Technische Universität Berlin

Projekt AAL

Applikationsgruppe

Dokumentation

Tom Nick
Jonathan Seilkopf
Niklas Gebauer
Maximilian Bachl
Tom Lehmann

5. März 2014

Inhaltsverzeichnis

Kapitel 1

Entwicklerhandbuch

1.1 Installation

Für die vollständige Lauffähigkeit unserer finalen Abgabe, müssen folgende Programme installiert sein:

- Play
- node.js
- Java 1.7
- Google Chrome

Zum Starten des Projekts muss zunächst das Play-Backend gestartet werden. Das kann getan werden, indem aus dem Projektverzeichnis heraus Play mit `play run` via Konsole gestartet wird. Nachdem der Server fertig geladen hat, muss die Seite einmalig über die Adresse `http://localhost:9000` gestartet werden. Bei diesem Aufruf werden die Jiac-Agenten initialisiert und gestartet. Wird die Seite mehrmals über diese URL geladen, werden die Jiac-Agenten mehrfach gestartet. Das kann zu undefiniertem Verhalten führen und sollte deshalb vermieden werden. Nach dem ersten Aufruf, wechselt der Status zu `http://localhost:9000/index.html#/nouser`. Ab sofort reagiert die Wall auf einkommende Nachrichten und ändert ihren Status selbstständig.

1.2 Projektstruktur

1.2.1 Allgemeiner Aufbau

Aus diversen Gründen haben wir uns dazu entschieden das Frontend mit dem, von Google entwickelten, Javascript-Framework AngularJS[1] zu entwickeln. die komplette Frontendimplementierung befindet sich im Unterordner `public/angular/app`. Die Widgets haben wir als Angular-Directives implementiert und diese befinden sich im `scripts/directives`-Ordner. Allgemeine Funktionen, welche die gesamte Applikation beziehungsweise den gerade relevanten Teil der Applikation betreffen, werden in den Controllern realisiert. Für häufig genutzte und ausgliederbare Funktionalität, benutzen wir die Services. Ein weiterer zentraler Bestandteil unserer Applikation ist der AngularUI Router[2] welcher für die Anzeige und den Wechsel der einzelnen Zustände zuständig ist. Sämtliche visuell relevanten Codeteile befinden sich in dem Unterordner `views`.

Unser Backend-Code ist im `app`-Verzeichnis abgelegt. Die Aufgabe des Backends besteht im wesentlichen darin, sich um die Kommunikation mit anderen Gruppen des Projekts via Jiac zu kümmern und das Frontend mit Daten zu versorgen. Weiterhin stellt es der Wallapplikation sowie den Mobilgeräten, welche zur Bedienung ebenjener verwendet werden die Websockets als Kommunikationskanal zur Verfügung.

[1] http://angularjs.org/
[2] http://github.com/angular-ui/ui-router

1.2.2 Frontend

Controllers

AuthCtrl Der Auth-Controller ist aktiv, wenn sich die Applikation in einem für die Nutzererkennung relevanten Zustand befindet. Also wenn sie anzeigt, dass gerade eine Erkennung durchgeführt wird oder ein Nutzer als bekannt oder unbekannt identifiziert wurde. Er stellt unter Anderem die Funktionen `startTraining` sowie `recognizeAgain` bereit, welche über Buttonklicks aufgerufen werden können und die Einleitung eines Trainings- oder Erkennungsprozesses durch das Backend initiieren. Weiterhin stellt er die, für die Anzeige des QR-Codes, relevanten Daten zur Verfügung.

MainCtrl Der Main-Controller ist in unserer Applikation praktisch der root-Controller. Er wird beim Start als erstes geladen, empfängt `ADD_USER` und `REMOVE_USER` Nachrichten unseres Backends und kümmert sich um die korrekte Zustansänderung der Applikation.

MobileCtrl Der Mobile-Controller ist auf allen, sich auf der Mobilseite `http://localhost:9000/index.html#/mobile` befindenden, Geräten aktiv. Er wartet auf Nachrichten von der Wall, mit der er gepaired ist und initiiert Zustandsänderungen auf der Angularinstanz, welche auf dem Mobilgerät aktiv ist. Weiterhin sendet er, sofern das `modal`-Objekt verändert wurde, das neue Objekt an die Wall, sodass sich die Nutzereingaben auf dem Smartphone direkt im, auf der Wall eingeblendeten, Modal-Fenster verfolgen lassen. Initial befindet das Mobilgerät im Zustand `wrapper.mobile.navigation`. Das bedeutet zum Einen, dass sowohl der Main- als auch der Mobil-Controller aktiv sind und zum Anderen, dass in der Subview der `views/mobile.html` die `views/-widgets/mobile/mobile.navigation.html` geladen wird. Diese bietet direkten Zugriff auf die Core-Features der Applikation vom Mobilgerät aus. Das heißt, man kann ohne auf der Wall navigieren zu müssen, Funktionen wie „Facebook-Login“ oder „Add Calendar Entry“ ausführen.

ModalCtrls Die Modal-Controller sind jeweils aktiv, sobald auf der Wall das entsprechende Eingabemodal für die angeforderte Funktionalität angezeigt wird. Sie kommunizieren mit dem Mobilgerät über Websockets und initiieren dadurch gewünschte Zustandsänderungen im MobileCtrl auf dem Mobilgerät. Die gesamte Funktionalität des ModalSocialCtrl, ist in der finalen Abgabe jedoch nicht erreichbar, da das Posten von anderen Gruppen nicht unterstützt wurde.

SettingsCtrl Der Settings-Controller ist aktiv, sobald in das Settings-Menü navigiert wurde. Das ist lediglich mit einem Mausklick auf den kleinen Button oben rechts möglich. Für den normalen Betrieb muss das Settings-Menü nicht aufgerufen werden. Es bietet sich aber für Testzwecke an, hier bestimmte Aufgaben manuell auslösen zu können (wie z.B. Facebook-Logout). Da das Settings-Menü für den User nicht sichtbar ist, finden sich hier auch noch Buttons, welche nicht mehr gebraucht werden, oder nicht mehr funktionieren.

TestCtrl Der Test-Controller simuliert lediglich das Eintreffen von `ADD_USER` bzw. `REMOVE_USER` Nachrichten mit verschiedenen User- und Nite-IDs um die, sich im Main-Controller befindende, Logik zu testen oder einzelne Widgets mit Testdaten zu versorgen. Dazu werden die Nachrichten auf den entsprechenden Kanälen der Websockets versendet.

Directives

Calendar Das Calendar-Widget holt sich seine Daten über einen eigenen „CALENDAR“-Channel des Websockets. Außerdem ist hier die Funktionalität zum Hinzufügen eines eigenen Kalendereintrags vorhanden, welche dann das entsprechende Modal öffnet, dessen Controller sich dann um die Kommunikation mit dem Mobilgerät kümmert. Weiterhin benutzen wir Bootstrap[3] Popovers, um weitere Detailinformationen zu einem Kalendareintrag anzuzeigen. Dazu dient die Funktion `showCalendarEntry` welche das Popover öffnet und mit diesem die entsprechenden Daten übergibt. Die zusätzlichen Directives `widgetCalendarSmall`, `widgetCalendarMiddle` bzw. `widgetCalendarBig`

[3] http://getbootstrap.com/

dienen lediglich dazu eine modularere, übersichtlichere und elegantere Notation in den HTML-Files zu ermöglichen.

Debug Das Debug-Widget sollte in der finalen Version selbstverständlich nicht mehr angezeigt werden, da es, wie der Name schon andeutet, lediglich zu Debugzwecken verwendet wird. Es zeigt die Anzahl aller bekannten sowie unbekannten Nutzer und die drei zuletzt erkannten Gesten an. Weiterhin zeigt es das für die Gesichtserkennung verwendete Bild, sowie User- und NiteID des jeweiligen Nutzers an.

Fair Das Messe-Widget, zeigt lediglich hart codierte Daten an. Diese werden im in der `views/fair.html` übergeben. Logik zum holen von Daten ist noch nicht implementiert, da andere Gruppen dieses Szenario nicht unterstützen, ließe sich aber mit geringem Aufwand aus einem der anderen Widgets ableiten.

Mail Das Mail-Widget holt sich seine Daten über einen eigenen „MAIL“-Channel des Websockets. Der Rest der Funktionalität ist beinahe Äquivalent zum Calendar-Widget.

Widget Das Widget-Directive ist eine Art Wrapper dem wir die Daten für das eigentliche Widget sowie einen Widget-Type übergeben. Es fügt dann einen HTML-Tag mit typspezifischen CSS-Klassen um den eigentlichen Widget-Tag herum ein. Das hat den Vorteil, dass man beim Hinzufügen von Widgets in der `views/main.html` weniger beachten muss.

Maps TODO

News Das News-Widget wartet lediglich auf Daten in dem „NEWS“-Channel des Websockets und bietet eine Funktion um einen Newsbeitrag in einem Modal detailiert anzuzeigen.

Personal Das Personal-Widget zeigt persönliche Informationen des Users an. Diese holt es sich über den „FACEBOOK“-Channel des Websockets. Weiterhin kann man sich den QR-Code, welcher auf die Mobilseite verlinkt mit einem Modal in groß anzeigen lassen.

Social-Comparison TODO

Social Das Social-Widget zeigt dem User aktuelle Facebook-Posts aus seinem Facebookstream an. Die Daten bekommt es aus dem „SOCIAL“-Channel. Weiterhin stellt es die Möglichkeit zur Verfügung detailierte Informationen zu einem Post, wie Anzahl an Likes und Kommentaren, sowie die ersten Kommentare in einem Popover anzuzeigen. Auch das Liken eines Facebookposts ist möglich.

Todo Das Todo-Widget zeigt die ToDos an, welche auf dem „TODO“-Channel ankommen. `showTodo` kümmert sich hier um die Anzeige detailierter Informationen zu einem ToDo.

Services

CSS-Service Hier wird ein Array von CSS-Klassen erstellt, was für jedes Widget eine eigene CSS-Klasse erstellt, die die Farbendarstellung im Widget beschreibt.

Radial-Service TODO

1.3 Erweiterung

Kapitel 2

Nutzeranleitung

Kapitel 3

Projektbericht