# Project Salutogen

# Sautogen Server

Written in Microsoft .NET Core C#  
IDE used: Visual Studio

Protocols used: HTTPS, SSL (currently self-signed)

The Salutogen server is implemented using .NET Core C#. The server consists of two parts:

1. Web application
2. Web API

## Web application

The web application provides an interface for the end-user to apply CRUD (Create Read Update Delete) operations with an intuitive and simple user interface. Authorization is implemented using Identity User, while the database operations are implemented with Entity Framework Core which in turn is using the Code first approach to design the database.

The current database provider used is Microsoft SQL Server. However, this can easily be changed to other database providers supported by .NET Core.

## Web API

The main purpose of the web API is to provide database read and write operations for external application such as the Salutogen android application.

To increase the security, the WEB API supports a very limited subset of CURD operation. For example, it is not possible for a third-party application to delete data. Please refer to the Appendix – section API to get an overview of the tested API functionalities.

# Salutogen Android application

Written in: Java  
IDE used: Android Studio

Salutogen android application is an intuitive and user-friendly activity monitor which collects data about the user's step activity and send them to the web server through the web API.

The application consists of these parts:

1. Main user interface
2. Services
   1. Data acquisition service
   2. Sync service
3. Database

## Main user interface

The graphical user interface allows the administrator to set-up connection and user parameters such as the GUID (Global Unique Identifier) which identifies the user and the API URL to use for communication with the Salutogen Server. Furthermore, it gives the user information about the application, such as errors or the last synchronization time.

The graphical user interface is divided into various activities

1. Main Activity
   1. Gives useful information about the status of the application
2. Settings Activity
   1. Gives the user to edit the settings for the app.
   2. Ability to reset/clear all app data including the SQLite database.
3. User agreement Activity
   1. Every time the GUID is changed, the new user must review the user agreement. User Argument shall contain information about how, what and why the data is collection and how the collected data is going to be used.

Language: all the language resources are in a separate file. Currently only in Swedish.

## Services

### DAQ service

The data acquisition service is responsible to collect the sensory information from the phone and store them in the local database. Every time a step is detected, an event is raised and the number of steps will be written to the local SQLite database.

The DAQ service is always running the background if the application hasn’t encountered any errors.

## Sync service

Every time certain criteria's such as battery status, network availability and periodicity are met, the sync service creates a new background thread to upload the data stored in the database to the server using HTTP post requests. The maximum number of sync failures can be decided by the developer. The sync service is always running periodically in the background if there the application is running normally.

After a successful sync, the uploaded data is deleted from the local database.

### Errors

# Distribution method

The distribution of the Android app has not been discussed yet.

# Appendix

## API

**Introduction**  
API to read/write access user activity data acquired from the Android devices.

**Overview**  
All the requests are using JSON and the charset is utf-8.

**Authentication**  
OAuth 2.0

**Error Codes**  
Standard HTML 5.0 errors

**Rate limit**  
Not available

GET **User**

[/api/User/502ca87e-c18a-43b8-a4d2-cea54b83ffac](https://localhost:44337/api/User/502ca87e-c18a-43b8-a4d2-cea54b83ffac)

GET **Steps**

<https://localhost:44337/api/user/cc906675-aedb-4ef9-b1ac-7acb3e035f2c/steps>

POST **Steps**

<https://localhost:44337/api/user/502ca87e-c18a-43b8-a4d2-cea54b83ffac/steps>

**Headers**

|  |  |
| --- | --- |
| Content-Type | application/json |

**Body**

raw (application/json)

[{

"Timestamp": "2017-04-01T19:01:55.714942+03:00",

"Count" : 1},{

"Timestamp": "2017-04-01T19:01:55.714942+03:00",

"Count" : 1}, …]