DIGIFiT

Towards Standardization in Communication to Online Fitness

Date:	3rd May 2010
Name Student:	Peter Gels
Student Number:	1536478
Name Company:	DIGIFiT
Project Supervisor Company:	drs. P. Braam
Project Supervisor VU:	prof.dr. A. Eliens
Second Reader:	dr. M. Klein

Table of Contents

1 Introduction

1.1 Definition of the Problem

1.1.1 Situation and Problem Description

In March of 2009, VirtuaGym was released in a closed beta. In the meantime, the platform has been tested by a large amount of users and VirtuaGym has now graduated out of open beta. In a short term the online gym will be launched as well, in which members of VirtuaGym can exercise with others through the internet from out of home. In time, VirtuaGym will become a part of Vitalence, which will be a collection of health and fitness products and services.

DIGIFiT has the vision that VirtuaGym needs to be an open platform, on which product developers can also connect their own applications. This means that producers can connect their interactive exercise equipment and/or exercise games to the lifestyle management and competition system of VirtuaGym, so that users can keep track of among others their burned calories within VirtuaGym, or compare themselves to others in an online competition. This allows third parties to add the lifestyle tools, the social factor and the competition element easily to their products for a small investment. This connection will be available under a name that has yet to be decided. There are already a number of parties that are going to connect their products to VirtuaGym.

1.1.2 Goal of the Assignment

This project consists of an independent research on how VirtuaGym Connect can be set up in the most optimal way, where the key is that it needs to be an open standard. This research will look at the requirements for such an open communication standard in exergaming and life-style, taking into account business requirements with respect to security, scalability, identification, simplicity and standardization. An assessment of the data that will be communicated also needs to be made.

During the research, a working proof of concept for an open communication standard will be developed, where the results of my pre-research will support on workability and efficiency. As a test case, contributions will be made to the development of the prototype for a fitness application that will communicate with VirtuaGym through the API.

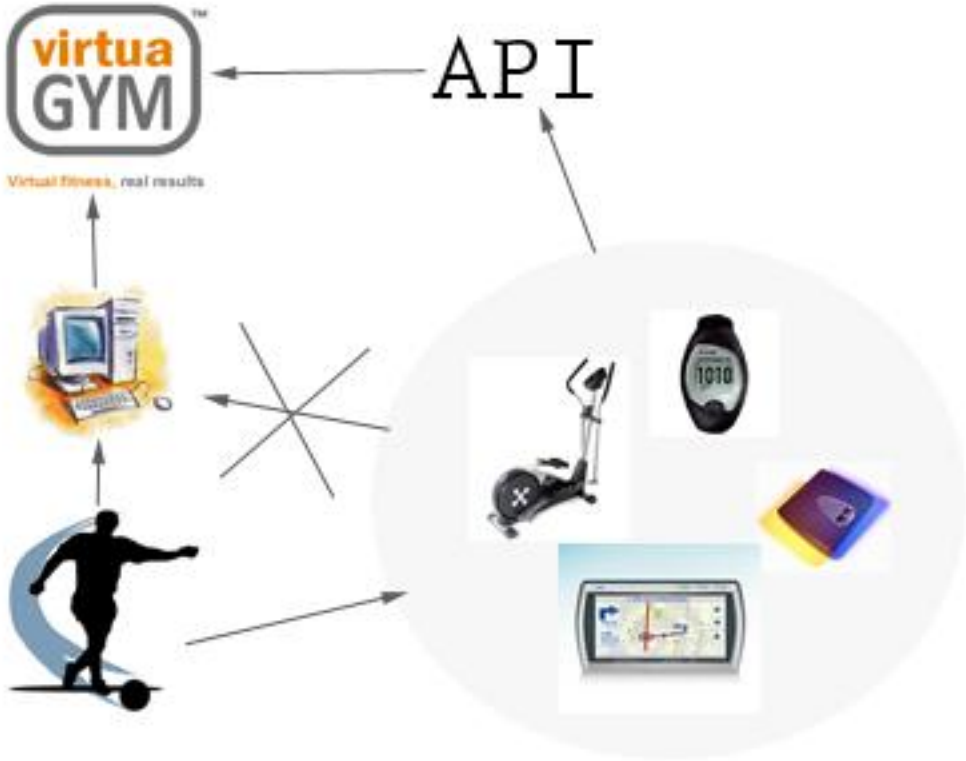


Figure 1: a description of the purpose of the API: people can now have the opportunity for automatically submitting their content to VirtuaGym, instead of having to do this manually with a PC.

1.2 Research Questions

For this projects, we formulated a number of research questions, to help to focus the assignment. The Central Research Question for this project will be:

- **What are the elements of an open communication standard in fitness and lifestyle platforms, to which parties as VirtuaGym can adhere?**

In order to answer this question, a number of Part Questions have been defined that split up this question in various areas:

- What kind of architecture is the most suited for a connection to an online fitness and lifestyle platform with regards to security, scalability, identification, simplicity and standardization.
- What kind of standards exist in the Fitness and Lifestyle Application Development Community and what does the landscape of Fitness and Lifestyle Applications look like?
- Which techniques and technologies are suited best to implement the API and how should it display its information to the third parties?
- What are the critical success factors of a fitness and lifestyle communication standard?

In order to develop a connection to an online fitness and lifestyle application, certain considerations need to be made: in what form will this be the most usable? How do you make such a thing secure?

How can it be made so that it can grow easily in the future? How will people log in? How do we turn it into a standard for other parties to adopt as well? This is what the first part question is about.

DIGIFiT is also looking for easy and accessible ways to make connections to fitness equipment in order to exchange information, for example talking to a home trainer to automatically get its data displayed on VirtuaGym. The second part question aims to discover the possibilities here, and what kinds of standards already exist that might be able to pull this off.

During this research, the API for VirtuaGym will be developed .The third part question examines in what way this can be done, and which methods are the optimal in making it as effective as possible.

The fourth part question examines what is the minimal needed to achieve a successful fitness and lifestyle communication standard: what is necessary to achieve such a communication protocol?

There were some areas that we also decided not to focus on. This thesis will not deal with trying to stream real-time data, which is in a completely different scope.

1.3 Document Organization

The organization of this document is as follows. The section “Current Standards in Communication to Fitness Equipment” contains the report of a research to methods that can be used to connect to fitness equipment, and the ways in which this can be beneficial to VirtuaGym and Vitalence. After that “Practices in Online Fitness and Lifestyle Communities” outlines a small case study of similar applications to the API. The “Architecture” section will describe various practices, techniques and technologies in terms of developing an API.

“API Design Decisions” will outline the design of the API. The ‘Proof of Concept” section will describe the proof of concept we made for the API. “Possible Uses” is a small chapter dedicated to the possible uses for the API that was developed, and the “Conclusion” section will then attempt to answer the research questions. The “Appendix A” contains all of the scenarios and use cases. Appendix B contains various diagrams that attempt to describe the API, Appendix C will outline a proposal for a design we made for Google Maps Integration for the landscape study and Appendix D will contain a list of sources that can communicate to VirtuaGym.

1.4 Reading Guide

The different sections of this thesis aren’t all meant for the same audience. In this sub-section follows a reader’s guide for management (containing the sections that are interesting from a business perspective) application developers (for people who want use the API that was developed in this project) and developers (developers at VirtuaGym).

The sections that are written for a management perspective are section 2 (Current Standards in Fitness Equipment), 3 (Practices in online Fitness and Lifestyle Communities), 7 (Possible Uses), 8 (Conclusion) and 12 (Appendix D: Sources of Fitness Data for VirtuaGym.

The sections that are written for the perspective of application developers are sections 4 (Architecture, in particular sections 4.1 (Possible Frameworks), 4.3 (Identification) and 4.4 (Security)), 5 (API Design Decisions), 6 (Proof of Concept), 7 (Possible Uses) and 8 (Conclusion).

The sections that are written for people with a development perspective are 3 (Current Standards in Fitness Equipment), 4 (Architecture), 5 (API Design Decisions), 8 (Conclusion) and 12 (Appendix D: Sources of Fitness Data for VirtuaGym).

2 Current Standards in Communication to Fitness Equipment

VirtuaGym is an online fitness platform. There are a lot of opportunities for connecting it to all kinds of real-life fitness equipment, and allowing them to communicate with each other. In this section we sketch a global look of the environment of communication to fitness equipment, and it will discuss a number of the different possibilities that VirtuaGym can take to connect to these technologies. It's followed by a conclusion which will give an advice to VirtuaGym about a number of concrete ways in which this can be used for its platform.

Google Maps[1] is becoming more and more useful to track running and cycling routes, for example. At the time that this thesis was written, they supported a wide variety of ways to store routes, to calculate routes, and recently they even integrated Google Earth, which is especially interesting to see when you've been cycling or running in a mountainous area.

Social media also are increasingly more popular for fitness users. VirtuaGym and The Daily Burn[2] are examples in which users can't just keep track of their own fitness results, but also encourage each other. Other online communities as forums, Facebook, Hyves and Twitter have also proved to be very popular when it comes to people encouraging each other to lose weight and share tips.

Mobile technologies are also very popular in assisting people to get in shape and do more about fitness. The android mobile phone for example features a ton of different applications with a wide variety of functions. Take for example Calorie Counter [3], a mobile application that lets you track the calories you consume, a virtual personal trainer, an application that tracks where you've jogged, an application that analyzes your statistics when you run and one to keep track of your weight. Then there are also manufacturers of mobile devices such as GPS's and Heart Rate Monitors, such as Polar and Garmin

2.1 Overview Fitness Equipment and its Technologies

This section is meant to give an overview of what kind of fitness equipment exists today, and what kinds of standards they use. First we'll discuss various fitness equipment and their manufacturers, then we'll look at some specific standard that's widely adopted: CSAFE, ANT+, Polar and then a small paragraph containing information about iPhone Applications.

2.1.1 Industry Standards

There are many different producers of fitness equipment, however in terms of sales it is a concentrated market: in the United States, out of the active 100 companies, the top 5 generates 50% of the total revenue[4]. This top 5 consists out of Cybex International, ICON Health & Fitness, Life Fitness, Precor and Nautilus.

Cybex international sells both strength and cardio equipment. The cardio equipment, such as treadmills, cross-trainers and home-trainers use the CSAFE protocol in order to connect to the outside. They communicate with a dual RJ-45[5] cable, or simple Ethernet cables. The same goes for

[1] http://maps.google.com/
[2] http://dailyburn.com/
[3] http://www.eazycheezy.net/2010/03/7-free-android-apps-to-get-in-shape-track-your-fitness.html
[4] http://www.firstresearch.com/Industry-Research/Fitness-Equipment.html
[5] http://nl.wikipedia.org/wiki/RJ-45

Life Fitness: while their home products do not support CSAFE, their cardio equipment meant for professional use in gyms does. Precor also only supports CSAFE for its Cardio machines.

The biggest player in the market, ICON Health & Fitness however does not support CSAFE. It does have something called iFit Live that allows you to connect from the internet to their fitness equipment, but that is a closed system: it only allows access to Google maps and Facebook, not to anyone who wishes to connect to it. Nautilus however, does not seem to have any sort of feature that allows you to connect to its data from the outside without the help of specialized equipment such as heart rate monitors.

The following table summarizes this:

Manufacturer	Standards supported
ICON Health and Fitness	iFit Live
Cybex International	CSAFE (All Cardio Equipment)
Life Fitness	CSAFE (Only for their professional equipment)
Precor	CSAFE (All Cardio Equipment)
Nautilus	No uniform standard found.

When you look beyond the “big five”, there seem to be less companies that support the standard of CSAFE. We collected a sample of 32 fitness manufacturers: Absolo, Bodycraft, Bodyguard, Bowflex, First Degree Fitness, Hoggan Sprint Cardio, Landice Treadmills, Multisports, Nautilus, Noramco fitness, Quantum Fitness, Schwinn, Spirit Fitness Store, Stamina Products, ST Fitness, Tunturi, Motus Usa, True, Freemotion fitness, Life Fitness, Reebok, Cybex, Precor, Star Trac, Woodway, Vision, Stair Master, Techno Gym, Workoutwarehouse, Weslo, Healthrider and Proform. Out of them, only twelve supported CSAFE, with just 10 supporting CSAFE for a majority of their products.

We could not find any other standard that was usable for cardio exercise equipment such as treadmills, cross trainers and home trainers. The ones that didn’t support CSAFE just did not advertise a feature that could allow communication to the outside. In terms of wireless technology however, there are a lot of different standards: there is ANT+, Garmin Connect and Polar also has its own standards of communication. The next sections will show a bit more about these three technologies and CSAFE.

2.1.2 CSAFE

CSAFE is a communication protocol for fitness equipment, standing for “Communications Specification for Fitness Equipment”. [6] It defines both the physical wiring scheme to which the equipment needs to adhere and the format of the data frames. It’s licensed royalty-free for any person, company or organization wishing to use it, which means that it’s free for anyone to use under normal circumstances, as long as you don’t try to sell products that centre around it.

CSAFE’s architecture is based on the master/slave principle: you have one master (pc, website, server, etc) who communicates with one or more simple slaves (fitness equipment)[7]. The slaves only speak when they respond to requests from the master, with perhaps one or two well defined exceptions. Because of this, the slaves are kept as simple as possible for the protocol.

[6] http://en.wikipedia.org/wiki/CSAFE
[7] http://www.fitlinxx.com/CSAFE/Framework.htm

The amount of fitness equipment that supports CSAFE is wide, but the majority of the fitness equipment that gets released today does not guarantee support to its functions. Even products of the same company may differ here: some adhere to the CSAFE standards, while others either do not, or just fail to specify.

Another big disadvantage of CSAFE is the lack of support. At the time when this thesis was written, it had been six years since the last update on the official CSAFE website. E-mails that get sent to the official support email address are not answered anymore, and on the internet, you can't find any tutorial, documentation or help on how to implement the protocol. The only implementations that are available are a number of commercial implementations that don't share their expertise. The most notable being Concept 2.

2.1.3 ANT+

ANT+[8] is a wireless technology. It's created with the principle that its wireless connection should be low on power. It works on top of the ANT proprietary wireless sensory network technology and it is mainly designed to collect, transfer and track sensor data from equipment such as heart rate monitors, pedometers, speed sensors and force meters. It can support many different network protocols. It's currently being used by 27 manufacturers, on 160 applications.[9]

2.1.3.1 Garmin Connect

One of the most known applications of the ANT+ protocol is Garmin Connect. Garmin is a manufacturer of mobile navigation software and heart rate monitors. They've developed the technology called Garmin Connect, in order to allow third parties to easily connect to their devices. You do this with a simple API, that communicates either with REST, SOAP or even XML-RPC.

It even seems possible to link ANT+ devices directly to fitness equipment, using Garmin Connect. The process is outlined with the following picture:[10]

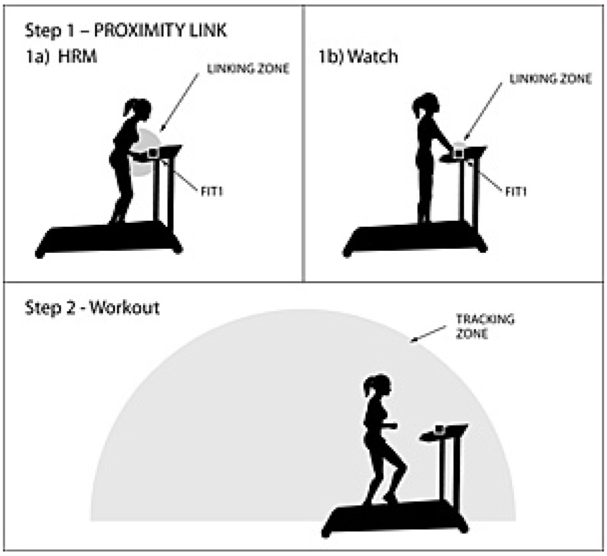


Figure 2: the steps in which you can connect a Garmin device to an excercise machine.

[8] http://www.thisisant.com/
[9] http://www.thisisant.com/modules/mod_product-directory.php?t=search
[10] http://www8.garmin.com/intosports/antplus.html

The grey part indicates the range of the equipment: you need to hold your device inside that zone in order to be able to connect. When you're in a busy gym, you don't want the equipment next to you to immediately catch the signal as well, so for initializing you'll need to hold your device very close to the control panel. As soon as everything is set up, you can move more freely, and your data will be caught in a much wider area.

It is difficult to determine exactly how many fitness manufacturers support this technology as well, however, since the ones that do fail to mention this on their websites and specification lists. It is confirmed that Nautilus, Concept2, Woodway and Star Trac[11] are able to connect to ANT+ and Garmin Connect and Octave Fitness plans to connect to it somewhere in the future, but we failed to find any clue about whether or not the rest of the above-mentioned big five (Cybex International, ICON Health & Fitness, Life Fitness, Precor and Nautilus) can also connect.

There are a lot of different ways to communicate to Garmin devices. You can directly read out the data from their devices, but they also provide a large amount of abstraction layers, most notably XML and Javascript.

With the use of the so-called "Training Center Database"[12], you can transport fitness data from Garmin devices through XML. Here, all the data you need for a device, such as heart rate, burned calories at different points in time is stored into hierarchical xml-files that can be easily read by an XML-parser, but also by a human reader. Garmin's Training Center is a software developed by Garmin which automatically connects to all of their products. As a third party developer, you then simply have to connect to that program (with XML, for example), and read its data. The format in which the data is sent is formally known as the Training Center XML (TCX)[13].

You can interpret this XML yourself with your own programs, but Garmin also created an entire Javascript library[14] to do this for you. The TCXActivityFactory-class[15] in particular can just automatically create an array out of a TCX-structured XML-file.

Then there's also the FIT-protocol. [16] It only operates on a select[17] and limited amount of devices, but it does offer a Java and C++ library that can be used for connecting to these Garmin devices and exchange information with them.

While the Fit and TCX standards are interesting for development, they're all standards that are sort-of only supported by Garmin Products. There also is the GPX standard, however. This one stands for "GPS Exchange Format", and it is another XML-based format, used to store data from GPS devices. The great thing about this standard is that all kinds of other applications from other companies also support it. An example of this is the mobile MyTracks Application, which tracks cycling routes, however also Trailrunner[18] supports it.

[11] http://www.tradingmarkets.com/.site/news/Stock%20News/2234980/
[12] http://developer.garmin.com/schemas/tcx/v2/
[13] http://en.wikipedia.org/wiki/Training_Center_XML_%28TCX%29
[14] http://developer.garmin.com/web/communicator-api/jsdoc/index.html
[15] http://developer.garmin.com/web/communicator-api/jsdoc/symbols/Garmin.TcxActivityFactory.html
[16] http://thisisant.com/pages/ant/fit-license
[17] http://developer.garmin.com/web-device/garmin-communicator-plugin/device-support-matrix/
[18] http://www.zdnet.com.au/downloads/mac/trailrunner_sw-10851588.htm

2.1.4 Polar Heart-rate Monitors

Polar is a company that manufactures heart rate monitors, speed sensors and various other kinds of wireless equipment to support online sports. They also created an application called polarpersonaltrainer.com, which allows you to connect to these devices. However, you can also use their Device Interface Toolkit[19].

Polar devices don't just store heart rate, but also a detailed report about exercises, your temperature, the weather, the results of the exercise (if you were running, then it reports how many kilometres you have run, with cycling it can list your average speed, et cetera). It has a lot of support for outdoor exercising, but it can be fit to indoor exercise equipment just as easily. It's also possible to link your data to a PC in real time, making it interesting for exergames.

Polar's heart rate monitors are supported by a variety of fitness equipment. The products of Star Trac and Life Fitness support them, along with a number of other companies.

2.1.5 iPhone

In terms of mobile connections there are not many standards. There is one company however that did create its own iPhone application: Bowflex[20]. It acts as a personal trainer, you can log your workouts, it automatically selects exercises for you and even has a facebook integration. The technology to do this is possible, however it does connect directly with Bowflex equipment, which makes it hard for third parties to connect to it.

2.2 Possibilities for VirtuaGym and Vitalence

2.2.1 Connection to the VirtuaGym Fitplan

This section contains a number of different possibilities that can be considered for VirtuaGym to connect to fitness equipment. We also developed a concept for a connection to Google Maps. That can be found at Appendix C of this thesis.

The easiest way to connect with fitness equipment would probably be to either use a Polar or Garmin Heart Rate Monitor, and use its SDK to automatically log whenever a user exercises. This information can then be easily uploaded to the internet and VirtuaGym's fitplan.

In order for this to be viable, VirtuaGym does need to provide an overview of exactly which equipment supports this functionality, in order to prevent confusion. There are a lot of different heart rate monitors out there, and the same goes for the exercise equipment. It's good for potential users if they know exactly what will work for them.

Garmin devices also support the ability to write data to them. We can use this to automatically plan in workouts, based on the goals that the user has set for himself. An idea is that the user can specify the time that he wants to spend on an exercise, so that some kind of program or algorithm can evaluate the best and most efficient workout based on this data. This can then be sent to the Garmin monitor, which in its turn can communicate with the appropriate fitness equipment, which selects the appropriate workout program.

[19] http://www.polar.fi/en/support/downloads/device_interface_toolkit
[20] http://appshopper.com/healthcare-fitness/iflex

Because not every fitness equipment supports the technologies of Garmin or Polar's heart rate monitors, there should also be some sort of default option: the ability to manually add to your Fitplan when the devices fail to make a proper connection to VirtuaGym.

Using CSAFE is a good option if you want to be able to support as many different fitness equipment with just one technology, however unlike Garmin and Polar's functions, there is no online activity about it whatsoever: there are no online forums, no tutorials, the only solid documentation that's there is the official one. Developing an application for CSAFE will probably require a lot of collaboration with Fitlinxx.

However, CSAFE's use does have a number of interesting applications compared to the Garmin and Polar heart rate monitors. Consider the apple's iPhone. People have already been creating iPhone apps

2.2.2 Real-Time data

While tricky to get it to work, it is possible to stream real-time data from the Garmin devices as well[21]. While this can be interesting for heart-rate monitors, it also allows for a lot of possibilities for products such as the Garmin Forerunner[22], which comes included with a food pod that can track your paces on treadmills. Suppose that you can connect a number of treadmills with each other, and develop some sort of race game, in which the users can control the speed of their characters with the speed that they're running at. Steering will perhaps be a bit of a problem on treadmills, but you can also think of simulating the 100 metres running at an athletic competition. This will be a great way to motivate people to exercise when they're with friends. In the absence of friends, you could also get some basic AI to create virtual opponents for the user, whose speed depends on the user's average fitness level.

Carrying this idea further, you can then proceed to link your cardio equipment to any other platform, for an example you can have people run around in Google Earth[23], or cycle around in its virtual environment. The same principles can be applied to other virtual environments as Second Life[24], Bing Maps[25], Terraexplorer[26], World Wind Central[27] and Activeworlds[28], or just some new created worlds with for example the Croquet Consortium SDK[29].

With each type of cardio equipment you can find something to simulate it. You can run around in these virtual worlds with treadmills, cross trainers can simulate walking or langlaufing[30] you can cycle with exercise bikes, you explore water bodies like rivers and lakes with an indoor rower, and a stair climber can simulate mountain hiking or climbing. The only issue would be steering. The solution can either be to force the user to walk down a straight path, or you could perhaps program some miscellaneous buttons on either the cardio equipment or heart rate monitors to steer a bit.

---

21 http://developer.garmin.com/forum/viewtopic.php?p=6284&sid=334764e7367d46aba33b51abc123c767
22 http://www.amazon.com/Garmin-Foot-Pod-Forerunner-305/dp/B000UO61HC
23 http://earth.google.com/intl/nl/
24 http://maps.secondlife.com/
25 http://www.bing.com/maps/
26 http://www.skylinesoft.com/SkylineGlobe/Corporate/Products/TerraExplorer.aspx
27 http://worldwindcentral.com/wiki/Main_page
28 http://www.activeworlds.com/
29 http://www.opencroquet.org/index.php/Main_Page
30 http://www.embeddedfitness.nl/bewegingsgames/games/Pages/Crosstrainer.aspx

3 Practices in Online Fitness and Lifestyle Communities

Before we talk about the technical matters of an API, we first want to show a small case study.

3.1 Dailyburn

Dailyburn.com[31] is an online fitness community in which users can keep track of their own fitness results over time. It also allows users to keep track of their food intake, allowing them to keep a precise log of all of the calories they consume. It has many similarities to VirtuaGym. At this point it already has its own API, that despite its Beta stage already has a selection of applications that communicate with it. This section will describe the Dailyburn API.

3.1.1 API Description and Design Decisions

Since Dailyburn bears a lot of similarities to VirtuaGym, you can see a lot of parallels in their API, and the API that will be developed for VirtuaGym. This section will list some of the most notable design decisions they made that will aid in making VirtuaGym's API better.

3.1.1.1 Authentication

When you want to create an application for the Dailyburn API, you first need to send an email to one of the administrators, containing information as the name of your application, its description and URL. A reply to this email is then sent back, containing a "consumer key" and "consumer secret key". User authorization is done by either Open Authorization or Basic HTTP Authentication.

While Basic HTTP Authentication is supported, the documentation hints that somewhere in the future it will be deprecated. In order to successfully use it, you need to also provide the user's password credentials, as well as the consumer key that was included with the registration confirm email. How this key should be sent is not detailed within the documentation, other than that it should be called "key".

The documentation doesn't bother to explain what Open Authorization is, and instead just points to a bunch of tutorials and assumes the user to find his own library to work with it. The Oauth authentication scheme is very standard, you'll have to specify your application URL and call-back URL when you send the registration email to the administrator (which likely are not even used in the server's database, other than for simple administration), and the url scheme consists out of the standard `request_token`, `authorize` and `access_token` pages. For encryption, they only support one protocol, alongside just plain text.

3.1.1.2 Response format

The Dailyburn API is modelled after REST, however not strictly. The documentation lists a bunch of URLs that give you access to data like user information, workouts, foods and body metrics. You can issue GET-requests in order to retrieve them. PUT and DELETE requests are a bit inconsistent, however: with food log entries for example, you simply send PUT or DELETE requests to their respective urls. If you want to put or delete an entry of your favourite foods however, you need to send a POST request to a sub-domain of the favourites URL that contains a function that can do that.

Responses are sent in default in XML. When you want to receive the values in the JSON[32] format (which is more efficient), you need to append the string "`.json`" to the end of the url of the page

[31] http://dailyburn.com/
[32] http://www.json.org/

that you are accessing. Error responses are sent along with an HTTP status code of either 400 or 500, along with a simple message.

Most values (except for strings) are accompanied by a type (for example, boolean, decimal or integer). For values such as weight and length, which can be expressed in the metric and the English system, an extra field is sent along to specify under what standard that value is sent along.

3.1.1.3 What they did right

- The usage of Basic HTTP Authentication is a good choice for getting people familiar with the API, and on top of that they do especially note that it's only going to be there temporarily, forcing developers to go after the safer but more complicated Open Authorization.
- It forces all developers to register, so that they can keep track of everyone who uses their API.
- Responses can be read and interpreted without documentation.
- All API calls are sent through a secure HTTPS connection.

3.1.1.4 What they did wrong

- It's a very inconsistent API. At some entries they use direct HTTP GET, POST, PUT and DELETE requests, at others they use just POST requests to a bunch of functions. To make matters even more confusing, in order to create a new food log entry, you need to use POST, while to update one you need to send a PUT request. That's not how http was designed.

3.1.2 Results

In this section we will show a number of the applications that have been developed with the Dailyburn API since its launch.

3.1.2.1 The dailyburndroid

The Dailyburn droid is an android application that works for the Daily burn. It's mostly dedicated to bringing the functionality of the Daily Burn to the android mobile platform. At the moment it's still in its development stage, but these are some of the features it currently supports[33]:

- You can search for foods in the Dailyburn database, as well as view a user's food log entry (a food log is a list of foods that a user has consumed). You can search for foods by barcode scan, add them to your favourite foods.
- You can view your user data.

The developer's weblog[34] showed that he did run into a number of problems when he tried to develop his application:

- The documentation for the API was incomplete or wrong at times, which at times discouraged him to keep developing.
- The creators of the API also kept updating the API without updating the documentation, which lead to some confusion.

These examples show how important documentation can be, and it's vital to have it to be complete and up to date at all times. The developer of the android application solved his issues in the end by using the developers' forums, where he could get direct feedback from the Dailyburn staff.

---

[33] http://code.google.com/p/dailyburndroid/
[34] http://dailyburndroid.wordpress.com/2010/03/02/the-importance-of-communication/

3.1.2.2 WP-Dailyburn

WP-Dailyburn[35] is a Wordpress extension that can be installed on any Wordpress weblog (or blog). Its features are very basic:

- It lists your current weight, followed by your goal weight.
- It lists the calories that you've eaten (assuming for the current day) and burned.
- It shows a bar that shows your exercise status and nutrition status, with the purpose of giving an indication of your progress in both areas.

At the time when this thesis was written (April 27th, 2010 to be exact), WP-Dailyburn had already proven to be successful: it has only been released for one month, but Google already lists thirteen weblogs that have the plug-in installed.

3.1.2.3 Ideas

Aside from the above-mentioned examples, people are also toying around with ideas for the use of the Dailyburn API. Despite the fact that we couldn't find anything concrete about them, the following list contains some of the ideas that have been suggested[36][37]:

- Dailyburn defines workouts as a set of exercise sets. One ideas is to create an application that gives a one-page summary of the last exercise set you performed.
- A connection with Google Earth, which would allow connection with various kinds of running applications.
- A weight manager for the Android.
- A C# library for the API.

[35] http://solongfatass.com/wp-dailyburn/
[36] http://dailyburn.com/forums/developers_forum/topics/api_documentation
[37] http://dailyburn.com/forums/developers_forum/topics/early_devs_-_what_are_you_building

4 Architecture

There are a number of possible frameworks that can be chosen to implement an open communication standard in. The two that are the most widely used and have the most available documentation at the moment are REST (Representational State Transfer) and WS-* (Web Service Specifications). This section outlines these two frameworks and compares them to each other.

4.1 Possible Frameworks

4.1.1 WS-*

WS-* is a general term, describing a collective description of Web Services. Web Services are just APIs that can be accessed over a network. There is not *one* standard for implementing a Web Service, but instead there are many different specifications that sometimes even compete with each other. In essence however, it's an interoperability standard meant for middleware.

4.1.1.1 SOAP

In WS-*, messages are generally sent though the XML-based SOAP-protocol. You have a SOAP-client and a SOAP server. The server possesses a WSDL (Web Service Definition Language)-file which specifies the methods, functions and data that a client can access, displayed in an XML-format. The client can look at the specification for this server, request the data it wants to access with the right function, and get an XML-reply back. This message protocol has a number of advantages and disadvantages, which will be outlined by the following two sections.

4.1.1.2 Advantages of SOAP and WS-*

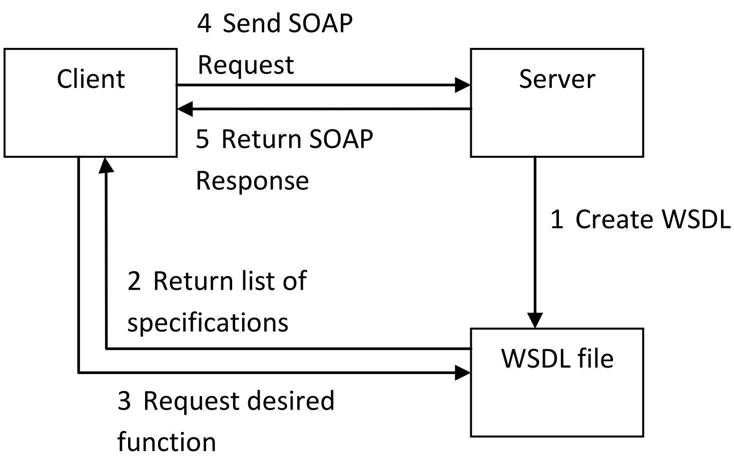
- SOAP Provides a layer of abstraction over its underlying architecture. The architecture is layered, from the ground up in factored and distinct specifications.
- It's been in use for many years now, which has lead to a lot of standards that you can adhere to.
- Development tools can hide the complexity of the protocol.
- It's flexible: the same interface can easily be changed to adhere to different protocols as business and technological requirements change.
- Its independent of its underlying transport mechanism.
- SOAP Messages protect themselves: you do not need to run them over a secure connection.

4.1.1.3 Disadvantages of SOAP and WS-*

- Its complicated syntax will scare off many who are new to the protocol.
- You could easily get lost within the novels of documentation that have been created by now.
- Even though there are standards, they were formed by combining competing specs, so there is a lot of pointless feature overlap.
- Resources are not consistently named so finding them could prove to be a challenge.
- Very complicated security standards that are defined in multiple places (there are many standards defined at w3: there's WS-Security, XML Encryption and XML Signature without a proper point of reference to starting developers).
- It's relatively slow compared to its alternatives.

4.1.1.4 SOAP Protocol Structure

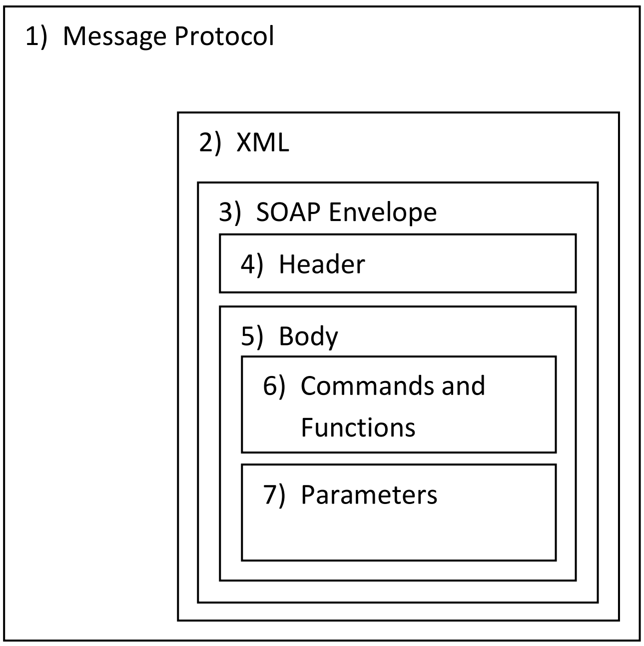
SOAP is a message protocol. The steps that it needs to take are outlined in the following diagram:



Here is what they mean:

1) The server creates a WSDL-file. This file contains all of the functions that can be requested to it, the parameters that they accept, the types (int, string, decimal) that these types need to be in, and its return values. It's written in such a way that it can be understood with the right software.
2) When a client wants to connect to the server, it uses the UDDI protocol in order to access the WSDL-file at the server.
3) The server then returns the contents of the WSDL-file, which contains all of the data that the client needs in order to connect. The client then looks up this WSDL-file, and picks out the parts that it needs.
4) The client now sets up a SOAP envelope in XML, containing the request that it wants to send, and sends this to the Server.
5) The server interprets the SOAP message, it executes the request from the client and returns the contents in another SOAP message to the client.

SOAP Messages are sent in SOAP Envelopes. A typical SOAP message has the following structure:



1) The message protocol is usually HTTP, but with SOAP it can be anything: SMTP, TCP, JMS, BEEP, or MQ.
2) All SOAP messages are sent in the XML format.
3) All messages are wrapped in SOAP Envelopes. They require a specific envelope template that needs to be accessed with every message.
4) The header is used to define a variety of meta-information, for example to define the different data types in the rest of the message.
5) The body contains all of the data that needs to be sent.
6) In the case of a SOAP request, a client can request a number of functions to be executed on the server. These are listed here.
7) In the case of a SOAP request, the parameters for the functions that are called are listed here. In the case of a SOAP response, section 6 is merely empty and this section contain the results of the functions and commands that were executed.

4.1.1.5 WS-* Extensions

Standalone SOAP is merely a protocol. It was originally meant as a "Simple Object Access Protocol", but its complexity mostly can be attributed to the way it evolved over the years. People tried to add more and more things to the WS-* specifications. Below you can see a table with the most important extensions[38]:

WS-Coordination	Manages the context across web services.
WS-Transaction	Enables Distributed transactions.
WS Reliable Messaging	Guarantees delivery of SOAP messages.
WS-Policy	Determines the policy and constraints of Web services.
WS-Security	Handles the security for Web Services.

4.1.2 REST

In contrast to WS-*, which was an interoperability standard between middleware, REST is an architectural style for the web. If a client wants to interact with a server, it can do so by issuing so-called CRUD-requests: Create, Read, Update and Delete, also known as GET, POST, PUT and DELETE. GET-requests can be sent through a simple url, while responses can be displayed in markup languages as XML, JSON, Atom or YAML. It was envisioned by Roy Fielding, co-author of the HTTP-specification. He summarizes REST as follows:

REST enables intermediate processing by constraining messages to be self-descriptive: interaction is stateless between requests, standard methods and media types are used to indicate semantics and exchange information, and responses explicitly indicate cacheability[39].

4.1.2.1 Architecture Principles

Whenever an application adheres to the principles of REST, it is considered RESTful. These principles are the following:

[38] For a complete overview of the currently existing standards: http://www.innoq.com/resources/ws-standards-poster/
[39] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

- *Resource Identification:* Name everything you want to talk about. Think of “product” in an online shop, or “weight” in an online fitness community.
- *Uniform Interface*: The same small set of operations applies to everything. Compare it to verbs and nouns: ideally you want a small set of verbs to appeal to a large set of nouns. If a noun needs a new verb, it can be created.
- *Self-Describing Messages:* Resources are identified because of the Resource Identification, the Uniform Interface allows us to access them. This is done through resource representations, such as XML or JSON.
- *Hypermedia Driving Application State:* Resources and States can be used by navigating through links. RESTful applications navigate instead of call: representations of resources contain information about possible traversals, and the application navigates to its next resource depending on these link semantics.
- *Stateless Interactions:* The idea is to avoid long-lasting transactions in applications. This means that the server does not keep track of the state of the clients. The client in its turn does not keep track of the state of the resources: that’s the job of the server. The client just accesses these resources.

4.1.2.2 Advantages

Based on the following specifications, a number of advantages can be attributed to the REST Architecture:

- The statelessness and the uniform interface make it very scalable (as demonstrated by the http protocol, which is the most famous example of a RESTful architecture).
- Very easy to use and connect to, developers can already get responses with a few simple lines of code.
- There are many different ways to output data, to share and restrict data.
- The URL-based names are a consistent method for naming resources.
- Security is easy to implement with SSL over HTTPS.
- The server is light-weight, so relatively fast.
- When implemented right, it has very light coupling between the client and the server: the server can change a lot of its structure without breaking clients.

4.1.2.3 Disadvantages

- There’s a lack of standards, so every developer makes his own choices and syntax.
- It doesn’t support asynchronous communication.
- As soon as you start dealing with very complex servers with complex functionalities, the interface will get very hard to understand, though.
- The link-based discovery principles will make it a bit difficult for starting clients to navigate at first.

4.1.2.4 Examples

Suppose you have the following URL in your API:

```
http://www.thisisaninterestingurl.com/api/users/43
```

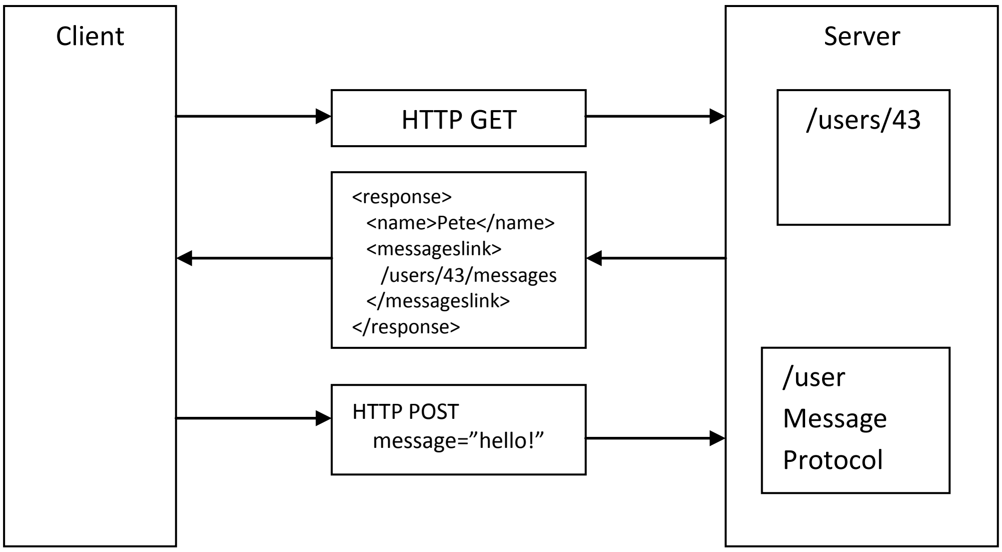
A REST protocol would communicate with this URL by sending it GET, POST, PUT and DELETE requests over the http protocol. In this way, it can view and edit a user’s content, and you can insert or delete this user. This of course requires the necessary restrictions on the server to prevent unauthorized parties from accessing and changing your data.

Suppose that our example site in question also has the option to view the messages posted by a particular user. Suppose that the server allocates the following url format, in order to list a particular user's messages:

```
http://www.thisisaninterestingurl.com/api/users/43/messages
```

Now, one way to publish this would be to just publish this direct link for all developers to access. Another option with REST is instead to include this link when a client accesses the user's user info. By allowing clients to browse through their content this way, a server can always change its internal link structure, as long as it keeps the name of this link and the initial link that is used to access the server (so in this case the user page) the same.

The following diagram illustrates this as an example. Here, we want to post a message on user 43's account. First we look up user 43's details, which includes a link to the place where we can post this message, after which we send this message using POST:



The result of this operation is that a message is now posted at the user's account. Do note that this example does not take into account security or authorization. The sections below will get into more detail about that.

4.1.2.5 Richardson Maturity Model

Adhering to true REST is difficult. Leonard Richardson[40] outlined a maturity model consisting out of four levels of increasingly strict adherence to REST.

Level 0 stands for simple HTTP requests back and forth. The server publishes a number of functions that can be used while the client sends calls to these functions to the server and waits for a response. There are no other constraints beyond that, and the next section will go into more detail about that.

[40] http://martinfowler.com/articles/richardsonMaturityModel.html
http://www.crummy.com/writing/speaking/2008-QCon/act3.html

Level 1 adds the concept of resources: instead of just sending requests to the server, the client sends requests to *resources* at the server. For example, instead of telling a server to fetch user data, it tells a resource that represents a user to fetch its own data. If you want to make an appointment at a certain date, you access the resource that represents that particular time and request an appointment.

Level 2 adds the concept of verbs. In the above two levels you could request with any sort of function, for example `get_userdata()` or `make_appointment()`, in order to acces nouns (resources). Level 2 however adds a constraint to these verbs: the only verbs that are allowed are GET, POST, PUT and DELETE. In this case we have a framework in which a lot of nouns can be accessed by few verbs, which adds to the scalability of the system.

Level 3 adds discoverability; on this level you also include links that reference other relevant resources. This allows a client to browse through resources from its starting point and adds a discoverability that allowed the HTTP protocol to grow into the way that it has now.

Strictly speaking, any protocol that is not on level three isn't technically REST, but rather an RPC framework. However, incorporating even some of REST's principles can contribute greatly to scalability, usability and simplicity of an API.

4.1.3 RPC

RPC stands for Remote Procedure Calls. While it can have many different contexts, an API in RPC-style means that you let third parties access your frameworks through functions. Rest and SOAP can be seen as subsets of RPC. Where in REST, you can access each resource only though four functions, you can access each resource through a wide variety of functions. Take the following URL:

```
http://www.thisisaninterestingurl.com/api/users/43
```

REST only has the functions GET, POST, PUT and DELETE. RPC however can have a wide variety of functions that you can apply, for example:

```
getname()
getlocation()
adduser()
changeaddress()
listusers()
```

4.1.3.1 Advantages

- The simplest way to create an API: there are no constraints for the API developers and it's the fastest way to create one.
- It's lightweight, so it's fast.
- With small systems that only have a limited amount of functionality it's easy and simple to use, support and document.

4.1.3.2 Disadvantages

- With very large systems or APIs, potential clients will have a lot of difficulty finding the functions they need, and whether or not they actually exist, while some may not even realize that some functionalities exist due to the relatively unstructured naming conventions.
- It's very hard to change something or the structure.
- It has a lack of standards. Even more than is the case with REST.

4.1.4 Conclusion

The difference between REST and SOAP is that REST is an architectural style, while SOAP is more of a message protocol for middleware. However, the REST approach is far simpler than having to use the SOAP approach. It will make it much easier for third party developers to connect to your application, without having to learn all of the SOAP syntax, especially when the programs are getting bigger. REST messages are also much shorter than SOAP-messages, which all require an envelope. If you were to package REST-requests in an XML-file, the result would look much more elegant, and you can even choose to process data through a JSON format, or the ATOM publishing protocol.

In contrast, SOAP possesses a collection of tools that will make developing easier. WSDL-files are often automatically generated, and a SOAP standard has been outlined in a lot of details by the World Wide Web Consortium. In terms of security, it also has a very broadly defined standard in the form of WS-Security.

In terms of simplicity, RPC is even easier to implement, but because of its lack of constraints it does have its disadvantages. Because you can identify any kind of function for each data element, it's going to be much harder for third parties to figure out what kind of functions they can implement when and where, without having to resort to the documentation. It's also much less flexible than REST: suppose that you want to change the architecture in the future, it'll be much easier to simply add new REST-objects, rather than adding a collection of new functions to support the new functionality.

In the context of fitness and lifestyle applications, you want them to be accessible. Even though SOAP is a widely supported standard, it will be hard to learn for developers who have never used it, and unnecessarily complex. SOAP is a technology that requires more implementation effort on the client's side, while it provides a number of convenient development tools for the server side. REST on the other hand requires more implementation effort on the server side due to the lack of concrete standards, but it is very easy for clients to connect to it. Because of this, we will favor the simplicity and accessibility of the REST-framework over the complexity of the SOAP framework.

Adhering to true REST, as described by the Richardson Maturity Model does have its disadvantages, however. The concept of having a single starting point, and forcing your clients to explore the rest of your API by following links is very useful in volatile systems, and while you can easily add new functionality, or restructure your functionality, it still does not offer flexibility if you want to change your message protocol or if you want to deprecate a function. Having users search through multiple layers of links also adds an extra unwanted layer of complexity that limits the usability for application developers. In a lot of cases (especially relatively small and simple APIs), it can be wiser to instead just opt to implement REST to only Level 2 of the Richardson Maturity Model.

The next sections will explore the challenges that lie in building a REST API so that it can be simple, how to handle the identification properly, make it an appropriate standard for fitness and lifestyle applications, prevent people from misusing it and making sure that it is scalable for in the future.

4.2 User Friendly Designs, Interfaces and Protocols

When designing usable and user-friendly systems, a good practice is often to keep things as simple as possible. A simple system has a number of advantages: it's easy to learn, it's easy for developers to use, and it's also easy to extend. Especially in creating an open communication standard, you want your interface to be as easy to get into as possible, however without sacrificing functionality in the process. The list below is created by Joshua Bloch[41], and contains a number of guidelines that must be taken into account when designing this standard:

- *It should be easy to learn.*
- *It should be easy to use, even without documentation.* This makes sure that developers won't be forced to look at the documentation, for every request that they want to perform.
- *It should be hard to misuse.* It should be designed so that even though the user will try to use the system in ways that weren't intended, it should still work.
- *It should be easy to read and maintain code that uses it.* When various parties want to connect to the API you're developing, they should be able to read their own code easily, even if it's been half a year since they wrote it.
- *It should be sufficiently powerful to satisfy the requirements.* So it shouldn't be the case that you sacrifice functionality for simplicity.
- *It should be easy to extend.* At which I'll delve into more detail in section 4.6..
- *It should be appropriate to the audience.* Different people have different ideas about simplicity, and the system should be developed to be as simple as possible for all of the people who will be connecting to it in the future.

This is a great advantage of a REST-framework: as a developer, you don't have a number of functions that you need to check every time. Instead, you have a number of pages or resources, which all can be accessed through just GET, POST, PUT and DELETE requests. It's easy to learn, if you know how to access one resource, you'll also know how to access the other resources, minimizing the need to look at the documentation. These four operations are simply sent through HTTP-requests, which are clear, hard to misuse and easy to read. With very limited effort, you can easily extend the framework with extra resources, and it's a framework that's often used, and generally seen by developers as a simple way to connect to a framework.

4.2.1 Creating Friendly URLs

One way for web APIs for online platforms to increase their readability is by having more easily readable URLs. This can be done on Apache web-servers with the mod_rewrite functionality, and on non-apache servers it can be done with a simple PHP script. Take the following URL:

```
http://www.thisissomewebsite.com/users.php?id=15
```

This syntax is a bit hard to remember, especially if you want to type it down somewhere. Mod_rewrite can be used to turn this into a much friendlier url. With regular expressions you can turn it into something more readable:

```
http://www.thisissomewebsite.com/users/15
```

Of course, this can be made to look even nicer. Suppose that instead of checking a user, based on his identity number, you instead ask for his username:

```
http://www.thisissomewebsite.com/users/pete
```

[41] http://lcsd05.cs.tamu.edu/slides/keynote.pdf

This will lead to an URL that's much easier to remember for end users, and it also is much friendlier for search engines as Google to find it. In the context of a Web API, this will allow users to pick up the pages with their own information very easily, making it more accessible.

4.3 Possible Identification Protocols

In order to connect to a system, a client must be able to log into it. This section outlines the two most common approaches for this: Basic HTTP Authentication, which sends the user's username and password along with the URL, or Open Authorization, which logs you in at the server side, instead of having to send your password across the world in which it may be intercepted.

4.3.1 Basic HTTP Auth

Basic HTTP Authentication allows you to provide credentials over an HTTP protocol. Suppose you have the following site:

```
http://www.website.com/login.php
```

With basic http authentication, you can put a username (say, `johndoe`) and a password (say, `applepie`) in the URL in the following way:

```
http://johndoe:applepie@www.website.com/login.php
```

The string `johndoe:applepie` is then encrypted, and sent to the server, at which it's decoded. In languages as PHP, this process is often done automatically with libraries as cURL.

4.3.1.1 Advantages

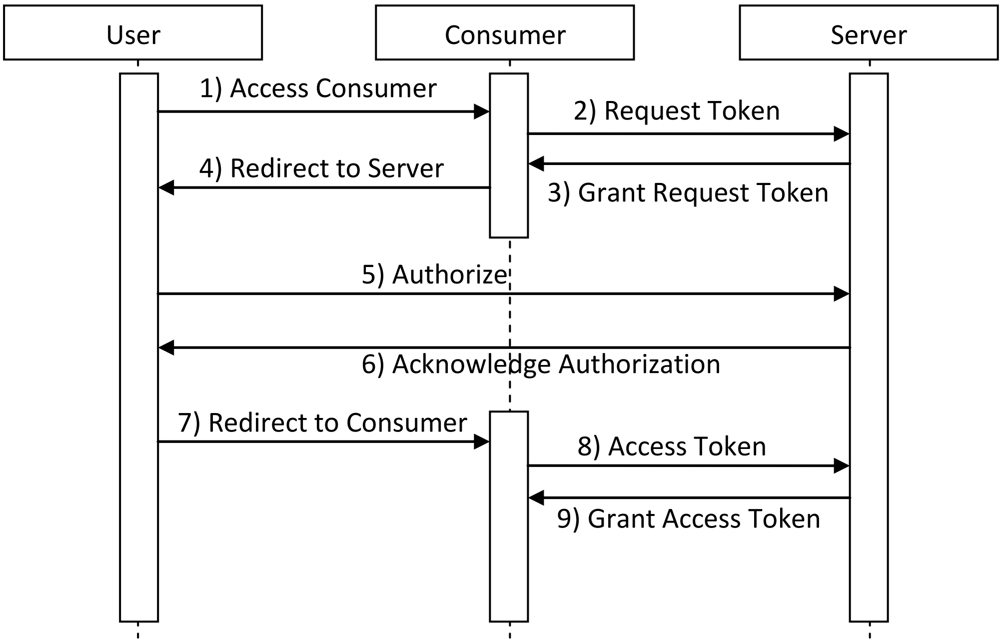
- It's very easy to implement: all you need is some library in some language (whether its' a scripting or programming language) that can send HTTP requests, and you can connect.

4.3.1.2 Disadvantages

- This method is very unsafe. You need to send it over an HTTPS connection. Attempting to do so over regular HTTP is just suicide.
- Internet Explorer does not support this. When you try to access the above url, it will simply generate an error. It does support sending the username and password along if they're not in the URL, though (using the POST method of HTTP).

4.3.2 OAuth

An alternative way to log in a user is Open Authorization, or Oauth for short. While it is harder to use in comparison to Basic HTTP Authorization, it's a much more secure method to login a user. Instead of sending your password throughout a network in which it may be intercepted, Oauth instead offers to redirect a user to the server, in which he can log in. The following diagram shows the procedure.



The terminology used in the diagram is defined as follows:

- `user` stands for the human user who tries to access the server.
- `consumer` stands for the application that the user uses in order to access the server. This can be a website, a desktop application or even a mobile application.
- `server` is the side that provides the content for the user.

This unfortunately is not the clearest terminology because user and consumer can often be confused with each other. In essence: the human user uses the consumer application in order to access the server. Here's what the individual steps mean:

1. A user accesses a particular consumer in order to interact with the server.
2. The consumer then requests a request token from the server. This token, once authorized, will be used in order to validate access to the user's data on the server.
3. The server grants a request token for that particular user, and sends this to the consumer.
4. The user is then redirected to an authorize page at the server.
5. At the authorization page at the server, the user logs in to the server with his account there, and determines whether or not he will allow the consumer access to his data.
6. With this, the server now is able to grant the consumer access to his files; he acknowledges this to the user. If not, then the user is redirected to some sort of error page and access will not be granted.
7. The user then is directed back to the consumer.
8. The consumer now requests an access token for this user.
9. The server confirms that the user has given approval, turns the consumer's request token into an access token and sends it back to the consumer. After this the consumer is able to access the user's data on the server.

Figure 3: the Oauth Logo

4.3.2.1 Advantages

- User passwords are sensitive information. Oauth allows you to prevent sending this information across a network that could potentially be intercepted, or misused at third sites at which the user fills it in.

4.3.2.2 Disadvantages

- Oauth is a lot harder to understand for both developers at the server and the client. A lot of steps have to be executed correctly before you can get something that actually works.
- Whenever malicious users get hold of your tokens, they can still access your data without any problems. Changing your password won't even help to stop it, instead you'll need to explicitly disable this token in order to stop them from accessing your data.
- This protocol is not friendly for mobile applications. These kinds of applications often can't redirect users to a webpage, and redirecting back is also impossible.

4.3.2.3 Open Authorization 2.0

Currently, an updated version of the Open Authorization Protocol is also in development, in order to fix the disadvantages mentioned above. The major differences and improvements when compared to the first version of Open Authorization are as follows[42]:

- While the first version of Oauth could be sent across a regular unsafe network, Oauth 2.0 instead will be sent across SSL, eliminating the need for complex signature schemes to keep them safe.
- There is no request token. Instead there is a single access token that can be granted by immediately redirecting the user to the server so that he can give permission.

Oauth 2.0 is a very promising technology, especially considering that Facebook has already adapted it, despite refusing to adopt the first version of Oauth[43]. Unfortunately, at the time when this thesis was written, the formal specifications of Oauth 2.0 were still in a draft phase.

4.4 Security Considerations

This section discusses the security aspects that play in developing an API. An API should be as open as possible, but malicious users should not be able to gain access to parts for which they have no permissions.

[42] http://developers.facebook.com/docs/authentication/
[43] http://iiw.idcommons.net/Why_Facebook_doesn%27t_implement_OAuth_today

4.4.1 Brute-force Attacks

One thing that you'd like to prevent is for bots or malicious users to keep trying out different passwords for a certain user in a dictionary attack. Such attacks can prove to be exceptionally vulnerable for users who have passwords that appear in the dictionary.[44]

One way to prevent these attacks is to limit the amount of accesses you can do at a time. If you detect that someone is failing to login after 30 attempts, it's very likely for this person to be some sort of hacker, so he can be blocked. Spam-bots can also be given limited access this way, whenever they attempt to connect to your site at a rate that isn't possible for a regular human.

4.4.2 Authorization

Authorization with Basic HTTP Authentication and Open Authorization has been discussed above already, but there are a number of security issues that need to be taken into account. First and foremost: eavesdropping. With Basic HTTP Authentication, you simply need to intercept the HTTP messages that get sent over a network and you'll have access to the user's password. With Open Authorization, you can also just grab an unprotected message and use its key to send your own data along.

There are a number of ways to counter this, first and foremost: the HTTPS protocol, standing for HTTP Secure[45]. It's a web protocol that's meant to provide reasonable security over an insecure network. It relies on the concept of certificates: whenever you access a page through HTTPS, you first verify with a certificate authority whether or not the page you're trying to access has a valid certificate. If not, then a malicious party is probably pretending to be the site you want to access. HTTPS messages are sent using the SSL or TLS[46] protocol, which relies on a special key-based handshake protocol in order to provide security.

Another issue that needs to be taken into account is the authorization of not the users, but the clients that access the server. When left alone, malicious users can create all kinds of applications that needlessly spam the API, or trick users into making them sign up (which is especially dangerous for Basic HTTP Authentication). One way to identify each client is to give each of them a unique key, forcing them to provide this key at every call. This can be a simple random key, but a combination between a private and public key works as well. You could for example require a digital signature to be sent with every request, which can only be encrypted with a public key, and decrypted with the corresponding public key.

4.4.2.1 Injections

This section mostly refers to preventing SQL-injections, but any content that is sent to the server needs to be checked for abuse, in order to prevent people from inserting either rubbish or attacks into your database. A general guide-line should be that you should always escape special characters that you receive from a user, and always check whether or not the values and data you receive complies with the format that the server accepts.

---

44 http://www.wired.com/threatlevel/2009/01/professed-twitt/
45 http://en.wikipedia.org/wiki/HTTP_Secure
46 http://en.wikipedia.org/wiki/Transport_Layer_Security

4.4.2.2 SSL

SSL may be a good solution to prevent man in the middle attack and eavesdropping, but alone it does have its vulnerabilities. [47] Consider Intrusion Detection Systems or Intrusion Prevention Systems, which detect malicious access to your system by checking all of the incoming messages. If those messages are encrypted with SSL, they can't be checked. So when a malicious user has gained access to your connection, he can't be detected.

A solution for this problem is applying transparent SSL Proxies[48]. The principle of these proxies is that they're located at the server and they possess the same SSL certificates. Whenever they intercept a message, they decrypt it, and send it to some sort of Intrusion Prevention System for verification. If no threats are found it passes on the messages to the server.

4.5 Adherence to Standards

Like we mentioned at one of the above sections: one of the biggest problems with REST is a lack of widely supported standards. At this point, client developers that want to connect to an application programming interface that supports REST or RPC will have to learn a new interface for every different API.

The biggest standard that is linked with REST is the HTTP. This is why it pays off to model your API after the HTTP-protocol, for example by supporting the HTTP status and error codes, rather than going for your own numbers to return whenever something goes wrong. People are already familiar with those standards, so they won't have to learn these new kinds of standards.

The same goes for the authentication schemes. Basic HTTP authentication is built in in every major browser and is very widely known. Open Authorization is also widely used by famous APIs, such as the Twitter, Google and Yahoo API. By adhering to these standards, you will make it easier for your users to get to know your code, although there still are too many differences between all of the Web APIs out there to form a good standard for every single aspect, such as a return format, the language in which returned messages are specified, and the protocol with which these messages are sent.

4.6 Scalability

The statelessness of a REST server is a big advantage when it comes to scalability: you don't have to care about which clients are connected to you, which will allow you to process many different servers without memory issues. With this, the server hardware will only have to take care of being able to process every incoming request.

Traffic management is an important concept for this: how do you successfully limit the accesses to your server?[49] These are some of the considerations that you need to take into account for that:

- What kinds of rate limits do you need? Do you need to limit by developer, application or customer organization? Do you need buffers?
- How do you monitor and respond to traffic management? How will you know when someone takes up too much of the server's bandwidth? And how can you respond to this?

[47] http://esj.com/Articles/2010/04/06/SSL-Risks.aspx?Page=1
[48] http://esj.com/articles/2010/04/13/ssl-risks-2.aspx
[49] http://blog.sonoasystems.com/detail/dont_have_a_meltdown_api_traffic_management/

- What about caching? How will it be implemented, and how will it prove to be the most efficient at handling consumer data?

At the server side meanwhile, adhering to full REST (with a common starting point, in which clients navigate through links) will enable the server structure to grow almost infinitely, as shown by the HTTP Protocol. If your system is based on a lower level of REST, the following considerations need to be made:

- How will you version your API? How will you facilitate deprecated functions, and how will you make sure that clients who use older versions of your API won't break?
- What will happen when the functionality of your API doubles or even quadruples from your original design. Will it still be easy to use, despite being so much larger that you at first thought?
- How easy will it be to add extra functionality? When designing an API you need to assume that it will grow.

4.7 User Management

Setting up an API is one thing, but getting people to use it also requires some thoughts. We already discussed this issue a bit at section 4.2: making it easy to use and get into is a very important part in this.

Apart from that however, the following issues also could be used in order to make an API more user-friendly for developers[50]:

- Should you allow developers to manage their own applications at the server side? Take for example the ability to accept and deny certain users, an overview of who has accessed their applications, etc? And how much freedom should you give them in this?
- How to you prevent developers from abusing your API? Do you have a way to prevent them access? Do developers need to be registered in order to make use of your API at all?
- What kind of information or tools do users need to have in order to sign up to your API?
- How much power do the administrators have over the applications and websites that the developers create for your API?
- How do you advertise your API? How far will you go to promote it without annoying users who have no intention to become a developer?

There are also a number of things that you can do in order to get developers interested in creating something for your API. In terms of advertising, the following methods can be used in order to collect an audience for your API:[51]

- Establish a developer community: a place for developers to gather, talk to each other and inspire each other to create new things. It's also a good centralized place for you to offer technical support for those who need it.

[50] http://blog.sonoasystems.com/detail/welcome_aboard_api_user_management_and_onboarding/

[51] http://blog.sonoasystems.com/detail/if_you_build_it_will_they_come_part_4_community_tools_and_audience/

- Plug into other developer communities. There are a lot of other developer communities out there with which you and the developers can exchange information. Take for example Microsoft's MSDN[52], Google's community[53] or Stack Overflow[54].
- Create a working example application. This will not just help users understand your API, it will also show off its capabilities and provide inspiration for other developers.
- Be visible on Google. Try to get high Google rankings on the search term "API + Your business".
- Communities often thrive due to one passionate core who keeps everything together. Make sure attract at least one person of that calibre, and make sure to provide plenty of feedback for the developers that did decide to get involved.

[52] http://msdn.microsoft.com/nl-nl/default.aspx
[53] http://code.google.com/intl/nl/
[54] http://stackoverflow.com/

5 API Design Decisions

This chapter will detail describe, explain and argue the decisions that were made in the design of the API. It's split up into three sections: the Developer Interface talks about the functions that developers of applications should have when they want to create something that communicates with the API, the API Syntax specifies the interaction protocol in order to communicate with the API, and the Admin Functions is a small section that describes the admin page for the API. You can find the Use Cases for these functions in Appendix B, subsection 2, and their scenarios can be found at Appendix A, subsection 2.

5.1 Developer Interface

The developer interface[55] is meant to guide developers across the API, and provide them with all of the information they need to know in order to get started and create applications that communicate with VirtuaGym, with the intention to make it as easily accessible as possible. It consists out of a number of pages, which will be outlined below.

5.1.1 Getting Started

This is the home page of the API: when new developers go to the developer interface, this page is the first thing they see. Because of this, there is little text and an eye catching lay-out, which will motivate potential developers to experiment a bit, as opposed to immediately throwing a heap of text at them with explanations , which might throw them off. Here they're guided through three simple steps in order to get started with their application:

- First they need to request an API key. For this they simply need to fill in a small form with information about what kind of application they're going to make: their email address and phone number, along with the title of the application they're going to make and the link at which it can be found (plus whether it's going to be a website or software application, the definition of those two is explained in the section 5.1.5). This form can be useful for VirtuaGym to contact these people and cooperate with them and their applications, but at the same time it shouldn't be too large, or else people will either just abort, or enter nonsense data. For example we did not require developers to fill in a description or motivation of the applications they have in mind: they would just be thrown off with such a sudden requirement. As soon as they filled in the required data, they are brought to a page that lists their Public key and Secret key. They can always look these keys back up, which is detailed in the section below.
- Then they're directed to a few bits of example code. These are the simplest pieces of code that are needed to communicate with the API. More detail about this is outlined in the section below.
- And once they've gone through these simple examples, the developers are directed to the full API documentation which contains a complete description and explanation of the API. More detail about that follows in the section below.

5.1.2 API Keys

This will be a list of API Keys that this user has registered. In practice there will only be one or two per user, but we did want to hold open the possibility for developers to register multiple keys, for example for different applications that they want to develop. Suppose that you have a manufacturer of three kinds of scales, it may be desirable to give each of those scales its own API Key.

[55] http://virtuatest.com/api.php - you can log in on a test account with username guest and password guestguest

There are different things you can do here at this page. First of all, you can always look your API Key back up in case you lost it (this is why it's located on such a prominent place on the page). You can also see the amount of times they've been accessed (they're simple statistics, and also give insight into how often their product is used). It's also possible to edit registration data (suppose you made a mistake, or change email address), or delete a key altogether.

For services that make use of Open Authorization it's also possible to show a list of tokens, and exactly which users have been making use of these tokens. Again this will allow developers to keep track of which users make use of their applications, and also allows them to block particular users in the case of misuse. This functionality was easy to implement because every user is linked to a token. With Basic HTTP Authentication showing this would require a lot more implementation effort, which wasn't worth it for a relatively insignificant feature.

5.1.3 Example Code

The example code is meant to give developers the easiest possible code that could make the API work. This is to show them how the protocol works, and give them incentive to make something more complex out of it. There are three sections at this page.

The first example only requires you to copy a link, insert your own data at the right places and paste it in a browser. This already should be enough to get some data out of VirtuaGym. It's the classic "hello world"-example of an url-based API because it's so easily accessible.

The next example is in PHP code. This is a widely used language, and fairly easy to immediately get started at as long as you have access to some server that runs it. Here, you're basically also reading in a variable, but in this example you can manipulate a lot more of what is returned because it's in an actual language, rather than simply copying a link into your browser. The easy to use cURL library is used to communicate with VirtuaGym's API with only a few lines of code.

The third example explains another basic feature of the API: submitting something. In this case, a new weight is submitted to the user's VirtuaGym account. This again is done in PHP , for consistency's sake. Like with the previous example, all the user needs to do here is paste the code in a php file and substitute his data at the right places.

5.1.4 Documentation

The documentation is a structured document that is meant to explain every part of the API to the developers. For the sake of easy navigation, we added a navigation bar to the side of the page, rather than splitting every section up in separate document because we felt that it would be easier to navigate and search through.

The documentation also makes use of a lot of examples, especially when illustrating how to use the Oauth Library. With this we hope to be as clear as possible for the potential developers, and so that they'll have enough examples for them to show how everything works. Without these examples it will especially for novice developers be very hard to imagine how to use the API the most efficiently.

5.1.5 Access Limit

While it is not likely to happen any time soon, it will be a problem when a spam bots or other kinds of malicious users spam the server with requests. Because of this, there's an access limit for every key, that shouldn't be triggered in normal usage.

The access limit for VirtuaGym limits accesses to a certain amount for every two minutes. This should prevent spam bots from going out of control, and at the same time when a regular user accidentally triggers it, it will be solved within a few minutes. For every access, the IP Address is also logged, so that whenever such a limit is broken frequently, appropriate actions can be taken (the API key in question along with its IP address can be banned from the site).

Because there are different types of applications, there are also different kinds of access limits. For example, we can log the IP addresses of individual users with applications for mobile devices, or personal computers: these kinds of applications are all locally installed. However, when a website communicates with the API, it will mean that all of the incoming requests will have the IP address of this requesting website, which will then very quickly flow its access limit when a large amount of users make use of it. This is why we make a distinction between two kinds of access limits:

- "Website" is meant for websites. These have a high limit for each API Key with its own IP address.
- "Application" is meant for applications that are directly installed on user's devices. Here, every IP address at every API Key gets his own (considerably lower compared to the website) access limit.

The database only stores the most recent access data for each user or website. So only the amount of accesses of the last two minutes in which a user or website was active are stored in the database. This is mainly done for performance reasons, because if you save past access data for too long, it'll just flood the database and lead to very poor performance. When something goes wrong, it will be written to the API Logging instead.

This method unfortunately is not flawless. Proxies nowadays still provide a lot of trouble for webmasters around the internet, and malicious users, can easily hide their IP addresses behind a proxy. When such a (unlikely) case turns up, it will have to be dealt with individually.

5.2 API Syntax

This section will outline the syntax that applications need to use in order to communicate with the API. First it will describe the protocols with which you can communicate with it. A MoSCoW list of all of the functionalities we considered to provide with the API can be found in Appendix B at subsection 1. A detailed description of the interface is located in Appendix B at subsection 2.

5.2.1 Communication Protocol

In this section we will explain the exact communication protocol that needs to be used in order to get access to the data at VirtuaGym.

Like we mentioned earlier in this Conclusion, we decided that a REST-like RPC communication architecture would be the best choice for VirtuaGym's API:

- The API consists out of a number of URLS, which can be accessed with GET, PUT, POST or DELETE HTTP requests in order to communicate with their data. By making this uniform, developers will know exactly how to talk to the API, with only a list of URLs. SOAP's communication scheme to do this for this would just be too complicated for the developer.
- We decided not to incorporate REST's link-based navigation scheme, in which developers need to ask the API for links to URLs that they're looking for. Even though this would make the API more scalable, it would also mean that we're giving an extra step for the developers to figure

out, and considering VirtuaGym's size, it is not likely that the API will grow so big that it becomes inconvenient enough for REST's link-based navigation to become worthwhile.

- We decided to support two authentication schemes: Basic HTTP Authentication and Open Authorization. The reason for this is that neither are sufficient enough on their own: Basic HTTP Authentication is very easy, but very unsafe. The developers who want safety can then decide to use Open Authorization, which is much safer, but a lot harder to understand and will turn off a lot of potential developers. We also decided not to go with Oauth 2.0[56], because at the time when this thesis was written it still was too young of a technology: it hasn't been properly tested or supported enough to become a safe option.

5.2.1.1 Oauth Communication Scheme

The Oauth Communication Scheme follows the communication scheme of the Oauth Library by Mediamatic[57]. Out of all the Oauth Libraries we could find for PHP, this was the most solid, structured and easy to understand.

The urls for requesting a request token, authorizing and requesting an access token are the following:

```
http://virtuagym.com/api.php?mode=request_token
http://virtuagym.com/api.php?mode=authorize
http://virtuagym.com/api.php?mode=access_token
```

We found this to be the friendliest place to place them, because it's in the same page as the other developer pages. This way, every the developers can just look at one place for all of the information they need to get the API running, and yet it is away from the actual data part from the API, making it more modular.

5.2.1.2 Basic HTTP Authentication Communication Scheme

This section outlines the protocol that needs to be followed in order to communicate with the API through Basic HTTP Authentication. Note that a prerequisite is that the user already registered an API key.

Take the following url as an example:

```
https://yourusername:yourpassword@virtuagym.com/api/v0/user?api_key=ap
ikey
```

All you need to do is replace `yourusername` with the user's username, `yourpassword` with the user's password and `apikey` with your API Key. It is also possible to send an encrypted version of your username and password inside the HTTP request instead of the URL, as according to the HTTP Auth Specifications[58].

You can see that the API Key needs to be appended at the end of the url. This can also be sent along as a POST variable. While it is more user friendly to not require such a key, we opted not to do this out of safety. This way we can see and control who try to access the API, and block them in the case

[56] http://wiki.oauth.net/OAuth-2.0
[57] http://code.google.com/p/oauth-php/
[58] http://en.wikipedia.org/wiki/Basic_access_authentication

of misuse. If we didn't require some sort of key to be sent along, we would have no way of knowing or blocking the identity of a malicious developer.

The API Key is the same as Oauth's Public Consumer Key. This allows developers to choose dynamically between Basic Auth and Oauth, so that you don't have to store different keys for both schemes. Once you have a correct HTTP request based on this URL, you can simply send it to VirtuaGym and get a response back, according to the HTTP Auth specifications.

5.2.1.3 URL Syntax

The structure of the urls that are used to communicate with the API is as follows:

- First there is the website address. `http://virtuagym.com`, `http://vitalence.com`, etc.
- Then it's followed by `/api`. This is to show that you're communicating with the API of this website, and not anything else.
- Then the version number follows. See the sub-section below.
- Then the page that needs to be accessed is appended. For example `/user` or `/bodymetrics`. These are rewritten using Apache's url rewriting module.
- Optionally, the output format follows then. See the sub-section below.
- In the case of Basic Authentication, the key also needs to be appended, in the following way: `?apikey=2314ae69c97c0b7a9`.

We found this to be the most readable scheme. We could have chosen to also append the API key with a slash through URL rewriting, but that didn't turn out to be easier on the eyes. In the same way, we chose to use url rewriting instead of listing various parameters through conventional GET-parameters because this would also limit the readability.

5.2.1.3.1 Versioning

In the future the API is likely to be adjusted quite a few times. In this case, versioning needs to be applied in order to not outcast the applications that were created under an older version.

Versioning could be done either by sending an extra POST-variable along with the HTTP request, containing the version name, however with this you won't be able to access the API anymore through a regular browser. It may seem like a function that doesn't have much practical use, but at the same time it is the first example that is used to explain the interaction with the API to new developers: it immediately shows them something that works and something they can immediately do something with. This is why we decided to just put the version number in the url. With the simple syntax: `/v0`. This is the shortest string that still immediately makes it clear to the user that it stands for a version. In future versions, this text will be updated to `/v1`, `/v2`, `/v3`, et cetera.

5.2.1.3.2 Output Format

With "Output Format", we mean in what way will the results get returned. The default is XML: when no parameter for this output format is sent along, results will be returned in XML. Another possibility is having the results returned in JSON. This is done by appending the string `/json` at the url, after the page that needs to be accessed.

Outputting your data in XML or JSON seems to be a bit of a holy war in the REST API development community, so instead we decided to just support both of them. These are the most popular and widely supported output formats out there, plus they're both easily readable by both humans and computers.

5.2.2 Error Handling

Whenever something goes wrong, an error gets returned. This is done in two ways. First of all, an appropriate status code is returned as an HTTP request[59]. The status codes are that are used are the standard HTTP status codes, and for the statuses, errors or faults that don't fit any of these numbers, a new, unused number is made up. As long as the status code is 200, the user knows that everything went right.

To help the developer a bit more, the status code and status message also get returned through the regular output. So for example an unauthorized call to `http://virtuagym.com/api/v0/user` results in the following reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<reply>
   <statuscode>401</statuscode>
   <statusmessage>Your login failed. Please go to VirtuaGym.com for
more information</statusmessage>
</reply>
```

Replies are often a bit more elaborate than their HTTP Counterparts, in order to give the developer good information as possible about what went wrong. If this isn't sufficient for the developer, then the documentation will explain this error even further.

5.2.3 Metric or Imperial units

In practice, APIs tend to consistently stick with just one unit type (metric or imperial) and leave it to application developers to convert this when needed. However, because the target market of VirtuaGym is Europe, it will have to deal with people who are used to these different unit systems for example weight, length and calories. What you want to avoid is that developers have no idea what kind of values their units are returned in. It may seem natural for a Dutch developer that all values are returned in kilograms, rather than ounces, but this isn't the case for an English one. This is why the API responses are included with fields containing the unit of each value, to make them more understandable for third party developers.

We do need to be consistent with this, however. It's very confusing when some values are returned in meters and others in pounds. With this consistency, we also don't have to request developers, when trying to insert new values for a user, to also specify the units for them. This would just be too hard to find out (requesting developers to find out whether they must use feet or inches, for example), and it would make too much of a mess around the code. Because of this, we decided to return all values in default in the metric system.

5.2.4 Supported Data

As for the question of what kinds of information should be obtainable by the VirtuaGym API, we decided against just putting every single functionality on-line. For example, there is little to no practical use of being able to access the user's location from outside of VirtuaGym, or the amount of time that his profile has been visited. Having an option for a third party application to access these

[59] http://en.wikipedia.org/wiki/List_of_HTTP_status_codes

values would only make the API more unnecessarily complicated. This is why only information with any practical use will be shared.

5.3 Admin Functions

This section is small, but it does outline the functionality for the VirtuaGym administrators over the API, which will allow them to moderate the API traffic. It's not much, and it probably won't have to be used in the next couple of years, but in the event of a bug or malicious user it should be sufficient to help the administrators respond appropriately.

5.3.1 The Admin Panel

The admin panel allows administrators to quickly oversee all of the API Keys that got registered at VirtuaGym. It mostly shows the basic information, but it also supports the option to delete API keys, or to simply disable them in the case something goes wrong with them or someone is misusing them. It also shows some small statistics (the total amount of accesses for each key) to give a global idea of how well all of the API Keys are used. Any more specific statistics would just clutter this page and not contribute something significantly new.

5.3.2 Logging

The API also has a logging function, which can be found in the same directory in which the other logs at VirtuaGym are stored. The only thing it logs however, are the error messages that the API generates. This will allow the VirtuaGym administrators to provide adequate support when something goes wrong, but reporting anything more would just introduce useless noise. Again, statistics like these don't contribute anything significantly new, and they just make it much harder to find something in the log files.

6 The Proof of Concept

In order to demonstrate the functionality of the API, a proof of concept was developed, which was meant to act as an example application that communicated with the API. Two versions were developed throughout the process of the development of the API, and this section will describe these two applications.

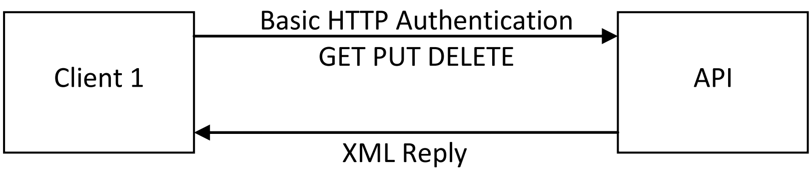
6.1 Version 1

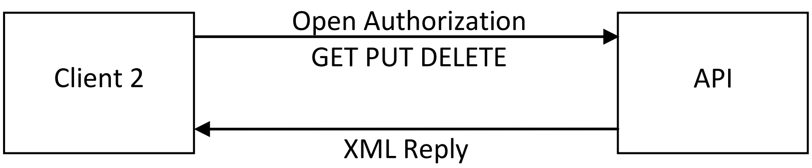
6.1.1 Description

The first version was mostly meant to test out the basic functionality of the first versions of the API. It was not meant to be complete, but rather cover the basic functionality of the API:

- Logging in with Basic HTTP Authentication and Open Authorization.
- Retrieving data from the server.
- Inserting new data into the server.
- Deleting data from the server.

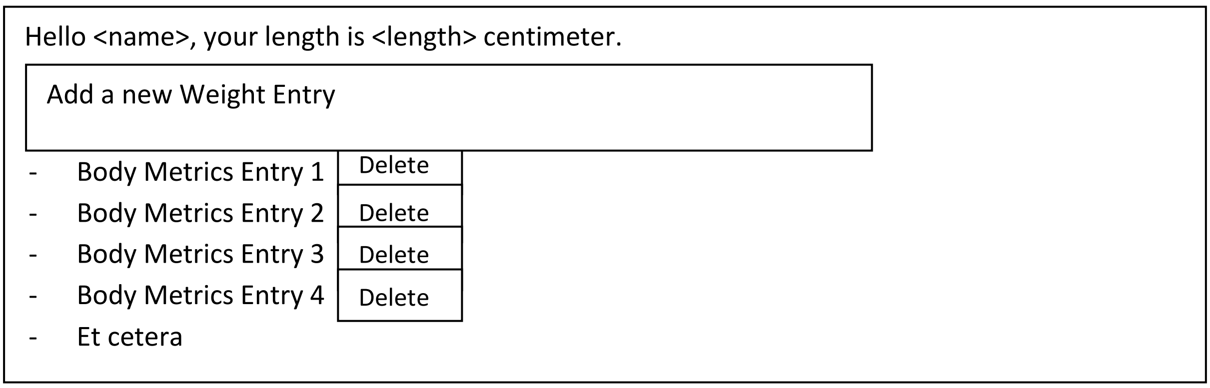
Based on this, two versions were made: one that allows the user to log in using Basic HTTP Authentication and one using Open Authorization. Combining the two would have made it unnecessarily complicated at that stage. Inside the application, you could view the user's name and age, along with the body metrics he submitted, which you could also delete. On top of that, you're also able to insert a new weight for the logged in user. The following diagrams illustrate this:





Even though the API supports replies in both XML and JSON, we decided not to include a JSON interpretation function in the proof of concept. This is because of simplicity, but there are currently enough JSON interpreters out there that it's not very interesting for us to show another version here, so we decided to simply stick with just one output format.

The lay-out of both clients is identical, and described in the following diagram:



“<name>” and “<length>” are values that can be obtained through the user-page of the API. Of course, there is more data on that page, such as the user’s age, birthday, location or user id, but those two values were enough to demonstrate that this function worked, which was the main purpose of this application.

The block “Add a new Weight Entry” consists out of an input field and a button, in a regular HTML form. It’s linked to a function that takes the value that is submitted in that text box and submits it to VirtuaGym as a new weight entry. So if you were to insert 67 and click submit, if everything went right then your new weight should be 67 kg. At the time when this version of the proof of concept was developed, units and the imperial system were not important enough to warrant extra effort to implement a function that also allows you to submit your weight in pounds. This was meant to give an as clear overview as possible.

Below that are all of the body metrics that the user has submitted at VirtuaGym. This can be weight data, but also BMI, fat percentage, your virtual age and the number of lunches, crunches or push-ups you performed at VirtuaGym’s Fittest.

When an error occurs (for example the user inserted a wrong password), this error is simply shown with a Javascript alert dialog. When the user closes this dialog, he is directed back to the page he started at in order to try again with a correct value.

There are a number of scenarios for the functionality of this Proof of Concept: submitting and deleting a weight, both with Basic HTTP Authentication and Open Authorization. They can be found in Appendix A at section 9.3.1.

6.1.2 Authentication

Like said in the previous section: there are two versions with both their own ways of authenticating. This section will detail the algorithm for both of them.

6.1.2.1 Client 1: Basic HTTP Authentication

When they access Client 1, users are treated to a login screen that asks the user’s username and password at VirtuaGym. When they fill this in, they’re sent to a new page containing the client itself. The client keeps track of this username and password through forms, and this again shows what an insecure method Basic Authentication can be, because this means that the user’s password needs to be printed in plain HTML in the source code. This is why Client 2 with Open Authorization takes a different approach.

6.1.2.2 Client 2: Open Authorization

For open authorization the authentication process is a bit more complicated. These are the steps that are followed:

1. The user is greeted with a login screen. This screen asks for just a nickname. This does not have to be VirtuaGym’s nickname, as it is only used to login at the client’s side, this is to allow different users to sign in.
2. The user is then directed to the VirtuaGym Authorization page, at which he enters his username and password and confirms the application access to his data at VirtuaGym.
3. The client then creates an access token that can be used to exchange information with VirtuaGym without needing to login anymore.

4. The user is then directed to a callback page at the client, from which he can go back to the main page of the client.
5. The user logs in to the client again, but because he's already logged in at VirtuaGym he doesn't need to login there anymore, and he can immediately start exchanging information.

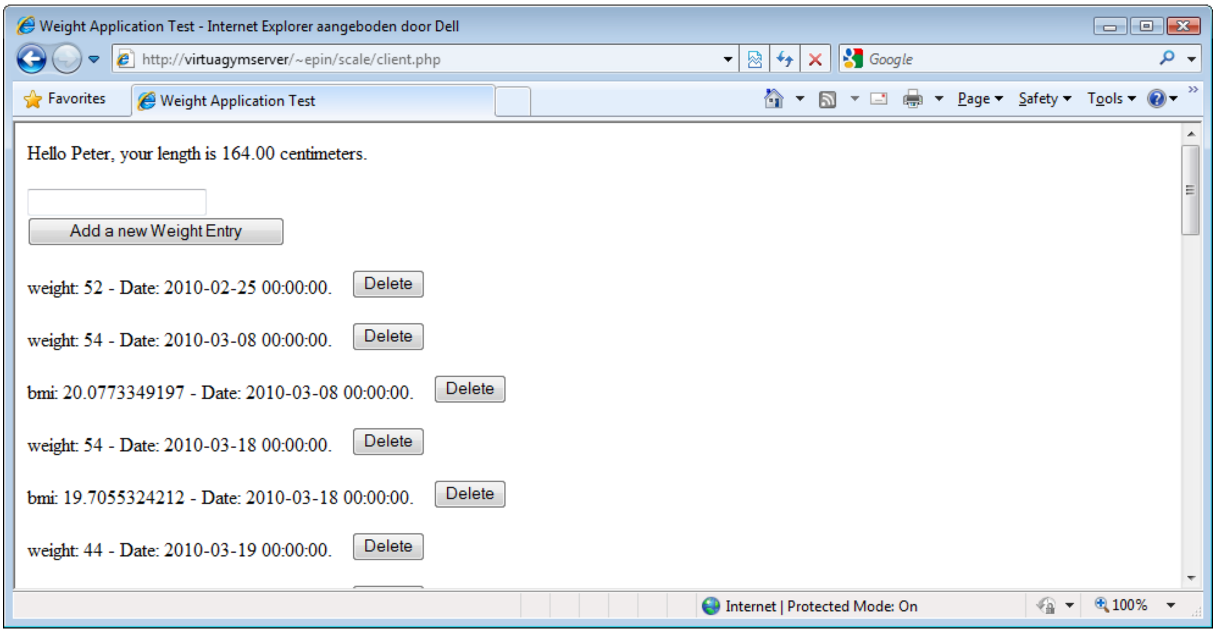
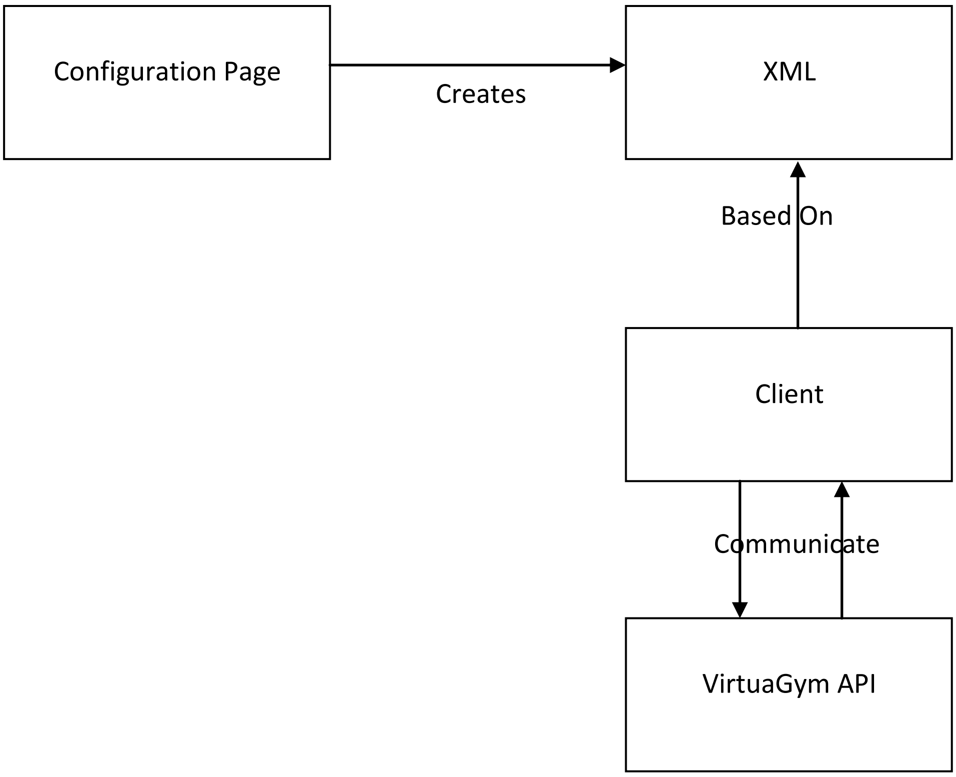


Figure 4: A screenshot of the first version of the proof of concept

6.2 Version 2

The second version of the proof of concept attempts to give a complete overview of the API. The idea is for users to compile their own Application Website, with the functionality that they are interested in. It is illustrated through the following diagram:



The configuration page consists out of a list of check-boxes that the user can set and unset. To prevent them from cluttering the entire page, they can be easily hidden with Javascript. Each check-box is linked to one of the fields of one of the pages of the API (for example, the date of the body metrics, or the exercise id of the activities). Each field that can be obtained through the GET function has its own check-box, each field that can be a parameter when sending a new entry to Vitalence using PUT gets its own check-box, and the values that can be deleted (body metrics and activities) also get one check-box each.

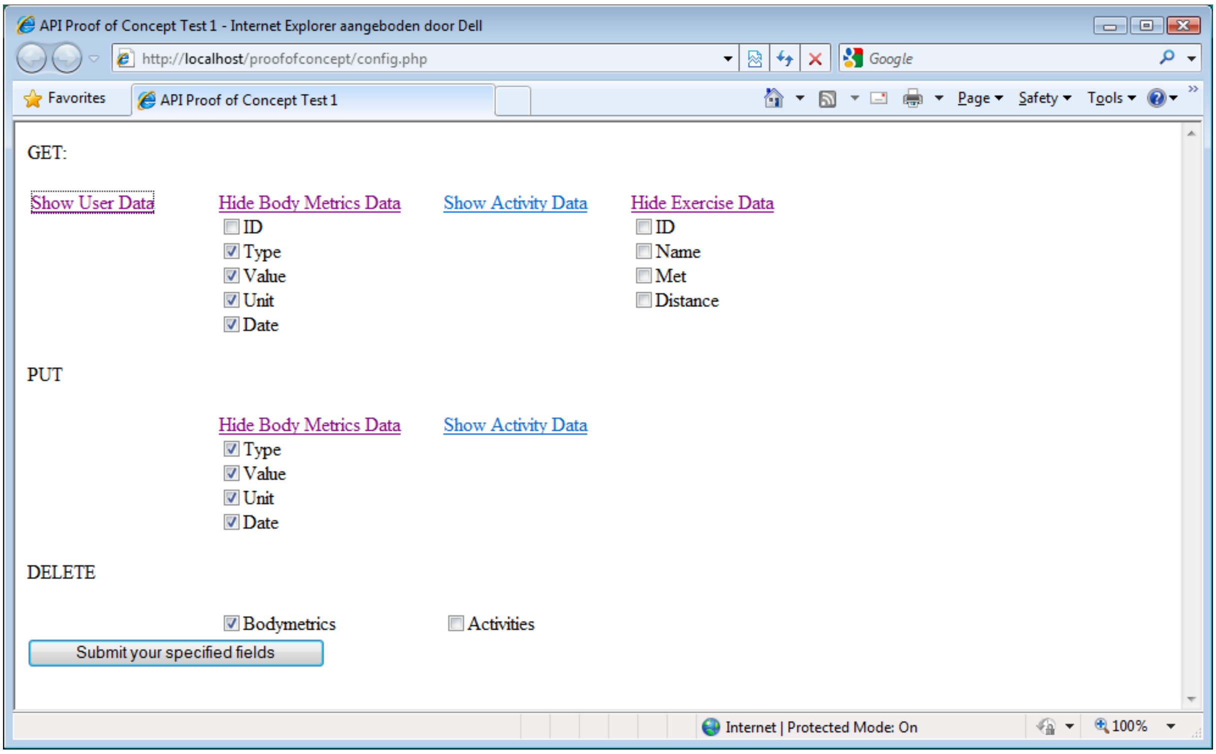


Figure 5: a screenshot of the config page in which you can check and uncheck the data you want to see at the demonstrator

This is all submitted and stored somewhere. We decided to do this in an XML-file because it's easy to read by a human (which also allows it to be edited in an XML editor, or even manually). We did consider to send this across using the GET method of PHP, but this proved to be just too unfeasible: the URLs would be too flooded with different parameters, and even though you would be able to save it more easily (it's just one big URL now that controls what the client looks like), there are just too many variables here to send along with it. Especially considering that it's still unknown how many more values will be added in the future.

The client then imports this XML-file, and implements each option that was checked in by the user, while leaving out the functionality that wasn't checked at all. For example, if the user only checked his name and birthday at the configuration file, only those will be shown: any body metrics or activities will be hidden, as well as the options to insert or delete them from the VirtuaGym Database.

To insert entries, some values are mandatory (for example, each submitted body metric needs a type and a value, while each activity needs an exercise id, duration and burned calories). So when the user checks in the date parameter for adding a new Body Metrics Entry, then automatically also the value and type get checked. Any entry that is not optional has the same behaviour when another entry in its page is checked.

The client works on the same principle as the client for Version 1 of the Proof of Concept, only with the extra added functionality. Authentication is done with either Basic HTTP Authentication or Open Authorization, and the client simply prints all of the required GET-data on the screen. PUT data meanwhile is asked through either HTML text field. In case the user selected any units, he also gets the chance to specify what kind of units he wants to submit these values in (only if these are supported by VirtuaGym).

The goal of this is to provide a complete environment of the API's functionalities, safe for the output in JSON. This is to fully show potential developers the capabilities of the API, give them inspiration and perhaps even offer them the start of an application that they were thinking of.

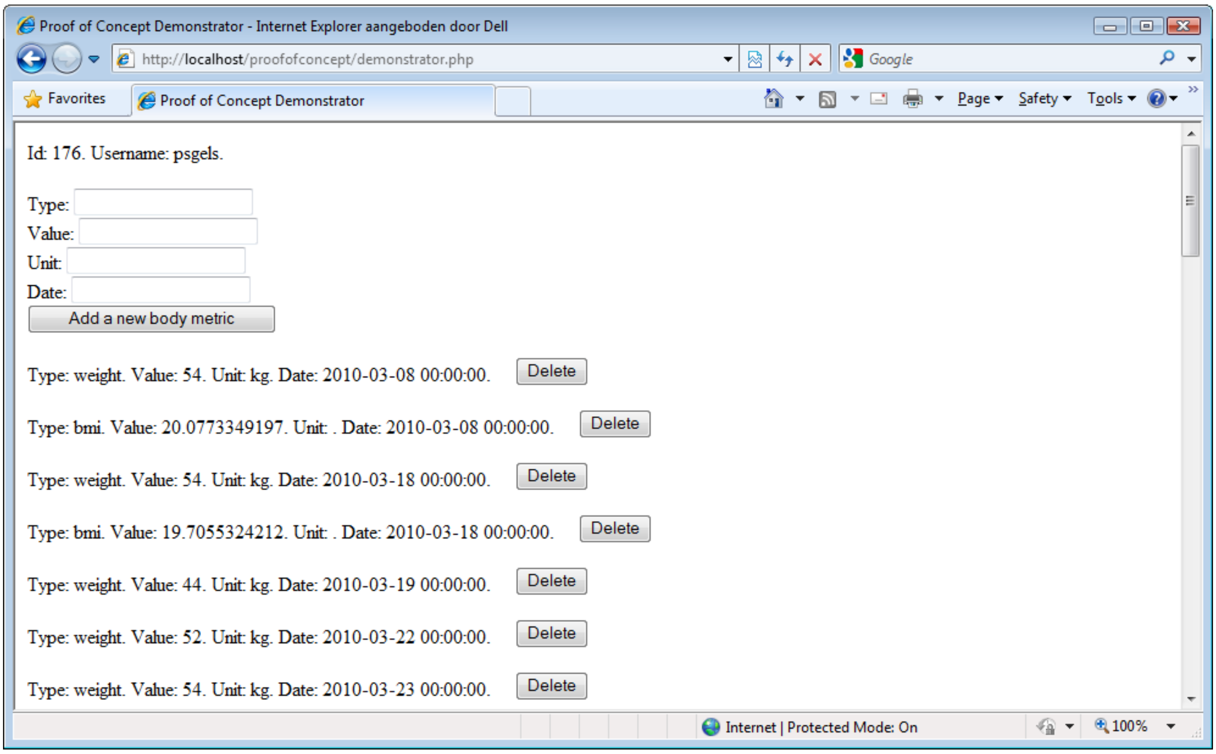


Figure 6: a screenshot of the proof of concept in action.

Like in the previous section, there are also scenarios for the functionality of this Proof of Concept: editing the configuration file, adding something to and deleting something from VirtuaGym. They can be found in Appendix A at section 9.3.2.

7 Possible Uses

With the API now designed, it is also important to look at the possible uses of the API: who are going to use it, what kinds of applications or websites will be developed for it, and what is going to communicate with it? This section provides an overview of different possible applications of the API when it is officially released.

There are a lot of possibilities for the mobile market, such as the iPhone and Android (and in the future also the iPad). At the moment, there are already an VirtuaGym iPhone application and Android app, but now people can make their own applications. Think about applications that let you submit body metrics or activities through your mobile. This can be combined with a GPS application, or a pedometer during running. You can submit body metrics or activities from outside of VirtuaGym, which will make interaction with the site a lot more user friendly.

This can be applied to scales in the same way. Currently VirtuaGym already has a partnership with Tanita Scales, and using the API to communicate to them saves a lot of time and effort compared to just going into VirtuaGym in order to connect it. And this way, Tanita developers can also establish this simple connection.

When it comes to communicate with parties that already have their own API, Mashups can provide the answer. Mashups are basically applications that try to connect multiple APIs with each other. With a bit of effort, you can write applications that automatically update data into VirtuaGym (or applications that check once a day for new updates) from platforms such as Withings[60], Garmin[61], or perhaps even Spanish websites as Fitbook[62].

A lot of these possible applications are also discussed at section 2 of this thesis (Current Standards in Communication to Fitness Equipment). In the same way, Mash-ups can also be created that periodically poll Google maps accounts for GPS data that was submitted by VirtuaGym users, to automatically submit activities containing the distance and speed found in the maps.

At the moment, there are already a number of parties with concrete interests in the API. Heartrate Games[63] manufactures fitness games, and they plan to use the API to automatically submit fit-scores to VirtuaGym.

In the same vein, the API provides a great opportunity for fitness games: new developers of Fitness games can now easily submit the achievements that users make on their own platforms to VirtuaGym, offering a great link in communication that will benefit both parties by combining them together.

We can also learn a lot about the possible applications and mash-ups that could be created by looking at what was created for the Dailyburn's API (see Section 3: Practices in online Fitness and Lifestyle Communities). Take for example the guy who created a Wordpress plug-in that allowed you to list your fitness progress on your blog: there are a great number of fitness blogs out there, and this would be a great opportunity for bloggers to show off their progress. However, you can also imagine

60 http://www.withings.com/nl/api/weegschaal
61 http://www8.garmin.com/products/communicator/
62 http://www.fitbook.es/apis_de_fitbook.php
63 http://www.heartrategames.com/index.htm

plug-ins that can be created for all kinds of other types of social networks, for example forums like Invisionfree, PHPBB or community websites as Joomla or Drupal.

It's also vital that some kind of group on VirtuaGym is created, in order to assist potential developers in the case that they have questions or can't seem to figure out how something works. This group will also serve as brining these developers and their ideas together. One idea can also be to create some sort of topic that stimulates people to list the ideas they have to use the API for.

7.1 Future Perspective

We imagine that in the future of VirtuaGym and Vitalence, a lot more of VirtuaGym's functionality can easily be made available through the API. Right now, you can only communicate with the user's user data, body metrics, activities and exercises, but it's very easy to extend the API so that it also includes user data body metrics and activities of other users, or your goals.

One of the limitations of VirtuaGym and the API is that you can only choose a set amount of activities or body metric types. In the future, it might pay off to allow users to submit their own types so that they can really customize the way they want to track their sports activities. For the icons of each activities, you could choose to use some default icon for the ones who don't want to provide one.

This can also become handy for game developers who want to submit their own scores, rather than making use of VirtuaGym's fitscore (Heartrate Games, for example).

7.2 For Developers

To close off, we would like to provide a list of suggestions that future developers can build further upon:

- Right now only the user data, body metrics and activities of the logged in user in VirtuaGym are supported by the API. How about goals, or the data of other users around you?
- Mashups that will combine the API of VirtuaGym with other APIs (Garmin? Google Maps?)
- Develop the connection between Garmin and Google Maps as proposed by Appendix C.
- Improve the security aspects by implementing Oauth 2.0, as soon as it is fully tested and accepted as the standard.

8 Conclusion

For this thesis, we developed an Application Programming Interface for VirtuaGym and researched the elements of an open communication standard in fitness and lifestyle platforms, to which parties as VirtuaGym can adhere. A landscape study was performed that examined in what ways it would be possible to easily communicate to fitness equipment. We discussed various types of architectures and considerations that need to be made when developing such an API, and we detailed the design decisions that were made for the API we developed.

An architecture for a connection to an online fitness and lifestyle platform should be as simple and usable as possible: in general, they will not be expected to grow to unmanageable sizes, so a RPC framework that has the uniformities of REST, but not its specific syntax that needs to be studied before it is fully understood, will suffice. There is not one optimal way to authenticate users: a combination of two techniques (such as Basic HTTP Auth and Oauth) is necessary to find the right balance between security and usability. The critical success factors consist of an easy to use but secure interface that satisfies the minimum requirements.

For VirtuaGym, REST-like RPC also suffices best as a framework: clients can send GET (retrieve),PUT (insert) and DELETE requests to a number of pages, a list of which is provided to the user in advance. Since Oauth and Basic Auth both have their disadvantages, we decided to support both authentication methods. Communications to treadmills, cross trainers and home trainers (with CSAFE or iFIT live) is unfortunately hard to do without cooperation of the manufacturers. Wireless technology however such as Garmin's GPS devices is very open and features a lot of ways to communicate to and use their data.

9 Appendix A: Scenarios

9.1 Scenarios Google Maps Integration

9.1.1 Garmin

9.1.1.1 Submit Course Scenario

9.1.1.1.1 Characteristic information

Description	In this scenario the user submits a track he made with his Garmin GPS Device to VirtuaGym.
Precondition	- The user is logged into VirtuaGym. - The user has the Garmin Communicator Plug-in Installed into his browser.
Success End Condition	A new activity containing the course's data has been submitted to VirtuaGym and the user must now be able to view his course on an embedded Google Map.
Failure End Condition	A new activity has not been submitted to VirtuaGym or the viewer is not able to view his course on an embedded Google Map.

9.1.1.1.2 Main Success Scenario

Step	Actor	Action Description
1	User	Creates a course on his Garmin Device.
2	User	Uploads his course data to VirtuaGym using Garmin Communicator Plug-in.
3	VirtuaGym	Reads the uploaded .GPX-file, calculates the total distance travelled and submits this as a new activity to VirtuaGym's fitplan.
4	VirtuaGym	Stores the GPX-file somewhere on its server for further reference.

9.1.1.1.3 Scenario Exceptions

Step	Actor	Action Description
2	User	Interrupts / refuses uploading of course data.
3	VirtuaGym	The GPX-file is somehow corrupt or unreadable (too big?)

9.1.1.2 View Course Scenario

9.1.1.2.1 Characteristic information

Description	In this scenario the user views a map containing one of the tracks he submitted with his Garmin Device to VirtuaGym earlier.
Precondition	- The user is logged into VirtuaGym. - The user has already uploaded a course to VirtuaGym through Garmin's Communicator Plug-in.
Success End Condition	The user sees a Google Map containing his selected course in his browser.
Failure End Condition	The user does not see a Google Map containing his selected course in his browser.

9.1.1.2.2 Main Success Scenario

Step	Actor	Action Description
1	User	Accesses a page on VirtuaGym that contains an overview of the courses that he has submitted.
2	User	Picks one of the courses to view.
3	VirtuaGym	Reads the appropriate GPX File, accesses Google Maps' API and overlays the course data on top of a web-page embedded map.

9.1.1.2.3 Scenario Exceptions

Step	Actor	Action Description
2	User	Leaves the page

9.1.2 MyTracks

9.1.2.1 Submit Course Scenario

9.1.2.1.1 Characteristic information

Description	In this scenario the user submits a track he made with MyTracks to VirtuaGym.
Precondition	- The user is logged into VirtuaGym.
Success End Condition	A new activity containing the course's data has been submitted to VirtuaGym and the user must now be able to view his course on an embedded Google Map.
Failure End Condition	A new activity has not been submitted to VirtuaGym or the viewer is not able to view his course on an embedded Google Map.

9.1.2.1.2 Main Success Scenario

Step	Actor	Action Description
1	User	Creates a course with MyTracks on his android phone.
2	User	Uploads his course data to Google Maps .
3	User	Requests the direct link to this Google Map, and saves it to the clipboard.
4	User	Accesses VirtuaGym and submits this link.
5	VirtuaGym	Saves the link to the VirtuaGym Database and converts the link into an RSS Link.
6	VirtuaGym	Requests the RSS of the track from Google Maps, calculates the total time and distance that was travelled and submits a new activity to the user's fit plan.

9.1.2.1.3 Scenario Exceptions

Step	Actor	Action Description
1	User	Creates a track outside of GPS coverage.
2	User	Aborts the uploading process.
3	User	Aborts the page, doesn't copy the URL.
4	User	Leaves the web page.

9.1.2.2 View Course Scenario

9.1.2.2.1 Characteristic information

Description	In this scenario the user views a map containing one of the tracks he submitted with MyTracks to VirtuaGym earlier.
Precondition	- The user is logged into VirtuaGym. - The user has already uploaded a MyTracks course to VirtuaGym.
Success End Condition	The user sees a Google Map containing his selected course in his browser.
Failure End Condition	The user does not see a Google Map containing his selected course in his browser.

9.1.2.2.2 Main Success Scenario

Step	Actor	Action Description
1	User	Accesses a page on VirtuaGym that contains an overview of the courses that he has submitted.
2	User	Clicks on one of the courses that was submitted with MyTracks to view.

3	VirtuaGym	Grabs the link to this track from its database and embeds it as a Google Map.

9.1.2.2.3 Scenario Exceptions

Step	Actor	Action Description
2	User	Leaves the page

9.2 API Scenarios

This section contains a detailed description of the use cases that were provided in the previous section. Every sub-section contains characteristic information about the scenario, which contains the description, its pre- and post conditions. The main success scenario outlines the steps that need to be followed in order to complete the scenario successfully. The Scenario Exceptions list the steps at which actions can be taken which will result in a non-successful execution of the scenario. The actors that will be used throughout the scenarios are defined as follows:

- Developer — The developer of the Application.
- Application — The piece of software that communicates with VirtuaGym through the API.
- User — The human person who uses the Application.
- VirtuaGym — The VirtuaGym and Vitalence Server.
- Admin — The human administrator of VirtuaGym.

9.2.1 Register Key

9.2.1.1 Characteristic information

Description	In this scenario the developer registers an API key in order to be able to build an application that communicates with the API
Precondition	- The developer is logged into VirtuaGym.
Success End Condition	A new API key and API secret key has been displayed on the screen for the user.
Failure End Condition	-

9.2.1.2 Main Success Scenario

Step	Actor	Action Description
1	Developer	Accesses the API registration page.
2	Developer	Fills in his e-mail, phone number . application title, application URL and whether it's an application or website and submits.
3	VirtuaGym	Creates a new API key based on the data that the developer submitted.
4	VirtuaGym	Displays the API Key and API Secret Key for the user, alongside a confirmation that the API key has been created.

9.2.1.3 Scenario Exceptions

Step	Actor	Action Description
2	Developer	Fills in an invalid E-mail address or phone number.
2	Developer	Leaves the page

9.2.2 Edit Key Information

9.2.2.1 *Characteristic information*

Description	In this scenario the developer edits the information for one of the keys that he has registered.
Precondition	- The developer is logged into VirtuaGym. - The developer already has an API Key registered.
Success End Condition	The information of one of the registered keys of the developer has been changed.
Failure End Condition	-

9.2.2.2 *Main Success Scenario*

Step	Actor	Action Description
1	Developer	Access the API Keys Page.
2	Developer	Clicks on one of the API Keys.
3	VirtuaGym	Redirects the user to the information edit page of that API Key.
4	Developer	Changes his email address, phone number, application title, application URL or whether it's a website or application and submits.
5	VirtuaGym	Changes the API Key Information of the key in question.

9.2.2.3 *Scenario Exceptions*

Step	Actor	Action Description
2	Developer	Leaves the Page
4	Developer	Leaves the Page / clicks on a "go back "-link.
4	Developer	Fills in an invalid email address or phone number.

9.2.3 Toggle Disable Oauth Tokens

9.2.3.1 *Characteristic information*

Description	In this scenario the developer disables or enables the access to one of the Oauth Tokens that has been registered for one of his API Keys.
Precondition	- The developer has an API Key registered. - The developer has an Oauth token registered for his API Key. - The developer is logged into VirtuaGym
Success End Condition	Access to the Oauth Token in question Is disabled if it was enabled at the start of the scenario, and it is enabled if it was disabled at the start of the scenario.
Failure End Condition	The disabled status of the Oauth Token has not changed compared to its state at the beginning of the scenario.

9.2.3.2 *Main Success Scenario*

Step	Actor	Action Description
1	Developer	Accesses the API Keys Page
2	Developer	Clicks on the link to the Oauth Token Page for one of his API Keys.
3	VirtuaGym	Redirects the developer to the Oauth Token Page for this API Key.
4	Developer	Clicks on a button that toggles whether or not one of the tokens has been disabled.
5	VirtuaGym	Disables that token if it was enabled and enables the token if it was disabled and

		shows a confirmation on the screen.

9.2.3.3 Scenario Exceptions

Step	Actor	Action Description
2	Developer	Leaves the page
4	Developer	Leaves the page

9.2.4 Delete Oauth Tokens

9.2.4.1 Characteristic information

Description	In this scenario the developer deletes one of the Oauth Tokens that has been registered for one of his API Keys.
Precondition	- The developer has an API Key Registered - The developer has an Oauth Token registered for his API Key - The developer is logged into VirtuaGym
Success End Condition	The Oauth Token in question has been deleted.
Failure End Condition	The Oauth Token in question has not been deleted.

9.2.4.2 Main Success Scenario

Step	Actor	Action Description
1	Developer	Accesses the API Keys Page
2	Developer	Clicks on the link to the Oauth Token Page for one of his API Keys.
3	VirtuaGym	Redirects the developer to the Oauth Token Page for this API Key.
4	Developer	Clicks on a button that deletes one of the Tokens.
5	VirtuaGym	Deletes the token and shows a confirmation on the screen.

9.2.4.3 Scenario Exceptions

Step	Actor	Action Description
2	Developer	Leaves the page
4	Developer	Leaves the page

9.2.5 Get User Info

9.2.5.1 Characteristic information

Description	In this scenario, the user uses an application that is registered at VirtuaGym in order to view his user information (such as his name, avatar URL and length).
Precondition	- The application has access to VirtuaGym.
Success End Condition	The application has displayed the user's information to the user.
Failure End Condition	The application has received an error code, along with an error message.

9.2.5.2 Main Success Scenario

Step	Actor	Action Description
1	User	Uses the Application to request the retrieval of his information.
2	Application	Sends a request for the user's information along with authentication to

		VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
3	VirtuaGym	Looks up the user's information and sends it back to the Application.
4	Application	Displays the user's information for the user.

9.2.5.3 Scenario Exceptions

Step	Actor	Action Description
3	VirtuaGym	The user's authentication turns out to be invalid.

9.2.6 Delete Body Metric

9.2.6.1 Characteristic information

Description	In this scenario the user uses the Application in order to view his body metrics, in order to delete one of them subsequently.
Precondition	- The user has already a body metric submitted.
Success End Condition	The Application has displayed a confirmation that the body metric has been deleted.
Failure End Condition	The Application has received an error code along with an error message.

9.2.6.2 Main Success Scenario

Step	Actor	Action Description
1	User	Uses the Application to request the retrieval of his body metrics.
2	Application	Sends a request for the user's body metrics along with authentication to VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
3	VirtuaGym	Looks up the user's body metrics and sends them back to the Application.
4	Application	Displays the body metrics.
5	User	Picks one of the body metrics and uses the Application to delete it.
6	Application	Sends a request to delete this particular body metric with authentication to VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
7	VirtuaGym	Deletes the body metric in question and sends a confirmation back to the Application.
8	Application	Displays a confirmation back to the user.

9.2.6.3 Scenario Exceptions

Step	Actor	Action Description
3	VirtuaGym	The user's authentication turns out to be invalid.
5	User	The user aborts the scenario and does not pick a body metric to delete.
7	VirtuaGym	The user's authentication turns out to be invalid.
7	VirtuaGym	The id of the body metric turns out to be invalid.

9.2.7 Add Body metric

9.2.7.1 Characteristic information

Description	In this scenario the user uses the Application in order to insert a new body metric .
Precondition	-
Success End Condition	The Application has received a confirmation that the body metric has been added to VirtuaGym

Failure End Condition	The application has received an error code and error message.

9.2.7.2 Main Success Scenario

Step	Actor	Action Description
1	User	Uses the Application to request the addition of a new body metric.
2	Application	Sends a request plus the required data for the body metric along with authentication to VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
3	VirtuaGym	Adds the body metric and sends a confirmation back to the Application.

9.2.7.3 Scenario Exceptions

Step	Actor	Action Description
3	VirtuaGym	The user's authentication turns out to be invalid.
3	VirtuaGym	The data for the body metric turns out to be invalid (for example a height that doesn't match).

9.2.8 Delete Activity

9.2.8.1 Characteristic information

Description	In this scenario the user uses the Application in order to view his activities, in order to delete one of them subsequently.
Precondition	- The user has already an activity submitted.
Success End Condition	The Application has displayed a confirmation that the activity has been deleted.
Failure End Condition	The Application has received an error code along with an error message.

9.2.8.2 Main Success Scenario

Step	Actor	Action Description
1	User	Uses the Application to request the retrieval of his activities.
2	Application	Sends a request for the user's activities along with authentication to VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
3	VirtuaGym	Looks up the user's activities and sends them back to the Application.
4	Application	Displays the activities.
5	User	Picks one of the activities and uses the Application to delete it.
6	Application	Sends a request to delete this particular activities with authentication to VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
7	VirtuaGym	Deletes the activity in question and sends a confirmation back to the Application.
8	Application	Displays a confirmation back to the user.

9.2.8.3 Scenario Exceptions

Step	Actor	Action Description
3	VirtuaGym	The user's authentication turns out to be invalid.
5	User	The user aborts the scenario and does not pick an activity to delete.
7	VirtuaGym	The user's authentication turns out to be invalid.
7	VirtuaGym	The id of the activity turns out to be invalid.

9.2.9 Add Activity

9.2.9.1 Characteristic information

Description	In this scenario the user uses the Application in order to insert a new activity .
Precondition	-
Success End Condition	The Application has received a confirmation that the activity has been added to VirtuaGym
Failure End Condition	The application has received an error code and error message.

9.2.9.2 Main Success Scenario

Step	Actor	Action Description
1	User	Lets the Application know that it wants to add a new activity.
2	Application	Sends a request for a list of exercises to VirtuaGym, along with authentication. Authentication can be done with either Basic HTTP Auth or Oauth.
3	VirtuaGym	Returns a list of all the possible exercises that a user can add as an activity to the Application.
4	Application	Lets the user choose out of a subset of these exercises.
5	User	Uses the Application to request the addition of a new activity for the exercise he has done.
6	Application	Sends a request plus the required data for the activity along with authentication to VirtuaGym. Authentication can be done with either Basic HTTP Auth or Oauth.
7	VirtuaGym	Adds the activity and sends a confirmation back to the Application.

9.2.9.3 Scenario Exceptions

Step	Actor	Action Description
3	VirtuaGym	The user's authentication turns out to be invalid.
7	VirtuaGym	The user's authentication turns out to be invalid.
7	VirtuaGym	The data for the activity turns out to be invalid (for example a duration that makes no sense).

9.2.10 Toggle Disable Key

9.2.10.1 Characteristic information

Description	In this scenario the admin toggles whether or not one of the API keys that are registered at VirtuaGym is disabled.
Precondition	- The admin needs to be logged in as admin. - An API Key needs to be registered at VirtuaGym.
Success End Condition	The API Key in question has been disabled and can't be used to access VirtuaGym anymore if it was enabled. If it was disabled, it's now enabled again. .
Failure End Condition	The API Key in question has not been disabled.

9.2.10.2 Main Success Scenario

Step	Actor	Action Description
1	Admin	Accesses the API Keys Admin Page
2	Admin	Toggles the disable function for one of the API Keys.
3	VirtuaGym	Enables the API Key if it was disabled, and disables the token if it was enabled,

		and sends some sort of visual confirmation to the user.

9.2.10.3 Scenario Exceptions

Step	Actor	Action Description
2	Admin	Leaves the page.

9.2.11 Delete Key

9.2.11.1 Characteristic information

Description	In this scenario the admin deletes one of the API Keys that are registered at VirtuaGym.
Precondition	- The admin needs to be logged in as admin. - An API Key needs to be registered at VirtuaGym.
Success End Condition	The API Key in question has been deleted.
Failure End Condition	The API Key in question has not been deleted.

9.2.11.2 Main Success Scenario

Step	Actor	Action Description
1	Admin	Accesses the API Keys Admin Page
2	Admin	Clicks on the delete button for the API Key that he wants to delete.
3	VirtuaGym	Deletes the API Key.

9.2.11.3 Scenario Exceptions

Step	Actor	Action Description
2	Admin	Leaves the page.

9.3 Scenarios Proof of Concept

9.3.1 Version 1

9.3.1.1 Insert New Weight Basic HTTP Auth

9.3.1.1.1 Characteristic information

Description	In this scenario the user uses Client 1 (the version that logs in using Basic HTTP Authentication) in order to insert a new weight.
Precondition	-
Success End Condition	The new weight that has been submitted is now added to VirtuaGym and listed at the list of body metrics at the clients.
Failure End Condition	The application has received an error code and error message, which are displayed to the user.

9.3.1.1.2 Main Success Scenario

Step	Actor	Action Description
1	User	Logs in at the Client with his username and password.
2	Client	Asks the user’s name, length and submitted body metrics from VirtuaGym, using the user’s username and password and displays this to the user.
3	User	Inserts a value and submits a new weight to VirtuaGym through the client.

4	Client	Sends a request plus the required data for the weight along with the user's username and password to VirtuaGym.
5	VirtuaGym	Adds the weight and sends a confirmation back to the Client.
6	Client	Asks the user's name, length and submitted weight (which now include the newly submitted weight) from VirtuaGym, using the user's username and password and displays this to the user.

9.3.1.1.3 Scenario Exceptions

Step	Actor	Action Description
2	VirtuaGym	The user's username or password turns out to be invalid.
5	VirtuaGym	The value of the weight turns out to be invalid.

9.3.1.2 Delete Weight Basic HTTP Auth

9.3.1.2.1 Characteristic information

Description	In this scenario the user uses Client 1 (the version that logs in using Basic HTTP Authentication) in order to delete a body metric.
Precondition	-
Success End Condition	The body metric in question is deleted from VirtuaGym and disappeared at the list of body metrics at the clients.
Failure End Condition	The application has received an error code and error message, which are displayed to the user.

9.3.1.2.2 Main Success Scenario

Step	Actor	Action Description
1	User	Logs in at the Client with his username and password.
2	Client	Asks the user's name, length and submitted body metrics from VirtuaGym, using the user's username and password and displays this to the user.
3	User	Picks one of the body metrics that is listed and clicks on the delete button associated with it.
4	Client	Sends a request plus the id of the body metric along with the user's username and password to VirtuaGym.
5	VirtuaGym	Deletes the body metric and sends a confirmation back to the Client.
6	Client	Asks the user's name, length and submitted body metrics (which now miss the deleted body metric) from VirtuaGym, using the user's username and password and displays this to the user.

9.3.1.2.3 Scenario Exceptions

Step	Actor	Action Description
2	VirtuaGym	The user's username or password turns out to be invalid.
5	VirtuaGym	The id supplied by the client turns out to be invalid.

9.3.1.3 Insert New Weight Oauth

9.3.1.3.1 Characteristic information

Description	In this scenario the user uses Client 2 (the version that logs in using Open Authorization) in order to insert a new weight.
Precondition	-
Success End	The new weight that has been submitted is now added to VirtuaGym and listed

Condition	at the list of body metrics at the clients.
Failure End Condition	The application has received an error code and error message, which are displayed to the user.

9.3.1.3.2 Main Success Scenario

Step	Actor	Action Description
1	User	Logs in at the Client with a username that doesn't need to match his username at VirtuaGym.
2	Client	Performs the necessary Oauth steps to get access to the user's information at VirtuaGym, which it then asks and displays to the user.
3	User	Inserts a value and submits a new weight to VirtuaGym through the client.
4	Client	Sends a request plus the required data for the weight to VirtuaGym, using the user's access token.
5	VirtuaGym	Adds the weight and sends a confirmation back to the Client.
6	Client	Asks the user's name, length and submitted body metrics (which now include the newly submitted weight) from VirtuaGym, using the user's access token and displays this to the user.

9.3.1.3.3 Scenario Exceptions

Step	Actor	Action Description
2	VirtuaGym	The user's username or password turns out to be invalid.
5	VirtuaGym	The value of the weight turns out to be invalid.

9.3.1.4 Delete Weight Oauth

9.3.1.4.1 Characteristic information

Description	In this scenario the user uses Client 2 (the version that logs in using Open Authorization) in order to delete a body metric.
Precondition	-
Success End Condition	The body metric in question is deleted from VirtuaGym and disappeared at the list of body metrics at the clients.
Failure End Condition	The application has received an error code and error message, which are displayed to the user.

9.3.1.4.2 Main Success Scenario

Step	Actor	Action Description
1	User	Logs in at the Client with a username that doesn't need to match his username at VirtuaGym.
2	Client	Performs the necessary Oauth steps to get access to the user's information at VirtuaGym, which it then asks and displays to the user.
3	User	Picks one of the body metrics that is listed and clicks on the delete button associated with it.
4	Client	Sends a request plus the id of the body metric to VirtuaGym, using the user's access token.
5	VirtuaGym	Deletes the body metric and sends a confirmation back to the Client.
6	Client	Asks the user's name, length and submitted body metrics (which now miss the deleted body metric) from VirtuaGym, using the user's access token and displays this to the user.

9.3.1.4.3 Scenario Exceptions

Step	Actor	Action Description
2	VirtuaGym	The user's username or password turns out to be invalid.
5	VirtuaGym	The id supplied by the client turns out to be invalid.

9.3.2 Version 2

9.3.2.1 Edit Config File

9.3.2.1.1 Characteristic information

Description	In this scenario, the user accesses the configuration page and changes the functions that will be provided by the Proof of Concept Client Application.
Precondition	-
Success End Condition	The functions that will be provided by the Proof of Concept Client Application have been changed.
Failure End Condition	The functions that will be provided by the Proof of Concept Client Application have not been changed.

9.3.2.1.2 Main Success Scenario

Step	Actor	Action Description
1	User	Accesses the Config Page.
2	Config Page	Reads the Configuration XML File and displays a list of check-boxes for each function of the API, auto-checking the values as dictated by the Configuration XML File.
3	User	Checks one or more of the check boxes on or off and submits.
4	Config Page	Writes a new XML file based on the check boxes that were checked on as the user submitted.

9.3.2.1.3 Scenario Exceptions

Step	Actor	Action Description
3	User	Refuses to check or uncheck any of the checkboxes.

9.3.2.2 Insert Something Into VirtuaGym

9.3.2.2.1 Characteristic information

Description	In this scenario the user uses the Proof of Concept Client Application (known in this section as Client) in order to insert something into VirtuaGym.
Precondition	- The "something" that the user wants to insert needs to be something he is authenticated to do through the API. - The Configuration XML File needs to at least one enabled parameter for PUT operations.
Success End Condition	Something has been added to the VirtuaGym database.
Failure End Condition	The application has received an error code and error message, which are displayed to the user.

9.3.2.2.2 Main Success Scenario

Step	Actor	Action Description
1	User	Logs in at the Client, using either Basic HTTP Authentication or Open

		Authorization.
2	Client	Reads the Configuration XML File and in case the user wants to GET something, it requests this data from VirtuaGym and displays it to the user, along with the other forms that were marked at the Configuration XML File.
3	User	Fills the values he wants to insert in the appropriate text fields and submits.
4	Client	Sends this data to VirtuaGym in a PUT HTTP request.
5	VirtuaGym	Adds the values in question to VirtuaGym and sends a confirmation back to the Client.
6	Client	In case the user wanted to GET something, it requests this data from VirtuaGym and displays it to the user, along with the other forms that were marked at the Configuration XML File.

9.3.2.2.3 Scenario Exceptions

Step	Actor	Action Description
2	VirtuaGym	Something went wrong authenticating.
5	VirtuaGym	The values supplied by the user turn out to be invalid.

9.3.2.3 Delete Something From VirtuaGym

9.3.2.3.1 Characteristic information

Description	In this scenario the user uses the Proof of Concept Client Application (known in this section as Client) in order to delete something from VirtuaGym.
Precondition	- The “something” that the user wants to delete needs to be something he is authenticated to delete through the API. - The Configuration XML File needs to at least one enabled parameter for DELETE operations.
Success End Condition	Something has been deleted from the VirtuaGym database.
Failure End Condition	The application has received an error code and error message, which are displayed to the user.

9.3.2.3.2 Main Success Scenario

Step	Actor	Action Description
1	User	Logs in at the Client, using either Basic HTTP Authentication or Open Authorization.
2	Client	Reads the Configuration XML File and in case the user wants to GET something, it requests this data from VirtuaGym and displays it to the user, along with the other forms that were marked at the Configuration XML File.
3	User	Picks the value he wants to delete submits.
4	Client	Sends the id of the value that needs to be deleted to VirtuaGym in a DELETE HTTP request.
5	VirtuaGym	Delete the entry in question from VirtuaGym and sends a confirmation back to the Client.
6	Client	In case the user wanted to GET something, it requests this data from VirtuaGym and displays it to the user, along with the other forms that were marked at the Configuration XML File.

9.3.2.3.3 Scenario Exceptions

Step	Actor	Action Description
2	VirtuaGym	Something went wrong authenticating.

5	VirtuaGym	The ID of the entry that needs to be deleted turns out to be invalid.

10 Appendix B: Scenarios

10.1 MoSCoW List

Out of all the functions of VirtuaGym, we made a MoSCoW[64] list, in order to detail our priorities in the functionalities that are to be implemented. The following table shows these decisions, along with the motivation for them.

/user/	Must Have
The user page allows an application to retrieve information about the currently logged in user. This is very important, because it will allow the user access to his own data, such as his length, avatar, et cetera.	
/users/<userid>/	Could Have
This allows users to view the user data of other users. This functionality isn't so important due to the limited amount of scenarios in which you would want to ask the data of a completely different user. This is only useful for very specific scenarios as friends data and high scores.	
/bodymetrics/	Must Have
This url allows the user access to his own body metrics, as in his weight, waist size, fat percentage, virtual age, et cetera over time. (Authentication is required for this page, so there is no need to supply a user id if the user wants to look up his own data). Since this belongs to the core of the VirtuaGym platform and there are a lot of possibilities to use it (think of weight scales and other measuring devices), it is a must have.	
/bodymetrics/<userid>/	Could Have
This allows you to view the bodymetrics of another user. Again, this only has a limited use and therefore isn't that important.	
/activities/	Must Have
An activity is submitted by a user, and details what kind of activity it was (sports, running, et cetera), along with the calories burned in the process. This is especially a good functionality for mobile devices such as pedometers and other applications that track a user's fitness events, so it's a Must Have.	
/activities/<userid>/	Could Have
This lets you access the activities of other users. There is a limited use in this, and therefore it isn't the most important part of this list.	
/sports/	Won't Have
In VirtuaGym users can keep track of a list of sports that they practice. This is such a trivial option that it's not worth it to implement.	
/clubs/	Could Have
Users can also keep track of a list of clubs that they exercise at. Could be useful to check whether there are more VirtuaGym users at the same gym, but its use is very limited.	
/goals/	Should Have
This allows you to view and add fitness goals, like when they're trying to reach a certain goal. Specialized fitness equipment can use this feature to plan out a training schedule, and viewing the progress on your goals can be very motivating.	
/goals/<userid>/	Could Have
This allows you to view the goals of other people. There is hardly any use for this, other than cheering each other on at the gym, or checking out the progress of your friends.	
/achievements/	Could Have
Whenever a user has reached his goal, he gets an achievement. It could be useful for him to view his own achievements, but the use for this is limited.	

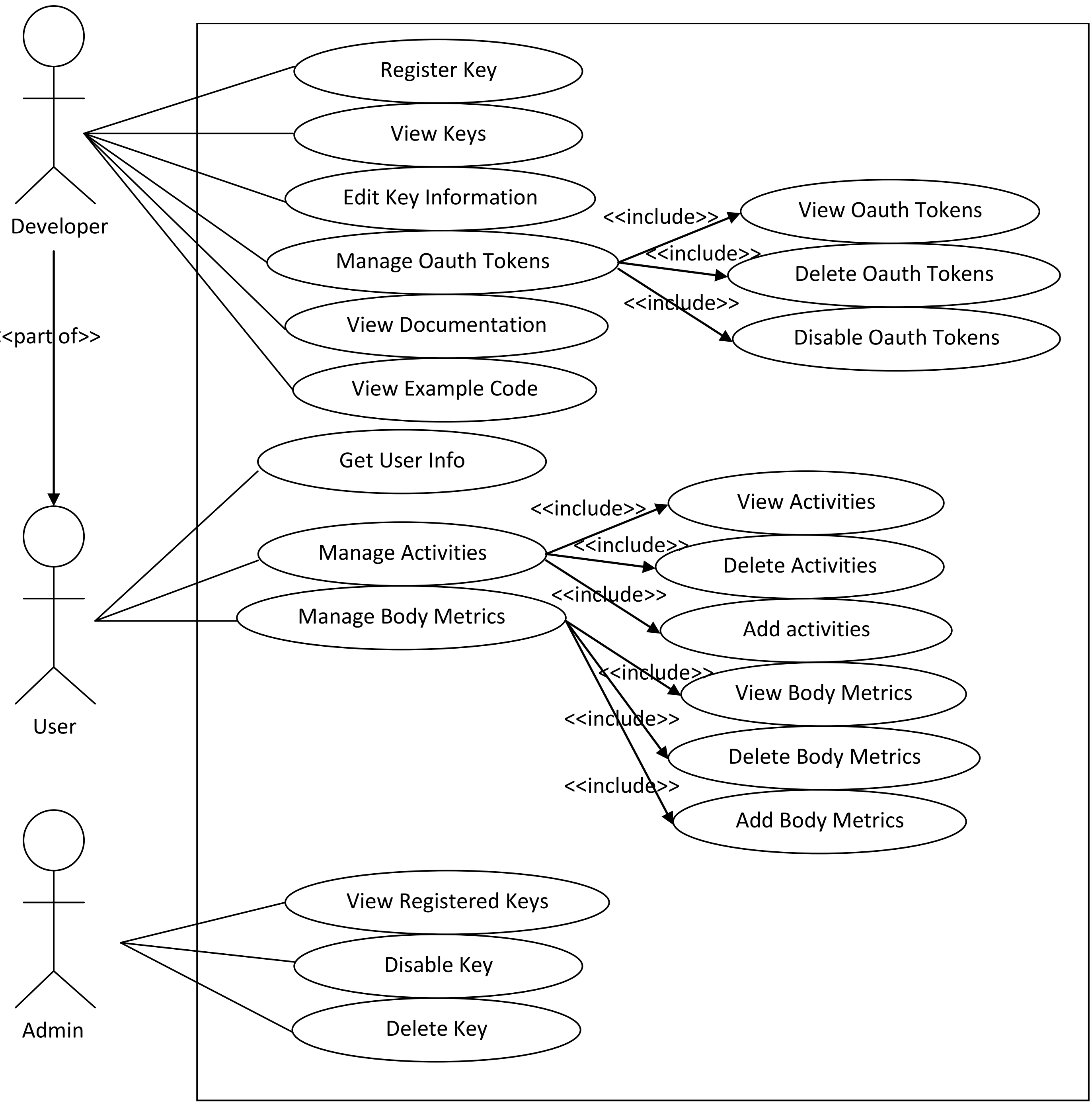
[64] http://en.wikipedia.org/wiki/MoSCoW_Method

/achievements/<userid>/	Could Have
Again this is functionality that is nice to have: to be able to view the achievements of others (like your friends), but the use is limited.	
/friends/	Could Have
Out of all the could haves this is the most important: being able to view who your friends are, but VirtuaGym prefers to focus on the fitness elements, not the community elements.	
/friends/<userid>/	Could Have
With this you're able to view the friends of your friends. There are hardly any practical uses for this and therefore it's a could have.	
/visitormessages/ /visitormessages/<userid> /groups/<groupid> /blogs/ /blogs/user/<userid> /blogs/groups/<groupid> /blogcomments/ /blogcomments/user/<userid> /blogcomments/groups/<groupid> /polls/<groupid> /photos/ /photos/user/<userid> /photos/group/<groupid> /group_memberships/ /group_memberships/<userid> /videos/ /private_messages/	Won't Haves
These functionalities are all part of VirtuaGym's online community: being able to post and view visitor messages, blogs, photos, comments on blogs, user groups, the members of these groups, videos and private messages are nice, but the focus of the API is the fitness part of VirtuaGym, not the community. That's why these features will not be included in the API.	

10.2 Use Cases

This section will outline the Use Cases that were made for the API. Here you'll find the use cases explained, while they are detailed in Appendix A, at subsection 2, which describes the scenarios for these use cases.

The following diagram contains the functionality of the API, where the user functions are based on the Must Haves that were specified in the **Fout! Verwijzingsbron niet gevonden.** at the section above. The sub-sections below will explain the contents of the diagram. A detailed description of the scenarios that belong to these use cases is provided in Appendix A, subsection 2.



10.2.1 User functions

A "user" is someone who uses the API in order to get to interact with his data.

The user three main functions should be getting access to his user information, managing his activities and managing his body metrics. With his user information, he should only be able to use the API to view it, not edit or delete it. With managing his activities, we mean that he should be able to add new activities with the API, but also view them, and delete an activity. The same goes for body metrics: the user must be able to use the API in order to view, insert and delete his body metrics.

10.2.2 Developer functions

A developer is someone who develops applications, websites or other mediums that let users interact with their data.

In order to get access to the API, they need to have an API key, this is done by registering for a key. At any time, developers must also be able to view their keys and edit the data that they entered with their keys (their e-mail, phone number, application name and application url). They must also be able to view the documentation of the API, as well as a page with a few very simple sample codes to help them get started.

If they end up deciding to use Open Authorization as their authentication scheme, then developers must also be able to view the tokens that are registered with them, in order to prevent abuse when it does happen, in which case he can disable or even entirely delete these tokens.

10.2.3 Admin functions

An admin is a member of the VirtuaGym staff who manages the keys that have been registered at the API. In order to do this, he should be able to view all of the registered keys, and he should be able to disable them and deny access or even delete them in the case of abuse.

10.3 Interface Description

The API will consist out of a number of urls. For example:

```
http://virtuagym.com/api/user
http://virtuagym.com/api/bodymetrics
http://virtuagym.com/api/activities
```

This section will describe all of these urls, what you can do with them and exactly what kind of information you can exchange with them, based on GET, POST, PUT and DELETE requests.

Note that the urls as presented lack the authentication scheme. The API will allow access to both Basic HTTP Authentication and Open Authorization. For Open Authorization, you need to complete the open authorization handshake before you can access these urls. For Basic HTTP Authentication you need to include your username, password and API key in the url, as in the following way:

```
http://username:password@www.virtuagym.com/api/user?api_key=apikey
```

Outputs are also always provided in either XML or JSON. In the sections below they will be represented as simple lists. So for example when you have the following list for the entry:

- Option 1
- Option 2
- Option 3

This means that every entry that is returned will have these three options as parameters. The return of a GET-call will look like this in xml, in which the statuscode and statusmessage show whether everything went okay:

```
<?xml version="1.0" encoding="UTF-8" ?>
<reply>
 <statuscode>200</statuscode>
 <statusmessage>Everything OK</statusmessage>
```

```
 <entry>
  <option1>value1</option1>
  <option2>value2</option2>
  <option3>value3</option3>
 </entry>
 <entry>
  <option1>value4</option1>
  <option2>value5</option2>
  <option3>value6</option3>
 </entry>
<entry>
  <option1>value7</option1>
  <option2>value8</option2>
  <option3>value9</option3>
 </entry>
</reply>
```

And the following representation in JSON:

```
{
   "statuscode":"200",
   "statusmessage":"Everything OK",
   "entry":[
      {
         "option1":"value1",
         "option2":"value2",
         "option3":"value3",
      },
      {
         "option1":"value4",
         "option2":"value5",
         "option3":"value6",
      },
      {
         "option1":"value7",
         "option2":"value8",
         "option3":"value9",
      },
   ]
}
```

10.3.1 User

10.3.1.1 Description

Here the user is able to retrieve his own data, for example his username, the url to his avatar and his language. For now it won't be able for the user to edit his data from the outside: it falls beyond the primary requirements of the API.

10.3.1.2 Url

```
http://www.virtuagym.com/api/user
```

10.3.1.3 Values

- id — the id of the user
- username — the user's username
- name — the user's real name
- gender

- age
- length
- length_unit   the unit of the user's length. This is standard in centimeters.
- avatar_url   the url to the avatar of the user.
- birthday
- website
- pro   whether or not the user is a pro member of VirtuaGym
- language   the language in which the user views VirtuaGym

10.3.1.4 Possible Operations

- GET   Retrieves the user's information

10.3.1.5 Parameters

- GET: none

10.3.2 Bodymetrics

10.3.2.1 Description

Here the user can view, add and delete body metrics entries, such as weight, fat percentage or waist size.

10.3.2.2 Url

```
http://www.virtuagym.com/api/bodymetrics
```

10.3.2.3 Values

- id   the id of the body metrics entry.
- type   the type of the bodymetric. Currently the user's weight, bmi, fat percentage, waist, virtual age, fitscore, number of crunches, number of lunches, number of pushups and number of pushups on your knees.
- value
- unit   the unit of the bodymetric, or empty if it doesn't have one.
- date   the date at which the body metric was measured.

10.3.2.4 Possible Operations

- GET   Retrieves all of the bodymetrics of the user.
- PUT   Insert a new bodymetric entry for the user.
- DELETE: Delete a bodymetric entry of the user.

10.3.2.5 Parameters

- GET: none
- PUT: type, value (optional: date)
- DELETE: id (the id of the body metric)

10.3.3 Activities

10.3.3.1 Description

Here the user can add, delete or view activities. For example he can submit that he spent a certain amount of time, practicing a certain sport like baseball. The amount of calories he burned are then calculated at VirtuaGym. You can't just submit anything: in order to view which exercises or sports you can submit, you need to access the exercises page on the API, which contains a full list of this.

10.3.3.2 Url

```
http://www.virtuagym.com/api/activities
```

10.3.3.3 Values

- id — the id of the activity.
- exercise_id — out of the list of sports supported by VirtuaGym. Go to the exercises page in order to find out which ones are supported.
- date
- duration
- duration_unit — the unit of the duration. Seconds.
- distance — only for specific activities as running or cycling; otherwise empty.
- distance_unit — the unit of the distance. Kilometers.
- speed — only for specific activities as running or cycling; otherwise empty.
- speed_unit — the unit of the speed. Kilometers per hour.
- burned_calories
- burned_calories_unit — the unit of the burned calories. Kilocalories.
- notes
- intensity — there's a special scale that determines the intensity of your workout
- intensity_unit — the unit for the intensity. Kcal/kg/hour
- share — whether or not you want to publish this on network sites as twitter, hyves and facebook.

10.3.3.4 Possible Operations

- GET — Retrieves all of the user's activities.
- PUT — Add a new activity for the user.
- DELETE: Delete an activity of the user.

10.3.3.5 Parameters

- GET: none
- PUT: exercise_id, duration (optional: date, distance, speed, burned_calories, notes, intensity, share).
- DELETE: id

10.3.4 Exercises

10.3.4.1 Description

In order for you to submit an activity, you will need to find out its id. This page provides you with a full list of all of the exercises and sports that can be submitted as an activity.

10.3.4.2 Url

```
http://www.virtuagym.com/api/exercises
```

10.3.4.3 Values

- id — the id of the exercise.

- name   the name of the exercise
- met   the intensity of the exercise in kcal/kg/hour
- distance   a boolean number that indicates whether or not you can submit distances for this exercise.

10.3.4.4 Possible Operations

- GET   Retrieves all of the possible exercises that a user can submit for his activities.

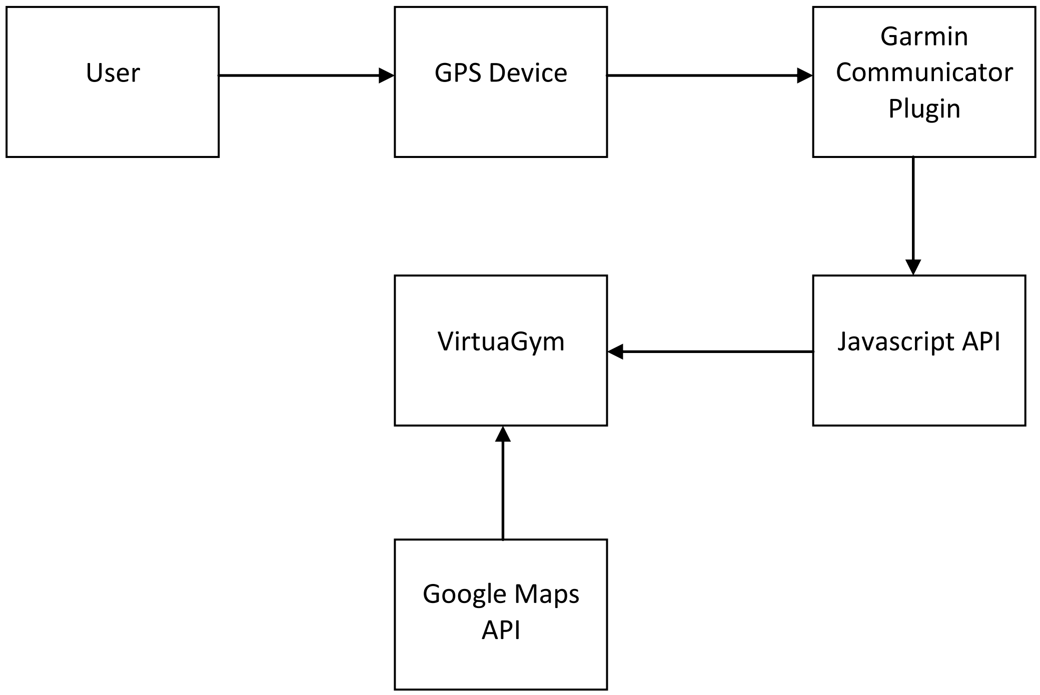
10.3.4.5 Parameters

- GET: none

11 Appendix C: Google Maps Integration

11.1 Garmin

This section will now outline an concrete approach for VirtuaGym to connect to fitness equipment. It's relatively easy to implement, usable for the people who are going to use it, and accessible for anyone with a Garmin device. Globally, the interaction scheme looks as follows:



The user is the VirtuaGym user, who has the possession of a GPS device. This device can be any Garmin Device; a full list of all of the devices supported can be found at this foot note.[65]

The key is that it should be able to support the so-called "Garmin Communicator Plug-in"[66]. This is a browser plug-in that works for both Internet Explorer and Mozilla Firefox, and it can communicate directly with Garmin Devices, without the need of having to take some kind of detour using Garmin's other software.

[65] http://developer.garmin.com/web-device/garmin-communicator-plugin/device-support-matrix/
[66] http://www8.garmin.com/products/communicator/

Figure 7: the garmin communicator plugin and its usage instructions.

For a website to communicate with the Garmin Communicator Plug-in, you need to use its Javascript API[67]. More specifically, one of its methods which can export the data in your GPS device to a GPX-file, which as mentioned above is a widely adopted standard. It's also written entirely in XML, so you can just read out its data with any XML-library. From there, VirtuaGym should be able to easily calculate the distance that was travelled and automatically add this kind of activity to its fitness plan.

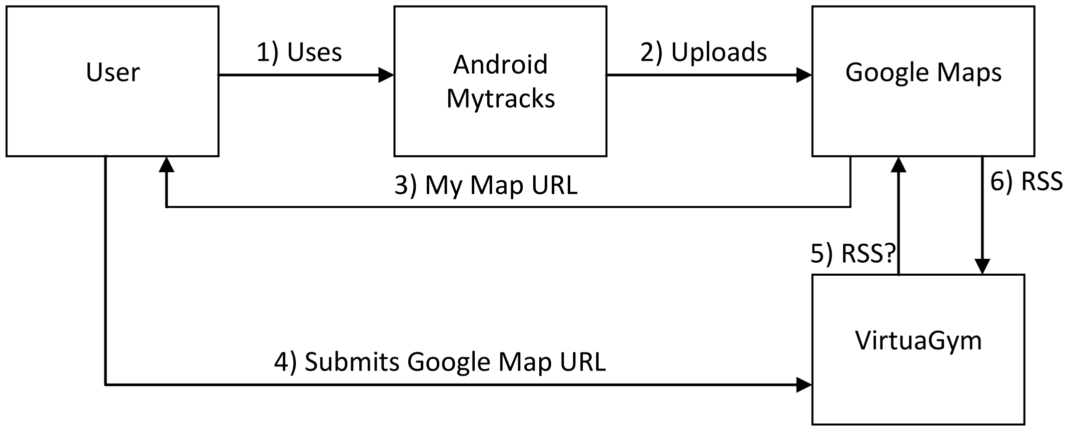
On top of that, it should also be possible to use Google Maps here. VirtuaGym can store the uploaded .GPX-files of the routes that the user has travelled. Whether this is done with a map for each route that has been travelled, or put all routes together on one map (which will cause a lot of lag with a lot of data, though), you can use Google Maps' API[68] to embed those GPX-files onto the map of the actual route you travelled. This can again be done with simple Javascript. An example of an application that successfully added GPX data on a Google Map is Where's The Path[69]. Simply studying its classes should be enough to show how to do this efficiently and correctly.

The scenarios for this approach can be found at Appendix A, at section 9.1.1.

11.2 MyTracks

MyTracks is an Android Application from Google[70]. It enables you to view GPS data of any track you made (whether it's running, cycling, sailing or anything else) and view its statistics. Users can also very easily upload this to Google Maps, which is also interesting for VirtuaGym. The process of submitting maps to VirtuaGym so that they can be viewed and added to the Fitplan requires a bit more effort from the user compared to when they use the Garmin Communicator Plug-in, but it's still a very interesting option to add, and it will also reach people who don't have Garmin devices. The global interaction scheme for submitting a MyTracks Track looks as follows:

[67] http://developer.garmin.com/web/communicator-api/jsdoc/index.html
[68] http://code.google.com/intl/nl/apis/maps/documentation/javascript/basics.html
[69] http://wtp2.appspot.com/wheresthepath.htm
[70] http://mytracks.appspot.com/



The steps are numbered by time, so the scheme starts with 1) and ends with 6). The following happens in these steps:

1. The user first creates a track with MyTracks. He makes some kind of tour by foot, bicycle, etc, and records this using the GPS functionality of MyTracks.
2. When he's done with his workout, the user uploads his track to Google Maps. MyTracks offers an easy way to do this automatically.
3. The user then goes to this map at 'Google My Maps'. There, he clicks on "Link", and copies the url in the top text box. This is a direct link to that map.
4. The user then goes to VirtuaGym, and submits this url at the page at which you can submit new tracks.
5. VirtuaGym then takes this link, and converts it into a link to the map's RSS url. Google's URLs are so created that it's very easy to do this: you just need to replace the map id on an already existing RSS url.
6. Google Maps then sends the RSS of that map back. This is important because we can now calculate the total distance that the user travelled based on the coordinates that are logged in the RSS response, which can then be easily added as a new activity to Virtuagym's fitness plan. MyTracks offers the option of including summary pointers in its tracks, uploaded to Google, but this however is not mandatory.

And at the same time, these maps can be very easily embedded onto a page with just one line of code, using a simple iframe. The advantage of this method over the one discussed at the previous section (using the Garmin Communicator Plugin) is that you don't even need to store GPX-files on your server for it to work. The disadvantage is that the user does need to do a few more things that unfortunately can't be automated. The following sub-sections will formally define these scenarios that were just discussed. The scenarios for this approach can be found at Appendix A, at section 8.1.2.

12 Appendix D: Sources of Fitness Data for VirtuaGym

CSAFE[71]

A communication standard to communicate to home trainers, cross trainers and treadmills. Equipment is controlled through various states, and at the end the data of each work-out is submitted. This data consists out of values as the workout duration, burned calories, speed, pace, cadence, MET and heart rate, along with equipment information.

iFit Live[72]

Used to communicate to equipment of ICON Health and Fitness. We could not find any online documentation, so it is not known what kind of information it supports.

Garmin Connect[73]

Is used to connect to various Garmin equipments such as GPSs and Heart rate monitors. For GPSs, individual points of running and cycling tracks are submitted, but also the total track length, time, and speed.

Mytracks[74]

Mytracks is an application for the android that tracks your running and cycling tracks. To the outside, the individual points of each track are submitted, along with the distance, time, time in which you were moving, your average and maximum speed and elevation data (min, max, difference and road grades).

Polar Device Interface Toolkit[75]

Another brand of mobile devices including heart rate monitors and GPSs. A wide variety of data is supported through this toolkit, but for VirtuaGym the most important are the total distance, average and maximum speed, time and energy consumption. Data is submitted for individual exercises, but also in weekly or daily totals.

Withings[76]

Withings manufactures scales. Their API allows you to communicate with the data of these scales. It stores weight, but also height and BMIs. Every measurement is stored separately for each date, which can then be accessed through this API.

Overall, the data that is the most useful for VirtuaGym is the kind of data that can be directly submitted to its website. For body metrics, this means weight, BML, waist size, fat percentage and your virtual age. For activities, this means the amount of calories that was burned (based on an estimation, of course), MET[77] value and duration, along with speed and distance for the activities that are applicable for it. At the moment, you also aren’t able to submit activities that aren’t in VirtuaGym’s database. Whether this will change in the future is not certain.

[71] http://www.fitlinxx.com/CSAFE/
[72] http://www.ifit.com/
[73] http://www8.garmin.com/intosports/antplus.html
[74] http://mytracks.appspot.com/
[75] http://www.polar.fi/en/support/downloads/device_interface_toolkit
[76] http://www.withings.com/nl/api/weegschaal
[77] http://en.wikipedia.org/wiki/Metabolic_equivalent