



System Architecture

Hepatitis C Screening Management Information System

2018

Table of Contents

1	ABBREVIATION AND ACRONYMS.....	4
2	TABLE OF FIGURES	5
3	INDEX OF TABLES	5
4	SYSTEM ARCHITECTURE DOCUMENT PURPOSE	6
5	STOP C – BRIEF FUNCTIONAL DESCRIPTION.....	6
6	STOP C SOFTWARE ECOSYSTEM.....	6
6.1	NCDC.eHealth	7
6.2	MOH.ElimC	7
6.3	MOH.eHealth.....	8
6.4	Other Connected Modules.....	8
6.5	Ecosystem Future Vision.....	8
7	STOP C MODULES GENERAL DESCRIPTION AND FUNCTIONS.....	9
7.1	Portal	10
7.2	User Management Module.....	12
7.2.1	General Information about UM Module	12
7.2.2	Functions of UM Module	13
7.3	Analytical Module	13
7.4	Service Bus	13
7.5	Common Data Module	14
8	PLATFORM AND USED TECHNOLOGIES	15
9	SYSTEM ARCHITECTURE	17
9.1	Main Module (Portal).....	17
9.1.1	Portal Classes	17
9.1.2	Physical Infrastructure Architecture	24
9.2	User Management Module.....	25
9.2.1	General Architecture.....	25
9.2.2	Physical Infrastructure	28
9.2.3	Informational Streams	28
9.2.4	Security.....	30
9.3	Common Data Module	30

9.3.1	Common Data Module Classes	30
9.3.2	Common Data Module – Information streams	31
10	DATABASE STRUCTURE AND DEFINITIONS	32
10.1	User Management Module Database.....	33
10.1.1	Database Structure	34
10.1.2	User Management Module Database Tables and Columns.....	35
10.1.3	Table Relations	38
10.2	Central Module (Portal) Database.....	41
10.2.1	Microsoft SQL: Database Structure.....	42
10.2.2	Microsoft SQL: Database Tables and Columns	43
11	NECESSARY LIBRARIES.....	47
11.1	NPOI.....	47
11.2	ASP Chart Control	47
11.3	DevExpress	47
11.4	NHibernate	47
11.5	Bootstrap.....	47
12	HARDWARE REQUIREMENTS.....	48
12.1	Mnimum Architecture	48
12.2	Components Addresses	48
12.3	Recommendations andout Backup Procedure	49
13	INSTALLATION AND CONFIGURATIONS.....	49
14	APPENDIX A: EXPRESSION SYNTAX	50
14.1	Operators	50
14.2	Functions.....	51

1 Abbreviation and Acronyms

STOP C / Module	Hepatitis C Screening Management Information System
Analytics	Hepatitis C Data Analytics System
Portal	Portal
UMM	User Management Module
CDM	Common Data Module
SB	Service Bus / Integration Module
MOH / MoLHSA	Ministry of Labour, Healthcare and Social Affairs
ElimC	Hepatitis C Elimination Program
SSA	Social Service Agency
NCDC	National Center of Disease Control
CRA	Civil Registry Agency
SDA	Service Development Agency

2 Table of Figures

Figure 1. STOP C Software Ecosystem diagram indicating the technologies.....	7
Figure 2. STOP-C Software Ecosystem Future Vision (Estimation)	9
Figure 3. STOP-C Web-form sample page (form of personal information)	10
Figure 4. Logical Diagram of STOP-C form, data and process.....	11
Figure 5. High-level business process diagram of STOP-C Screening forms administration	12
Figure 6. . Process-Concept diagram of Service Bus communication loop.....	14
Figure 7. STOP-C Components System Architecture	17
Figure 8. Class diagram (UML) of Portal Helper-classes	18
Figure 9. Class diagram (UML) of Portal Converter-classes	19
Figure 10. Class diagram (UML) of Portal Entity-classes.....	20
Figure 11. Class diagram (UML) of Portal Model-classes.....	22
Figure 12. Various layers of User Management Module	25
Figure 13. User Management Administration - Business Process	27
Figure 14. User Management Module - Information Streams	28
Figure 15. STOP-C components database infrastructure.....	33
Figure 16. Attributes schema.....	34
Figure 17. User rights and attribute definitions	35
Figure 18. Attribute schemas.....	40
Figure 19. Users and attribute definitions	41
Figure 20. Table structure of the Portal database.....	42

3 Index of Tables

Table 1. Helper Classes	18
Table 2. Converter classes: Entity to Model and Model to Entity	19
Table 3. Entity classes	20
Table 4. Model classes	23
Table 5. Received information streams	29
Table 6. Sent information streams	29
Table 7. Received information.....	32
Table 8. Sent information	32
Table 9. User Management Module databas tables and fields.....	35
Table 10. Connections between database tables and their types.....	39
Table 11. Portal database tables and fields.....	43
Table 12. STOP-C Ecosystem component addresses	48
Table 13. Expression syntax: operators	50
Table 14. Expression syntax: functions.....	51

4 System architecture Document Purpose

System Architecture document for the Hepatitis C Screening Management Information System¹ (STOP-C Screening) is an integral part of Development and Implementation of STOP-C system. The document describes technical architecture of the system components, specification of their interrelations and used technologies. Target group of the document are system administrator and developers, but can contain relevant information for the other parties who has access to the system and/or the document. The document contains all the needed and necessary information for the system administration, support and future development.

Current document does not cover the detailed manual for the administration interface. For this purpose, an important document are STOP-C system manuals, that are also necessary components of the system documentation:

- Hepatitis C Screening Management Information System Administrator Manual
- Manual form the electronic registration of the data related to the taking blood sample for the test under State program of Hepatitis C ²
- Manual for the electronic registration of the data related to Hepatitis C Screening ³

5 STOP C – Brief Functional Description

The purpose of the Hepatitis C Screening Information System - “STOP-C” is accounting and reporting of the Hepatitis C screening results under the Hepatitis C State Program and electronic registration of the data related to the transportation and taking of the blood sample necessary for the tests attributed to screening component under Hepatitis C State Program.

Hepatitis C Screening Information System – “STOP-C” can be reached on the following address:
<http://stop-c.moh.gov.ge>.

6 STOP C Software Ecosystem

Besides the STOP-C System, Hepatitis C Elimination software ecosystem consists of the various types of systems and modules, that are located on the various hardware and software platforms. These modules are clustered into the following four groups:

- NCDC.eHealth
- MOH.ElimC
- MOH.eHealth
- Other Modules

¹ <http://stop-c.moh.gov.ge>

² <http://stop-c.moh.gov.ge/Files/doc1.docx>

³ <http://stop-c.moh.gov.ge/Files/doc2.docx>

Figure 1 presents the STOP-C Ecosystem components and their relation, indicating also some of the technical characteristics.

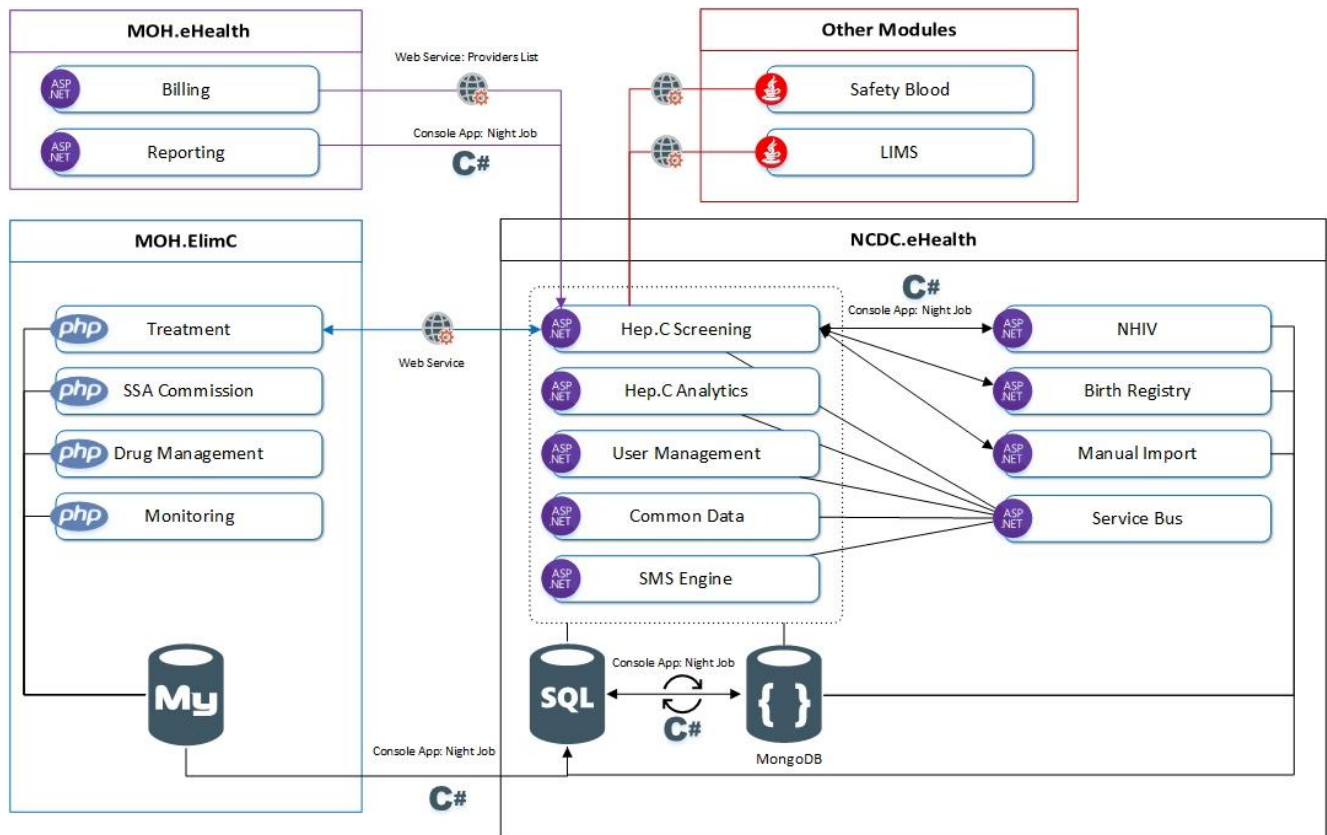


Figure 1. STOP C Software Ecosystem diagram indicating the technologies

6.1 NCDC.eHealth

These modules constitute National Center of Disease Control sub-group consisting of various systems enabling different processes and programs management, including STOP-C program. The list following:

- NCDC.eHealth.Screening: Hepatitis C Screening
- NCDC.eHealth.Analytics: Hepatitis C Analytics
- NCDC.eHealth.UM: User Management Module
- NCDC.eHealth.CM: Common Data module
- NCDC.eHealth.SB: Service Bus / Integration Module (RPC)
- NCDC.eHealth.Tools: Tools Library
- NCDC.eHealth.SMS: SMS Module (Short Message Service)

6.2 MOH.ElimC

These modules are parts of Social Service Agency modules, acting as “Elimination C” implementation and support tools. Following is the list of the modules:

- MOH.ElimC.Treatment: Treatment
- MOH.ElimC.SSA: SSA Committee
- MOH.ElimC.Drugs: Pharmaceuticals management
- MOH.ElimC.Monitoring: Monitoring

6.3 MOH.eHealth

Ministry of Labour, Health and Social Affairs of Georgia maintains independent complex ecosystem of more than 30 various systems (Health Management Information System, HMIS). However, either as the information destination and/or target for STOP-C ecosystem act only the following modules:

- MOH.eHealth.Billing: Financial Module
- MOH.eHealth.Reporting: Reporting Module

6.4 Other Connected Modules

The ecosystem contains also additional modules, which participate in STOP-C business-processes:

- SafeBlood: eSystem form Safe Blood program ⁴
- LMIS: Laboratory Management Information System

6.5 Ecosystem Future Vision

It is worth to mention that the current hardware-software ecosystem represents an example of decentralized, heterogeneous ecosystem solution, which among the positive impact, has some drawback. Two of them are especially obvious:

1. Redundant applications and services
 - a. Some critical data and service are located in physically and sometime even technologically different environments, that prevents data unification, seamless data interchange and increases administration time and needed resources.
2. Need of vast software integrations
 - a. In case of remote applications, it is necessary to integrate and due to differences in technology frameworks, the only solution is web-service. As a result, vast data is interchanged though internet (although secure channels), which affects ecosystem overall performance.

Because of these and some other factors, ecosystem modification is plausible to the direction of centralization and homogenization. **Figure 2** presents possible future vision of STOP-C ecosystem development. Unlike the current ecosystem:

1. The applications of similar types and functions are located in one infrastructural environment and use shared resources (network infrastructure, shared databases, common directories, server infrastructure, etc.)
2. The main components of the ecosystem are based on the same or similar technologies (Microsoft .NET Framework ⁵, Microsoft .NET Core ⁶).

⁴

<http://www.ncdc.ge/Pages/User/LetterContent.aspx?ID=384af120-c396-400e-82ec-86cbca75c6ea&language=en-US> (available only in Georgian)

⁵ <https://www.microsoft.com/net/download/dotnet-framework-runtime>

⁶ <https://docs.microsoft.com/en-us/dotnet/core>

- a. This significantly decreases time and cost for information and data exchange, meanwhile reducing operational time and necessary other resources, which enables increased performance of the ecosystem as a holistic solution.

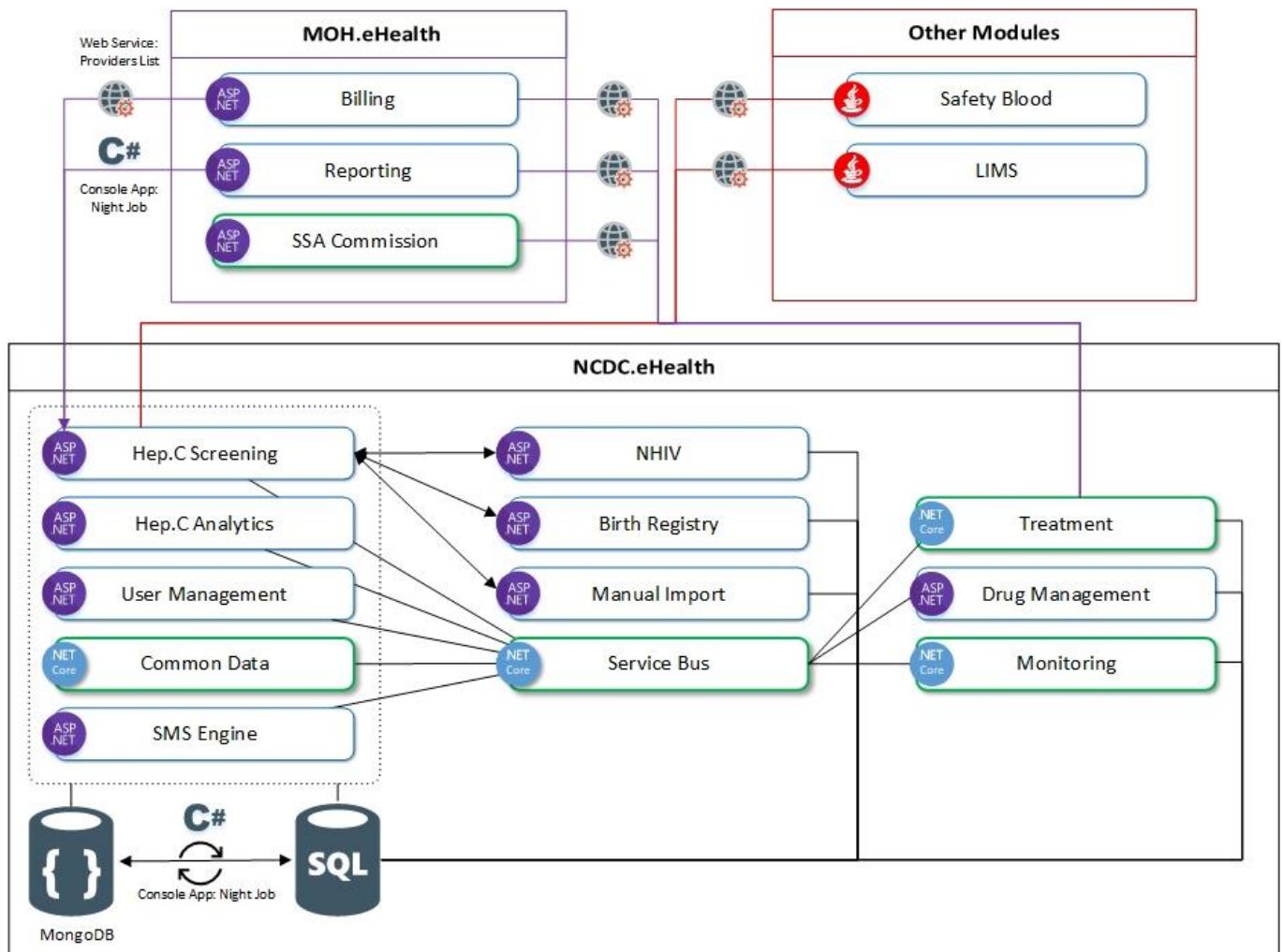


Figure 2. STOP-C Software Ecosystem Future Vision (Estimation)

7 STOP C Modules General description and Functions

STOP-C solution consists of 3 main and 3 auxiliary modules:

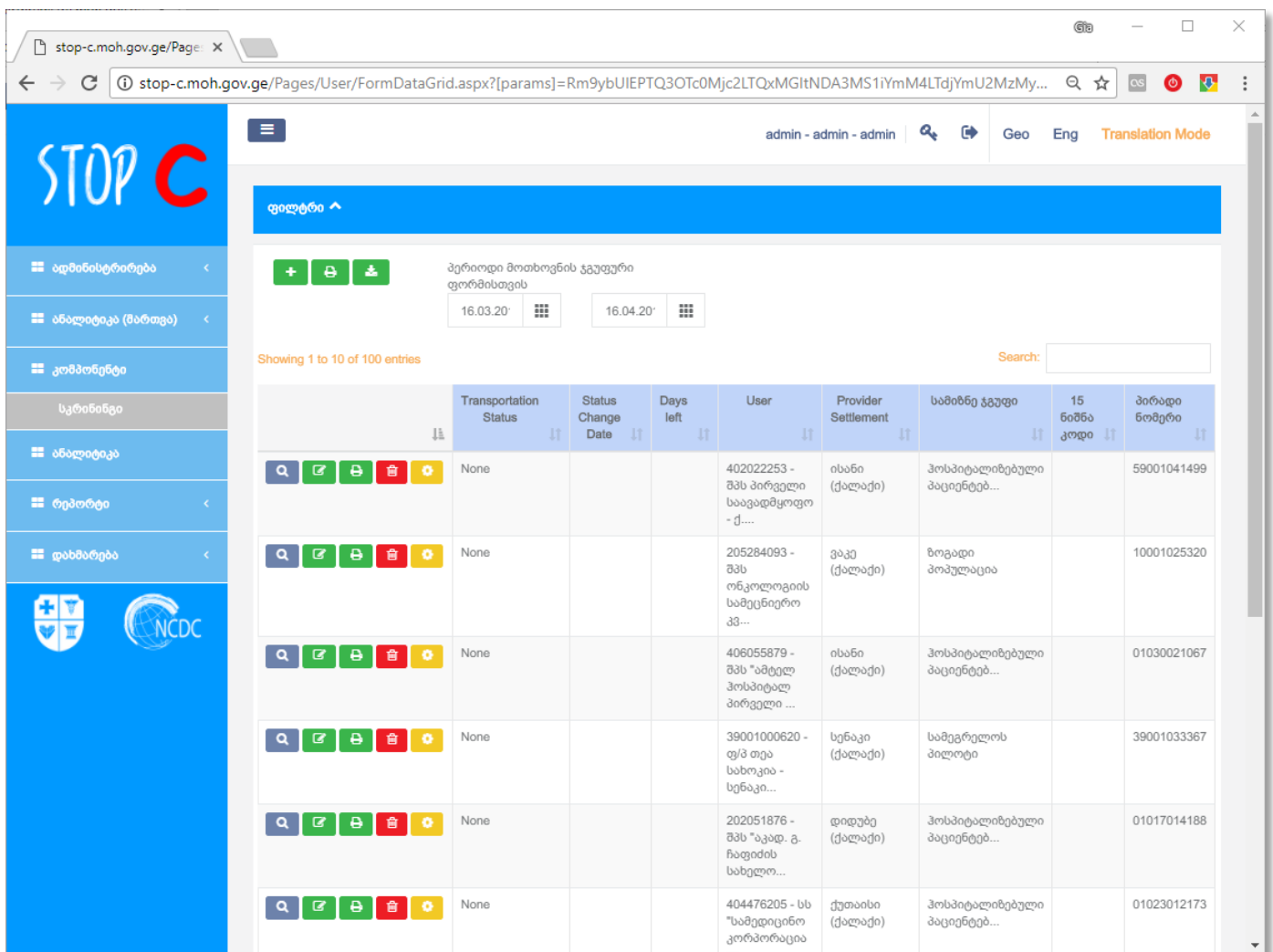
- Main modules
 - Central Module / Portal
 - STOP-C primary functions, navigation tools
 - Analytical Module ⁷
 - Dynamic reporting and analytics of the data related with Hepatitis Screening and the other associated modules
 - User Management Module (UMM)

⁷ Description of the Analytical Module is not part of the current document as presented in the separate document system architecture resource: [STOP-C Analytics: System Architecture](#)

- Tool for user rights and access management
- Auxiliary modules
 - Common Data Module (CDM)
 - Storing and processing of the common directories for the other modules
 - Multi-language functional
 - Logging
 - Service Bus (SB)
 - Enabling data exchange between modules
 - SMS Module
 - *Currently not in use and is not included in the solution architecture document*

7.1 Portal

Portal (HMIS.StopC.Web) is the central component of the system. The main function of this module is a possibility of dynamic creation and management of data structures for web-forms, population of the forms with data and as a result – electronic administration of STOP-C program processes. Brief Description of the Administration Process



The screenshot shows a web application interface for the STOP-C program. The left sidebar contains navigation links: ადმინისტრირება, ანალიტიკა (მართვა), კომპონენტები, სკრინინგი, ანალიტიკა, რეპორტი, and დახმარება. The main content area displays a table of personal information with columns: Transportation Status, Status Change Date, Days left, User, Provider Settlement, სამიზნე ჯგუფი, 15 ნიშნა კოდი, and პირადი ნომერი. The table shows six entries with various user IDs and provider settlements.

Transportation Status	Status Change Date	Days left	User	Provider Settlement	სამიზნე ჯგუფი	15 ნიშნა კოდი	პირადი ნომერი
None			402022253 - შპს პირველი საავადმყოფო - ქ...	ისანი (ქალაქი)	პოსპიტალიზებული პაციენტები...		59001041499
None			205284093 - შპს ონკოლოგიის სამეცნიერო კვ...	ვაკე (ქალაქი)	ზოგადი მოპულაცია		10001025320
None			406055879 - შპს "ამტელ პოსპიტალ პირველი ...	ისანი (ქალაქი)	პოსპიტალიზებული პაციენტები...		01030021067
None			39001000620 - ფ/პ თეა სახოკია - სენაკი...	სენაკი (ქალაქი)	სამეგრელოს პილოტი		39001033367
None			202051876 - შპს "აკად. გ. ჩაფიძის სახელო...	დიდუბე (ქალაქი)	პოსპიტალიზებული პაციენტები...		01017014188
None			404476205 - სს "სამედიცინო კორპორაცია ...	ქუთაისი (ქალაქი)	პოსპიტალიზებული პაციენტები...		01023012173

Figure 3. STOP-C Web-form sample page (form of personal information)

Administration STOP-C portal is possible after authorization into the system with the role of appropriate administrative user (admin).

After authorization it is possible to edit content, form and design for any component of the STOP-C portal. The detailed information about how to manage the system is available in the “Administrator Manual of STOP-C System”⁸. However, it has to be mentioned, that the main enterprise functions (forms design, content, data analysis and visualization) depends on the specific data architecture. **Figure 3** shows sample view of this functional from administrator panel (sample view of personal information form) and **Figure 4** represents data and process architecture behind this tool.

The system enables building capability of desired forms and processes for administrator. In addition to the forms management, the system supports simplified syntax and semantics for the form fields relations and validations (Appendix A: Validation Syntax).

The basic elements of the system administration are web-forms, creation, modification and deletion of which is possible from the administrator interface.

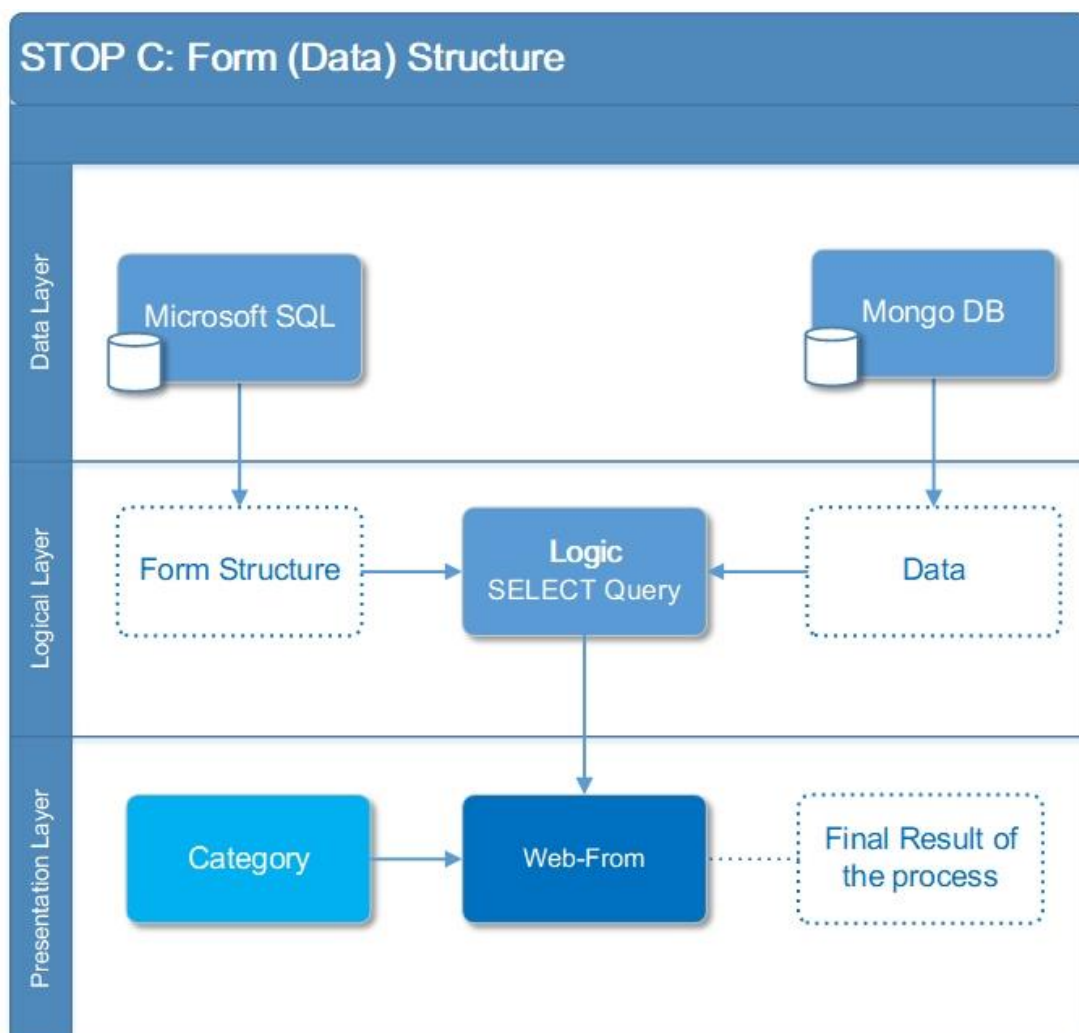


Figure 4. Logical Diagram of STOP-C form, data and process

⁸ [Link to STOP-C system administrator manual](#)

General process scheme of the forms management is the following:

1. At the initial step it is necessary to create form category, which have to be defined for all the created form in the future;
2. Creation of the directory of forms
 - a. Fill the form with data
3. Creation of a form
 - a. Define category for the corm (choosing form the existing category list);
 - b. Adding form elements (controls)
4. Define logic for filling form with the data

Figure 5 shows schematic diagram for this process and **Figure 4** represents the visualization of the components for the same process components in various views (data, logic and presentation layers).

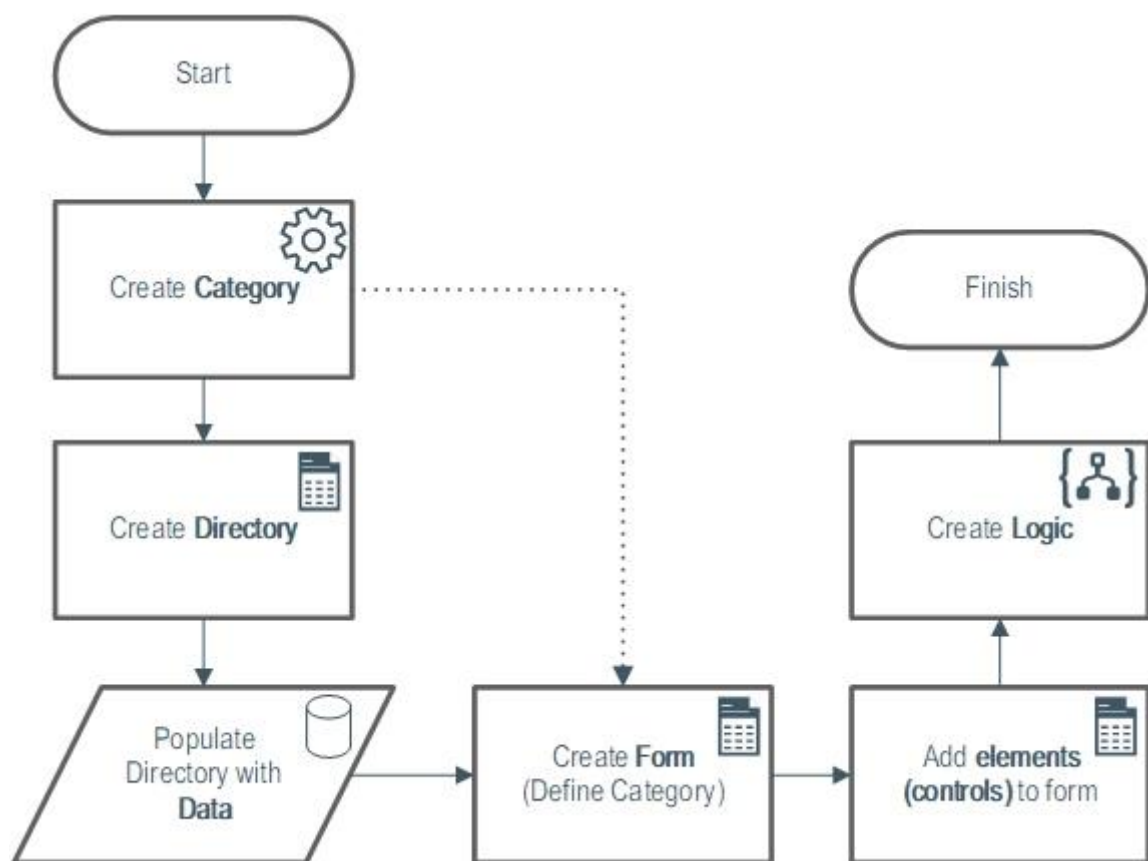


Figure 5. High-level business process diagram of STOP-C Screening forms administration

The detailed guideline for creation of reports and corresponding tables and charts are given in the documents “STOP-C: administrator manual” (chapter 11 – Reports) and “STOP-C: Analytics user manual”.

7.2 User Management Module

7.2.1 General Information about UM Module

User Management module is part of STOP-C system, purpose of which is to manage users and access rights for the central module (portal). More specifically, it defines users, right and access levels, which also supports that by validation mechanism for prevention of unauthorized access to the central module of the

system. UM Module contains unified directory of the users, defined roles, access rights for the roles and limitations. Simplified (one-step) authorization is realized in this module and consists of the interfaces that are enabled after single authorization, without need in re-authorization after changing the interfaces between various modules. The access rights are unified across the modules.

7.2.2 Functions of UM Module

The enterprise functions of the module are legalized by the following business processes:

- Management of and control of the unified user directory. This directory represents the list of the persons and/or organizations who have access rights for the central module (portal).
- When needed, based on the current or future necessities of the central module, possibility of creation of roles and user groups. The task of role creation encompasses strict definition of the resources for the role. The resource is later can be managed (created, edited or deleted) by the defined user based on the specific needs.
- Access right definition for the user groups according to the resources. Creation of the additional attributes and specific values for the attributes.
- Registration of various kinds of informative messages for the STOP-C various modules.

7.3 Analytical Module

Analytical module maintains functionality of unified dynamic reporting and analytics in various views for STOP-C screening and other related modules. Dynamic reports mean the function, which enables user (administrator) creation of the new reports and charts and modification of the existing ones, based on the current data. This process does not need involvement of software developer and the detailed guidelines are described in the document “STOP-C: Analytics user manual”⁹, and the solution architecture and specification is described in the separate document “STOP-C: Data Analytics System Architecture”¹⁰.

7.4 Service Bus

Data transport between STOP-C system modules is realized using internal Service Bus. Part the system is communication module (HMIS.messaging.web), using of which all the module of STOP-C system sends messages and requests to the other modules, not with direct connection. Every sent message should contain the following data:

- Data of that method which have to be executed in the destination module
 - Format: module_name.class_name.method_name
- Method attributes

⁹ ანალიტიკის ადმინისტრატორის სახელმძღვანელოს ბმული

¹⁰ STOP C მონაცემების ანალიტიკის სისტემის არქიტექტურა

- Serialized method attributes (using *BinaryFormatter*)

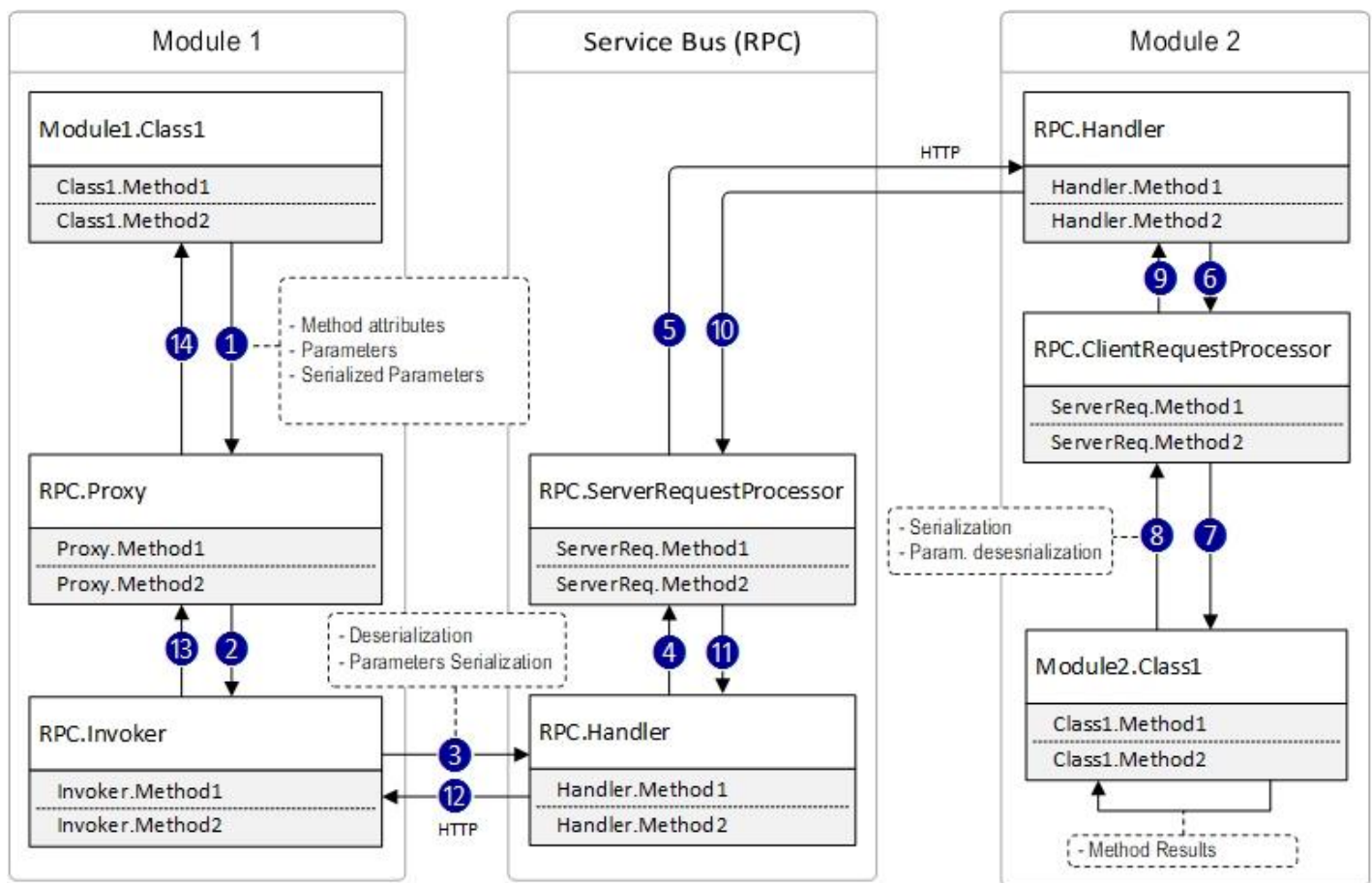


Figure 6. . Process-Concept diagram of Service Bus communication loop

Service Bus reads module name provided with the request and send the request to the corresponding module. When the target module receives request from Service Bus, it tries to find the class and that method in the class, which was specified in the message and farther tries to execute it. After that the results of the execution is returned to the requesting module. For the data transfer details please refer to the **Figure 6**.

For this purpose, to exchange data between modules so called proxy-classes are used. These classes are responsible form exchanging requests and results between modules. The messages, which are exchanged contain binary data, serialized using *BinaryFormatter*. No other standard (eg.: SOAP) is used in this process.

7.5 Common Data Module

Common Data Module (CDM, HMIS.commondata.web) is a combination of universally used electronic services, which contains types of services and data, that are common throughout all the STOP-C modules. Such data can be various directories: phone index, location names, texts used for Multilanguage support and others.

Common Data Module has interface, using of which is possible to translate the pages into various languages and add and modify them in the database. In case of need, CDM enables logging of accesses to

all the STOP-C modules (Which use accessed which page and how much time took server to return the reply. In CDM specific interface exists where it is possible to overview the information above.

The interfaces are located on the following addresses:

- Translation Interface:
<http://birthregistry.moh.gov.ge/HMIS/HMIS.CommonData.web/Pages/TranslationsList.aspx>
- Log interface:
<http://birthregistry.moh.gov.ge/HMIS/HMIS.CommonData.web/Pages/ActionLogViewer.aspx>

8 Platform and used Technologies

For all the System modules architecture types and technologies are the same and are based on the following platforms and technologies:

- Microsoft .NET Framework 4.5.1 ¹¹

For STOP-C, Microsoft .NET represents software development environment, which contains large libraries of the classes (Framework Class Library - FCL) and enables interoperability. The code, written for .NET Framework, is executed in software environment (different from hardware environment) in Common Language Runtime (CLR), which is virtual machine for the application and supports such services as security, memory management and exceptions processing. FCL together with CLR represents Microsoft .NET Framework. STOP-C is built on one of the latest stable version - 4.5.1

- Microsoft ASP.NET Web Forms ¹²
 - Microsoft ASP.NET enables supports dynamic architecture of STOP-C, which makes it possible to create and edit desired web-pages (forms) easily, without involvement of developers.
- CITI.EVO.TwoWayModelBinding – MVC¹³ analogue for ASP.NET Web Forms
 - Web-application environment that is based on Model–view–controller methodology.
- NHibernate ¹⁴
 - NHibernate is an object-relation connector (Object-Relational Mapper - ORM), which means to “translate” STOP-C data from .NET environment to almost any relational database format (Oracle DB, IBM DB2, mysql, postgre SQL და ს.შ.). As a result, STOP-C system is not depending on the specific type of database and any relational database can be used in the solution.

¹¹ <https://www.microsoft.com/en-us/download/details.aspx?id=40779>

¹² <https://www.asp.net/web-forms>

¹³ [https://msdn.microsoft.com/en-us/library/dd381412\(v=vs.108\).aspx](https://msdn.microsoft.com/en-us/library/dd381412(v=vs.108).aspx)

¹⁴ <http://nhibernate.info>

- Microsoft SQL ¹⁵
 - Microsoft SQL acts as a storage for STOP-C metadata, that means the information about data structure, but not the data itself.
- MongoDB ¹⁶
 - MongoDB act as a data storage for STOP-C modules. It is not a relational (SQL) database, but a document-oriented one, which enables possibility to create dynamic web-pages in STOP-C system. Therefore, form-corresponding data is also created dynamically, without help from a software developer, by system administrator. MongoDB is based on one of the mostly widespread principle - key-value¹⁷ Pair Paradigm.

¹⁵ <https://www.microsoft.com/en-us/sql-server/sql-server-2017>

¹⁶ <https://www.mongodb.com>

¹⁷ [https://www.wikiwand.com/en/Key-value database](https://www.wikiwand.com/en/Key-value_database)

9 System Architecture

Figure 1 shows logical diagram of STOP-C Screening electronic Management System components, in from high-level view of applications and databases.

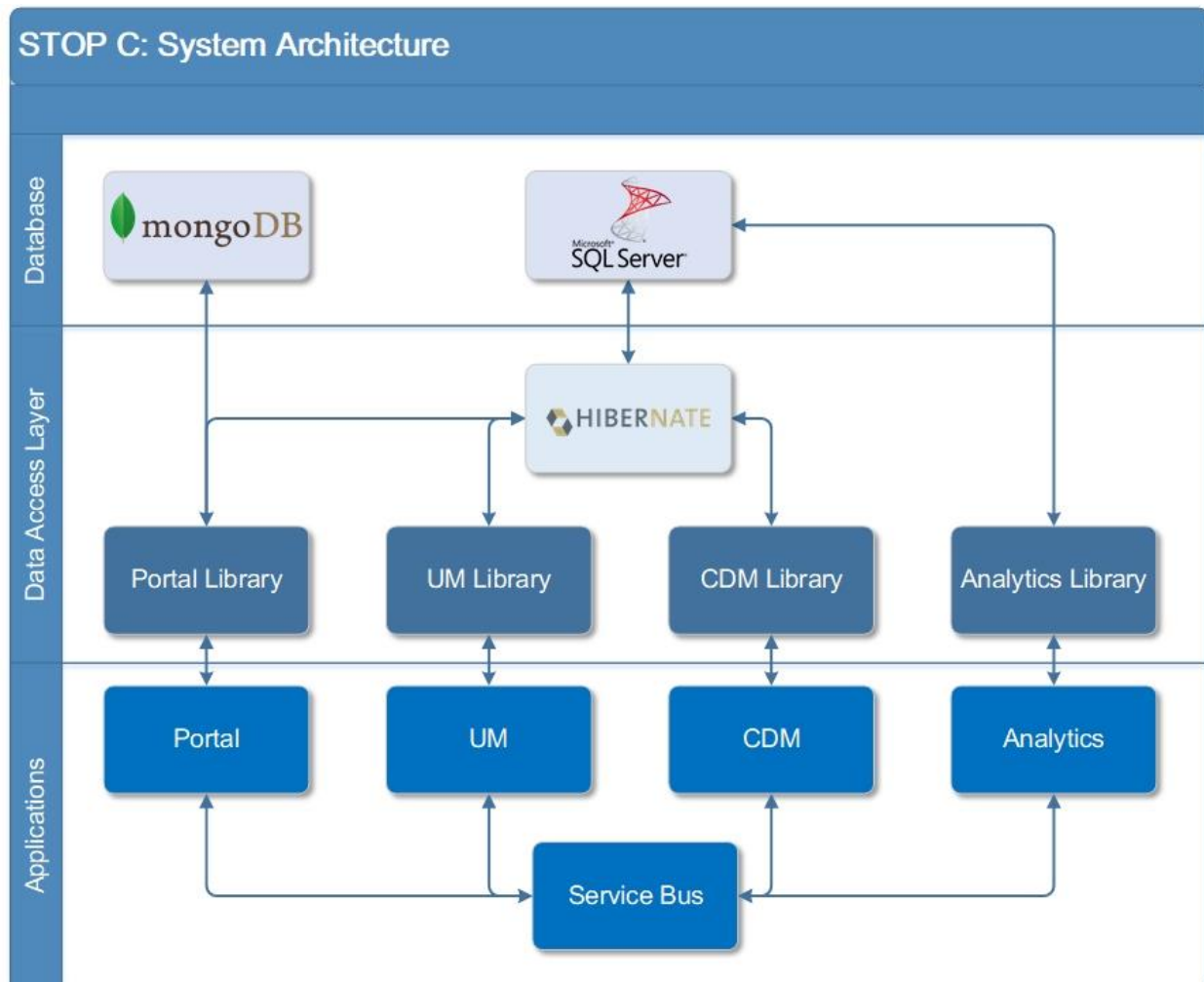


Figure 7. STOP-C Components System Architecture

9.1 Main Module (Portal)

9.1.1 Portal Classes

Portal Classes, according to the roles and functions are grouped by several categories:

- **Helper Classes (Figure 8, Table 1):** Various assistant classes
- **Converter Classes (Figure 9, Table 2):**
 - Entity to model converter classes – these classes convert database table classes (DB Entities) into corresponding Model classes
 - Model to entity converter classes: these classes convert Model classes into corresponding database table classes (DB Entities).
- **Entity classes (Figure 10, Table 3):** Contains classes that encompass various data, structure and other information.

- Model classes (**Figure 11, Table 4**): the classes for the specific user interface fields. The structure of all these classes resemble the elements of the corresponding controls.



Figure 8. Class diagram (UML) of Portal Helper-classes

Table 1. Helper Classes

Full Name of Class	Library	Class Description
Gms.Portal.Web.Caches.DataStatusCache	App_Code	Cash of the data statuses
Gms.Portal.Web.Caches.UmUsersCache	App_Code	User Cash
Gms.Portal.Web.Helpers.BsonDocument Converter	App_Code	Helper class which converts JsonDocument into FormDataUnit or Dictionary and vise versa

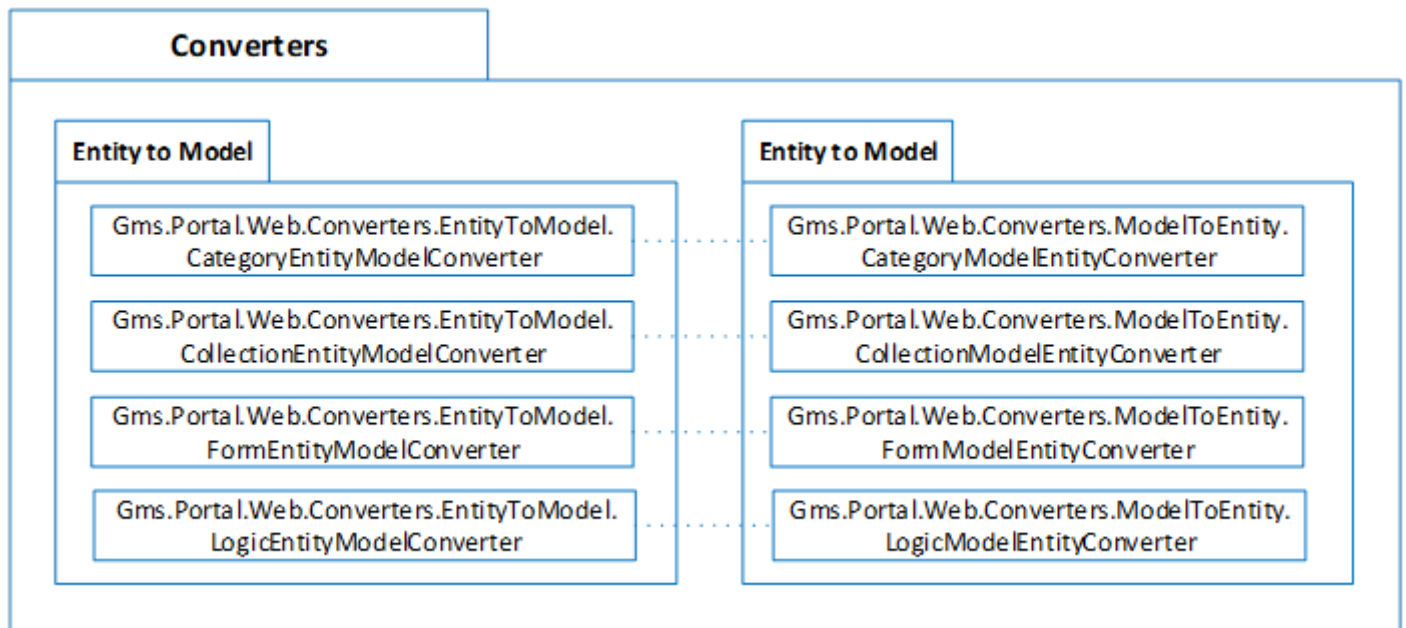


Figure 9. Class diagram (UML) of Portal Converter-classes

Table 2. Converter classes: Entity to Model and Model to Entity

Full Name of Class	Library	Class Description
Gms.Portal.Web.Converters.EntityToModel		Namespace contains converter classes, that convert Database structural entity class (DB entity) into Interface Model Class
Gms.Portal.Web.Converters.EntityToModel.CategoryEntityModelConverter	App_Code	Categories converter
Gms.Portal.Web.Converters.EntityToModel.CollectionEntityModelConverter	App_Code	Collections (Directory structure) Converter
Gms.Portal.Web.Converters.EntityToModel.FormEntityModelConverter	App_Code	Form (form structure) converter
Gms.Portal.Web.Converters.EntityToModel.LogicEntityModelConverter	App_Code	Logic converter
Gms.Portal.Web.Converters.ModelToEntity		Namespace contains converter classes that convert interface model class into database structural entities (DB entity)
Gms.Portal.Web.Converters.ModelToEntity.CategoryModelEntityConverter	App_Code	Category converter
Gms.Portal.Web.Converters.ModelToEntity.CollectionModelEntityConverter	App_Code	Collection (directory structure) converter
Gms.Portal.Web.Converters.ModelToEntity.FormModelEntityConverter	App_Code	Form (form structure) converter
Gms.Portal.Web.Converters.ModelToEntity.LogicModelEntityConverter	App_Code	Logic converter

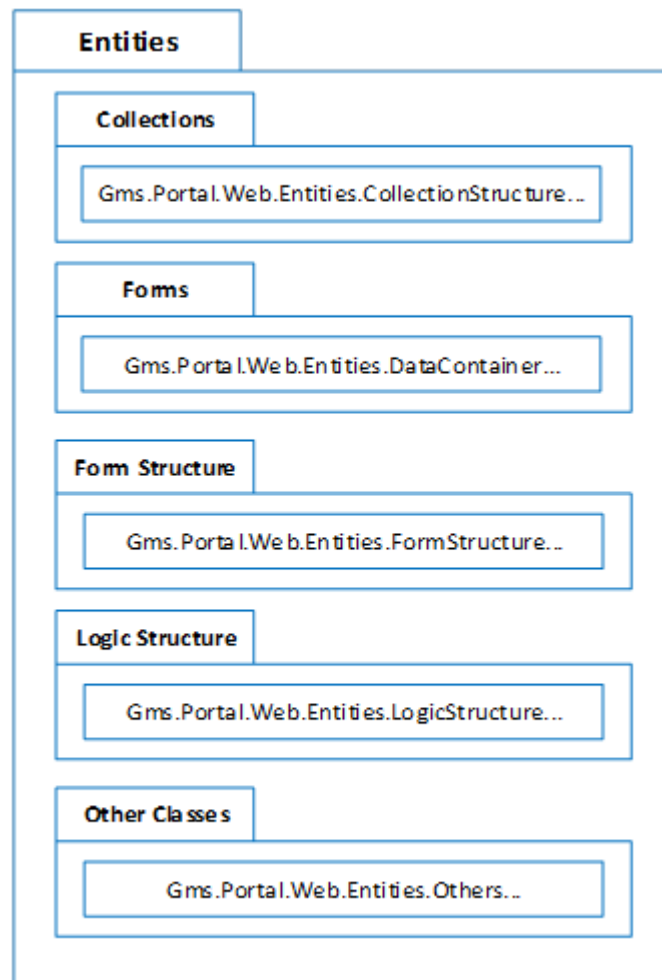


Figure 10. Class diagram (UML) of Portal Entity-classes

Table 3. Entity classes

Full Name of Class	Library	Class Description
Gms.Portal.Web.Entities.CollectionStructure	App_Code	Encompasses classes that contain collection (directory)structures (that are later stored as XML in GM_Collections.XmlData column)
Gms.Portal.Web.Entities.CollectionStructure.C ollectionEntity	App_Code	Collections (Directory structure) data
Gms.Portal.Web.Entities.CollectionStructure.F ieldEntity	App_Code	Collection (directory) fields data
Gms.Portal.Web.Entities.DataContainer.Form DataBase	App_Code	Basic class containing form data
Gms.Portal.Web.Entities.DataContainer.Form DataUnit	App_Code	Class containing form data
Gms.Portal.Web.Entities.DataContainer.Form DataBaseList	App_Code	Class containing form data list
Gms.Portal.Web.Entities.DataContainer.Form DataLazyList	App_Code	Class contains form data list (with function of loading when needed)

Full Name of Class	Library	Class Description
Gms.Portal.Web.Entities.DataContainer.FormDataListRef	App_Code	Reference type class, which contains link for form data list
Gms.Portal.Web.Entities.FormStructure	App_Code	Contains classes for form data structures (that are later stored as XML in GM_Forms.XmlData column)
Gms.Portal.Web.Entities.FormStructure.ContentEntity	App_Code	Control basic class (may contain child-controls)
Gms.Portal.Web.Entities.FormStructure.ControlEntity	App_Code	Control basic class
Gms.Portal.Web.Entities.FormStructure.FieldEntity	App_Code	Class containing details about the field
Gms.Portal.Web.Entities.FormStructure.FormEntity	App_Code	Class containing form information
Gms.Portal.Web.Entities.FormStructure.GridEntity	App_Code	Table Class
Gms.Portal.Web.Entities.FormStructure.GroupEntity	App_Code	Controls group class
Gms.Portal.Web.Entities.FormStructure.TabContainerEntity	App_Code	Type container class
Gms.Portal.Web.Entities.FormStructure.TabPageEntity	App_Code	Tab-page class
Gms.Portal.Web.Entities.LogicStructure	App_Code	Contains classes from logic structure (that are later stored as XML in GM_Logics.RawData column)
Gms.Portal.Web.Entities.LogicStructure.ExpressionEntity	App_Code	Logic expression class
Gms.Portal.Web.Entities.LogicStructure.LogicContainerEntity	App_Code	Expression container basic class
Gms.Portal.Web.Entities.LogicStructure.LogicEntity	App_Code	Basic container class, that contains string command or logic
Gms.Portal.Web.Entities.LogicStructure.LogicExpressionsEntity	App_Code	Expression container class
Gms.Portal.Web.Entities.LogicStructure.LogicQueryEntity	App_Code	String command container class
Gms.Portal.Web.Entities.LogicStructure.NamedExpressionEntity	App_Code	Named expression container class
Gms.Portal.Web.Entities.Others.ControlTreeEntity	App_Code	Helper class form control data container
Gms.Portal.Web.Entities.Others.ElementTreeNodeEntity	App_Code	Helper class form element data container
Gms.Portal.Web.Entities.Others.TreeNodeEntity	App_Code	Helper class tree node data container

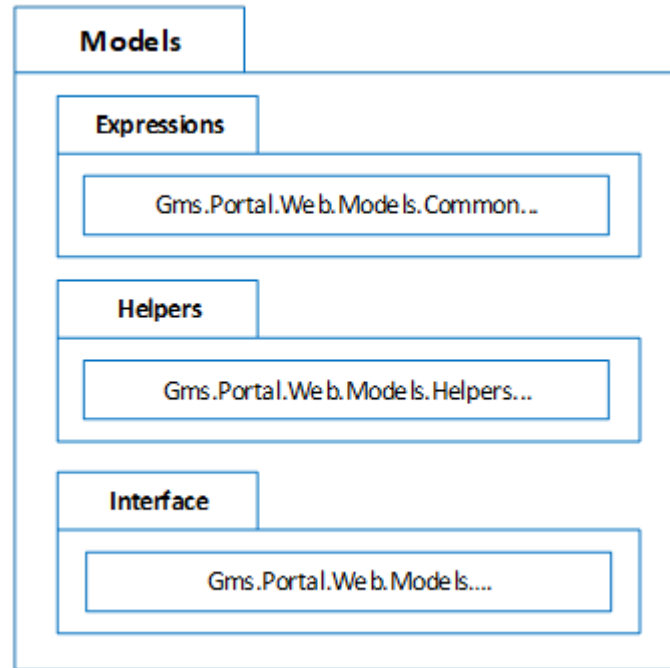


Figure 11. Class diagram (UML) of Portal Model-classes

Table 4. Model classes

Full Name of Class	Library	Class Description
Gms.Portal.Web.Models		This Namespace contains user interface models (as a result, the structure of all the classes match to the element of the corresponding controls)
Gms.Portal.Web.Models.Common.ExpressionModel	App_Code	Logic expression class
Gms.Portal.Web.Models.Common.ExpressionListModel	App_Code	Logic expression list class
Gms.Portal.Web.Models.Common.ExpressionLogicModel	App_Code	Logic expression grouper class (Filter, Group By, Order By, Select expressions together)
Gms.Portal.Web.Models.Common.NamedExpressionModel	App_Code	Name expression class (e.g. (A + B) AS Amount)
Gms.Portal.Web.Models.Common.NamedExpressionsListModel	App_Code	Named expression list class
Gms.Portal.Web.Models.Common.NamedModel	App_Code	Class for editing system simple elements (e.g. categories). May contain only 2 fields at most (name and parent ID)
Gms.Portal.Web.Models.Helpers.CategoriesFormsModel	App_Code	Model, that displays in category and form menus together (contains the following classes list)
Gms.Portal.Web.Models.Helpers.CategoryFormModel	App_Code	Model, that displays in category and form menus together
Gms.Portal.Web.Models.Helpers.ElementNodesModel	App_Code	Model that sorts and displays elements in from as a tree (Contains ElementTreeNodeEntity classes list)
Gms.Portal.Web.Models.Helpers.FieldUnitsModel	App_Code	Contains collection (directory) fields list (for displaying in grid)
Gms.Portal.Web.Models.CategoriesModel	App_Code	Contains category list
Gms.Portal.Web.Models.CategoryModel	App_Code	Contains information about category
Gms.Portal.Web.Models.CollectionDataModel	App_Code	Contains data about one row/record
Gms.Portal.Web.Models.CollectionDataModelsModel	App_Code	Contains one directory data (CollectionDataModels list)
Gms.Portal.Web.Models.CollectionModel	App_Code	Contains directory structure data
Gms.Portal.Web.Models.CollectionsModel	App_Code	Contains directories list
Gms.Portal.Web.Models.ElementModel	App_Code	Contains form elements data
Gms.Portal.Web.Models.FormDataGridModel	App_Code	Contains form data table
Gms.Portal.Web.Models.FormDataModel	App_Code	Contains data from one form record
Gms.Portal.Web.Models.FormModel	App_Code	Contains form structure
Gms.Portal.Web.Models.FormsModel	App_Code	Contains forms list
Gms.Portal.Web.Models.LogicModel	App_Code	Contains logics data
Gms.Portal.Web.Models.LogicsModel	App_Code	Contains logics list

Full Name of Class	Library	Class Description
Gms.Portal.Web.Models.RecordStatusModel	App_Code	Contains form record status data
Gms.Portal.Web.Models.TableDataModel	App_Code	Not used on this stage, but contains methods for the future use

9.1.2 Physical Infrastructure Architecture

Physical architecture of the portal embodies application (Web) and Database servers and their addresses are presented below:

- Web Server: <http://STOP-C.moh.gov.ge> 172.17.216.53
- MongoDB Server: 172.17.7.132
- Microsoft SQL Server: 172.17.7.132

9.2 User Management Module

User Management module is realized using web technologies. The dialed information is available in Chapter 5 - General Information about UM Module.

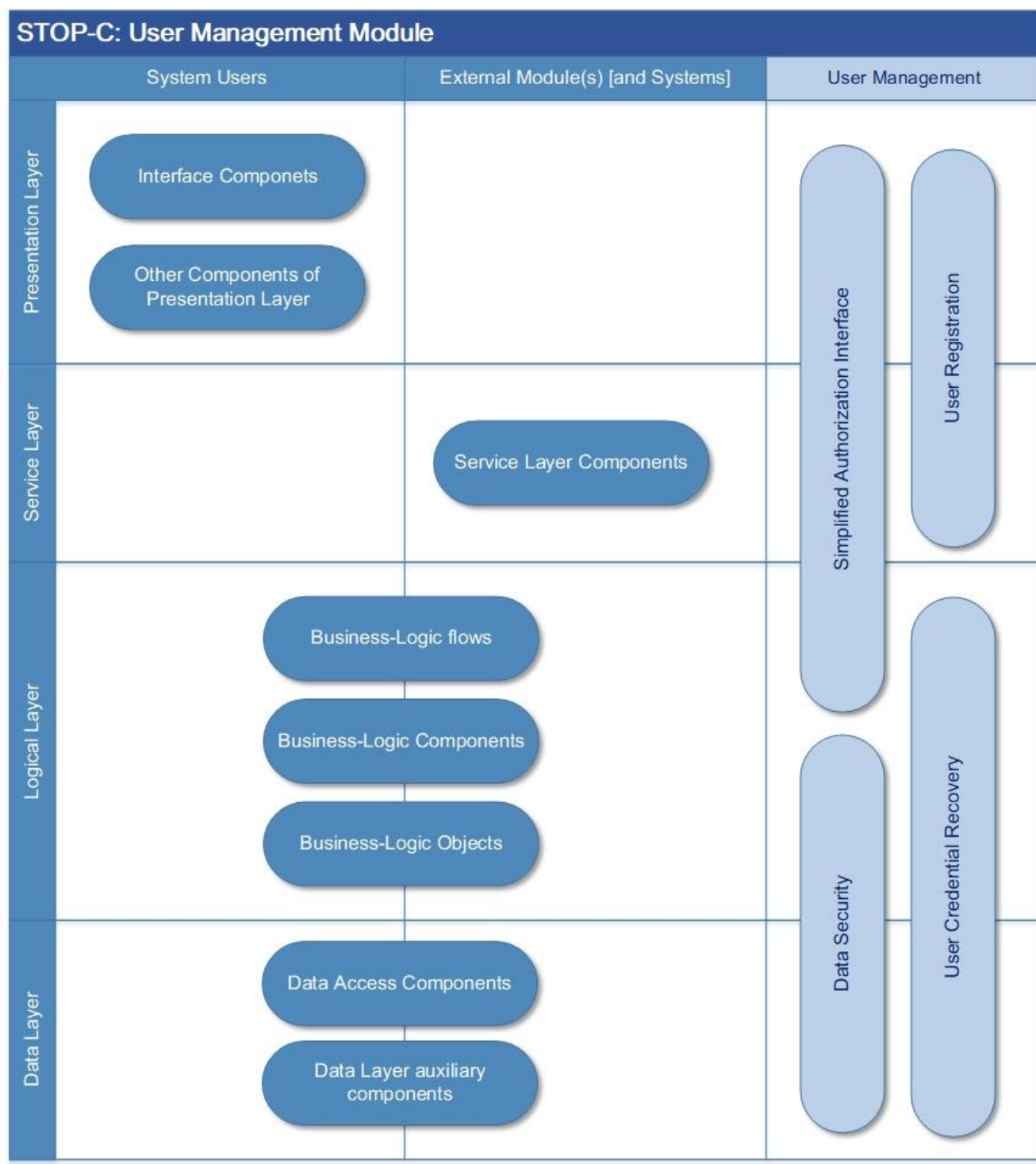


Figure 12. Various layers of User Management Module

9.2.1 General Architecture

User Management Module general architecture encompasses the following layers:

- Presentation Layer
- Business Logic Layer
- Service Layer
- Data Layer

Each of the layer has specific role and function under User management module unified architecture. Same time, these layers are tightly connected to each other are actively exchange necessary information streams (see **Figure 12**).

9.2.1.1 Presentation Layer

Presentation layer is represented by the collection of those interfaces and working web-forms, using of which enables administrators to work in the system. This layer is such an interface, via which realized scenarios on the business level are executed and being operated on.

9.2.1.2 Business-Logic Layer

Business-Logic Level is an important component in the general architecture of UM Module. All those scenarios, validations and control mechanisms are collected here, that defines module characteristics and purpose. This layer is connected with the other layers under general architecture. It is business-logic layer, through which presentation and service layers are connected to the database layer.

The list of validations is realized in User Management Module, that are necessary for the system to fulfil the designed aims, to gather the correct information about users and distribute it to the other parts of the system. Due to the characteristics of the system, validations enable functions of UM module, depending Central module (Portal) and all the other modules in future. Below there is a list of the validations (with definitions) which are critically important for valid operation of UM Module:

- **Validation of necessary fields** - For example it is necessary for the user to have username and password
- **Prerequisite for a group creation** – user must necessarily be a member of a desired module (at this stage only Portal is necessary) (administrator and/or regular user).
- **Duplicity validation** – Validation on the duplicity according to username.
- **Various additional validations** – Due to the specific characteristics of the interfaces, there are validations which support seamless operation of the module (for instance, which execution of the specific commands, there are necessary attributes which must be defined in the method). Also validations based on the field format, so that correct format value is sent to the needed service.

Figure 13 shows validations and logical information flows between the modules, according to the specific tasks.

9.2.1.3 Service Layer

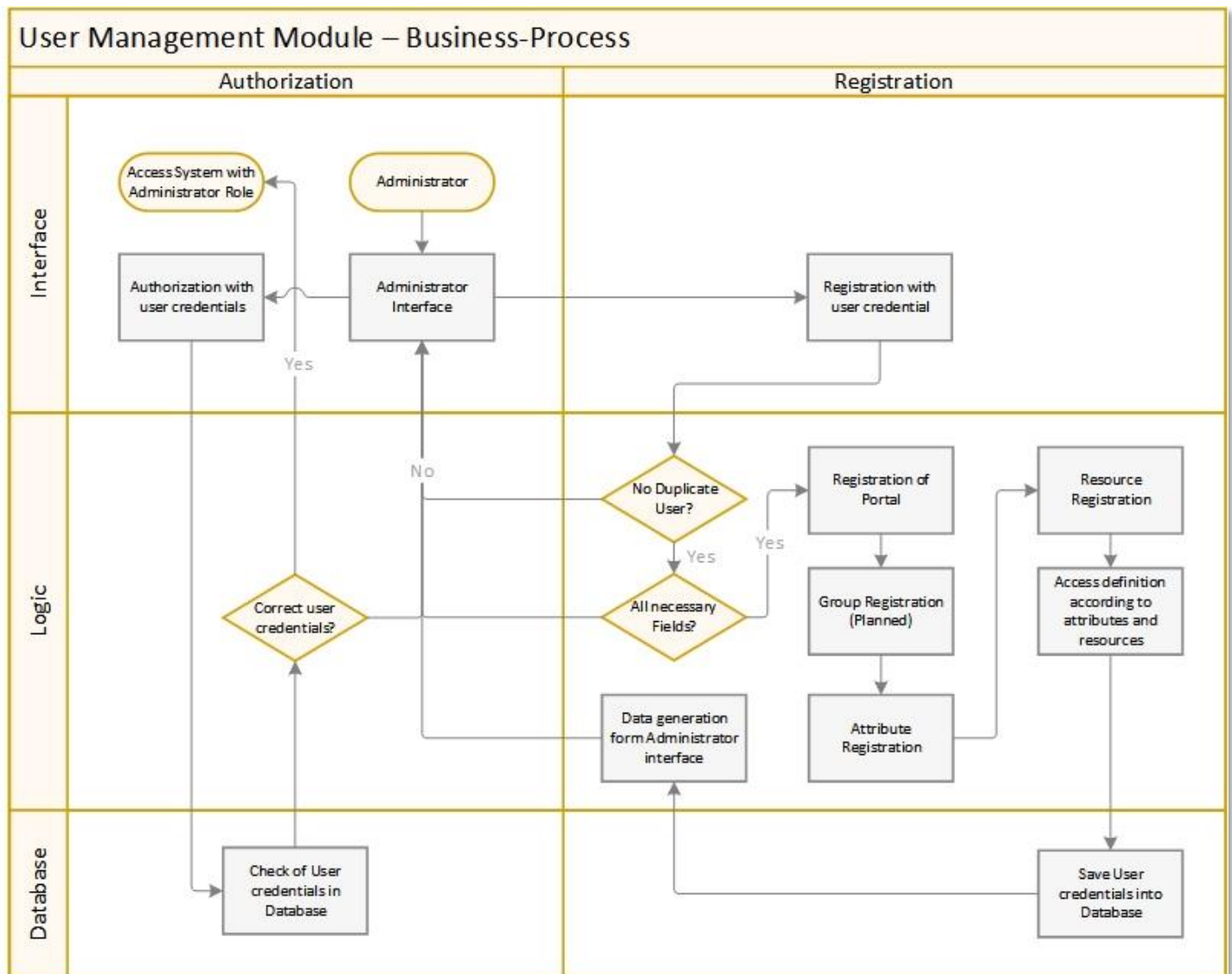


Figure 13. User Management Administration - Business Process

Service layer is one of the most important components of the UM module, however on this stage the usage of its full functions is limiter, because the system is comprised by only two modules, therefore the need in information exchange between them is low. The main purpose of this layer is to organize communication between the system modules and external environment (in case if corresponding access is there). All external and internal modules exchange information using this service layer.

9.2.1.4 Data Layer (DB)

The main purpose of Data layer is to collect and organize data from the other layers of UM module in an optimal way, which enables the module to fulfil the tasks: fast search, indexing, data generation and data output. Performance of this layer is one of the most critical and defines overall performance of the module. No business-process is realized on this layer not to duplicate scenarios with business-logic layer.

9.2.2 Physical Infrastructure

Physical architecture of the portal embodies application (Web) and Database servers and their addresses are presented below:

- **Web Server** <http://STOP-C.moh.gov.ge> 172.17.216.53
- **Database Server:** 172.17.7.130

9.2.3 Informational Streams

With help of various methods and channels, User Management Module receives and sends various information. Therefore, these information streams are divided into two groups “input/received information” and “output information”. In addition, the methods of receiving or sending, also methods of saving the data into the database differ. The figure below (**Figure 14**) shows graphical diagram of information streams, where green color defines input/received information streams and red color – output information streams.

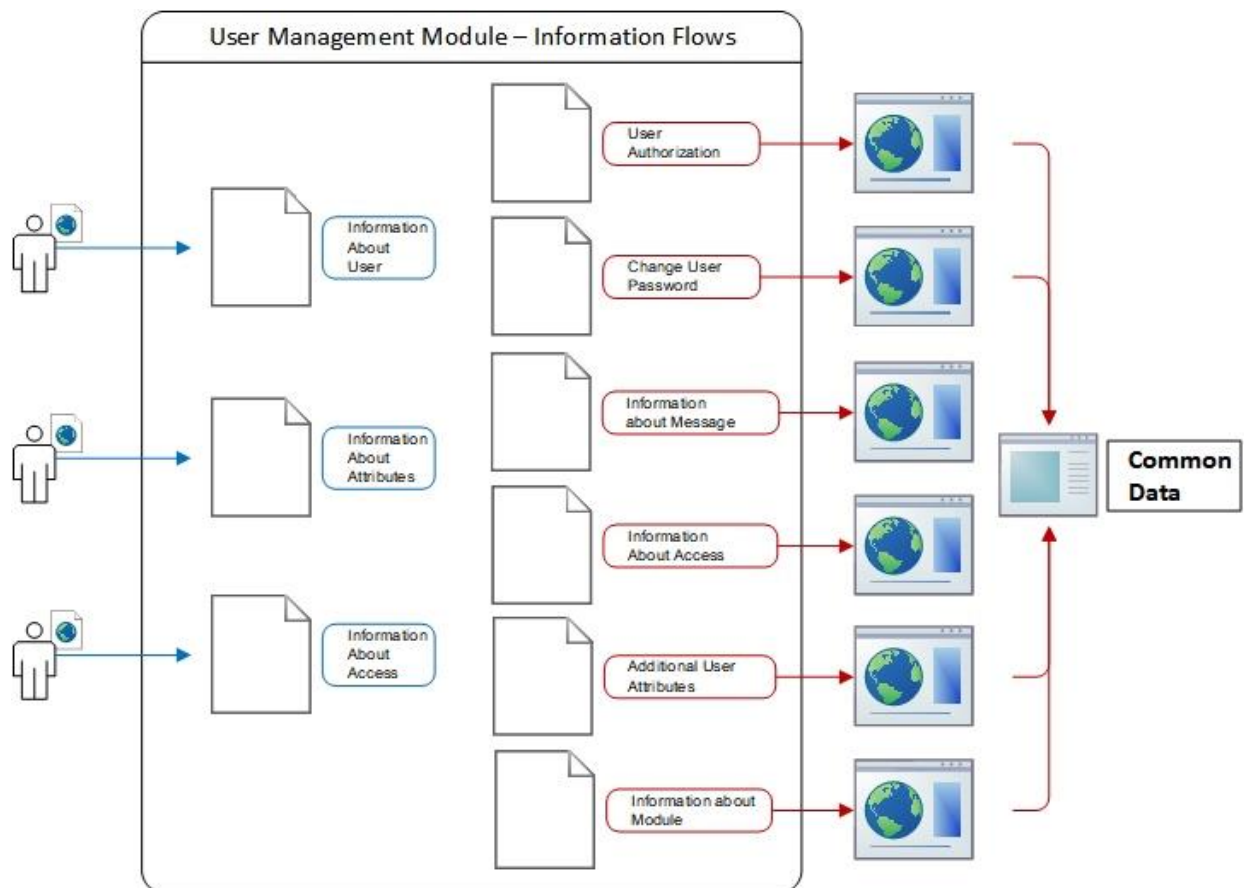


Figure 14. User Management Module - Information Streams

Table 5 and **Table 6** show the details regarding “input/received information” and “output information”.

Table 5. Received information streams

Information Stream	Source	Input Form	Period	Stored in Database	Address	Method
Information about user	Module user	Interface	Per request	Yes	18	N/A
Information about changed password	User Management Module (UMM)	Web-service	Per request and/or after password expiration period			19
Information about attribute schema, attribute schema node and attribute values	User Management Module (UMM)	Interface	Per request			N/A
Information about resources and the access rights for the resources	User Management Module (UMM)	Interface	Per request			N/A

Table 6. Sent information streams

Information Stream	Destination	Input Form	Period	Address	Method
Information about all the users, according to the authorized users	Portal	Web-service	Per request	20	21
Information about current user					22
Information about active/passive status of the authorized user token					23
Authorization					24
Information about the members of user groups					25
Information about user groups					26

¹⁸ <http://birthregistry.moh.gov.ge/HMIS/HMIS.StopC.web>

¹⁹ PasswordChangeResultEnum ChangePassword(Guid token, String newPassword, String oldPassword)

²⁰ <http://birthregistry.moh.gov.ge/HMIS/HMIS.UserManagement.web/Services/UserManagementWcf.svc>

²¹ List<UserContract> GetAllUsers(Guid token)

²² public UserContract GetCurrentUser(Guid token)

²³ bool IsTokenActual(Guid token)

²⁴ Guid? Login(String loginName, String password, bool encryptedPassword)

²⁵ List<UserContract> GetGroupUsers(Guid token, Guid groupID)

²⁶ List<GroupContract> GetUserGroups(Guid token, Guid userID, Guid projectID)

Information Stream	Destination	Input Form	Period	Address	Method
Information about group attributes (additional information)					27
Information about user attributes (additional information)					28

9.2.4 Security

Non authorized access and information security to the User Management Module is controlled dynamically by current module. It also supports single authorization functionality using corresponding credentials (username and password), which means getting possibility to access the central and other modules (for future development), using access rights and periods, separately for each of the modules. The competences of User Management Module also contain the possibility to define and control on the usage of its resources by the users and corresponding access rights.

9.3 Common Data Module

The architecture of Common Data Module (CDM) is the same as for User Management Module and encompasses the unity of following layers:

- Presentation Layer
- Business Logic Layer
- Service Layer
- Data Layer

For more details, refer to the Chapter 9.2.1.

9.3.1 Common Data Module Classes

- **Caches:** Classes for storing temporary data
 - TranslationCache.cs
- **Configs:** Application configuration classes
 - **InsuranseService:** Service Class
 - ServiceUrlElement.cs
 - ServiceUrlElementCollection.cs
 - ServiceUrls.Section.cs
- **Logger:** Log configuration classes
 - LoggerCompitabilitiesSection.cs
 - LoggerCompitabilityElement.cs
 - LoggerCompitabilityElementCollection.cs
- **Entities:** Classes necessary fo application objects
 - InsuranceEntity.cs
 - PersonInfoSsaEntity.cs

²⁷ List<GroupAttributeContract> GetGroupAttributes(Guid token, Guid groupID)

²⁸ List<UserAttributeContract> GetUserAttributes(Guid token, Guid userID, Guid projectID)

- TranslationEntity.cs
- **Extensions:** Class for converting database data into application internal classes and vice-versa, plus binding additional custom methods for some classes
 - ContractsExtensions.cs
- **Handlers:** Class for processing general data for pages (e.g. file, image, etc.)
 - PersonPhotoHandler.cs
- **Helpers:** Various helper classes
 - **GovTalk:** Classes for communication with State Service Development Agency (former CRA.gov.ge)
 - EncryptKey.cs
 - GovTalkApiClient.cs
 - GovTalkCallApi.cs
 - GovTalkHelpers.cs
 - KeyExchange.cs
 - Result.Status.cs
 - XmlCrypting.cs
 - XmlSigning.cs
 - Cryptography.cs
 - DynamicEntity.cs
 - LogHelper.cs
 - SyncPersonDeathInfoLoader.cs
 - SyncPersonInfoLoader.cs
 - TrustAllCertificatePolicy.cs
- **Services:** Classes necessary for web-services
 - **Managers:** Web-service management classes
 - CommonDataChangeTimeManager.cs
 - CommonDataManager.cs
 - Common.ServiceWrapper.cs
 - MohServicesManager.cs
 - CommonDataWcf.cs
 - CommonDataWeb.cs
 - CommonDataWcf.cs
- **Util:** Additional utility classes (Tools)
 - EmailUtil.cs
 - MohServicesUtil.cs
 - RemoteAccessUtil.cs
 - UserManagementUtil.cs

9.3.2 Common Data Module – Information streams

Common Data Module is a shared data storage of information and services under STOP-C system solution. The information streams, methods of the information exchange and the corresponding sources are gathered into the following two categories:

- STOP-C Modules interface components' names in Various languages;

- User The operations list (Logs) under STOP-C system;

Module also receives and sends various data using various methods. Therefore, these streams are divided into two groups – “received information” and “Sent information”. The methods of information stream input/output, also the methods of storing the data into database are different. The list of information streams, with their direction, input/output methods and periods are given below:

Table 7. Received information

Information Stream	Source	Input Form	Period	Stored in Database	Address
Description of the interface components of the STOP-C modules in various languages	Interface	Interface	Solely for a single word	Yes	29
SetTranslatedText: adds desired translation to the specific control	STOP-C Modules	Web-service	Solely for a single Control		

Table 8. Sent information

Information Stream	Source	Input Form	Period	Address
GetLanguages: Returns the list of the language into which STOP-C modules are translated	STOPC Modules	Web-service	Per request	30
GetTranslatedText: Returns the translation of the interface control into the desired language	STOPC Modules	Web-service	Per request	

10 Database Structure and Definitions

Data Storage, processing and output tasks in STOP-C databases are decentralized into the two various database systems. This solution makes the tasks related to the data more dynamic and so solve them, in various cases two different databased are used per task – the database is chosen according to the characteristics of the task and when using one specific database is more effective and efficient.

- Microsoft SQL³¹ : See the Chapters 10.2.1 and 10.2.2.

²⁹ <http://birthregistry.moh.gov.ge/HMIS/HMIS.CommonData/Services/CommonDataWcf.svc>

³⁰ <http://birthregistry.moh.gov.ge/HMIS/HMIS.CommonData/Services/CommonDataWcf.svc>

³¹ <https://www.microsoft.com/en-us/sql-server/sql-server-2017>

- MongoDB³²

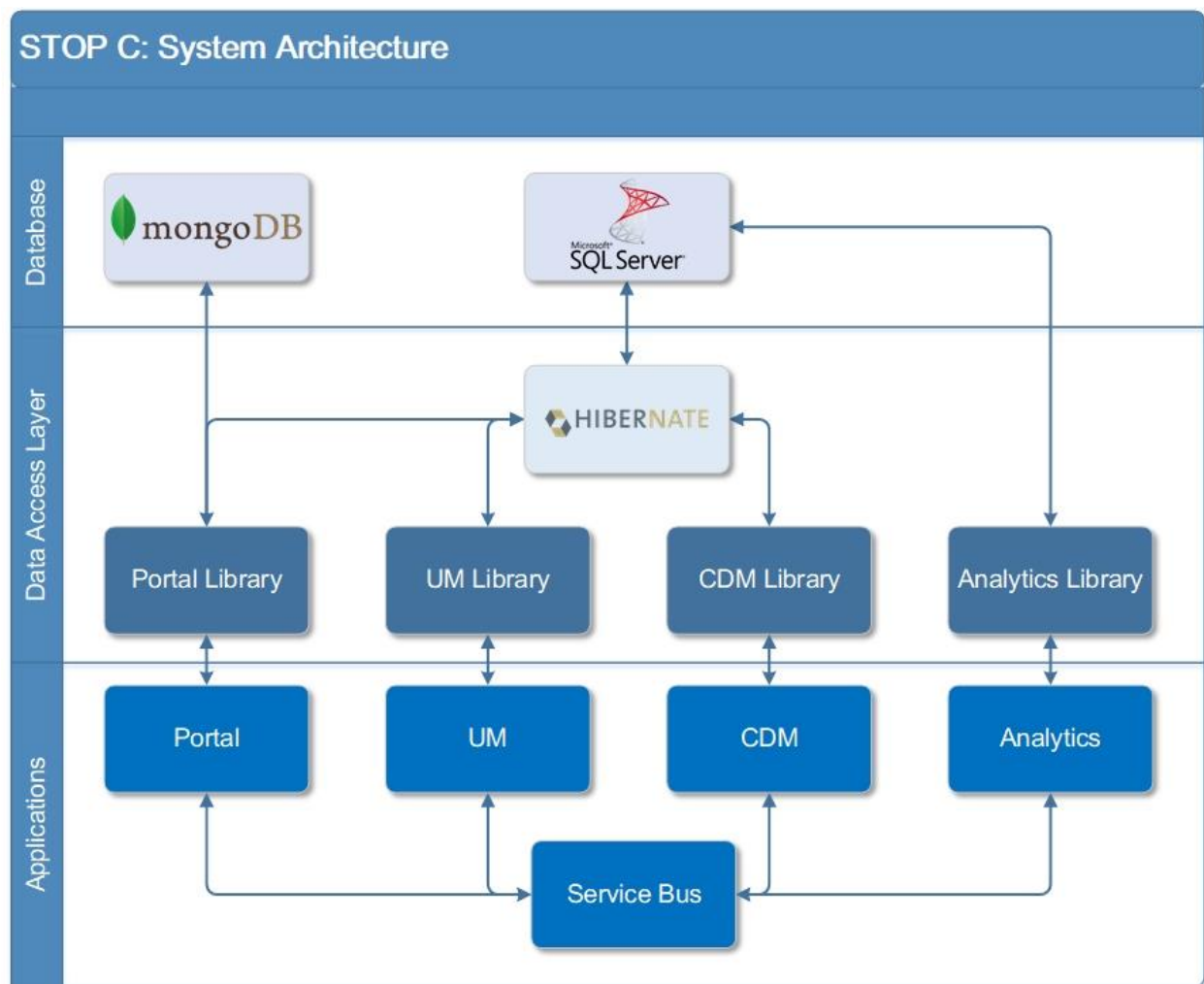


Figure 15. STOP-C components database infrastructure

10.1 User Management Module Database

User Management Module Database is realized as a centralized database for a single module, which is located on the database server (For the address and hardware details see Chapter 12.2.). Logging functionality is partly realized for those data, data historical analysis of which might be needed in the future (for this purpose the tables contains the following columns set: DateCreated, DateChanged, DateDeleted).

Logging system for modules performance monitoring is also realized in the solutions. Its interface enables accounting and monitoring of the time spent by used on the specific interface operations.

³² <https://www.mongodb.com>

10.1.1 Database Structure

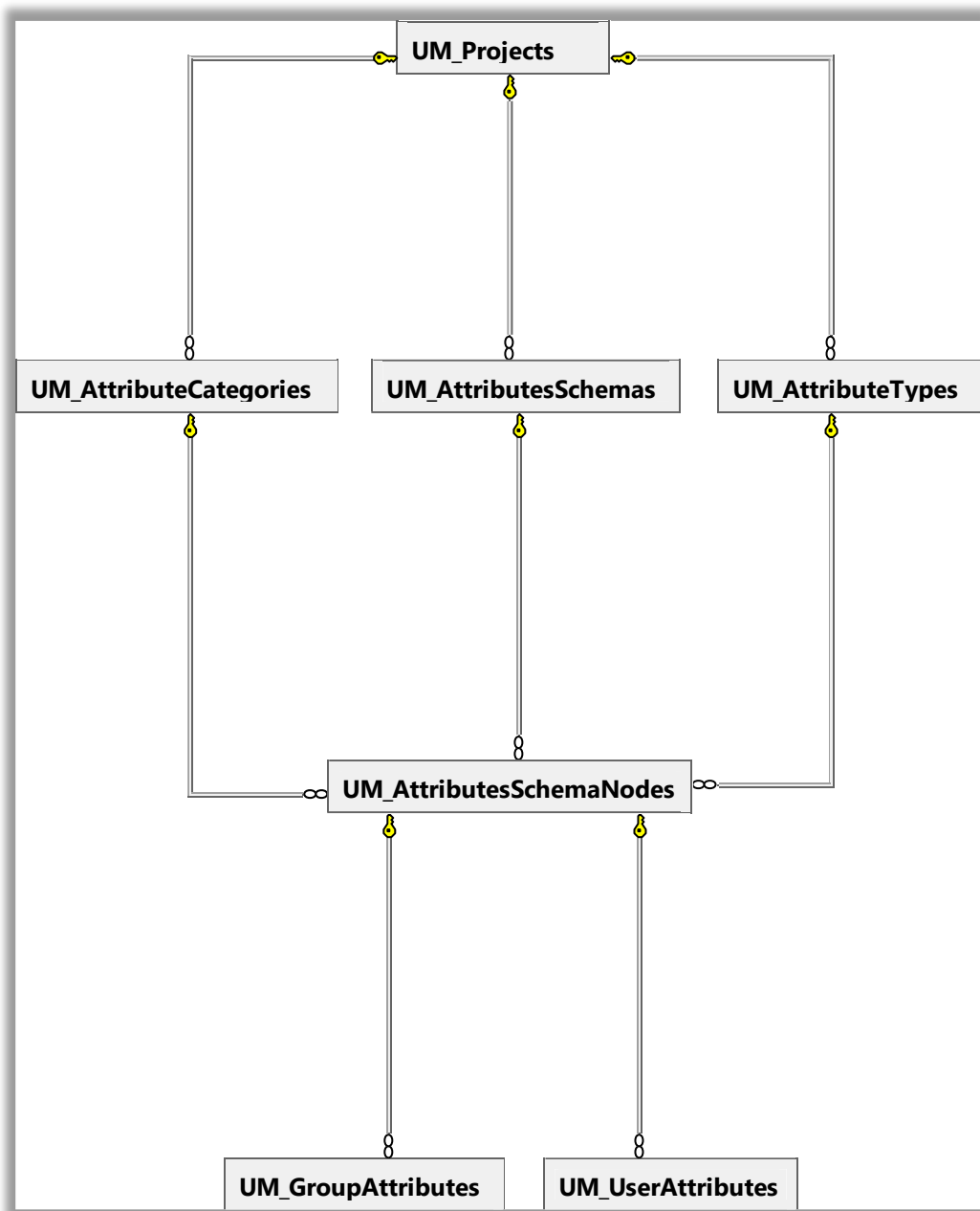


Figure 16. Attributes schema

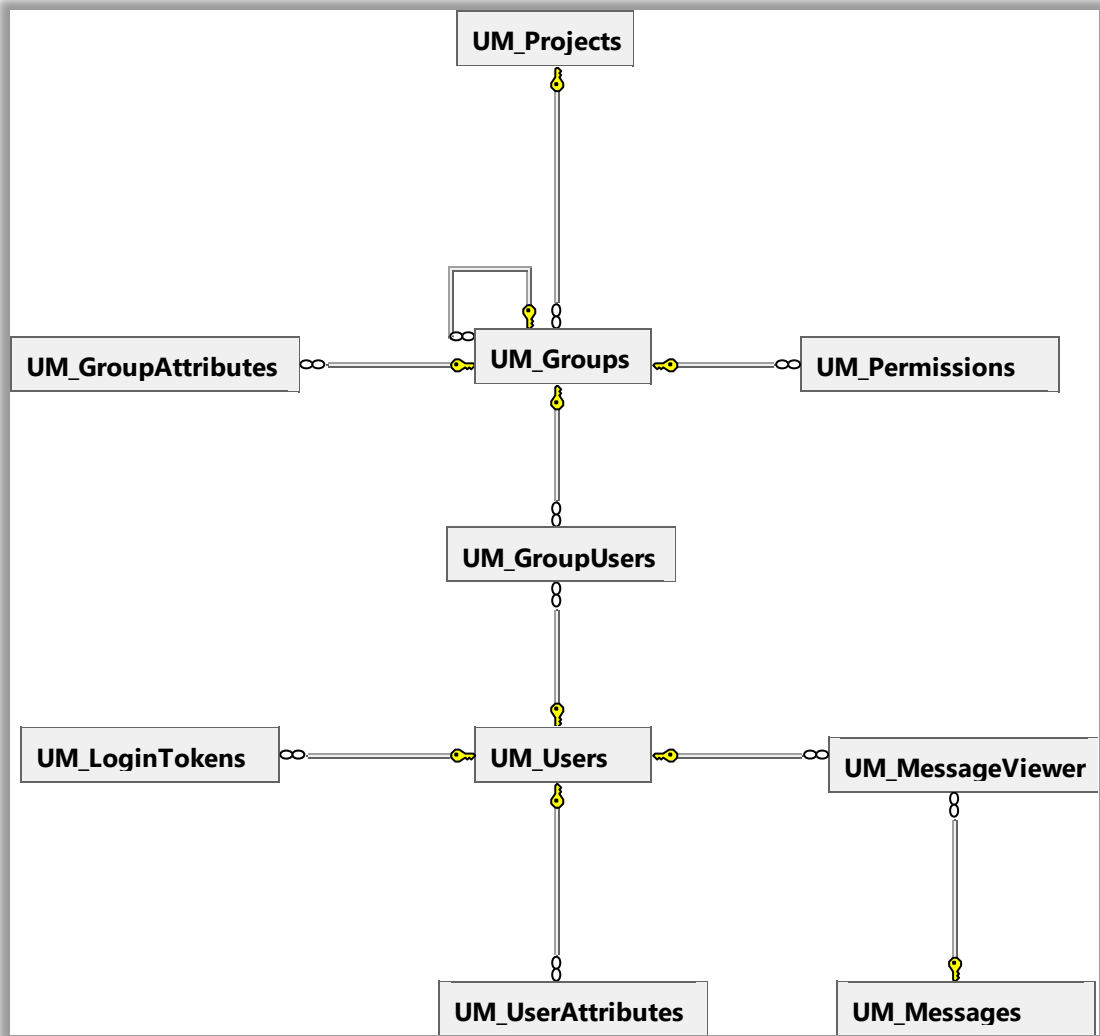


Figure 17. User rights and attribute definitions

10.1.2 User Management Module Database Tables and Columns

Table 9. User Management Module database tables and fields

Table name	Table Description	Columns
UM_AttributesSchemaNodes	According to category and attribute types attribute schema nodes are created. For example "Provider ID", which itself is included in attribute schema	ID
		Name
		AttributeCategoryID
		AttributeTypeID
		AttributesSchemaID
		DateCreated
		DateChanged
		DateDeleted
UM_AttributesSchemas		HashCode
		ID
		ProjectID

Table name	Table Description	Columns
	Attributes schema defined at the module level, which acts as attribute schema node grouper	Name
		DateCreated
		DateChanged
		DateDeleted
		Hashcode
UM_GroupAttributes	Additional information definition for the groups, which is created according to attribute schemas, attribute categories, attribute types and attribute schema nodes	ID
		GroupID
		AttributesSchemaNodeID
		Value
		DateCreated
		DateChanged
		DateDeleted
		Hashcode
UM_Groups	Groups defined at the module level	ID
		ParentID
		ProjectID
		Name
		DateCreated
		DateChanged
		DateDeleted
		Hashcode
UM_GroupUsers	Table for users and user groups relation	ID
		UserID
		GroupID
		AccessLevel
		DateCreated
		DateChanged
		DateDeleted
		Hashcode
UM_LoginTokens	Table for tokens created as a result of authorization in various modules	ID
		LoginToken
		UserID
		ExpireDate
		LastAccessDate
		DateCreated
		DateChanged
		DateDeleted
		Hashcode
		DeleteReason
		ID

Table name	Table Description	Columns
UM_Messages	Table of messages at the module, group and user level	Subject
		Text
		ObjectID
		Type
		DateCreated
		DateChanged
		DateDeleted
UM_MessageViewer	Table of read/unread messages	ID
		MessageID
		UserID
		DateCreated
		DateChanged
		DateDeleted
UM_Permissions	Table for the permissions on View, Add, Edit, Delete according to the groups and the resources of the module	ID
		GroupID
		ResourceID
		RuleValue
		DateCreated
		DateChanged
		DateDeleted
UM_Projects	Modules table	HashCode
		ID
		Name
		DateCreated
		DateChanged
		DateDeleted
		IsActive
UM_Resources	Table of the resources, where it is possible to store virtual addresses, as well as the addresses of the pages in various modules and components of those pages. For example: module/pages/default.aspx	HashCode
		ID
		ParentID
		ProjectID
		Name
		Description
		Type
		Value
		DateCreated
		DateChanged
		DateDeleted
		HashCode
		ID

Table name	Table Description	Columns
UM_UserAttributes	Additional information definition for users, which is created according to the attribute schemas, attribute categories, attribute types and attribute schema nodes	UserID AttributesSchemaNodeID Value DateCreated DateChanged DateDeleted Hashcode
UM_Users	Users table	ID LoginName Password FirstName LastName Email Address OrganizationName Department Division Position Telephone IsSuperAdmin IsActive UserCategoryID PasswordExpirationDate DateCreated DateChanged DateDeleted Hashcode

10.1.3 Table Relations

Database tables are related to each other by the corresponding columns and using defined logic. **Figure 18** and **Figure 19** show diagrams for those relations and

Table 10 contains detailed information about the relations and relation types. Also, the definitions are available that describe the purpose of those relations.

Table 10. Connections between database tables and their types

Relation name	Table(1)	Table (2)	Relation Type	Description
FK_UM_Groups_UM_GroupsParent	UM_Groups	UM_Groups	One To Many	Parent-child relation between the User management Module groups
FK_UM_Groups_UM_Projects	UM_Groups	UM_Projects	One To Many	Groups relation to the modules
FK_UM_GroupUsers_UM_Groups	UM_GroupUsers	UM_Groups	Many To Many	Users relation to the user groups
FK_UM_GroupUsers_UM_Users	UM_GroupUsers	UM_Users	Many To Many	User groups relation to the users in the group
FK_UM_LoginTokens_UM_Users	UM_LoginTokens	UM_Users	One To Many	User token relation to the authorized users
FK_UM_MessageViewer_UM_MessageViewer	UM_MessageViewer	UM_Messages	One To Many	Read/unread messages relation to the messages. Is used for messages management under STOP-C modules
FK_UM_MessageViewer_UM_Users	UM_MessageViewer	UM_Users	One To Many	Read/unread messages relation to the users. Is used for messages management under STOP-C modules
FK_UM_Permissions_UM_Groups	UM_Permissions	UM_Groups	One To Many	Permissions relation to the groups. Is used for roles management under STOP-C modules
FK_UM_Permissions_UM_Resources	UM_Permissions	UM_Resources	One To Many	Permissions relation to the resources. Is used for roles management under STOP-C modules
FK_UM_Resources_UM_Projects	UM_Resources	UM_Projects	One To Many	Resources relation to the modules. Is used for roles management under STOP-C modules
FK_UM_UserAttributes_UM_AttributesSchemaNodes	UM_UserAttributes	UM_AttributesSchemaNodes	One To Many	User attributes relation to the attribute schema nodes. Additional information about users.
FK_UM_UserAttributes_UM_Users	UM_UserAttributes	UM_Users	One To Many	Users attributes relation to the users. Additional information about users.

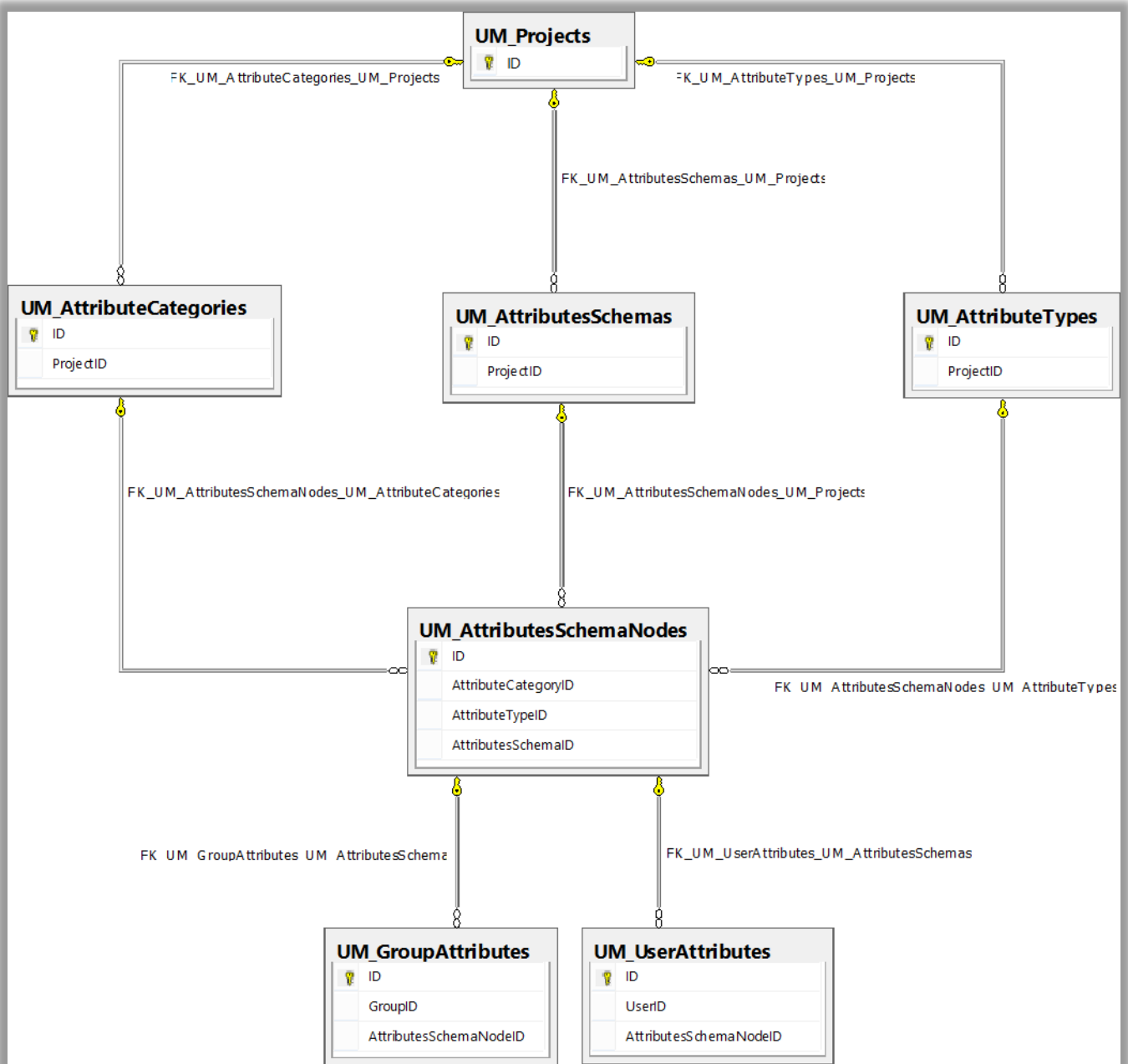


Figure 18. Attribute schemas

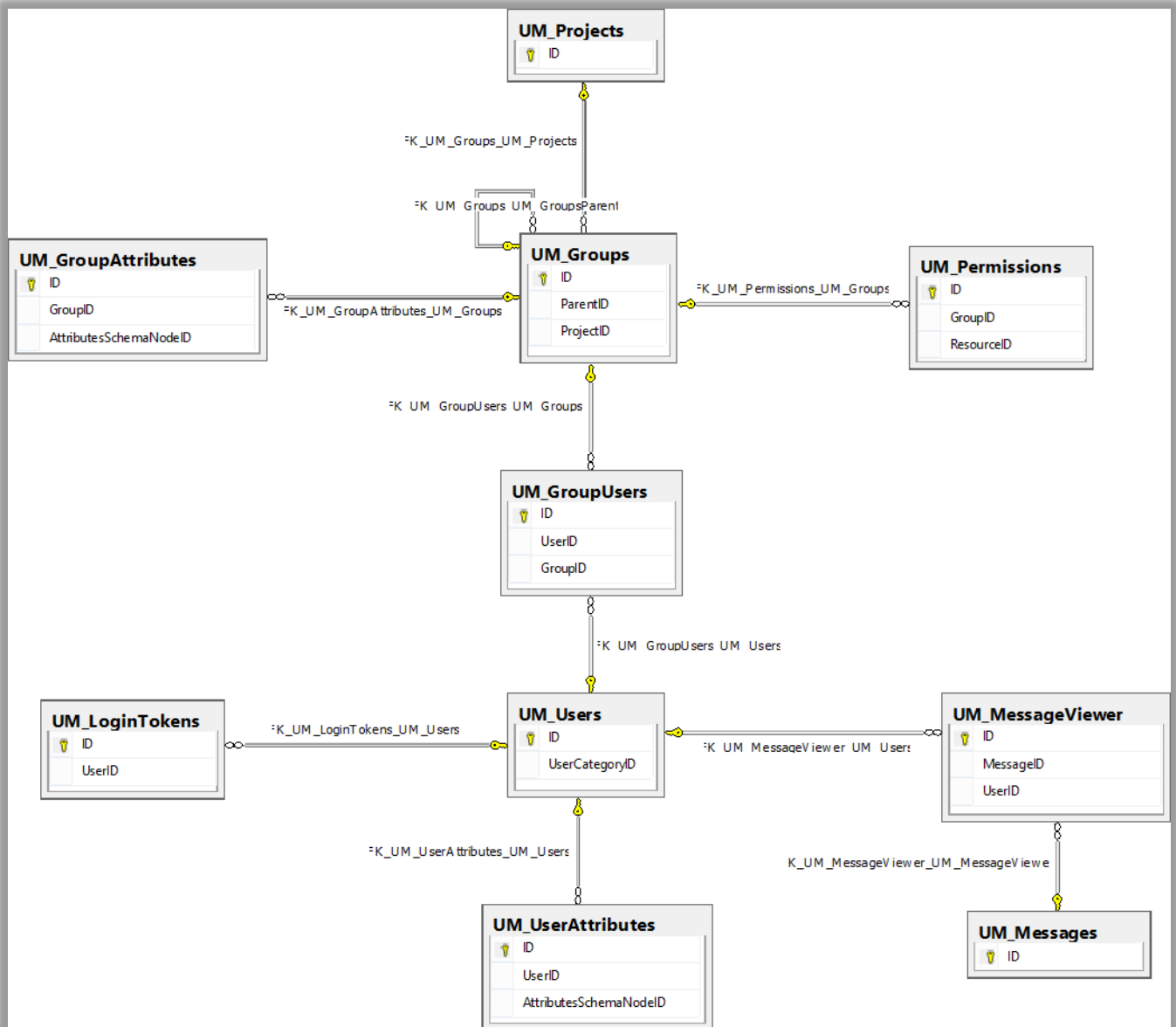


Figure 19. Users and attribute definitions

10.2 Central Module (Portal) Database

Portal database architecture is realized by the pair of 2 databases – Relational database (Microsoft SQL) and document-oriented database (MongoDB). For the detailed information about addresses and the technologies, refer to **Chapter 12.2**.

10.2.1 Microsoft SQL: Database Structure

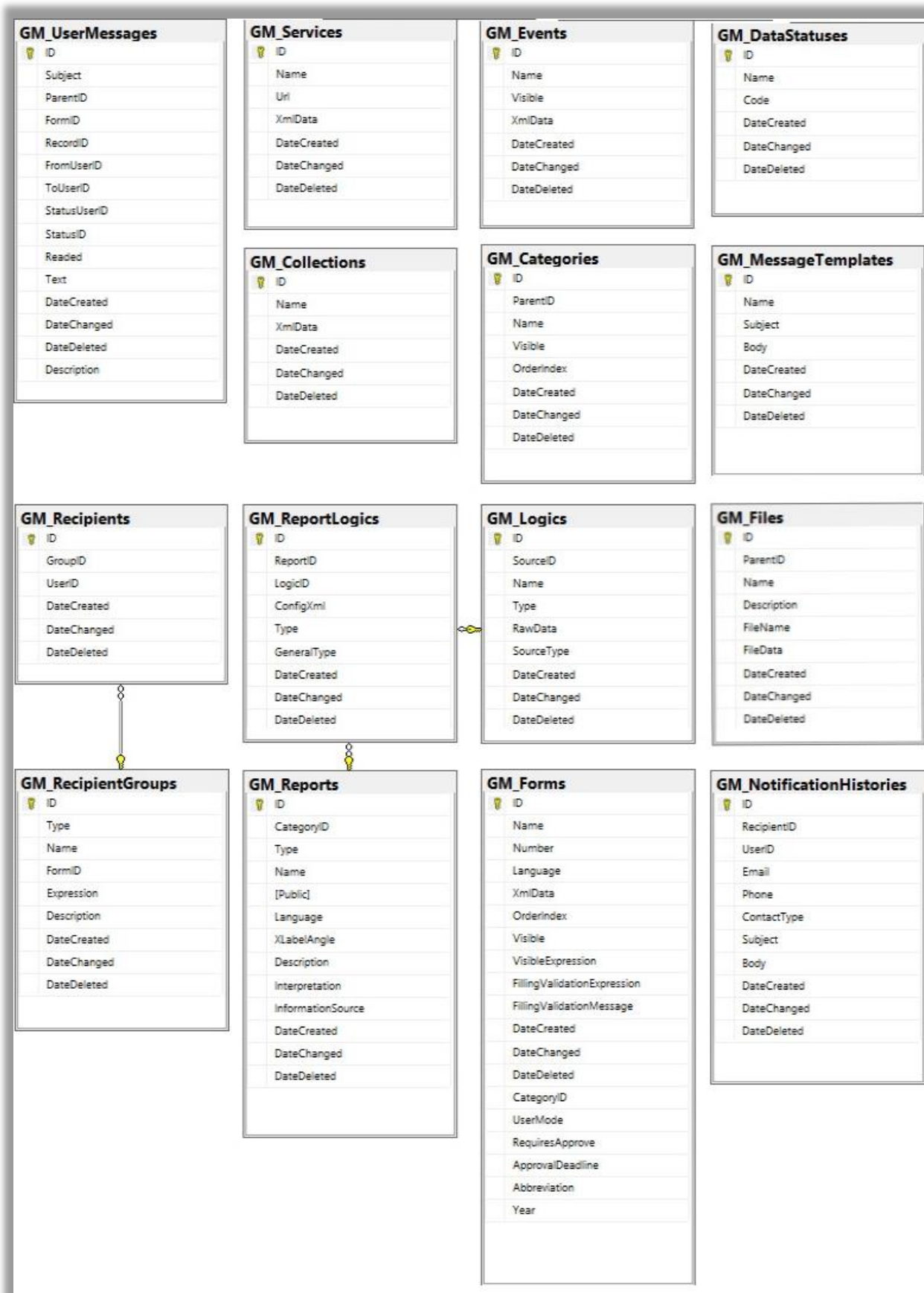


Figure 20. Table structure of the Portal database

10.2.2 Microsoft SQL: Database Tables and Columns

Table 11. Portal database tables and fields

Table	Column		Description
GM_Services	Description of the external services		
	ID	Unique Identifier	
	Name	Name	
	Url	Service Address	
	XmlData	List of the directory fields stored as XML	
	DateCreated	Date of creation	
	DateChanged	Data of the last change	
	DateDeleted	Date of Deletion	
GM_UserMessages	Message between the users		
	ID	Unique Identifier	
	Subject	Message Subject	
	ParentID	Parent Message ID	
	FormID	Message Form ID	
	RecordID	Record ID	
	FromUserID	ID of the sender user	
	ToUserID	ID of the receiver user	
	StatusUserID	ID of the User status	
	StatusID	Message Status ID	
	Readed	Read/Unread status	
	Text	Message text	
	DateCreated	Date of record creation	
	DateChanged	Date of the last change	
	DateDeleted	Date of Deletion	
	Description	Description	
GM_Recipients	Table for the message receiver user		
	ID	Unique Identifier	
	GroupID	Users Group ID	
	UserID	Receiver User ID	
	DateCreated	Date of record creation	
	DateChanged	Date of the last change	
	DateDeleted	Date of Deletion	
GM_Files	Table of the files uploaded to the form		
	ID	Unique Identifier	
	ParentID	Parent file ID	
	Name	Name	
	Description	File description	
	FileName	File Name	
	FileData	File itself	
	DateCreated	Date of record creation	
	DateChanged	Date of the last change	
	DateDeleted	Date of Deletion	

Table	Column	Description
GM_NotificationHistories	Table of sent messages	
	ID	Unique Identifier
	RecipientID	Reviver user ID
	UserID	Sending User ID
	Email	User Email
	Phone	User Phone
	ContactType	Type of the contact
	Subject	Message Subject
	Body	Message body
	DateCreated	Date of record creation
	DateChanged	Date of the last change
	DateDeleted	Date of Deletion
GM_RecipientGroups	Table of the message receiver group	
	ID	Unique Identifier
	Type	Group Type
	Name	Group Name
	FormID	Form ID
	Expression	Expression
	Description	Description
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_MessageTemplates	Mesagge templates Table	
	ID	Unique Identifier
	Name	Name
	Subject	Messages template subject
	Body	Message template body
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_Events	Calendar Events Table	
	ID	Unique Identifier
	Name	Name
	Visible	Visibility status (Yes/No)
	XmlIData	List of the directory fields stored as XML
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_Categories	Table for the reports and form categories	
	ID	Unique Identifier
	ParentID	Parent category Unique Identifier
	Name	Category Name
	Visible	Visibility status in menu (Yes/No)

Table	Column	Description
	OrderIndex	Order index
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_Collections	Table contains directory structure definition	
	ID	Unique Identifier
	Name	Name
	XmlData	List of the directory fields stored as XML
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of deletion
GM_DataStatuses	Table contains form data status (Sent, Rejected, Confirmed, etc.)	
	ID	Unique Identifier
	Name	Name
	Code	Data code
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_Forms	Table for form definitions	
	ID	Unique Identifier
	Name	Name
	Number	Unique Number
	XmlData	Form structure definition in XML (fields list tables, types, groups, etc.)
	OrderIndex	Order Index
	Visible	Visibility status for the use
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
	CategoryID	Category Unique Identifier
	UserMode	Mode of form filling by the user
GM_Logics	Table contains logic of data processing	
	ID	Unique Identifier
	FormID	Form Unique Identifier (the form from where the data is retrieved)
	LogicID	Parent Logic Unique Identifier (from where the data is retrieved)
	Name	Logic Name
	Type	Logic type (String command (Query) if the logic is built in the designer)
	RawData	Logic or command stored as XM
	SourceType	Data source type (form or logic)
	DateCreated	Date of record creation

Table	Column	Description
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_ReportLogics	Relational table for linking reports with logic	
	ID	Unique Identifier
	ReportID	Report Unique Identifier
	LogicID	Logic Unique Identifier
	ConfigXml	Configuration XML
	Type	Relation Type
	GeneralType	Relation General Type
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion
GM_Reports	Table contains reports definitions (diagram/table)	
	ID	Unique Identifier
	CategoryID	Category Unique Identifier
	Type	Type (diagram/table)
	Name	Name
	Public	Status if report is public (In case of Yes, report is visible for all the user, otherwise only for the administrator)
	Language	Report Language
	XLabelAngle	Angle for X axis text
	Description	Short Description
	Interpretation	Interpretation (filled by the user)
	InformationSource	Information source (filled by the user)
	DateCreated	Date of record creation
	DateChanged	Date of last change
	DateDeleted	Date of Deletion

11 Necessary Libraries

To fully function, the STOP-C system requires additional necessary libraries. The following additional libraries are used in the system and are necessary requirements:

11.1 NPOI

This library is .NET version of POI Java Project³³. POI is an open source project³⁴ using which it is possible to read and write .xls, .doc, and .ppt. In STOP-C system NPOI is used to import and export Excel files, without installation of Microsoft Office on server side. This solution is more effective and efficient, than running Microsoft Excel ActiveX in background mode.

11.2 ASP Chart Control

Chart controls^{35,36} supports creation of ASP.NET and Windows Forms applications to create simple, intuitive and visually easily absorbable diagrams for complicated statistical and financial data analytics. Using this library, it is possible to create 35 charts and diagrams of various types.

11.3 DevExpress

DevExpress³⁷ is a collection of tools and libraries for software development, which using additional software plugins makes software development faster and more effective. In STOP-C system DevExpress ASP.NET user interface controls are used.

11.4 NHibernate

NHibernate³⁸ is an Object-Relational Mapper (ORM), that means translation of STOP-C system data from .NET environment into the format compatible with any relational Database (Microsoft SQL, Oracle DB, IBM DB2, mysql, postgres SQL and others). As a result, STOP-C system is not depending on specific database and any relational database can be used according to the specific task.

11.5 Bootstrap

Bootstrap³⁹ is tool for creation of user interface controls and represents front-end⁴⁰ environment for designing web-page and web-applications. Bootstrap contains HTML- and CSS-based design templates, using of which it is possible to create STOP-C user interfaces (forms, buttons, navigation and other components of the interface; it is also possible to create and use Javascript addons).

³³ <http://poi.apache.org>

³⁴ <https://npoi.codeplex.com>

³⁵ <https://code.msdn.microsoft.com/mschart>

³⁶ <https://msdn.microsoft.com/en-us/library/dd456632.aspx>

³⁷ <https://www.devexpress.com>

³⁸ <http://nhibernate.info>

³⁹ <http://getbootstrap.com/css>

⁴⁰ https://www.wikiwand.com/en/Front-end_web_development

12 Hardware Requirements

12.1 Minimum Architecture

Hardware requirements

- CPU: 4x – 2000Mhz
- RAM: 16GB
- HDD: 200GB

Software environment requirements

- Microsoft Windows Server 2008 R2⁴¹
- Microsoft Information Services (IIS)⁴²
- Microsoft Visual Studio.Net 2010/2012/2013⁴³
- ASP.NET, DevExpress UI controls, Microsoft Chart controls
- NHibernate-compatible relational database (e.g.: Microsoft SQL)

12.2 Components Addresses

Table 12. STOP-C Ecosystem component addresses

STOP-C System Component	Address	IP
Portal	http://Birthregistry.moh.gov.ge/HMIS/HMIS.StopC.web	172.17.216.53
Portal Database - MongoDB	N/A	172.17.7.132
Portal Database - Microsoft SQL	N/A	172.17.7.132
User Management Module	http://birthregistry.moh.gov.ge/HMIS/HMIS.usermanagement.web	172.17.216.231
User Management Module Database	N/A	172.17.7.130
Common Data Module	http://birthregistry.moh.gov.ge/HMIS/HMIS.commondata.web	172.17.216.231
Common Data Module Database	N/A	172.17.7.130
Service Bus	http://birthregistry.moh.gov.ge/HMIS/HMIS.messaging.web	172.17.216.231
SMS Module	N/A	
SMS Module Database	N/A	

⁴¹ [https://technet.microsoft.com/en-us/library/dd349801\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/dd349801(v=ws.10).aspx)

⁴² <https://www.iis.net>

⁴³ <https://www.visualstudio.com>

12.3 Recommendations andout Backup Procedure

Due to the average critical level of STOP-C system, at this stage implementing additional High Availability solution technology is not necessary. However, to secure system and the data from possible errors, it is recommended to use standard Backup scheme with the following periodicity:

- Full backup
 - Full reservation of system components and data
 - Period: Weekly
- Incremental Backup
 - Reservation only of those data, which were changed after the latest full backup
 - Period: Hourly

13 Installation and Configurations

For installation or backup recovery of STOP-C system, the platform must be prepared according to the following order:

1. Installation and standard configuration of operation system (Microsoft Windows Server 2008/2012 R2)
2. Installation and configuration of Microsoft Internet Information Server
3. Installation and configuration of Microsoft .NET Framework
4. Installation and configuration of DevExpress tools and libraries
5. Installation and configuration of Microsoft Chart Controls
6. Installation and configuration of Database servers
 - a. Creation of database and the tables (or recovery from backup package) – Microsoft SQL
 - b. Creation of database (or recovery from backup package)– MongoDB
7. Execute file: iis.config.cmd.

14 Appendix A: Expression Syntax

For validations of and relations between form field components and attributes. STOP-C system uses custom simple syntax of the expressions.

14.1 Operators

Table 13. Expression syntax: operators

Operator	Definition	Comment
!	Logical "No"	Function IsEmpty(value) returns true/false depend on parameter value is empty or not. But if we want reverse option Is not empty, has any value, than we write "IsEmpty(value)"
+	Addition	
++	Increment	Typically adds the integer 1
-	Subtraction	
--	decrement	Typically subtracts the integer 1
==	Equality	e.g. a==b, checks value of a, if it is equal to b or not. This returns reference equality (identity test)
!= <>	Not equal.	e.g. a!=b or a<>b, checks if a is not equal to b
<=	Less than or equal to	
>=	Greater than or equal to	
<	Less than	
>	Greater than	
&&	Logical AND	E.g. a==b && b==c means that a is equal to b and b is equal to c
	Logical OR	E.g. a==b b==c means that a is equal to b or b is equal to c
^	Exclusive XOR	a^b - a power of b
&	Concatisation	String Concatisation Operation. If we have firstName and lastName and want to concatinare their values and separate with "-", we use firstName & ' - ' & lastName
%	Percentage	a%b same as [a / 100 * b]
	Reminder	
*	Reminder	
/	Multiplication	

14.2 Functions

Table 14. Expression syntax: functions

Function	Description
sqrt(value)	Square Root
sin(value)	Sinus
cos(value)	Cosinus
tan(value)	Tangens
asin(value)	Arcsinus
acos(value)	Arccosinus
atan(value)	Arctangens
abs(value)	Absolute value
ceil(value)	Rounds up the nearest integer
exp(value)	Exponential value
floor(value)	Rounds to an integer towards 0
log(value, base)	Logarithm calculator with Base
log10(value)	The logarithm to the base 10
pow(value, base)	Exponent used to raise the base
or(value1, value2)	Binary "OR" Operation
and(value1, value2)	Binary "AND" Operation
xor(value1, value2)	Exclusive „OR“ (XOR)
mod(value1, value2)	Modulus operation - returns remainder but also calculates division in the process (E.g: mod(19, 4) returns 3 as 19 divided into 4 is equal to 4 and reminder is 3)
substring(value, index)	Substring method attempts to extract characters from index startIndex to index startIndex + length - 1. To extract a substring that begins with a particular character or character sequence, call a method such as IndexOf or LastIndexOf to get the value of startIndex.
substring(value, index, count)	Returns sub string Index and substring length (E.g. if value is equal to abcdefg, then substring (value, 3, 2) returns d, as count started from 0 and thirds 3 symbol is d)
getdatetime(value)	Returns Current Date and Time
getDate()	Returns Current Date
getDate(year)	Returns just the year (Month, day and others will be current)
getDate(year, month)	Returns Date (others :Day, Hours and so on will be current)
getDate(year, month, day)	Returns Date (is necessary for Datetime operations, e.g. addDays(getDate(2017, 02, 21), -15))
getLength(value)	Returns length
isEmpty(value)	The ISEMPTY function returns TRUE if the value is a blank cell or uninitialized variable. The ISEMPTY function returns FALSE if the value is a cell or variable that contains a value
isDate(value)	Determines if a value is a date
isDay(value)	Determines if a value is a day (More than 0 and less than 32)

isMonth(value)	Determines if a value is a Month (More than 0 and less than 13)
isYear(value)	Determines if a value is a Year
isNumber(value)	Determines if a value is a number
isInteger(value)	Determines if a value is a Integer
rgx(value, exp)	Defines a search pattern with Regex
min(a, b)	Returns Minimum
max(a, b)	Returns maximum
if(value, a, b)	if(x==1, a, b) – if x=1, returns a, or b
comp(value)	Comparator (Returns -1, 0, 1)
lower(value)	Changes to Lower letter
upper(value)	Changes to Upper letter
contains(value, text)	Function returns the location of one text string inside another. (Returns true/false), e/g contains (value, "hello")- If Text in value Contains the word hell
startsWith(value, text)	Returns a logical value indicating whether a text value substring was found at the beginning of a string
endsWith(value, text)	Returns a logical value indicating whether a text value substring was found at the end of a string
trim(value)	The trim() method removes whitespace from both sides of a string
LTrim(value)	This function returns a string with whitespace stripped from the beginning of string
RTrim(value)	Returns a string with whitespace (or other characters) stripped from the end of string
get(collection, nameOrIndex)	Returns item of array by index or value of dictionary by key
getDaysInMonth(value)	
getDaysInMonth(value, year)	Returns the number of days in a specific month of a year (if year is not defined in parameters, it counts current year) e.g.: getDaysInMonth(2) returns 28, but getDaysInMonth(2, 2016) returns 29
getYears(date)	Returns the year for the specified date (2017-02-17 15:32:25)
getMonths(date)	Returns the month (from 0 to 11) for the specified date, (2017-02-17 15:32:25)
getDays(date)	
getDayOfMonth(date)	Returns the name of the weekday (2017-02-17 15:32:25) (If not Returns primitive int value for the day-of-month.
getDayOfWeek(date)	Returns the day of the week
getDayOfYear(date)	Returns primitive int value for the day-of-year.
getHours(date)	Returns Hours (2017-02-17 15:32:25)
getMinutes(date)	Returns Minutes (2017-02-17 15:32:25)
getSeconds(date)	Returns Seconds (2017-02-17 15:32:25)
getTotalDays(date)	Returns total number of days from 0001-01-01 01:01:01- till indicated date

getTotalHours(date)	Returns total number of Hours from 0001-01-01 01:01:01till indicated date
getTotalHours(date)	Returns total number of Minutes from 0001-01-01 01:01:0101till indicated date
getTotalSeconds(date)	Returns total number of Seconds from 0001-01-01 01:01:0101till indicated date
addYears(date, count)	To add a given number of years to a date
addMonths(date, count)	To add a given number of months to a date (count quantity)
addDays(date, count)	To add a given number of days to a date (count quantity)
addHours(date, count)	To add a given number of hours to a date (count quantity)
addMinutes(date, count)	To add a given number of minutes to a date (count quantity)
addSeconds(date, count)	To add a given number of Seconds to a date (count quantity)
select(collection, selector, condition)	Selects rows/items. Finds data collection by condition (if condition is specified) E.g. there is a collection Products with three columns: Name, Amount, Cost. To select all product, which has name and value more than 10: select(Products, get(@, 'Name') & ' ' & get(@, 'Cost'), get(@, 'Amount') > 10) @ - Means current record, so function “select “should check all records in the collection, checks if each record meets the conditions and returns result to selector.
sum(collection)	Returns sum of numeric values in collection. Like Select, if we want to retrieve Products from the collection, which has price more than 100: sum(select(Products, get(@, 'Amount'), get(@, 'Cost') > 100)) min – works the same ways as sum, but returns minimum value max - works the same ways as sum, but returns maximum value