COGNITIVE DEFUSION MOBILE APPLICATION

Developing a single page application for cognitive defusion exercises

Bachelor thesis
Submitted to Prof. Oscar Nierstrasz
7 July 2015

Pascal Zaugg
pascal.zaugg@students.unibe.ch

Table of contents

1 Introduction

2 Technologies

In this part you will find a short description of all technologies used in my project and how those technologies are linked to each other.

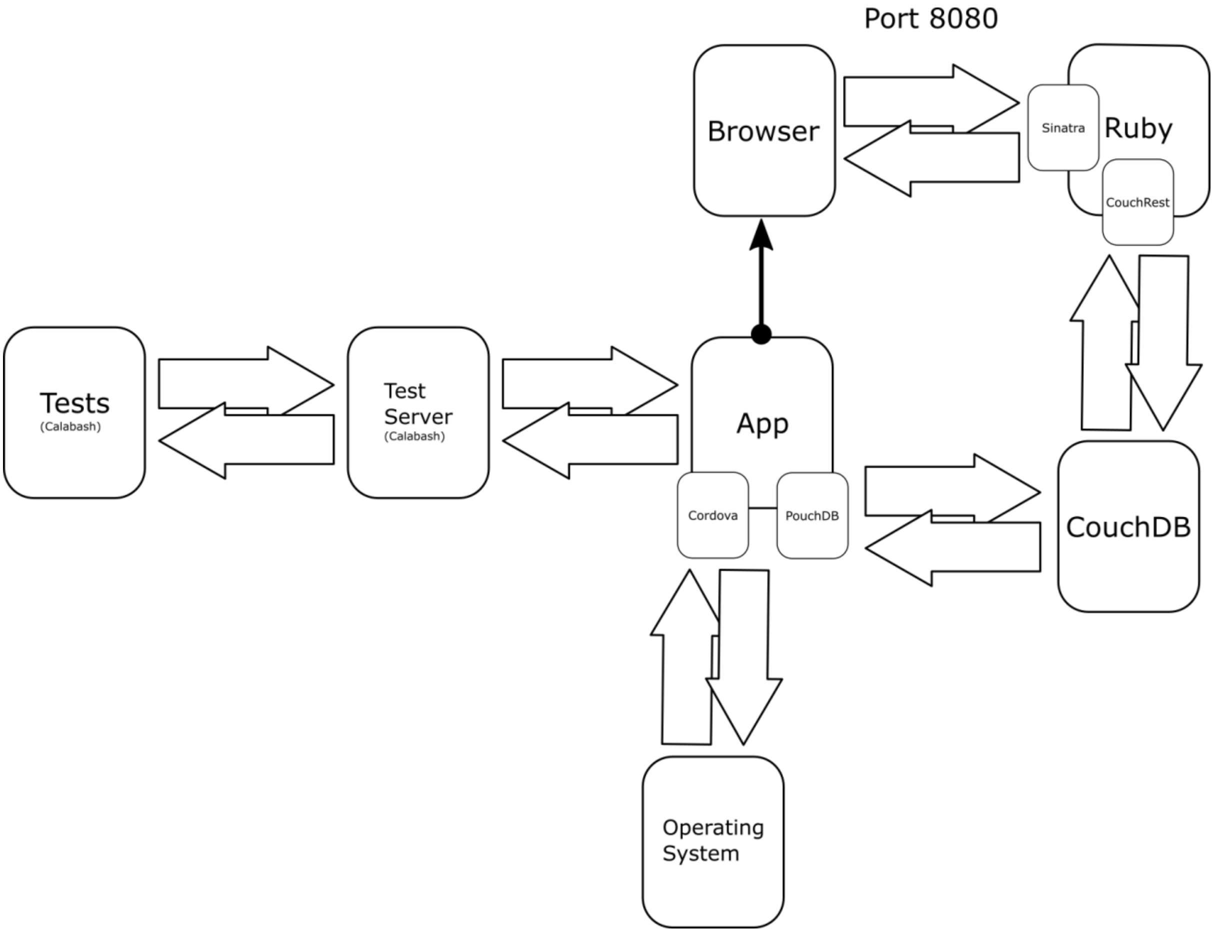


Figure 1 Application overview

The main application is built with Apache Cordova. In some parts of the user has to leave the application to fill in a questionnaire on the server. After that he returns to the application. The synchronization between the server side and the application side is automated by PouchDB.

2.1 Sinatra

Sinatra is “a DSL [domain specifique language] for quickly creating web applications in Ruby with minimal effort” (Sinatra, n.d.). It was designed and developed by Blake Mizerany (Sinatra, n.d.). A popular project that uses Sinatra is the widely known continuous integration server Travis CI (Travis, n.d.).

Sinatra is a wrapper around Rack Middleware (Harris & Haase, 2012, p. 2) which, in simple words, listen to a port (in this project to port 8080) and delegates requests to methods defined by the user of Sinatra. It handles HTTP requests and delivers responses to clients. Moreover, it “ties specific URL directly to relevant Ruby code and returns its output in response” (Sinatra, 2012). Rack Middleware is an enhanced interface to talk to web servers like Mongrel, Thin or Apache via Passenger. However it has some more features like grouping and ordering modules (AmberBit, 2011).

The next examples shows a very simple routing. If this code runs on pas-web.ch a call to http://pas-web.ch:8080/question is delegated by Sinatra to the get method and then the simple HTML `<h1>Question!</h1>` is returned. If there is a post request to http://pas-web.ch:8080/answer, for example when sending a form, then the request is rerouted to http://pas-web.ch:8080/answer and again `<h1>Question!</h1>` is returned.

```
require 'sinatra'

get '/question' do
  '<h1>Question!</h1>'
end

post '/answer'
  //do something with the answer
  redirect “/question”
end
```

Furthermore Sinatra has an easy to use support for templates. I used erb templates which can easily be rendered as seen in next example. Since erb is already included in Ruby, there was no need to add another dependency.

```
require 'sinatra'

get '/question' do
  erb :question, locals => { :questionnrs => [1, 2, 3] }
end
```

```
<html><head><title>Question</title></head>
  <body>
    <% questionnrs.each do |questionnr| %>
      <%= questionnr%><br>
    <% end %>
  </body>
</html>
```

In my project Sinatra ran on the server side and handled browser requests done by the user.

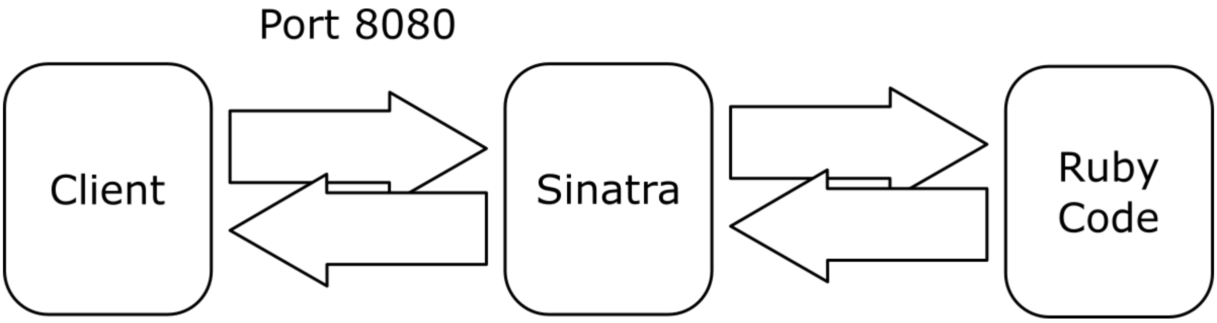
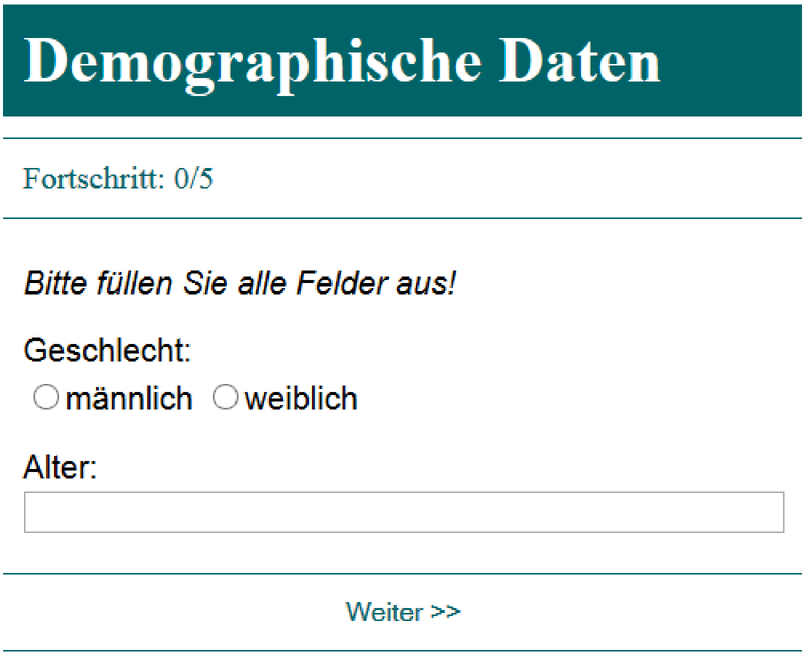


Figure 2 Sinatra working with Ruby code and client

Sinatra proved to be an adequate choice, since it is lightweight and allowed for a simple form of a questionnaire and store the answers into the CouchDB. Around 750 lines of code in combination with couchrest gem to connect to the CouchDB instance this was all that was needed to get the server side of the application running. There was no urge to a more heavyweight framework like RubyOnRails.



Demographische Daten

Fortschritt: 0/5

Bitte füllen Sie alle Felder aus!

Figure 3 page of the questionnaire

2.2 Storage

2.2.1 CouchDB

CouchDB is a document-oriented NoSQL database. It was released in 005 and became an Apache project in 2008. It stores data as JSON-documents (Apache Software Foundation, n.d.) and provides a REST API to access data. Interestingly enough, the renowned BBC uses CouchDB is BBC (Erlang, 2009) as well as many smaller software and databases (Apache Software Foundation, 2013).

To retrieve a data, you send a simple get request to the database:

```
curl -X GET http://pas-web.ch:5984/demo
```

Which would, in our case, (since the database does not exist) simply return a JSON-File stating an error: `{"error":"not_found","reason":"no_db_file"}`.

To create a database, only the following simple command is required:

```
curl -X PUT http://127.0.0.1:5984/demo
```

In my project this database is used to store the part of the user data that is later used to do some research by Alexandra Barth. A big advantage of this database is that there exists a client called PouchDB (see chapter 2.2.2) which hides the automatic synchronization between the local database on smartphones and the database on the server.

2.2.2 PouchDB

PouchDB is a fresh, browserbased java-script database created and maintained by a group around Nolan Lawson. Version 1.0.0 was released in the year 2013 (PouchDB, 2013).

PouchDB is not self-containing. It is an abstraction layer over other databases. Supported databases are, for example, IndexedDB and WebSQL (PouchDB, n.d.).

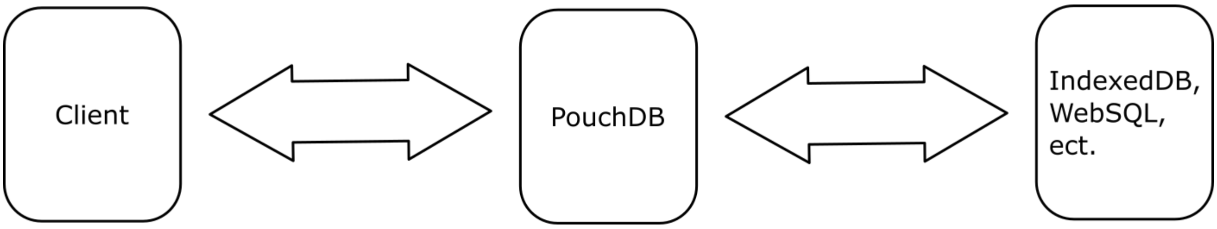


Figure 4 PouchDB as abstraction layer

It was designed to be easily adjusted to synchronize with a CouchDB instance on a server. Thus it is possible to store data on a smartphone without having an internet connection and, as soon as PouchDB connects, it synchronizes with the CouchDB database on the remote side.

Creating a database is as simple as that:

```
pouchdb = new PouchDB("my_pouch")
```

Post a document with an ID created by the system:

```
pouchdb.post({
  title: 'Heroes'
});
```

Delete a database:

```
pouchdb.remove(docId, docRev)
```

The automated synchronization is a huge advantage because one does not have to worry about making synchronization work. A problem that remains is the conflict resolution. In my case I took especially care that data changed by either the client or the server side may only be changed by the client side and only read by the server side or vice versa. Another advantage is that the database is environment aware so it uses different approaches to store data on Android and IOS-Smartphones.

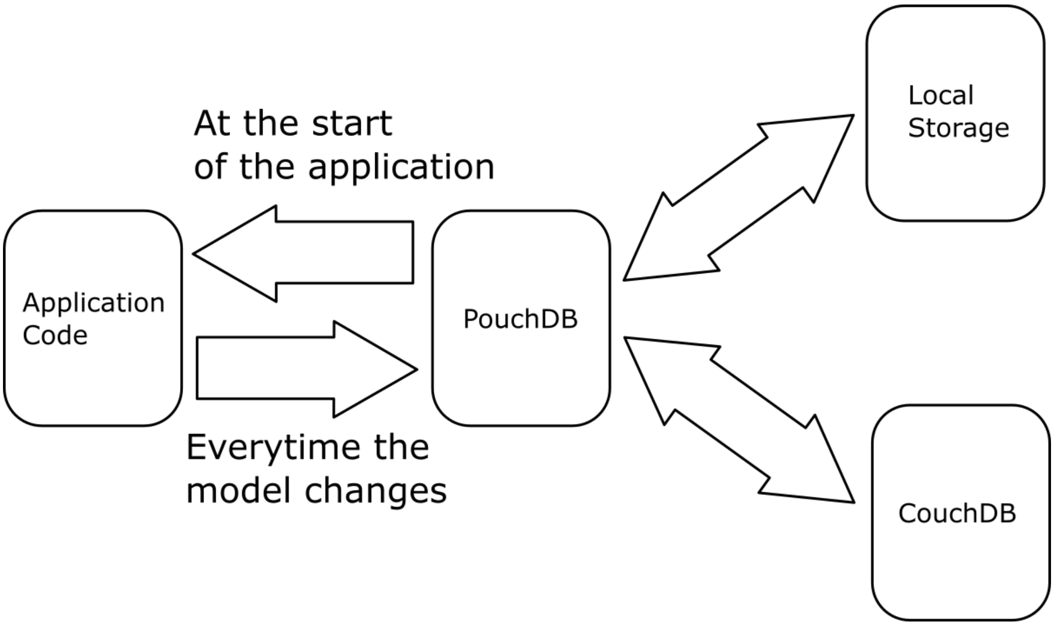


Figure 5 PouchDB working together with local storage and CouchDB. PouchDB is synchronizing with the local storage as well as with the CouchDB on the server side. The only time the application reads from the PouchDB is when it is started after that it only writes to it.

2.3 Cordova

“Apache Cordova […] is a free, open source framework for building cross-platform native application using HTML5” (Wargo, 2014, p. 1). Cordova was initially developed as PhoneGap by a software company called Nitobi. Nitobi donated its PhoneGap software to the Apache Foundation and was then acquired by Adobe which was when two different kind of PhoneGaps emerged. The one called Apache Cordova is the less tightly bound to Abobe products version of PhoneGap (Wargo, 2014, p. 7). Popular companies using Cordova/PhoneGap for some of their applications are Wikipedia, Facebook, Logitech and Microsoft (Trice, 2012).

Cordova can be seen as an abstraction layer between the business application code and the operating system. Apache Cordova provides a multiple consistent APIs through java-script to do basic manipulations on the operating system level. With so called plug-ins, sometimes written by third parties, this API may be extended. However, Cordova does not translate HTML5 code into the native language of the device platform (like Objective-C or Java). It runs the code unmodified within a provided single webview, which renders the web content within the native application. At the beginning of the application Cordova loads the start-up page and then passes the control to the webview (Wargo, 2014, pp. 2-3). In other words, it provides a container to run business application code written in HTML5. In most cases a Cordova application shares many many similarities with a web application that is hosted on a server with the considerable advantage that it is possible to interact with the built-in microphone, camera, the contacts application, etc.

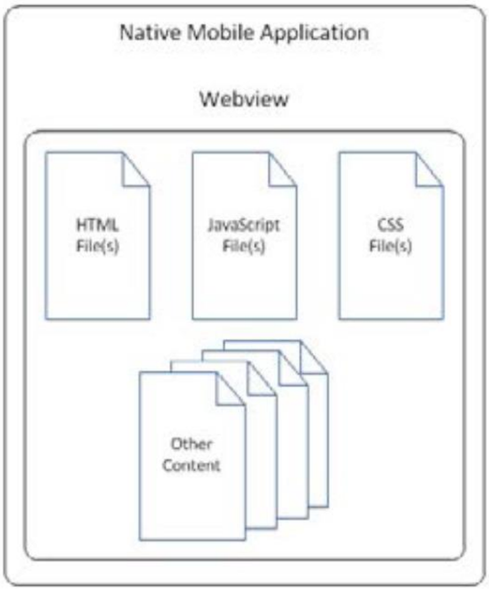


Figure 6 Cordova as container

Creating a very simple app is as simple as creating an index page as seen in the next code sample and put it in the www folder.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-
scale=1, user-scalable=no, width=device-width">
    <title>Hi!</title>
    <!-- cordova script (this will be a 404 during development)
-->
    <script src="cordova.js"></script>

  </head>
  <body>
           <h1>Hello Apache Cordova</h1>=
  </body>
</html>
```

Run in the prompt in the console and deploy the .apk file to an android device

```
cordova build android
```

In my project Apache Cordova was used in combination with Ionic and AngularJS (see chapters 2.4 and 2.5) to provide all the business logic of the application.

2.4 AngularJS

AngularJS is a java-script web application framework developed by Google. It enhances static HTML to create dynamic websites.

Key features of AngularJS

- Declarative HTML approach
- Two way data binding
- Encourages MVC Design Pattern
- Easy unit testing
- Templating
- Routing
- Concepts like injector, directive and scope

Slightly altered from (Shekhawat, 2013, p. 5)

AngularJS creates a highly dynamic HTML webpage. At page start-up it waits for the DOM content loaded event. As soon as this event is emitted an injector is created. Then the webpage is compiled and displayed in the browser.

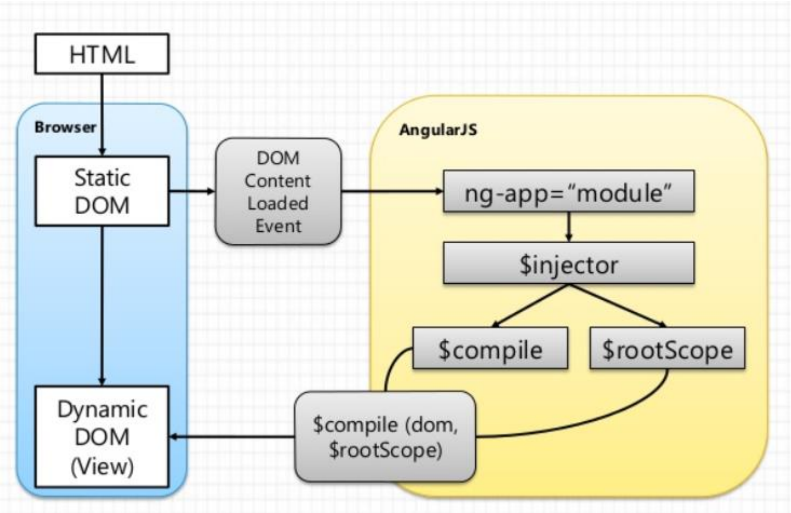


Figure 7 AngularJS at start-up (Eyal, 2013, p. 2)

2.4.1 MVC Design Pattern

The model is defined in plain JavaScript. AngularJS encourages people to use the built-in service constructor to create and access the model. Nevertheless java-script is a prototype based programming language, it does not support objects by default, but by associating functions to variables, you may create something that is object like.

```
function Exercise() {
      this.name = undefined;

      this.value1 = undefined;
      this.value2 = undefined;
      this.value3 = undefined;

      this.thought = undefined;

      /*
      * Returns true if all values for the initial
      * position are filled in. This consist the following
      * values not to be undefined:
      *
      * - value1, value2, value3
      * - thought
      * - exercise name
      */

      this.finishedInitialPosition = function
finishedInitialPosition() {
            return this.value1 && this.value2 && this.value3 &&
this.thought && this.name;
      };
}
```

The view is defined in HTML5 with a powerful addition, the declaratives (see chapter 2.4.5). Controllers are defined via the function module on the angular variable which is provided by the AngularJS framework.

```
angular.module('starter.controllers', [])
.controller('Ctrl', function() {
      //I control stuff
});
```

AngularJS provides a useful way to create controllers which, consequently, can be associated to parts of the view (see chapter 2.4.5). After a controller was defined AngularJS allows you to assign controllers to parts of the view. Controllers may be reused for other parts.

```
<div ng-controller="Ctrl">
      <!-- this part is controlled by the Ctrl controller -->
</div>
```

2.4.2 Injector

The injector provides a lookup for services and objects. Dependencies are resolved by the injector when using the invoke method on it.

```
//create a module
var myModule = angular.module('myModule', [])

//create a factory in that module
myModule.factory('serviceA', function() {
      return "Hello World!"
});

//write a function
function hello(serviceA) {
      console.log(serviceA)
}

//retrieve injector
var $injector = angular.injector(['myModule']);

//let angularjs do the work
angular.invoke(hello)
```

Example altered from (Eyal, 2013, pp. 8-10)

2.4.3 Double binding

In AngularJS variables are observed. If a change to a variable in the mode occurs those changes are reflected in the view and vice versa. To make AngularJS clear that such a change happened, one uses the $apply method.

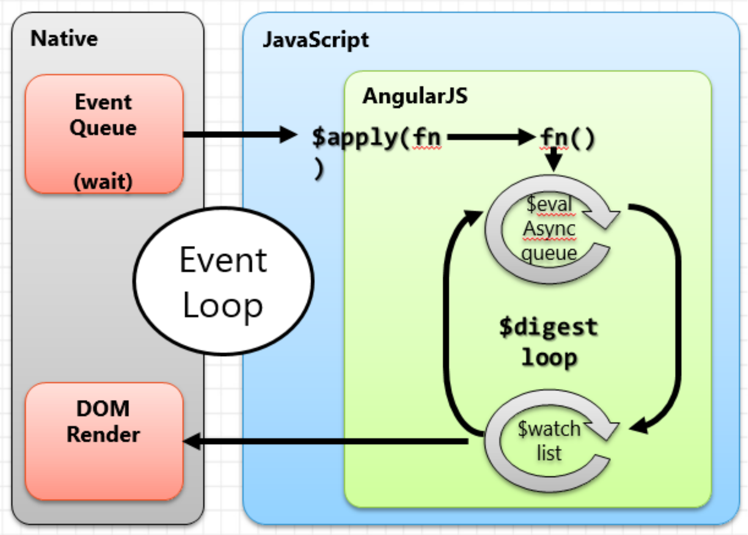


Figure 8 $apply, $digest life cyle from (Eyal, 2013, p. 28)

As soon the $apply method is called. The $digest cycles begins. As a consequence all watched variables are reevaluated and associated functions are called. One $apply cycle may consist multiple $digest cycles (Eyal, 2013, pp. 28-33).

Bindings in the view are denoted by the double curly brackets

```
{{name}}
```

Name refers here to the name variable in the $scope variable of the controller. (AngularJS, n.d.)

```
$scope.name = "Pantalaimon"
```

2.4.4 Scope

A crucial part of AngularJS is the concept of the scope. It lets model, controller and view communicate with each other without binding them together too strongly. Moreover a scope can be inherited. (AngularJS, n.d.). Each controller has its own scope. If controllers are nested, the nested controller inherits the scope of its parent, but not vice versa. Variables defined in the child are not accessible in the parent even if they are called the same.

```
angular.controller(..
      $scope.greeting = “hey parent”
      // Has no access to the $scope.child variable

      //waits for a message from its child

      //prints always “hey parent”
)
angular.controller(..
      console.log($scope.greeting) //prints “hey parent”
      $scope.greeting = “hey child”
      console.log($scope.greeting) //prints “hey child”

      //sends message to parent to print its variable
      //and set it to “hey parent” again
      console.log($scope.greeting) // prints “hey child”
)
```

2.4.5 Routing

In standard AngularJS routing is done with the built-in $routeProvider (AngularJS, n.d.). Due to Ionic using a more advanced routing called UI-Router that uses a state machine the $routeProvider will not be covered here. Instead, the concepts of the UI-Router will be described.

2.4.6 Directives

One of the most important concepts is that of a directive. Directives are added as tags to the html code. AngularJS has many built-in directives, for example, the ng-repeat directive, that allows for iteration over a list and thus repeats parts of your html code

```
<html ng-app="myApp">
<head>
  <script src="angular.js"></script>
</head>
<body ng-controller="PhoneListCtrl">
  <ul>
    <li ng-repeat="number in [3, 2, 1, “take off!”]">
      <p>{{number}}</p>
    </li>
  </ul>
</body>
</html>
```

Example altered from (AngularJS, n.d.)

The example above results in something similar such as

```
<html ng-app="myApp">
```

```
<head>
  <script src="angular.js"></script>
</head>
<body ng-controller="PhoneListCtrl">
  <ul>
    <li>
      <p>3</p>
    </li>
    <li>
      <p>2</p>
    </li>
    <li>
      <p>1</p>
    </li>

    <li>
      <p>take off!</p>
    </li>
  </ul>
</body>
</html>
```

In this project AngularJS was used internally by the Ionic Framework. The Ionic framework used it to create its own directives.

2.5 Ionic

Ionic is a framework written in java-script to build mobile apps with HTML5. It was created in 2013 by Drifty, a small company founded by Max Lynch and Ben Perry (Drifty, n.d.).

Ionic heavily depends on AngularJS and Apache Cordova (see chapters 2.3 and 2.4) and “to make building hybrid mobile apps fast, easy, and beautiful” (Wilken, 2015). Developers called it “a powerful HTML5 native app development framework that helps you build native-feeling mobile apps all with web technologies like HTML, CSS, and java-script” (Ionic, n.d.) and its goal is “to make it easier to develop native mobile apps with HTML5, also known as hybrid apps” (Adam, 2013). Ionic itself was mostly designed to create smooth user interfaces and experiences.

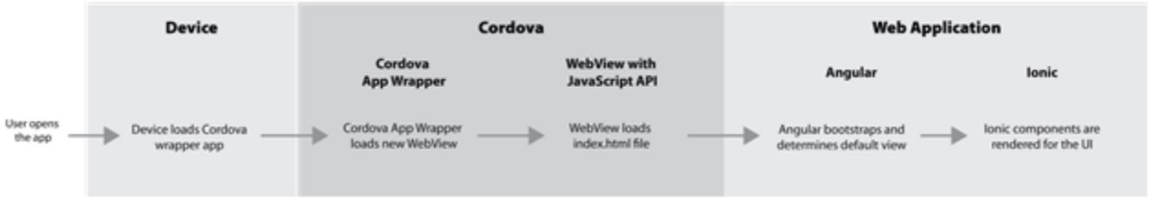

Device
Cordova
Web Application
Cordova App Wrapper
WebView with JavaScript API
Angular
Ionic
User opens the app
Device loads Cordova wrapper app
Cordova App Wrapper loads new WebView
WebView loads index.html file
Angular bootstraps and determines default view
Ionic components are rendered for the UI

Figure 9 Ionic mental stack (Wilken, 2015)

A decisive advantage of the Ionic Framework is that, firstly, provides a large set of prebuild css styles to make your app immediately look pleasant and secondly implements concepts widely used in mobile applications like side menus and header menus.

2.6 calabash-android

Calabash allows readable tests written in Gherkin. It expresses the software to be built in real examples and brings stakeholders and developers together. It is an adaption of the cucumber test framework which was developed by a team around Aslak Hellesøy (Wikipedia, 2015).

“By using real-world examples to describe the desired behaviour of the system we want to build, we stay grounded in language and terminology that makes sense to our stakeholders: *we’re speaking their language*” (Wynne, 2012, p. 25)

This allows customer and developer to come up with an ubiquitous language and to exchange requirements in a way both sides understand. Gherkin, as in seen in the next example, is a DSL to describe a requirement and scenario of the application in business language.

```
Scenario: As a user I should be redirected to the overview page
     When I filled in the questionnaire
     And I wait for a maximum of "10" seconds
     Then I see the "overview" page
```

Due to its high level of abstraction Gherkin is then bootstrapped to Ruby. Furthermore, each step in Gherkin is then processed in Ruby where it is wired to the model.

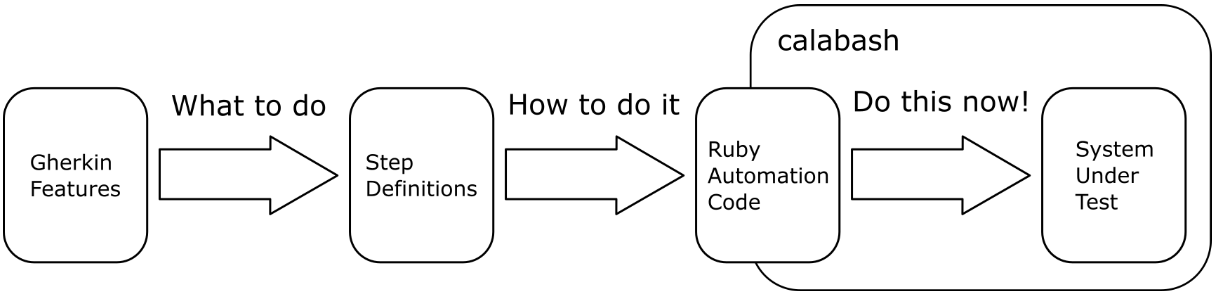


Figure 10 wiring of Gherkin, step definitions and ruby code. Figure slightly altered from (Wynne, 2012, p. 41).

```
Given(/^I have not received my attestation$/) do
     wait_for_page("overview")
     execute_javascript(:file => "attestation-not-
received.js")
end
```

What calabash does is provide possibilities to communicate through gestures (touch, double-tab, pinch, etc.) with the application. It also provides a way to inject java-script code into the mobile application which allows one to manipulate runtime java-script objects.

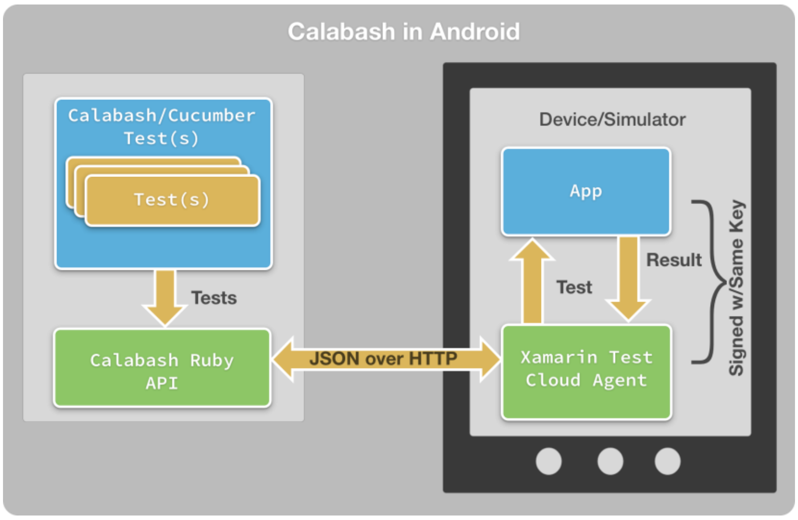


Figure 11 Calabash in android (Xamarin, n.d.)

3 Literature

Adam. (2013, 10 28). *Where does the Ionic Framework fit in | The Official Ionic Blog*. Retrieved 5 7, 2015, from http://blog.ionic.io/where-does-the-ionic-framework-fit-in/

AmberBit. (2011, 7 13). *Introduction to Rack middleware*. Retrieved from https://www.amberbit.com/blog/2011/07/13/introduction-to-rack-middleware/

AngularJS. (n.d.). *AngularJS Tutorial 2 - Angular Templates*. Retrieved 5 5, 2015, from https://docs.angularjs.org/tutorial/step_02

AngularJS. (n.d.). *AngularJS: API: $rootScope.Scope*. Retrieved 5 6, 2015, from https://docs.angularjs.org/api/ng/type/$rootScope.Scope

AngularJS. (n.d.). *AngularJS: Tutorial: 7 - Routing & Multiple Views*. Retrieved 10 5, 2015, from https://docs.angularjs.org/tutorial/step_07

Apache Software Foundation. (2013, 6 19). *CouchDB_in_the_wild - Couchdb Wiki*. Retrieved 7 6, 2015, from http://wiki.apache.org/couchdb/CouchDB_in_the_wild

Apache Software Foundation. (n.d.). *Apache CouchDB*. Retrieved 6 23, 2015, from http://couchdb.apache.org/

Cucumber. (2015, 6 7). *Cucumber*. Retrieved from https://cucumber.io/

Drifty. (n.d.). *Drifty Blog - Building a great company with great products*. Retrieved 7 6, 2015, from http://blog.drifty.com/

Erlang. (2009). *Erlang Factory - Enda Farrell, Software Architect for internet scaling*. Retrieved 7 6, 2015, from http://www.erlang-factory.com/conference/London2009/speakers/endafarrell

Eyal, V. (2013, 5 12). *AngularJS architecture.* Retrieved 5 4, 2015, from http://de.slideshare.net/EyalV/angularjs-architecture

Harris, A., & Haase, K. (2012). *Sinatra: Up and Running.* Sebastopol: O'Reilly.

Ionic. (n.d.). *Ionic Documentation Overview - Ionic Framework*. Retrieved 5 6, 2015, from http://ionicframework.com/docs/overview/

Larsen, J. (2012, 3 7). *blog.LessPainful.com*. Retrieved 5 25, 2015, from http://blog.lesspainful.com/

PouchDB. (2013). *Releases · pouchdb/pouchdb · GitHub*. Retrieved 7 3, 2015, from https://github.com/pouchdb/pouchdb/releases?after=2.0.2

PouchDB. (n.d.). *Adapters*. Retrieved 5 7, 2015, from http://pouchdb.com/adapters.html

Shekhawat, M. (2013, 6 14). *Introduction to Angularjs.* Retrieved 5 4, 2015, from http://de.slideshare.net/manishekhawat/angularjs-22960631

Sinatra. (2012, 11 3). *Sinatra Book*. Retrieved 6 30, 2015, from https://github.com/sinatra/sinatra-book/blob/master/book/Introduction.markdown

Sinatra. (n.d.). *Sinatra: About*. Retrieved 6 29, 2015, from http://www.sinatrarb.com/about.html

Sinatra. (n.d.). *Sinatra: README*. Retrieved 5 11, 2015, from http://www.sinatrarb.com/intro.html

Travis. (n.d.). *Travis CI: Continous Integration And Deployment That Just Works*. Retrieved 29 5, 2015, from https://travis-ci.com/

Trice, A. (2012, 4 27). *Who Uses PhoneGap/Apache Cordova | ANDREW TRICE*. Retrieved 7 6, 2015, from http://www.tricedesigns.com/2012/03/27/who-uses-phonegapapache-cordova/

Wargo, J. M. (2014). *Apache Cordova 3 Programming.* Upper Saddle River NJ: Addison-Wesley.

Wikipedia. (2015). *Cucumber (software) - Wikipedia, the free encyclopedia*. Retrieved 6 7, 2015, from https://en.wikipedia.org/wiki/Cucumber_%28software%29

Wilken, J. (2015). *Ionic in Action - MEAP Version 9.* Manning Publications. Retrieved 7 6, 2015, from http://www.manning.com/wilken/IonicinA_MEAP_ch01.pdf

Wynne, M. &. (2012). *The Cucumber Book.* Raleigh: Pragmatic Programmers, llc.

Xamarin. (n.d.). *Introduction to calabash - Xamarin*. Retrieved 5 7, 2015, from http://developer.xamarin.com/guides/testcloud/calabash/introduction-to-calabash/