| *Project Identification* | |
|---|---|
| *Project Number* | AAL-2013-6-129 |
| *Duration* | 24 months |
| *Coordinator* | Thomas Kamps |
| *Coordinator Organization* | YouPers AG |
| *Website* | www.youpers.com |

# revolution

# Solution Design

| *Document Identification* | |
|---|---|
| *Deliverable ID* | D2.5 Solution Design |
| *Release number/date* | V2.0 2015-07-17 |
| *Checked and released by* | Reto Blunschi |

| *Key Information from "Description of Work"* | |
|---|---|
| *Deliverable Description* | The Solution Design document builds the basis for all research and development work of the project. It documents the system and software architecture to be used for this solution.<br>Each component will be integrated into the overall system architecture.<br>• The consortium will decide on which types of smartphone will be supported<br>• The characteristic for the platform will be specified<br>• Collaboration: Definition for the cooperation of the community |
| *Level* | RE = Restricted |
| *Deliverable Type* | D = Document |
| *Original due date* | 2015-03-31 |

| *Authorship & Information* | |
|---|---|
| *Editor* | Reto Blunschi, YP |
| *Partners contributing* | YP, XIM, HSLU |
| *Reviewed by* | XIM, HSLU |

# Release History

| *Release Number* | *Date* | *Author(s)* | *Release description /changes made* |
|---|---|---|---|
| ***V 0.0*** | 2014-05-31 | Reto Blunschi, YPS | Basic Document structure |
| ***V 0.1*** | 2014-07-02 | Reto Blunschi, YPS | Adapted after scope definition and solution design workshop in Leuven |
| ***V0.2*** | 2015-01-18 | Paul Schmieder, HSLU | Technical Amendments and corrections, description of the internal structure of the App. |
| ***V 0.9*** | 2015-04-12 | Paul Schmieder, HSLU | Client documentation |
| ***V 1.0*** | 2015-05-31 | Reto Blunschi, YPS | Finalizations |
| ***V1.01*** | 2015-06-08 | Laurenece Pearce, XIM | Review comments |
| ***V1.02*** | 2015-06-10 | Reto Blunschi, YPS | Updates after Iteration 1 und 2 |
| ***V1.03*** | 2015-05-15 | Laurence Peace, XIM | Updates and English language corrections |
| ***V1.04*** | | | |
| ***V1.05*** | | | |
| ***V1.06*** | 2015-07-08 | Paul Schmieder, HSLU | Updates section 6.3 |
| ***V1.07*** | 2015-05-14 | Reto Blunschi, YPS | General review and updates |
| ***V1.08*** | 2015-07-16 | Clemens Nieke, HSLU | Overall review |
| ***V2.0*** | 2015-7-17 | Reto Blunschi, YPS | Finalizations and Release |

# REVOLUTION Partners

REVOLUTION (2013-06-129) is a project within the AAL Joint Programme Call 5.

The consortium members are:

| ***Partner 1:*** | ***YouPers (YP), project coordinator*** |
|---|---|
| Contact person: | Thomas Kamps |
| e-mail: | thomas.kamps@youpers.com |
| ***Partner 2:*** | ***Lucerne University of Applied Science and Arts – CEESAR iHomeLab (HSLU)*** |
| Contact person: | Clemens Nieke |
| e-mail: | clemens.nieke@hslu.ch |

| | |
|---|---|
| ***Partner 3:*** | ***terzStiftung TERZ)*** |
| Contact person:<br>e-mail: | Sabine Kaiser<br>sabine.kaiser@terzstiftung.ch |
| ***Partner 4:*** | ***XIM (XIM)*** |
| Contact person:<br>e-mail: | Laurence Pearce<br>laurence.pearce@xim.co.uk |
| ***Partner 5:*** | ***Creagy (CRE)*** |
| Contact person:<br>e-mail: | Reto Blunschi<br>reto.blunschi@youpers.com |
| ***Partner 6:*** | ***ANA ASLAN INTERNATIONAL Foundation (AAIF, with voluntary contribution)*** |
| Contact person:<br>e-mail: | Prof. Dr. Luiza Spiru<br>lsaslan@brainaging.ro |

# Table of Contents

# Abbreviations

| *Abbrev.* | *Description* |
|---|---|
| AAL | Ambient Assisted Living |
| AAL JP | Ambient Assisted Living Joint Programme |
| HEATLHY@WORK | Acronym of this Project |

# 1 Introduction

## 1.1 Background

This solution design document is intended for the research and development project REVOLUTION. The project is financed by the European Union (EU), through the AAL Joint Program, and by national authorities in the respective countries represented in the consortium (National Contact Points).

It is the goal of REVOLUTION to address the fields of voluntary, unpaid occupation of people in the post-employment phase. The idea of REVOLUTION is based on a new, very flexible way to plan and support help services brought by these older adults on a voluntary basis. Both the volunteers and the assisted persons (people of all age), will sustain and/or increase their participation in social life with positive effects on personal satisfaction, demographic ageing and overall healthcare costs.

It is the aim of our international team to develop an innovative solution to keep the sprightly pensioners healthy and help them to enjoy their life.

## 1.2 Purpose of the Solution Design

The solution design has the goal to define system and software design decisions to enable different partners and teams of the REVOLUTION consortium to work in parallel on multiple components. The Solution Design document details the system architecture by defining the different components and their interaction.

Thus, it ensures that the non-functional system requirements will be met by our system.

The level of detail of this document to be achieved is a compromise between a traditional waterfall approach, that defines everything upfront and a modern, iterative, test-driven approach. REVOLUTION has a project plan that defines 4 iterations – so we expect this Solution Design document at least to be updated before the start of each iteration.

## 1.3 Referenced Documents

D2.1 End User Requirements

D2.2 Report on ethical issues

D2.3 Usability concept and prototype (Wireframes and Screen Map)

D2.4 Data Security and Privacy concept

# 2 Principles and guidelines

The technical design of REVOLUTION follows a set of principles, which are the base criteria for the design choices the consortium has to make during the project.

**Data security and encryption are fundamental**

The safety of the data of the REVOLUTION users is a fundamental design goal of REVOLUTION. Protecting sensitive data is the end goal of almost all IT security measures. Two strong arguments for protecting sensitive data are to avoid identity theft and to protect privacy.

Therefore authentication, authorization and encryption are fundamental components of the REVOLUTION system architecture.

**Privacy is Priority**

The evolution of Web technologies has increased collection, processing and publication of personal data. Privacy concerns are raised more often as applications built on the Web platform have access to more sensitive data — including location, health and social network information — and users' activity on the Web is ubiquitously tracked.

The privacy of REVOLUTION's users is therefore a paramount concern. REVOLUTION will not use any of the user's personal data for anything outside the core REVOLUTION system use cases – no advertising and no selling of statistical data. The solution design will make sure that every user has only access to his own data and shares only the minimum information needed with other users to enable communication and interaction inside REVOLUTION. Other user related information is kept and stored securely on the backend, a user's device will only ever receive the data of the currently logged in user and his interactions with selected other users.

The system architecture needs to ensure that a single end-user device or a single end-user will never have access to the full data of REVOLUTION.

**Reuse before make**

The REVOLUTION technology work group is driven by a goal to efficiently provide value to our business partners and end user organizations. Whenever possible we will evaluate and use existing software, modules and components before we consider writing new software. In today's internet and cloud driven ecosystems a very large amount of software components and services is available, either as Open Source or as commercial components.

The REVOLUTION consortium will employ those existing components whenever they fulfil our needs and their quality is acceptable.

**Open Source first**

Open-source software (OSS) is computer software with its source code made available with a license in which the copyright holder provides the rights to study, change and distribute the software to anyone and for any purpose. Open-source software is often developed in a public, collaborative manner.

Today's global software development environment especially in the area of Internet and cloud-based systems relies heavily on open source software. To be able to deliver applications and systems within reasonable timeframes it is taken for granted that many of the major base components needed to build these systems are not written over and over again but taken from the huge amount of available open source components.

These are the principles of REVOLUTION's solution design. If there is a required feature that is not available as Open Source and cannot be adjusted thereof, a special custom component is built or a closed-source commercial component will be acquired and integrated.

The REVOLUTION development team also contributes to open source projects in form of all adaptations, improvements or bug fixes made during the project in existing open source components.

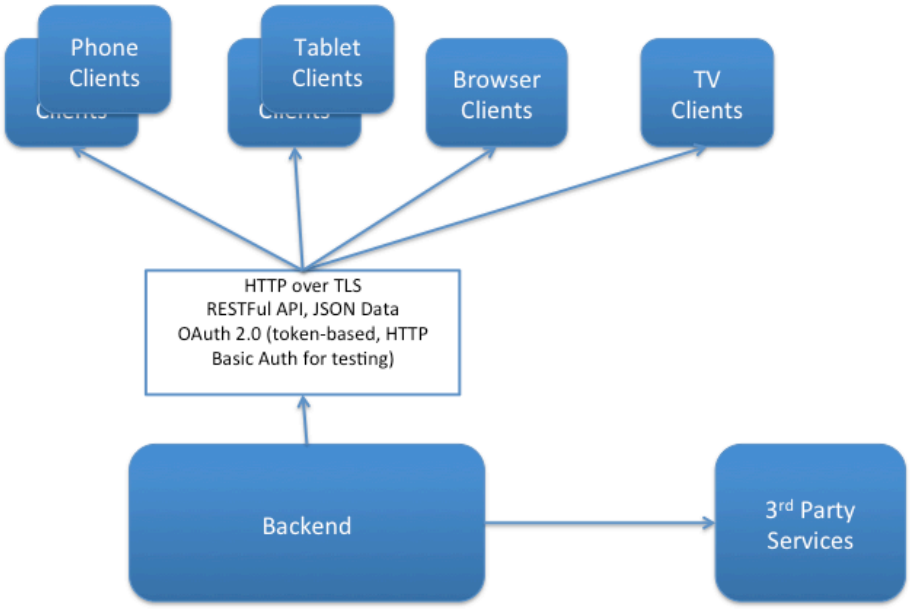
# 3 System Architecture

## 3.1 Overview

The basic system architecture of the envisioned full-scale REVOLUTION system is similar to any of today's modern Internet based applications. It contains the following main system components: a web based client software and several native mobile client applications that all communicate with a central backend infrastructure. The latter keeps the master data and allows all client applications to stay in sync and communicate with each other.

The Backend component is able to communicate not only with the client applications that are part of the REVOLUTION system, but also offers the possibility for other services to integrate with the REVOLUTION platform. Thereby third party systems either act as providers of information like a provider for Activity Tracking information (e.g. your fitbit data), or as a consumer of REVOLUTION data, e.g. a personal calendar on "Google Calendar" or "Microsoft online Calender" displaying REVOLUTION tasks.

RESTFul Services, oAuth 2.0, HTTPS and other contemporary integration mechanisms make this integration of services possible in a secure way.




## 3.2 System Hardware Architecture

The system **hardware** architecture is based on current best practices of today's internet based application architectures. There is no hardware development planned as part of the REVOLUTION project.

### 3.2.1 Client Hardware

REVOLUTION will use client mobile and tablet devices that exist on the consumer market today. As part of the project planning the consortium has decided to first focus on Smartphone sized devices of the two main ecosystems Android and iOS.

The evaluation of which ecosystem should be prioritized has given no clear preference. Distribution between Android and iOS is currently close to even in the consortium's target markets. Also the end-user organizations that will do trials have no clear preference. The result of this evaluation even shows that to be able to execute real trials with the user's personal devices, we need to support both ecosystems right from the beginning.

So the consortium has decided to set the requirements for supported devices to:

- iOS and Android ecosystems
- smartphone-sized devices
- reasonably modern devices (iOS 6+, Android 4+)

This decision may be adjusted after the second development iteration. Eventually, in a planned dissemination phase, all common client architectures will be supported:

- Browser-based large-screen notebooks/computers
- Tablet sized devices
- Other phone ecosystems like Windows Phone

The technology work group of the AAL consortium, after careful consideration of the requirements, has decided to base the client side development on the so-called "hybrid approach" using technologies like phone-gap/cordova and the ionic-framework. For more information about this decision and the experience made with this decision see chapter 6.3

In iteration 3 and 4, where activity tracking is part of the requirements it may be decided to include other client-hardware to support activity tracking. The consortium will rely also for these requirements on existing sensors that are available on the mobile devices and develop software-based integration to those sensors.

There is no custom client hardware development planned in this AAL project.

### 3.2.2 Backend

The backend of REVOLUTION builds on and extends the existing Youpers Health platform backend.

The backend platform of the Youpers Health platform uses virtualized server machines running in multiple data centers located in Switzerland under the jurisdiction of a Switzerland based company. All virtual machines required for development, testing and production environments are rented from this Swiss based IAAS (Infrastructure as a Service) provider.

The IAAS provider provides a guaranteed, scalable service of running the virtual machines that make up the Youpers Platform:

- Database Servers
- Backend Application servers
- Load Balancing servers
- Static Assets servers
- De/Encryption servers
- Management and Monitoring Servers

All these services are running on identical Debian Linux based virtual machines. There is no custom hardware development planned as part of the REVOLUTION project. All needed backend hardware resources will be rented from Swiss-based IAAS providers.

The backend infrastructure will be provided and managed by the consortium partner Youpers.

## 3.3 System Software Architecture

It is the aim of the REVOLUTION consortium to limit the development of custom software to the parts of the system where the real innovation and value of the REVOLUTION project are. For any other components we are employing proven open source solutions wherever feasible.

This strategy is the base for many of the architectural decisions documented in this Solution Design. As an example for this, one may consider the use of the node.js platform for the backend software stack. The node.js platform offers an amazing wealth of high quality modules that can be easily integrated with the npm package management environment. The strategy of using existing software components when possible allows us to focus development on the business value as it limits the amount of boilerplate code to be written to an absolute minimum.

In the following the main software components used as part of the REVOLUTION backend system are listed and described.

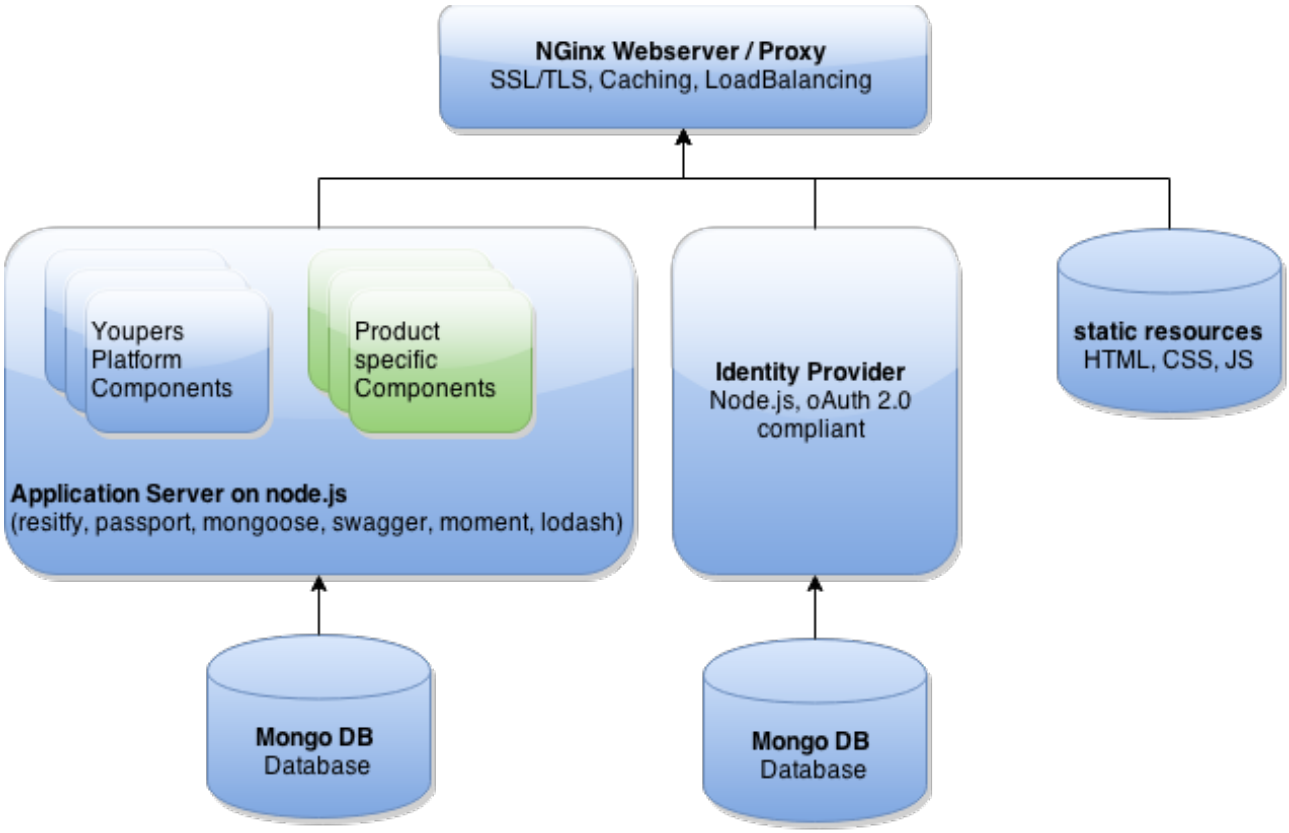
### 3.3.1 Backend Server

The REVOLUTION backend software is based on the following main components:

- Multiple Linux Debian VMs managed by ansible (see https://github.com/ansible/ansible)
  - Openvpn based internal encrypted network
  - SSH based management over encrypted communication
  - Iptables based firewalling
- MongoDb Database
- Node.js javascript based Backend with the following main npm modules:
  - Restify — RESTful server
  - mongoose: — ORM mapper
  - node-Mailer, Email-templates: — email sending
  - passport: — oAuth Client Side
  - iCalender: — Office Calendar integration
  - moment.js — Date/Time/Timezone manipulation
  - async — management of async execution
  - lodash — javascript tooling
  - graphicsmagix — image manipulation
  - socket.io — socket communication
  - apn, node-gcm — mobile push notifications

  All these components are open source and can be found on github.com.
- Youpers Platform backend library (closed source, owned by Youpers)
  - Provides functionality that is common for multiple projects in the YouPers ecosystem like user management, profile management, authentication, authorization, mobile push support, test management

- nginx static Webserver and loadbalancer




Custom Platform Components developed on the node.js platform specifically for the REVOLUTION service:

- User Location Management:
  Maintain the user's home location together with his other "usual locations" in a secure and private server-side storage in order to support the "Smart Matching algorithm".
- Help Request Management:
  Lifecycle Management of all help requests – moves help requests through the state engine defined in the backend: OPEN -> SETTLED -> RATED -> DONE (DELETED – EXPIRED)
- Application Management:
  Manages multiple applications for a single help requests – moves the applications through the application state engine: APPLIED -> ACCEPTED -> RATED (DENIED – EXPIRED – DELETED)
- User Rating/Feedback Component:
  Allows one user to rate another user based on the execution of a help request in several dimensions. After careful evaluation of different approaches to rating, the consortium decided to use a simple 3 value scale similar to the one used by popular services like eBay or ricardo.ch. This component also provides a summarized anonymous view upon a user's part ratings to be displayed as part of his public profile.
- Smart Matching Component:
  Matches a potential volunteer to an open help request in real-time. Based on different input events and stored data the smart matching component notifies the "best matching

volunteers" at the best possible time about a help request they might be able to fulfil now or in near future. See Chapter 6.2.3 for more information.

New general component designed and developed in the REVOLUTION project, destined to become a Youpers Platform component, that will also benefit other projects.

- Notification Management:
  The REVOLUTION service relies heavily on realtime notifications to possible volunteers and help requesters. The main communication channel used is "Push Notifications" on the mobile phone. While the Youpers Platform already had the functionality to send push notifications to all supported platforms before this project, REVOLUTION required us to go a step further. A component to maintain open notifications (unread/read/dismissed/expired) and display a suitable inbox / alert list to the user was needed. Also a more fine granular way to adjust profile settings about which notifications a user wants on which channel how often was added to this component.

  Integration of this component into the smart phone app is part of development iteration 3.

### 3.3.2 Client

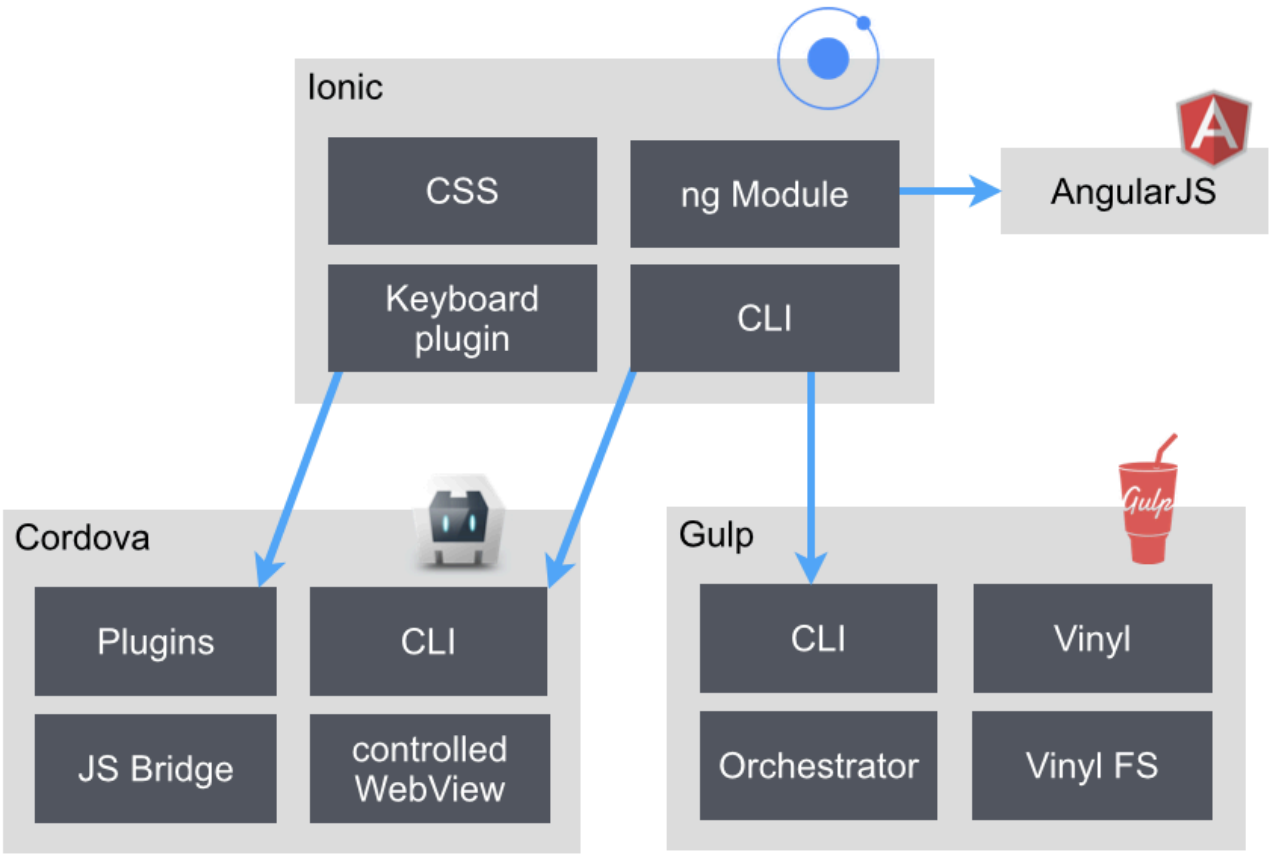
As documented in Chapter 3.2.1 the consortium has decided to focus client prototype development on smartphone sized devices from both popular ecosystems – Android and iOS. Additionally, management interfaces for system and content-administrators will be developed for the large screen browser platform.

The technology work group of the Consortium has carefully considered the non-functional requirements and evaluated different software architectures for the client software:

- Browser-only client using the framework jQuery Mobile
- Hybrid technology mobile app clients using the PhoneGap/Cordova stack with the ionic framework: http://ionicframework.com/
- Native apps using the native software stack for each ecosystem
  - Java-based Android app
  - Objective-C or Swift-based app for the iOS ecosystem

Based on the evaluation document (ClientTechnolgy_Evaluation_V100.pdf), that summarized the advantages and issues of each option the technology work group of the consortium decided to go with the **hybrid technology approach and the ionic framework** for this project's client software architecture.

A short overview of the hybrid architecture together with the ionic framework can be seen in the following picture:




The Ionic framework, at its core, consist of four parts:

- A stylesheet that defines a mobile optimized base layout. This layout serves as the foundation for an app. It includes many best practices that the developer does not need to think about anymore.
- The AngularJS module defines the directives (custom elements), navigation patterns and best practices to simplify development.
- Ionic's developers have a background in developer tooling and is evident in their command line tool. The CLI allows a quick start and can act as a proxy to the Cordova and Gulp CLI.
- The last component is a keyboard plugin. Though technically not required, the plugin provides more information about the current state of the app.

A more detailed description of the client software design can be found in chapter 5 and chapter 6.

## 3.4 Communication Architecture

Communication between clients and backend follows best practices of current Internet based systems. The main defining concepts for the communication architecture are:

- All communication over public networks is securely encrypted by using TCP/IP with HTTPS/TLS
- Authentication and Authorization with standard protocols (oAuth 2.0)
- Stateless, RESTFul Backend API

- JSON data representation
- Notifications of new important events/invitations/messages are transmitted using:
    - Standard Email infrastructure
    - Mobile Push notification (Google Cloud Messaging on Android, Apple Push Notification service –APN on iOS devices)

Communication between mobile clients and the backend is further simplified by the provided Youpers SDKs for Angular JS, Android and iOS respectively. Each SDK is a library that defines the backend's vocabulary making it easy for programmers to develop client applications.

### 3.4.1 Main Backend to Smart Phone client communication

The main communication channel between the client applications and the backend is the communication of the HTML5/Javascript/AngularJS/Ionic app running in the SmartPhone and the RESTful API provided by the backend Server.

This communication is using the AngularJS service architecture and employs the Restangular Framework (https://github.com/mgonto/restangular) to further reduce the needed boilerplate code to be written.

All communication is done using asynchronous Javascript calls (so called AJAX calls) over TLS/SSL encrypted HTTP with perfect forward secrecy – currently the most secure way of encryption for this type of communication that is also used by today's banking internet solutions.

REVOLUTION follows the Best Practice by the angular.js and Restangular frameworks Using these separate components allows splitting responsibilities between partners of the REVOLUTION consortium. For example, the consortium partner developing the user interface (UI) development may use mock services to communicate with the backend while another partner develops the actual services simultaneously.

### 3.4.2 Additional Communication Channels: Email

The Youpers platform uses email as a channel to notify users of certain events while they are offline. Thus, if configured accordingly, a user gets email notifications about new invitations, new recommendations or new messages. Email notifications contain a link to the new information typically viewed in browser.

The communication channel email is the default for users that mainly use a web interface. So, for REVOLUTION this is of minor importance only.

### 3.4.3 Additional Communication Channels: Push Notification

The Youpers platform uses mobile push notifications as a channel to notify users of certain events that happen in the system while they are offline or also to notify a user of other users interaction while they are online. Depending on the configuration and his personal profile settings a user may receive push notifications about new help requests and accepted requests.

The communication channel 'Push Notification' is default for users that mainly use a mobile device to interact with the Youpers platform. Thus, for REVOLUTION this is the dominant communication channel.

The currently supported push notification types for REVOLUTION are:

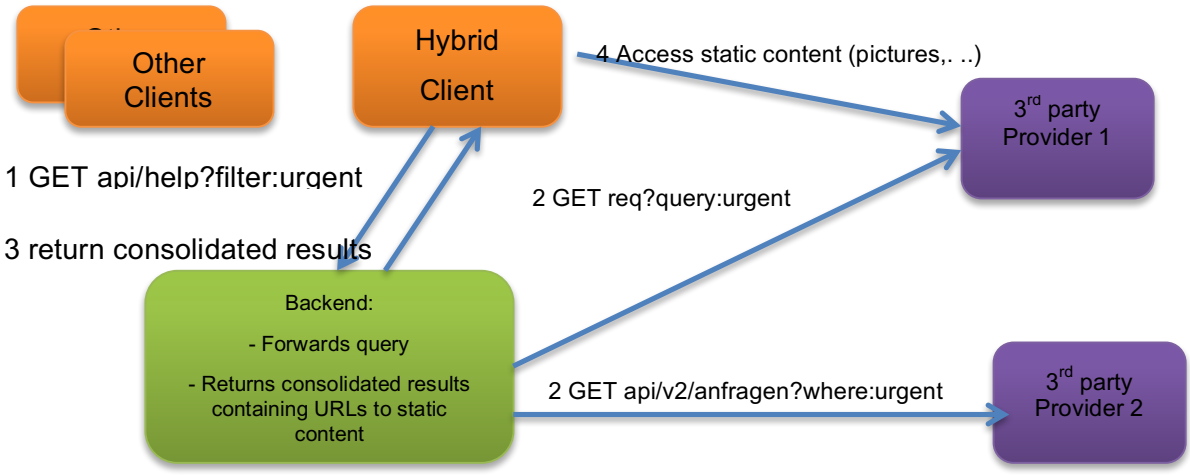
| Technical - ID | Name | Description |
|---|---|---|
| newApplication | Application to your Help Request | new application has arrived |
| applicationAccepted | Your Application was accepted | Volunteer is informed, that his application is chosen |
| urgentHelpRequest | New urgent help request | Volunteers closeby with the appropriate setting |
| newHelpRequest | New matching help request | Volunteers closeby with the appropriate setting |
| pendingRating | pending Rating | Invitation to rate |

Those messages are detailed in: REV_ Push Messages Overview.pdf

### 3.4.4 Communication with 3rd party providers

Communication with 3rd party providers is required to integrate Service Providers like food providers, (semi-)professional volunteering providers or other sites that publish help requests.

According to the overall architecture, communication with 3rd party providers is done via the backend. The backend will combine several sources of information and offer them via a unified API to the clients.




The REVOLUTION prototype as of end of Iteration 2 does not yet include any 3rd party provider integration. As soon as we progress to further iteration more information will be added to this chapter.

Authentication and authorization with 3rd party APIs follows the requirements of each content provider, usually oAuth2 over HTTPS.

## 3.5 Authentication and Authorization

All communication between client and the backend as well as communication between the backend and 3rd party providers is done over encrypted, authenticated and authorized channels.

The details of security, data protection measures, authentication and authorization are documented in "D2.4 Data Security and Privacy"- In this chapter only the aspects of immediate relevance to client and backend developers are documented.

The Youpers platform supports current best practices being used for todays cloud based Internet applications. There is a local Youpers authentication and authorization provider, but the backend API also supports "social authentications", e.g. "log in with your Google account" or "log in with your Facebook account.

At the beginning of a session a client may authenticate via any of these channels. After the successful authentication the client receives an encrypted token. For all further requests the client needs to include this access token (a Bearer token) in the request. The token may expire or be invalidated after a predetermined time, after which the client has to re-authenticate to receive a new access token.

In production systems the authentication with access token / bearer token is the only supported form of authentication for backend API calls. The only backend call accepting other means of authentication is the "exchangeAuthForToken" call. In test systems / prototyping a client may also use HTTP Basic auth over SSL (meaning encrypted username/password) for each backend request cutting out the need for a token exchange before doing any requests.

| Type of environment | Authentication means / flow |
|---|---|
| Production | 1. Authenticate to platform using oAuth with:<br>- Youpers Identification Provider<br>- Social Logins (TBD for REVOLUTION: supported Social Identity providers)<br><br>2. Exchange Auth grant for Youpers access token<br>https: POST /api/login returns user object and Youpers Accesstoken in body).<br>(include the credentials you got from the Auth provider in step 1 in your request)<br><br>3. Authenticate further Backend Rest calls using the Bearer Token<br>- add: HTTP Header "Authorization: Bearer 34345nk3j4n5" |
| Test / Prototype | Use production version or:<br>- authenticate every call with HTTP Basic Auth. |

# 4 Data and API Design

## 4.1 Conceptual bases

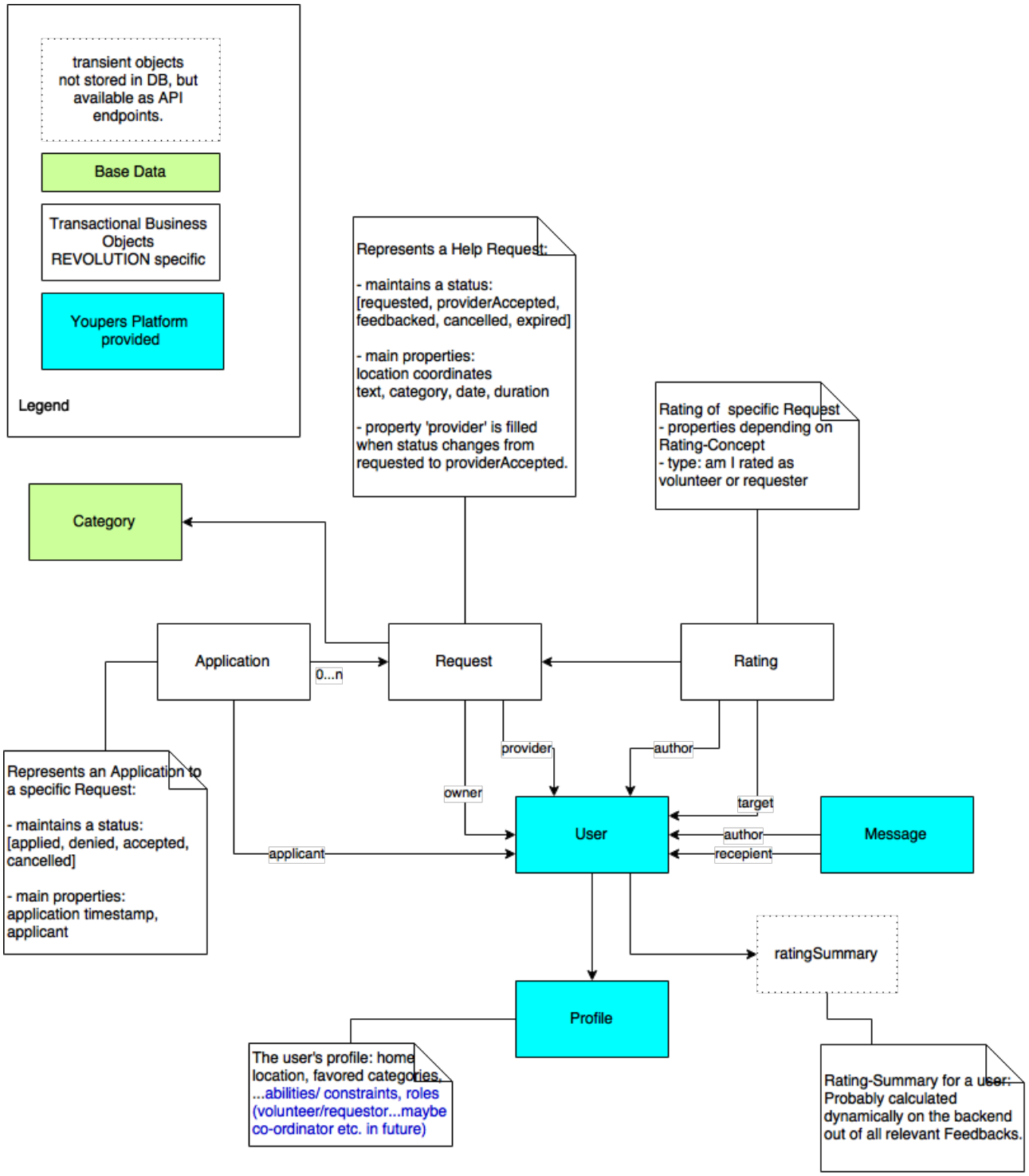
A fundamental solution design element of the REVOLUTION system is the data model of the backend API. This backend API will be used by all client devices – be it a smartphone app, a tablet, a web browser or a TV – so this is the common denominator of the system. It defines the language that is used to communicate on a technical level and also the language used by developers in different teams to communicate which each other.

The actual storage model - the physical data structure on the database server - is encapsulated in the backend server and therefore unknown to the application clients. It may be similar to the model defined in chapter 4.2, e.g. when using a database engine that support JSON as a data structure, or it may significantly differ, for example when using a relational database based structure.

This allows the backend implementation to change the physical data structure without affecting clients, as along as the API contract remains stable.

## 4.2 Data Model

A first version of the REVOLUTION API data model has been developed and needs to be further discussed and extended for iteration 3 and 4. The model shown below is based on the Youpers platform and extends the existing platform API data model with product/project specific components.




This image only serves as a preview for the online version here:

https://drive.draw.io/#G0B95w28y1cwlsRlZ3d0dhN3QyRFk

you need to login with your REVOLUTION google account that has access to our Google Drive.

## 4.3 API Design

The Youpers platform uses a standard RestFul API Design, allowing standard CRUD operations using the standard HTTP verbs:

| Operation | HTTP Verb | Examples |
|---|---|---|
| Create | POST | POST /api/missions<br>with body content as json {} |
| Read | GET | GET /api/missions/324324ab34cd<br>GET /api/missions?filter[name]="bla"&sort="id"&limit=10 |
| Update | PUT | PUT /api/missions/324324ab34cd<br>with full body content as json {} |
| Delete | DELETE | DELETE /api/missions/324324ab34cd |

The API is documented by the formal language specified in the Swagger spec (https://github.com/wordnik/swagger-spec). An interactive documentation of the spec is generated and published in the same format as this example (http://petstore.swagger.wordnik.com/#!/pet/addPet).

The API supports the following standard query parameters:

- Sort, limit, skip: May be used to implement a pagination like behaviour.
  - limit: limits the number of objects to be returned
  - skip: skip the first N objects that would be returned
  - Sort: sort the objects by a property
- Filter: Flexible search / filtering options
  - filter[property]=abc
- Populate: delivers a full object instead of a reference for properties who reference another object:
  - Populate='author' will deliver the full user object in the author property instead of just the id.

The REVOLUTION backend API is documented online at: https://rev-ci.youpers.com/

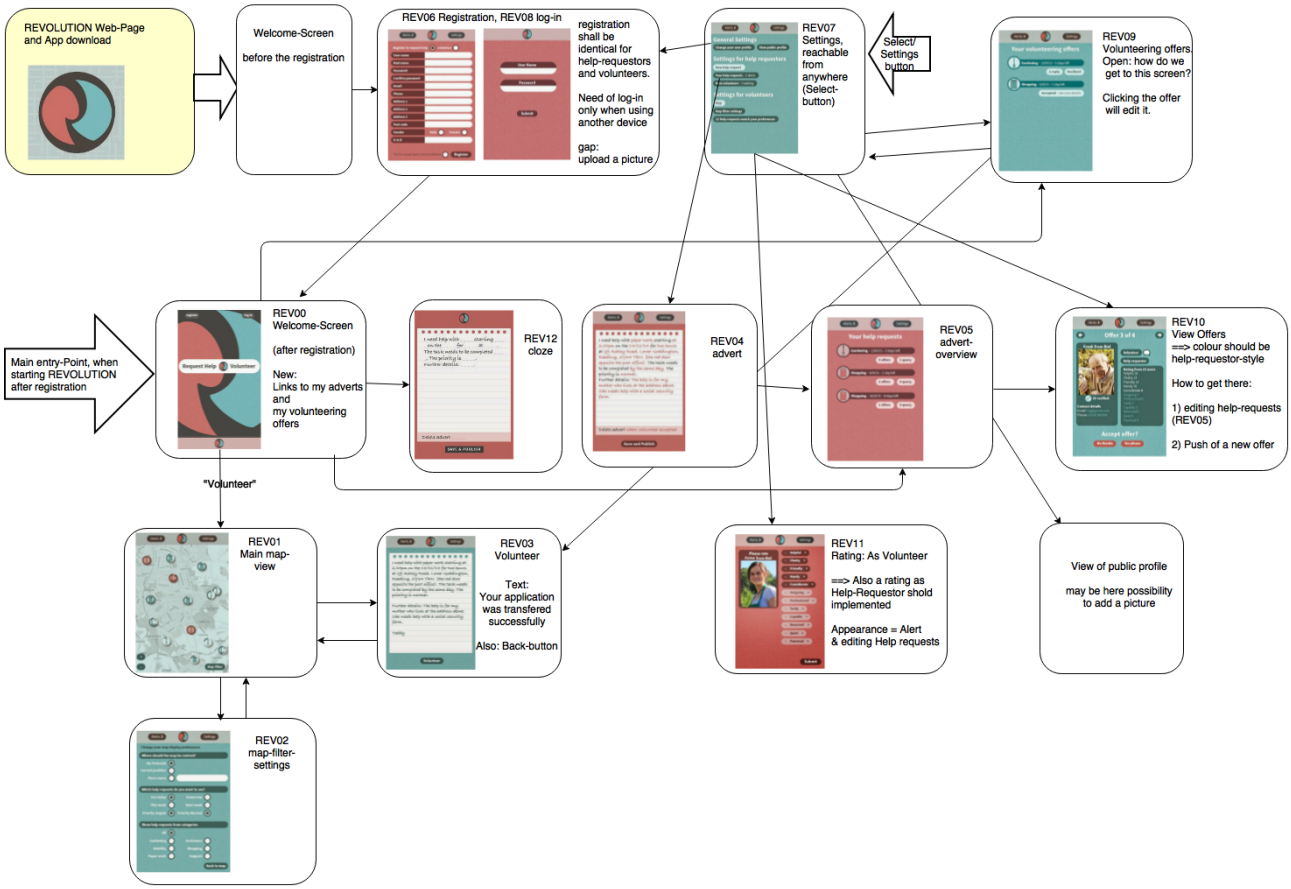
# 5 Human Machine Interface

In the REVOLUTION scoping workshop it was decided that the prime user interface (UI) for REVOLUTION will be a Smartphone-sized device. For the first two iterations the project therefore focuses only on these device types which usually have a screen diameter between 4 and 6 inches. It is open whether other sizes/platforms/ecosystems will be supported in later phases of the project.

The user interface has been designed and documented with two artefacts, a screen map and set of wireframes.

## 5.1 Screen Map

The REVOLUTION screen map shows an overview of all iteration 1 screens and shows the main navigation paths.




This image just serves as a preview, please consult the original screenmap

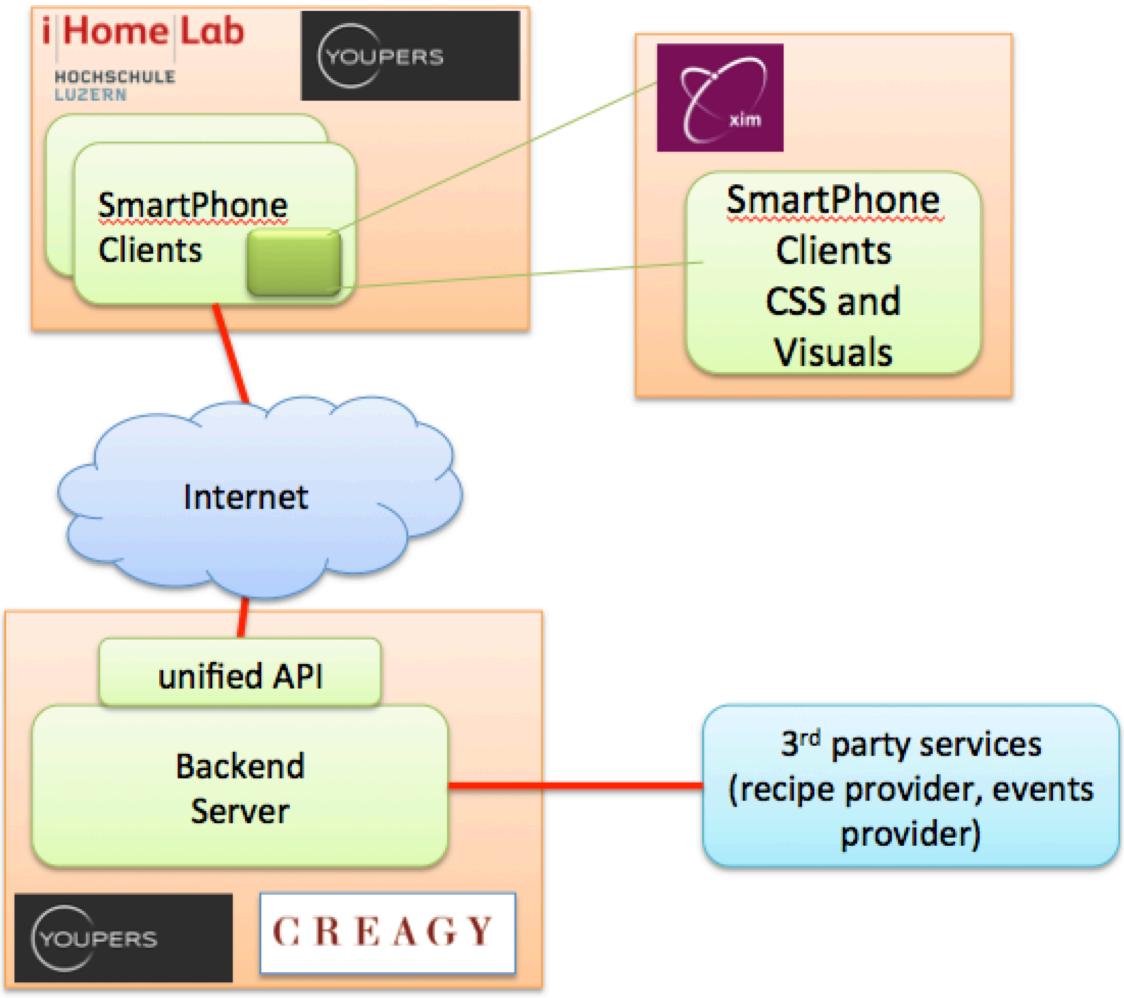
## 5.2 Wireframe screens

The detailed screen designs can be found online at http://www.xim.ai/revolution/home.html

# 6 Detailed Component Design

## 6.1 Component Breakdown and organization

The following graphic shows how the development team has split up the responsibilities and components during the first iteration of REVOLUTION development.




## 6.2 Backend and Database

### 6.2.1 Backend Application Server

The Youpers Application Server platform bases on the modern node.js technology stack. Node.js allows the creation of web servers and networking tools, using JavaScript and a collection of "modules" that handle various core functionality concerns. Modules handle File system I/O, Networking (HTTP, TCP, UDP, DNS, or TLS/SSL), Binary data (buffers), Cryptography functions, Data streams, and other core functions. Node's modules have a simple and elegant API, reducing the complexity of writing server applications.

Frameworks can be used to accelerate the development of applications, and common frameworks are Express.js, Socket.io and Connect. Node.js is primarily used to build network programs such as web servers, making it similar to PHP and Python. The biggest difference between PHP and Node.js is that PHP is a blocking language (commands execute only after the previous command has completed), while Node.js is a non-blocking language (commands execute in parallel, and use callbacks to signal completion).

Node.js brings event-driven programming to web servers, enabling development of fast web servers in JavaScript. Developers can create highly scalable servers without using threading, by using a simplified model of event-driven programming that uses callbacks to signal the completion of a task. Node.js was created because concurrency is difficult in many server-side programming

languages, and often leads to poor performance. Node.js connects the ease of a scripting language (JavaScript) with the power of Unix network programming.

Thousands of open-source libraries have been built for Node.js, and can be downloaded for free from the npm website. Node.js has a developer community centered around two mailing lists and the IRC channel #node.js on freenode. The community gathers at NodeConf, an annual developer conference focused on Node.js.

The REVOLUTON consortium's experiences with the node.js platform have been extraordinarily positive. The ability to reuse

### 6.2.2 Database

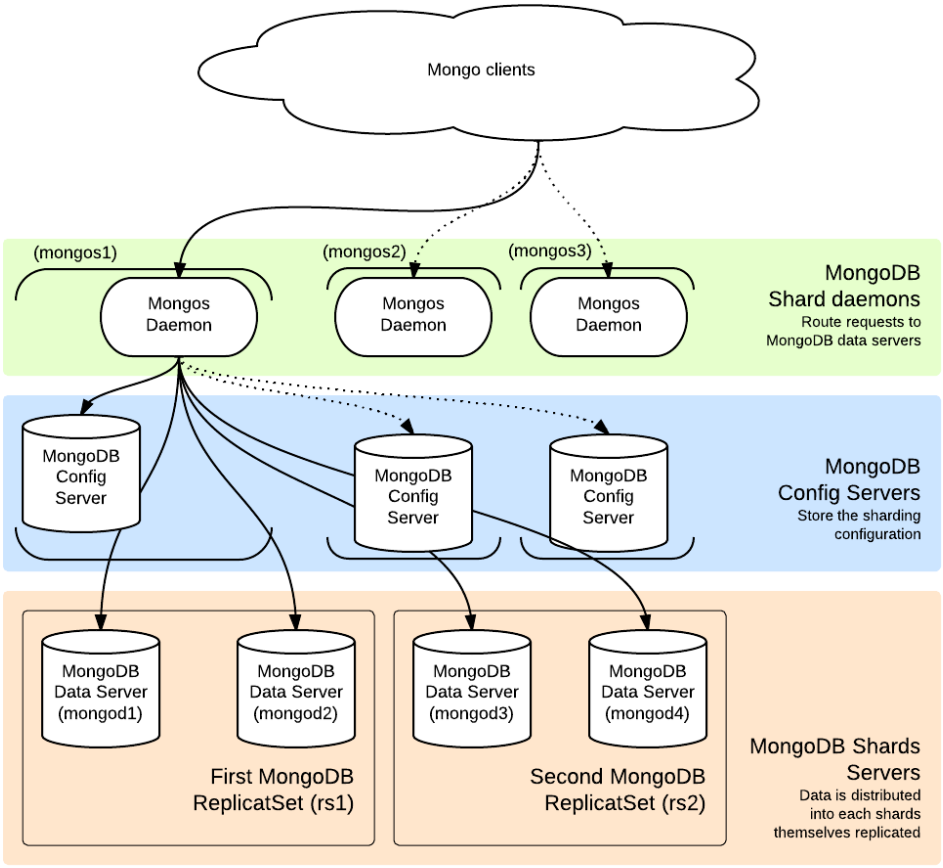
The Youpers Platform is built upon modern best practices for real-time web applications. It uses a NoSQL database. A NoSQL (often interpreted as Not only SQL) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling, and finer control over availability. The data structures used by NoSQL databases (e.g. key-value, graph, or document) differ from those used in relational databases, making some operations faster in NoSQL and others faster in relational databases.

Youpers has chosen to use the Open Source Product "MongoDB" as its database layer (https://www.mongodb.org/). Main reason for this choice was development efficiency and the very good integration into the main backend technology stack based on node.js. The mature ODM (Object-Database-Mapping) node.js Module mongoose (http://mongoosejs.com/) makes accessing the Database from the application server code simple and efficient.

For the prototype development in this AAL project a single mongodb instance is providing sufficient availability and performance and we can get a large benefit in development efficiency. Typical data size to be managed in a AAL prototype trial is a few Megabytes. Any database product can handle this efficiently as all data easily fits into fast memory.

But MongoDb also allows scaling if a dissemination of this project will require it. It is easily possible to scale using data sharding and replicasets as shown in the picture below.

MongoDB has proven its performance in production with thousands of deployments also with Big Data applications.




### 6.2.3 Activity Recognition and Prediction

An important part of the REVOLUTION project is the smart matching between a help requester and a potential volunteer. Making REVOLUTION smarter is the main objective of development iterations 3 and 4. To make REVOLUTION appear smarter the system needs to recognize the current activity a user is performing or even predict future activities a user might be planning to do and apply this knowledge to his potential interactions within REVOLUTION

The business process and requirement analysis conducted during iteration 1 and 2 has given us the following requirement areas where activity recognition will play its role:

1. **Realtime smart matching of a help request to a volunteer**
   Goal: Suggest the right open help request to the most likely available and willing volunteer at the best possible time to increase the probability of an acceptance by the volunteer without spamming a huge number of uninterested volunteers.

2. **Automatically suggest adding new help requests**
   Goal: Simplify the addition of a new help request for a help requester and increase the number of available help requests by triggering the adding of new requests. Recognize patters in the way help requesters request help to automatically suggest adding a new help request with prefilled default values.

3. **Recognize when a volunteer actually performs a request**
   Goal: recognize when a volunteer, that previously accepted a help request actually meets this help requester at the place of the requested help. With this knowledge REVOLUTION recognizes that the help request has most probably been fulfilled and it can trigger a message to request a rating of both, the volunteer and the help requester.

For solutions to all these requirements REVOLUTION needs to access personal user data - the most important elements of user data to recognize activities are – in order of importance:

- **Geolocation Tracking and Geofencing:**
  REVOLUTION can react a lot more adaptive if it knows a user's position and can relate this location to existing or new help requests or other users. New Geofencing APIs greatly simplify these interactions.
- **Past REVOLUTION transactions:**
  Revolution can recognize patters in the user's interactions with REVOLUTION and adapt its behaviour to meet these patters.
- **Calendar information:**
  REVOLUTION can react a lot more adaptive if it has access to a user's calendar information, since it can see whether a user is busy or can see scheduled future appointments including their location.

Requirement Analysis and the following Solution Design process has shown that with these data clusters we can achieve the desired goals. There is little additional value for the REVOLUTION service in trying to detect what exactly a user is or might be doing (as in "Shopping") as long as we know where he is, where might want to go, what his preferences are and his personal agenda.

#### 6.2.3.1 Realtime smart matching of a help request to a volunteer

There are three possible triggers that are to be considered for a realtime match of a help request to a potential volunteer:

- A new urgent help request is being entered in into the system. Then REVOLUTION needs to find and notify the most likely available and willing volunteers.
  REVOLUTION will notify a configured number of potential volunteers. The concrete number is a compromise between actually finding an accepting volunteer and not wanting to spam a large number of potential volunteer because at the end only one of them will be able to execute the request – first come, first serves.
  Whether a specific user is notified or not depends on:
  - distance between request and last known user location
  - the preferences of this user (categories)
  - the availability based on the user's calendar
  - the user's past interactions within REVOLUTION. (example: users who have already performed help requests for the same requester are preferred)

- A potential volunteer is moving (walking, driving, …) and passes the location of a open urgent help request within a specified radius. Whenever a potential volunteer is leaving home, REVOLUTION starts watching and notifies the user if he passes the location of a current, open and urgent help request nearby. This is made possible by geofencing APIs.
  REVOUTION sets up a fence around the user's home location to be notified when a user leaves his home and starts moving. Upon this event it will download the set of close by open urgent requests and setup new geofences for each of them to be notified when the user passes by closely. A larger geofence is used to be notified when the user leaves the area within the requests have been loaded already, so it can update the list of geofenced requests – remove the ones that are too far away now and add new ones.
- When a number of volunteers (all that were notified?) have ignored or denied an urgent help request notification so that there are no more options available, REVOLUTION needs to find more potential volunteers and notify them as well. It will apply the same algorithm as specified above, excluding users that have already gotten the same notification.

#### 6.2.3.2 Automatically suggest adding new help requests

This functionality relies mostly on analysis of past help requests of the same user and tries to find patterns in them to suggest adding similar help requests. The exact algorithms to be used are subject of the analysis work planned in Iteration 3.

#### 6.2.3.3 Help request execution recognition

This functionality relies on the available geofencing APIs on both primary platforms Android and iOS. These APIs allow an app to setup automatic triggers to be notified whenever a device enters a specific area – typically a circular area with a specific radius around a specific centre point. To fulfil the requirements REVOLUTION sets up a specific geofence with a small radius around the location of this help request on the volunteer's device whenever a volunteer is accepted to perform a help request. Now the device will notify the app whenever it is entering that fence.

This is a trigger for REVOLUTION indicating that the help request is being executed. After a delay given by the attribute "time required" of the help request REVOLUTION displays a message to both involved users to confirm the execution of the help request and rate the other counterpart.

REVOLUTION uses a specific an existing cordova plugin to achieve this functionality:

https://github.com/cowbell/cordova-plugin-geofence

Operating systems manufacturers have been putting in a lot of effort to these geofencing APIs in the last years. Especially being battery efficient has been a big goal. REVOLUTION does not have enough data yet to judge whether the value that this functionality is adding is worth the additional battery drainage. This is subject of the planned field trials.

## 6.3 Structure of the Client App

The Revolution client application runs on portable devices featuring different operating systems and is designed for small displays. An iPhone 5s and a Nexus 5 are used for development featuring a 4 inch display and a 4.95 inch display respectively. With these devices, two of the major platforms namely iOS and Android are addressed thus allowing Revolution to run on the vast majority of smartphones currently available.

This section focuses on the internal architecture of the Revolution application by explaining the following three aspects in greater detail: general application architecture, internal app structure and notable app features.

### 6.3.1 General Architecture

Revolution has been developed using the Ionic framework. To develop smartphone apps for different platforms, one either has to develop native apps specific to exactly one platform or choose a hybrid approach. The latter has the advantage that an app is developed once and can be converted to run on multiple platforms with little extra effort. The Ionic framework allows exactly that: develop once and deploy on multiple platforms such as iOS, Android and Windows Phone. To achieve this platform compatibility, Ionic uses technologies every smartphone supports and thus is able to offer a modern and flexible alternative to developing apps native. To take advantage of existing work and knowledge, Ionic combines existing software while adding functionality mainly addressing the app's look and feel and UI interaction of the app to the mix.

Ionic is built on the Cordova platform which in turn uses web technologies that run in a target device's web view while maintaining the look and feel of a native app. Web standards are supported by most smartphone platforms including iOS and Android meaning that such devices understand technologies such as HTML, CSS and JavaScript. The interface is typically designed using HTML and CSS and the functionality with JavaScript. To get access to device specific functionality such a sensors or the contact list, Cordova uses plugins: a plugin typically offers a JavaScript API for the web view to interact with and another part containing the JavaScript equivalent code in a language the device platform understands.

To make it easier to build an app's frontend, Ionic is based on AngularJS. The latter connects HTML, CSS and JavaScript and introduces concepts and structures to tie these three together in a consistent way. Together with Ionic's stylesheets optimized for mobile layouts, the already familiar functionalities provided by Cordova and AngularJS offer a compelling alternative to develop smartphone applications.

The last step is to tie together the different parts of the developed app into one application and its platform specific structures to actually run on a smartphone. When developing the application on the computer, its structure is optimized for development. To build the smartphone app, Ionic offers processes that convert the development structure into one the smartphone understands. To do so, Ionic takes advantage of existing applications that execute these build processes such as Gulp. Once build, the developed app can be copied onto a smartphone for local testing and/or distributed via the existing market platforms such as iOS app store and/or Google Play.

### 6.3.2 Internal App Structure

To develop Revolution, we choose to group its content by topic (so called states in Ionic). The state basically represents a screen such as the login screen with its interface definitions and associated functionality. The seven states are in seven separate folders each containing the related HTML and JavaScript files. Revolution also uses an index.html containing all the references to the used resources such as the aforementioned states' files.

Another concept provided by Ionic and used by Revolution is directives. A directive is a component building block that can be used at different places in the code without having to rewrite the code. An example of such a building block is Revolution's own progress bar including its interface definitions (written in HTML) and associated functionality (written in JavaScript). There are three directives in the client app and they are referenced seven times.

Revolution uses already existing and newly developed services that allow functionality to be used across the app. One already existing service used by Revolution is a translation service allowing easy integration of different languages and switching between them. Another existing service is RESTangular which allows easy communication with the backend server by providing common GET, POST, DELETE and UPDATE requests. The user service is a newly created one which encapsulates the functionality associated with a user such as log in, log out, update, get information and many more.

Revolution uses different plugins to access phone specific functionalities. An overview is provided in Table 1.

**Table 1 The plugins used in Revolution.**

| Plugin | Purpose |
|---|---|
| com.google.playservices | Required for Google features such as Google Maps |
| com.ionic.keyboard | Makes interacting with the keyboard easier |
| com.phonegap.plugins.PushPlugin | Enables to receive and process push notifications. |
| cordova-plugin-whitelist | Controls which URLs the WebView itself can be navigated to. |
| org.apache.cordova.camera | Accesses the phone's camera. |
| org.apache.cordova.console | Add additional functionality to the console.log() command. |
| org.apache.cordova.device | Defines global object containing information about the device's hardware and software |
| org.apache.cordova.file | Implements a File API allowing read/write access to files. |
| org.apache.cordova.file-transfer | Manages file uploads such as the upload of a new profile picture. |
| org.apache.cordova.geolocation | Provides information about the device's location, such as latitude and longitude |
| org.apache.cordova.globalization | Obtains information and performs operations specific to the user's locale, language, and timezone. |
| cordova-plugin-statusbar | Customises the device's status bar |
| mobi.moica.whatsapp | Integrates WhatsApp funtions |
| com.cordova.plugins.sms | Allows to easily send SMS |
| cordova-plugin-crosswalk-webview | Makes Cordova application use the Crosswalk WebView instead of the System WebView |
| cordova-plugin-splashscreen | Displays and hides a splash (or load) screen during application launch |

| | |
|---|---|
| cordova-plugin-media | Enables Revolution to record and play back audio files on a device |
| https://github.com/cowbell/cordova-plugin-geofence | Geofencing API for Android and iOS |

### 6.3.3 Offline Capabilities

The output of the requirement analysis phase has shown that offline capabilities can only provide very limited value to a Realtime volunteering Online solution, because - well - it should be realtime and online.

However there is a small, very specific part of REVOLUTION that needs to support offline capabilities. When a help request is settled – meaning a requester has accepted a volunteer – and before this request is fulfilled, it is crucial that both, the requester and the volunteer have access to all data of this help request, especially the contact information and the place where the request should be fulfilled needs to be available at all times.

REVOLUTION solves this by storing this specific set of data into the webview local storage of the phone and it offers a very limited functionality set when the device is offline. In offline mode REVOLUTION allows to look at two lists:

- my help requests
- my applications

This offline data is updated whenever the app gets the data in online mode.

Offline capability is planned to be implemented in Iteration 3.

### 6.3.4 App Features

There are a number of features specific to the Revolution application. In this section, the user interface, app permissions and other notable features are introduced.

The UI is based on the wireframes developed with XIM (section 5.2). All delivered wireframes have been implemented resulting in 22 HTML files each dedicated to a screen. Adding the 39 JavaScript files defining the functionality behind the interface, over 10.000 lines of code have been written.

The Revolution client app requires several permissions to run properly. Permissions are necessary as they ensure that any application neither has the right to anything that impacts other applications, the operating system or the user. For the Revolution app, there are 16 required permissions for Android and 6 for iOS. Among these permissions are Internet access, read and write local storage and phone contacts access rights.

#### 6.3.4.1 Special App Features: Google Maps integration

A crucial visual app element is the map showing the locations of the help requests. By looking at the map, users who want to help see the open help requests nearby. There are different maps services publicly available such as Google Maps, Leaflet, Mapbox and OpenStreetMap. A more detailed comparison can be found here: http://www.creuna.no/blogg/leaflet-mapbox-a-google-maps-competitor/. We chose Google Maps because of the experiences we already have in the consortium and the additional Google APIs like Google Places and Geocoding that are used in the application. Pricing is not an issue as the services are nearly free for our required usage.

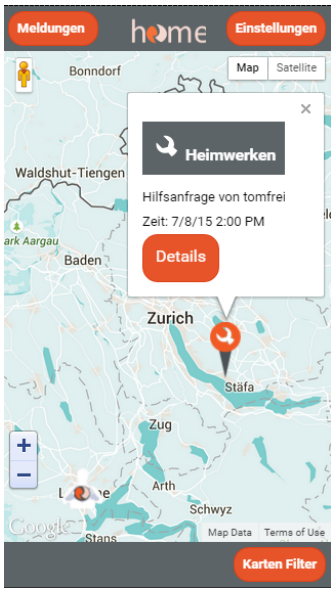



**Figure 1 Revolution's help screen. The current user location is shown as well as a help request including its information box.**

Using the Google Maps API for Revolution allowed the easy integration of a variety of features that come with the API. The custom styled map is part of one of the central app screens: the help screen (Figure 1). It features a map showing the points of interest including the user's position and help requests. The map is positioned depending on the user's location and the help request in close proximity using the Geolocation service. The locations are labelled using markers which indicate different help request categories. In addition, information boxes are shown upon a click on a marker detailing the request's category and time. Besides communicating the help request's location and category, the marker is a link to navigate to the request's detailed view. Too many markers in close proximity are grouped together at lower zoom levels and split again when the zoom level is sufficiently high.

Implementing the map using the Google Maps API was a straight forward and problem free experience. Because of the hybrid nature of the Revolution app it was not obvious which problems will arise when using 3rd party APIs. To our surprise little code is needed to achieve quite complex functionality on maps.

#### 6.3.4.2 Special App Features: Seamless communication integration

Revolution's main objective is to bring people seeking help and those offering together in a timely manner. To bring people together Revolution offers them to communicate via existing and already familiar channels such as calling, SMS, email and WhatsApp. Rather than having to remember specific contact details, these communication functionalities are accessed via buttons within the Revolution application.

Implementing the across communication channels functions was more challenging than expected. Unique URL-schemes each representing a specific communication channel are used to invoke a corresponding channel. Unfortunately, these URL-schemes are not properly standardized and support on different device types and versions is varying; especially Android is making this a lot more difficult than it should be. Of the arising issues, proper permission settings were the most challenging: there is very little and only very inconsistent documentation available. There are at least three different permission types that seem to be involved:

- The permission specifying the access rights of the web view specified in the config.xml
- The permission specifying the access rights the native app is requesting from the operating system specified in the config.xml also
- The Content Security Policy (CSP) permission set in the index.html META tag

- The user settings on the smart phone

Instead of using the URL-schemes we ended up using two Cordova plugins namely mobi.moica.whatsapp and com.cordova.plugins.sms.

#### 6.3.4.3 Special App Features: Internal direct Communication Channel

An internal communication channel like a chat / bulletin board, scoped to one help request between a help requester and a potential volunteer is planned to be added in the Development iteration 3. Screen designs have been prepared and a data model has been developed for this.

First user feedback has now shown that most users prefer to use existing communication channels they already know like Phone, SMS, WhatsApp or email instead of "just another chat/messaging application". This user feedback is being confirmed as part of the Usability tests.

Therefore integration of this feature has been postponed until a confirmed user feedback is available. Depending on the user feedback this feature might be swapped out and as a replacement the integration with existing communication channels might be further improved.

#### 6.3.4.4 Special App Features: Speech recognition

As part of the initial requirements it is planned to integrate speech recognition into the REVOLUTION app. Since those requirements have been written as part of the REVOLUTION proposal the two mobile operating systems Android and iOS have progressed tremendously in this area. As a consequence adding speech recognition into a mobile app has become a very simple task. All the needed APIs are easily available.

Business and Requirement analysis conducted by the consortium partners has shown that the area in REVOLUTION that would most benefit from Speech Recognition is "entering a help request". Therefore it is planned to add Speech recognition abilities to this screen as part of Iteration 4.

## 6.4 Web Frontend

REVOLUTION will use a web frontend for two distinct purposes:

- **Content Management and Administration App:**

  We envision a restricted management application where administrators can review existing help requests and applications, evaluate the service usage, review user's actions, ban users who are abusing the side and communicate with users

  The Administrational information is only needed as soon as we start a larger trial – for testing and usability tasks it is possible to rely on information that the Youpers Platform provides as is. Short term we can also use specific database queries to get the needed information.

  Timeframe for this app: Iteration 4 or after the AAL project.

- **Web App for users (team leaders and participants)**

  To more easily reach more users we may also need an access to the Revolution service through large-screen web browsers, predominantly used on corporate desks.

### 6.4.1 Architecture of the web app components

The web apps will be based on the YOUPERS standard web app architecture. This is a "Single Page App" (SPA) talking to the same API as all our mobile clients. This ensures that the same security and privacy measures are in effect for all clients.

The main framework used is angular.js, which is also the main building block of the ionic framework used for our mobile applications. Therefore we will be able to reuse a large part of the code written for the mobile apps also in the Web app.

# 7 External Interfaces

In the first and second iteration of REVOLUTION, no external interfaces will be deployed. It is planned, to use a activity tracking provider to improve the smart matching algorithm. State of the art API-programming will be used.

Furthermore third Party Identity Providers could be added – but only when an end user need is clearly identified.

This chapter will be amended as a preparation of iteration 3.

# 8 System Integrity Controls

Sensitive systems use information for which the loss, misuse, modification of, or unauthorized access to that information could affect the conduct of programs, or the privacy to which individuals are entitled.

Developers of sensitive systems are required to develop specifications for the following minimum levels of control:

- Internal security to restrict access of critical data items to only those access types required by users
- Audit procedures to meet control, reporting, and retention period requirements for operational and management reports
- Application audit trails to dynamically audit retrieval access to designated critical data
- Standard Tables to be used or requested for validating data fields
- Verification processes for additions, deletions, or updates of critical data

Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.

Youpers follows these standards and implements the mentioned system integrity controls for all "production" systems and services. Any system holding data of real users that are not explicitly instructed to only enter test data into that system is considered a "production" system.

For REVOLUTION a "production" system will be set up as soon as a User Trial will be conducted – currently planned after iteration 3.