INFORM@RISK SMARTPHONE APP

TABLE OF CONTENTS

APP OVERVIEW

TECHNOLOGIES

FRAMEWORK7

https://framework7.io/

Framework7 is a HTML/CSS/Javascript based cross-platform app development framework. It can use any other purely CSS/JS based libraries without any additional effort and produces webview based hybrid apps.

Framework7 comes with a wide variety of predefined UI elements and also handles the basics of app initialization and routing between different screens. It does not directly interface with phone hardware in any meaningful way. Hardware connections are established through a combination of common HTML functionality and via the chosen underlying webview development framework.

In case of the Inform@Risk app, this framework is Apache Cordova

CORDOVA

https://cordova.apache.org/

Apache Cordova handles compilation of the app itself and in-app hardware functionality. Major necessary hardware functionality includes:

- Camera access for reporting purposes
- Location access for reporting purposes
- Opening HTTP links with currently installed default web browser
- Receiving push notifications

The remaining installed plugins are necessary default functionality like keyboard, vibration, hard drive access for storage of local data, etc…

JQUERY

https://jquery.com/

Framework7 actually provides most commonly used JQuery functionality like DOM manipulation and AJAX requests on its own. JQuery was integrated nonetheless, since F7’s DOM manipulation is noticeably worse in total provided functionality than JQuery and during development there were repeated issues with F7’s own version of POST requests.

Normal unmodified GET and Post requests were working as expected, but any advanced manipulation of communication like modifying sent headers led to enough issues to switch to JQuery, since there are no major downsides to this decision anyway. DOM access via native F7 methods or JQuery also does not measurably affect performance.

CHART.JS

https://www.chartjs.org/

Chart.js is used for displaying data gathered by sensors on the detail screen for individual sensor installations. It was chosen since it is a very lightweight solution and sensor installation details screens need to render a significant amount of graphs at the same time. An additional library is also necessary to correctly format timestamps into displayable annotations on graphs.

MOMENT.JS

https://momentjs.com/

Although Moment.js is no longer under official active development, Chart.js explicitly depends on it for display purposes, so it was chosen for general time output formatting for the app.

OPENLAYERS

https://openlayers.org/

The app features a wide variety of interactive maps with different kinds of content. Due to ease of use and developer familiarity, OpenLayers is used in the app for this purpose.

FIREBASE CLOUD MESSAGING

https://firebase.google.com/

The Google provided Firebase Cloud Messaging(FCM) service is used to send push notification to all installed users. It has extensive example coverage in both how to send and how to receive notifications on clients and supports all common push notification functionality as expected (deep linking into app, handling received messages both from outside or inside the app, message priority, etc…)

STRUCTURE

GENERAL GLOBAL CODE IN APP.JS

The central logic file app.js contains both app initialization and all global code that does not belong to specific major functionality of the app that is described later on.

App initialization sets the global app state by both retrieving data from permanent storage during startup and also querying the server for updated data. Global event handlers are set to handle back button presses on the phone and correct translations are set according to either user preference and underlying phone language.

TEMPLATE STRUCTURE AND NAVIGATION

Templating structure is very simple in F7; the template with the global container for the entire application is called index.html and it is the only top-level template in the entire application.

Main.html contains the initial entry point into the app and serves as navigation hub. The app is never deeper than 3 levels of sub screens and all lead back to the main screen.

Navigation is defined purely within routes.js and all templates access these defined navigation routes via ordinary HTML hyperlinks.

SPECIFIC MODULES – MAPS, SENSORS, REPORTS

Code is bundled according to functionality for some specific purposes and not contained in the general global function store of app.js:

- Reporting
- Interactive maps
- Sensor storage and management
- Firebase messaging

FCM and maps will be covered in more detail, since they are non-trivial in complexity.

FCM

Firebase messaging functionality is exclusively bundled in firebase/index.js. It handles both registration to the messaging channel on the remote server on initial app startup and contains the event handler on what to do in the app in case a push notification is received while the app is open (push notifications are not displayed in case the app happens to be open while the communication is received).

Functionality to send push notifications to individual clients only is not considered due not being needed at this point and possible data privacy concerns since firebase is a Google service and individuals communication would necessitate unique device identifiers.

MAPS

Map display logic is divided into 3 separate files since it is fairly complicated.

The original plan of the app was to mix and match layers on different screens and to avoid excessive code duplication of layer and display logic, map initialization, layer and style data were separated into different staic global objects.

The central file map.js deals with map initialization, creating global events like popups opening and access/manipulation methods of existing rendered map layers.

The layer definition file maplayers.js contains individual openlayers layer object definitions and various callbacks for retrieving data, filling empty popups with actual data, select feature events, etc…

The final file mapStyles.js is focused on both openlayers object style definitions for objects on maps itself and also the styles of popups for individual selected features for the various layers (markup + CSS). The styling file is truly separate from actual function code of the corresponding layers, since a lot of the styles for features depend on various attributes being set for display purposes (e.g. the state of a sensor influences the border color of its icon on the sensor overview map).

DATA STORAGE

The currently used storage solution for persistent data storage between app startups is localStorage for all data.

In case localStorage becomes unfeasible as a solution in the future, any kind of synchronous, global key/value store based on strings can be used with essentially no effort. The central logic file app.js contains a global setter and getter method to access localStorage and was designed to be replaced at a later point if necessary.

Complex objects are serialized into JSON before storage and properties define in their own getters/setters if their data needs to be serialized.

SERVER OVERVIEW

TECHNOLOGIES

LARAVEL

https://laravel.com/

The core of the app backend is an Apache2 webserver running Laravel, a PHP framework.

Laravel serves both the API for the app itself and the web based administration interface to manage app content.

POSTGRESQL

https://www.postgresql.org/

PostgreSQL is chosen as RDBM backend for the webserver instead of the more common MySQL/Maria due to better support for geographic data.

POSTGIS

https://postgis.net/

Although explicit plugins for geographical data are not necessary in the current use case of this app since it only need to save and display basic shapes(Points,Lines,Polygons), PostGIS was still used as a data storage solution in case and advanced geographical calculations like dynamic routing are ever required.

IMAGEMAGICK

https://imagemagick.org/index.php

Imagemagick is an image display and manipulation program. It is used during the image upload process when a user submits a report.

During downsampling for later use, images need to corrected for their rotation using their EXIF data. PHP plugins take care of this for the vast majority of images, but Imagemagick exists as a fallback. Smartphones from certain vendors contain EXIF data that cannot be correctly interpreted by default PHP image manipulation libraries and Imagemagick takes over in case PHP fails.

STRUCTURE

The structure follows basic Laravel conventions, Laravel itself being a fairly standard implementation of the MVC concept.

The structure of Laravel projects is summarized in their own official documentation:

https://laravel.com/docs/9.x/structure