Entwicklung interaktiver Systeme

-

Rapid Prototyp

Moritz Müller, Johannes Kimmeyer

4. November 2016

Inhaltsverzeichnis

1 Einleitung

In diesem Dokument finden Sie alle für den Rapid Prototyp wichtigen Dokumente. Der Rapid Prototyp zeigt die Realisierbarkeit von Alleinstellungsmerkmalen und verteilter Anwendungslogik und umfasst die Durchführung von technischen Proof of Concepts.

2 Alleinstellungsmerkmale

Mit den in der Marktrecherche vorgestellten Anwendungen lassen sich jeweils Teilaspekte des Nutzungsproblems lösen. Allerdings fehlt eine Anwendung, die die wesentlichen Funktionen vereint und gleichzeitig eine Kommunikation mit der Fachhandlung ermöglicht. Nachdem man eine Wasserprobe zu der Fachhandlung gebracht hat, wird diese dort ausgewertet und über die Anwendung können die Ergebnisse direkt an den Kunden übermittelt werden. Dabei werden sowohl der Privatperson als auch der Fachhandlung Tools wie z.B. die Berechnung von Düngemittel angeboten. Außerdem hat die Fachhandlung sofort Einblick in die vom Kunden eingetragenen Aquarium Daten, wie zum Beispiel Größe und Inhalt (Pflanzen, Lebewesen), und kann dem Kunden aufgrund dessen passende Vorschläge bzgl. Erweiterungen geben.

3 Verteilte Anwendungslogik

Nach der Wasseranalyse beim Fachmann kann dieser den täglichen Verbrauch von Nährstoffen berechnen. Ebenso trägt er andere Wasserwerte ein die von Bedeutung sind. Der Kunde erhält diese Werte anschließend und kann mit diesen Werten den CO2- Gehalt des Wassers berechnen oder auch das benötigte Verhältnis bei einem Wasserwechsel. Die Berechnung des täglich aktuellen Nährstoffes soll ebenfalls beim Client geschehen und von ihm zum Fachhandel zu schicken sein, damit dieser ihn optimal beraten kann.

4 Projektrisiken

Beschaffung der Formeln für einzelne Berechnungen

Die Berechnungen, die wir in unserem System benutzen wollen, sind nicht gerade einfach. Diese müssen erst recherchiert werden. Sollten wir nicht an die Formeln für einzelne Berechnungen gelangen, fehlt ein wichtiger Teil unserer Anwendungslogik. Dies kann dazu führen, dass unsere Anwendungslogik nicht mehr ausreicht und wir im schlimmsten Fall das Projekt beenden müssen. Um das Risiko zu minimieren sollten wir uns schon frühzeitig um die Formeln kümmern und uns Alternativen überlegen.

Umsetzung der Formeln (Programmierung)

Sollten wir alle Formeln bekommen haben gilt es diese in Java umzusetzen. Dies könnte aufgrund von mangelnden Programmierkenntnissen oder auch durch Faktoren wie zum Beispiel die Unwissenheit über den natürlichen Nährstoffverbrauch im Aquarium scheitern und das könnte wiederum zu einem Abbruch des Projekts führen, da uns in dem Fall einiges an Anwendungslogik

verloren geht. Um dies frühzeitig zu erkennen ist hier ein Proof of Concept sinnvoll.

Umgang mit Firebase Cloud Messaging

Da bis jetzt noch niemand von uns mit dem FCM gearbeitet hat, ist es nicht garantiert, dass wir unser Vorhaben damit umsetzen können. Sollten wir nicht in der Lage sein, die Kommunikation mit dem FCM umsetzen zu können, müssen wir darauf verzichten und eine alternative Möglichkeit für die Kommunikation zwischen den Clients suchen. Zur frühzeitigen Erkennung des Ereignisses ist auch hier ein Proof of Concept angebracht.

Ausfall eines Teammitglieds (z.B. durch Krankheit)

Sollte einer von uns beiden zum Beispiel durch Krankheit für mehrere Tage ausfallen, steht das Projekt auf der Kippe, da die Bearbeitung der Artefakte alleine um einiges länger dauert und wir somit unseren Zeitplan nicht mehr einhalten können.

Unterschätzung des Zeitaufwands

Das Einhalten der Fristen ist ein wichtiger Punkt, da das Projekt ansonsten scheitern könnte. Es ist also nötig, den Zeitaufwand von Anfang an richtig einzuschätzen und falls möglich noch ein bisschen Freiraum bis zum Abgabetermin einzuplanen, um Verzögerungen ausgleichen zu können.

Fehlendes MCI Wissen

Da wir beide noch keine MCI Prüfung absolviert haben, kann es sein, dass uns das Wissen bei der Bearbeitung der MCI-Artefakte fehlt. Um dies so gut wie möglich zu verhindern haben wir bereits am Anfang des Semesters begonnen, die MCI Materialien zu wiederholen.

5 Proof of Concept

Umgang mit Firebase Cloud Messaging

Um eine Kommunikation mit „Push-Notifications“ von der Fachhandlung zum Kunden realisieren zu können, benötigen wir das Firebase Cloud Messaging. Da bisher noch keiner von uns beiden Erfahrung damit hat, könnte es sein, dass die Umsetzung bzw. ‚dass was wir uns vorgenommen haben scheitert.

Exit: Die Kommunikation von der Fachhandlung zum Kunden mit Hilfe des FCM wurde erfolgreich implementiert

Fail: Die Implementierung konnte aufgrund von mangelndem Wissen nicht durchgeführt werden. Somit gilt das PoC als gescheitert. Dies führt dazu, dass die Fachhandlung nicht optimal mit dem Kunden kommunizieren kann, was wiederum dazu führen kann, dass das System weniger von Fachhandlungen und deren Kunden in Anspruch genommen wird.

Fallback: Falls die Implementierung mit dem Firebase Cloud Messaging nicht funktioniert, werden wir uns nach Alternativen umschauen müssen. Da diese aber vermutlich genauso kompliziert sein werden, steht ein Fallback hier leider nicht zur Verfügung.

Beschaffung der Formeln für einzelne Berechnungen

Die Berechnungen, die wir in unserem System benutzen wollen, sind nicht gerade einfach. Diese müssen erst recherchiert werden.

Exit: Die Formeln für die geplanten Berechnungen wurden gefunden

Fail: Wir konnten nicht alle gewünschten Formeln ermitteln. Somit fehlt uns ein wichtiger Teil der Anwendungslogik und das PoC gilt als gescheitert. Der Mangel an Anwendungslogik kann dazu führen, dass das Projekt nicht mehr die geforderten Erwartungen erfüllt und somit beendet werden muss.

Fallback: Die für uns wichtigste Berechnung, ist die Berechnung des durchschnittlichen Nährwerteverbrauchs eines Aquariums anhand der Einflüsse ohne die Nährstoffveränderungen. Falls wir hierzu keine genaue Formel finden, werden wir eine Umfrage mit Teilnehmern, welche ein Aquarium besitzen,

durchführen, um eine grobe Annäherung anhand der einzelnen Einflüsse zu finden.

Umsetzung der Formeln (Programmierung)

Sollten wir alle Formeln bekommen haben, gilt es diese in Java umzusetzen. Dies könnte aufgrund von mangelnden Programmierkenntnissen oder auch durch Faktoren wie zum Beispiel die Unwissenheit über den natürlichen Nährstoffverbrauch im Aquarium scheitern.

Exit: Die Formeln konnten erfolgreich in Java umgesetzt werden

Fail: Nicht alle Formeln konnten erfolgreich umgesetzt werden. Dadurch können die funktionalen Ziele nicht mehr erreicht werden und das PoC gilt als gescheitert. Dies kann dazu führen, dass die Anwendung nicht mehr Interessant genug für mögliche Kunden ist.

Fallback: Bei der Programmierung kann man immer noch das Internet zur Hilfe ziehen, falls uns etwas nach unglaublich vielen Versuchen immer noch nicht gelingt, werden wir uns externe Hilfe dazuziehen.

6 Proof of Concept Durchführung

Umgang mit Firebase Cloud Messaging

Die Kommunikation von der Fachhandlung zum Kunden mit Hilfe des Firebase Cloud Messaging war eines unserer Proof of Concepts, da wir beide vorher noch keine Erfahrung damit gemacht hatten. Deshalb haben wir diese Kommunikation bereits in unserem Prototyp umgesetzt. Im Folgenden dokumentieren wir die Vorgehensweise und das Ergebnis.

Das Ziel ist es, dass die Fachhandlung (Desktop Anwendung) Wasserwerte über den Server (Node.js) an den Kunden (Android App) schicken kann und dass dieser dann eine Push-Benachrichtigung bekommt. Also haben wir erstmal mit der Implementierung des Servers angefangen. Diesen haben wir mit Node.js und Express.js umgesetzt. Dort haben wir zwei Routen programmiert. Eine für

Benutzer und eine für Wasserwerte. Ruft man diese Routen mit GET auf, werden alle Benutzer bzw. alle Wasserwerte ausgegeben. Wenn man einen POST Request dahin schickt, wird ein Benutzer angelegt bzw. neue Wasserwerte in der Datenbank gespeichert.

Als nächstes haben wir dann mit der Android App weitergemacht. Dort haben wir zuerst versucht, eine Verbindung via POST und GET mit dem Server aufzubauen. Als das geklappt hat, haben wir mit der Implementierung des Firebase Cloud Messaging begonnen. Die Integration in die App war sehr einfach und konnte direkt über Android Studio vollzogen werden. Es wurden zwei Klassen dafür angelegt. Die eine Klasse sorgt dafür, dass bei der ersten Nutzung ein Token generiert wird. Dieser musste nun an den Server geschickt werden. Dafür haben wir dann unsere zuvor programmierte ServerRequest Klasse verwendet, mit der der Token nun via POST Request an die Benutzer Route des Servers geschickt werden konnte. Die zweite Klasse sorgt für das Empfangen der Nachrichten, dazu später mehr.

Jetzt kam erstmal die Desktop Anwendung für die Fachhandlung. Da haben wir zuerst eine graphische Oberfläche mit drei Textfeldern und einem Button erstellt. Dem Button haben wir einen ActionListener hinzugefügt, in dem wir einen POST Request mit den Werten aus den Textfeldern an die Wasserwerte Route des Servers geschickt haben. Neben den Wasserwerten war ein Textfeld für den Empfänger vorhanden. Dort musste der Token des Empfängers angegeben werden, den man der Datenbank bzw. der Benutzer Route entnehmen konnte. Für die Durchführung des PoC reichte uns das erstmal, später wollen wir die Adressierung des Empfängers anders angehen, damit nicht immer der lange Token eingegeben werden muss. Die ServerRequest Klasse konnten wir zum größten Teil von der App übernehmen.

Nachdem die Kommunikation der beiden Clients mit dem Server umgesetzt war, haben wir der Wasserwerte Route des Servers noch den entscheidenden Code hinzugefügt, der dafür sorgt, dass die Nachricht, die von der Fachhandlung kommt, nicht nur in der Datenbank gespeichert wird, sondern auch an den adressierten App Benutzer geschickt wird. Dieser kann die Nachricht dann mit der zuvor programmierten FCM Klasse empfangen und erhält nun direkt eine Push-Benachrichtigung, sobald Wasserwerte von der Fachhandlung verschickt wurden. Wenn er auf die Benachrichtigung drauf klickt, aktualisiert sich die App und die Wasserwerte werden aus der Datenbank geladen.

Die Durchführung des PoC war also erfolgreich.

Beschaffung der Formeln für einzelne Berechnungen

Dabei war uns besonders eine Formel, welche basierend auf den Inhalten im Aquarium nach einmaliger Analyse den täglichen Nährstoffverbrauch ungefähr berechnen kann, von großer Bedeutung. Nach einer langen Recherche im Internet und Fachliteratur sind wir leider auf ein anderes Ergebnis gekommen, bei den anderen Berechnungen waren wir aber erfolgreicher.

Vor allem in Internetforen waren Hinweise auf die vielen chemischen Formeln und Einflüsse auf ein Aquarium und die unzähligen Berechnungen zu finden. Für einige weitere Berechnungen reichte einfaches logisches Denken.

Resultierend sind für uns folgende Berechnungen von Bedeutung:

- Täglicher Nährstoffverbrauch (logisches Denken)
- Düngerdosierung beim Erreichen eines bestimmten Zielwertes (logisches Denken)
- Berechnung der Gesamthärte[1]
- Berechnung des CO2- Gehalts anhand von pH-Wert und Karbonathärte[2]
- Verhältnis bei einem Wasserwechsel, um eine bestimmte Gesamthärte oder Karbonathärte zu erreichen[3]
- Berechnung der täglichen Nährstoffänderung mit Hilfe der Pflanzen, Fische, dem Licht, den anderen Einflüssen im Aquarium und einer einmaligen Wasseranalyse

Für die letzte Berechnung haben wir leider keine Berechnung gefunden, es gibt bei der Nährstoffänderung viel zu viele Einflüsse, so dass es einfach nur zusätzlicher Aufwand sein würde. Damit gilt dieses Proof of Concept als gescheitert und wir müssen auf unser Fallback zugreifen.

Im nächsten Schritt werden wir nun also eine Umfrage bei Aquarienbesitzern in mehreren Facebookgruppen und Aquaristikforen durchführen, um eine Annäherung an eine Formel zu bekommen. Da diese Umfrage und ihre

[1] http://www.trinkwasserspezi.de/Berechnungen.pdf (letzte Sichtung, 04.11.16 21:00)
[2] http://www.flowgrow.de/db/calculator/ph (letzte Sichtung, 04.11.16 19:25)
[3] http://web31722.greatnet-hosting.de/4aq-technik/Aquarium-Rechner/tww_mit_zielwert.php (letzte Sichtung 03.11.16 15:32)

Ergebnisse sehr umfassend sein werden, werden wir uns auf folgende Einflüsse reduzieren:

- Täglicher Nährstoffverbrauch
- Lichteinfluss (Lichtstunden und Lichtstärke)
- Beckengröße
- Anzahl an Pflanzen
- Anzahl an Fischen
- Art des Filters

Mit Hilfe des Ergebnisses werden wir eine sehr grob angenäherte Berechnung für die tägliche Nährstoffänderung erreichen. Ebenso werden wir unsere Anforderungen iterieren und weitere Aspekte für die Anwendungslogik finden.

Umsetzung der Formeln

Wir haben die ersten Berechnungen in unserem Prototyp bereits implementiert, dazu gehören:

- Berechnung des täglichen Nährstoffverbrauchs (beim Fachhändler)
- Düngerdosierung beim Erreichen eines bestimmten Zielwertes
- Berechnung des CO2- Gehalts anhand von dem pH-Wert und der Karbonathärte

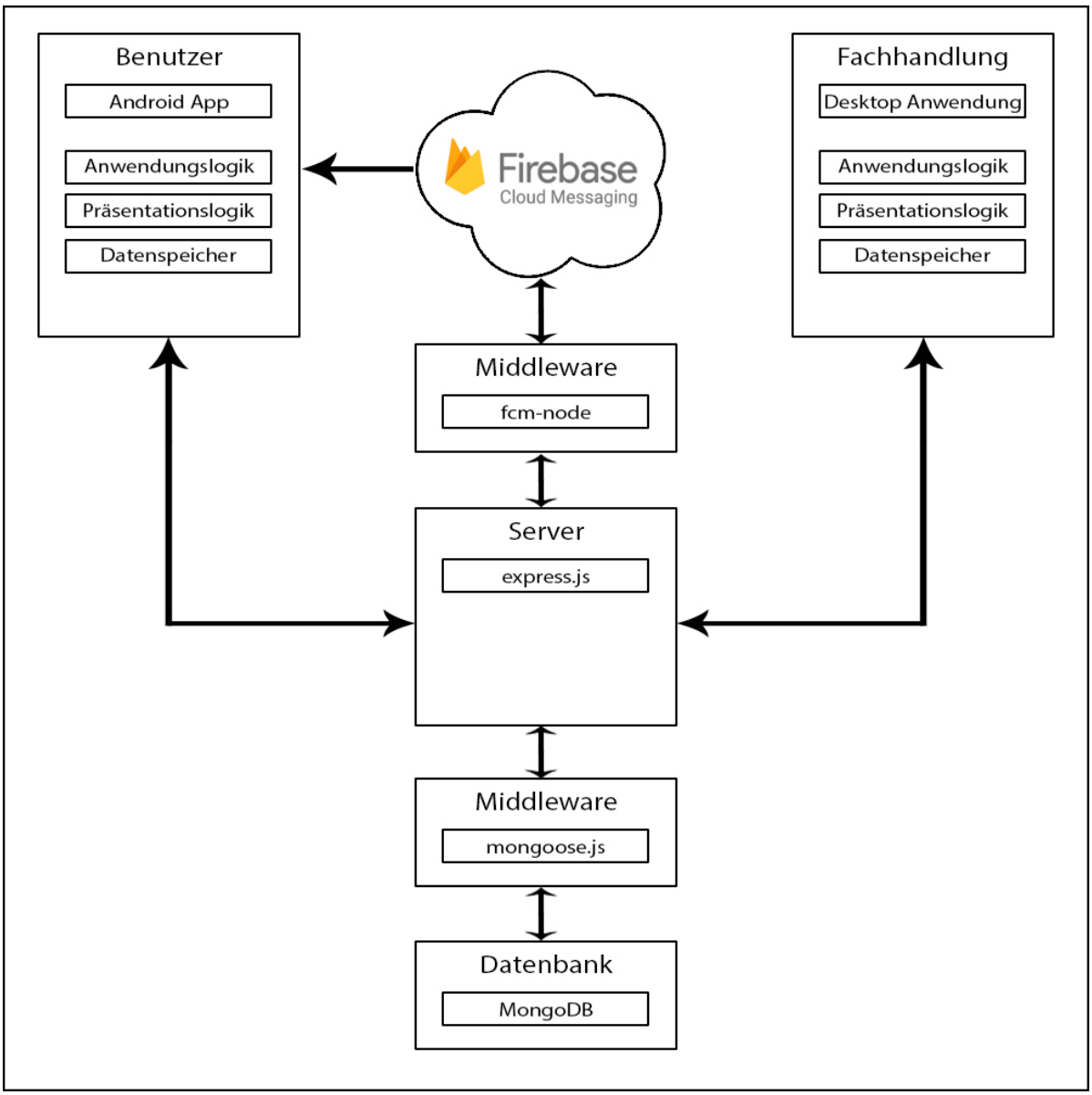
Neben der eher einfachen Programmierung der Berechnungen an sich, wurde ein besonderes Augenmerk auf die Verarbeitung der einzelnen Werte und das Mitteilen über das Firebase Cloud Messaging gelegt. In der weiteren Entwicklung werden wir besonders auf die einzelnen Maßeinheiten Acht geben müssen, da es hier des Öfteren schon in der Umsetzung zu Problemen gekommen ist. Die Kommunikation zwischen dem Client des Fachhandels, dem Server und dem Kunden hat dabei sehr gut funktioniert. Dieses PoC wurde erfolgreich abgeschlossen.

7 Serverarchitektur

Da die Kommunikation zwischen den verschiedenen Clients eine bedeutende Rolle spielt, wird diese im Folgenden skizziert. Zuvor sind hier die verschiedenen Komponenten aufgelistet:

- Server
- Mobiler Client für Benutzer
- Desktop Client für Fachhandlungen
- Firebase Cloud Messaging
- Datenbank

- Middleware



Server

Der Server dient zum Datenaustausch zwischen den Komponenten. Bei der Umsetzung des Servers konnten wir zwischen PHP und Node.js wählen. Hier haben wir uns für eine Umsetzung mit Node.js und Express.js entschieden, da wir bereits Erfahrung damit haben und da es sich gut für unser System eignet.

Datenaustausch und Datenbank

Über http können Daten im JSON Format zwischen den Clients und dem Server ausgetauscht werden und diese können ggf. vom Server in der Datenbank gespeichert werden. Als Datenbank benutzen wir MongoDB, da diese gut in Verbindung mit Node.js eingesetzt werden kann. Als Middleware für die Interaktion mit MongoDB benutzen wir das Node Modul mongoose.js. In der

Datenbank werden zum Beispiel die Tokens für das Firebase Cloud Messaging gespeichert. Dieses benutzen wir, um Nachrichten direkt mit „Push-Notification“ an den App-Nutzer zu verschicken. Wenn zum Beispiel die Fachhandlung die Ergebnisse einer Wasseranalyse verschickt, gehen diese erst an den Server und der übermittelt diese per FCM an den Benutzer. Im Vergleich zum Google Cloud Messaging ist das Firebase Cloud Messaging (die neue Version vom Google Cloud Messaging) leichter in die App zu integrieren und hat weitere Features.

Benutzer Client

Für den Benutzer Client haben wir uns für eine Android App entschieden, da diese mit Java realisiert wird und es somit sich gut für das Projekt eignet. Der Client hat sowohl eigene Anwendungslogik, wie zum Beispiel das Berechnen der Düngemittel, als auch eigene Präsentationslogik und einen Datenspeicher. Dort werden zum Beispiel die Aquarium Daten gespeichert, damit der Benutzer auch offline darauf zugreifen kann.

Fachhandlung Client

Für die Fachhandlung haben wir uns für eine Desktop Anwendung entschieden, die wir ebenfalls mit Java realisieren. Die Fachhandlung soll eine gute Übersicht über ihre Kunden und deren Daten bekommen, weshalb eine Desktop Anwendung hier mehr geeignet ist als eine App, da der sichtbare Bereich größer ist. Auch dieser Client hat eigene Anwendungslogik, Präsentationslogik und Datenspeicher.

8 Installationsdokumentation

Node.js Server

Wir haben unseren Node.js Server online auf uberspace.de hochgeladen. Er ist unter http://eis1617.lupus.uberspace.de/nodejs/ zu erreichen. Da wir den Server online haben, läuft er durchgehend und wir müssen diesen nicht mehr auf allen Geräten, auf denen wir programmieren, installieren. Um den Server zu testen können Requests zum Beispiel von der Seite https://www.hurl.it/ abgeschickt werden. Dabei sollte man beachten, dass bei einem POST Request der Header „Content-Type“ gleich „apllication/json“ gesetzt werden sollte. Wenn Sie den Server auch lokal testen wollen, müssen Sie folgende Installationsschritte vornehmen:

Vorbereitungen:

1. MongoDB installieren (https://www.mongodb.com/download-center#community)
2. Folgenden Ordner manuell erstellen: C:\data\db
3. Die Datei C:\Program Files\MongoDB\Server\3.2\bin\mongod.exe ausführen
4. Der MongoDB Server läuft nun

Server Installation:

1. Repository (https://github.com/jkimmeyer/EISWS1617MuellerKimmeyer) runterladen und in den Ordner /MS1/proofofconcept/server gehen
2. Die Konsole in diesem Ordner öffnen und folgenden Befehl eingeben: npm install
3. Die Datei server.js öffnen und Zeile 23 auskommentieren und Zeile 25 entfernen
4. Nun in der Konsole eingeben: node server.js
5. Jetzt können Sie im Browser die Adresse localhost:61000 aufrufen oder mit dem REST Client Ihrer Wahl Requests an den Server schicken

(Die .htaccess Datei hat keinen lokalen Nutzen. Sie befindet sich auf unserem Server und sorgt dafür, dass man zum laufenden Node Dienst weitergeleitet wird. Wir haben sie trotzdem einfach mal mit in den Ordner gepackt.)

Benutzer Client – Android App

Vorbereitung:

1. Android Studio installieren (https://developer.android.com/studio/index.html)

Android App importieren:

1. Repository (https://github.com/jkimmeyer/EISWS1617MuellerKimmeyer) runterladen
2. Android Studio öffnen
3. File -> Open… und dann den Pfad zum Repository und dann zum Ordner \MS1\proofofconcept\benutzerClient\Aquaapp angeben
4. Links unter app->java->de.eis.muellerkimmeyer.aquaapp können nun die von uns erstellten Java Dateien geöffnet werden

App starten:

1. Oben in der Menüleiste auf den grünen Pfeil klicken

2. Dann ein virtuelles Device erstellen oder ein Android Smartphone anschließen und dieses auswählen

Fachhändler Client – Desktop Anwendung

Vorbereitung:

1. Eclipse installieren (https://www.eclipse.org/downloads/)

Projekt importieren:

1. Repository (https://github.com/jkimmeyer/EISWS1617MuellerKimmeyer) runterladen
2. Unter File->Open Projects from File System… und dann auf Directory… den Ordner „fachhandlungClient“ in unserem Repository auswählen und auf Finish klicken
3. Unter Aquaapp/src/de.eis.muellerkimmeyer können nun die von uns erstellten Java Dateien geöffnet werden
4. Anwendung kann jetzt gestartet werden

Testablauf

1. Rufen Sie im Browser die URL http://eis1617.lupus.uberspace.de/nodejs/users auf oder falls Sie den Server lokal gestartet haben http://localhost:61000/users. Nun können Sie alle bisher eingetragenen Benutzer sehen.
2. Öffnen Sie nun die Android App. Sobald die App gestartet ist, sollte ein Token generiert werden und dieser wird an den Server geschickt. Wenn Sie nun die Seite aus Punkt 1 aktualisieren, sollten Sie sehen, dass ein neuer Eintrag hinzugekommen ist.
3. Öffnen Sie nun den Fachhändler Client. Dort können Sie zwei verschiedene Wasserwerte eintragen sowie einen Empfänger. Gehen Sie jetzt nochmal auf die Seite aus Punkt 1 und kopieren Sie sich den neu hinzu gekommenen Token. Dieser kann bei Empfänger eingegeben werden. Wenn Sie jetzt auf absenden klicken, wird die Nachricht an den Server übermittelt und dieser schickt eine Push-Benachrichtigung an die Android App.

4. Gehen Sie nun wieder zu der Android App. Dort sollte jetzt eine Push-Benachrichtigung erschienen sein. Wenn Sie auf diese drauf klicken lädt die App neu. Jetzt sieht man, dass die im Fachhändler Client eingegebenen Wasserwerte auch in der Android App erschienen sind.

Testdaten

Testdaten haben in diesem Fall keinen Sinn gemacht, da die Daten beim Benutzen des Systems anfallen. Sollten Sie allerdings den Server nutzen, den wir online haben, kann es sein, dass noch ein paar Datensätze vorhanden sind.