Архитектура проекта Альтавиэль

## Глоссарий

**Реквестор –** пользователь системы, автор запроса о помощи.

**Сообщество –** множество людей, зарегистрировавшихся на сервисе, указавших свое стандартное местоположение в различные временные рамки.

**Хелпер –** пользователь системы, отозвавшийся на запрос и подтвердил желание и возможность помочь.

# Введение

## Определение проблемы

Нет простого способа попросить о помощи незнакомых людей, добрые люди не имеют возможности помочь.

## Требования

В подробной спецификации нет необходимости, поэтому ее решено описать в архитектуре.

Процесс разработки - эволюционное прототипирование, поскольку возможны частые изменения в требованиях. Разработка будет вестись по спирали.

**Шаг 1.** На главной странице загружается карта, производится попытка определить координаты пользователя. Если не удалось, предлагается выбрать местоположение. Координаты хранятся локально (Local Storage).

**Шаг 2.** Добавлена форма запроса. Указанная информация и координаты отправляются на сервер и хранятся в базе. На главной странице появляется toogle для отображения ближайших заявок. Заявки запрашиваются целиком. **Позже необходимо будет продумать синхронизацию только измененных.** Также на отдельной странице можно просмотреть запросы списком (пока без фильтрации). При клике на запрос на карте или в списке открывается страница с деталями запроса.

**Шаг 3.** Место главной страницы занимает форма авторизации/регистрации по e-mail. Регистрация упрощена. Если нет такого аккаунта, на указанный e-mail высылается письмо с просьбой подтвердить адрес, после процедуры авторизации пользователь может зайти с этой учеткой. Если учетка есть, происходит аутентификация. Впервые вошедший пользователь попадает в редактор профиля и вносит информацию – места, где он регулярно бывает (работа, дом), ник, ссылки на профили в социальных сетях. В деталях запроса появляется возможность ответить на запрос. Отправляются данные авторизованного пользователя.

**Шаг 4.** Вводится система рейтингов. Возможность отблагодарить хелпера. Приложение получает статус публичной беты. Последним шагом будет превращение веб-приложение в нативное приложение устройтсва (PhoneGap или альтернативы).

Далее возможны вариации. Идея монетизации – благодарность купленными виртуальными подарками, реклама служб, специалистов.

Точки ввода – карта с геолокацией, профиль пользователя, фильтры географических данных, форма запроса, форма ответа на запрос. Возможно – чат.

На выходе – отфильтрованный по дальности от определенной точки список запросов, детали запроса, детали о пользователе, карта с указанием мест запросов.

Прототип: <https://www.fluidui.com/editor/live/preview/p_tZzVmirbCMDQmkGHleJDO9enkh7JD4G1.1387885530840>

### Альтернативные решения

**Группа вконтакте –** хороший вариант, но требуется модерация, управление неудобно, нет системы рейтинга, сложно работать с картой, плюс – можно посмотреть профиль и понять что за человек.

**Ushahidi –** практически готовое решение, но трудно кастомизировать. Можно брать как пример.

### Целевая группа

Жители Казани и пригородов от 16 до 35 лет готовые помочь ближнему своему и обладающие смартфоном, планшетом или компьютером с доступом в интернет.

## Общее описание системы

Сервис представляет собой простую систему заявок с привязкой к географическому положению.

Для разделения данных от представления (для обеспечения большей гибкости в условиях эволюционного прототипирования) решено было использовать паттерн проектирования **MVC** и как следствие, старейший из представителей вида - **Backbone.js**.

Серверная часть доступна по **http** через **RESTful** **API** для простоты реализации и по причине заслуженного статуса безопасного метода взаимодействия.

## Базовые usecases

* CRUD запросы
* Просмотр списка запросов (на карте или списком)
* CRUD профиля пользователя

## Хранение данных

Решено использовать универсальную ER-модель данных. Из требований можно выделить две сущности**: Request** и **User.** Для реализации common places возможно будет добавить сущность **Places** для хранения мест и периодов времени когда пользователь там бывает.

Система хранения данных должна уметь обрабатывать одновременно множество запросов, быть простой в составлении запросов и работать со связями один ко многим и многие ко многим.

У пользователя может быть несколько запросов, в которых он автор и несколько запросов, в которых он хелпер. У запроса может быть только один автор и только один хелпер. Таким образом, реализуется модель один ко многим.

\_id : ObjectId  
name : String  
nickname : String  
email : String  
requests : [{  
 type : ObjectId  
 ref : ‘Request’  
 }]

User

Request

\_id : ObjectId  
title : String  
lat : Number  
long : Number  
description : String  
type : Number  
requestor : {  
 type : Number  
 ref : ‘User’  
 }  
helper : {  
 type : Number  
 ref : ‘User’  
 }

Возможные запросы:

* Выборка всех строк из таблицы Request. (синхронизация с локальным стораджем); включая информацию об авторах.
* Информация по одному пользователю, включая список его запросов.
* Добавление строки в Users
* Добавление строки в Requests

Для обеспечения гибкости и неопределенности с выбором СУБД, за предоставление данных будет отвечать отдельный класс, который будет предоставлять данные в стандартизированном JSON формате, но логика работы может быть переписана для другой БД.

Варианты БД – mysql, mongodb. **TBC**

# Frontend

Фронтенд – одна html страница с адаптивным дизайном. Для ускорения разработки решено использовать **jquery mobile**. В качестве альтернативы рассматривался bootstrap 3, но в приложении не так много информации, чтобы делать полноэкранную версию. В основном приложение будет использоваться с мобильных устройств.

# Backend

На втором и следующем этапах необходим backend. Он будет выполнен в виде nodejs RESTful API, который будет выдавать index.html и json в необходимом формате по запросам. Фронтенд при этом – одна динамическая страница. База данных – mongodb до тех пор пока она будет удовлетворять потребности. В процессе разработки нужно будет понять принцип работы связей и если все устраивает – переходить на mysql нет необходимости.

# Процесс внесения изменений

Внесение изменений в требования или архитектуру должны быть отражены в соответствующих документах.

Используется система контроля версий Git. Репозиторий на гитхабе <https://github.com/amIwho/altaviel.git>.

Две ветки – production и master. Пока используется только master. Production будет использоваться для мажорных версий.

# Методы и инструменты

Jquery mobile  
Google maps api