Lars Ullrich (2172937)
Dokumentation Abschlussprojekt
Mobile Webanwendungen
Wintersemester 15/16

'AddMe' – QR-Codes für Kontaktdaten

Dokumentation

Table of Contents

Lars Ullrich (2172937)
Dokumentation Abschlussprojekt
Mobile Webanwendungen
Wintersemester 15/16

Motivation

Die App ist eine Alternative zur normalen Kontaktverwaltung, die das Teilen der eigenen Kontaktinformationen erleichtert. Zum Teilen wird ein QR-Code generiert, der nur die Informationen enthält, die der Nutzer auch teilen möchte. Die generierten Codes können von der App gescannt werden, um neue Kontakte schnell hinzuzufügen.

Die Informationen, die zur Zeit enthalten sind, sind Name, Adresse, Telefonnummern, E-Mail sowie Social-Media-Informationen (Twitter, Github und Reddit). Diese könnten perspektivisch noch beliebig erweitert werden. Um die grundlegende Funktionalität der App prototypisch zu veranschaulichen, reichen die momentan enthaltenen und teilbaren Informationen jedoch aus.

Aufbau der App

Die App startet mit einem simplen Home-Bildschirm, der Links zu den drei zentralen Elementen der App beinhaltet, zum Nutzerprofil, zum Barcode-Scanner sowie zu den Kontakten. Im Header ist außerdem ein Menü-Button, der die Seitennavigation öffnet. Da es in der Benutzung der App keinen allgemeingültigen Ablauf gibt, ist es dem Nutzer selbst überlassen, welche Funktion er nutzen möchte. Die App ist so konzipiert, dass man schnell zu der Funktion kommt, die man gerade benötigt, ohne dass man sich lange druchklicken muss.

Scan

Der Scan-View enthält zunächst nur einen Button, mit dem der Scanner aufgerufen wird. Hat man einen QR-Code erfolgreich gescannt, werden die Kontaktinformationen, die der Gegenüber geteilt hat, angezeigt. Informationen, die nicht im QR-Code enthalten waren, aber teilbar gewesen wären, werden nicht mit angezeigt, um die Anzeige nicht zu überfrachten.

Durch einen Klick auf den 'Save Contact'-Button wird der neue Kontakt in den Kontakten gespeichert, und der Nutzer wird zur Kontakteseite weitergeleitet. Sollte ein Konflikt vorliegen, klärt ein Popup den Nutzer darüber auf, ansonsten teilt das Popup dem Nutzer das erfolgreiche Speichern des Kontakts mit.

Lars Ullrich (2172937)
Dokumentation Abschlussprojekt
Mobile Webanwendungen
Wintersemester 15/16

Contacts & Contact

Die Contacts-Seite ist ein einfacher List-View mit allen lokal vorliegenden Kontakten. Diese Kontakte leiten auf die jeweilige Detailseite weiter.

Die Detailansicht der Kontakte listet die zu dem jeweiligen Kontakt gespeicherten Informationen auf.

User

Die User-Seite bietet eine Übersicht über die im Nutzerprofil gespeicherten Informationen, sowie die Möglichkeit, diese Informationen (oder auch nur Teile davon) mittels eines dynamisch erzeugten QR-Code mit anderen Nutzern der App zu teilen. Bis auf den Namen des Nutzers sind alle Informationen optional über einen Schalter rechts in der jeweiligen Spalte ein- und ausschaltbar. Wenn die Schalter deaktiviert sind, werden die zugehörigen Informationen nicht in den QR-Code mit aufgenommen. Der Button 'show QR' im Namensfeld generiert den QR-Code und blendet ihn ein. Über einen Regler lässt sich seine Größe ändern, damit er einfach an unterschiedliche Displaygrößen angepasst werden kann.

Der Nutzer kann sich über zwei Buttons über seinem Profil mit Twitter und Github verbinden. Nach erfolgreicher Anmeldung bei Twitter, wird der *screen_name* im Profil unter dem Punkt Twitter gespeichert (diese Funktionalität fehlt leider bei Github noch). Der Nutzer wird hierüber durch ein Popup informiert. Über den Button 'Edit' unter dem Profil kann man das Profil bearbeiten.

Settings

Die Settingsseite ist ein einfaches Formular, in das der Nutzer seine Daten eingeben oder ändern kann. Wenn alle Eingaben erfolgt sind und der Nutzer sein Profil über den 'Save'-Button speichert, wird er zur Profilseite weitergeleitet und kann somit die Informationen direkt über den QR-Code teilen.

Frameworks & Komponenten

Bei der App handelt es sich um eine hybride Anwendung, die mithilfe des auf AngularJS

aufbauenden Ionic-Framework erstellt worden ist. Ionic erweitert Angular um zahlreiche Elemente, die die Umsetzung von Präsentation und Funktionalität erleichtern sollen. Insbesondere in Apps immer wieder benötigte Funktionen wie List-Views, große Buttons oder Modals und Popups sind mittels der enthaltenen Direktiven und Provider teils sehr einfach umzusetzen. Das Routing übernimmt in Ionic der populäre alternative UI-Router anstatt des zu Angular gehörenden Routers. Dieser ist etwas komplexer, ermöglicht jedoch erweiterte Funktionalitäten wie verschachtelte oder multiple Views. Die App macht hiervon durch die abstrakte übergeordnete app-Route Gebrauch, die einige in mehreren Views benötigte Funktionen bündelt (in diesem Fall das Initialisieren der Datenbank sowie die Nutzerverwaltung).

Ionic bietet nicht nur Funktionen zum erleichterten Umgang mit Angular, sondern auch Möglichkeiten zur Verwaltung von Cordova. Das zugehörige Command-Line-Interface bietet einen Webserver für die Entwicklung im Browser, Funktionen zum Kompilieren der APK und für das Installieren auf einem Gerät oder in einem Emulator. Auch Plugins für Cordova lassen sich mit dem CLI verwalten.

Plugins & Bibliotheken

In-App-Browser und Oauth

Für die Anmeldung über Twitter und Github werden zwei Plugins verwendet. Zum einen der In-App-Browser, der das Aufrufen der jeweiligen Anmeldeseiten übernimmt, ohne dass man die App verlassen muss (was nicht funktionieren würde), zum anderen das Oauth-Modul von ng-Cordova. Mit diesem Plugin ist der Prozess der Authentifizierung mittels Oauth bei zahlreichen Anbietern sehr leicht umzusetzen. Das Modul übernimmt das Aufrufen des In-App-Browsers und die Datenübertragung in beide Richtungen. Da Oauth für die Authentifizierung von Web-Anwendungen entwickelt wurde, kommt es beim Aufrufen der Callback-Route von den Anbietern leider kurzzeitig zu einer Fehlerseite des In-App-Browsers, da diese Route in der hybriden App nicht existiert bzw. hier ein Webserver läuft.

QR-Code und Angular-qrcode

Für das dynamische Erzeugen des QR-Codes wird Angular-qrcode verwendet, das auf einer Javascript-Bibliothek für die Erstellung von QR-Codes basiert und die Einbindung mittels einer

Angular-Direktive sehr vereinfacht. Optionen werden direkt im QR-Tag angegeben und mittels *two-way data binding* lassen sich sowohl der codierte Inhalt als auch Eigenschaften des erstellten Bilds selbst sehr leicht verändern. Dies ermöglicht es, die zu teilenden Inhalte durch die Schalter im Profil gezielt zu steuern sowie die Größe des Codes durch den Regler in Echtzeit zu verändern.

Cordova-QR-Scanner

Ng-Cordova bietet neben zahlreichen anderen Funktionen auch eine einfache Möglichkeit, QR-Codes schnell mit der Gerätekamera zu scannen und die codierten Daten zu erhalten. Der $cordovaBarcodeScanner-Provider liefert über die scan-Methode ein *promise* zurück, das dann im Controller weiterverarbeitet werden kann. Das Aufrufen der Kamera und des Scanner-Fensters geschieht hier vollautomatisch. Da der codierte Text im JSON-Format vorliegt, muss dieser im Controller lediglich verarbeitet und die Daten letztlich gespeichert werden.

PouchDB

Für das Speichern des Nutzerprofils und der Kontakte werden zwei interne Datenbanken benutzt. Bei beiden handelt es sich um PouchDB-Datenbanken, die den WebSQL-Speicher des unter Android laufenden Web-View benutzen. Eine andere Möglichkeit wäre gewesen, die Daten in einer SQLite-Datenbank zu speichern, da es sich jedoch bei den Kontakten und dem Profil um nur sehr kleine Textdateien handelt, ist die Speicherbeschränkung der WebSQL-Datenbank hier kein Problem.

Lodash

Lodash ist eine Bibliothek für funktionale Programmierung in Javascript, die zahlreiche Utility-Funktionen für den Umgang mit verschiedenen Datentypen bereitstellt. Insbesondere das Arbeiten mit Arrays und Objekten wird durch Lodash sehr vereinfacht. Lodash ist dabei darüber hinaus auch häufig schneller als native Implementationen der selben Funktionen. In der App kommt Lodash beim Erstellen des QR-Codes zum Einsatz, da hier die nicht eingeschalteten Felder ausgeschlossen werden müssen, sowie beim Abrufen der Kontakte.

Controller

Die App kommt mit fünf relativ kurzen Controllern aus, die hier nicht mit der ControllerAs-Syntax

von Angular benutzt werden, da die Vererbung durch die abstrakte Route 'app' und den zugehörigen Controller bei der Entwicklung so leichter umzusetzen war.

AppCtrl

Der AppCtrl ist für die abstrakte übergeordnete Route zuständig und kümmert sich um die Nutzerdaten. Hier wird der Nutzer mittels 'User.init()' initialisiert und über die doLogin-Funktion gespeichert (das Editieren und Erstellen des Nutzers war ursprünglich in einem Login-Fenster ähnlichen Modal implementiert, das jedoch der dedizierten Settings-Seite weichen musste. Der Name ist ein Relikt davon). Dies hat den Vorteil, dass die Nutzerdaten theoretisch mehreren Views zur Verfügung stehen, wovon aber in der aktuellen Version nicht mehr Gebrauch gemacht wird.

UserCtrl

Der UserCtrl kümmert sich um das Erstellen des QR-Codes, das Filtern der in den Code aufgenommenen Informationen sowie um die Anmeldung bei Twitter und Github.

Die Funktion 'createQR' kümmert sich um das Filtern der Informationen und das Erstellen des JSON-Dokuments, das letztendlich durch die QR-Code-Direktive im User-View codiert und angezeigt wird. Das User-Objekt ist so aufgebaut, dass es für jedes Attribut zwei Felder gibt. Eines dieser Felder ist ein boolescher Wert, der angibt, ob etwas in den Code mit aufgenommen werden soll oder nicht, das andere ist der Wert selbst. Der zweite Wert kann dabei jede Form annehmen, da es z.B. bei der Adresse eher Sinn macht, über die Aufnahme der Adresse als Ganzes zu entscheiden, anstatt über jede einzelne Komponente. Im User-Service ist ein Beispielnutzer zu finden, der das Schema veranschaulicht:

```
scope.user = {
        name: {
            include: true,
            value: {
              fname: 'First Name',
              lname: 'Last Name',
            }
        },
```

```
        address: {
          include: true,
          value: {
            street: 'Street',
            no: 'Number',
            zip: 'Zip Code',
            city: 'City',
            country: 'Country'
          }
        },
        tel: {
          include: true,
          value: '555 555 555'
        },
        mobile: {
          include:true,
          value:'999 999 999'
        },
        email: {
          include:true,
          value:'email@example.com'
        },
        twitter: {
          include: true,
          value:'screen_name'
        },
        github: {
          include: true,
```

```
          value:'username'
        },
        reddit: {
          include: true,
          value:'username'
        }
      };
```

Die include-Werte des User-Models sind an die Schalter im Profil gebunden. Nur Werte mit 'include === true' werden in den QR-Code mit aufgenommen. Lodash bietet hierfür eine Filtermöglichkeit, die in *createQR* zur Anwendung kommt ('var userinfo = _.pickBy($scope.user, 'include');').

Die Größe des QR-Codes wird über den Regler im View bestimmt, der mittels *two-way data binding* an das Attribut 'qrsize' gebunden ist.

Die Aouth-Funktionen sind duch das Plugin sehr einfach. Ursprünglich waren auch Optionen zum automatischen folgen geplant, leider habe ich diese nicht umsetzen können. So wird im Moment nur der Twitter-screen_name gespeichert, wenn die Anmeldung erfolgt ist.

ContactsCtrl & ContactCtrl

Beide Controller kümmern sich lediglich darum, beim Wechsel zu zugehörigen Route, die benötigten Daten aus der Datenbank zu laden. ContactCtrl ist der Controller für die Detailseite und muss deshalb die benötigte ID für den aktuellen Kontakt aus den Parametern der Route beziehen. ContactsCtrl muss darauf nicht zugreifen, da hier alle Dokumente aus der Kontaktedatenbank geladen werden.

ScanCtrl

Der ScanCtrl ist zuständig für das Scannen und Verarbeiten von QR-Codes und den darin codierten Daten. Es war etwas schwierig, diese Funktion zu testen, da dies nur auf einem Testgerät möglich war, weil Zugriff auf die Kamera benötigt wurde. Letztendlich musste ich den QR-Code im Browser auf dem PC generieren und mit dem Testgerät vom Bildschirm scannen. Dies hat zwar nicht immer sofort funktioniert, aber nach ein paar Versuchen werden die Daten nun korrekt vom

Gerät erkannt.

Wenn die Daten nach erfolgreichem Scan vorliegen, können sie in der Kontaktdatenbank gespeichert werden. Die ID des neuen Datensatzes besteht aus Nachname und Vorname des neuen Kontakts, wodurch es zu Konflikten kommen kann. Eine gute Lösung für diese Konflikte ist mir nicht eingefallen, aber ein Popup klärt den Nutzer zumindest über den Konflikt auf.

Services

Datenbanken & zugehörige Services

Die App macht für das Instanziieren der Datenbanken von Angular-Constants Gebrauch. Hierbei handelt es sich um unveränderbare App-weite Konstanten. Es werden zwei Datenbanken benötigt, weil für die Kontakte alle Dokumente in der zugehörigen Datenbank geladen werden. Auch die Zuordnung der Kontakte zu einem Nutzer muss nicht gegeben sein, da es zu jeder Zeit nur einen Nutzer gibt. Die Trennung von Nutzerobjekt und Kontakten macht daher Sinn.

Der User-Service kümmert sich um die Initialisierung des Nutzerobjekts. Er nutzt die Nutzerdatenbank-Konstante ('db'), um den einzig möglichen Nutzer zu laden, wenn dieser vorliegt (das Nutzerobjekt hat die ID 'user' und wird von der App stets nur aktualisiert). Wenn kein Nutzer vorliegt, es also einen Fehler gibt, wird ein Beispielnutzer instanziiert.

Der Contacts-Service kümmert sich um den Umgang mit der 'contacts'-Datenbank. Er hat zwei Methoden:' getAll' lädt alle in der Datenbank vorhandenen Dokumente und 'getOne' lädt nur den Nutzer mit der als Parameter übergebenen ID.

OAuth

Die Services für die Oauth-Funktionen von Twitter und Github enthalten lediglich die für die Nutzung der Services notwendigen Informationen. In diesen Fällen sind das consumerKey und consumerSecret für Twitter und clientID, clientSecret sowie scopes (Rahmen der Zugriffsrechte) für Github. Die eigentliche Implementierung des Oauth-Services geschieht mit dem Plugin $cordovaOAuth, womit dies extrem einfach war.

Lars Ullrich (2172937)
Dokumentation Abschlussprojekt
Mobile Webanwendungen
Wintersemester 15/16

Ausblick

Leider habe ich viele meiner Ideen nicht umsetzen können. Dies lag einerseits an meinem Anspruch, dass alles, was implementiert ist, auch verlässlich funktionieren soll, und andererseits an der Komplexität des Oauth-Verfahrens. Leider habe ich es nicht geschafft, die Services dahingehen anzupassen, dass die App Zugriff auf die entsprechenden Endpunkte der Twitter- bzw. Github-API hat. Letztendlich habe ich mich dafür entschieden, diese Funktionen eventuell in der Zukunft hinzuzufügen, wenn ich mich damit etwas besser auskenne. Mir war bei diesem Projekt die Stabilität der App wichtiger als die Menge an Funktionen. Der momentane Stand der App ist daher eher eine Momentaufnahme, da sie zum jetzigen Zeitpunkt eher eine solide Grundlage für eine sukzessive Weiterentwicklung ist.