für Aktionen entsprechend verfügbar. Die Verbindungsinitialisierung dient hierbei der Verkürzung der Reaktionszeit, die bei einem Roboter sonst entsprechend hoch wäre.

5.2.2 Synchronization

Im Use Case Synchronization werden Daten entsprechend des gesetzten Typen zwischen den Komponenten übertragen. Dabei können einerseits die Roboter als Objekte, oder ganze Szenarien übertragen werden. Dies dient zur Gegenseitigen Synchronisierung der Daten.



(a) Update all

(b) Update

Abbildung 6: Synchronization

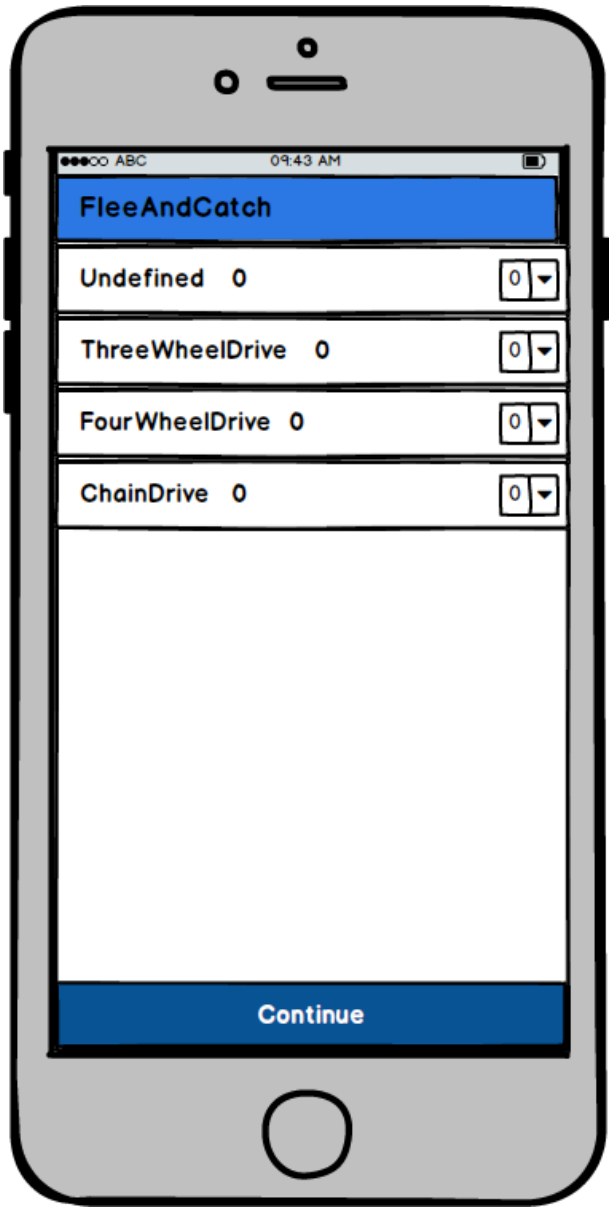


Abbildung 7: Robot list

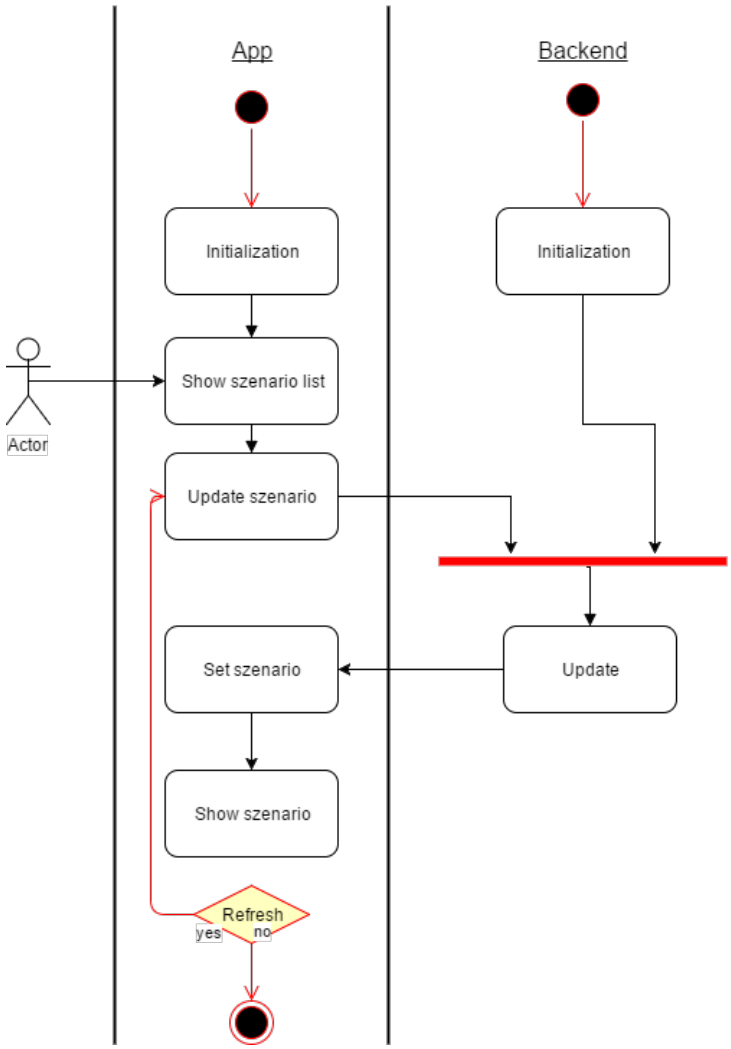


Abbildung 8: Spectator

5.2.3 Szenario

5.2.4 Exception

5.3 Kommunikation

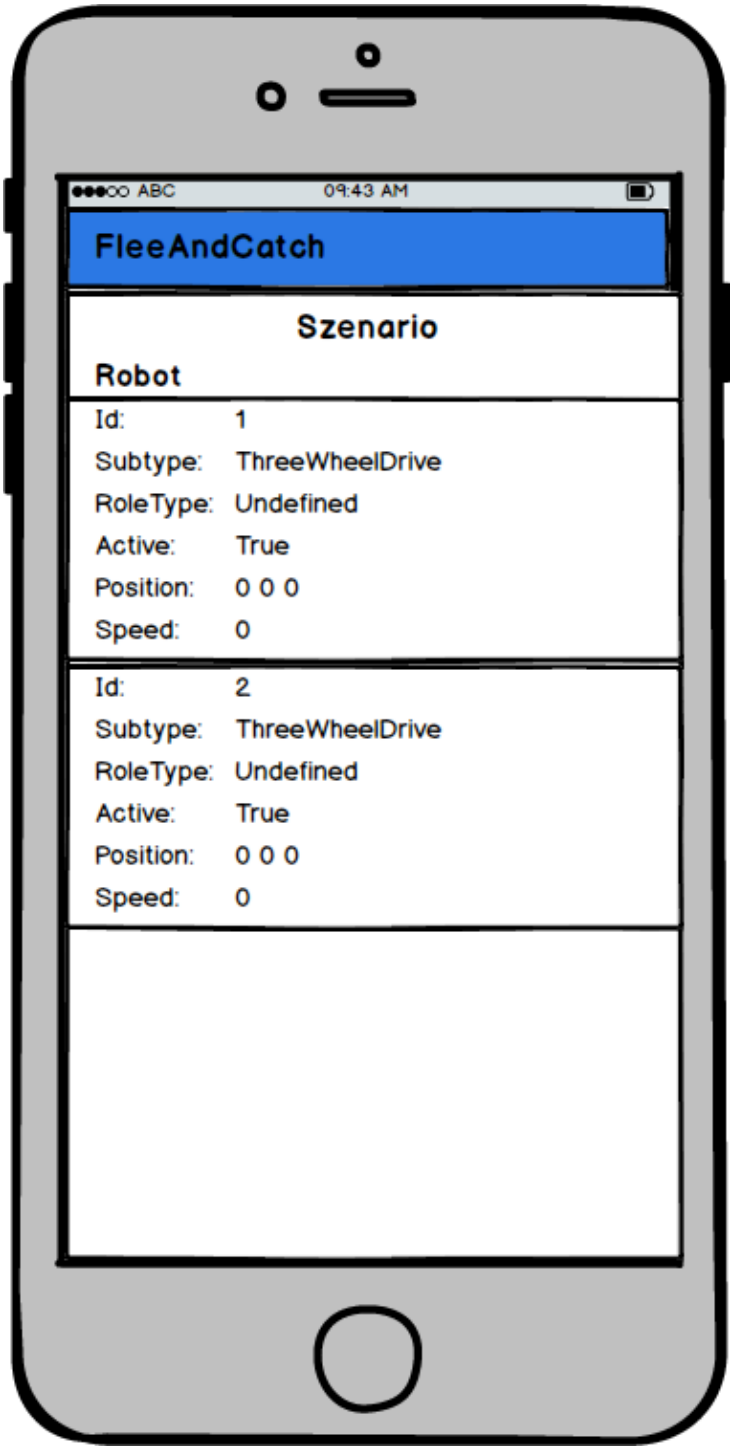


Abbildung 9: Spectator

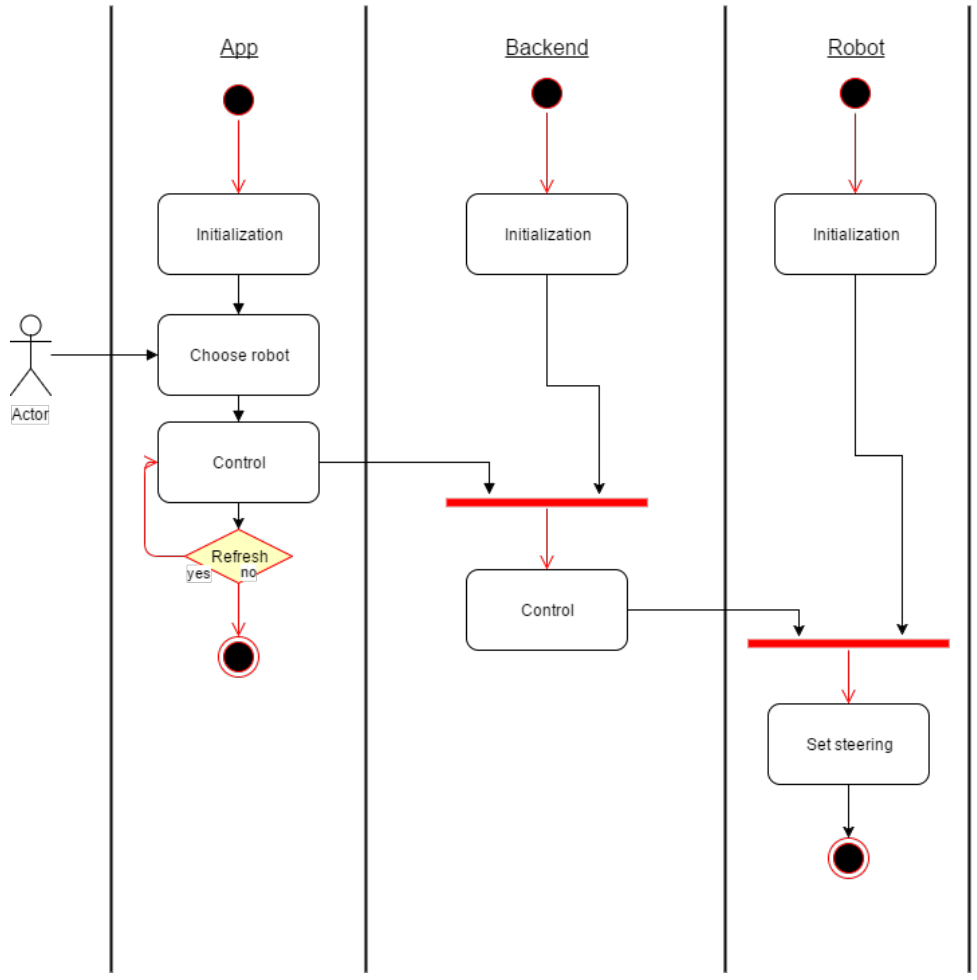


Abbildung 10: Control

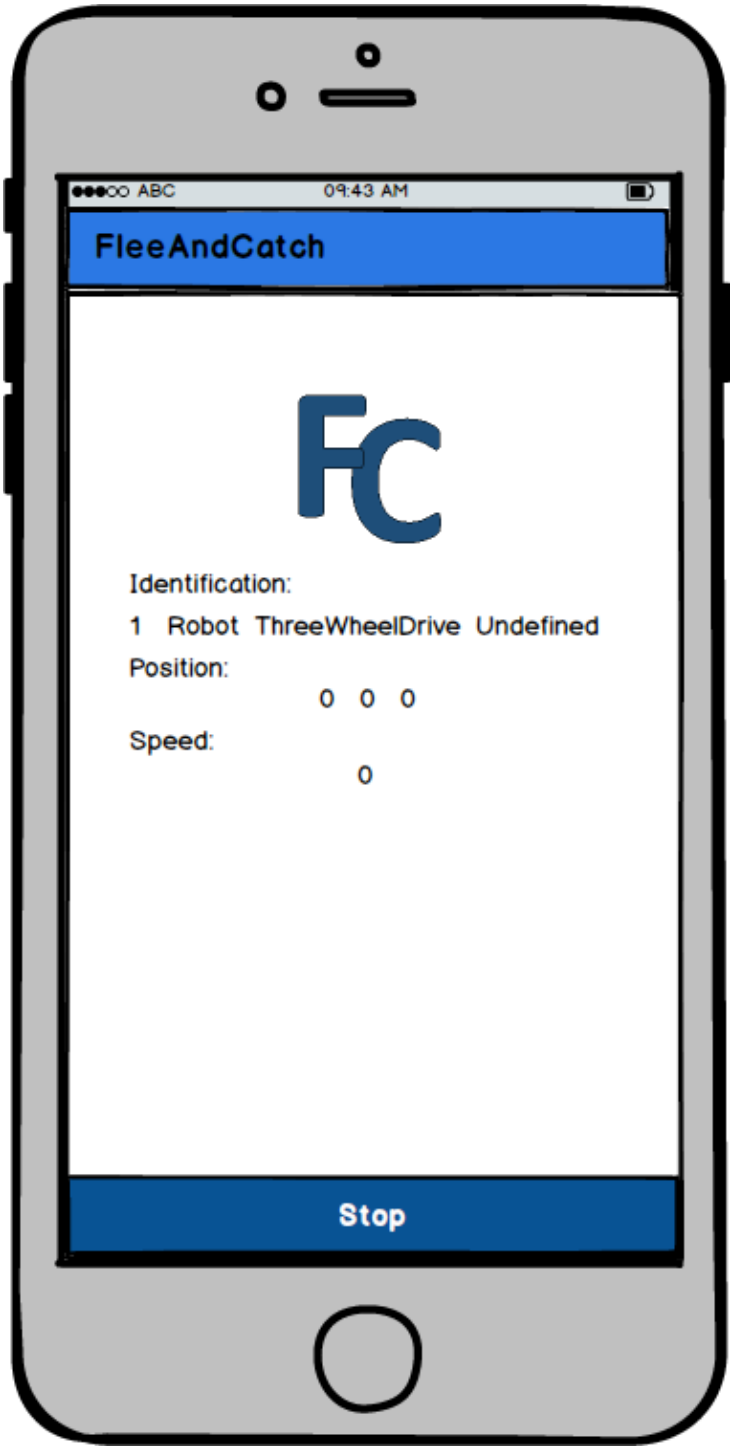


Abbildung 11: Control

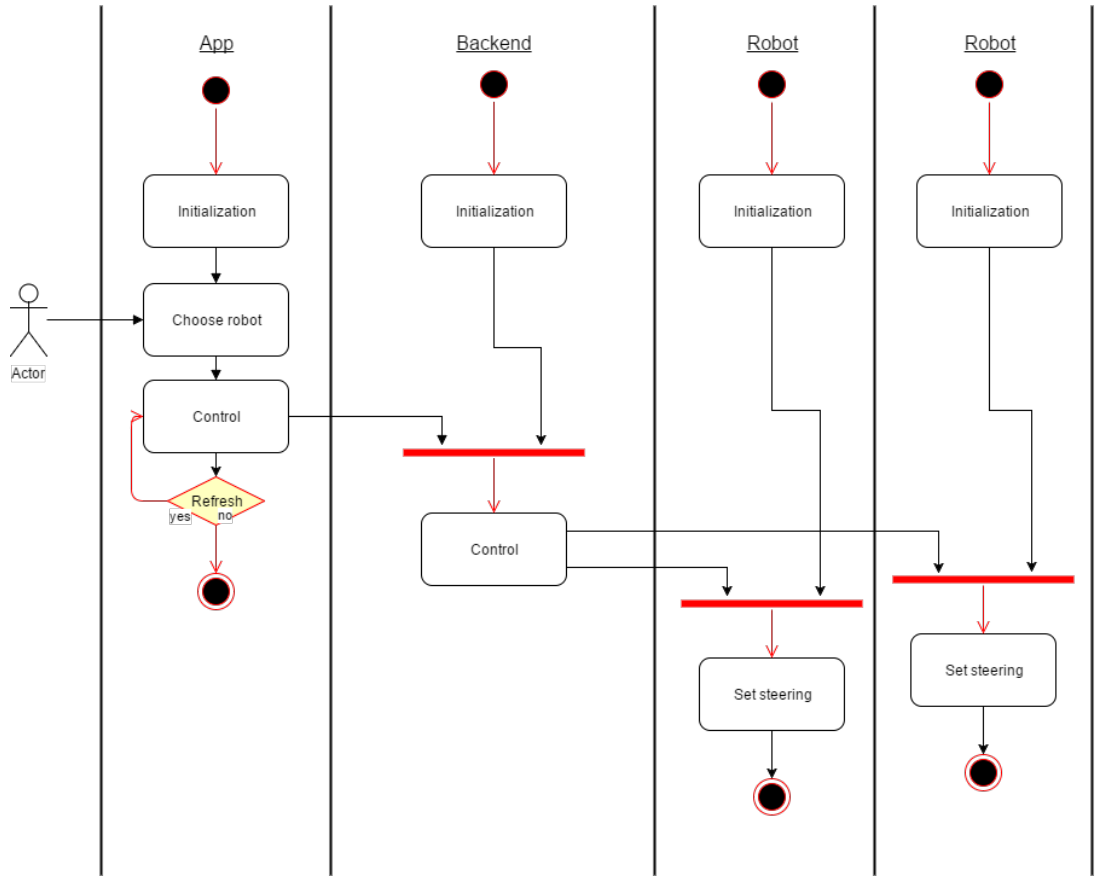


Abbildung 12: Synchron

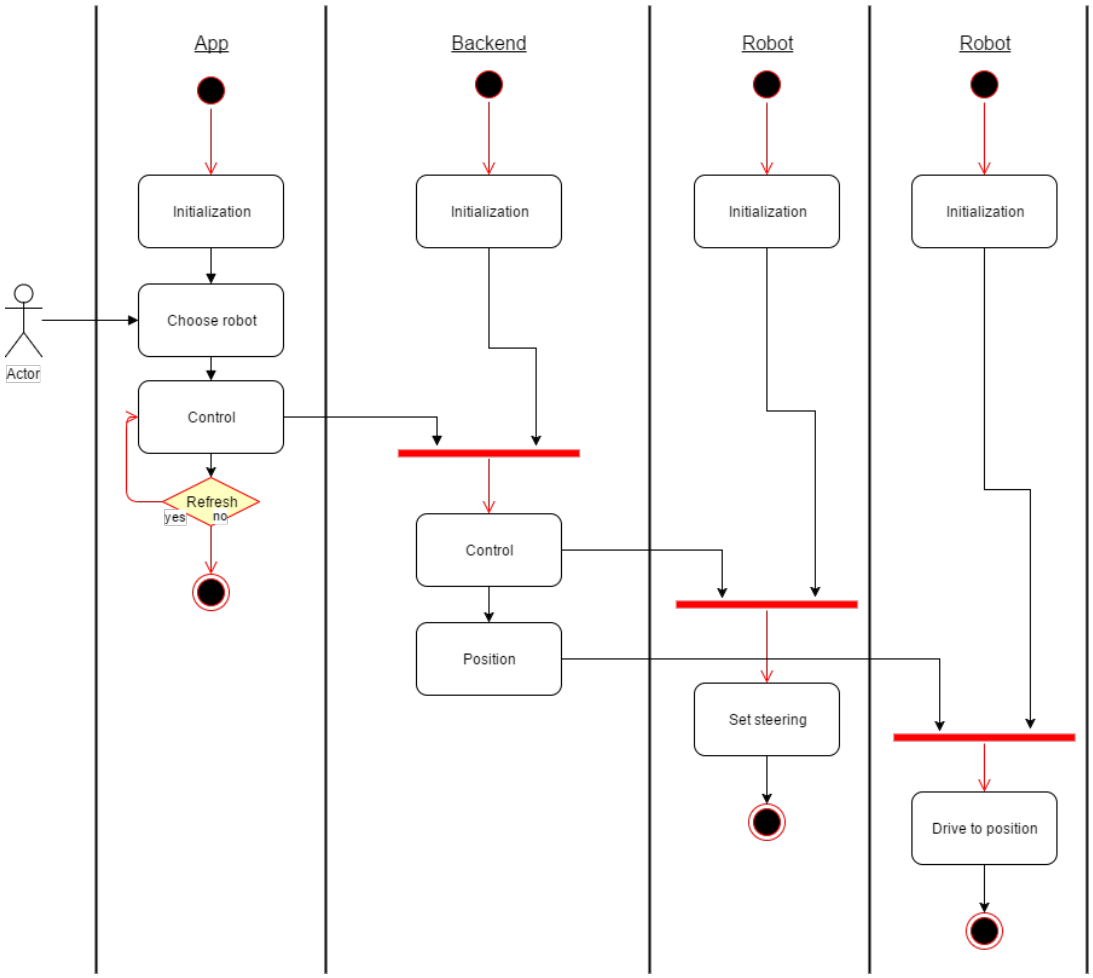


Abbildung 13: Follow

6 Lösungsansatz

7 Umsetzung

8 Evaluation

9 Zusammenfassung und Ausblick

Literatur

[1] Daniel Würstl. Unterschiede und vergleich native apps vs. web apps. URL http://www.app-entwickler-verzeichnis.de/faq-app-entwicklung/11-definitionen/107-unterschiede-und-vergleich-native-apps-vs-web-apps.

[2] Petra Riepe. Native app, web app und hybrid app im überblick: Warum native wenn es auch hybrid geht? URL http://www.computerwoche.de/a/warum-native-wenn-es-auch-hybrid-geht,3096411.

Anhang