Yamdu

ARRI

Technische Dokumentation

Semesterprojekt „Yamdu“

Technikteam:
Sebastian Antosch
Christian Baur
Patrick Bumiller
Susanne Rauchbart
Christian Reichart
Raymund Zacharias

Wintersemester 2014/15
Hochschule Augsburg

Gemeinschaftsprojekt der Studiengänge
Kommunikationsdesign und Interaktive Medien

Betreuung:
Prof. Robert Rose
Prof. Dr. Michael Kipp

Im Auftrag der Seriotec GmbH,
vertreten durch:
Florian Reimann
Peter Berchtold
Stefan Kammler
Dr. Robert Straßer

Inhaltsverzeichnis

Inhaltsverzeichnis

1.
Hintergrund

Was steckt dahinter?

Einführung

Im Zeitraum des Wintersemesters 14/15 entstand innerhalb einer Projektarbeit die mobile Yamdu-App für Android und iOS.

Die Beteiligten setzten sich aus Studenten der Fachrichtungen Kommunikationsdesign und Interaktive Medien zusammen.

Die Anwendung macht die aus der bereits vorhandenen Yamdu-Webapplikation der Seriotec GmbH, die als Auftraggeber und Projektpartner fungiert, bekannten Funktionen auch mobil verfügbar.

Da das Endprodukt und die Weiterentwicklung an die Vertreter von Seriotec übergeben wird, ist diese Dokumentation entstanden, die, ergänzend zur Code-Dokumentation, einen ersten Einblick in das System und dessen Arbeitsweise ermöglicht. Zudem werden häufige oder esentielle Arbeitsschritte aufgeführt, um den Entwicklunseinstieg zu erleichtern. Auch werden aus der Sicht des Projekt-Teams die nächsten Schritte und wichtigsten Hinweise erläutert.

Frameworks

Cordova

Ionic

Angular JS

Sass

Apache Cordova ist ein Framework zur Erstellung von Hybrid-Apps auf Basis von Webtechnologien.

Ionic ist ein Frontend-Framework speziell für die Umsetzung von Hybrid-Apps mit Hilfe von Webtechnologien.

Ionic basiert auf AngularJS und kann mit Sass an das gewünschte Look & Feel angepasst werden.

Frameworks

Apache Cordova:

Cordova ist ein Framework zur Erstellung von Hybird-Apps.

Das visuelle Erscheinungsbild und die Funktionalität der Anwendung entsteht mittels Webtechnologien, anschließend können die Apps wie native Apps publiziert werden (.ipa und .apk).

Ionic:

Ionic ist ein Frontend-Framework zur Erstellung mobiler Web-Apps (Hybrid-Apps) auf Basis von HTML5, CSS3 und JavaScript.

Ionic stellt bereits viele Komponenten zur Verfügung, die ein schlichtes User-Interface und schnelle Ergebnisse ermöglichen.

Es basiert auf dem JavaScript Framework Angular und die Gestaltung kann über SASS, mithilfe von, von Ionic bereitgestellten, Sass-Variablen angepasst werden.

Zusätzlich wird eine Open Source Bibliothek mit über 440 Icons bereitgestellt.

Frameworks

Sass:

Sass erzeugt CSS. In Sass können Regeln geschachtelt werden.

Auch Media Queries können in Selektoren geschachtelt werden.

Sass erlaubt, im Gegenzug zu reinem CSS, auch die Verwendung von Variablen, wodurch Grundaspekte der Gestaltung schneller und einfacher angepasst werden können.

Sogenannte Mixins erlauben die wiederholte Referenzierung ganzer Code-Abschnitte, dabei können Argumente übergeben werden.

AngularJS:

Angular verfolgt den Ansatz eines MVCS-Patterns (Model-View-Controller-Service). Dabei sind die genannten Bereiche voneinander getrennt und können unabhängig voneinander agieren, woraus eine hohe Flexibilität und Erweiterbarkeit resultiert.

Zur Auswahl von Elementen verwendet Angular jQuery Light. Sollte das vollwertige jQuery in die Seite eingebunden sein, wird dieses anstatt der Light Version verwendet (für mobile nicht empfohlen).

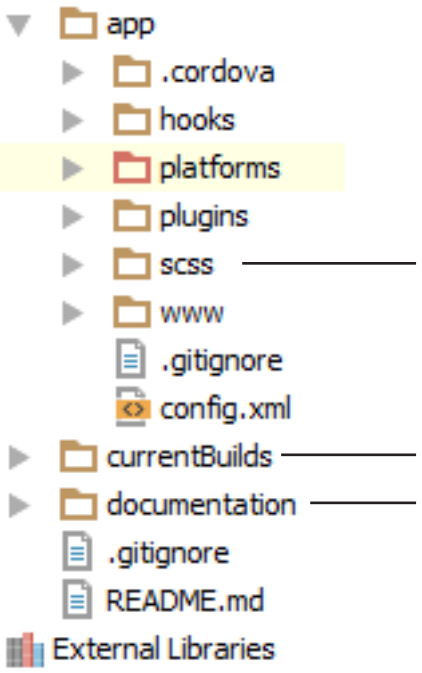
Angular ermöglicht die Erstellung von Controllern, Direktiven, Services, Verwendung von Routen und die Kommunikation zwischen Scopes.

2.

Struktur

Wo finde ich was?

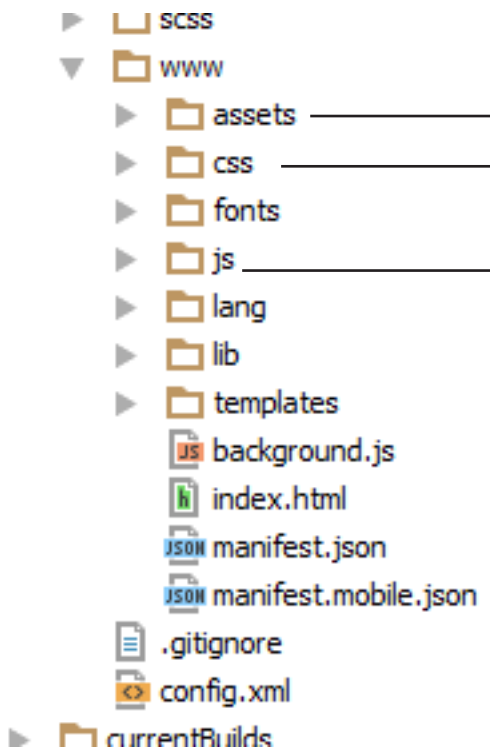
Verzeichnisstruktur Yamdu



- von Ionic bereitgestellte SCSS-Files
- selbst erstellte Yamdu-SCSS-Files
- diese werden alle bei der Auslieferung der App zu einem CSS-File kompiliert, dieses liegt im Verzeichnis www/css

stets aktuelle Builds für Android und iOS

- Dokumentation der Icon-Font
- generiertes JS-Doc

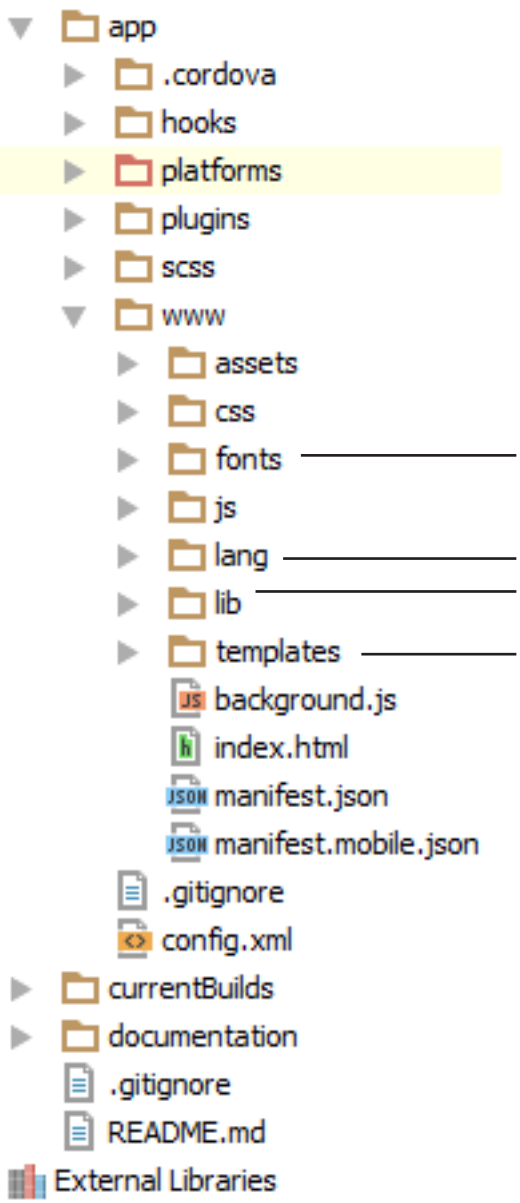


Ablage der verwendeten Bilder

kompiliertes CSS-File

- Controller, Direktiven, Filter, Services, Variablen
- zentrales JS-File in dem die Angular-Module verwaltet (registriert) werden

Verzeichnisstruktur Yamdu



- Verwendete Schriftarten
- Icons in Form eines Icon-Fonts

Dictionaries

Bibliotheken

- HTML-Files für Direktiven
- HTML-Files für die einzelnen Views

Wo passiert was?

Services

Services enthalten statusunabhängige Ablauflogik und dienen oftmals als Schnittstelle zu externe Ressourcen. Services werden als Singleton instanziiert.

Bereits von Angular zur Verfügung gestellte Services sind an dem Präfix $ zu erkennen (z.B. $http).

Lokale Datenspeicherung:

Der **Local-Storage** erlaubt die dauerhafte Speicherung und Abfrage von Daten. Diese werden mithilfe des HTML5 Local Storage auf dem Gerät gespeichert und sind damit auch offline verfügbar.

Da im HTML5 Local Storage nur Text gespeichert werden kann, werden die Daten beim Abspeichern und Auslesen automatisch entsprechend umgewandelt.

Schnittstelle zur Api:

Grundsätzlich dient der **Api-Abstraction-Layer** als Schnittstelle zur Yamdu-Api. Er kapselt die Aufrufe und cashed sie automatisch im Local-Storage.

Die URLs der Requests werden hier festgehalten. Sie können um beliebige Parameter erweitert werden, die im Funktionsaufruf entsprechend übergeben und ausgezeichnet werden müssen.

Der Api-Abstraction-Layer stellt sowohl für GET- und POST-, als auch PATCH-, PUT- und DELETE-Requests passende Funktionen bereit.

Alle einmal geladenen Daten werden im Local-Storage zwischengespeichert, um beim nächsten Aufruf und offline schnell zur Verfügung zu stehen.

Services

Offline-Verhalten:

Dank der gecacheten Requests können im Offline-Modus bereits zu einem früheren Zeitpunkt abgerufene Daten aus dem **Local-Storage** abgerufen werden.

Auch für das Offline-Verhalten bei der Bearbeitung, Erstellung oder Löschung von Inhalten ist der **Api-Abstraction-Layer** zuständig. Er erstellt eine Queue von Requests, die nacheinander abgearbeitet werden. So werden bei erneutem Zustandekommen einer Internetverbindung die Post-Requests, die im Offline-Modus getätigten Änderungen übertragen.

Die Queue wird ebenfalls im Local-Storage abgelegt, damit offline getätigte Änderungen nach dem Neustart der Anwendung nicht verloren gehen.

Start der Anwendung:

Beim Start der Anwendung wird überprüft ob bereits ein Nutzer eingeloggt ist und gegebenenfalls dessen Einstellungen geladen, wie beispielsweise die Spracheinstellung. Sollte kein Nutzer eingeloggt sein, wird auf den Login-Screen verwiesen.

Sollte ein Projekt ausgewählt sein, wird der Nutzer direkt auf das Dashboard dieses Projektes geleitet, falls nicht, auf die Projektübersicht.

Dies geschieht im **Startup-Service**.

Objekt-Markierung:

Die Markierung der Objekte kann im Service **MarkObject** gesetzt und gelöscht werden.

Objekt-Entfernung:

Auch die Entfernung von Objekten wird in einem Service realisiert, dieser heißt **DeleteObject**. Vor dem Löschen des Objektes erscheint eine warnende Rückfrage, ob sich der Nutzer mit seiner Entscheidung sicher ist.

Direktiven

Direktiven sind benutzerdefinierte HTML-Elemente und -Attribute.

Es gibt bereits von Angular und Ionic vordefinierte Direktiven, die an entsprechenden Namensräumen erkennbar sind (z.B. ng-repeat oder ion-item).

Menüelemente:

Die Direktive **menu-item** ist grundsätzlich dafür zuständig Kacheln innerhalb des Hauptmenüs zu erzeugen und gegebenenfalls mit einer Vorschau zu versehen (z.B. Bekanntmachungen, Aufgaben und Terminen).

Reiter:

Die Kontrolle und Ansicht der Tabs werden in der Direktive **yamdu-tabs** definiert.

Scrollverhalten:

Die Direktive **scroll-detect** erkennt das Scrollen und meldet dies an den Service **footerBar-Observer**.

Wird nach oben gescrollt blendet der footer BarObserver die footerBar ein (Annahme, Nutzer möchte Navigieren), beim scrollen nach unten wird sie ausgeblendet (Annahme, der Nutzer möchte den Inhalt lesen).

Akkordeon:

Ein Akkordeon ermöglicht das Aus- und Einklappen von Textabschnitten. Die Funktion des Akkordeons wird in der Direktive **accordion** definiert.

Direktiven

Kommentare:	Kommentare lassen sich über die Direktive **comments** erstellen, bearbeiten und löschen, diese erzeugt auch die Kommentaransicht.
Datums-Auswahl:	Die Direktive **datetime-select** erzeugt einen Popup-Kalender, über den der Nutzer ein Datum und Uhrzeit auswählen kann.
Benutzer-Auswahl:	Die Auswahl von Stabmitgliedern oder Gruppen wird an mehreren Stellen benötigt, die zugehörige Direktive **user-select** ermöglicht diese über eine Liste.
Textkarten:	Die Textkarten werden im Projekt an vielen Stellen zur Darstellung von Inhalten genutzt. Sie werden mithilfe der Direktive **yamdu-card** realisiert. Diese enthält Optionen für vordefinierte Größen, Farben, Thumbnails und Icons, sowie eine Swipe-Funktion die den Zugriff auf möglicherweise hinter der Karte verborgene Buttons ermöglicht.

Filter

Datumsformate:

Durch den Filter **localizedEvent** wird die Zeit- und Datumsangabe, welche von der Yamdu-Api (einheitlich) bereitgestellt wird, abhängig vom Sprachraum (den Spracheinstellungen des Nutzers) angepasst dargestellt.

Darüber hinaus wird eine angenehm lesbare Ausgabe von Start- und Endzeiten bzw. ganztägigen Terminen ohne unnötige Doppelnennungen erzeugt.

Durch den Filter **yamduDate** werden eingegebene Zeit- und Datumsangabe in das Format umgewandelt, in dem es in der Yamdu-Api abgelegt wird.

Farbformat:

Der Filter **color2hex** wandelt die verbal festgelegten Farbbegriffe, die beispielsweise in der Api als Projektfarbe hinterlegt sind, in ihren entsprechenden Hexadezimalwert um.

Textvorschau:

Der Filter **cut** kürzt lange Texte wortweise auf ein gegebenes Limit und ergänzt sie um „ …“.

Variablen

Konstanten:

In den **constants** werden die wichtigsten Konstanten festgehalten, Werte die sich nicht ändern und oft verwendet werden.

Bisher sind das zum Beispiel die URL für die Requests, die voreingestellte Sprache oder das Intervall für die automatische Aktualisierung des Dashboards.

Globale Variablen:

In den **globals** werden globale Variablen verwaltet, die häufig verwendet werden und von vielen unterschiedlichen Controllern/Services angesprochen werden müssen, so zum Beispiel die aktuelle Projekt-ID, der Name des Nutzers und Informationen über den Zustand der Anwendung (Login- und Netzwerkstatus).

Controller

In einem Controller wird ein statusbezogene Ablauflogik definiert. Die Controller werden zu einem Modul zusammengefasst.

Die Verbindung von View und Model ist bidirektional (Benutzereingaben wirken sich auf das Modell aus, aber programmatische Änderungen am Modell auch auf die Benutzersicht).

Templates

In den Templates werden die Ansichten definiert, d.h. deren HTML Struktur festgehalten. Diese Templates werden an entsprechender Stelle in die Anwendung geladen.

Pages:

In dem Verzeichnis Pages werden alle Templates für Appstati hinterlegt, wie z.B. Listen-, Detail- und Editieransichten.

Die hier hinterlegten Templates müssen in der Datei app.js an einen Appstatus gebunden werden.

Direktiven:

Die Struktur der Direktiven wird in deren entsprechenden HTML-Dateien definiert.

Sonstiges

Scope-Kommunikation:

Jeder Controller besitzt ein $scope Objekt. Dieses kapselt die Funktionen und Daten des Controllers.

Das $scope Objekt stellt $emit und $broadcast Methoden bereit, zur Kommunikation von Controllern untereinander oder mit Services.

Mit der $emit Methode können Nachrichten an alle übergeordneten Scopes und mit der $broadcast Methode an alle untergeordneten Scopes gesendet werden.

Um reagieren zu können müssen sich Controller und Services mit der $on Methode für den Nachrichtentyp registrieren.

Einbindung:

Mit Hilfe von doppelt geschwungenen Klammern {{ ... }} können JavaScript-Ausdrücke im HTML eingebettet werden.

Mit dem PipeOperator | können diesen Filter hinzugefügt werden.

Lokalisierung:

Die Lokalisierung wird mithilfe des Moduls angular-translate realisiert, welches den Filter translate zur Verfügung stellt.

Die Dictionaries wurden gegenüber den in der Web-App verwendeten abgeändert. Die Auszeichnung der möglichen Variablen wird nicht mit __var__ sondern {{var}} gekennzeichnet.

Sonstiges

Schriften/Icons:

Die Icons werden über eine Icon-Font eingebunden, hierzu findet sich eine Dokumentation mit der Bezeichnung **Iconfont_documentation** im HTML-Format in dem Dokumentationsverzeichnis. Die Icon-Font ist, wie auch der Schriftsatz (OpenSans), im Verzeichnis **Fonts** zu finden.

Bibliotheken:

Die verwendeten Bibliotheken finden sich im lib-Verzeichnis. Derzeit sind die Bibliotheken ionic, angular-translate, angular-bootstrap-datetime und videogular eingebunden.

Plugins:

Die Plugins müssen über **cca** installiert werden, derzeit verwendet werden **com.ionic.keyboard, org.apache.cordova.file-transfer** und **org.apache.cordova.contacts**.

3.

Handhabung

Wie mache ich was?

Installation

NodeJS:

NodeJS muss heruntergeladen und über die Kommandozeile installiert werden:

```
npm install -g learnyounode
```

SDK:

Die mobilen Software-Developer-Kits (SDKs) müssen heruntergeladen werden:
- Download und Installation von stand-alone Android SDK Tools
- damit entsprechende SDKs herunterladen (SDK Manager als Administrator ausführen)
- falls nicht vorhanden zuvor JDK installieren

ANT Library:

Für den späteren Build-Prozess muss die ANT Library eingebunden werden:
- Android SDK zum System Path hinzufügen
- Apache ANT Library herunterladen und entpacken
- Umgebungsvariablen bearbeiten:
  Neue Systemvariable:
  Name: ANT_HOME
  Wert: ...\ApacheANT\apache-ant-1.9.4
  (hier Pfad einfügen!)
  Neue Systemvariable:
  Name: JAVA_HOME
  Wert: ...\Java\jdk1.8.0_25
  (hier Pfad einfügen!)
- Systemvariable Path bearbeiten,
  folgendes Anfügen: ;%ANT_HOME%\bin

Cordova:

Cordova muss über die Kommandozeile installiert werden:

```
npm install -g cordova
```

CCA:

Auch CCA wird über die Kommandozeile installiert:

```
npm install -g cca
```

SASS:

Im Anschluss muss SASS installiert werden.

Installation

Plugins: Die verwendeten Plugins werden mithilfe von CCA über die Kommandozeile installiert:

```
cca plugin add com.ionic.keyboard
cca plugin add org.apache.cordova.contacts
cca plugin add org.apache.cordova.file-transfer
```

Veröffentlichung

Erstellung einer APK: Zur Erstellung einer APK muss über die Kommandozeile in das Projekt-Verzeichnis navigiert werden, um folgende Aufrufe zu verwenden:

```
cordova platform add android
cordova build
```

Die erzeugte APK liegt im Projekt-Verzeichnis/platforms/android/ant-build und heißt CordovaApp-debug.apk.

Anschließend muss diese APK nur noch auf dem Endgerät installiert werden.

Signierung der APK: Für die Erzeugung der **finalen** APK muss diese signiert werden, hierfür wird in der Kommandozeile folgendes ausgeführt:

```
cordova build --release
```

Die erzeugte APK liegt im Projekt-Verzeichnis/platforms/android/ant-build und heißt CordovaApp-unsigned.apk.

Zusätzlich muss zur Systemvariable „Path“ hinzugefügt werden:
;“%JAVA_HOME%\bin“

jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -keystore meinkeystorename.keystore platforms\android\ant-build\CordovaApp-release-unsigned.apk keyname

zipalign.exe von D:\Programme\AndroidSDK\

Veröffentlichung

Um die erzeugte APK zu signieren muss in der Kommandozeile (als Administrator) folgendes ausgeführt werden:

```
keytool -genkey -v -keystore meinkeystorename.keystore -alias
keyname -keyalg RSA -keysize 2048 -validity 10000
```

```
jarsigner -verbose -sigalg SHA1withRSA -digestalg SHA1 -
       keystore meinkeystorename.keystore platforms\android\
       ant-build\CordovaApp-release-unsigned.apk keyname
```

Im Anschluss muss die zipalign.exe im Verzeichis...\Programme\AndroidSDK\build-tools\21.0.2 liegende, nach ..\Programme\AndroidSDK\tools kopiert werden.

```
zipalign -v 4 platforms\android\ant-build\CordovaApp-release
       -unsigned.apk testapp.apk
```

Die fertige App liegt nun im Projektverzeichnis und hat den Namen testapp.apk.

Filter

color2hex: Der color2hex Filter konvertiert die Projektfarbenkennung in den passenden Hex-Farbcode

Verwendung HTML

```
{{ response.projectcolor | color2hex }}
{{ ‚green‘ | color2hex }}
```

Verwendung Javascript

```
angular.module(‚starter.controllers‘)
    .controller(‚BeispielCtrl‘, [‚color2hexFilter‘,
function(color2hexFilter){

       var color = ‚green‘;
       var hexcolor = color2hexFilter(color);
       alert(hexcolor); // #319643
}]);
```

yamduDate: Der yamdudate Filter konvertiert ein Datum ins vom Server angeforderte ISO-Format

Verwendung Javascript

```
angular.module(‚starter.controllers‘)
    .controller(‚BeispielCtrl‘, [‚yamdudateFilter‘,
function(yamduFilter){

       var date = new Date();
       var isodate = yamdudateFilter(date);
       alert(isodate); // 2015-01-28T15:36:11.667Z
}]);
```

Filter

localizedEvent:

Der localized Event Filter nimmt ein Startdatum, optional ein Enddatum und optional eine allday-flag entgegen und gibt ein schön formatiertes Datum ohne Doppelnennung des Tages in der ausgewählten Sprache aus.

Verwendung HTML

```
{{startdate | localizedEvent: enddate : allday}}

{{ ‚2015-01-27T14:36:11.667Z‘ | localizedEvent :
‚2015-01-28T15:36:11.667Z‘ : false }}
       // 28.1.2015 14:36 Uhr - 27.1.2015 15:36 Uhr
{{ ‚2015-01-27T14:36:11.667Z‘ | localizedEvent :
‚2015-01-27T15:36:11.667Z‘ : false }}
       // 27.1.2015 14:36-15:36 Uhr
{{ ‚2015-01-27T14:36:11.667Z‘ | localizedEvent : null : true }}
       // ganztägig am 27.1.2015
{{ ‚2015-01-27T14:36:11.667Z‘ | localizedEvent}}
       // 27.1.2015 14:36 Uhr
```

Verwendung Javascript

```
angular.module(‚starter.controllers‘)
    .controller(‚BeispielCtrl‘, [‚localizedEventFilter‘,
function(localizedEventFilter){

       var startdate = ‚2015-01-27T14:36:11.667Z‘
       var enddate = ‚2015-01-27T15:36:11.667Z‘
       var allday = false;
       var localizedEventText = localizedEventFilter(startdate,
              enddate, allday);
       alert(localizedEventText); // 27.1.2015 14:36-15:36 Uhr
}]);
```

Filter

cut:

Der cut Filter beschneidet Texte auf eine gegebene Länge, auf Wunsch auch wortweise.

Verwendung HTML

```
{{text | cut: wordwise : length}}
{{ ‚Dies ist ein toller Text‘ | cut : true : 10 }}     // Dies ist
```

Verwendung Javascript

```
angular.module(‚starter.controllers‘)
    .controller(‚BeispielCtrl‘, [‚cutFilter‘, function(cutFilter){

        var text = ‚Dies ist ein toller Text‘;
        var wordwise = true;
        var length = 10;
        var cutText = cutFilter(text, wordwise, length);
        alert(cutText); // Dies ist
}]);
```

pageCountFormat:

Der pageCountFormat Filter nimmt den Pagecount der Drehbuchabschnitte im Format 1_1_8 entgegen und gibt eine lesbare Zahl aus.

Verwendung HTML

```
{{ pagecount | pageCountFormat }}
{{ ‚1_1_8‘ | pageCountFormat }}            // 1 1/8
```

Verwendung Javascript

```
angular.module(‚starter.controllers‘)
    .controller(‚BeispielCtrl‘, [‚pageCountFormatFilter‘,
function(pageCountFormatFilter){

        var pagecount = ‚1_1_8‘;
        var formattedPageCount = pageCountFormatFilter(pagecount);
        alert(formattedPageCount); // 1 1/8
}]);
```

Direktiven

accordion:

Erzeugt einen aus- und einklappbaren Akkordeoncontainer

Parameter:
- **title**: dictionary-key für den Titel
- **count** (optional): Eine zusätzlich anzuzeigende Zahl

Verwendung

```
<accordion title=“Toller Titel“ count=“3“>
       Toller Inhalt
</accordion>
```

comments:

Zeigt Kommentare und ein Kommentierformular zu einem Objekt

Parameter:
- **id**: die ID des Objekts
- **itemtype**: der Typ des Objekts

Verwendung

```
<comments id=“1337“ itemtype=“‘announcement‘“></comments>
```

datetime-select:

Zeigt ein Auswahlfeld für ein Datum, mit Popup-Kalender

Parameter:
- **ng-model**: variable, in der das Datum gespeichert werden soll
- **label**: anzuzeigendes Label

Verwendung

```
<datetime-select ng-model=“create.start“ label=“‘calendar.
start‘|translate“></datetime-select>
```

Direktiven

menu-item:

Rendert ein Menüitem (Kachel)

Parameter {menuicon, new, linkedstate, review}:
- **menuicon**: icon-klasse
- **new**: Anzahl Änderungen die in der Flag angezeigt werden
- **linkedstate**: der App-State auf den die Kachel verlinken soll
- **review**: bei normaler Kachel nichts, ansonsten key für die anzuzeigende Preview (announcements, tasks, calendar)

Hinweis: Bei dieser Direktive müssen alle Parameter als Variablen gesetz oder explizit als String ausgewiesen werden.

Verwendung

```
<menu-item title="item.title"
        menuicon="item.icon"
        new="item.new"
        linkedstate="item.linkedstate"
        preview="item.preview"
</menu-item>
```

scroll-detect:

Attribut-Direktive für ion-content container - sorgt dafür, dass beim Scrollen des Containers die Footerbar ausgeblendet/eingeblendet wird.

Verwendung

```
<ion-content scroll-detect>
...
</ion-content>
```

Direktiven

user-select:

Erstellt einen Dialog, der es ermöglicht Nutzer und Nutzergruppen auszuwählen

Parameter:
- **selected-items**: Die Variable, in die die ausgewählten User/Gruppen gespeichert werden
- **label**: Anzuzeigendes Label
- groups: boolean, ob Gruppen ausgewählt werden können
- **members**: boolean, ob Member ausgewählt werden können
- **title**: Titel des Auswahlpopups

Verwendung

```
<user-select
       selected-items="create.associations"
       label="'announcements.recipients'|translate"
       groups="true"
       members="true"
       title="'tasks.delegatee'|translate"
></user-select>
```

yamdu-tabs:

Erstellt aus einem TabSet eine Ansicht mit swipebaren Tabs und nachfolgender Tableiste.

Parameter:
- **tabs**: Ein Set von Tabs, welches angezeigt werden soll

Tabset:
Array von Objekten mit folgenden Keys:
- **title**: dictionary-key für den Tabtitle
- **secondtitle**: {optional} zweiter dictionary-key zum Anhängen an den Tabtitle (für Doppeltabs auf dem Dashboard)
- **templateUrl**: url zum anzuzeigenden Tabtemplate
- **isMasterDetail**: {optional} true, wenn es sich um eine Master-Detail View handelt (wichtig für korrektes Scrollverhalten)

Direktiven

HTML-Template:

```
<ion-view ng-controller="BeispielCtrl as beispiel" title="Beispiel!">
    <titlebar-buttons></titlebar-buttons>
    <yamdu-tabs tabs="beispiel.tabs"></yamdu-tabs>
</ion-view>
```

Controller:

```
angular.module(,starter.controllers')
    .controller(,BeispielCtrl', [,$scope', function($scope){

       this.tabs = [
              {
               title: ,dashboard.myStream',
               templateUrl: ,templates/pages/dashboard/foryou.html'
               },
               {
               title: ,dashboard.departmentOverview',
               templateUrl: ,templates/pages/dashboard/whatstodo.html'
               },
               {
               title: ,dashboard.markedObjects',
               templateUrl: ,templates/pages/dashboard/marked.html'
               },
               {
               title: ,dashboard.projectStream',
               templateUrl: ,templates/pages/dashboard/goingon.html'
               }
       ];
}]);
```

Direktiven

yamdu-card:

Erstellt eine Karte für Inhalte mit Thumbnail/ Icon und Swipefunktion, welche Optionsbuttons enthüllt.

Parameter:
- **color**: {optional} Farbklasse, red/yellow/green/ conflicted/pending/ok - erstellt eine farbige Border am linken Rand der Karte.
- **customcolor**: {optional} HEX-Farbcode, erstellt eine Border am linken Rand der Karte, der eine eigens definierte Farbe hat.
- **thumbnail**: {optional} Link zum Thumbnail
- **icon**: {optional} icon class Name
- **optionbuttons**: {optional} Array von Options- buttons, sollte vom cardButtonProvider Service eingeholt werden.
- **urgent**: {optional} Dringlichkeitsindikator, true / false / number
- **marked**: {optional} Markiert-Flagge true / false
- **new**: {optional} Neu-Flagge true / false
- **size**: {optional} feste Kartengröße default / middle / small
- **item**: {optional} - {required} wenn option- buttons übergeben werden
- **active**: {optional} soll die Karte gehighlighted werden true/false

Verwendung

```
<yamdu-card
      ui-sref="app.taskDetail({ id: task.id })"
      marked="task.is_marked"
      new="task.has_changed"
      optionbuttons="tasks.buttons"
      item="task"
      color="{{task.is_completed?,green':task.is_overdue?,
             red':,yellow'}}"
      >

      <h2>{{task.title}}</h2>
      <p>Toll!</p>
</yamdu-card>
```

4.

Hinweise

Was ist noch wichtig?

Nächste Schritte

Das Abspielen der Videos im Offline-Modus muss noch für Android umgesetzt werden.

Die offene Frage bezüblich der Sicherheit der heruntergeladenen Videos auf dem Android System ist noch zu klären.

Der Videoplayer soll noch um die Funktionen Zeitlupe und Erstellung von Kommentaren zu genauen Stellen des Timecodes erweitert werden.

Die weiteren Departments wie beispielsweise Maske und Kostüm müssen noch implementiert werden.

Die Herstellungsplanung und Drehplanung müssen noch umgesetzt werden.

Die Kalenderansicht kann optimiert werden.

Die fertige App, oder ein veröffentlichbarer Zwischenstand muss im AppStore veröffentlicht werden.

Zusammenfassung

Die im Wintersemester 14/15, innerhalb einer Projektarbeit entstandene mobile Yamdu-App für Android und iOS deckt die Erwartungen des Projekt-Teams mit vollster Zufriedenheit. Die implementierten Funktionen entsprechen dem geforderten Umfang und bieten genug Flexibilität und Erweiterbarkeit um auch die restlichen gewünschten Funktionen in Eigenregie der Seriotec GmbH umzusetzen.

Dank der durchdachten und konsistenten Konzeption, sowohl des Designs als auch der Anwendungsstruktur und -logik, dient der übergebene Projektstand als solide Grundlage für eine erfolgreiche Veröffentlichung und Weiterentwicklung der Anwendung.

Die Zusammenarbeit mit der Seriotec GmbH verlief sehr erfreulich und war für alle Beteiligten eine interessannte und lehrreiche Erfahrung.

Das Technik-Team

write@sebastian-antosch.de

Sebastian Antosch

Sebastian Antosch ist schon seit seinem 18. Lebensjahr als selbstständiger Webdesigner tätig. Nach seinem Abitur im Jahr 2011 widmete er sich dem Studium der Interaktiven Medien und ergänzt seit 2013 das Team einer mittelständischen Werbeagentur als Webentwickler. Seine Schwerpunkte liegen beim Einsatz der CMS Joomla und TYPO3, sowie der Frontendentwicklung mit HTML5, CSS3, jQuery und dem Bootstrap-Framework.

Schwerpunkte bei der Entwicklung der YAMDU-APP waren die Grundstruktur, der ApiAbstractionLayer, Filter, die yamdu-card Direktive, das Dashboard, das Menü, der cardButtonProvider und der resizeObserver.

contact@christianbaur.de

Christian Baur

Christian Baur ist seit 2010 als selbstständiger Webdesigner und Webentwickler tätig und fokussierte sich auch in seinem 2011 begonnenen Studium der Interaktiven Medien zunehmend auf diese Bereiche. Schwerpunkte seiner Arbeit liegen in der Frontendentwicklung mit HTML5, CSS3 und JavaScript und dem Einsatz, der Anpassung und Erweiterung des CMS Joomla.

Seine Schwerpunkte bei der Entwicklung der YAMDU-App waren die Grundstruktur, die Services ApiAbstractionLayer (Offline-Modus), DeleteObject, MarkObject und Master-Detail Views (ListSelectionService), die Direktiven accordion, yamduTabs, comments, datetimeSelect und userSelect sowie die Bereiche Aufgaben, Bekanntmachungen und Büro.

pat-bumiller@web.de

Patrick Bumiller

Schon während seines Studiums „Interaktiven Medien“, das er im Sommer 2015 abschließt, beweist Patrick Bumiller als Gestalter und Informatiker Kompetenz. Zu seinen Schwerpunkten, die er auch in seiner Tätigkeit als Werkstudent ausführt, gehört die client- und serverseitige Webentwicklung, sowie die Entwicklung von nativen iOS Apps.

Schwerpunkte bei der Entwicklung der Yamdu-App: Navigationsstruktur, iOS- Kompatibilität, ApiAbstractionLayer, Postproduction, Fehlerbehebung

susanne.rauchbart@gmx.de

Susanne Rauchbart

Susanne Rauchbart begann nach ihrer Ausbildung zur Fachinformatikerin für Anwendungsentwicklung das Studium der interaktiven Medien an der Hochschule Augsburg, um ihrer Kreativität und konzeptuellen Fähigkeiten mehr Raum geben zu können. Ihre Interessen liegen unter anderem in den Bereichen Konzeption, Organisation und Interaktion. Ihre Tätigkeiten während des Studiums in Projekten, im Praxissemester und in ihrer Tätigkeit als Werkstudentin konnte sie sich nicht nur fachlich sondern auch persönlich weiterentwickeln.

Ihre Tätigkeitsfelder im Semesterprojekt reichten von Konzeption und Organisation, über die Rolle des Scrum-Masters und die Erstellung der technischen Dokumentation, bis hin zur Entwicklung (Tagesdispositionen, pageCountFormat, Kalender und Termine).

rayza8891@yahoo.de

Raymund Zacharias

Raymund Zacharias interessiert sich vorallem für Interface-Entwicklung und Entwicklung von Spieleanwendungen, wozu er im Studium desöfteren Gelegenheit hatte.

Im Rahmen des Designprojekts war er an der Umsetzung der App unter anderem mit Cordova und Ionic beteiligt. Bereiche hierbei waren vorallem der Projekt-Stream und Kalender. Außerdem war er für die Generierung der Quellcode-Dokumentation zuständig.