Back-end development of mobile application for the collection of dietary data

Fredrik Bäck

November 5, 2012
Master’s Thesis in Interaction and Design, 30 credits
Supervisor at UmU: Åsa Holmner Rocklöv
Examiner: Håkan Gulliksson

UMEA UNIVERSITY
DEPARTMENT OF APPLIED PHYSICS AND ELECTRONICS
SE-901 87 UMEA
SWEDEN
Abstract

Smartphones are used by incredibly many people, and in 2011 there where a total of 491.4 million units soled worldwide. Smartphones are known as advanced phones, or personal digital assistants (PDAs). This makes a relevant technique for performing dietary studies when the test patients are on the move. This thesis shows how to create a back end environment for an Android application with existing techniques linked together, using MS SQL database, Visual Studio Web Service, web pages and C# classes and ASP.NET security. The back end development is used in a dietary study on Gothenburg University, but could be applied on many similar back end projects using databases and server development. Techniques used in this thesis are: REST (Representational State Transfer) -client implementation and development inside the Android application, using HTTP methods to set and get information from the server and database, and JSON-format to read and transfer information in an easy and understandable way, both from the Android application and from the database. FileMaker is also used in this project as a third part programme to visualise the information in the database.
## Contents

1 Introduction ................................................. 1
   1.1 Gothenburg and Umeå university ................. 2
   1.2 Statens livsmedelsverk ........................... 2
   1.3 Paper outline .................................. 2

2 Background .................................................. 5
   2.1 PhD thesis project plan by Åsa Svensson .... 5
   2.2 The database from SLV ............................ 6
   2.3 Mats Högdals Android application ............. 6
      2.3.1 Application database ...................... 12

3 Problem Description ....................................... 15
   3.1 Goals ........................................... 15
   3.2 Limitations ...................................... 16

4 Technology and software .................................. 17
   4.1 Limitations in software ......................... 17
      4.1.1 Choosing software for android .......... 17
      4.1.2 Choosing software for the database .... 18
      4.1.3 Choosing software for the server ...... 18
   4.2 Related Work .................................. 20
      4.2.1 Native vs. Mobile Web applications .... 20
      4.2.2 Android techniques ...................... 20
   4.3 Working with android .......................... 20
      4.3.1 REST client ................................ 21
      4.3.2 JSON ....................................... 22
   4.4 Working with server ............................ 23
      4.4.1 Web service ................................ 25
      4.4.2 C# .......................................... 26
      4.4.3 Web pages ................................ 26
      4.4.4 Security .................................. 27
4.5 Working with database ........................................ 27
  4.5.1 Graph databases on social web .................. 29

5 Methodology .................................................. 33
  5.1 Android .................................................. 33
  5.2 Server .................................................... 33
    5.2.1 Web page design .................................. 34
  5.3 Database .................................................. 34
  5.4 FileMaker .................................................. 34
  5.5 User guide .................................................. 34

6 Resulting work .............................................. 35
  6.1 Android .................................................. 35
    6.1.1 Implementing REST-client ......................... 35
    6.1.2 Programming and design ............................ 37
  6.2 Server .................................................... 38
    6.2.1 Web Service ......................................... 39
    6.2.2 Web page ............................................. 41
    6.2.3 Security ............................................. 44
  6.3 Database .................................................. 45
    6.3.1 Database of dietary study ......................... 46
    6.3.2 ODBC ................................................ 47
  6.4 FileMaker .................................................. 47

7 Conclusions .................................................. 51
  7.1 Future work and changes ............................... 51

8 Acknowledgements .......................................... 53

References .................................................... 55

A User guide .................................................. 59
# List of Figures

1.1 Smartphones market in quarter 4, 2010 ........................................ 1
2.1 Start page and registration page. .................................................. 7
2.2 Menu page and statistics page. ..................................................... 8
2.3 Register intake. ........................................................................... 9
2.4 Search food. ............................................................................... 10
2.5 Image of food size on plate. ......................................................... 11
2.6 Register physical activity. ............................................................. 12
2.7 Android application database. ....................................................... 13
4.1 Content in ASP.NET web service. ............................................... 19
4.2 Complete software overview. ....................................................... 19
4.3 JSON Example. ........................................................................... 23
4.4 ASP Framework. ................................................................. 24
4.5 WebForms round trip. ............................................................... 25
4.6 XML request ............................................................................ 25
4.7 Comparison between VB and C#. ............................................... 26
4.8 Security levels. .......................................................................... 27
4.9 Relational database. ................................................................. 28
4.10 Relationship in database. ........................................................... 29
4.11 Graph database theory. .............................................................. 30
4.12 Difference in graph database. ..................................................... 30
4.13 Difference in web, from early usage to todays social web. ............ 31
6.1 REST-client path. ................................................................. 36
6.2 Inside the REST-client. .............................................................. 37
6.3 REST-client process. ............................................................... 38
6.4 Get information from Web Service. ........................................... 39
6.5 Set information to Web Service. ............................................... 40
6.6 Webpages. .............................................................................. 42
6.7 Login webpage. ...................................................................... 42
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.8</td>
<td>Default webpage.</td>
<td>43</td>
</tr>
<tr>
<td>6.9</td>
<td>Admin webpage.</td>
<td>44</td>
</tr>
<tr>
<td>6.10</td>
<td>Webpage security.</td>
<td>45</td>
</tr>
<tr>
<td>6.11</td>
<td>Authentication process.</td>
<td>45</td>
</tr>
<tr>
<td>6.12</td>
<td>Database information line.</td>
<td>46</td>
</tr>
<tr>
<td>6.13</td>
<td>Project database.</td>
<td>47</td>
</tr>
<tr>
<td>6.14</td>
<td>Filemaker user page.</td>
<td>49</td>
</tr>
<tr>
<td>6.15</td>
<td>Filemaker nutrition per meal page.</td>
<td>50</td>
</tr>
</tbody>
</table>
List of Tables

4.1 Native development. ................................................. 20

6.1 Ordinary database table structure. .......................... 41
6.2 Transponded table structure. ................................. 41
Chapter 1

Introduction

The focus of this thesis has been the back-end development of a pre-developed phone application using an Android operating system. The application is a dietary tracker which will be used in a study aiming to teach children and teenagers to understand their dietary habits.

The development of Internet and smartphones offers an unprecedented opportunity to perform dietary studies. The possibilities given by the smartphones are many, as they make it easy to have high definition graphic screens right in the pocket of its users, and enough capacity to perform advanced technical computing such as video or image capitation, downloading from Wifi or surfing on the Internet.

Android is the largest selling operating system in the forth quarter of 2010 according to Canalys, making it the preferred choice.

![Pie chart showing the smartphones market in quarter 4, 2010](image1.png)

Figure 1.1: Chart showing the smartphones market in quarter 4, 2010[9].
1.1 Gothenburg and Umeå university

Göteborgs university (GU) and The Department of Food, Nutrition and Sport Science performs research and education with the ambition to lay a ground for good health and sustainable development.

Areas of focus for food and nutrition/home economics, includes eating habits, meals, what kind of impact food has on public health and leadership/organisation in restaurants and larger households. The focus on sport science is broad and ranges from exercise and health promotion to elite sports. [21].

Umeå University (UmU) and The Department of Food and Nutrition have two mayor study programmes, the dietetics programme and the food and nutrition/catering management programme.

The department’s general focus is the health and dietary habits of specific groups, with a particular emphasis on children´s health[24].

1.2 Statens livsmedelsverk

The swedish national food agency, named statens livsmedelsverk, SLV, is the central administrative authority for matters concerning food in Sweden. They are responsible for dietary recommendations to the general population, but also specific groups who might have food allergies. SLV:s leading words in there work is safe food, fair practise in the food trade and healthy dietary habits.

SLV is also active in matters concerning trade areas referring food, which help the consumers to rely on the labelling in many products. Labels are, for example, used to ensure food composition, quality and origin of the food[32].

1.3 Paper outline

Chapter two - Background

This thesis work is part of a research project on dietary studies which is part of Åsa Svensson´s Phd studies. The background provides a introduction to her project about dietary studies using validation, reliability and user friendliness as key aspects. The database from SLV are overlooked and explained to understand what parts are relevant to the project and Mats Högdahl’s front end development, the user interface, is shown.

Chapter three - Problem description

This chapter describes the main task in the project, and describes the goals of the project from a technical viewpoint, discussing what is needed for the server, database and android application to co-work. The project limitations are defined to show what is possible in the project.

Chapter four - Technology and software

This chapter describes the techniques used in this project, from server control to database abstraction. Limitations in software is explained to show why specific software choices where made, and Related work show why some techniques are useful in this project. In addition,
1.3. Paper outline

techniques such as JSON-file format and REST-clients used in the android application as well as the server is explained.

Chapter five - Methodology

Methodology in this project work where about choosing which computer programme, code technique or web page should be used. Many of the choices made are based upon observation and relevance to the project. Given some key aspects, the rest are following the similar path to make the project work smoothly.

Chapter six - Resulting work

This chapter explains what has been made, and how the parts are working together. It shows how to implement the REST-client specific to this project, and how it is combined with the android application to create a connection to the server. It shows the different C# classes used in the android application and how these are used to help the REST-client. The parts used in the Web Service inside the server are explained to show how to connect, receive and guard information sent from the android application and database, and vice versa. Also The database and Filemaker are reviewed, and explained.

Chapter two - Conclusions

Conclusion and future work are explained and analysed in this chapter, given an idea of how to proceed and learn from this project. The most common aspects of future work are the ability to update and optimise smartphone OS, programming code or computer programmes.
Chapter 2

Background

This thesis is a part of a research project at the Department of food, nutrition and sport science at GU, aiming to perform dietary studies on children using mobile technology. Specifically, the outcome of the development performed in this thesis will be applied and evaluated within the PhD project of Åsa Svensson. A front-end application has been designed and developed by Mats Högdal, whereas this thesis was limited to the back-end component allowing the application to retrieve data on specific food type and store data in a database from SLV.

The development of smartphones the resent years has made it simpler to use advanced techniques, which leads to the possibility to perform advanced studies on test persons outside of a usual laboratory or test environment, and the test persons could perform the test on their own terms within the tests conduct rules. The capability of smartphones and their ability to use external devices through cable, Bluetooth, Wi-Fi or some other technology gives the test participant the possibility to perform advanced measurements on blood pressure[35] and sleep rhythm.

Performing dietary studies with traditional methods, such as food diaries, questionnaires and interviews are somewhat complicated and time consuming, both for the test participant and the scientist. Moreover, these methods are relatively unreliable, since they all rely on the participant’s memory, truthfulness and ability to judge the correct amounts of food eaten[15].

2.1 PhD thesis project plan by Åsa Svensson

The PhD thesis main subject is an evaluation on different dietary study methods concerning minors. One of the aims is to develop a new dietary study method based on a smartphone and evaluate its accuracy, reliability and user friendliness[36].

The best ways to perform dietary assessments in the age group 11-16 is to use a dietary history interview. From the age 4-11 years old, the preferred approach is using a 24 hour recall in three different sessions, using the parent as the corresponding test participant[8].

The PhD thesis is a collection of several dietary studies performed in Sweden, where goal 1-2 are dietary studies performed at Umeå University, and part 3-4 are studies using valuation from a newly developed smartphone technique.

In part 3 and 4, the goal is to use new smartphone technology to monitor the intake of food from minors. Previous methods using smartphones already exist, though not in Sweden and not adapted to Swedish households and food habits[29]. That makes the need
of a new smartphone application necessary, and the development of this application is the basis of a master thesis project.

### 2.2 The database from SLV

The database from SLV is the database, which the smartphone application and the backend application use to get the nutrition values from different food types. Several different tables, holding the values from SLV’s own dietary studies, represent the database. The whole database is not needed when working with the applications, and it contains values that are not interesting. The main tables used are explained in the text below.

**Food table (Livsmedel):** This table contains all the food types that are used in the study. There are 1909 different food items, given a variety from carrots to soda. All foods are suited for a Swedish diet.

**Nutritional values table (Näringsvärden):** This table is the largest one, containing 101177 rows of nutritional values. The large size of the table is because of the many different kinds of nutritional values, 53 different values to be exact. 53 nutritional values times 1909 different food items equals 101177 rows of nutrition. All the nutritional values are listed in food weight of 100 gram.

**Nutrients table (Näringsämnen):** In this table the names of the 53 different nutritional values are listed. The table above (nutritional values table) only contains a number linked to this table for every name of the nutrient. This table also contains a description and a shortening for every nutrient.

**Main group table (Livsmedelgrupp):** All the food items are included in different main groups, making it possible to keep a record if the test participants are eating from many food groups or from one or two specific.

**Minor tables** These tables are used to make it easier for the test participant to choose a food, not knowing exactly what the food type is from the beginning.

- Cooking method (Matlagningsmetod): E.g. cooked, baked, fried and dried.
- Dish type (Maträttstyp): E.g. pizza, soup, salad and stewage.
- Food category (Livsmedelskategori): E.g. oil, rice, white bread, drinks and sweeteners.
- Dishes with (Rätter med): E.g. Pasta dishes, mushroom dishes and meat dishes.

### 2.3 Mats Högdals Android application

Mats Högdal has made the front-end development of this project, meaning the GUI (Graphical user interface), user tests, Android development and graphical guidelines. The application contains several features, being able to register your daily intake of food, and the possibility to view statistics over the nutrition values from a specific day.
Figure 2.1: Illustration of the graphical user interface. Start page (a) allows a new user to register or an old user to log in, and registration page (b) where personal information can be entered upon registration.
Figure 2.2: In the menu page (a), you are able to register the day, or look at saved days. The saved days (b) page is showing meals and physical activity saved from earlier days. It is also possible to choose statistics (c)(d), showing how much energy etc. your intake has been.
2.3. Mats Högdals Android application

(a) Register intake
(b) Choose date
(c) Choose meal
(d) Choose time

Figure 2.3: When registering food intake (a) the test participant could choose between date (b), meal (c) and time (d).
Figure 2.4: When the test participant needs to find some kind of food (a), he or she could either search by typing the name (b), or searching through categories of food (c). When the food is found, it is saved on a specific date (d).
Figure 2.5: When finding the food the test participant chose, it is given the choice of different sizes of food on a plate to estimate how much food they have eaten.
2.3.1 Application database

All of the functions in the application is relying on the internal database inside the android application, using some parts from the SLV database. When saving personal setting, meals or physical activities, it is placed in specific tables created inside the application. The food, pictures of the plates with food, and categories are from SLV used to simplify for the test participant. All of this information is then saved and sent to the server, saving the information for the dietary study.
Figure 2.7: Android application database.
Chapter 3

Problem Description

This project is part of a larger development involving an android application, divided into front-end development and back-end development. The task is to develop the back-end part of this application. For the application to work it is necessary to use a back-end developed part that is able to contact and send information back and fourth to a database, since the database is holding all information used in the nutrition study, and all information that is going to be saved from the study.

To help the database to communicate, a server is necessary, with the capability to communicate between the smartphone and the database over the Internet.

Also a third part programme is used, to show the database information in an accessible and readable way, and not only as values from a database column.

3.1 Goals

There are some specific goals that need’s to be reached in order for the application to work. All the goals are set in a back-end point of view when developing, with the assumption that the front-end part is fully developed.

These goals are parts used in an opening plan to make the back-end work as smoothly as possible. The goals/parts shouldn’t be to many, making it hard to keep track of what is doing what, e.g. in the source code.

**Application goals:** The application is using android OS, and to write an application in android the code is written in Java[18] and XML[37]. The goal in the application is to make it possible to communicate with the server, being able to send and collect information when needed. It should be able to make a connection using Wi-Fi and if necessary translate the values from the database.

**Server goals:** The main purpose of the server is to connect to the application and database, and act as a translator/delivery agent for the values collected. The server goals are divided into two: the first is that it should be possible to collect information coming from a source over the Internet, being the android application, and make something useful of the information, e.g. placing it into the database or using it to confirm values. The second goal is that the server should be able to send information back to the source brought from the database and sent back over the Internet.

**Database goals:** The database should contain the values given from SLV, and also the new values collected by the application. The database should be made as simple as
possible, using relations that is a common technique in database connections. Relations are the connections between the tables in the database.

**Database information goals:** This programme or application should be used to show the values from the database in a way that is readable in the programme SPSS[19]. The possibility to alter or only show parts of the information in the database should also be given though there is no one with technical education that is going to use the database. This makes it necessary to show the information in an accessible and readable way.

**Download goals:** The goal is to make it possible to distribute the application to a large number of users. The application should either be handed out or downloaded from a webpage or similar.

**Smart and simple code goals:** the goal is to write code that is smart and simple, making it easier to understand and read. When the project has ended, there is no one who is going to continue working with the code as a developer. If someone wants to make updates or change something it shouldn’t be too hard to achieve.

**Security and integrity goals:** All parts of the project should be protected. The applications and the server traffic over the Internet should be encrypted. Also the server connection to the database should be encrypted, since it contains information about private individuals and measurement data from SLV.

### 3.2 Limitations

Limitations of the project are technical knowhow and physical restriction.

**technical knowhow:** The persons leading the project are not experienced in the programmes and techniques used in this back-end project. This makes it hard to know exactly what the limitations in the project should be set at, e.g. what if some programme needs to be changed or updated, what could happen then?

**physical restriction:** The smartphone should be able to use the application without any phone card, making it reliable only to Wi-Fi. This means that the application is restricted to send its information and communicate to the server freely.
Chapter 4

Technology and software

This chapter of the report contains an explanation on why some software where used. There will also be a theoretical study consistent with the goals and what is required from the project.

4.1 Limitations in software

The choice of server and database was limited since the database from SLV was in MS (Microsoft) SQL format. The primary choice fell on using MS Server using ASP (Active Server Pages) [5] as the primary server tool when communicating to Internet and the database. One other option was to use PHP (Hypertext PreProcessor) [30] but it is newly developed to use with MS SQL databases and on MS Servers, making ASP (also developed by Microsoft and used for many years, being well documented) smoother to work with.

Important to note is the software used in this project is not the same software used to control and manage the physical server and database. The programme chosen in the project is used to alter and manage the content of the server and database.

4.1.1 Choosing software for android

Eclipse[11] is a free, well-known development programme, recommended by Android[17], making it the foremost option of software for this project. Also, Mats Högdal has started developing the application in Eclipse, therefor making it an Eclipse project, making it simple to import and export to and from Eclipse.

SVN

Eclipse is foremost used by Java developers, and since android is written in Java, it is a good match combining the two. To help the project SVN[4] was used. SVN is a version control system specifically developed to overcome some limitations in CVS (older version). A version control system keeps track of history and change and makes it possible to step back in time or to track who made previous changes. It allows for collaboration on the same project for developers who are spread across the world.

To use SVN you need to download an appropriate SVN plug-in. An SVN suitable for using in Eclipse are: http://www.eclipse.org/subversive/
4.1.2 Choosing software for the database

Given that the database from SLV where in MS SQL (Microsoft structured query language) format, the obvious choice of main programme for creating the database was Microsoft SQL. Also, it is not possible to use any other SQL programme other then Microsoft own when using a MS SQL database. This means it is possible to get and set values in the database, as with any other database, but it would not be possible to alter the database to the same extent as if you were developing the database from scratch.

ODBC

To make it possible to connect a third party programme to the database, the use of ODBC is needed. ODBC (Open DataBase Connectivity) is used as a separate small programme, making a connection to the database usable from many different applications or operation systems.

The third part programme in this project is used to show what is saved in the database, in a simple and useful way and also making it possible to export the information in specific format.

4.1.3 Choosing software for the server

The choice of software for the server was (for obvious reasons) based on the choice of database language. Thus, to make it easy to connect to the database, the natural choice fell on a software capable of using ASP. ASP is a technique developed to read and write dynamic web pages and application using information from databases. Microsoft programme Visual Studio is used when making Web pages, Web services and Web applications, containing ASP.NET[5], which are a web application framework and the successor of Microsoft ASP.

Web service

To make a connection between the android application and the database, the use of Internet application capable of GET and SET methods to and from the database is important. Creating a web service in visual studio makes this possible, since it contains web methods, web pages, and C#[27] classes. C# is a simple object-oriented programming language with similarities to Java, which was developed by Microsoft to be used within .NET. This is particularly useful when there is a need to perform more complicated, object-oriented programming, which is harder to do using only .NET.
4.1. Limitations in software

Figure 4.1: Content in ASP.NET web service.

Figure 4.2: Complete software overview. The unknown third part programme is FileMaker, but it is not combined in this illustration since it is used as a separate part.
4.2 Related Work

Similar to this project is the master thesis work *Speedflirt for Android, a study of porting a mobile phone application and its effects on user experience* from Erik Jansson and Niclas Jonasson[22] and the master thesis *Supporting communication and collaboration in the process automation industry* from Jonas Brönmark and Mikaela Akerlind[7]. The first thesis is about porting an Android application, and finding suitable techniques for cross-platform development. The thesis brings up information useful to this project, both in terms of techniques and user experience guidelines for Android. The second thesis is examining the use of mobile applications and social web in automation industry, which gives an insight of specific industrial us of applications.

4.2.1 Native vs. Mobile Web applications

Native applications are applications made from the phone’s own platform, making it possible to access the phone’s hardware and GUI (Graphical user interface). The disadvantages when creating native applications is the restriction to that specific platform, meaning a developer need to alternate between programming languages and developing programmes[7].

<table>
<thead>
<tr>
<th>Platform</th>
<th>Language(s)</th>
<th>IDE</th>
<th>OS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>iPhone</td>
<td>Objective-C, C</td>
<td>Xcode</td>
<td>OS X</td>
</tr>
<tr>
<td>Android</td>
<td>Java, C, C++</td>
<td>Eclipse/Netbeans</td>
<td>Windows, Linux, OS X</td>
</tr>
<tr>
<td>Blackberry</td>
<td>Java</td>
<td>Eclipse</td>
<td>Windows</td>
</tr>
<tr>
<td>Symbian</td>
<td>C, C++, QT, Java</td>
<td>Any text editor</td>
<td>Windows, Linux, OS X</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>C#, VB.NET</td>
<td>Visual Studio</td>
<td>Windows</td>
</tr>
<tr>
<td>webOS</td>
<td>HTML, Javascript, CSS</td>
<td>Any text editor</td>
<td>Windows, Linux, OS X</td>
</tr>
</tbody>
</table>

Table 4.1: Native development[22].

Mobile Web applications are the opposite of native applications, meaning they are developed on a separate platform, being able to be used on different native platforms. The application is constructed as a webpage, but it is encapsulated inside a native shell. This means an application could be used on Android, iPhone and Blackberry phones, looking similar and operating in a similar way. The disadvantage with using Mobile Web applications is that the developer doesn’t get access to the phone’s hardware, such as camera, GPS and Bluetooth.

4.2.2 Android techniques

The implementation of REST was an essential part of the work by Jansson and Jonasson, and this work describes how to implement a REST-client suitable for Android development. A REST-client is a client responsible for client-server communication using HTTP calls[22].

4.3 Working with android

Android is an open-source software containing operating system, Application programme interface (API) libraries and key functions for creating smart phone applications. Android applications created by developers, called third-party applications, are written in the same
API as the native applications preinstalled in Android smartphones. This makes it possible to use all Android features, and to create custom made applications suitable for your own purpose[25]. Some examples of useful features are:

- GSM, EDGE and 3G networks for data transfer, enabling you to send or receive data across mobile networks.
- Full multimedia hardware control, including recording with the camera.
- Background application and processes.
- Web browser.

**HTTP-connection**

The possibility to create a native Internet-based application is offered in Android. This means the application is relying on raw data coming from the Internet, e.g. XML or JSON-files. This possibility of creating an Internet-based application is preferred when creating social media application, not needing to go through Androids WebKit. I.e. cutting out the middle hand[34]. Since the application is using raw data from the Internet, the methods used for getting the data are HTTP methods, e.g. GET or POST[33].

**Application internet status**

The possibility to alter, examine or shut down the applications access to Wi-Fi and mobile net (e.g. GPRS, 3G) is useful, since the applications don’t always have the possibility to be connected to the Interned. This is useful when creating methods who need Internet, and the ability to test the application when the Internet connection is lost is vital. It needs to be able to save tasks, and perform these as soon as a reliable connection is set. Android broadcast intents is what describes changes in network connectivity and offers the API used when controlling network settings and connections[25].

**Threads**

In the beginning, every application runs on one main "thread", containing al tasks. A thread is the smallest sequence of programmed instructions that can be managed independently. This means that a simple task, such as downloading a image occupies the process. But when using multiple threads, it is possible to download the image and do other tasks at the same time, since different tasks are performed on different threads. To avoid a main thread to freeze, hang up or even close down the application, the usage of multiple threading is sometimes necessary. The reason to use threading is important, since the main thread don’t perform any other task while the requested task is operational[34].

**4.3.1 REST client**

Representational State Transfer (REST) is a software architecture used by distributed systems. It was originally created by Roy Fielding, who is one of the authors to HyperText Transfer Protocol (HTTP) v. 1 and v. 1.1[13].

REST-client architecture is depending on client and server communication. Clients pass a request to the server, the server process the request and return a response. In REST, the client who is passing the request is only doing it when it is making a transition into a new
state. When the client has passed a request, and is waiting for an answer, the client is said to be in transition. When the client receives the answer, it is ready to initiate a new state.

The name Representational State Transfer is chosen to simulate the transition the client goes through. E.g. when a client requests a webpage it needs to wait for the request to return (the webpage to show) before being able to initiate a new state (change to the next page)[13].

REST-client/server communication is a stateless communication, meaning there is no client information saved on the server. This makes it possible to change the client or server independently, without having to change the mode of communication[13].

RESTful web service

A RESTful web service is a web service using the principles of REST and implemented in HTTP. The client-requested answer from a RESTful server is often in the format of XML, HTML or JSON, and the web service needs to support HTTP methods, e.g. GET, PUT, DELETE and POST[33].

The two most important methods are[22]:

– Retrieve a recourse from the server, HTTP-GET.
– Update a recourse from the server, HTTP-PUT.

When a client is sending a request, it passes the information inside the URI as parameters in the HTTP-header. Example[22]:

– URI: http://www.example.com/login
– Parameters: username=Steve pass=Smith
– Complete request: http://www.example.com/login?username=Steve&pass=Smith

Additional information is sent inside the HTTP body, as metadata (data in data). This metadata could be JSON, XML or another valid Internet media type. The response from the server is then packed inside a HTTP-body (also containing metadata if requested), and returned to the client[22].

The large advantage of using a REST is the ability to request information from a server, e.g. a database server, and pass the information in a format easily read by the client (e.g. JSON or XML).

4.3.2 JSON

JavaScript Object Notation (JSON) is a lightweight text-data interchange format, which is language and platform independent. This makes it easy for both humans to read and for machines to parse and generate[38]. JSON is built on two structures[23]:

– A collection of name/value pairs. This is know as an object in many developing languages.
– An list of values in order. This is know as an array in many developing languages.
4.4 Working with server

The official Internet media type of JSON is application/JSON, and the primary usage of JSON is to send and receive data from a server as an alternative to XML. E.g. suppose an application wants to request something from a web page, if you are using XML the response from the server needs to be parsed and structure by a Document Object Model (DOM), and then used to update the web page. Alternatively, you could use JSON and get a JavaScript Object represented as a string (i.e. a JSON-String) back from the server and use directly as it is[20].

4.4 Working with server

Making use of the server programme, many different parts are used in one web project. As seen in picture 4.1, there are different parts needed, which has to cooperate to make the server work. Some of these parts are more specifically about programming, while some are a question about design, e.g. the web page.

Using the .NET framework when developing the web project is preferred since the framework is highly supported by the database and server, which are coming from the same developer. The .NET framework is made from four different parts, Common Language Runtime, a set of class libraries, a set of programming languages, and the ASP.NET environment [12].
Figure 4.4: ASP Framework.

Common Language Runtime (CLR) makes it possible to use all the programming languages together in one runtime environment. A runtime environment provides a set of operating system services to the executable code. These services take care of details in the low-level operations system, such as physical memory allocation. Previous to the .NET framework, e.g. Visual Basic and C++ had their own runtime environments, making it impossible for one language to work together with another, meaning they used double the amount of memory needed, since they couldn’t share functions. With the introduction of CLR it was possible to interoperate with each other, being able to allocate memory in one language and release it in another language[12].

WebForms

WebForms are used as a foundation in of a Web-based application. A WebForm is used to interact with the user in the application, using different server controls such as text boxes, buttons or check boxes. The WebForm contain two different parts, the user interface (UI) and the programming logic. The UI is the visual part of the WebForm, containing HTML. This is specified in a file ending with .aspx. The programming logic part, containing the code performing the function of the form, is used as an code-behind file, using the .aspx.cs extension and is written in C#. WebForms is used as an integrated part of the project, primary used in security and web pages but also used by the database to gain information[12].
4.4. Working with server

Figure 4.5: WebForm uses a function called round-trips. When a user interacts with a WebForm on a browser, it is requesting a process on the server, and the information is submitted by the server and processed, returning the information back through a browser (if an answer is requested).

4.4.1 Web service

A web service is a web site that interacts with programmes, instead of people. The development of .NET framework was to enhance the usage of Internet development using web services. The web service main function, instead of delivering web pages, is to make an XML request, perform a specific function, and return the response as an XML message. This makes it possible for the Android application to connect to the web service and to get or set a parameter in the function the web method uses.

Figure 4.6: XML request

Two different technologies are used when defining a Web service, the discovery mechanism and the service description. The discovery mechanism uses an XML document on the server to enable client applications to detect if the web service exists, and if there are any detailed description of that service. The service description describes the input and outputs of the Web service. A Web service can be invoked either by using raw HTTP, or by using an XML message in a specific form called SOAP (Simple Object Access Protocol). Unfortunately Android does not support SOAP, making raw HTTP-code the option to use when making the connection to the Web service.
4.4.2 C#

C# or C Sharp as it is also called, is an object oriented programming language developed by Microsoft in 2000. Since C# is object oriented it uses encapsulation, heritage and polymorphism. Encapsulation is when you place the functionality inside one package, heritage is a structured way to extend code and functionality to new programmes or part of the programme. Polymorphism is the ability to create a variable, a function, or an object that has more than one form [6].

Another programming language used in .NET framework is Visual Basic.NET (VB.NET), which is a BASIC programming language (Beginners All-purpose Symbolic Instruction Code) developed to show college students a basic and understandable language when first encountering computer programming. Both VB.NET and C# is part of the same .NET development framework from Microsoft. The difference between the two languages is mostly the syntactic, i.e. the words you use to write the code. Otherwise both of the languages compile to the same runtime framework in .NET, and are both object-oriented languages.

This makes the choice between C# or VB.NET hard, but since there is a shorter learning curve to C# than VB.NET, when using Java as a prior language (Which is used in the Android application), C# would become the more natural choice of developing language.

Figure 4.7: Comparison between VB and C#.

4.4.3 Web pages

Creating web pages is a straightforward technique in ASP.NET. Web page files could contain HTML, C#, VB.NET or some other ASP.NET language. Since it isn’t a pure HTML file, it is given the extension .aspx, being read by MS Server. The main page in any web project
4.5 Working with database

A database is a computerised record-keeping system used to save large amounts of information or collections of computer files in order. The largest used database language right now is *Structured Query Language* (SQL)[10], created originally by IBM. SQL is now an international standard used by almost any commercial product using databases. Databases
are used in many different sizes of systems and are called **singel-user systems** when describing smaller machines e.g. the phone book in a smartphone, and **multi-user systems** if many users are able to use the database at the same time[10].

Between the physical database and the user, there needs to be a Database Management System (DBMS). The DBSM in this project is MS SQL Server 2008 R2. MS SQL is a database using a *relational* system, not to be compared with *relationship* in databases, even though MS SQL uses relationship also.

**Relation** is how the database is built, and shown to users. The relational database is based on logic and mathematic, making it easy teaching the foundations and principles[10]. The relational database could be explained as:

1. The user perceives the data as tables (and only as tables).

2. The operators available to the user for (e.g.) retrieval are operators that derive "new" tables from "old" tables.

![Figure 4.9: Relational database.](image)

**Relationship** are the connections made between the tables in the database, used to facilitate the user when retrieving information from the database.
4.5.1 Graph databases on social web

As seen from the name, Graph databases are databases who’s not using the traditional relational database structure, where you save information in tables. Large companies as Facebook, Google and Amazon have made their own versions of this kind of database, for its capability to handle large amounts of data. The idea of a graph database is not new, going back to the eighties, but in recent years the large-scale usage of graph databases has emerged thanks to foremost social network developing companies having thousands of people connection to their databases at once[3].

The essential theory about graph database is similar to object oriented programming (OOP), making it’s structure understandable for developers using OOP languages. The idea is to have the information in the database inside nodes (or sometimes called vertex), and to have a connection between nodes called edge[28]. This means that the database is using a graph as its primary data structure.
Figure 4.11: Graph database theory.

Figure 4.12: In social web pages as Facebook, many users are connected to each other. This makes the connections aware of each other, making notifications to many people at the same time. On social web applications like Twitter, the users are followers of each other. This means that the information sent over the database are sent in a linear direction, going from one person to the next.

The usage of graph database is a natural evolution coming from the progress of Internet structure. In the beginning there where one person going on to one webpage, finding the information this person needed. This developed to becoming several webpages working together, moving on to become the social web. Social web are a way of social interaction using internet. It could mean online shopping, gaming or social web sites as Facebook[28].
In a comparison between regular relational database management system (RDBMS) and a graph database, the result showed the graph database to be considerably faster in searching through all the nodes[28]. The test compares PostgreSQL[2] versus Neo4J, a Swedish developed graph database, from Neo Technology[1].

Test parameters:

- 1 Million Vertex
- 4 Million Edges
- Scale Free Topology
- PostgreSQL vs. Neo4J
- Both Hash and BTree

<table>
<thead>
<tr>
<th>Depth</th>
<th>RDBMS</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100ms</td>
<td>30ms</td>
</tr>
<tr>
<td>2</td>
<td>1000ms</td>
<td>500ms</td>
</tr>
<tr>
<td>3</td>
<td>10000ms</td>
<td>3000ms</td>
</tr>
<tr>
<td>4</td>
<td>100000ms</td>
<td>50000ms</td>
</tr>
<tr>
<td>5</td>
<td>N/A</td>
<td>100000ms</td>
</tr>
</tbody>
</table>

**Restricted usage in project**

The usage of Graph databases was not possible in the project considering that the database from SLV was in MS SQL format. Since the test participants are not going to be over 300 persons, the need of using a large-scale database was not required. If the test participants would be many thousands of people, or if they needed to view or review each other in a dietary study the usage of graph databases would become relevant.
Chapter 5

Methodology

Creating the solutions and designs in this project was not advanced work, but it was more
time consuming than originally planned, resulting in a delay of several weeks. One of the
main reasons for this time consumption were the project specifications who kept altering,
making the design from one week obsolete, and new ones needed to be created.

Since this was not a front-end project (front-end development were made by Mats
Högdal), it was not a question about drawing buttons and interfaces, and making user
test on the interface. It was about how the back-end parts as simple as possible is made to
fit in the goals description.

Using the goals to perform and plan the back-end development gave some specific parts
needed. The ability to secure information from the application, and sending it to the server
were one of the most important goals. The information should also be structured and stored
in an efficient way.

5.1 Android

The android development started with the search on how to transfer information from the
smartphone onto the Internet.

The first step of development was what format the information should be. The choice was
between XML and JSON. JSON where chosen because of the easy readability in Android.
It could be contradictory since Android applications are written in Java and XML. But it
does not mean that XML would be easier to use when sending the information using server
and database.

The second step of development were to send the information using a REST-client. Having
read about REST-clients techniques, it suited this project well. REST uses a standardi-
sation, easy to adapt to the projects Internet transfer. Using this technique, the information
could be easily packed and a connection to the server would be created at the same time.

5.2 Server

In the beginning of the project, there was a discussion on whether or not to use PHP to
transfer information. PHP uses simple codes and do not take much space on the server.
However, this would mean that the application would be responsible of all calculations
needed to be done before sending the information to the database. When the database
from SLV were used, it were in MS SQL format, making the choice to use PHP obsolete since PHP and MS SQL connections were a new technique. That gave the point of using Microsoft developer programmes, being adaptable and commonly used techniques. Thus, the connection to the database would be of minimal problem, and the transition between programmes would be easy.

Inside the Web Project, a Web Service were made. This Web Service could do a numerous things, e.g. send and read HTTP methods (GET, PUT, POST, UPDATE), and it would fit the REST-client made in the application.

Inside the Web Service, the need to calculate information and transform information from the database to JSON, was possible because of the ability to use programming files next to the Web Service. These files could ether be made in C#, the programming language in ASP.NET or by Visual Basic. C# were more similar to the object oriented language Java, and therefor C# was chosen.

### 5.2.1 Web page design

The four web pages were kept in downsize, need to know basis, since it would be used in smartphones and/or computers. The information on the webpages was minimal, there was no images, only a small amount of text and buttons. There is a login/logout display at the top, underlined and visible.

### 5.3 Database

The design of the database were made to be a copy of the parts needed from the database in the application, and were made in a lowest necessary need basis. Some of the tables were direct copies from the application database, since it was identified to be the easiest way of sending information.

If the table in the application containing personal information were lost, the backup on the server wouldn’t have all the necessary information needed, but now the user don’t need to fill in the gaps. The other tables are a collection of information from the application database and the information needed from SLVs database.

### 5.4 FileMaker

The main function of FileMaker was its ability to show information easily from a database, and to export it to SPSS. The design used in FileMaker is simple, directly made from FileMakers own standard templates. There were no need to make any extra interface, since it only would show the information as it is inside the database.

### 5.5 User guide

There were no technical people in this project, the creation of a user guide for how programs work was necessary. It can be seen at Appendix A. The text are written in Swedish, since it was used by Swedish personnel.
Chapter 6

Resulting work

Inside this chapter a thorough overview on the work is shown containing all parts of the project. First of is the android application back-end development, implementing a REST-client and how to use it. The next step is the server, showing the web pages, security and how the information from the android application is received, translated and sent back if requested from the application. The database is explained, showing how it is created, and how to use it. FileMaker is the third part programme used to how the information in the database. It is explained and it is shown how to be used.

6.1 Android

Inside the Android project, different main parts where made to complete the goals. The first part where the ability to connect to the server, using a REST-client, with specific attributes to fit the applications purpose. The second part is going a bit deeper inside the project, focusing on the programming design and how the different classes are working together.

Since the android application is developed on two separate locations, the android classes inside this project are developed as autonomous as possible. This makes it easier for all project developer to make changes without disturbances or delaying.

6.1.1 Implementing REST-client

The REST-client is using a client to communication with the server, where the client passes the information to the server, the server process the information and returns a response [13]. This kind of information transfer is useful in an application, when its main Internet connection purpose is to send and receive simple information from a server, being the data to and from the database containing dietary information about the test participant.
Chapter 6. Resulting work

Figure 6.1: The REST-client path. (1) The information is sent from the android application to the **translator** class. (2) The **translator** class transforms the information to JSON-format and transfer it to **connect** class. (3) The **connect** class makes the connection to the server, and sends the information. (4) If the **connect** class requests a answer from the server, it is sent back in JSON-format. (5) The information is sent back to the **translator** class, and extracted. The **translator** class checks where to send the information inside the application before it is sent. (6) The information is used inside the application.

**Java Class Translator**

When passing information from the application towards the server, the first step is to transform the information to a format readable to the server. The main purpose of Translator is to get the information from the application and translate it into JSON, packing all the information, before it is passed to the Connect class. The same technique is used when the information comes from the server, or more specific, is passed down from the Connect class. However since the information from the server is sent in JSON format, which is manageable by Android, it is passed to the method in the application requesting the information.

**Java Class Connect**

The next step when sending the information to the server is to make the HTTP-connection, which is responsible for connecting to the server. This is made inside Connect, using HTTP-methods. If the method is passing information to the server, the method needs the information from Translator class, being the JSON-object and the web service method needed to perform the connection. When the method is requesting an answer from the server, the information is HTML text, and placed inside a new JSON-object before it is passed to Translator. This is because the JSON text is imbedded inside HTML text, and needs to be taken out and then placed inside a JSON object.
6.1.2 Programming and design

To assist the REST-client, some classes were made to perform an Android specific type of task. The different classes are used to control the applications Internet connection, to start new threads when needed and to use a timer when it needs to make check-ups. All of this is taking place inside the REST-client.

The Internet status control is used since the application can’t be relying on Internet connection as Wi-Fi or 3G. When a test participant wishes to send information to the server, and there is no Wi-Fi close by, this information needs to be sent at a later time. If the Internet connection is closed down, the application could easily save the information and send it as soon as it goes online. To do this, it needs to know the connection status of the application.

Creating new threads are important for the application to perform correctly and without delay. The main thread in Android is the one responsible for all main graphics, if nothing is altered or added to the project by the developer. For example, if the test participant needs to get dietary information from the server using the REST-client, it would try to connect to Internet (i.e. Web Service) and wait for the answer. Since this request is done on the main thread, all other activities would freeze until the task is complete. If the information (file size) is large it could take minutes to download and the application would be frozen. To solve this potential problem, a new thread is created for every request made by the application user, and the download is not noticeable.

The timer is used with a Service, which is not a thread, but a means in Android to do background processing inside the main activity thread. This is used for time consuming (but not process consuming) activities, such as a timer. Since it is part of the main thread, it will follow the applications warnings and interruptions.
Figure 6.3: This image shows the REST-client process. When the test participant sends the information, it firstly needs to check if it is connected to the Internet. If there is no connection, it saves the information and starts a timer trying to see if there is any connection possible. If there is a connection, or the timer finds a connection, a new thread is created to send the information.

## 6.2 Server

In this project, the server is used as a communicator, project holder, storage and calculation device. All of this was made possible inside the Web Service.

### .apk file

One of the main reasons to use a server was the ability to place an application package file (.apk) file on it, and being able to downloading it from the server simply by clicking on the file. This is done from the web browser, on the test participant’s Android phone. This makes it much easier as a developer to give the application to whoever needs it. I.e. only people performing the test have access to the web page containing the application file. The android phone has a restriction when using application files from unknown sources. To use the application the user needs to remove the restriction inside the phone, on the settings. This is simply done by accepting it inside the settings of the phone, and accepting that the application is using the phones Internet when installing it. Many other applications use the
same technique as this application; the only difference is that the test participants now need to accept the settings.

### 6.2.1 Web Service

Creating a connection to the database inside the Web Service is the primary function of this Web Service. It is done by using an .asmx file containing the web methods called from the Android application. All of these methods are written specifically to handle one task, but the most common one are simple GET or SET information to the database. When getting information from the database, it is called by the application, given some or none parameters of what is should get, and retrieves the information by using SQL Queries. The information is then packaged and sent back to the application.

![Diagram](image)

Figure 6.4: Get information from Web Service.

Sending information to the database through the Web Service is a bit more complicated. The information needs to be sent through the REST-client (see section Android). When the information is collected by the web method, the information is passed on to the web service classes and placed inside the right database table. If the information is about updating the database, it is passed down using SQL queries. However, if the information needs to be controlled against the database or calculated, it is sent from the web method into the right C# class, where it is calculated and passed down to the database.
Chapter 6. Resulting work

Figure 6.5: Set information to Web Service.

**JSON translation or ConvertClass**

The *ConvertClass* is the most important class inside the Web Service. It is the class that takes the information given from the database and transforms it into a JSON-array. The information from the database is shown the same way as they are placed, in tables. It is possible to send this information directly to the application, but then it would be harder to pinpoint some exact value, as it only would be read as a text file.

**CheckClass**

The *Checkclass* is the class used when some information needs to be confirmed inside the database. If the test participant is passing on some new private information, this class checks if the user exist, if the information in the database needs to be altered or replaced, and gives the correct response back to the web method. This class also examines if there are some food types needed to be changed in the database, or the information passed is older then the one in the database.

**CalculateMealClass**

CalculateMealClass takes the meal information from the application, and places it in the correct database table. This is demanding some GET and SET methods from the database, since the food needs to be calculated depending on how much you eat, what you eat and when you eat. E.g. if the test participant eats an apple, with a weight of 46 gram. The information is sent to the web method and passed to the *CalculateMealClass*. The class then finds out what kind of nutrition an apple contains, and since it is given in 100 gram, the nutrition needs to be reduced from 100 gram to 46 gram. The information is placed in the database, in the tables containing nutrition, nutrition per meal, nutrition per day and nutrition per main group. If the test participant changes the information, from an apple to an orange, all the information in the tables needs to be changed.
CalculateTranspondClass

To make the information in the tables usable in the programme SPSS[19], it has to be transponded, that is, shifted from row pairs into table columns. This is so the information is easily read and used inside FileMaker and SPSS. This kind of information altering is not something MS SQL does easily, so it was necessarily to make this kind of change manually using C# methods. The tables containing this information is made to match the tables used by CalculateMealClass, except that they are shifted to the side.

<table>
<thead>
<tr>
<th>Id</th>
<th>FoodId</th>
<th>NutritionId</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.14</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>3</td>
<td>0.001</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>4</td>
<td>55</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6.1: Ordinary database table structure.

<table>
<thead>
<tr>
<th>Id</th>
<th>FoodId</th>
<th>Nutrition 1</th>
<th>Nutrition 2</th>
<th>Nutrition 3</th>
<th>Nutrition 4</th>
<th>Nutrition 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.14</td>
<td>2</td>
<td>0.001</td>
<td>55</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 6.2: Transponded table structure.

DeleteMealClass

DeleteMealClass has the simple task of deleting information inside the database, though all information is not always replaced with new. It is also the class responsible for altering or deleting information when the user changes a meal item or some nutrition in the database. Since the database is containing nutrition information over different aspects, such as meal, day etc. it is important that the final amount is calculated correctly at the end of the day.

6.2.2 Web page

The web pages are the only parts of the web project observed from outside the server. It is the tool from which the test participant is able to reach the application, and also where access to the web pages can be administrated.

There are four web pages in the entire project. The first one is the Default.aspx page, containing the link which the test participants could download the information and the necessary information needed on how to install it. The second page is the GuestPage.aspx. This is the page where guests of this project are able to download the application, they wish to test it. The third page is the AdminPage.aspx, where administrative tasks are performed. This is possible to do from inside the Web Service itself, but using the web page makes it much easier for a non-technical person to understand. The fourth page is the Logon.aspx page, which you are directed to if you don’t have a valid username or is a new user. This page is overlooking all other pages, and if a person tries to move inside a page where they don’t have permission, the Login.aspx page is shown.
Chapter 6. Resulting work

Figure 6.6: Webpages.

Figure 6.7: Login webpage.
Användarnamn:

Åsa Logga ut

Guide till installation av applikationen:

1. I din android-telefon, gå in i huvudmenyn (menu på engelska).
2. Gå in på inställningar (settings på engelska).
3. Gå in på program (applications på engelska).
5. Ladda ner programmet från knappen längst ner på denna sida.
6. Installera applikationen.

Hittar du inte applikationen efter att den har blivit nedladdad, gå återigen in på meny, inställningar, program. Gå sedan in på "hantera program" (manage applications på engelska). Där finns det nedladdade programmet som du kan installera.

Figure 6.8: Default webpage.
6.2.3 Security

Security inside the Web Service is most important, otherwise all kinds of different persons could easily gain information from the database. It is also important to ensure the integrity of the participants by keeping names, personal information and dietary habits concealed to unauthorised people.

Security have been implemented in the server, and Web page security using cookies was used. This means that the login information is saved a short time, since the cookie used inside this Web Project is using a timeout after 10 minutes. Besides using the Login.aspx web page, there needs to be more backside information for this to work. Inside the Web Service Web.config file (the file where authentications and behaviour of the Web Service is written), the permission to all the web pages are given.
6.3. Database

The database is divided into three parts. The first part is from SLV, the second part is the one used in the dietary measurements and the third part is the one used by the Web Service.
The first part, containing all the different tables from SLV is explained in chapter 2 (Background). The third part is created and used by the Web Service containing all tables to keep track of all users of the web pages and the security if those. This is also where you assign administrators to the project.

6.3.1 Database of dietary study

Different parts of the database are copies of the database used in Mats Högdals front-end work, i.e. the database used in the android application. This part contains the Personal table and the Activity table. The usage of a database copy is because it would be easy to send back and forth information that the user is setting or adjusting. If the user changes the smartphone to another and continues the study, all the settings are saved in the database and sent to the new smartphone. The same approach is used for the Activity table.

Since all test participants get a unique id, this is what keeps track of the person’s meals. All meals eaten by this person are saved in the Meal table, and the food in every meal is saved in the Food table. The Dates table keeps a record of all dates in the calendar, giving a specific date a specific id, making it possible to easily overlook what date something was saved. The other tables contains the nutrition food measurements.

![Diagram](image.png)

Figure 6.12: The dietary study database is connected to several tables from the SLV database, taking all information from those tables and placing it inside the dietary study tables.
6.3.2 ODBC

ODBC (Open DataBase Connectivity) is used to create a connection between FileMaker and the database. In Windows, an ODBC source is available for free, but for Macintosh it has to be downloaded as a third party programme.

6.4 FileMaker

FileMaker was the programme chosen to simplify for the non-developers of the project to access and visualise the data. This programme is a way of reading database information in a simple manner and to export the information to a format readable to SPSS. First of, there are seven different pages created. All of these contain different information:

- (1) Get all users.
– (2) Get all users and activity.
– (3) Get all users, activity, meals and food.
– (4) Get nutrition per meal.
– (5) Get nutrition per food.
– (6) Get nutrition per day.
– (7) Get nutrition per main group.

The top three pages shows simple information as usernames etc. The later four shows the more important information of the project, e.g. nutrition. The information sent to these comes from the tables that are transponded (see chapter 6, section 3), and therefore these tables are also transponded and ready to be exported. In FileMaker, it is possible to search for a specific person or meal, making it possible to filter the information and show only information that is relevant for a certain user. Also, in the later four, the information about the test participants is added.
Figure 6.14: Filemaker user page.
Figure 6.15: Filemaker nutrition per meal page.
Chapter 7

Conclusions

The work in this project contained many different implementations, from the implementations of databases, programming classes to web pages. This makes it hard and time consuming to deliver all the pieces needed, and the focus on the work is to make the parts work together, and identify and choose parts that do work together.

The biggest challenge in this project was the time aspect. When needed to optimise and create a system with parts never been used by the developer, a lot of time is needed to understand the software and workflow. Creating a Web Service was in the beginning a bit confusing, but when understanding the GUI and workflow, it showed real good capacity.

All the original goals where reached, and the back-end is fulfilling its purpose. From the beginning, the goals where not set in stone, they where more guidelines of what was needed in the project. When the project developed forward, more specific goals could be created.

7.1 Future work and changes

For future work, there is a lot of things possible of optimising. Of course, many systems need to be updated since the phones OS (operating system) or the server gets updated with newer software, often making it run more smoothly.

In this project, almost all the modules would benefit from further development. The Android device is in need of a more reliable and functional REST-client. It could be developed in a more object-oriented manner and the classes do not work together in the most effectual way, making check-ups and thread-starts consuming both in terms of time and efficiency.

The server has several parts, which could use a look-over. One suggestion would be to change the website to an environment better suited for smartphones. Another is to develop the security further in the web service in order to keep the information safer and make the database harder to access. One major part that needs a look-over in the future is the connection between the database and the web methods. These classes contains methods specific to the database developed in this project, but if these databases would change you would need to change some parameters inside the code, making it fit the new database. It would have been preferred if they where completely adoptable, making it possible to change in databases or web methods without having to search for possible error.

The database is as simple as it gets, but it is still adapted to the database in the Android application. If the application changes, the database also needs a change. For the future, it would be preferred if the database and the application database where more alike, making the same changes to the both at one time.
FileMaker fills its purpose of use, but it has a flaw. It could only contain 66 columns [14] after each other in the views, making it necessary to create two separate views for one database containing all 53 nutrition values plus the information about the user. If this were changed in a newer version of FileMaker, it would be much appreciated and used.
Chapter 8

Acknowledgements

I’d like to give a big thanks to everyone who has been involved in the project:

– Åsa Svensson - Project commissioner and dietary expert.
– Christel Larsson - Project supervisor and supreme dietary expert.
– Mats Högdal - Co-worker and front-end developer.
– Claes Bergman - Very friendly office neighbor and narrator of anecdotes.
References


[34] J. Steele and N. To. The Android developers cookbook - building applications with the Android SDK. Addison-Wesley, 2010.


Appendix A

User guide
Sammanfattningsdokument
Kostapplikationen
Fredrik Bäck
Innehållsförteckning

Inloggningsuppgifter .................................................................................................................. 3
  GU-konto: .......................................................................................................................... 3
  MS-Server......................................................................................................................... 3
  MSSQL-databas ............................................................................................................... 3
  VPN .................................................................................................................................. 3
  Filemaker ......................................................................................................................... 3
  SVN (Sourceforge) ......................................................................................................... 3

Användning ............................................................................................................................ 3
  VPN .................................................................................................................................. 3
  MS-server (Microsoft-server).......................................................................................... 4
  MSSQL-databas (Microsoft SQL-databas) ...................................................................... 6
  Visual Studio ................................................................................................................... 8
  ODBC (Open DataBase Connectivity) .............................................................................. 10
  Filemaker (ofullständig) ............................................................................................... 15
  Ladda hem applikationen till telefonen från internet ...................................................... 21
  SVN(Sourceforge) ............................................................................................................ 22

Uppbyggnad .......................................................................................................................... 23
  Hela projektet (server, databas, webbtjänst och FileMaker) ......................................... 23
  Webbprojektet (Hemsidor och webservice) .................................................................... 25
  Databasen ....................................................................................................................... 26

Kontaktuppgifter .................................................................................................................... 26
  SLV .................................................................................................................................. 26
  GU .................................................................................................................................. 27
**Inloggningsuppgifter**

**GU-konto:**
Användarnamn:
Lösenord:

**MS-Server**
Serverutrymme:
URL: http:
Användarnamn:
Lösenord:

**MSSQL-databas**
Namn:
Användarnamn:
Lösenord:
URL:
IP-adress:

**VPN**
Samma som GU-konto

**Filemaker**
I programmet används samma som till MSSQL-databasen.

Installationsuppgifter:
Användarnamn:
Organisation:
Licensnyckel:

**SVN (Sourceforge)**
URL:
Användarnamn:
Lösenord:

**Användning**

**VPN**

*Virtual private network (VPN)*, på svenska ”virtuellt privat nätverk”, är en teknik som används för att skapa säkra förbindelser, så kallade *tunnlar*, mellan två punkter i ett osäkert datanätverk.

VPN måste alltid användas när man inte är inne på GU:s nätverk/Eduroam för att komma åt servern eller databasen.
http://www.it.gu.se/foranvandare/guwlan/eduroam/
För att skaffa VPN, gå in på:
http://www.it.gu.se/foranvandare/vpn

**MS-server (Microsoft-server)**
Till MAC:

Till Windows (rekommerdat):
För att komma åt utrymmet med Windows lägger man till en ny nätverksenhet.

1. Gå in på Dator.
2. Välj anslut nätverksenhet.
4. Ange länken till serverutrymmet.

    a. Ange användarnamn och lösenord där det är nödvändigt.
    b. Om du inte kan logga in, kontrollera att du är på GU:s nätverk inloggad med x-konto, eller att du använder VPN med x-konto.
Denna nätverksenhet används sedan när man ska lägga upp hemsidor eller liknande från Microsoft Visual Studio som ska användas på matmobil.ped.gu.se

Det som finns på nätverksenheten(servern) nu är skapat som ett webbprojekt i Microsoft Visual Studio, vilket är ett lämpligt program att använda.

**MSSQL-databas (Microsoft SQL-databas)**
Modell av databas: Microsoft SQL 2008 R2

Microsoft SQL-server 2008:
1. Starta SQL Server Management Studio.
2. Fyll i uppgifterna till databasen i rutan som kommer upp.
3. Om anslutningen fungerar kommer databasen synas till vänster i programmet.

Microsoft Visual Studio:
2. Tryck på connect to database.
3. Skriv in uppgifterna till databasen, testa sedan anslutningen (test connection).

Visual Studio
I Microsoft Visual Studio är webbprojektet skapat. För att kunna ladda upp projektet till servern behöver man först skapat en nätverksenhet (se MS-server ovan).

1. Kolla så att nätverksenheten är igång.
2. Starta VS-projektet.
3. Välj Build, Publish “namn på projektet”.
4. Ange nätverksenheten som mottagare.
**ODBC (Open DataBase Connectivity)**

ODBC till MAC används som ett som ett tredjepartsverktyg, dvs. man behöver köpa det till datorn. En bra ODBC-driver som rekommenderas finns på:


1. Gå in på kontrollpanelen, System och säkerhet, Datakällor (ODBC).

4. Döp kopplingen till wh-matmobil OL. Det är viktigt då kopplingen i FileMaker heter så.
5. Ange användarnamn och lösenord till SQL-databasen.
7. Testa anslutningen och spara. ODBC-källan kommer sedan hittas automatiskt av program som använder ODBC, som t.ex. FileMaker.


**Filemaker**

OBS. Innan man kan starta FileMaker behöver ODBC-kopplingen vara skapad. Om man inte är inne på GU:s nätverk eller Eduroam behöver VPN-anslutningen vara på.

Filemaker laddas hem från [www.ftp.gu.se](http://www.ftp.gu.se) och installeras som ett vanligt program.

Starta projektet i FileMaker:
1. Kontrollera att ODBC-kopplingen är skapad med namnet wh-matmobil OL.
2. Starta FileMaker.
3. Starta projektet genom att trycka på Bläddra i mitten längst ner och välj FileMakerfilen.
5. FileMaker kommer då starta men visa tomma tabeller.

8. Välj Ange för DNS.
9. Markera den nyskapade datakällan (SQL server native client 10.0). Tryck OK.
9. Tryck OK i programfönstret "Redigera datakälla".
10. Tryck OK i programfönstret "Hantera externa datakällor".
11. Nu ska databasen synas i FileMaker.

**Ladda hem applikationen till telefonen från internet**


För att ladda hem och installera applikationen:
1. Tillåt telefonen att ta hem okända källor (unknown sources)
   a. Gå in på Meny.
   b. Fortsätt till inställningar.
   c. Fortsätt till applikationer.

2. Markera rutan okända källor.

4. Installera applikationen i telefonen.
5. Hittar man inte programmet eller ska ta bort det finns det under hantera program (manage applications, se bild ovan).

SVN(Sourceforge)
SVN är ett versionshanteringssystem speciellt utvecklat för att komma tillrätta med vissa begränsningar i CVS(äldre version). Ett versionshanteringssystem håller reda på historik och förändringar och gör det möjligt att backa tillbaka i tiden eller att spåra vem som gjort vad. Dessutom möjliggör det samarbetet på samma projekt för utvecklare som är spridda över världen.

http://www.eclipse.org/subversive/

För att programmera applikationen används ett program som heter Eclipse. Eclipse finns att ladda ner från:

Använda SVN i Eclipse:
1. Gå in på File, Import, SVN, Project from SVN. Tryck Next.
2. Välj create a new repository location.
3. Ange den URL som finns vid inloggningssuppgifterna, samt användarnamn och lösenord.

**Uppbyggnad**

**Uppbyggnad**

Hela projektet (server, databas, webbtjänst och FileMaker)
1. Användarnas smartphones

2. Uppkopplingen mot internet sker genom Wifi (trådlöst nätverk) eller 3G.


   - **Android** ➔ **Server** ➔ **Databas** ➔ **Server** ➔ **Android**

   - Android skickar data till server.
   - Tar emot data och anropar databasen med ASP.NET
   - Lagrar alternativt skickar data.
   - Tar emot data, ASP.NET gör om det till JSON-filer och skickar till Android.
   - Tar emot data från server

4. Relationsdatabas med SQL (Structured Query Language).

5. Filemaker används som gränssnitt för att komma åt och exportera tabeller och värden i databasen.
Webbprojektet (Hemsidor och webbservice)

1. JSONWebService används för att skicka och ta emot data till och från android applikationen.

2. Login används för att kunna identifiera användaren.

3. Default är huvudsidan där man laddar ner applikationen, alla användare utom Guest har tillgång till denna sida.


Kontaktuppgifter

Fredrik Bäck  
Tel: 0733 52 43 93 
Mail: beck.fredrik@gmail.com

SLV  
Marianne Arnemo  
Tel: 018-175500  
Mail: marianne.arnemo@slv.se

Tekniker:  
Fredrik Willander  
Tel: 0733-899 558  
Mail: Fredrik.Willander@slv.se
GU
Servertekniker:
Joakim Larsson
Tel: 031-7865313, 0768-880255
Mail: Joakim.larsson@handels.gu.se