

DHSS Gateway

Building a system for supporting the storage and retrieval of demographic and health related data



**LUNDS
UNIVERSITET**
Lunds Tekniska Högskola

LTH School of Engineering at Campus Helsingborg

Bachelor thesis:
Robert Tublén

© Copyright Robert Tublén

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

Printed in Sweden
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2008

Abstract

About 80 percent of the earth's population live in poverty, the majority living in third world countries. To combat issues related to poverty, such as child mortality, it is important to have a system in place for continuous monitoring of demographic, health and socioeconomic events within the population.

A demographic surveillance system (DSS) continuously monitors a subset of a population within a specific area, registering events such as births, deaths, and migrations.

Information collected is used for educational purposes and for constructing public health program aimed at improving the general wellbeing of the population.

Furthermore, it acts as foundation for sub-studies related to health and socioeconomic factors such as mapping teenage sexuality and reproduction, sexual risk behavior, domestic violence and so forth.

There are numerous DSS sites set up today all over the world, most of which can be found in developing countries throughout Africa and Asia.

One such site is located in Central America, in the municipality of León, Nicaragua, where a subset of the population living within the municipality is under continuous surveillance.

The result of this thesis is a prototype for a system intended to be used at the DSS site in León, supporting the registration and usage of collected demographic, health and socioeconomic related information.

Keywords: demographic surveillance system, DSS, continuous surveillance

Sammanfattning

Ungefär 80 procent av jordens befolkning lever i fattigdom, majoriteten av dessa i tredje världen. För att bekämpa problem relaterade till fattigdom, så som till exempel barnadödlighet, är det viktigt att ha ett system på plats för kontinuerlig övervakning av demografiska, hälsorelaterade och socioekonomiska händelser inom populationen.

Ett demografiskt övervakningssystem (DSS) övervakar kontinuerligt en delmängd av populationen inom ett specifikt område, där händelser så som födslar, dödsfall och migrationer registreras.

Information som samlas in används i utbildningssyfte och för att konstruera program med syfte att förbättra populationens allmänna välmående.

Utöver detta används denna som grund för studier relaterade till hälsa och socioekonomiska faktorer så som kartläggning av tonårssexualitet och reproduktion, sexuellt riskbeteende, våld i hemmet och så vidare.

Det finns idag ett antal DSS platser uppförda runt om i världen där merparten av dessa återfinns i utvecklingsländer i Afrika och Asien.

En sådan plats finns i Centralamerika, i Leóns kommun, Nicaragua, där en delmängd av populationen inom kommunen är under kontinuerlig övervakning.

Resultatet av detta examensarbete är en prototyp för ett system tänkt att användas vid DSS-platsen i León för att stödja registrering och användning av insamlad demografisk, hälsorelaterad och socioekonomisk information.

Nyckelord: demografiskt övervakningssystem, DSS, kontinuerlig övervakning

Foreword and Acknowledgements

I remember the feeling of excitement and adventure I got while listening to the vivid lecture held by Nils Assarsson and Mattias Pedersen when presenting their thesis's [1] [2] at LTH School of Engineering, Helsingborg in the autumn of 2005.

When I heard that more work needed to be done at CIDS in León, Nicaragua to set up a new system for registering demographic, health and socioeconomic data, I didn't hesitate. Working on this thesis has truly been a life changing experience.

I extend my deepest gratitude to Nils and his family for their hospitality and for taking care of me during my stay, and for introducing me to such wonderful people as Dr Elmer Zelaya Blandón, Maria Teresa Orozco, and others whom I've had the pleasure to meet and work with during my work on this thesis.

I would also like to extend my thanks to the following people, in no particular order: Anders Hovén, Johan Augustsson, Jon Lennryd, and to all the people at CIDS.

I meet a lot of interesting people during my stay in Nicaragua, to many to mention, but who all had a positive impact in my life and for that I am grateful.

Finally, I would like to extend a huge thanks to my supervisor Christin Lindholm for her support and patience as deadline after deadline slipped away.

Definitions

Socioeconomic	The study of the relationship between economic activity and social life.
Epidemiological studies	Study of factors affecting the health and illness of populations.
Ad hoc	Performing something without a detailed plan.
Web Services	Software system designed to support interoperable machine-to-machine interaction over a network.
Resource-oriented architecture	Specific set of guidelines of an implementation of REST, a style of software architecture for distributed hypermedia systems.
Auditing	Tracking the actions of users.
Authorization	The concept of allowing access to resources only to those permitted to use them.
Authentication	The act of establishing or confirming something (or someone) as authentic; that claims made by or about the thing are true.
Functional requirements	Specifies functions of a software system or component. A function is described as a set of inputs, the behavior, and outputs.
Non-functional requirements	Requirements which impose constraints on the design or implementation, such as performance engineering, quality standards, or design constraints.

Table 1. List of definitions

Acronyms and Abbreviations

SAREC	Swedish Agency for Research Cooperation with Developing Countries
CIDS	Centro de Investigación en Demografía y Salud
DSS	Demographic surveillance system; The process of defining risk and corresponding dynamics in rates of birth, deaths, and migration in a population over time.
URI	Uniform Resource Identifier
DHSS	Demographic and Health Surveillance System. See DSS.
HTTP	Hypertext Transfer Protocol; use for retrieving inter-linked text documents.
API	Application Programming Interface
PDA	Personal Digital Assistant
REST	Representational state transfer; a style of software architecture for distributed hypermedia systems.
ORM	Object-Relational Mapping
DRY	Don't Repeat Yourself; defaults alleviate repetitive code and configuration

Table 2. List of acronyms and abbreviations

Table of contents

1 Introduction	1
2 Background	2
3 Problem Description	3
4 Purpose and goal.....	4
5 Limitations	5
6 Development process	6
6.1 Development methodology	6
6.2 Tools.....	6
7 Results	7
7.1.1 Deliverables	8
7.1.1.1 <i>DHSS Gateway</i>	8
7.1.1.2 <i>DHSS Gateway SDD</i>	12
7.1.1.3 <i>DHSS Gateway API Documentation</i>	12
7.1.1.4 <i>DHSS Gateway Client</i>	12
8 Further Development	15
8.1 Digitalize forms for use by disconnected devices.....	15
8.2 Enabling checkout of data	15
8.3 Implement authorization, authentication, and auditing	15
9 Conclusion.....	16
10 References	17

Appendix I: DHSS Gateway DSS

Appendix II: DHSS Gateway API

1 Introduction

All over the world numerous demographic surveillance sites (DSS) have been set up in developing countries to collect demographic, health and socioeconomic information with the purpose of supplying researchers and decision makers with information essential for creating effective public health programs for combating such issues as extreme poverty and child mortality.

Events such as births, deaths, and migrations are collected, together with site specific information, making out a core set of data which can be used as basis for performing sub-studies in related matters.

Nicaragua is considered to be a poor country with 80 percent of the population living in rural areas considered to be poor. Lacking a formal system for monitoring demographic and health related events, a DSS site was established in 1993, with the main goal of supplying researchers with data related to reproductively and child mortality.

However, problems related to data quality diminish the usefulness of the system.

The goal of this thesis has been to construct a prototype of a three tier system, consisting of the database, the gateway server and a client for entering and retrieving data collected by the DSS.

The system that interfaces with the database, in which the demographic, health and socioeconomic information is stored, is called DHSS Gateway.

The DHSS Gateway Client uses the DHSS Gateway to enter, retrieve and update information stored in the database.

2 Background

In 1991 a project called Reproductive and Child health was initiated in León, Nicaragua, with the purpose of conducting epidemiological research, collecting information pertaining to patterns of teenage sexuality and reproduction, and child mortality.

This was a collaboration between the Department of Preventive Medicine, León University and the Division of Epidemiology, Department of Public Health and Clinical Medicine, Umeå University, Sweden, with the financial support of Sida/SAREC.

Through this collaboration the foundation for a demographic and health surveillance system (DHSS) was set up beginning in 1993, collecting information such as births, deaths, migrations, and attributes used for measuring poverty.

The purpose of the DHSS was to

- operate a continuous monitoring system of vital statistics in a defined population in the León municipality,
- perform studies on fertility and monitoring trends and social determinants of infant and under five mortality
- generate basic health data, register trends and transitions in diseases and mortality pattern
- act as a platform for epidemiological studies and research,
- function as a planning instrument for health services such as antenatal care services for pregnant woman, vaccines for children, etc,
- serve as a sampling frame for research in various studies such as maternal mortality, teenage sexuality and reproduction, sexual risk behavior, domestic violence and impact of woman's access and control over resources on child survival

After several years of collecting demographic and health related information it was recognized that the database used for storing it needed to be redesigned since it was proving difficult extracting good quality data, making it less valuable.

3 Problem Description

Due to the ad hoc way the database schema has evolved since its inception in 1993, and the lack of sufficiently implemented constraints, both in the database and in the forms used for entering the information, redundant and faulty data now litters the database.

Besides being a maintenance nightmare, it makes it hard to extract good quality data, making it less valuable for researchers and public health services.

The system has two major problems:

- Badly implemented database design, leading to abundance of redundant and faulty data.
- Insufficient implemented constraints, or the complete lack thereof, in the forms used when entering information into the database, making it possible to register a man as having given birth to a child, and registering the date of birth for a newborn to a date before that of the mother.

To solve these problems a new database design needed to be developed together with a graphical interface for entering data.

4 Purpose and goal

The purpose of this thesis is to construct a prototype for a solution to the problems outlined in chapter 3.

A middle tier, designated DHSS Gateway, will be developed, safeguarding the database used for storing the collected core demographic data, providing access to it only via a specific set of Web Services, making out an API.

A resource centric approach will be taken when creating the system, focusing on domain objects mapping to tables using ORM, exposing them as resources via HTTP in a RESTful [4] way. This shifts the focus from the database to a higher level, leaving the creating of tables and mapping between them up to the ORM layer.

As such, the design of the database itself has not been of major focus since the mapping between objects and tables are performed by the ORM layer.

The middle tier should

- acts as gateway to the data stored in a database
- provide a uniform interface to the data via HTTP and HTTPS
- enforce constraints to prevent data from being corruption
- prevent unauthorized access to sensitive data

5 Limitations

Since it is a prototype security has not been of main focus. This would of course be of high importance were it to be developed further, with the purpose of being used in production.

Instead of focusing on the development of a new database schema for the system, ORM was used for mapping objects to tables. This means there are no ER-diagrams depicting the new database design.

If it turned out to be a performance problem in the future, having developed a fully functional system, the automatic mapping of objects to tables would have to be reviewed at that point.

However, until such time that it is deemed a problem, the use of ORM for automatically mapping objects to tables, creating these through software, will suffice.

6 Development process

This chapter explains the development process and tools used throughout the course of the work.

6.1 Development methodology

No formal development process was used throughout the course of the work, due to lack of planning at the start. This worked reasonable well due to it being a one man project. However, the failure to reach deadlines at multiple occasions can probably be attributed to the lack of a formal development process and the lack of planning in general.

Documents and code were being worked on simultaneously which worked reasonable well. It was an iterative process, writing code and updating the documentation as the vision of the new system crystallized.

No formal milestones existed, besides the occasional hand-ins of the final report for review by the supervisor and the final hand-in of all documents and code at the end of the project.

Since there was no actual customer in this case and therefore no one to consult regarding desired functionality, this unstructured way of working worked fairly well even though time slippage occurred throughout major part of the project, leading to extended deadlines and delayed delivery.

6.2 Tools

For the DHSS Gateway, Grails [8], an open source web framework was used to build the API, partly due to it's currently popularity, having been used for such big sites as LinkedIn [14], among others.

The framework is highly suitable for rapid prototyping due to its ORM capabilities and DRY principles. Also, due to the fact that it runs on the JVM it can leverage existing, time tested frameworks such as Spring [15] and Hibernate [16].

Eclipse [10], an open source IDE was used for writing the code making up the system DHSS Gateway. For the DHSS Gateway Client, Flex Builder was used since the client is built using the Flex framework [17].

Git [11], an open source version control system, was used to keep track of all files and documents, using the services provided by github.com [12].

Microsoft Word 2003 was used for producing all documents throughout the project.

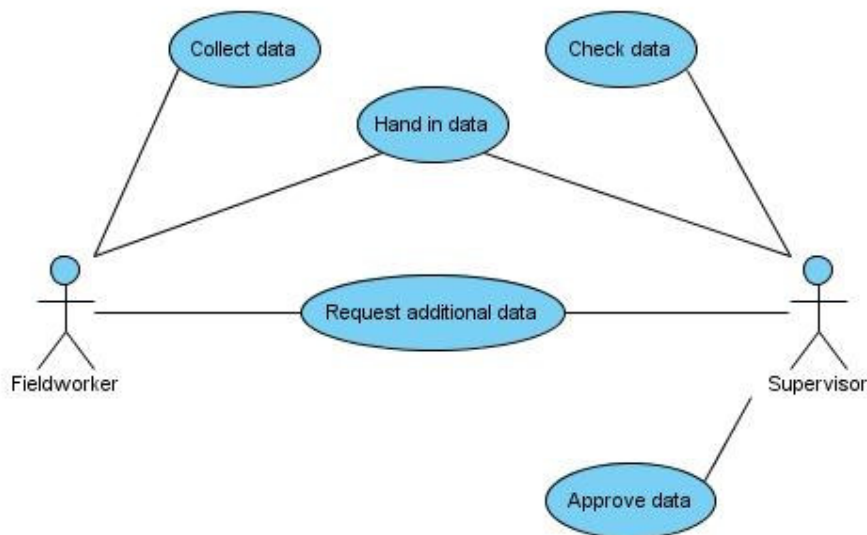
Visual Paradigm for UML Community Edition [13] was used for all UML diagrams. Having tried several alternatives, this generated the best looking diagrams.

The choice of frameworks and tools were mainly done from a perspective of wanting to try out some of the latest technologies. Another factor affecting the choices that were made was the aim of using open source alternatives. Both Grails and Flex are open source frameworks.

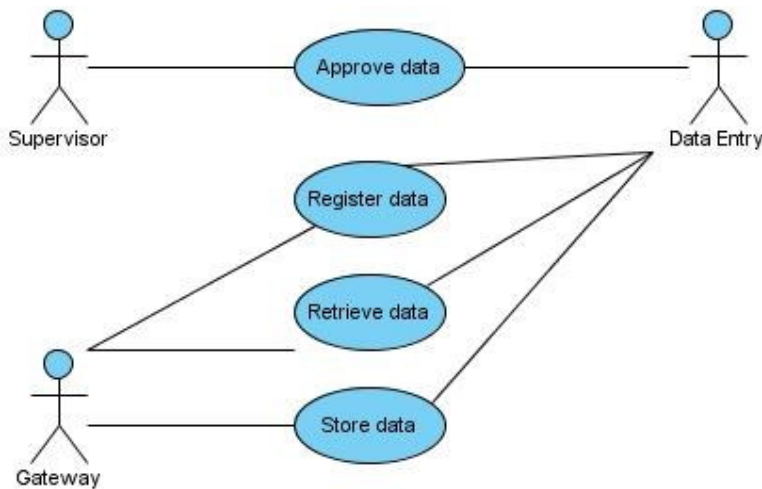
7 Results

The results of this thesis is a prototype for a middle tier acting as a gateway to a database storing demographic, health, and socioeconomic data. Also, a prototype for a client using this gateway has been developed.

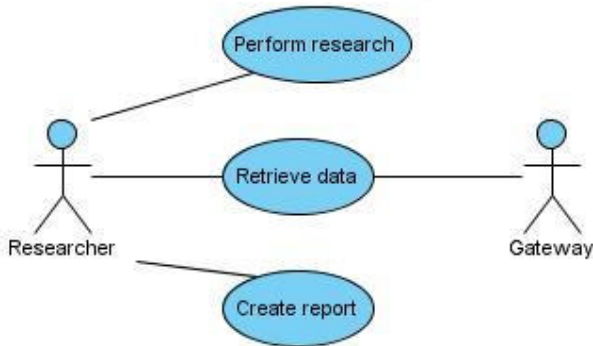
For the DHSS Gateway and DHSS Gateway Client, support for the following uses cases has been implemented. However, since it is a prototype, the security aspects of the system have not been of main concern. See chapter 8 for suggestions on further development.



Use case diagram 1: Displays the use case involving collection and verification of data.



Use case diagram 2: Displays use case involving approval, and registration of collected data.



Use case diagram 3: Displays use case involving the retrieval of data from the gateway.

7.1.1 Deliverables

7.1.1.1 DHSS Gateway

The gateway implements a resource-oriented architecture, exposing the data stored in the in the database as resources in a RESTful [4] way.

REST [5] is an abbreviation which stands for Representational state transfer and is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

It outlines principles for how resources are defined and addressed. A resource is anything that is important enough to be referenced as a thing in itself.

Examples of resources within the DHSS are

- individual with an ID of MA101B401,
- list of individuals,
- individuals who died before they reached the age of 5,
- list of households with dirt floors,
- pregnancy episode for individual with an ID of SB101C103.

Every resource can be accessed through at least one unique URI, e.g. /individual/MA101B401, and they all support the same HTTP methods (GET, PUT, POST, DELETE), although not all resources will implement all methods.

This means that by calling GET HTTP for the resource /individual/MA101B401 we will, by default, get a HTML representation of that resource.

Similarly, by requesting GET HTTP for the resource /individuals we will get a list of individuals, and by calling DELETE HTTP /individual/MA101B401 we are requesting for that resource to be deleted.

Since we are on the Web, implying HTTP is used as protocol, we can access resources using nothing more than a web browser, if no authentication methods have been put in place to prevent anonymous access to the data that is.

A resource can have several different representations depending on which are supported by the system, i.e. which have been implemented in the gateway.

Figure 1 below shows the response received from the gateway when accessing the resource /domain/household/MA101B8. By default a HTML representation of the resource is returned.

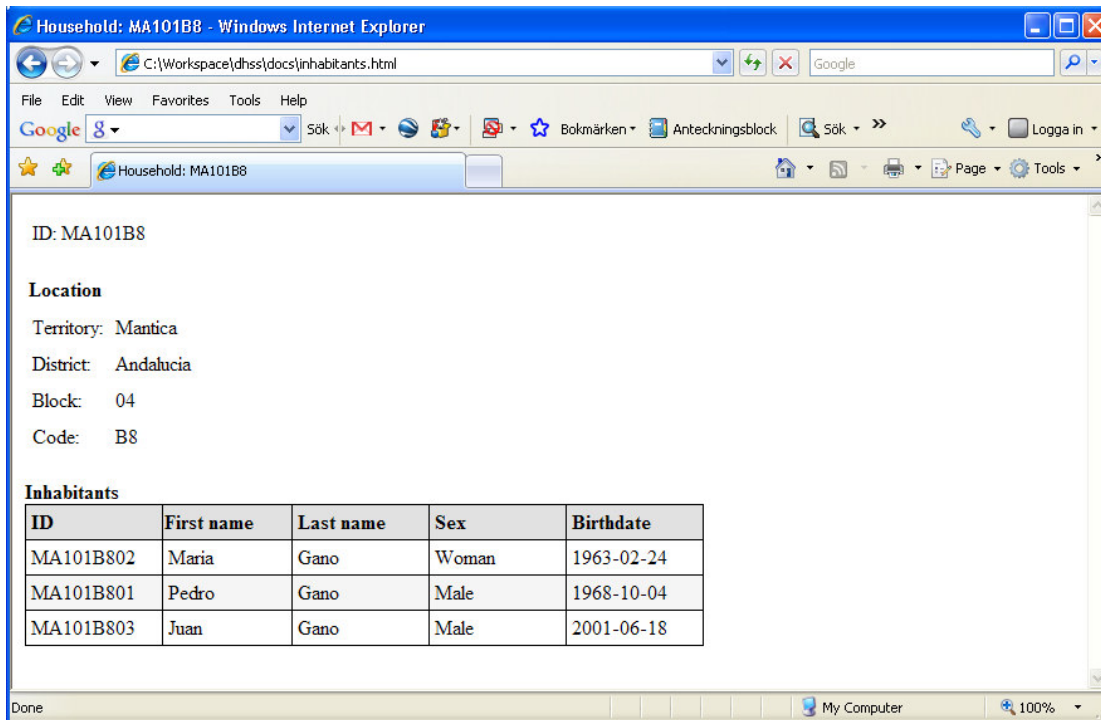


Figure 1: Requesting an HTML representation of the resource /domain/household/MA101B8.

In this case HTML was received from the gateway since that is the default representation. By appending .xml to the resource an XML representation is received instead, as can be seen in figure 2 below.

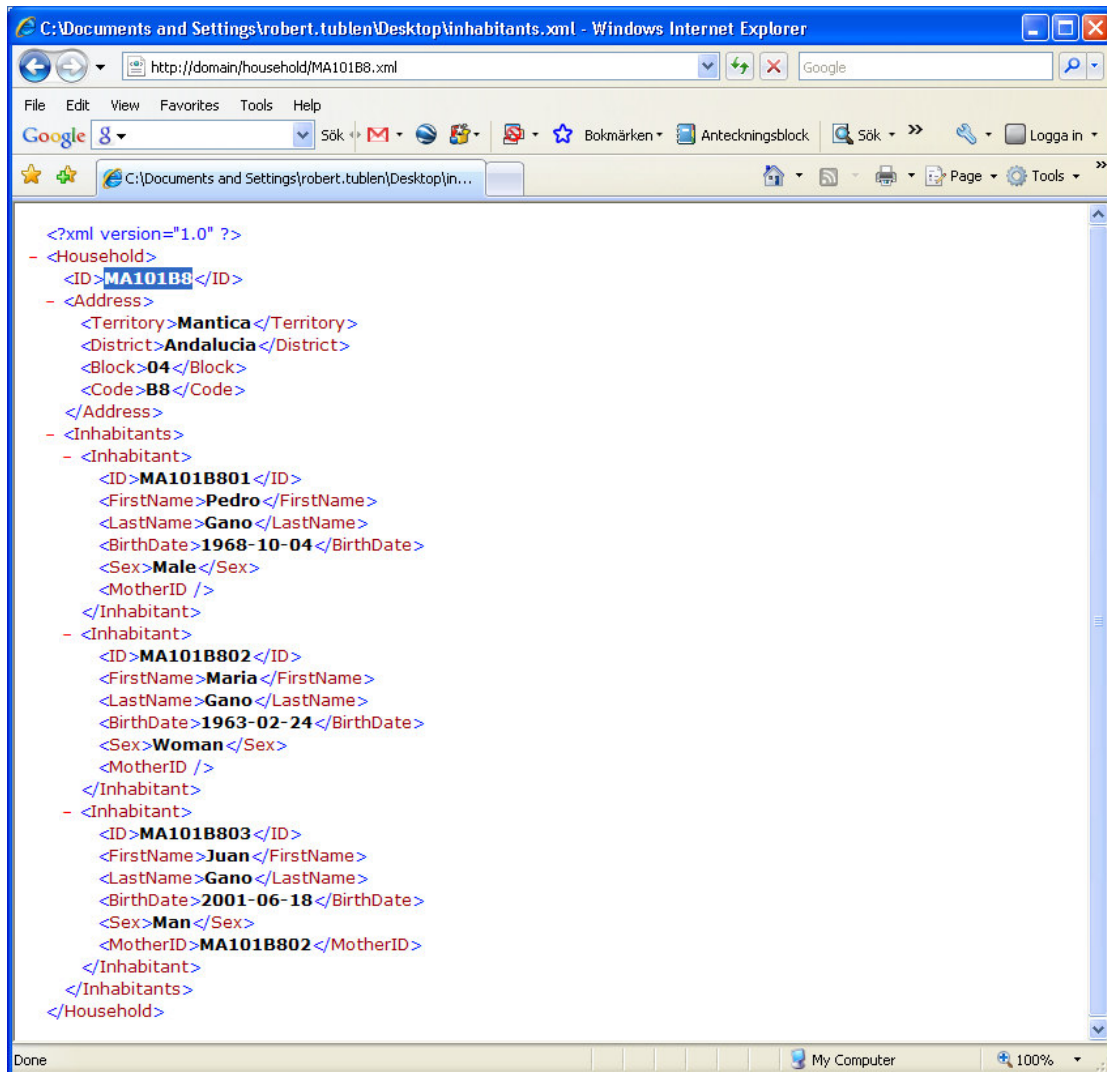


Figure 2: Requesting an XML representation of the resource /domain/household/MA101B8.

As long as the gateway has implemented the format, it can return the requested resource in the requested representation, for example PDF.

This is what representational means when referring to representational state transfer.

By following the principles of REST [5] and implementing the gateway in a RESTful [4] way, a unified interface for obtaining the data stored in the database is obtained, using only the HTTP verbs GET, PUT, POST, and DELETE to specify the requested action.

7.1.1.2 DHSS Gateway SDD

A SDD (Software Design Description) [7] document is an IEEE standard that specifies the form of the document used to specify system architecture and application design in a software related projects.

This document contains, among other things, class diagrams depicting the domain objects implemented in the gateway. It describes the chosen architecture and lists the components making out the system.

The document describes the advantages of constructing the system in a RESTful [4] way, exposing the data guarded by the gateway via a unified HTTP-based API.

See Appendix I for more information.

7.1.1.3 DHSS Gateway API Documentation

This document specifies the available resources exposed by the gateway, together with their HTTP verbs used for manipulating or acquiring a representation of each resource.

This is only a draft intended to be further developed.

See Appendix II for more information.

7.1.1.4 DHSS Gateway Client

The client runs in a standard web browser and is built using the Flex framework, which means that the application is Flash-based. This in turn makes it possible to build slick user interfaces which are user friendly and responsive.

Figure 3 below displays an image depicting the registration of an observation. An observation is connected to a household and is performed within a census. It's performed on a specific date and by a specific person deemed as the observer.

Register observation [X]

Household
SA102B4

Census
Census 1, 2008

Date
20 / 2 / 2008

Observer
Maria Teresa Gonzales

Abort Begin

Figure 3: Starting a new observation.

There are two main parts to an observation: the observation of attributes pertaining to the household, and details pertaining to residents therein.

Figure 4 below depicts the registration of household attributes and figure 5 shows a form used for registering residents and related information.

Register observation [X]

Household	Date
SA102B4	20 / 2 / 2008
Census	Observer
Census 1, 2008	Maria Teresa Gonzales

Comments

Abort Register

Household Residents

Walls
Select material

Floors
Select material

Water facilities
Select supply

Lavatory facilities
Select facility

Bedrooms

Electricity

Figure 4: Registering information pertaining to household.

Register resident X

Type: **Baseline** | Date: 20 / 2 / 2008

Global ID: SA1028401

First name: Maria

Last name: Ortega

Sex: **Woman**

Birthdate: 8 / 6 / 1974

Head of household

Relation to head of household: -

Education level: **Professional**

Occupation: **Unemployed**

Migrated from: -

Migration reason: -

Other:




Figure 5: Registering information pertaining to residents.

8 Further Development

This chapter lists suggestions for further development.

8.1 Digitalize forms for use by disconnected devices

In order to use disconnected devices, such as laptops or PDAs, for collecting demographic and health related information, replacing the paper forms currently being used, digitalized versions of the forms needs to be developed.

This would make it possible to implement constraints in the forms used when collecting the information, preventing simple errors from being entered and thereby increasing the quality of the data.

In order to support the use of digitalized forms one would also need to implement the ability to check out a subset of data for use in the field, as suggested by section 8.2.

8.2 Enabling checkout of data

To enable the use of disconnected devices, such as laptops and PDAs, for use in the field when collecting information one would need the ability to check out a subset of data.

This would allow the digitalized forms described in section 8.1 to implement constraints that can only be implemented when having access to historical data.

8.3 Implement authorization, authentication, and auditing

In order to prevent unauthorized users from accessing sensitive information, and performing actions they are not allowed to perform, authorization and authentication features would need to be implemented.

Also, in order to track who does what, auditing features would need to be added to the system.

9 Conclusion

The project started out two years ago with the purpose of implementing a simple web-based front-end to a database storing demographic, health, and socioeconomic information.

The database had been created by previous students working on their thesis at CIDS and the purpose of the web front-end was to supply the data entry department with simple forms for entering and updating information stored in said database, and for researchers to more easily access information therein.

However, as developers often do when scrutinizing work done by others, feeling unsatisfied with the new database design, additional time was spent working on it, delaying the work that was originally planned.

After having struggled for a couple of months, redoing much of the work that had previously been done, including the requirements elicitation, the new improved database design was complete and a good understanding of the domain had been acquired in the process.

After working with several prototypes, both web-based and Microsoft Access 2002 based, time finally ran out and I had to return back home.

Once back, the work was put on hiatus, having started my career as a system developer, finding it hard to set aside time to continue the work that had been done.

After a number of failed attempts to get the project back on track, it finally got revamped with a new solution and a new goal.

Thanks to the patients of my supervisor who approved numerous extended deadlines, the project was finally completed.

The intention is to continue the development of the DHSS Gateway and DHSS Gateway Client until a fully functional system is available for use by organizations all over the world running DSS sites.

10 References

- [1] Demographic Surveillance System Database
Nils Assarsson, Tommy Ljunggren, Lund University, 2003
- [2] A Conceptual Description of the Database version 1.0
M.Pedersen, A.Mårtensson
- [3] CIDS, “Centre for Demographic and Health Research in Leon, Nicaragua (informative folder), CIDS, 2003
- [4] RESTful Web Services
Leonard Richardson & Sam Ruby, 2007
ISBN: 978-0-596-52926-0
- [5] Architectural Styles and the Design of Network-based Software Architectures, Roy Thomas Fielding, 2000
<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- [6] IEEE Standard for Software Requirements Specifications
IEEE Std 830-1993
- [7] IEEE Standard for Software Design Description
IEEE 1016-1998
- [8] Grails
<http://grails.org/>
- [9] Groovy
<http://groovy.codehaus.org/>
- [10] Eclipse
<http://www.eclipse.org/>
- [11] Git
<http://git.or.cz/>

- [12] Github
<http://github.com/>
- [13] Visual Paradigm for UML Community Edition
<http://www.visual-paradigm.com/product/vpuml/communityedition.jsp>
- [14] <http://www.linkedin.com/>
- [15] <http://www.springframework.org/>
- [16] <http://www.hibernate.org/>
- [17] <http://www.adobe.com/products/flex/>

DHSS Gateway

Software Design Description



LUNDS
UNIVERSITET
Lunds Tekniska Högskola

LTH School of Engineering at Campus Helsingborg

Revisions

Date	Name	Version	Description
2008-11-18	Robert T.	1.0	Baseline set.

Definitions

Census	The procedure of systematically acquiring and recording information about the members of a given population.
Longitudinal data	Pertaining to changes in data over time.
Episode	A period in time delimited by a start and an end date.
Event	Indicates an action; that something has happened.
Web services	Software system designed to support interoperable machine-to-machine interaction over a network.
Resource-oriented architecture	Specific set of guidelines of an implementation of the REST architecture.
RESTful	Systems which follow Fielding's REST [1] principles are often referred to as RESTful.
HTTP method	HTTP defines eight methods indicating the desired action to be performed on the identified resource.

Table 1. List of definitions

Acronyms and abbreviations

API	Application Programming Interface; the public interface to a component or sub-system
CIDS	Centro de Investigación en Demografía y Salud
DSS	Demographic surveillance system; The process of defining risk and corresponding dynamics in rates of birth, deaths, and migration in a population over time.
DHSS	Demographic and Health Surveillance System; see DSS.
HTTP	Hypertext Transfer Protocol; use for retrieving inter-linked text documents.
XML	Extensible Markup Language
HTML	Hypertext Markup Language
XHTML	Extensible Hypertext Markup Language; the same as HTML but conforms to XML syntax.
REST	Representational state transfer; a style of software architecture for distributed hypermedia systems.
ROA	Resource-oriented architecture
URI	Uniform resource identifier
RPC	Remote procedure call
CRUD	Create, read, update, delete

Table 2. List of acronyms and abbreviations

Table of contents

1 Introduction	1
1.1 Design Overview	2
1.2 Requirements Traceability Matrix	6
2 System Architectural Design	6
2.1 Chosen System Architecture	6
2.2 Discussion of Alternative Designs	6
2.3 System Interface Description	7
3 Detail Description of Components	7
3.1 DHSS Gateway	7
3.1.1 Resources.....	7
3.1.1.1 <i>Census</i>	7
3.1.1.2 <i>Observation</i>	7
3.1.1.3 <i>Observer</i>	8
3.1.1.4 <i>Household</i>	8
3.1.1.5 <i>Individual</i>	9
3.1.1.6 <i>Pregnancy episode</i>	9
3.1.1.7 <i>Resident episode</i>	10
3.1.1.8 <i>Death event</i>	10
3.1.1.9 <i>Birth event</i>	10
3.1.1.10 <i>Abortion event</i>	10
3.1.1.11 <i>In-migration event</i>	10
3.1.1.12 <i>Out-migration event</i>	11
3.1.1.13 <i>Territory</i>	11
3.1.1.14 <i>District</i>	11
3.1.2 Use cases	11
3.1.3 Class diagrams	13
4 User Interface Design	15
4.1 Description of the User Interface	15
4.1.1 Screen Image	15
4.1.2 Objects and Actions	15
5 Additional Material	15
6 References	15

1 Introduction

This document describes the software architecture for DHSS Gateway, a middle tier server application acting as a gateway to a database storing demographic and health surveillance (DHSS) data.

The gateway exposes an API consisting of a number of web services, used for retrieving and manipulating demographic, health, and socioeconomic data.

These services have been implemented using a resource-oriented architecture, in accordance with the principles of REST [1], exposing data as resources, each accessible via at least one unique URI, through a uniformed interface.

By using the HTTP methods for manipulating resource we gain the advantage of a uniform interface which simplifies the system and makes it easier to learn and use.

The system is modelled according to current procedures at CIDS [3], in the city of León, Nicaragua, where a continuous monitoring system has been in place since 1993, tracking demographic, health and socioeconomic events for a select number of households throughout the municipality.

The primary goal and purpose of the new system is to be able to replace the one currently being used, with the aim of increasing the quality of the data that is being collected and stored in the database, as well as simplifying the extraction of data used for research and for creating public health planning programs.

The secondary goal is to ease the work load for those involved in collecting, validating, entering and using the data.

However, being a prototype it is not a fully functioning system and further development is needed before it can be put into production.

This document is the result of a thesis at LTH School of Engineering, Campus Helsingborg.

1.1 Design Overview

The system is a server application, acting as a gateway to a database storing demographic, health and socioeconomic data. The gateway exposes a number of resources as web services, making up an API which can be used to retrieve and manipulate data stored in the database.

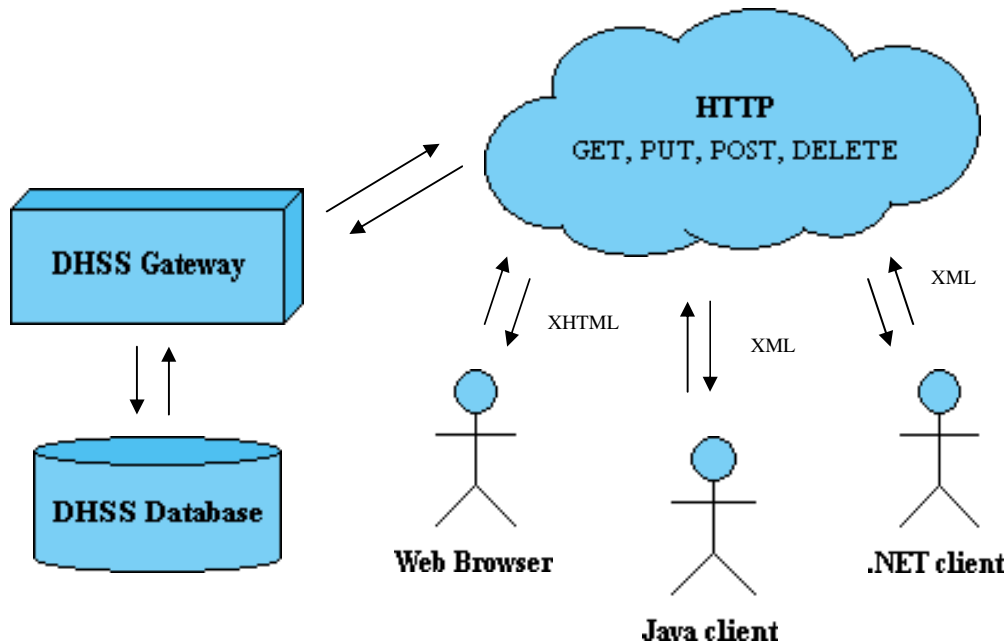


Image 1: System overview showing the flow of communication between components.

Communications with the API is done via HTTP, making it possible to access data exposed by the server, via RESTful web services, using nothing more than a simple web browser.

By default the server will serve XHTML when a client asks for a representation of a resource.

A resource is anything that is important enough to be referenced as a thing in it self. Examples of resources within the DHSS are

- individual with an ID of MA101B401,
- list of individuals,
- individuals who died before they reached the age of 5,
- list of households with wood used as material for the walls,
- pregnancy episode for individual with an ID of SB101C103.

A resource can have several different representations depending on which are supported by the server, for the specific resource. The DHSS Gateway will by default serve a XHTML representation of the requested resource if the desired representation isn't specified.

For example, by requesting the URI `http://domain/household/MA101B8` a XHTML representation of that resource will be returned to the client. To get a XML representation of the same resource, append `.xml` to the URI, e.g. `http://domain/household/MA101B8.xml`.

Image 1 below shows the response received from the gateway when accessing the resource located at URI `http://domain/household/MA101B8`. By default a XHTML representation of the resource is returned.

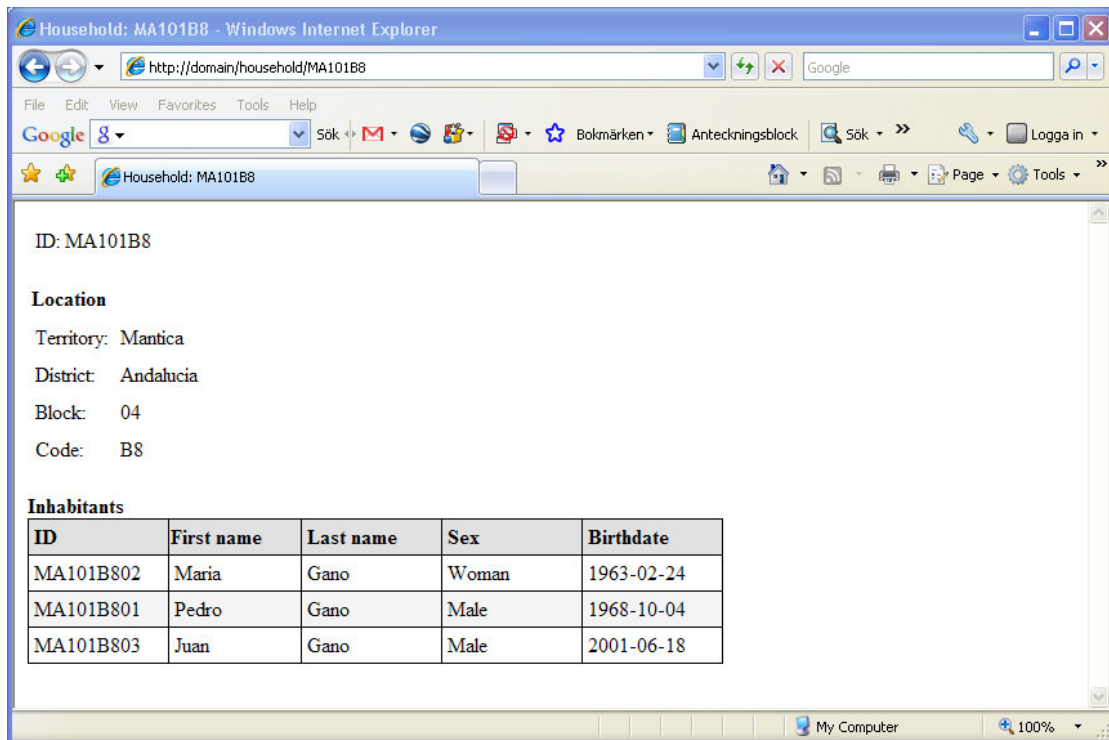


Image 2: Requesting an XHTML representation of the resource `/domain/household/MA101B8`.

In this case XHTML was received from the gateway since that is the default representation. By appending `.xml` to the resource an XML representation is received instead, as can be seen in image 2 below.

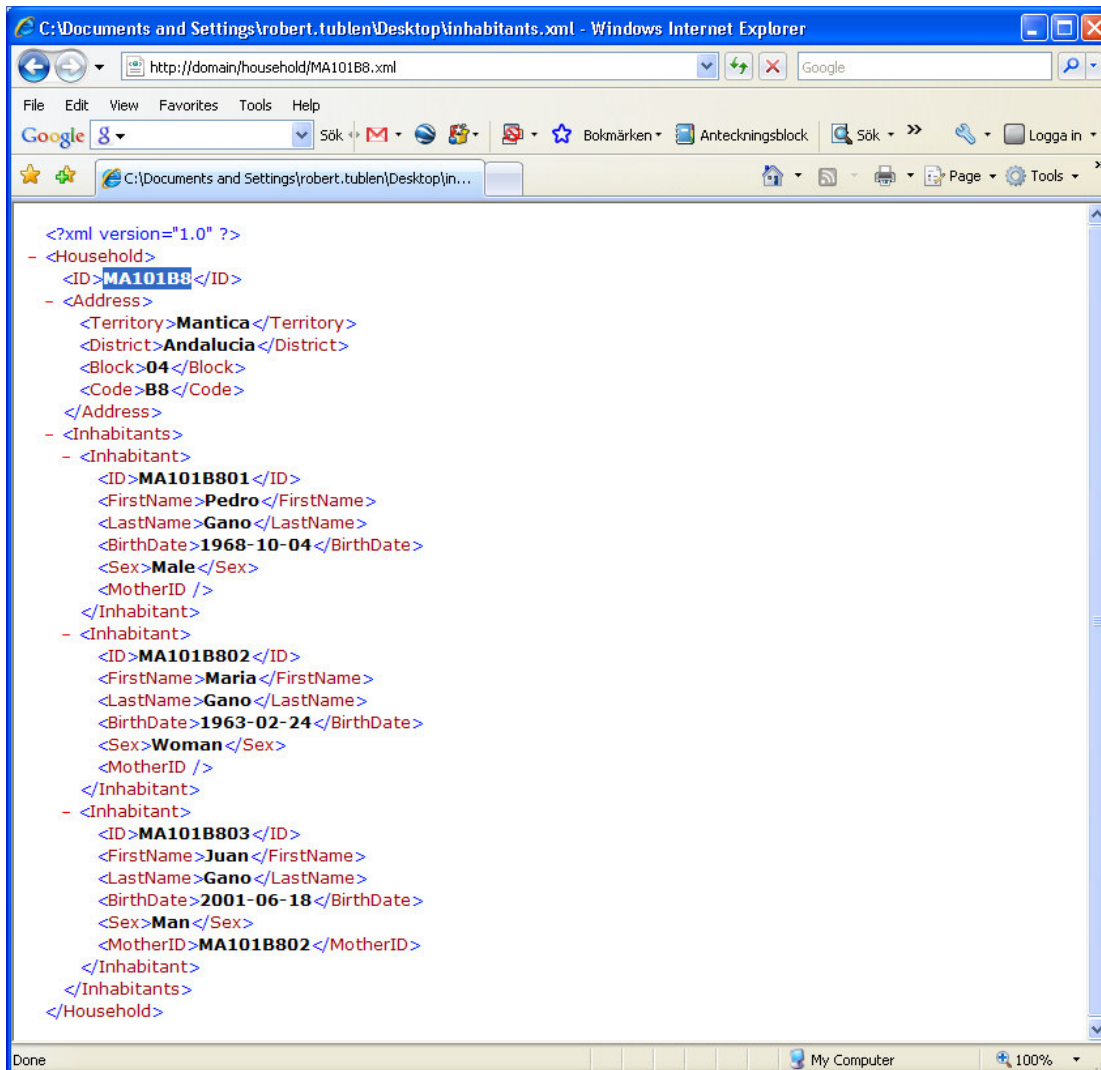


Image 3: Requesting an XML representation of the resource /domain/household/MA101B8.

As long as the gateway has implemented the ability, it can return the requested resource in the requested representation.

This is what representational means when referring to representational state transfer.

Instead of appending .xml to the URI, the entity type Content-Type located in the header of the request can be set to xml/text, which will indicate to the server that a XML representation is desired.

Clients acting as front-end applications will most probably want to communicate with the server using XML.

Exposing the data stored in the database in a RESTful way implies that the HTTP methods be used for retrieving and manipulating the data. HTTP has a total of 7 methods, of which 4 are used for retrieving, inserting, updating and deleting data: GET, PUT, POST, and DELETE

The gateway does not implement any logic for the rest of the HTTP methods.

By using these to specify the desired action, instead of specifying it in the URI, which is a common practice, the interface becomes more intuitive and easier to use.

Instead of `http://domain/individual/MA104B401?action=delete` we get HTTP DELETE `http://domain/individual/MA104B401`.

The URI `http://domain/individual/MA104B401`, when requested using a web browser, i.e. HTTP GET is used, will retrieve a XHTML representation of the individual with the ID MA104B401. Similarly, by requesting `http://domain/household/MA104B4`, again using HTTP GET, we get a XHTML representation for the household with the ID MA104B4.

By implementing the web services in this way, users of the API can easily discern how to get a representation for a household resource as well as any other resources exposed by the system.

By following the principles of REST [5] and implementing the gateway in a RESTful [4] way, a unified interface for obtaining the data stored in the database is obtained, using the HTTP verbs GET, PUT, POST, and DELETE to signal the requested action.

This simplifies the use of the API but adds the necessity for developers to think differently when designing the classes that make out the resources that expose underlying data.

```
class Household {  
    public void select() { // ... }           HTTP GET  
    public void insert() { // ... }         HTTP POST  
    public void update() { // ... }        HTTP PUT  
    public void delete() { // ... }        HTTP DELETE  
}
```

Code example 1: By following the principles of REST we get a simplified interface.

1.2 Requirements Traceability Matrix

Not applicable since it is only a prototype, a proof of concept, without any actual customer.

2 System Architectural Design

This chapter outline the architectural design decisions for the DHSS Gateway.

2.1 Chosen System Architecture

A resource-oriented architecture (ROA) was chosen for they DHSS Gateway implying the implementation of the principles laid out by REST [1], a style of software architecture for distributed hypermedia systems such as the World Wide Web.

By constructing the system in a RESTful [2] way focus is on exposing data as a set of uniquely identifiable resources which all share the same interface, that of the HTTP methods.

By building the API in this way it becomes more intuitive and easier to use, instead of using a RPC approach.

A complete system consists of three parts:

- Database
- Gateway (middle tier server)
- Client (front-end or other)

Any of the major databases in existence today can be used, such as Oracle, MySQL and SQLServer.

The implementation of a client is outside the scope of this project.

2.2 Discussion of Alternative Designs

No alternative designs have been contemplated. The chosen architecture is considered suitable enough for the problems it was constructed to solve.

2.3 System Interface Description

The system interface is made up of a number of unique URIs accessed over HTTP and HTTPS, which implement some or all of the HTTP methods for retrieving, inserting, updating, and deleting data from the underlying database.

Together they form a simple to use API to the system. This API specified in the document DHSS Gateway API.

3 Detail Description of Components

3.1 DHSS Gateway

3.1.1 Resources

This section lists the available resources in the DHSS Gateway and gives a short description of each. This list is not a complete list of all resources in the system. Some resources would normally be declared as actions in other resources, such as the creating of a death event, signalling the end of a resident episode.

However, since this is a prototype only the resources of main interest are listed.

3.1.1.1 Census

Every three months a census is performed involving all households included in the DHSS operated by CIDS, collecting information such as pregnancies, births, deaths, cause of death, and migrations.

The census is performed by field workers, called observers in the system (see 3.1.1.3), who are assigned households with which to conduct interviews with the head of the family.

A census has a start and an end date, and a description.

The gateway implements full CRUD support for census.

3.1.1.2 Observation

An observation is the act of collecting demographic, health, and socioeconomic data within the time period of a census, and is performed by an observer (see 3.1.1.3).

The observation has a start and an end date which must be within the time period of the census.

If approved by the supervisor, the data collected during the observation is sent to the data department for entry into the DHSS database, by using a front-end application communicating with the DHSS gateway through the API.

However, if the supervisor finds inconsistencies in the data collected, it is sent back to the observer which visits the household in question to acquire the missing information.

Every piece of data collected and stored in the DHSS system is done in the context of an observation. That way, it's possible to trace any changes in the information making out the longitudinal data.

The gateway implements full CRUD support for observation.

3.1.1.3 Observer

An observer is a person who performs an observation (see 3.1.1.2) at a specific household included in the DHSS, collecting information such as pregnancies, births, deaths, cause of death, and migrations.

The collected information is then handed to the supervisor for inspection and approval.

The term used at CIDS for an observer is field worker.

The gateway implements full CRUD support for observer.

3.1.1.4 Household

A household is a place inhabited by a group of people making up a family, each household having a person who is considered to be the head of family, with whom the observer performs the interview.

A household is located within a block which in turn is located within a district (see [3.1.1.13].), and is given an ID that is unique within that block. The id is made up of a letter between A to Z followed by a number ranging from 1 to 9, e.g. A1.

Further more, each household is assigned a unique ID within the system itself, a global ID composed of the territory id, followed by the district id, then the block id and finally the house id, e.g. MA104B4.

By using this ID we can give each household a unique URI, such as <http://domain/household/MA104B4>.

Information related to household attributes such as the materials used for the floors, walls and ceiling, availability of hygienic factors such as access to running water and lavatory facilities, and the availability of electricity, is collected when the household is registered for the first time in the system.

This is normally only collected once, or has been the case so far for the DHSS currently operated by CIDS, meaning that longitudinal data is not generated for this information.

Since it is used for assessing the level of poverty for the household and the inhabitants living therein, and since it would be of interest to see how this changes over time, the DHSS Gateway has built in support for the tracking of longitudinal information for these attributes.

The gateway implements full CRUD support for household.

3.1.1.5 Individual

An individual is a person registered in the DHSS, having been so as part of an observation at a specific household. Information such as first name, last name, sex and date of birth is collected, together with current and historic pregnancy history if the individual is a woman between the age of 15 and 55.

Also registered is the current occupation, if at least 7 years old, and the current level of education, if at least 15 years old.

Each individual registered in the DHSS is assigned a unique ID by composing the global ID for the household with a number ranging from 01 to 99, e.g. MA104B401.

This is a global ID used for tracking the individual throughout the DHSS system.

The gateway implements full CRUD support for individual.

3.1.1.6 Pregnancy episode

A pregnancy episode is initiated when it is observed that a woman residing within the household being visited is pregnant. The date of conception is estimated as the start date of the pregnancy.

The end date consists of the date the woman gave birth or when the pregnancy was terminated, by giving birth or by abortion, either provoked or spontaneous, meaning she had a miscarriage.

Previous pregnancy history is also recorded the first time a woman is registered in the system, estimating the start and end dates and specifying the outcome of the pregnancy.

In the case of birth, two possible outcomes exist: live birth or still born. Method of delivery, caesarean or vaginal, is also stored for each birth.

The gateway implements full CRUD support for pregnancy episode.

3.1.1.7 Resident episode

A resident episode is the period of time an individual resides within a specific household, and can start by the individual being born into the household, or by moving in, which is referred to as in-migration in the current DHSS system.

A resident episode can end by the individual moving out of the household, referred to as out-migration in the current DHSS, or by death.

A special case exists for when setting up the baseline for the DHSS, i.e. when registering all households that are to make up the foundation for the construction of longitudinal data. In this case, a resident episode is started by a baseline setup event.

The gateway implements full CRUD support for resident episode.

3.1.1.8 Death event

A death event occurs when an individual dies. It contains a date of death and cause of it.

The gateway implements full CRUD support for death event.

3.1.1.9 Birth event

A birth event occurs when a woman has given birth to a child. It contains date of birth, outcome (live born, stillborn) and type of delivery (caesarean or vaginal).

The gateway implements full CRUD support for birth event.

3.1.1.10 Abortion event

An abortion event occurs when a pregnancy terminates through abortion and has a date of abortion and type (provoked or spontaneous).

The gateway implements full CRUD support for abortion event.

3.1.1.11 In-migration event

An in-migration event occurs when a new individual is registered as a resident in a household.

The gateway implements full CRUD support for in-migration event.

3.1.1.12 Out-migration event

An out-migration event occurs when an individual is registered as having moved out of a household.

The gateway implements full CRUD support for in-migration event.

3.1.1.13 Territory

A territory is a large area containing several districts and can span both urban and rural areas. The convention at CIDS is to use the first letter in the name of the territory to act as an id for it.

Currently there are only three territories included in the DHSS operated by CIDS in León: Mantica, Subtiava, and Perla Maria

The gateway implements full CRUD support for territory.

3.1.1.14 District

A district is an area contained within a territory (see 3.1.1.4) and can be regarded as either urban or rural. The convention at CIDS is to assign each district an id made up of a letter followed by a number between 1-9, e.g. A1.

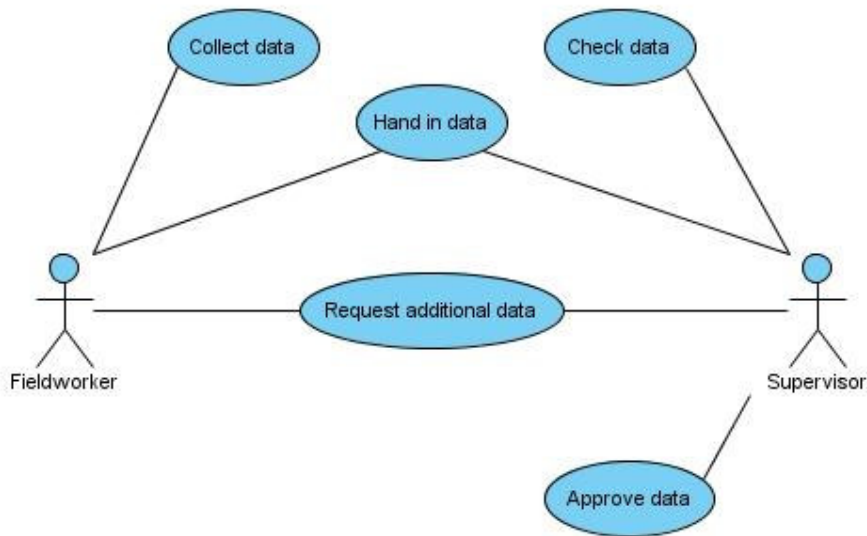
Besides being given a unique ID (unique in the territory in which the district is located), each district also has a name.

A district contains a number of blocks, numbered between 01-99, in which a number of households are located, numbered between [A-Z][1-9], e.g. A1.

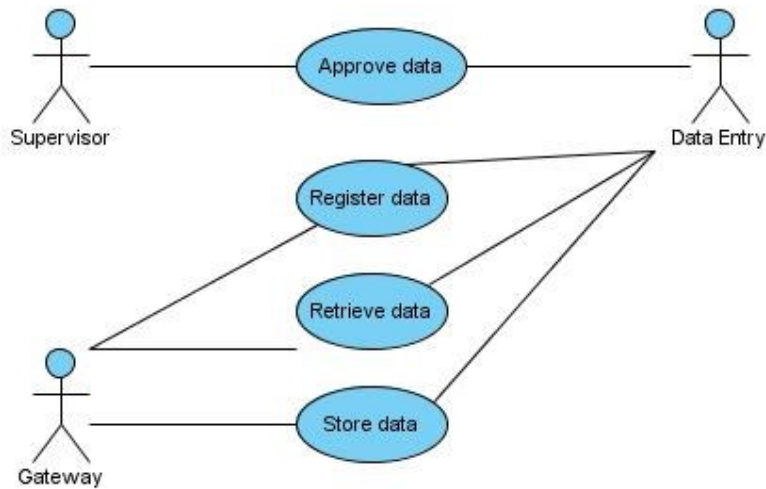
The gateway implements full CRUD support for district.

3.1.2 Use cases

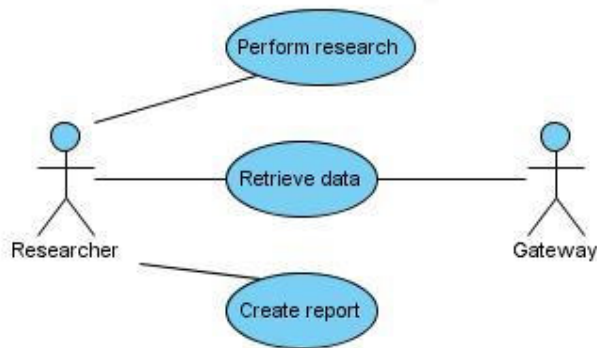
This section specifies a number of use cases for the system.



Use case 1: Use case depicting the actions and interactions performed by fieldworkers (observers) and supervisors.



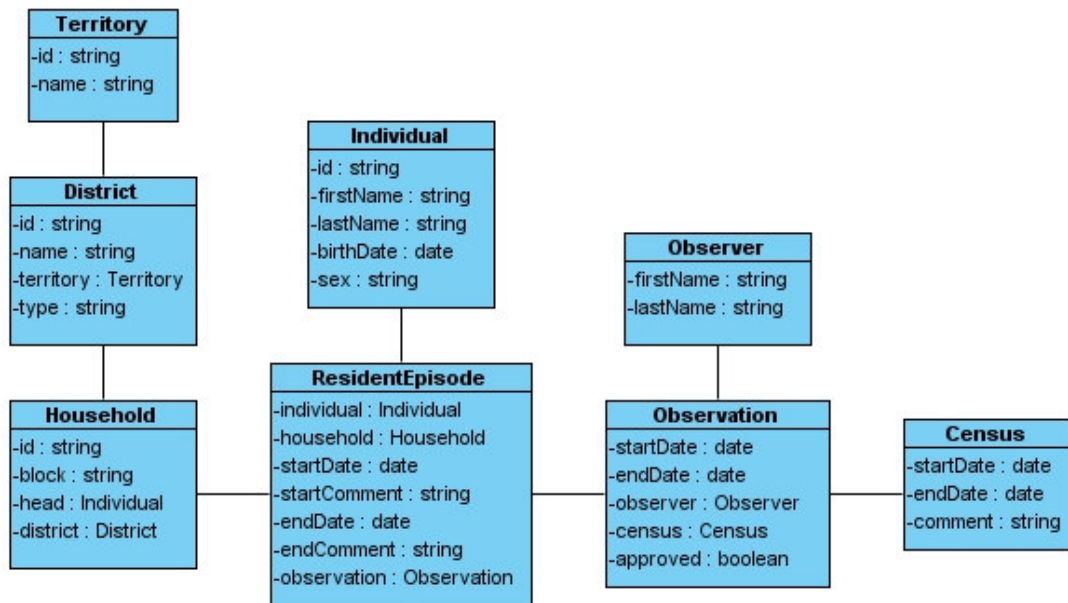
Use case 2: Use case depicting the actions and interactions performed by supervisors, data entry and the gateway.



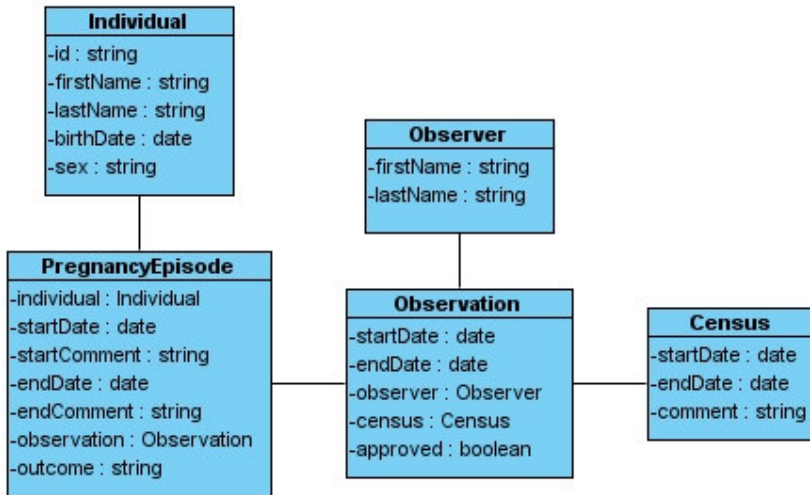
Use case 3: Use case depicting the actions and interactions performed by researchers and the gateway.

3.1.3 Class diagrams

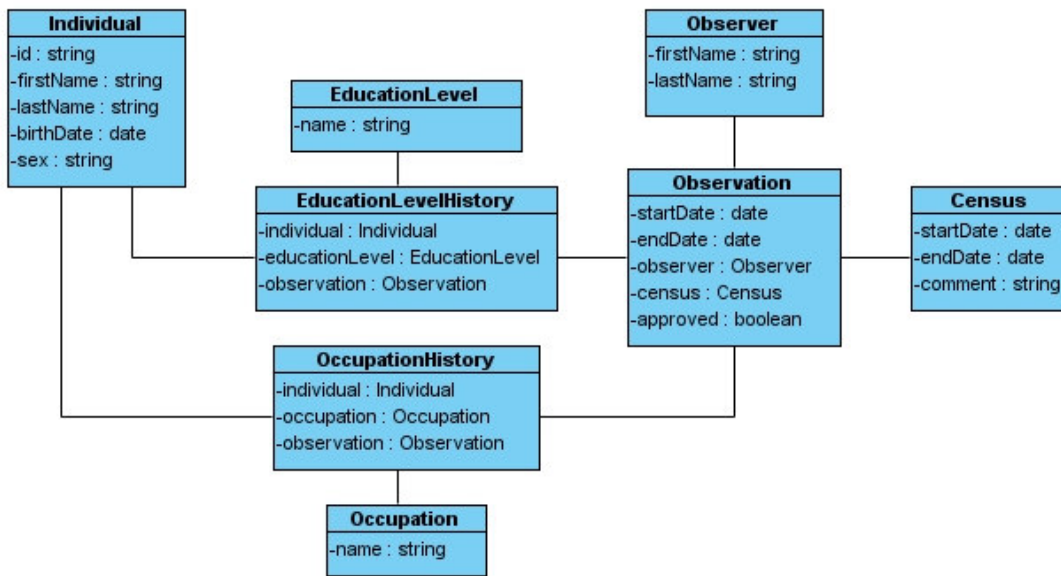
This section lists the class diagrams that were created during the construction of the system.



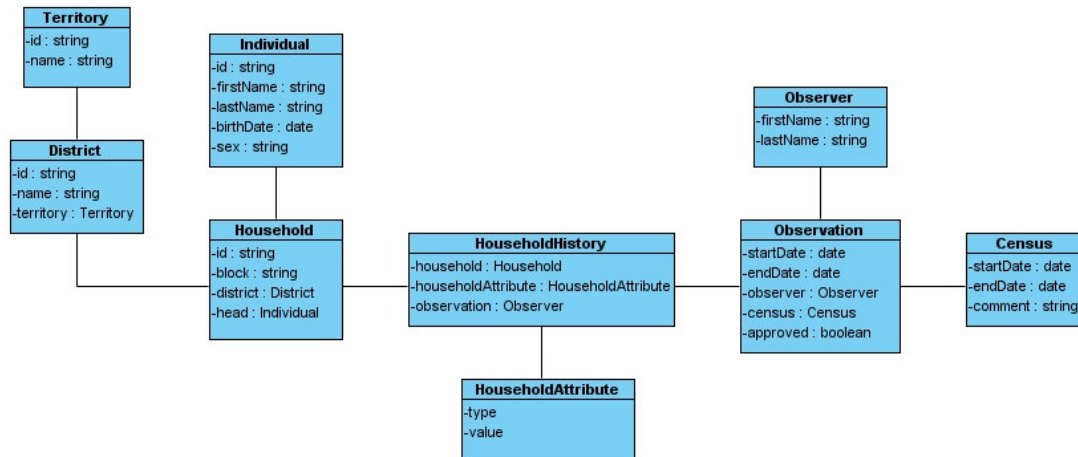
Class diagram 1: Displaying classes involved when dealing with a resident episode.



Class diagram 2: Displaying classes involved when dealing with a pregnancy episode.



Class diagram 3: Displaying classes involved when dealing with occupation and education level history.



Class diagram 4: Displaying classes involved when dealing with household attributes history.

4 User Interface Design

Not applicable since the DHSS Gateway has not graphical components except for the simple XHTML once. However, these are mostly for demonstration purposes and are not intended to be used in production and will therefore not be listed here. In a production system most of these will be inaccessible.

4.1 Description of the User Interface

Not applicable.

4.1.1 Screen Image

Not applicable.

4.1.2 Objects and Actions

Not applicable.

5 Additional Material

No additional material.

6 References

- [1] Architectural Styles and the Design of Network-based Software Architectures, Roy Thomas Fielding, 2000

<http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>

- [2] RESTful Web Services

Leonard Richardson & Sam Ruby, 2007

ISBN: 978-0-596-52926-0

- [3] CIDS, “Centre for Demographic and Health Research in Leon, Nicaragua (informative folder), CIDS, 2003

DHSS Gateway

Application Programming Interface



**LUNDS
UNIVERSITET**
Lunds Tekniska Högskola

LTH School of Engineering at Campus Helsingborg

Revisions

Date	Name	Version	Description
2008-11-19	Robert T.	1.0	Baseline set.

Acronyms and Abbreviations

API	Application Programming Interface; the public interface to a component or sub-system
CRUD	Create, read, update, delete
XML	Extensible markup language
XHTML	Extensible hipertexto markup language
HTTP methods	HTTP has a total of seven methods; GET, POST, PUT, DELETE, HEAD, TRACE, and OPTIONS.

Table 2. List of acronyms and abbreviations

Table of contents

1 Purpose	1
2 Open Issues	1
3 Resources	2
3.1 Census	2
3.1.1 Retrieve census	2
3.1.2 Retrieve list of censuses	2
3.1.3 Create census	3
3.1.4 Update census	3
3.1.5 Delete census	3
3.2 Observation	4
3.2.1 Retrieve observation.....	4
3.2.2 Retrieve list of observations	5
3.2.3 Create observation.....	5
3.2.4 Update observation	5
3.2.5 Delete observation.....	5
3.3 Observer	5
3.3.1 Retrieve observer	5
3.3.2 Retrieve list of observers	6
3.3.3 Create observer	6
3.3.4 Update observer	7
3.3.5 Delete observer	7
3.4 Household	7
3.4.1 Retrieve household.....	7
3.4.2 Retrieve list of households	8
3.4.3 Create household.....	9
3.4.4 Update household.....	9
3.4.5 Delete household.....	9
3.5 Individual	9
3.5.1 Retrieve individual	10
3.5.2 Retrieve list of individuals	10
3.5.3 Create individual	11
3.5.4 Update individual.....	11
3.5.5 Delete individual.....	11
3.6 Territory	11
3.6.1 Retrieve territory	12
3.6.2 Retrieve list of territories	12
3.6.3 Create territory	13
3.6.4 Update territory	13
3.6.5 Delete territory	13
3.7 District	13

3.7.1 Retrieve district.....	13
3.7.2 Retrieve list of districts	14
3.7.3 Create district.....	14
3.7.4 Update district.....	15
3.7.5 Delete district.....	15
3.8 In-migration event	15
3.8.1 Create in-migration event.....	15
3.8.2 Update in-migration event.....	15
3.8.3 Delete in-migration event.....	16
3.9 Out-migration event	16
3.9.1 Create out-migration event.....	16
3.9.2 Update out-migration event.....	16
3.9.3 Delete out-migration event.....	16
3.10 Death event.....	17
3.10.1 Create death event.....	17
3.10.2 Update death event.....	17
3.10.3 Delete death event.....	17
3.11 Pregnancy event.....	17
3.11.1 Create pregnancy event.....	18
3.11.2 Update pregnancy event.....	18
3.11.3 Delete pregnancy event.....	18
3.12 Birth event.....	18
3.12.1 Create birth event.....	18
3.12.2 Update birth event.....	19
3.12.3 Delete birth event.....	19
3.13 Abortion event.....	19
3.13.1 Create abortion event	19
3.13.2 Update abortion event	19
3.13.3 Delete abortion event	20

7 Purpose

This document specifies the application programming interface (API) for DHSS Gateway, a server based solution used for retrieving and manipulating demographic, health, and socioeconomic data stored in a database.

The API consists of a number of web services exposing data in the underlying database as resources, each implementing methods responding to at least one of the HTTP methods GET, PUT, POST, or DELETE, corresponding to what is designated CRUD.

This document is the result of a thesis at LTH School of Engineering, Campus Helsingborg.

8 Open Issues

Being a prototype, the API for the DHSS Gateway has not been fully implemented. The purpose of this document is, at this stage, to give the reader an overview over the resources and available methods exposed by these.

9 Resources

This section lists the available resources in DHSS Gateway, each one containing the following information:

- URI
- HTTP Method
- Parameters
- Representations
- Error codes
- Example usage

9.1 Census

A census is the procedure of systematically acquiring and recording information about the members of a given population, and it has a description, start date and end date.

9.1.1 Retrieve census

HTTP Method GET
URI /census/<census id>
Representations XHTML, XML

Parameter	Description
census id	Census id

Error code	Description
200 OK	Specified census found.
404 Not Found	Specified census not found.

Example 1

Invocation HTTP GET /somedomain/census/1.xml
Explanation Retrieve XML representation of census with an ID of 1.
Head return 200 OK
Body return <Census id="1">
<Description>Census period 1</Description>
<StartDate>2008-01-01</StartDate>
<EndDate>2008-03-01</ EndDate >
</Census>

9.1.2 Retrieve list of censuses

HTTP Method GET
URI /censuses
Representations XHTML, XML

Parameter	Description
N/A	

Error code	Description
N/A	

Example 1

Invocation	HTTP GET /somedomain/census/1.xml
Explanation	Retrieve XML representation of list of censuses.
Head return	200 OK
Body return	<Censuses> <Census id="1"> <Description>Census period 1</Description> <StartDate>2008-01-01</StartDate> <EndDate>2008-03-01</ EndDate > </Census> </Censuses>

9.1.3 Create census

HTTP Method	POST
URI	/census
Representations	N/A

Error code	Description
201 OK	The new census was successfully created.
400 Bad Request	The server was unable to create the census.

9.1.4 Update census

HTTP Method	PUT
URI	/census/<census ID>
Representations	N/A

Error code	Description
200 OK	The specified census was successfully updated.
400 Bad Request	The server was unable to update the specified census.

9.1.5 Delete census

HTTP Method	DELETE
URI	/census/<census ID>
Representations	N/A

Error code	Description
201 OK	The specified census was successfully deleted.
400 Bad Request	The server was unable to delete the specified census.

9.2 Observation

An observation is performed by an observer. An observation can belong to a census, but can also be performed outside of a census, for instance when setting up a baseline.

Whether it is setting up a baseline or performing census, everything is performed within the context of an observation.

9.2.1 Retrieve observation

HTTP Method	GET
URI	/observation/<id>
Representations	XHTML, XML

Parameter	Description
observation id	Observation id

Error code	Description
200 OK	Specified observation found.
404 Not Found	Specified observation not found.

Example 1

Invocation	HTTP GET /somedomain/observation/1.xml
Explanation	Retrieve XML representation of observation with an ID of 1.
Head return	200 OK
Body return	<Observation id="1"> <CensusID>3</CensusID> <ObserverID>2</ObserverID> <StartDate>2008-01-01</StartDate> <EndDate>2008-03-01</ EndDate > <Approved>True</Approved> </Observation>

9.2.2 Retrieve list of observations

Invocation	HTTP GET /somedomain/observations.xml
Explanation	Retrieve XML representation of list of observations.
Head return	200 OK
Body return	N/A

9.2.3 Create observation

HTTP Method	POST
URI	/observer/<observer ID>/observation
Representations	N/A

Error code	Description
201 OK	The observation was successfully created.
400 Bad Request	The server failed to create the observation.

9.2.4 Update observation

HTTP Method	PUT
URI	/observer/<observer ID>/observation/<observation ID>
Representations	N/A

Error code	Description
201 OK	The specified observation was successfully updated.
400 Bad Request	The server failed to update the specified observation.

9.2.5 Delete observation

HTTP Method	DELETE
URI	/observer/<observer ID>/observation/<observation ID>
Representations	N/A

Error code	Description
201 OK	The specified observation was successfully deleted.
400 Bad Request	The server failed to delete the specified observation.

9.3 Observer

An observer is a person who performs the census, visiting selected households to gather information. An observer has a first and a last name.

9.3.1 Retrieve observer

HTTP Method	GET
URI	/observer/<observer id>
Representations	XHTML, XML

Parameter	Description
observer id	Observer id

Error code	Description
200 OK	Specified observer found.
404 Not Found	Specified observer not found.

Example 1

Invocation	HTTP GET /somedomain/observation/1.xml
Explanation	Retrieve XML representation of observation with an ID of 1.
Head return	200 OK
Body return	<pre><Observer id="1"> <FirstName>Gilberto</FirstName > <LastName>Gano</LastName > </Observer ></pre>

9.3.2 Retrieve list of observers

HTTP Method	GET
URI	/observers
Representations	XHTML, XML

Parameter	Description
N/A	

Error code	Description
200 OK	List of observers found.
404 Not Found	No observers found.

Example 1

Invocation	HTTP GET /somedomain/observers
Explanation	Retrieve XML representation of list of observers.
Head return	200 OK
Body return	<pre><Observers> <Observer id="1"> <FirstName>Gilberto</FirstName > <LastName>Gano</LastName > </Observer> <Observer id="2"> <FirstName>Maria</FirstName > <LastName>Teresa</LastName > </Observer> </Observers></pre>

9.3.3 Create observer

HTTP Method	POST
URI	/observer

Representations N/A

Error code	Description
201 OK	The observer was successfully created.
400 Bad Request	The server failed to create the observer.

9.3.4 Update observer

HTTP Method	PUT
URI	/observer
Representations	N/A

Error code	Description
201 OK	The specified observer was successfully updated.
400 Bad Request	The server failed to update the specified observer.

9.3.5 Delete observer

HTTP Method	DELETE
URI	/observer
Representations	N/A

Error code	Description
201 OK	The specified observer was successfully deleted.
400 Bad Request	The server failed to delete the specified observer.

9.4 Household

A household is a place inhabited by a group of individuals making up a family, each household having a person who is considered to be the head of the family.

When first registered each household is assigned a globally unique ID, using the following algorithm:

Territory ID + District ID + Block code + Household code

Example: MA101B4

9.4.1 Retrieve household

HTTP Method	GET
URI	/household/<household id>
Representations	XHTML, XML

Parameter	Description
household id	Household id

Error code	Description
-------------------	--------------------

200 OK	Specified household found.
404 Not Found	Specified household not found.

Example 1

Invocation	HTTP GET /somedomain/household/ MA101B4.xml
Explanation	Retrieve XML representation of household with an ID of MA101B4.

Head return	200 OK
Body return	<pre> <Household> <ID>MA101B4</ID> <Head>MA101B401</Head> <Location> <Territory>Mantica</Territory> <District>Cococobana</District> <Block>01</Block> </Location> <Residents> <Resident> <IndividualID>MA101B401</IndividualID> </Resident> <Resident> <IndividualID>MA101B402</IndividualID> </Resident> <Resident> <IndividualID>MA101B403</IndividualID> </Resident> </Residents> </Household> </pre>

9.4.2 Retrieve list of households

HTTP Method	GET
URI	/households
Representations	XHTML, XML

Parameter	Description
district id	District id
territory id	Territory id

Error code	Description
200 OK	Specified household found.
404 Not Found	Specified household not found.

Example 1

Invocation	HTTP GET /somedomain/households
-------------------	---------------------------------

Explanation	Retrieve XML representation of list of households.
Head return	200 OK
Body return	<Households> <Household> (See 2.4.1, example 1) </Household> </Households>

9.4.3 Create household

HTTP Method	POST
URI	/household
Representations	N/A

Error code	Description
201 OK	The household was successfully created.
400 Bad Request	The server failed to create the household.

9.4.4 Update household

HTTP Method	PUT
URI	/household/<ID>
Representations	N/A

Error code	Description
201 OK	The specified household was successfully updated.
400 Bad Request	The server failed to update the specified household.

9.4.5 Delete household

HTTP Method	DELETE
URI	/household/<ID>
Representations	N/A

Error code	Description
201 OK	The specified household was successfully deleted.
400 Bad Request	The server failed to delete the specified household.

9.5 Individual

An individual is a person that has been registered in the system as part of being a member of a previously registered household and has a first name, last name, sex and a date of birth.

When registered for the first time an individual is assigned a globally unique ID, meaning it is global within the system.

The following algorithm is used for generating the global ID:

Territory ID + District ID + Block ID + Household ID + [01-99]

The block code is a serial number between the ranges 01-99.

Example of an id is MA104B401.

9.5.1 Retrieve individual

HTTP Method	GET
URI	/individual/<individual id>
Representations	XHTML, XML

Parameter	Description
individual id	Individual id

Error code	Description
200 OK	Specified individual found.
404 Not Found	Specified individual not found.

Example 1

Invocation	HTTP GET /somedomain/individual/MA101B401.xml
Explanation	Retrieve XML representation of individual with an ID of MA101B401.
Head return	200 OK
Body return	<Individual> <ID>MA101B4</ID> <FirstName>Juan</ FirstName > <LastName>Domingo</LastName> <Sex>Male</Sex> <BirthDate>1968-06-01</BirthDate> </ Individual >

9.5.2 Retrieve list of individuals

HTTP Method	GET
URI	/individuals
Representations	XHTML, XML

Parameter	Description
N/A	

Error code	Description
200 OK	List of individuals found.
404 Not Found	No individuals found.

Example 1

Invocation	HTTP GET /somedomain/individuals
Explanation	Retrieve XML representation of list of individual.
Head return	200 OK
Body return	<individuals> <Individual> (See 2.5.1, example 1) </ Individual > </individuals>

9.5.3 Create individual

HTTP Method	POST
URI	/individual
Representations	N/A

Error code	Description
201 OK	The individual was successfully created.
400 Bad Request	The server failed to create the individual.

9.5.4 Update individual

HTTP Method	PUT
URI	/individual/<ID>
Representations	N/A

Error code	Description
201 OK	The specified individual was successfully updated.
400 Bad Request	The server failed to update the specified individual.

9.5.5 Delete individual

HTTP Method	DELETE
URI	/individual/<ID>
Representations	N/A

Error code	Description
201 OK	The specified individual was successfully deleted.
400 Bad Request	The server failed to delete the specified individual.

9.6 Territory

A territory is a large area covering several districts, each one considered as being part of the territory. A territory can span both urban and rural areas, has a name and an ID which consist of the first letter of the name of the territory, e.g. "T".

An area residing within a territory and is considered to be either urban or rural. Each district has an ID made up of a composition of a letter between A-Z and a number ranging from 1-9, e.g. "A1".

9.6.1 Retrieve territory

HTTP Method GET
URI /territory/<territory id>
Representations XHTML, XML

Parameter	Description
territory id	Territory id

Error code	Description
200 OK	Specified territory found.
404 Not Found	Specified territory not found.

Example 1

Invocation HTTP GET /somedomain/territory/S.xml
Explanation Retrieve XML representation of territory with an ID of S.
Head return 200 OK
Body return <Territory>
<ID>MA101B4</ID>
<Name>Subtiava</Name>
</Territory >

9.6.2 Retrieve list of territories

HTTP Method GET
URI /territories
Representations XHTML, XML

Parameter	Description
N/A	

Error code	Description
200 OK	List of territories found.
404 Not Found	No territories found.

Example 1

Invocation HTTP GET /somedomain/territories
Explanation Retrieve XML representation of a list of territories.
Head return 200 OK
Body return <Territories>
<Territory>
(See 2.6.1, example 1)
</Territory >
</Territories>

9.6.3 Create territory

HTTP Method	POST
URI	/territory
Representations	N/A

Error code	Description
201 OK	The territory was successfully created.
400 Bad Request	The server failed to create the territory.

9.6.4 Update territory

HTTP Method	PUT
URI	/territory/<ID>
Representations	N/A

Error code	Description
201 OK	The specified territory was successfully updated.
400 Bad Request	The specified server failed to update the territory.

9.6.5 Delete territory

HTTP Method	DELETE
URI	/territory/<ID>
Representations	N/A

Error code	Description
201 OK	The specified territory was successfully deleted.
400 Bad Request	The specified server failed to delete the territory.

9.7 District

A district is an area residing within a territory and is considered to be either urban or rural. Each district has an ID made up of a composition of a letter between A-Z and a number ranging from 1-9, e.g. A1.

The ID is unique only within the territory to which the district belongs. A district also has a name.

9.7.1 Retrieve district

HTTP Method	GET
URI	/territory/<territory ID>/district/<district id>
Representations	XHTML, XML

Parameter	Description
district id	District id

Error code	Description
200 OK	Specified district found.

404 Not Found Specified district not found.

Example 1

Invocation HTTP GET /somedomain/territory/S/district/A1.xml
Explanation Retrieve XML representation of a district with an ID of A1, residing within the territory S.
Head return 200 OK
Body return <District>
 <ID>A1</ID>
 <Name>Santa Maria</Name>
 <TerritoryID>T</TerritoryID>
 </ District>

9.7.2 Retrieve list of districts

HTTP Method GET
URI /territory/<territory ID>/districts
Representations XHTML, XML

Parameter	Description
N/A	

Error code	Description
200 OK	At least one district found.
404 Not Found	No districts found.

Example 1

Invocation HTTP GET /somedomain/territory/S/districts
Explanation Retrieve XML representation of a list of district, residing within the territory S.
Head return 200 OK
Body return <Districts>
 <District>
 (See 2.7.1, example 1)
 </ District>
 <Districts>

9.7.3 Create district

HTTP Method POST
URI /territory/<territory ID>/district
Representations N/A

Error code	Description
201 OK	The district was successfully created.
400 Bad Request	The server failed to create the district.

9.7.4 Update district

HTTP Method	PUT
URI	/territory/<territory ID>/district/<district ID>
Representations	N/A

Error code	Description
201 OK	The specified district was successfully updated.
400 Bad Request	The server failed to update the specified district.

9.7.5 Delete district

HTTP Method	DELETE
URI	/territory/<territory ID>/district/<district ID>
Representations	N/A

Error code	Description
201 OK	The specified district was successfully deleted.
400 Bad Request	The server failed to delete the specified district.

9.8 In-migration event

An in-migration event represents the moving in to a household for an individual and is the trigger for the start of a resident episode for the said individual. The event contains the ID for the individual moving in and the date when he or she moved in.

9.8.1 Create in-migration event

HTTP Method	POST
URI	/individual/<individual ID>/in_migration_event
Representations	N/A

Error code	Description
201 OK	The n-migration event was successfully created.
400 Bad Request	The server failed to create the in-migration event.

9.8.2 Update in-migration event

HTTP Method	PUT
URI	/individual/<individual ID>/in_migration_event/<in_migration_event_id>
Representations	N/A

Error code	Description
201 OK	The specified in-migration event was successfully updated.
400 Bad Request	The server failed to update the specified in-migration event.

9.8.3 Delete in-migration event

HTTP Method	DELETE
URI	/individual/<individual ID>/in_migration_event/<in_migration_event_id>
Representations	N/A

Error code	Description
201 OK	The specified in-migration event was successfully deleted.
400 Bad Request	The server failed to deleted the specified in-migration event.

9.9 Out-migration event

An out-migration event represents the moving out of a household for an individual and is the trigger for the end of a resident episode for the individual in question. The event contains the ID for the individual who moved and the date he or she moved.

9.9.1 Create out-migration event

HTTP Method	POST
URI	/individual/<ID>/out_migration_event
Representations	N/A

Error code	Description
201 OK	The out-migration event was successfully created.
400 Bad Request	The server failed to create the out-migration event.

9.9.2 Update out-migration event

HTTP Method	PUT
URI	/individual/<individual ID>/out_migration_event/<out_migration_event_id>
Representations	N/A

Error code	Description
201 OK	The specified out-migration event was successfully updated.
400 Bad Request	The server failed to update the specified out -migration event.

9.9.3 Delete out-migration event

HTTP Method	DELETE
URI	/individual/<individual ID>/out_migration_event/<out_migration_event_id>
Representations	N/A

Error code	Description
201 OK	The specified out -migration event was successfully deleted.
400 Bad Request	The server failed to delete the specified out -migration event.

9.10 Death event

A death event represents the decease of an individual and is the trigger for the end of a resident episode for the individual in question. The event contains the ID for the deceased individual and the cause of death.

9.10.1 Create death event

HTTP Method	POST
URI	/individual/<ID>/death_event
Representations	N/A

Error code	Description
201 OK	The death event was successfully created.
400 Bad Request	The server failed to create the death event.

9.10.2 Update death event

HTTP Method	PUT
URI	/individual/<individual ID>/death_event/<death event ID>
Representations	N/A

Error code	Description
201 OK	The specified death event was successfully updated.
400 Bad Request	The server failed to update the specified death event.

9.10.3 Delete death event

HTTP Method	DELETE
URI	/individual/<individual ID>/death_event/<death event ID>
Representations	N/A

Error code	Description
201 OK	The specified death event was successfully deleted.
400 Bad Request	The server failed to delete the specified death event.

9.11 Pregnancy event

A pregnancy event represents a pregnancy and is the trigger for the start of a pregnancy episode. The event contains the ID for the woman being pregnant and an approximate conception date. It is only possible to create pregnancy events for women who have reached a minimum of 15 years of age.

9.11.1 Create pregnancy event

HTTP Method	POST
URI	/individual/<ID>/pregnancy_event
Representations	N/A

Error code	Description
201 OK	The pregnancy event was successfully created.
400 Bad Request	The server failed to create the death pregnancy.

9.11.2 Update pregnancy event

HTTP Method	PUT
URI	/individual/<individual ID>/pregnancy_event/<pregnancy event ID>
Representations	N/A

Error code	Description
201 OK	The specified pregnancy event was successfully update.
400 Bad Request	The server failed to update the specified pregnancy event.

9.11.3 Delete pregnancy event

HTTP Method	DELETE
URI	/individual/<individual ID>/pregnancy_event/<pregnancy event ID>
Representations	N/A

Error code	Description
201 OK	The specified pregnancy event was successfully deleted.
400 Bad Request	The server failed to delete the specified pregnancy event.

9.12 Birth event

A birth event represents a birth and is a trigger for the end of a pregnancy episode for the individual in question. It also acts as a trigger for the start of a resident episode for the baby which is registered as an individual in the system.

The event contains the ID for the woman who gave birth, the date of birth, the type of delivery which can be either vaginal or caesarean, and the outcome of the birth, which can be either a live birth or a still birth.

9.12.1 Create birth event

HTTP Method	POST
URI	/individual/<ID>/birth_event
Representations	N/A

Error code	Description
201 OK	The birth event was successfully created.
400 Bad Request	The server failed to create the birth event.
9.12.2 Update birth event	
HTTP Method	PUT
URI	/individual/<individual ID>/birth_event/<birth event ID>
Representations	N/A

Error code	Description
201 OK	The specified birth event was successfully updated.
400 Bad Request	The server failed to update the specified birth event.

9.12.3 Delete birth event	
HTTP Method	DELETE
URI	/individual/<individual ID>/birth_event/<birth event ID>
Representations	N/A

Error code	Description
201 OK	The specified birth event was successfully deleted.
400 Bad Request	The server failed to delete the specified birth event.

9.13 Abortion event

An abortion event represents an abortion and is a trigger for the end of a pregnancy episode for the individual in question. The event contains the ID for the woman who had the abortion, the date of abortion, the type which can be either spontaneous or provoked.

9.13.1 Create abortion event	
HTTP Method	POST
URI	/individual/<individual ID>/abortion_event/<abortion event ID>
Representations	N/A

Error code	Description
201 OK	The abortion event was successfully created.
400 Bad Request	The server failed to create the specified abortion event.

9.13.2 Update abortion event	
HTTP Method	PUT
URI	/individual/<individual ID>/abortion_event/<abortion event ID>
Representations	N/A

Error code	Description
201 OK	The specified abortion event was successfully updated.
400 Bad Request	The server failed to update the specified abortion event.

9.13.3 Delete abortion event

HTTP Method	DELETE
URI	/individual/<individual ID>/abortion_event/<abortion event ID>
Representations	N/A

Error code	Description
201 OK	The specified abortion event was successfully deleted.
400 Bad Request	The server failed to delete the specified abortion event.