Synchroner Datenaustausch zwischen Android und NodeJS mit Speicherung in MySQL

Es gilt in diesem "Proof-of-Concepts" zu testen, ob eine MySQL-Datenbank in NodeJS implementiert werden kann und ein synchroner Datenaustausch per http-Protokoll zwischen dem Android-Client und NodeJS möglich ist. Dabei soll der Android-Client Daten vom NodeJS-Server synchron abfragen und anschließend anzeigen und Daten über den NodeJS-Server in die Datenbank ablegen.

Bezug zum Projekt

Die Benutzer des Systems müssen über die Clients Daten mit dem Server austauschen können. Das Pflegepersonal muss in der Lage sein Daten über Verordnungen der Patienten vom Server zu erhalten, damit ein Medikationsplan erstellt werden kann. Ebenso müssen die Dokumentationen der Verabreichungen auf einer Datenbank persistent abgelegt werden können. Sollte dies nicht möglich sein, so sind die Alleinstellungsmerkmale und Kernfunktionen des Systems gefährdet.

Exit-Kriterium

Das "Proof-of-Concepts" gilt als erfolgreich abgeschlossen, wenn die Implementierung einer MySQL Datenbank in NodeJS möglich ist und Daten vom Android-Client aus dem NodeJS-Server abgefragt und Daten über den Node-Server fehlerfrei abgelegt werden können.

Fail-Kriterium

Das "Proof-of-Concepts" gilt als gescheitert, wenn eine Implementierung von einer MySQL-Datenbank in NodeJS nicht möglich ist oder die Daten vom Android-Client entweder fehlerhaft oder nicht in die Datenbank abgelegt oder abgerufen werden können.

Fallback

Sollte die Implementierung einer MySQL-Datenbank scheitern, so muss eine MongoDB-Datenbank verwendet werden. Wenn ein synchroner Datenaustausch per http Protokoll zwischen dem Android-Client und Node-JS nicht möglich ist, muss ein anderes Protokoll verwendet werden. Das Protokoll 'AMQP' wurde im Zusammenhang mit RabbitMQ bereits im "Proof-of-Concepts" "POC

Publish-Subscribe mit RabbitMQ" erfolgreich getestet. Daher ist ein Datenaustausch über RabbitMQ mit einem AMQP-Protokoll eine Alternative.

Prozessdarlegung

Zunächst muss eine MySQL-Datenbank installiert werden. Das Programm "XAMPP" bietet die Möglichkeit eine MySQL-Datenbank lokal auf dem Rechner laufen zu lassen. Die benötigten Verbindungsdaten für die Datenbank liegen im Installationsordner als Textdatei und können nun verwendet werden, um eine Verbindung von NodeJS zur Datenbank aufzubauen.

In NodeJS kann das Modul "mysql" verwendet werden, um eine Verbindung zur MySQL-Datenbank aufzubauen. Die Modul-API[1] ist knapp und übersichtlich. Eine Verbindung kann durch wenige Zeilen Code bereits hergestellt werden. Die Abfrage und das Ablegen von Daten soll zunächst mit einem Javascript-Client getestet werden. Dazu wurde eine HTML-Seite angelegt, die über die Route "/verordnungen" des NodeJS-Servers erreichbar ist. Sobald die Seite geladen ist, soll nun ein AJAX-GET-Request auf "localhost/verordnung" gefeuert werden.
Auf dem Server wurde in die Methode get('/verordnung') ein Query auf die Datenbank mit der Tabelle "verordnungen" programmiert. Die Tabelle wurde über den PHP-Admin-Bereich von XAMPP in unserer Datenbank angelegt und Testdaten wurden eingefügt. Die HTML-Seite wird nun mit einem kleinem Formularfeld und einer Tabelle erweitert, die für die Präsentation der Datenbankdaten verwendet wird. Auf dem Server wird dementsprechend eine post()-Methode programmiert, die greift und Daten in die Tabelle der Datenbank ablegt, wenn ein AJAX-POST mit dem Inhalt des Inputfeldes aus dem Formular der HTML-Seite gefeuert wird.
Nach einem Test konnten alle Daten aus der Datenbank präsentiert und auch über NodeJS in die Datenbank abgelegt werden können.

Da die Implementierung von einer MySQL-Datenbank in NodeJS gelungen ist, muss nun überprüft werden, ob ein synchroner Datenaustausch auch von einem Android-Client heraus zum NodeJS-Server programmiert werden kann.
Das Android SDK bietet für http-Verbindungen eine Klasse[2] "HttpUrlConnection". Die Beispiele aus der Dokumentation wurden verwendet, um eine Verbindung zum NodeJS-Server auszubauen. Wichtig ist an dieser Stelle, dass im Android-Manifest die Erlaubnis zur Verwendung des Internets eingetragen ist und für die Verbindung die IP des lokalen Rechners verwendet wird. Zur Präsentation muss der Response des Servers konvertiert werden, sodass

[1] https://github.com/felixge/node-mysql Sichtung: 05.11.2015, 18:20
[2] http://developer.android.com/reference/java/net/HttpURLConnection.html

wird die Daten in eine Textfeld schreiben und anzeigen können. Dazu wurde eine Klasse Verordnung angelegt, wodurch die Daten aus dem Response in Instanzen diese Klasse abgelegt werden und anschließend in das Textfeld geschrieben werden.
Der erste Test schlug allerdings fehl. Recherchen zeigten, dass es nötig war die http-Verbindung in Android in einem neuen Thread aufzurufen. Nachdem der der Code für die Verbindung innerhalb eines neu angelegten Threads implementiert wurde, konnten die Daten empfangen und im Textfeld angezeigt werden.
Über ein http-post soll nun noch getestet werden, ob Daten über den Android-Client auch in die Datenbank abgelegt werden können. Dazu muss der bereits vorhandene Code für den http-get leicht geändert werden und ein neuer Thread definiert werden.
Nach einigen Tests konnten keine Daten per http-post auf die Datenbank geschrieben werden. Der Grund dafür liegt im JAVA-Code, da die post-Methode des NodeJS-Servers nicht ausgerufen wird. Es muss weiter Recherchiert werden, wo genau der Fehler im Code liegt und welche Veränderungen vorgenommen werden müssen, damit auch der http-post aus dem Android-Client heraus möglich ist.

Status

Der "Proof-of-Concepts" muss im Hinsicht auf den http-post des Android-Clients weiter untersucht werden. Da ausser das Ablegen von Daten in die Datenbank über den Android-Client alle Exit-Kriterien erfüllt wurden, ist es sinnvoll weitere Recherchen zu unternehmen, um die Fehler im Code zu beheben. Daher ist der "Proof-of-Concepts" weiterhin in Bearbeitung.

Ausblick

Es ist nun wichtig, die Datenstrukturen, die auf der Datenbank liegen sollen zu diskutieren. Dazu müssen notwendige Tabellen und ihre Relationen sowie Attribute spezifiziert werden. Sobald dies festgelegt wurde, müssen entsprechende Java-Klassen für die notwendigen Datenobjekte konzipiert werden.

Anhang:

Installationen:

1. Node-JS
    a. mysql
2. XAMPP