Fachhochschule Nordwestschweiz
Hochschule für Technik

Cloudbasiertes Praxisrufsystem

IP 5

31. Juli 2021

Abbildung 0.1: Titlebild

Studenten	Joshua Villing, Kevin Zellweger
Fachbetreuer	Daniel Jossen
Auftraggeberin	Daniel Jossen
Studiengang	Informatik
Hochschule	Hochschule für Technik

Zusammenfassung

Das Abstract ist eine Art Zusammenfassung des ganzen Dokuments. Es gibt einen Einblick in die Aufgabenstellung, wie diese umgesetzt wurde und welches Ergebnis erreicht wurde. Aus diesem Grund wird das Abstract immer ganz am Schluss der Arbeit verfasst. Es besteht aus einem zusammengehörenden Absatz und umfasst ungefähr 10 bis 20 Zeilen. Formeln, Referenzen oder andere Unterbrechungen haben im Text nichts zu suchen. Direkt unter dem Abstract folgt eine Liste von drei bis vier Stichworten/Keywords. Diese werden in alphabetischer Reihenfolge aufgelistet und beschreiben das Themengebiet der Arbeit.

Keywords: Anleitung, LaTeX, Thesis, Vorlage

Management Summary siehe PF-IK.

Vorwort

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Fakultativ, siehe PF-IK (URL) Balalbala some edits. **knuth1990**

Inhaltsverzeichnis

1 Einleitung

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Einleitungsbeispiele siehe PF-IK (URL)

2 Vorgehensweise

2.1 Stakeholder

Am Projekt IP5 Cloudbasiertes Praxisrufsystem sind folgende drei Stakeholder beteiligt.

Prof. Daniel Jossen

- Rolle: Auftraggeber und Betreuer
- Kontakt: daniel.jossen@fhnw.ch

Joshua Villing

- Rolle: Student
- Kontakt: joshua.villing@students.fhnw.ch

Kevin Zellweger

- Rolle: Student
- Kontakt: kevin.zellweger@students.fhnw.ch

2.2 Projektplan

Übersicht

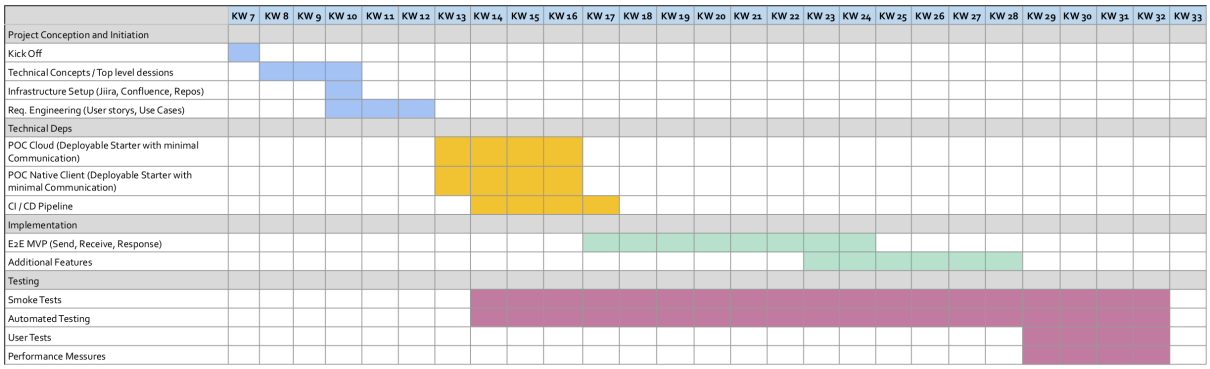

Abbildung 2.1: Projektplan

Milestones

Milestones

POC: Mobile Client ->Cloud Nachricht schicken und etwas persistieren

Wahrscheinlich über HTTP / Rest

POC: Cloud ->Mobile Client Nachricht schicken und etwas anzeigen

Wahrscheinlich über Message Broke

Versenden mit hinterlegter Konfiguration

Konfigurierte Notification Types

1:N Versenden, emfpangen und konfigurieren

Setup Wizard (Neu oder z.B. wie Praxiszimmer XY)

Voice to Speech

Voice Chat 1:n (Out Of Scope?)

2.3 Organisation

Kommunikation

Das Projekt IP5 Cloudbasiertes Praxisrufsystem wurde im FS21 gestartet. Die Organisation und Kommunikation des Projektes mussten dementsprechend für die Einschränkungen wegen Corona angepasst werden. Um sicherzustellen, dass die Kommunikation über die gesamte Projektdauer funktionieren kann, haben wir uns deshalb von Anfang an entschieden die Kommunikation über Remote- und Online Tools zu organisieren. Für Besprechungen und Planungen wurde Microsoft Teams gewählt. Die entsprechende Infrastruktur wurde von der FHNW zur Verfügung gestellt.

Dokumentation

Der Bericht wurde mit LateX und zusammen mit dem Quellcode verwaltet. Kurze Besprechungen, Notizen und interne Dokumentation erfolgten über ein geteiltes One Note Notizbuch.

Sämtliche Diagramme, Mockups und Skizzen wurden direkt in den Tools verwaltet, die zur Erstellung gebraucht wurden. Zum Schluss wurden alle für den Bericht relevanten Darstellungen exportiert und in den Bericht integriert.

Quellcodeverwaltung

Sämtlicher Quellcode der im Rahmen des Projektes entsteht, wurde mit Git verwaltet. Der Quellcode ist für Berechtigte unter dem Projekt IP5-Cloudbasiertes-Praxisrufsystem auf github.com einsehbar. (Referenz https://github.com/IP5-Cloudbasiertes-Praxisrufsystem). Berechtigungen können bei Joshua Villing oder Kevin Zellweger angefordert werden.

- IP5-praxis-mobile-client
- IP5-praxis-cloud-service
- IP5-praxis-admin-ui
- IP5-praxis-documentation

Tools und Werkzeuge

- draw.io
- moqus.com
- Visual Studio Code
- IntelliJ
- Git
- github.com

3 Anforderungen

Die im Rahmen des Projektes umzusetzenden Anforderungen wurden während des Projektes iterativ zusammen mit dem Kunden erarbeitet. Alle Anforderungen werden zuerst aus Fachlicher sicht mit User Stories festgehalten, die ein konkretes Bedürfnis der Benutzer beschreiben. Weiter werden User Stories aus Sicht des Kunden festhalten, welche Rahmenbedingungen und Bedürfnisse des Auftraggebers festhalten. Aufgrund der User Stories werden anschliessend Features definiert, welche konkrete Szenarien und die erwarteten Ergebnisse an definieren.

3.1 User Stories

Praxismitarbeiter

Id	Anforderung	Feature
U01	Als Praxismitarbeiter möchte ich Benachrichtigungen versenden können, damit ich andere Mitarbeiter über Probleme und Anfragen informieren kann.	F01
U02	Als Praxismitarbeiter möchte ich Benachrichtigungen empfangen können, damit ich auf Probleme und Anfragen anderer Mitarbeiter reagieren kann.	F02
U03	Als Praxismitarbeiter möchte ich nur Benachrichtigungen sehen, die für mich relevant sind, damit ich meine Arbeit effizient gestalten kann.	F02
U04	Als Praxismitarbeiter möchte ich über empfangene Benachrichtigungen aufmerksam gemacht werden, damit ich keine Benachrichtigungen verpasse.	F04
U05	Als Praxismitarbeiter möchte ich sehen welche Benachrichtigungen ich verpasst habe, damit ich auf verpasste Benachrichtigungen reagieren kann.	F04
U06	Als Praxismitarbeiter möchte ich eine Rückmeldung erhalten, wenn eine Benachrichtigung nicht versendet werden kann, damit Benachrichtigungen nicht verloren gehen.	F03
U07	Als Praxismitarbeiter möchte ich auswählen können an welchem Gerät ich das Praxisrufsystem verwende und die dafür erstellte Konfiguration erhalten, damit das Praxisrufsystem optimal verwendet werden kann.	F05
U8	Als Praxismitarbeiter möchte ich einen physischen Knopf am Behandlungsstuhl haben damit ich notifikationen darüber versenden kann.	F07
U9	Als Praxismitarbeiter möchte ich, dass mir Benachrichtigungen vorgelesen werden, damit ich informiert werde, ohne meine Arbeit unterbrechen zu müssen.	F08
U10	Als Praxismitarbeiter möchte ich einen anderen Client Unterhaltungen führen können damit Fragen direkt geklärt werden können.	F09
U11	Als Praxismitarbeiter möchte ich Unterhaltungen mit mehreren anderen Clients gleichzeitig führen können damit komplexe Fragen direkt geklärt werden können.	F10

Praxisverantwortlicher

Id	Anforderung	Features
U12	Als Praxisverantwortlicher möchte ich mehrere Geräte verwalten können, damit das Praxisrufsystem in mehreren Zimmern gleichzeitig genutzt werden kann.	F0x
U13	Als Praxisverantwortlicher möchte ich definieren können welches Gerät, welche Anfragen versenden kann, damit jedes Gerät Verwendungsort optimiert ist.	F0x
U14	Als Praxisverantwortlicher möchte ich die Konfiguration des Praxisrufsystems zentral verwalten können, damit das Praxisrufsystem für die Anwender optimiert werden kann.	F0x
U15	Als Praxisverantwortlicher möchte ich definieren können, welche Geräte mit welchen anderen Geräten telefonieren können damit meinen Mitarbeitern das Arbeiten erleichtere.	F0x
U16	Als Praxisverantwortlicher möchte ich definieren können, welche Benachrichtigung über einen physischen Knopf am Behandlungsstuhl versendet wird damit der Knopf für den Mitarbeiter optimiert ist.	F0x

Auftraggeber

Id	Anforderung	Features
T01	Als Auftraggeber möchte ich, dass das Praxisrufsystem über IPads bedient werden kann, damit ich von bestehender Infrastruktur profitieren kann.	F0x
T02	Als Auftraggeber möchte ich, dass das Praxisrufsystem über Android Tablets bedient werden kann, damit es in Zukunft für eine weitere Zielgruppe verwendet werden kann.	F0x
T03	Als Auftraggeber möchte ich, dass die Codebasis für das Praxisrufsystem für Android und IOS verwendet werden kann, damit ich die Weiterentwicklung optimieren kann.	F0x
T04	Als Auftraggeber möchte ich, dass wo möglich der Betrieb von Serverseitigen Dienstleistungen über AWS betrieben wird, damit ich von bestehender Infrastruktur und Erfahrung profitieren kann.	F0x

3.2 Features

F01 - Benachrichtigungen Versenden

Scenario: Benachrichtigung versenden

Given: Benutzer ist vollständig angemeldet
And: Mindestens 1 Empfänger ist konfiguriert
When: Praxismitarbeiter tippt auf einen Benachrichtigungs-Button
Then: Benachrichtigung wird an den zentralen Cloud Service gesendet
And: Benachrichtitung wird an alle Mobile Clients versendet
die sich für diese Benachrichtigung subscribed haben weitergeleitet
And: Praxismitarbeiter erhält optische Rückmeldung, dass Benachrichtigung versendet wurde

Scenario: Keine Empfänger konfiguriert

Given: Benutzer ist vollständig angemeldet
And: Kein Empfänger ist konfiguriert
When: Praxismitarbeiter tippt auf einen Benachrichtigungs-Button
Then: Benachrichtigung wird an den zentralen Cloud Service gesendet
And: Benachrichtitung wird nicht weitergeleitet

F02 - Benachrichtigungen Empfangen

Scenario: Empfangen

Given: Eine Benachrichtigung wurde von Mobile Client versendet
When: Cloud Service Notification an Empfänger Mobile Client weiterleitet
Then: Wird die Benachrichtigung vom Empfänger Mobile Client empfangen
And: In einer Übersicht für empfangene Benachrichtigung angezeigt.

F03 - Fehlgeschlagene Benachrichtigungen

Scenario: Fehler Rückmeldung

Given: Eine Benachrichtigung wurde von Mobile Client versendet
When: Weiterleitung von Cloud Service Notification an Empfänger schlägt auf Service Seite fehl
Then: Der Praxismitarbeiter wird über den Fehler Informiert
And: Der Praxismitarbeiter hat die Möglichkeit die Fehlgeschlagenen Benachrichtigungen zu wiederholen

Scenario: Confirm Retry

Given: Benachrichtigung ist fehlgeschlagen
And: Dialog zum wiederholen wird angezeigt
When: Praxismitarbeiter bestätigt, dass wiederholt werden soll
Then: Der Cloudservice versucht erneut, die fehlgeschlagenen zuzustellen

Scenario: Cancel Retry

Given: Benachrichtigung ist fehlgeschlagen
And: Dialog zum wiederholen wird angezeigt
When: Praxismitarbeiter klick, dass nicht wiederholt werden soll
Then: Werden die fehlgeschlagenen nicht wiederholt
And: Zurück zur Notificationsansicht

F04 - Über Benachrichtigungen Notifizieren

Scenario: Foreground

Given: Mobile Client ist geöffnet
When: Eine Benachrichtigung wird vom Mobile Client emfpangen
Then: Ein Audio Signal erklingt

Scenario: Background

Given: Mobile Client läuft im Hintergrund
When: Eine Benachrichtigung wird vom Mobile Client empfangen
Then: Ein Audio Signal erklingt
And: Eine Push Benachrichtigung wird angezeigt

Scenario: Nicht Quittiert

Given: Mobile Client ist geöffnet
And: Eine Benachrichtigung wurde empfangen
When: Benachrichtigung wird nicht quittiert
Then: Ein Audio Signal erklingt
And: Das Audio Signal wiederholt sich alle 30 Sekunden, bis die Benachrichtigung Quittiert wurde.

F05 - Login Mobile Client

Scenario: Startbildschirm wenn nicht angemeldet

Given: Mobile Client is geöffnet
When: Benutzer ist nicht angemeldet
Then: Benutzer wird zum Login aufgefordert

Scenario: Startbildschirm wenn angemeldet

Given: Mobile Client is geöffnet
When: Benutzer ist angemeldet
Then: Konfiguration die der Benutzer zuletzt gewählt hat wird angezeigt
And: Benachrichtigungs Buttons gemäss Konfiguration werden angezeigt.

Scenario: Anmelden korrekt

Given: Benutzer ist nicht angemeldet
And: Login Screen wird angezeigt
And: Für den Benutzer sind gültige Konfigurationen erfasst
When: Benutzer meldet sich mit korrekten Daten an
Then: Benutzer wird auf nächste Seite geleitet und kann dort die Konfiguration auswählen, die er Benutzen möcht

Scenario: Anmelden falsch

Given: Benutzer ist nicht angemeldet
And: Login Screen wird angezeigt
When: Benutzer meldet sich mit falschen Daten an
Then: Fehlermeldung
And: Benutzer wird nicht weitergeleitet

Scenario: Konfiguration Wählen

Given: Benutzer hat sich korrekt angemeldet
And: Konfiguration Auswählen Screen wird angezeigt
When: Der Benutzer wählt die gewünschte Konfiguration
Then: Der Benutzer wird weitergeleitet
And: Die gewählte Konfiguration wird geladen
And: Benachrichtigungs Buttons gemäss Konfiguration werden angezeigt.

Scenario: Logout

Given: Benutzer ist angemeldet
When: Benutzer klickt logout
Then: Benutzer wird zur Login Seite weitergeleitet

F06 - Konfigurationsverwaltung

Scenario: Login

Given: Benutzer ist nicht angemeldet
And: Admin UI Login Screen wird angezeigt
When: Admin meldet sich mit korrekten Daten an
Then: Admin wird auf Übersichtsseite weitergeleitet

Scenario: Anmelden falsch

Given: Benutzer ist nicht angemeldet
And: Admin UI Login Screen wird angezeigt
When: Admin meldet sich mit falschen Daten an
Then: Fehlermeldung wird angezeigt
And: Admin wird nicht weitergeleitet.

Scenario: Konfiguration verwalten

Given: Admin ist angemeldet
When: Admin UI wird aufgerufen
Then: Alle existierenden Konfigurationen werden angezeigt
And: Neue Konfigurationen können erstellt werden
And: Bestehende Konfigurationen können verändert werden
And: Bestehende Konfigurationen können gelöscht werden

F07 - Integration Behandlungsstuhl

Dieses Feature fällt ausserhalb des Projekt Scopes. Dementsprechend wurden dafür noch keine Szenarien definiert.

F08 - Text To Speech

Dieses Feature fällt ausserhalb des Projekt Scopes. Dementsprechend wurden dafür noch keine Szenarien definiert.

F09 - Direkte Anrufe

Dieses Feature fällt ausserhalb des Projekt Scopes. Dementsprechend wurden dafür noch keine Szenarien definiert.

F10 - Gruppen Anrufe

Dieses Feature fällt ausserhalb des Projekt Scopes. Dementsprechend wurden dafür noch keine Szenarien definiert.

4 Konzept

4.1 Systemarchitektur

Abgrenzung

Dieses Kapitel gibt einen Übersicht über die Systemarchitektur als ganzes. Die Architektur beschränkt sich dabei auf die Anforderungen die innerhalb des Projektrahmens umgesetzt wurden. Komponenten für Teile die Out Of Scope gefallen sind, werden hier nicht behandelt.

Übersicht

Das cloudbasierte Praxisrufsystem wird in vier Komponenten unterteilt. Im Zentrum steht eine cloudbasierte Applikation (Cloud Service) welche es ermöglicht Konfigurationen persistent zu verwalten und das Versenden von Benachrichtigungen anhand dieser Konfigurationen koordiniert. Der Cloud Service benutzt einen externen Messaging Service zum Versenden von Benachrichtigungen. Dabei ist der Messaging Service lediglich für die Zustellung von Benachrichtigungen verantwortlich. Zur Verwaltung der Konfigurationen wird ein Web Frontend (Admin UI) erstellt. Dieses bietet einem Administrator die Möglichkeit Konfigurationen aus dem Cloud Service zu lesen, erstellen, bearbeiten und löschen. Die Konfigurationen die über Admin UI und Cloud Service erstellt wurden, werden schliesslich von einem Mobile Client. Mit dem Mobile Client kann der Benutzer Benachrichtigungen an andere Mobile Clients senden. Welche Benachrichtigungen ein Mobile Client senden kann und an wen diese Benachrichtungen zugstellt werden, wird anhand der Konfiguration aus dem Cloud Service bestimmt.

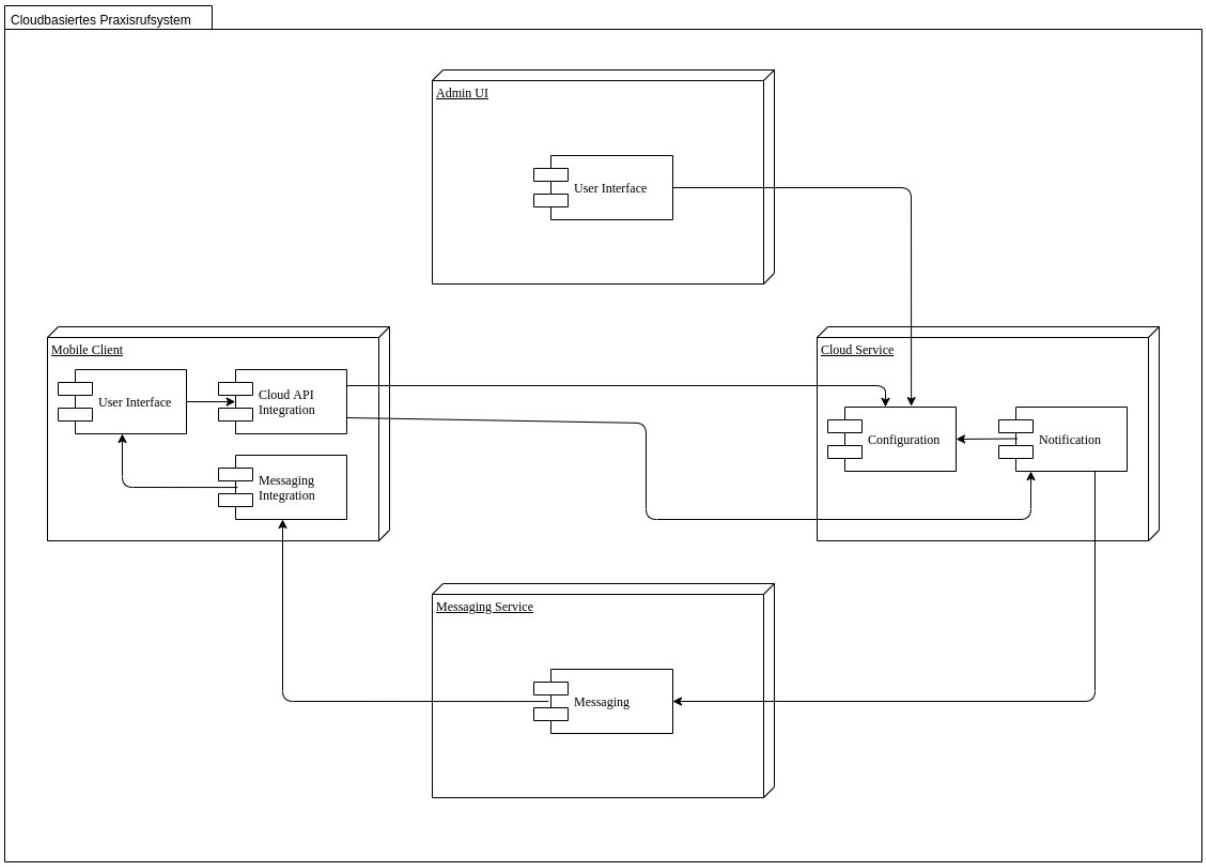

Abbildung 4.1: System

Mobile Client

Damit der Praxismitarbeiter wie in den Anforderungen U01 und U02 beschrieben Benachrichtigungen versenden und empfangen kann, benötigt das System einen Client, welcher dem Praxismitarbeiter diese Oprationen ermöglicht. Die technischen Anforderungen T01, T02 und T03 setzen voraus, dass dieser Client über die Mobil Geräte (IPads und Android Tablets) bedient werden kann.

- T01
- T02
- T03
- U01
- U02
- U03
- U04
- U05
- U06
- U07

Cloud Service

Der Cloud Service ermöglicht Konfigurationen persistent zu verwalten und das Versenden von Benachrichtigungen anhand dieser Konfigurationen koordiniert. Der Cloud Service benutzt einen externen Messaging Service zum Versenden von Benachrichtigungen.

- U01
- U02
- U03
- U06
- U07

Messaging Service

Der Messaging Service ist für die Zustellung von Benachrichtigungen verantwortlich.

- U01
- U02
- U03
- U04
- U06

Admin UI

Damit der Praxisverantwortliche die Konfiguration des Praxisrufsystems verwalten kann ist es notwendig, dass alle Lese und Schreib Zugriffe auf die Konfiguration über eine Zentrale Stelle gehen, auf die der Praxisverantwortliche Zugriff hat.

- U12
- U13
- U14
- T04

4.2 Mobile Client

4.2.1 Framework Grundlagen

NativeScript bietet eine Abstraktion zu den nativen Plattformen Android und IOS. Die jeweilige NativeScript Runtime erlaubt es in Javascript (oder einem entsprechenden Application Framework) Code zu schreiben, welcher direkt für die entsprechende native Umgebung kompiliert wird **ns-core-overview**.

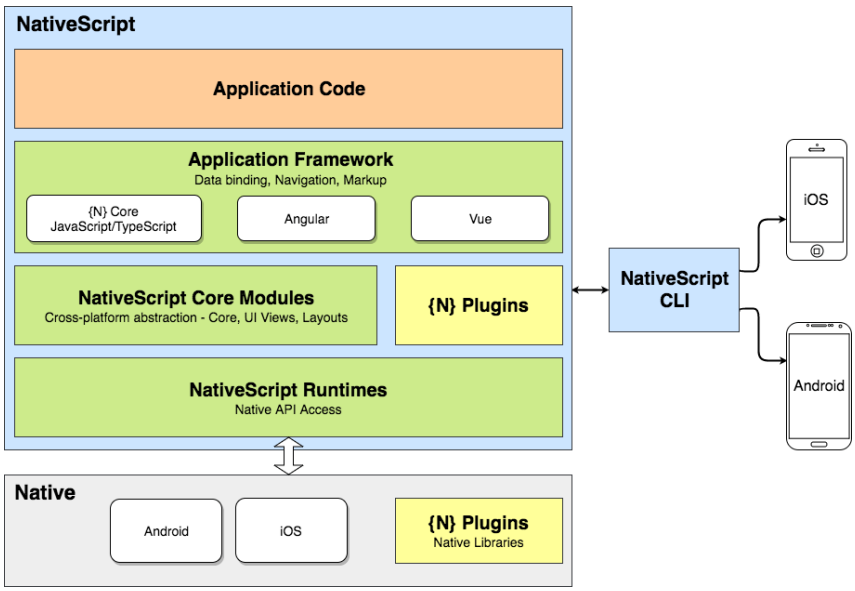

Abbildung 4.2: NativeScript-Overview
©OpenJS Foundation

Die Runtime agiert als Proxy zwischen Javascript und dem jeweiligen Ökosystem. Im Falle von IOS bedeutet dies u.A. das für alle Objective-C types ein JavaScript Prototype angeboten wird. Dies ermöglicht es direkt mit nativen Objekten zu interagieren. Im Umkehrschluss findet eine Typenkonversion via Marshalling Service statt**ns-ios-runtime**.

4.2.2 Anwendung

Wir verwenden NativeScript Core als Framework des Mobile-Clients. In Kapitel *Mobile Client Evaluation* gehen wir auf die weiteren verfügbaren Frameworks ein und erläutern, weshalb wir uns gegen sie entschieden haben.

Die Client-Applikation ist in Module unterteilt. Ein Modul wird aus folgenden Komponenten definiert:

- UI-Markup: Statische Darstellung in XML
- Backend: Verhalten und Dynamisierung in Javascript
- Styling: Layout und Styles in CSS

Ein minimales Modul kann alleine aus einer XML-Datei bestehen. Die optionalen Javascript und CSS Dateien müssen denselben Namen haben wie die XML Datei, um vom Framework korrekt verknüpft zu werden. Dateien mit anderen Namen werden grundsätzlich vom Framework ignoriert. Natürlich steht es Frei dennoch solche Dateien anzulegen und deren Funktionen zu verwenden z. B. als *Services* oder als *Code-Behind Komponenten*.

Zur Veranschaulichung der möglichen Interaktionen gehen wir auf die relevanten Aspekte des Home-Screen Modules ein.

Page Module

```
app
└── home
    ├── home-page.xml
    ├── home-page.css
    ├── home-page.js
    └── home-model.js
```

home-page.xml deklariert die umgebenden Komponenten. Diese Komponenten stellen je nach Typ diverse Properties und Events zur Verfügung. Properties können entweder statisch befüllt oder aus dem Binding-Context geladen werden. Den Events können Callback-Functions zugewiesen werden. Es stehen alle Funktionen zur Verfügung, welche im Backendscript *home-page.js* exportiert werden.

```
1  <Page loaded="onPageLoaded" navigatingTo="onNavigatingTo"
2        xmlns="http://schemas.nativescript.org/tns.xsd"
3        xmlns:profile="components/profile">
4      <StackLayout id="profile" class="home-body">
5          <profile:profile-container/>
6          <StackLayout class="btn-box">
7              <Label textAlignment="center" text="Meldungen" class="section_title"/>
8              <GridLayout loaded="onGridLoaded" columns="auto, auto, auto" rows="auto
       auto auto" horizontalAlignment="center">
9
10             </GridLayout>
11         </StackLayout>
12     </StackLayout>
13 </Page>
```

Listing 1: home-page.xml

Der Binding-Context ist ein JavaScript Objekt welches exklusiv im Page-Context zur Verfügung steht. Es ist allgemein Best-Practice dieses Objekt in einem eigenen Model zu verwalten. Das eigentliche Binding wird vom Backendscript *home-page.js* (Zeilen 8–11) während des ersten Ladens der Seite durchgeführt.

```
1  import {fromObject, Observable, ObservableArray} from '@nativescript/core'
2
3  export function HomeItemsViewModel() {
4      const viewModel = new Observable();
5      viewModel.notificationConfigurations = new ObservableArray([])
6
7      return viewModel
8  }
```

Listing 2: home-model.js

Das Backendscript ist für das dynamische Verhalten der Seite verantwortlich. Hier können die Interaktionen der Benutzer beliebig verarbeitet und der Binding-Context bei Bedarf verwaltet werden.

```
1  import {ApplicationSettings, Button, GridLayout} from "@nativescript/core";
2  import {MessageTrigger} from "~/components/message-trigger/message-trigger";
3  import { HomeItemsViewModel } from './home-model'
4  import {getClientConfiguration} from "~/services/configuration-api";
5  import {showError} from "~/error-dialog/error-dialog";
6
7  const model = new HomeItemsViewModel();
8  export function onPageLoaded(args){
9      const mainComponent = args.object;
10     mainComponent.bindingContext = model;
11 }
```

```
12
13 export function onGridLoaded(args){
14     const grid = args.object;
15     getClientConfiguration(ApplicationSettings.getString("clientId"))
16         .then(result => buildMessageUI(result, grid))
17         .catch((e) => showError("Loading Client Configuration failed", e, "OK"));
18 }
19
20 function buildMessageUI(clientConfiguration, grid){
21     let rowCounter, columnCounter, rowIdx, columnIdx = 0;
22     clientConfiguration.map((conf) => {
23         const messageComp = new MessageTrigger(conf.title, conf.id);
24         grid.addChild(messageComp);
25         GridLayout.setRow(messageComp,rowIdx);
26         GridLayout.setColumn(messageComp,columnIdx);
27         rowCounter++
28         columnCounter++
29         if(columnCounter > 2){
30             columnIdx = 0;
31             columnCounter = 0;
32         } else {
33             columnIdx++;
34         }
35         if(rowCounter > 3){
36             rowIdx++;
37             rowCounter = 0;
38         }else {
39             rowCounter++
40         }
41     })
42 }
```

Listing 3: home-page.js

Code-Behind Komponenten

Code-Behind Komponenten bieten die Möglichkeit zur Laufzeit dynamisch Grafikelemente dem UI hinzuzufügen. Komponenten die das Framework bereits zur Verfügung stellt können direkt mit `new <Component>()` instanziiert werden. Bei Bedarf können diese Komponenten auch erweitert und mit zusätzlicher Funktionalität ausgestattet werden.

Da der Home-Screen dynamisch in Abhängigkeit der Client-Configuration erstellt werden muss, werden eigene `MessageTrigger` Komponenten verwendet.

Services

In Services werden diejenigen Funktionen ausgelagert, welche nicht direkt im Zusammenhang mit der grafischen Representation stehen. So z. B. die REST-Calls zur API.

4.2.3 Architektur

4.2.4 User Interface

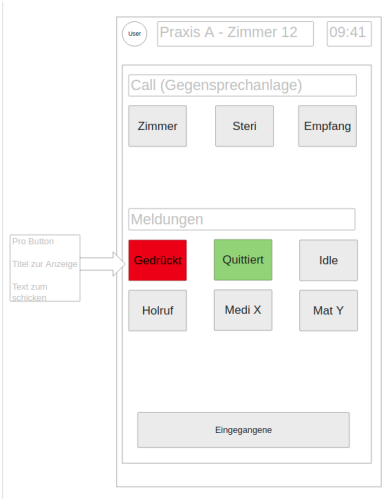

Abbildung 4.3: HomeScreen Mockup

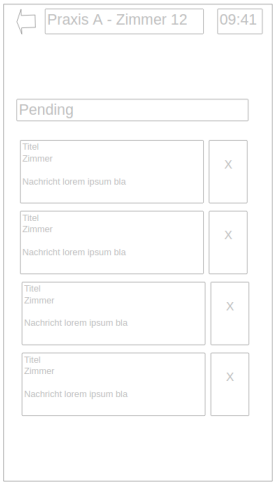

Abbildung 4.4: Inbox Mockup

4.3 Cloud Service

4.3.1 Architektur

Es gibt deren Domänen 2. Configuration und Notification.

So quasi als ob man 2 Microservices haben kann. Aber wär halt doof das für den stand jetzt schon so zu trennen, deshalb vorerst mal erst ein einzelnes.

4.3.2 Domänenmodell

Für die beiden Domänen gibt es natürlich auch so n paar Diagramme. Die gibts jetzt hier:

Domäne Configuration

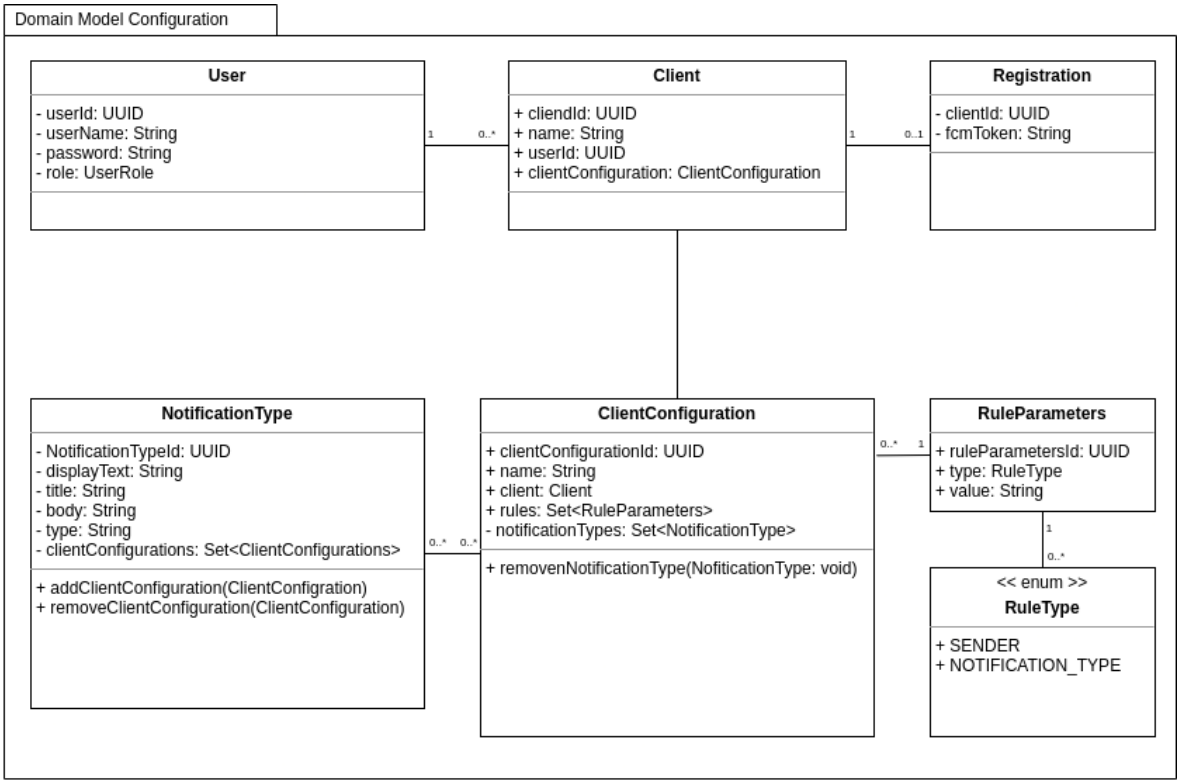

Abbildung 4.5: Domänenmodell Configuration

Domäne Notification

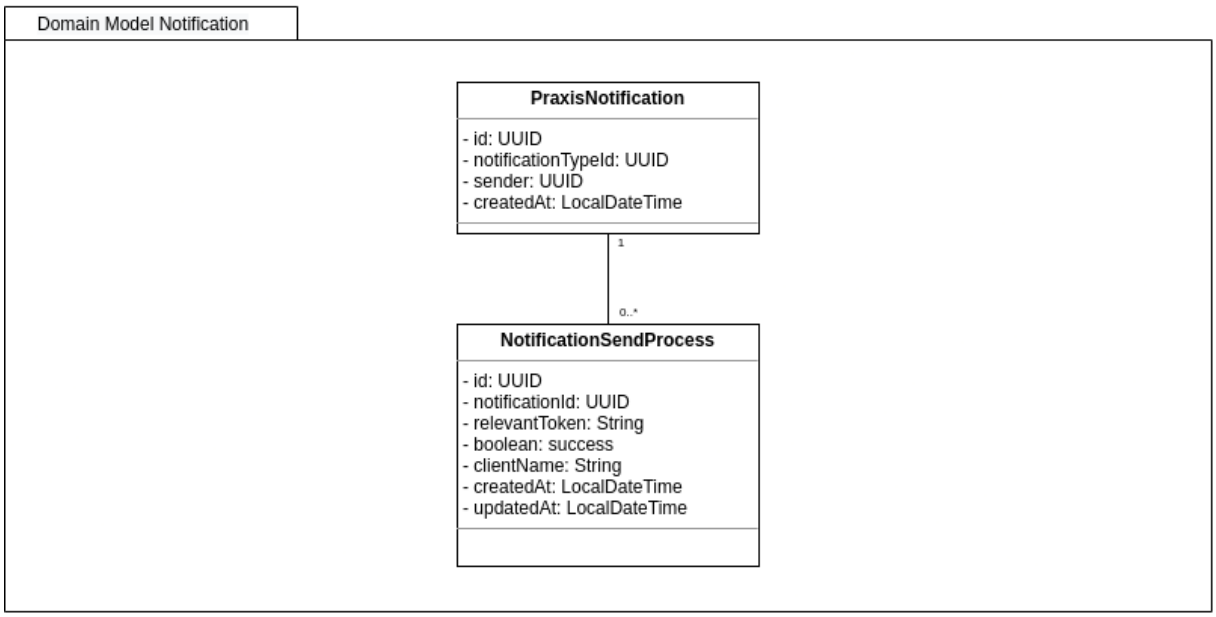

Abbildung 4.6: Domänenmodell Notification

Rules Engine

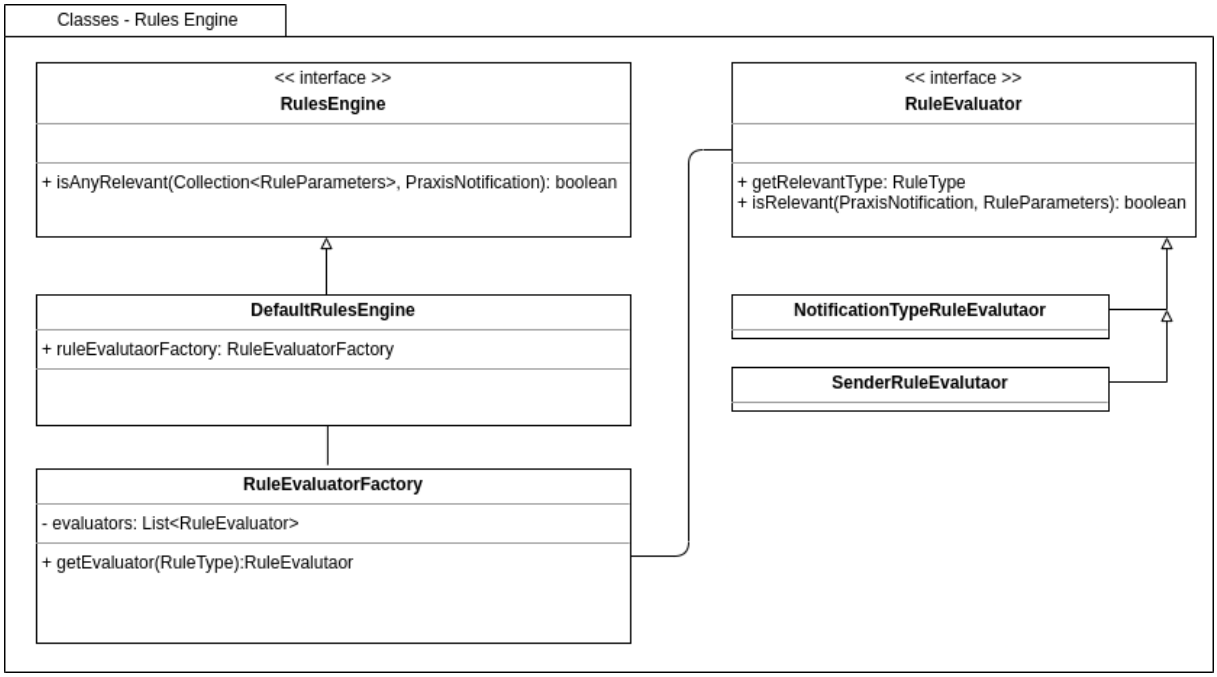

Abbildung 4.7: Klassendiagramm Rules Engine

4.3.3 API

S gibt da n paar controller und die brauchen ein paar services.

4.3.4 Laufzeitmodell

4.4 Admin UI

4.5 Proof Of Concept

Um sicherzustellen, dass die Anforderungen an das System mit den gewählten Technologien umgesetzt werden können, wird zunächst ein Proof Of Concept implementiert. Dieser Proof Of Concept hat einen deutlich kleineren Funktionsumfang als das Endprodukt. Im Wesentlichen muss der Proof Of Concept beweisen, dass es möglich ist mit den gewählten Technologien Benachrichtigungen zu Versenden und zu empfangen.

4.5.1 Funktionale Anforderungen

Um dies zu ermöglichen werden die folgenden Features in eingeschränktem Umfang umgesetzt.

F01 - Benachrichtigungen Versenden

Mit dem Proof Of Concept muss es möglich sein, Benachrichtigungen von einem Client zu versenden. Dementsprechend wird das Szenario ”Benachrichtigung versendenmit den folgenden Einschränkungen umgesetzt:

- Es erfolgt kein Login und keine Authentifizierung.
- Es gibt nur einen vordefinierten Button.
- Es wird immer dieselbe vordefinierte Benachrichtigung versendet.
- Es werden keine Empfänger konfiguriert, die Benachrichtigung wird zurück an den Absender versendet.

F02 - Benachrichtigungen empfangen

Mit dem Proof Of Concept muss es möglich sein, Benachrichtigungen mit einem Client zu empfangen. Dementsprechend wird das Szenario ”Benachrichtigung empfangenmit den folgenden Einschränkungen umgesetzt:

- Es wird keine Liste von Benachrichtigungen geführt.
- Die Benachrichtigung wird in einem einfachen Textfeld im Mobile Client angezeigt.

F04 - Über Benachrichtigungen Notifizieren

Mit dem Proof Of Concept muss es möglich sein, über den Einfang von Benachrichtigungen notifiziert zu werden. Dementsprechend werden die Szenarien Foregroundünd ”Backgroundmit den folgenden Einschränkungen umgesetzt:

- Die Notifizierung erfolgt ohne Audio Signal.

4.5.2 Technische Anforderungen

Damit der Proof Of Concept aussagekräftig ist, müssen die folgenden technischen Anforderungen umgesetzt werden:

T01 - IPad Client

Der für den Proof Of Concept umgesetzte Client muss auf einem IPad funktionieren und alle Anforderungen die an den Proof Of Concept gestellt werden erfüllen. Kommunikation mit Cloud Service muss funktionieren. Kommunikation mit Messaging Service muss funktionieren.

T04 - AWS Platform

Der Cloud Service muss auf AWS deployed werden. Die Kommunikation zwischen Mobile Client und Cloud Service muss funktionieren. Die Kommunikation mit Messaging Service muss funktionieren.

4.5.3 Laufzeitsicht

Im Wesentlichen muss der Proof Of Concept beweisen, dass es möglich ist mit den gewählten Technologien Benachrichtigungen zu Versenden und zu Empfangen.

Der Benutzer muss den Mobile Client auf dem IPad öffnen können. Aus dem Mobile Client muss der Benutzer über einen Butten eine Benachrichtigung versenden können. Das Versenden dieser Benachrichtigung erfolgt an den Cloud Service. Im Rahmen des Proof Of Concept wird eine Benachrichtigung immer an den Sender zurückgesendet. Dabei ist aber wichtig, dass die Benachrichtigung nicht direkt als Antwort auf die Versenden-Anfrage geschickt wird. Stattdessen muss das Versenden der Benachrichtigung aus dem Cloud Service über den Message Service erfolgen. .

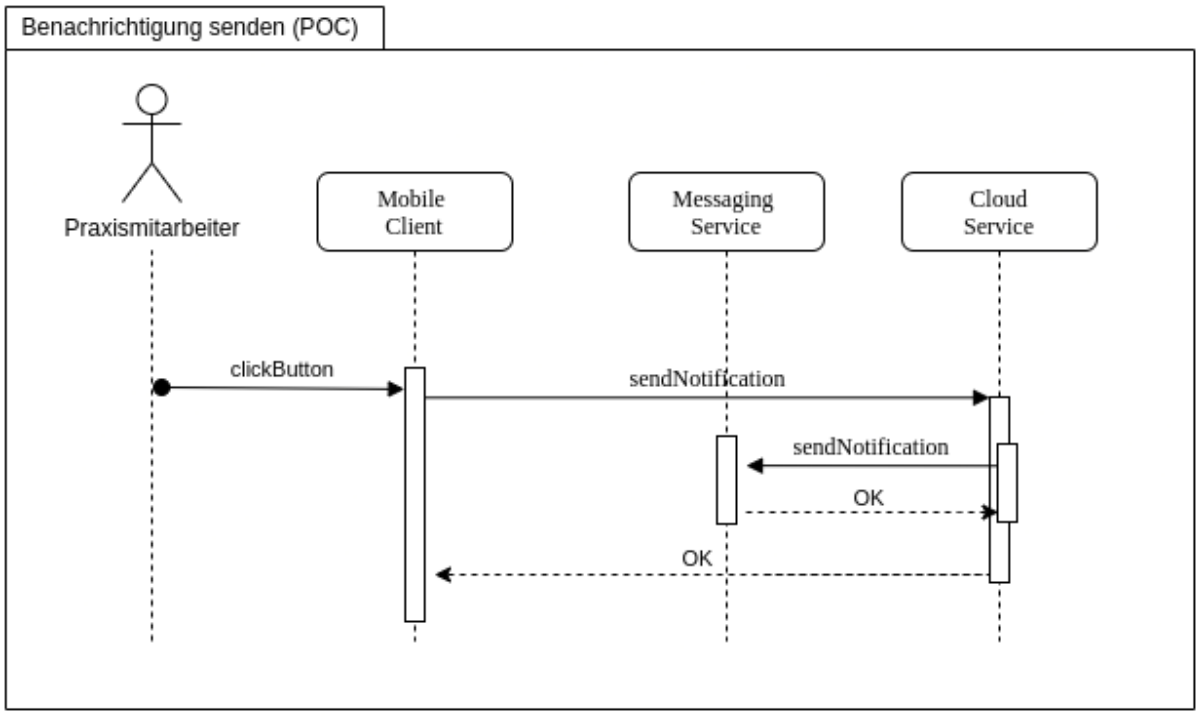

Abbildung 4.8: Proof Of Concept - Benachrichtigung versenden

Wurde eine Benachrichtigung über den Message Service an den Mobile Client versendet, muss diese vom Mobile Client empfangen werden. Als Reaktion auf den Empfang der Benachrichtigung, muss der Inhalt der Benachrichtigung im Mobile Client angezeigt werden. Zudem muss eine Push Benachrichtigung auf dem Host Gerät erfolgen.

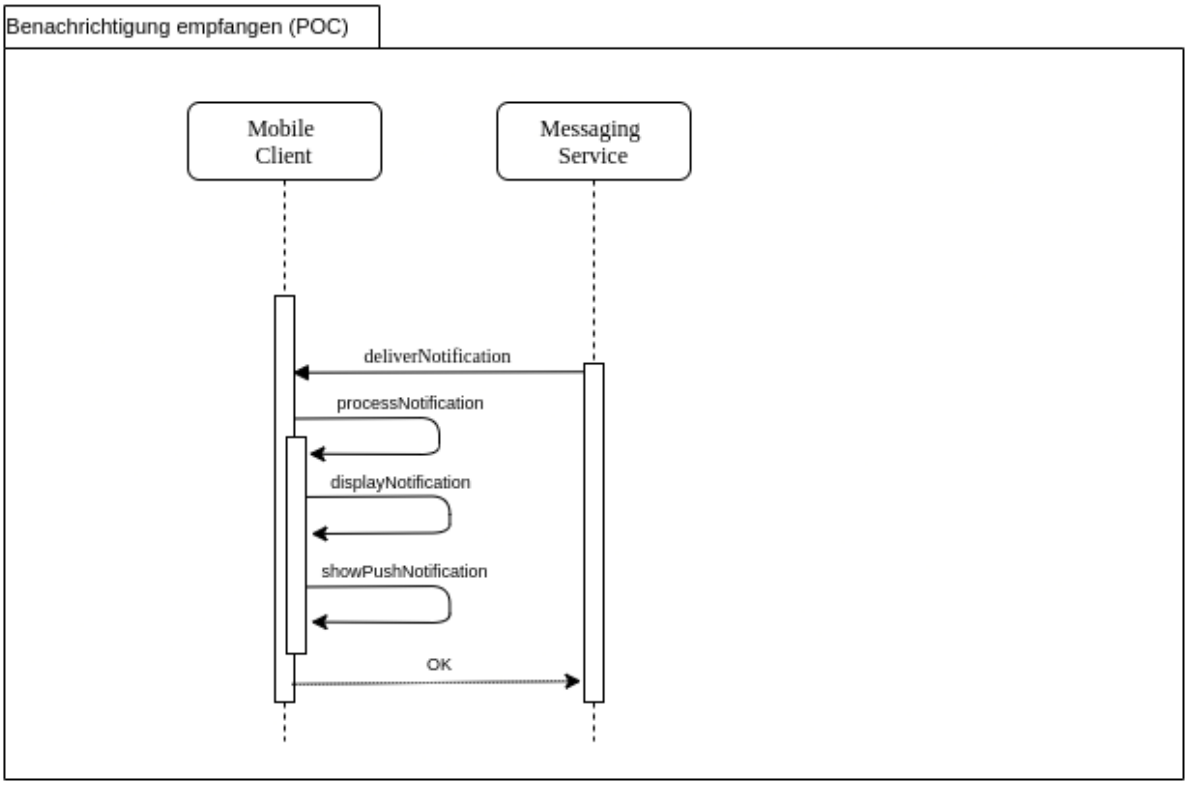

Abbildung 4.9: Proof Of Concept - Benachrichtigung empfangen

5 Evaluation Technologien

5.1 Mobile Client Evaluation

https://kotlinlang.org/lp/mobile/

+Jet Brains Infrastructure +We like Kotlin

-IoS Env. Needed to develop for Apple -Still has to develop separate API und UI Modules for Platforms

https://web.dev/progressive-web-apps/

+No need of Native Codebase +Perfect for Android -Eventually drawbacks because no entire API Access -PWAs on IOS suck

https://cordova.apache.org/

+ Popular Framework + Tons of plugins to access apis

-Still need to have a Mac for IoS development -Not a truly native app -¿ API Issues

https://nativescript.org/

+Provides a Workaround for nasty X-tools +Claims to be truly Native -Do we really trust it? (sorta new and passion project of a few people)

https://flutter.dev

-Why do you hate me?

SSimply Write Everything twice”

+Would definitely work

-Do most things twice -We don’t have time for that -Kunde wünscht ausdrücklich nur eine Codebasis für beide Clients.

https://stackshare.io/stackups/apache-cordova-vs-nativescript

https://nativescript.org/blog/build-nativescript-apps-remotely-from-windows-or-l

5.2 Cloud Service

https://aws.amazon.com/

https://spring.io/projects/spring-boot

Konfig der Clients könnte sich als No-SQL anbieten.

Config muss nur gelesen und an den Client geschickt oder abgespeichert werden

https://www.mongodb.com/

5.3 Betrieb und Platform

AWS ist MUSS

6 Umsetzung

6.1 Authentifizierung

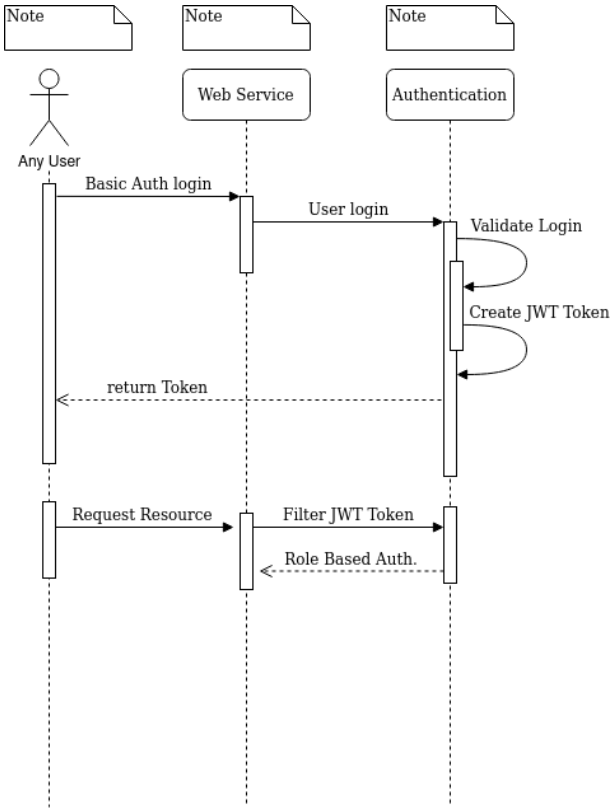

Abbildung 6.1: Authentifizierung-Sequenz

7 Schluss

Abbildungsverzeichnis

8 Anhang

8.1 Benutzerhandbuch

8.2 Betriebshandbuch

8.3 Entwicklerdokumentation

8.4 Ehrlichkeitserklärung