Statistics Poland

TECHNICAL SPECIFICATION

The prototype of IT tool supporting preparation of an on-line form to registration data in the Household Budget Survey

Statistics Poland

Eurostat

TECHNICAL SPECIFICATION

The prototype of IT tool supporting preparation of an on-line form to registration data in the Household Budget Survey

Table of contents

1. Preface

1.1. Purpose of the document

The document is intended for the project team for independent software development.
The purpose of the document is to describe the design environment.

1.2. Definitions

Abbreviation	*Description*
API	An application program interface (API) is a set of routines, protocols, and tools for building software applications. Basically, an API specifies how software components should interact. Additionally, APIs are used when programming graphical user interface (GUI) components
IDE	An integrated development environment - is a software application that provides comprehensive facilities to computer programmers for software development (e.g.Eclipse, Visual Studio)
MVC	Model–view–controller is a software design pattern commonly used for developing user interfaces which divides the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user
JSON	JavaScript Object Notation (JSON) is an open standard file format, and data interchange format, that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and array data types (or any other serializable value).
CSV	A CSV is a comma-separated values file, which allows data to be saved in a tabular format.
IIS	Internet Information Services (IIS, formerly Internet Information Server) is an extensible web server software created by Microsoft for use with the Windows NT family.
VUE	Vue.js (commonly referred to as Vue) is an open-source Model–view–view model JavaScript framework for building user interfaces and single-page applications.
EDP	Electronic data processing can refer to the use of automated methods to process data or external system/ database
PROTOTYPE	The prototype of IT tool supporting preparation of an on-line form to registration data in the Household Budget Survey
HBS	Household Budget Survey

Source: https://en.wikipedia.org

2. System architecture

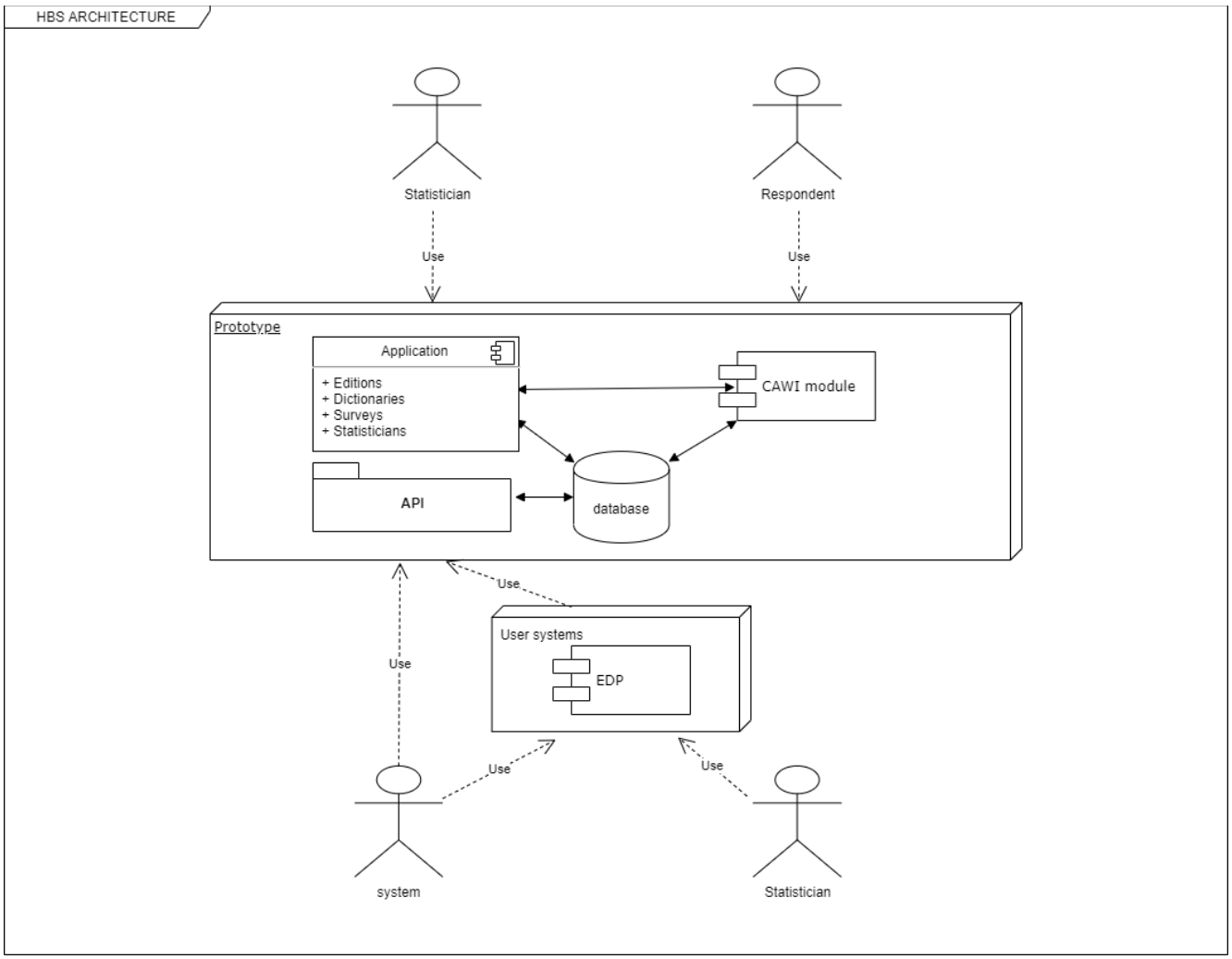


Diagram of components: General view of the prototype architecture.

2.1 Description of the prototype elements

PROTOTYPE allows statisticians to design a questionnaire for HBS and launch it as a CAWI module to be filled out by respondents. Through the API, it supports dictionaries and user accounts in the area of their creation and saving to the database. It consists of the following main elements:

Application - a software module containing GUI and API enabling the user to conveniently build a questionnaire designed to meet specific user needs.

API - procedures used to perform tasks not supported in the GUI, such as: creating and deleting accounts of respondents of a given survey edition, filling out dictionaries and transferring data collected in the survey to EDP or external system/ database

CAWI - a module with a questionnaire for the respondent, designed and made by a statistician

Database - data container - containing data necessary to develop the questionnaire and launch the CAWI module, including metadata, survey definitions, user accounts, registered data.

EDP - system processing data from household budget survey. This tool also prepares input data e.g. dictionaries, files. The system is powered by data obtained from the CAWI module.

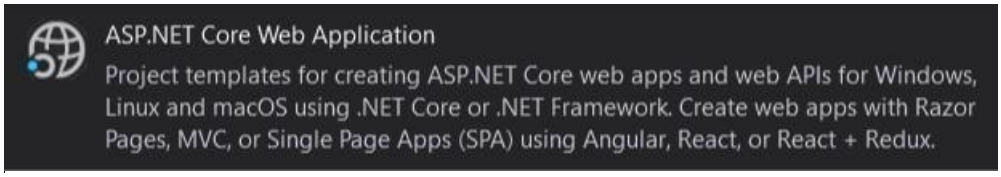
As a part of preparing the questionnaire to complete and generating the CAWI module, the application provides:

- visual design of the form (using the drag and drop technique)
- defining validations
- support for many editions of the survey
- support for dictionaries (import and view)
- support for card indexes (import and view)
- user account support (import and view)
- access via api (in case of import)

The application can be accessed via a browser from the server, local network or the Internet. The application interface is embedded in the browser window. This gives the respondent the opportunity to access from multiple devices.

3. Prototype design software platform

The system is built in Windows 10 operating system using IDE Visual Studio 2019 with using components such as: VUE, Entity Framework.
The project uses the **.NET CORE 3.1** platform to create solutions that run on various operating systems.
The result of the implementation of the project assumptions is a prototype WEB application built using the template



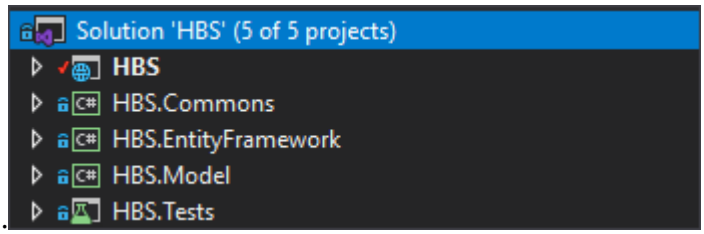
The prototype is developed as a WEB application in the SPA model - Single Page Application using MVC - Model View Controler template based on a relational database system.
Among others HTML, CSS and JavaScript as well as dedicated frameworks e.g. VUE are used to create the frontend. In the area of backend C# language, the database, TSQL language, middle layer mechanisms, e.g. Entity Framework are used.
In the database layer of the prototype, the MS SQL Server 2012 database is used.

4. Software structure

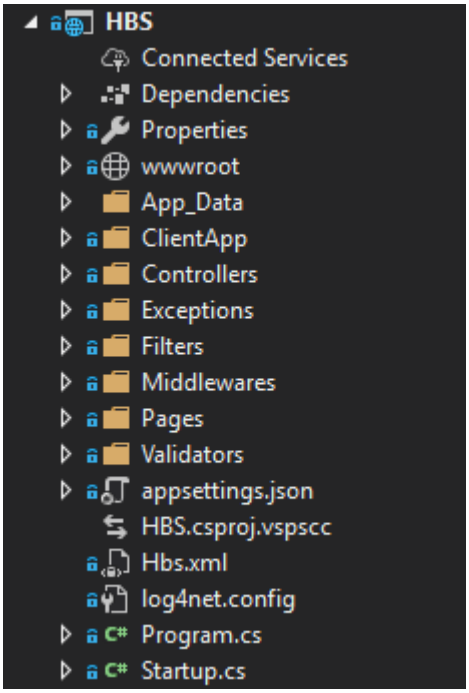
A solution called **HBS.sln** for PROTOTYPE elements is built in the Visual Studio environment.
The solution in Visual Studio is a basic container grouping elements of a specific product (e.g. project groups).
As part of the solution, basic branches / areas / modules are created containing specific design elements.
To modify the source code, run Visual Studio and open the solution file.
The solution is packed into .ZIP file. The **HBS_SOLUTION.zip** file should be unpacked in any folder on a computer.

4.1 Application components

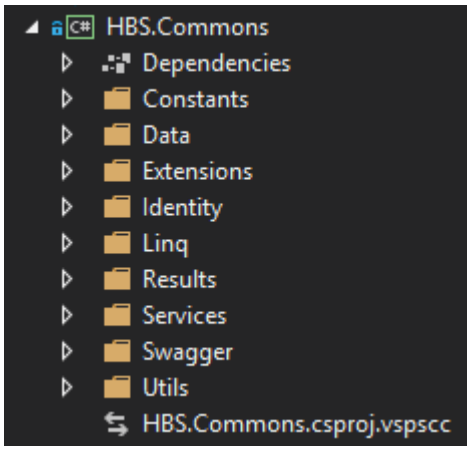
The solution contains five basic projects grouping the source code to implement / to support specific, logically related tasks – projects.



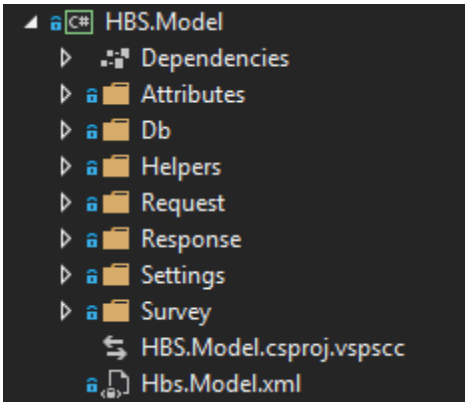
The main project includes the startup program *program.cs* and a set of class packages grouping the main functionalities.



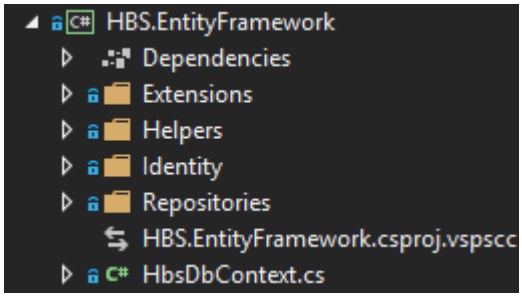
The HBS.Commons project includes general components and classes.



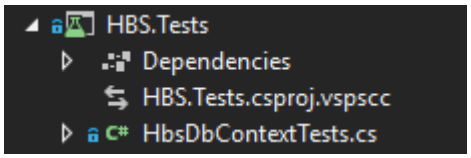
The HBS.Model project contains elements related to database support, its mapping in the form of program classes enabling the implementation of CRUD operations, i.e. create, read, update and delete. A model is made up of entity classes and a context object that represents a session with the database, allowing user to query and save data.



The HBS.EntityFramework project includes software components that use HBS.Model elements for database operations. EntityFramework Core can serve as an object-relational mapper (O/RM), enabling developers to work with a database using .NET objects, and eliminating the need for most of the data-access code they usually need to write. With EntityFramework Core, data access is performed using a model.



The HBS.Tests project includes unit tests.



4.2.API

The system provides an API for handling dictionaries, user accounts and the transfer of collected data to the EDP system.

The API service provides support for the prototype resources through the REST API using the parameterized methods "GET", "POST", "DELETE".

Resource Dictionaries

DictionaryEntries - POST method to upload dictionary entries from CSV by file name:

.../api/DictionaryEntries/{name}

Method requires one parameter:

- **name** - a name of the file to be uploaded

For example for a file's **name = *ISICRev4.csv***

API call is ... ***/api/DictionaryEntries/ISICRev4***

Resource Statisticians

Edition – GET method to create respondent Users for symbol Edition

.../api/Users/Edition{**Edition**=string}**&**{**Count**=integer}

Method requires two parameters:

- ***Edition*** -the name of the study edition to which the accounts will be assigned
- ***Count*** - number of generated user accounts

For example to create **25** logins for edition **2020/01**

API call is ***.../api/Users/Edition***?Edition=**2020/01&**Count=**25**

(**&** *-> concatenation of conditions/parameters*)

DeleteUsers – DELETE method to delete respondent Users for symbol Edition

To delete respondent Users for symbol Edition =**2020/01**

API call is ***.../api/Users/DeleteUsers***?**Edition**=**2020/01**

Resource Data

SurveysData GET method to get survey's data for symbol Edition

.../api/Users/ **SurveysData** {**Edition**=string}**&**{**page**=integer}}**&**{**page-size**=integer}

Method requires two parameters:

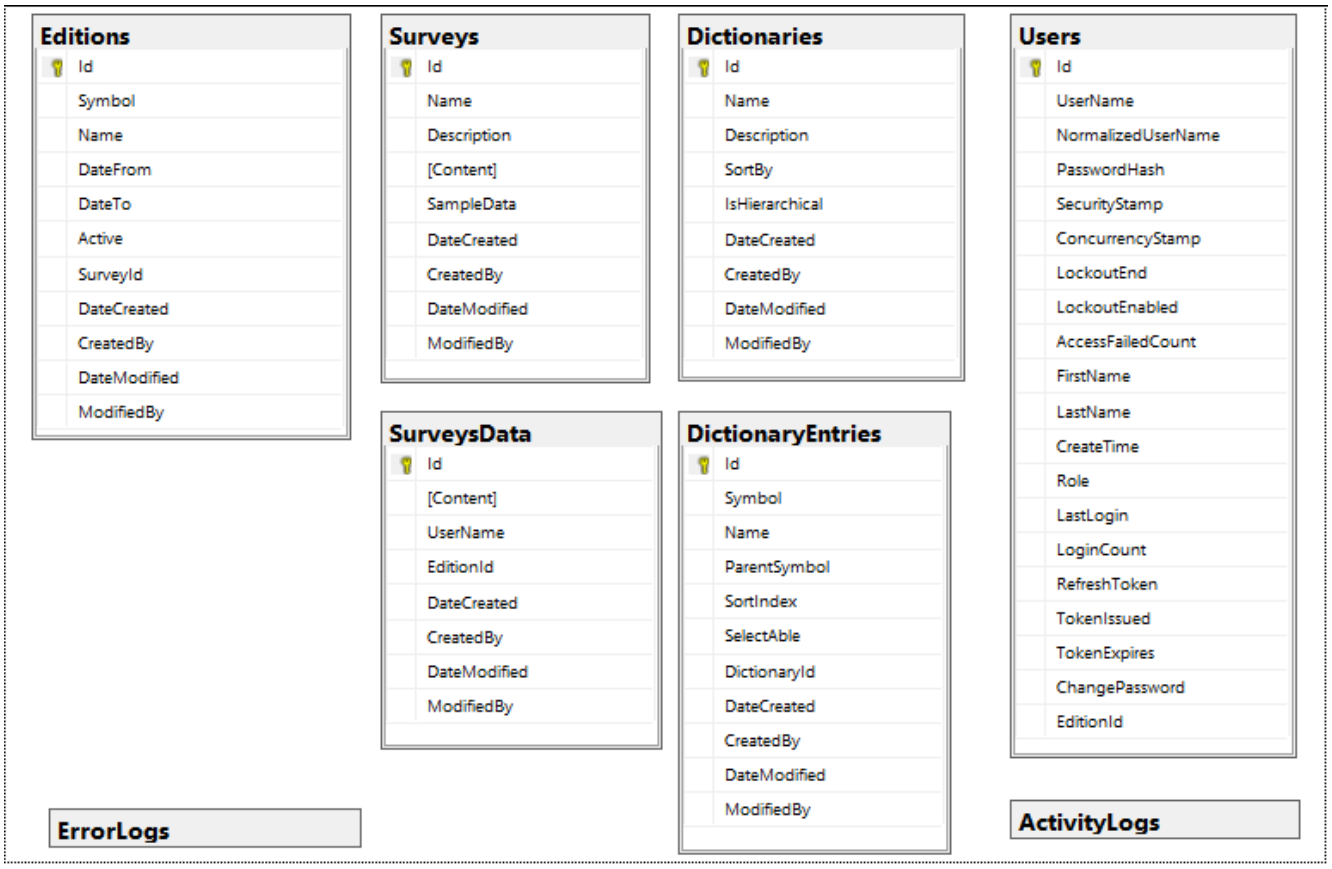
- ***Edition*** -the name of the study edition to which the accounts will be assigned
- **page** - number of data pages to get
- **page-*size*** - page-size of data page (number records per page)

For example to get data for edition **2020/01** , **10** pages of size **5**

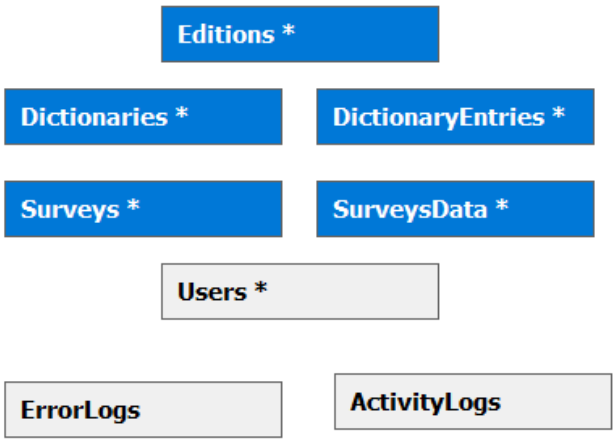
API call is ***.../api/Users/*** **SurveysData** ?**Edition**=**2020/01**&**page**=**10**&**page-size**=**5**

5. Database structure

The system database consists of main and auxiliary tables. Main tables store information about the main system objects such as: editions, dictionaries, projects and users. Auxiliary tables store data about events and application errors.



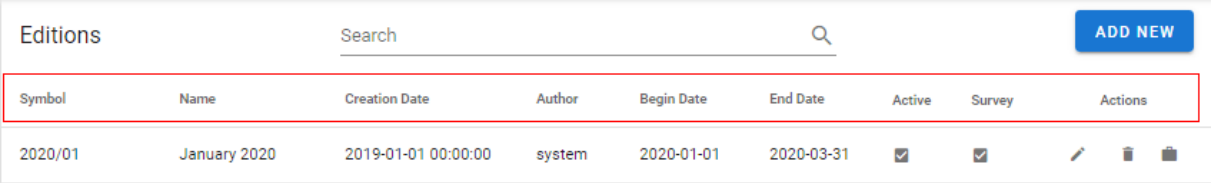
Database tables.



Contents of individual tables:

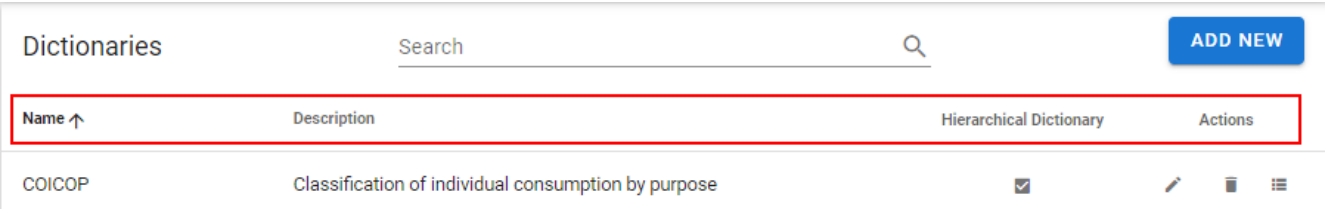
Editions

	Field	*Label GUI*	*Description*
1	Id		Edition id (primary key, unique)
2	Symbol	Symbol	Edition symbol
3	Name	Name	Edition name
4	DateFrom	Begin date	Date start of edtion
5	DateTo	End date	Date end of edtion
6	Active	Active	Status
7	SurveyId		Id of survey
8	DateCreated	Creation Date	Creation edition date
9	CreatedBy	Author	Creation edition user name
10	DateModified		Edition modification date
11	ModifiedBy		Edition modification user name



Dictionaries

	Field	*Label GUI*	*Description*
1	Id		Dictionary id (primary key, unique)
2	Name	Name	Dictionary name
3	Description	Description	Dictionary description
4	SortBy		Field name of sort
5	IsHierarchical	Hierarchical Dictionary	Is the dictionary hierachical (boolean)
6	DateCreated		Creation dictionary date
7	CreatedBy		Creation dictionary user name
8	DateModified		Dictionary modification date
9	ModifiedBy		Dictionary modification user name



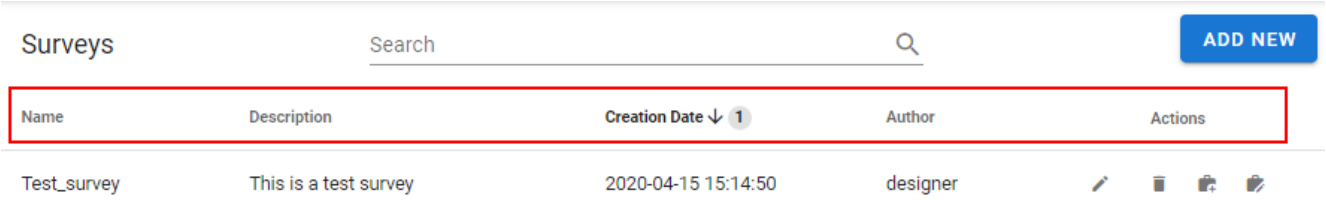
DictionaryEntries

	Field	*Label GUI*	*Description*
1	Id		Dictionary entry id (primary key, unique)
2	Symbol	Symbol	Entry symbol

3	Name	Name	Entry name
4	ParentSymbol		Parent entry symbol for the hierarchical dictionary
5	SortIndex		Sort index
6	SelectAble		Whether the entry is selectable (boolean)
7	DictionaryId		Dictionary id
8	DateCreated	Creation Date	Creation dictionary entry date
9	CreatedBy		Creation dictionary entry user name
10	DateModified		Dictionary entry modification date
11	ModifiedBy		Dictionary entry modification user name

Surveys

	Field	*Label GUI*	*Description*
1	Id		Survey entry id (primary key, unique)
2	Name	Name	Survey name
3	Description	Description	Survey description
4	Content		Form definition
5	SampleData		Survey sample data
6	DateCreated	Creation Date	Creation survey date
7	CreatedBy	Author	Creation survey user name
8	DateModified		Creation survey modification date
9	ModifiedBy		Creation survey modification user name

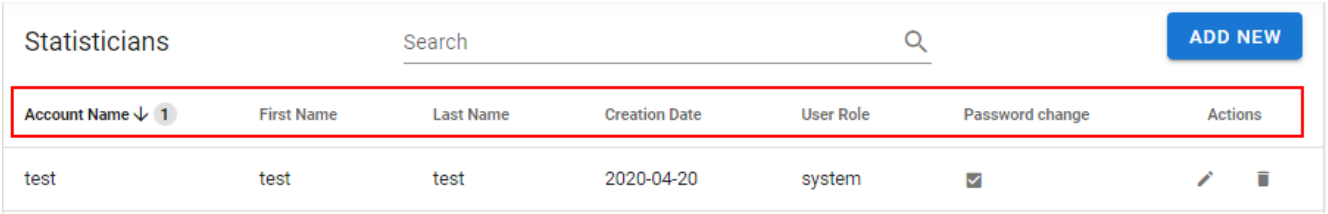


SurveysData

	Field	*Label GUI*	*Description*
1	Id		Surveys data id
2	Content		Surveys data
3	UserName		Account name
4	EditionId		Edition id
5	DateCreated		Creation surveys data date
6	CreatedBy		Creation surveys data user name
7	DateModified		Creation surveys data modification date
8	ModifiedBy		Creation surveys data modification user name

Users

	Field	*Label GUI*	*Description*
1	Id		user ID (primary key, unique, GUID)
2	UserName	Account Name	Account name
3	NormalizedUserName		Normalized user name (upper case)
4	PasswordHash		Password Hash
5	SecurityStamp		Security Stamp JWT (ang. JSON Web Token)
6	ConcurrencyStamp		Concurrency Stamp JWT
7	LockoutEnd		Lockout end
8	LockoutEnabled		Lockout enabled
9	AccessFailedCount		Access failed count
10	FirstName	First Name	First name user
11	LastName	Last Name	Last name user
12	CreateTime	Create Time	Create time account
13	Role	User Role	User role (admin, designer)
14	LastLogin		Last user login
15	LoginCount		Count of user logins
16	RefreshToken		Refresh token
17	TokenIssued		Date token issued
18	TokenExpires		Date token expires
19	ChangePassword	Password change	Force password change
20	EditionId		Edition id to which the user is assigned



6. System installation

6.1 Installation procedure

Installation of the HBS application

6.1.1 Database

The prototype environment

Database Name – **HBS**
Database Server - **MS SQL Server 2012** or **MS SQL Express** (min. version 2012)

Creating the HBS database

- on the selected database server run the script **DB_CREATE.sql**

1. for a mode **Windows Authentication** create the appropriate application user in **Windows domain,**
2. assign the created user rights in the **HBS** database to the dedicated **hbs_app** database role (the role will be created when the script is executed)

or

1. in the case of **SQL Authentication** mode create the appropriate application user on the **MS SQL server,**
2. assign the created user rights in the **HBS** database to the dedicated **hbs_app** database role (the role will be created when the script is executed)

6.1.2 Server WEB

Attention:

The VUE application at the time of building must have the subpath indicated on which it is published on the WEB server (it is necessary for proper linking of js and css files).

*In the installation version, the path is set to **/HBS**. If the application is published at an address with a different sub-path, it is required to reconfigure the VUE application and re-build it and create a new distribution version in the Visual Studio environment.*

This procedure is described in the chapter "Reconfiguring and extending the client side application (VUE)".

The prototype environment

IIS server

Application configuration before installation on the IIS server::

- In file **appsettings.json**:
  - set the appropriate **connection string** to connect to the created database **(DefaultConnection** variable in section **ConnectionStrings**)[1]
  - enter the application address where it is visible to end users (**ApplicationBaseUrl** variable in the **UrlSettings** section) - invalid entries may result in incorrect display of the Swagger[2] page attached to the application
  - the same base address should also be entered in the **Authentication** section in the **JwtIssuer** and **JwtAudience** variables
  - the title and name of the application can be configured in the **Properties** section (**Title** and **Description** variables respectively)

IIS server configuration

1. Install the IIS server on the target server (computer)
2. Install the .NET Core 3.1 runtime environment (installation file https://dotnet.microsoft.com/download/dotnet-core/thank-you/runtime-aspnetcore-3.1.3-windows-hosting-bundle-installer)
3. Create an application pool with the name IIS Manager **HBS AppPool**:
   - .NET CLR version: **No Managed Code**
   - Managed pipeline mode: Integrated
   - Create a pool by clicking OK

If the connection to the database is made using **a domain account**, then this account should be configured for the created pool by setting the **Identity** variable (user's domain name and password) in the **Advanced Settings** window - from that moment the application working in this pool works on the set user.

4. Copy the contents of the customized application's folder into proper place of **inetpub\wwwroot** localisation (depending on the given application subpath, eg. ***inetpub\wwwroot\hbs***)
5. Create Application with IIS Manager. Set the Application Pool to the one you created earlier (**HBS AppPool).**

[1] https://www.connectionstrings.com/sqlconnection/
[2] https://swagger.io/

6.1.3 Reconfiguration and extension of the client side application (VUE).

1. **Adding the language version**
2. **Change to the application paths**
3. **Building the production version of the client side application (VUE)**

Attention.
Testing changes in the application is possible after installation on the environment workstation ***node.js***[3].

The source code of the **VUE** application is located in the **ClientApp** directory of the **HBS** project.

The main variables controlling the application are placed in the environment variable files:

- **.env.development** – for the version that runs in the development environment
- **.env.production** – for the version built as production

Pliki różnią się wartościami zmiennych.

The files differ in variables values.

6.1.3.1 Adding the language version

To add to the PROTOTYPE a new language version:

- in directory \ ClientApp \ src \ i18n create a directory called the two-letter ISO code of a given language (country), e.g. 'pl'
- copy file \ ClientApp \ src \ i18n \ en \ index.js to the directory created in the previous step (i.e. e.g. \ ClientApp \ src \ i18n \ pl \ index.js')
- open the copied **index.js** file and translate all character strings (without changing the property keys). The file format complies with the standard described at http://i18njs.com. If the string contains a curly brace, its content should be left unchanged (it contains a variable substituted in the content)
- in the file **\ClientApp\src\i18n\index.js** you should add the import of the previously created file, example for the Polish language:

```
import pl from "./pl";
```

Then attach the imported variable to the default export, below the file comparing the original version and the version with the Polish translation added:

Original version	Version after adding a Polish translation

`import en from "./en";`
 	
`export default {`
`  en: en`
`};`
	`import en from "./en";`
	`import pl from "./pl";`
	`export default {`
	`  en: en,`
	`  pl: pl`
	`};`

In order for the application to be displayed using the added language, variables (*example for the Polish language*) should be set in the **.env.development** and **.env.production files**:

```
VUE_APP_I18N_LOCALE = 'pl'

VUE_APP_I18N_FALLBACK_LOCALE = 'pl'
```

[3] https://nodejs.org

6.1.3.2 Change for application paths.

Depending on the address at which the application is available, before building a VUE application in the production version it is necessary to set the appropriate variable in the application, i.e. **BASE_URL** in the **.env.production** file.

This variable can be set as a full **url**, e.g.

'https://example.server.com/subpath/'

or as a url relative to the root address of the server, i.e..

'/subpath/' .

In case the application is published at the root address e.g.

'https://example.server.com'

BASE_URL can be set to

'https://example.server.com/' or to '/'

6.1.3.3 Building the production version of the client side application (VUE)

An environment is needed to complete the task at the workstation **node.js**.

To build an application after changes (*from item 1 or 2*) the command:

npm run build

should be executed from the command line in the catalog **ClientApp.**

The production version is created in the HBS project's **'wwwroot\dist'** directory

(*this is the destination before the application is published*).

appsettings.json

The "to_be_completed" strings must be replaced with the appropriate strings as per the installation instructions.

```
{
  "ConnectionStrings": {
    "DefaultConnection": "to_be_completed"
  },
  "Authentication": {
    "JwtKey": "bm6o7g3b1hv2ntvva32yk7rzc3k9wfg89f5jkcxtpr
    "JwtIssuer": "to_be_completed",
    "JwtAudience": "to_be_completed",
    "JwtExpireMins": 10
  },
  "UrlSettings": {
    "ApplicationBaseUrl": "to_be_completed"
  },
```

Example for DefaultConnection

```
"DefaultConnection":
"Data Source=server_name_or_ip;Initial Catalog=hbs;Integrated Security=SSPI;"
```

The string server_name_or_ip change to appropriate string or IP mask 999.999.999.999

To change the application name and its description set **Title** and **Description**

```
  "Properties": {
    "Title": "HBS",
    "Description": "Household Budget Survey Online Application"
```

index.js

Preparing another language version

Original	Translation

export default {
yes: "Tak",
no: "Nie",
search: "Szukaj",
actions: "Zadania",
close: "Zamknij",
reset: "Zeruj",
wait: "Proszę czekać ...",
copy: "kopiuj",
copied: "skopiowano",
searchLenght: "Przynajmniej 3 znaki",
respondent: {
save: "Zapisz formularz",
inactiveTitle: "Badanie niedostępne",
inactiveDescription: "Edycja nie aktywna, tymczasowy problem, lub inna istotna przyczyna " },
	export default {
	yes: "Yes",
	no: "No",
	search: "Search",
	actions: "Actions",
	close: "Close",
	reset: "Reset",
	wait: "Please stand by...",
	copy: "copy",
	copied: "copied",
	searchLenght: "At least 3 characters",
	respondent: {
	save: "Save form",
	inactiveTitle: "Survey is not available",
	inactiveDescription: "Edition is not active, there are temporary technical problems, or another important reason" },

6.2 Evaluation version

The evaluation version allows to familiarize with the full functionality of the system.

The version is intended for demonstration of operation, not for production use.

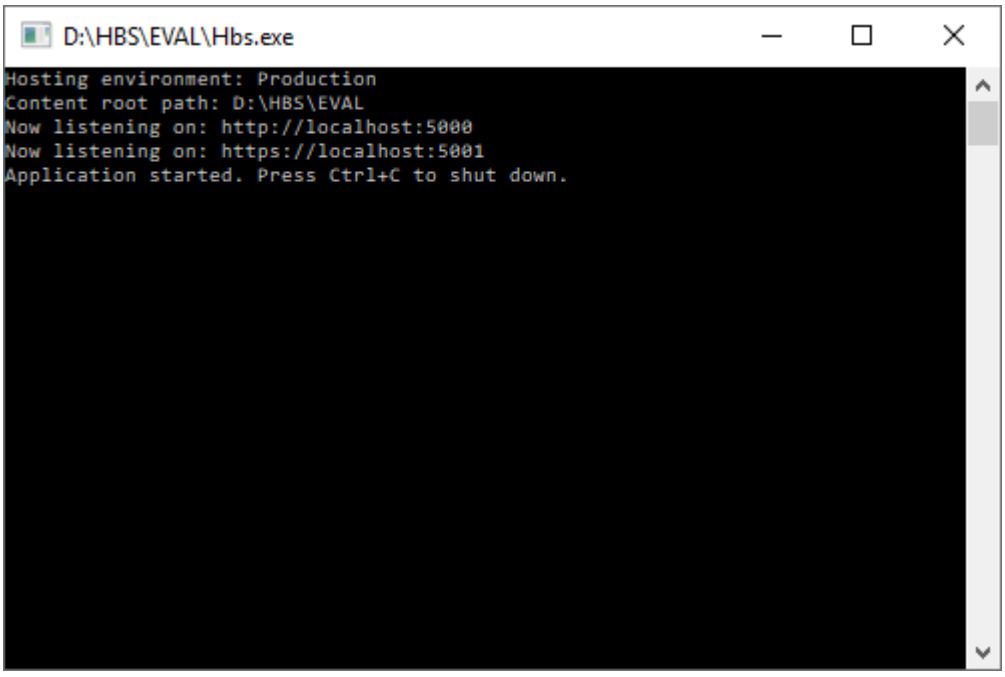
At the workstation, the evaluation version requires the following components:

1. NET Core 3.1 runtime environment
   (installation file https://dotnet.microsoft.com/download/dotnet-core/thank-you/runtime-aspnetcore-3.1.3-windows-hosting-bundle-installer)
2. local database: Microsoft SQL Server Express LocalDB
   (depending on the operating system
   64-bit version https://download.microsoft.com/download/8/D/D/8DD7BDBA-CEF7-4D8E-8C16-D9F69527F909/ENU/x64/SqlLocalDB.MSI
   or 32-bit version
   https://download.microsoft.com/download/8/D/D/8DD7BDBA-CEF7-4D8E-8C16-D9F69527F909/ENU/x86/SqlLocaLDB.MSI)

File **hbs_eval.zip** with the evaluation version should be unpacked to any folder on the workstation.

To start the application, run the **hbs.exe** program from the selected folder.

The program starts in the terminal window in text mode.



Terminal window

In the evaluation environment configuration, messages about subsequent processes and settings are displayed.

The last stage displays the addresses at which the application is available.

In the example it is a link: http://localhost:5000

At the workstation, launch a web browser and enter the appropriate link in the address field. An evaluation prototype of the application in the evaluation version appears in the browser window.

To access log on using the login : „**designer**" and the password: „**P@ssw0rd**"

The evaluation application is available in the browser until the terminal window is closed (shown in the figure).

6.3 List of installation files

DB_CREATE.zip Zip file with the script for creating data database

hbs_eval.zip Zip file with the evaluation version

hbs_prod.zip Zip file with the production version

HBS_SOLUTION.zip Zip file with the project of prototype with source codes

The system in the initial version contains an example diary form
that base on the diary used in the United Kingdom
(*selected because of language and variety of fields used in the diary*).