

SURVEYDAY

VALERIO ZANIBELLATO, ADOMAS KOCIUS, LUCIAN L. GOMBOS,
RAIMONDS NIKMANIS

<http://surveyday.azurewebsites.net>

Admin credentials:

- Username = admin, Password = admin

User credentials:

- Username = one, - Password = one

Contents

Project description.....	2
Methodology.....	2
• Agile method.....	2
• Visual studio Online	3
Slack (Communication tool).....	3
Interface design and digital aesthetics	5
• Wireframes	5
• Multiplatform consideration.....	6
Consistent	7
Continuous.....	7
Complementary	7
Database	7
• SQL vs MongoDB.....	7
• ER diagram	8
• Stored Procedures and views.....	9
Frontend.....	12
• Composition and colour scheme	12
• Application of Gestalt law	12
• Responsive Web Design (Bootstrap).....	14
• Implementation of interactivity with JavaScript.....	18
Backend.....	20
UML diagrams	20
• Validation rules	21
• Using the entity framework and workflow	22
Bibliography	25

Project description

In this project our team will be developing a prototype web application that will be used by a group of researchers at the University Hospital of Aarhus. This prototype will allow them to record patient symptoms related to the study of asthma and allergy.

Our product will be able to generate online surveys orchestrated by researchers and also used by patients. The patients will go to the website, log in with a username and a password provided by the admins and start a survey. After completion the admins will be able to see the results on daily basis.

Due to the abundant use of mobile devices, our application shall be optimized for both desktop and mobile use.

Methodology

- Agile method

For this project we will be using the agile method approach in order to successfully complete our work. We decided to use this method because the Agile approach is often utilized in software development.

Another reason for using this method is the use of sprints. Sprints come very handy when handling a project within a short time span.

We use sprints to complete small particular phases of our project. Usually allocating two to three days per sprint. Sprints are considered to be complete when the time period expires. There may be disagreements among the members of the team as to whether or not the development is satisfactory, however, we need to keep in mind the short period of time that is allocated for the entire project.

Our team focuses on the following agile principles:

- Working software is the primary measurement of progress.
- Face-to-face communication is the best way to transfer information to and within a team.
- A self-organised team usually creates the best designs.
- At regular intervals, the team will reflect on how to become more effective, and they will tune and adjust their behaviour accordingly

In addition to the agile method, we also used the scrum method which will suit as a framework for managing our project. Using scrum in our team means that there is no overall team leader who decides which person will do which task, or how a problem will be solved. Instead, the issues will be decided by the team as a whole.

- Visual studio Online

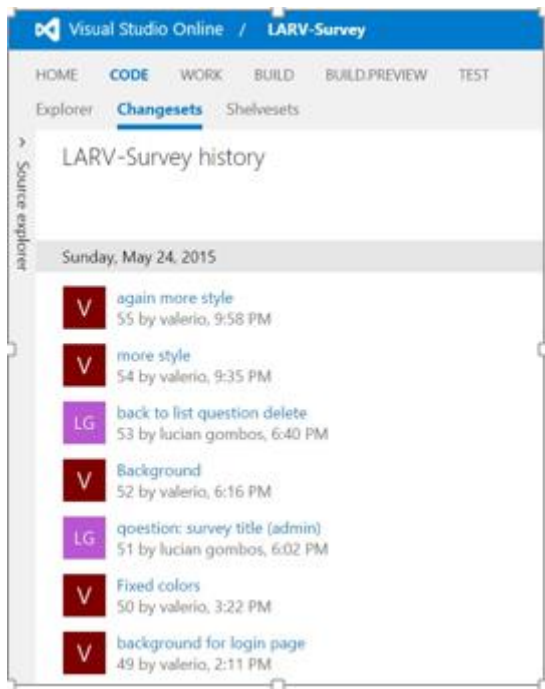


Fig.1. Visual Studio Online

In order to maximize the productivity of our team, we will be using the Visual Studio Online and the cloud based service Azure. Here we can work together at the same project in real time. We have to admit that this has been a new experience for all of us and that we have encountered some difficulties setting up the start of the project. However, there are still many benefits for using this service.

For example VS Online helps with the traceability and visibility where all our code changes are linked directly to the story, bug, or task driving the work. A feature that we will use to aid the agile approach.

Slack (Communication tool)

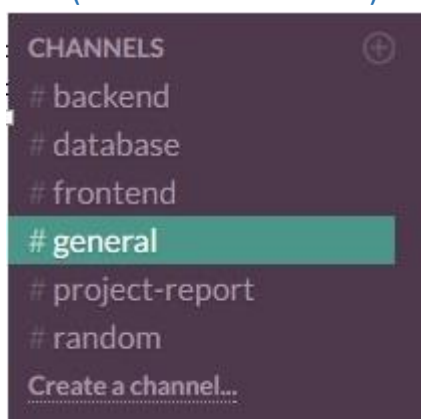


Fig. 2. The Slack communication tool

Slack is a team communication tool that our team has been using to communicate, share and organize different parts of the project.

Slack communications happen all in one place and can be segmented by creating Channels for various topics.

It is a great way to keep the team updated during all the stages of the project.

The advantage of this communication tool is that there are no distractions, compared to other social media messaging systems.

Interface design and digital aesthetics

- Wireframes

Before getting started with the wireframe, we needed as much information as possible in regards to our project. In our case we followed the business requirements, such as the requirements needed by the researchers to gather patient results. As web developers we will use the wireframe to understand the expected functions and behaviours of the design.

Based on this knowledge, we first created a simple site map that will be later used to shape our wireframe.

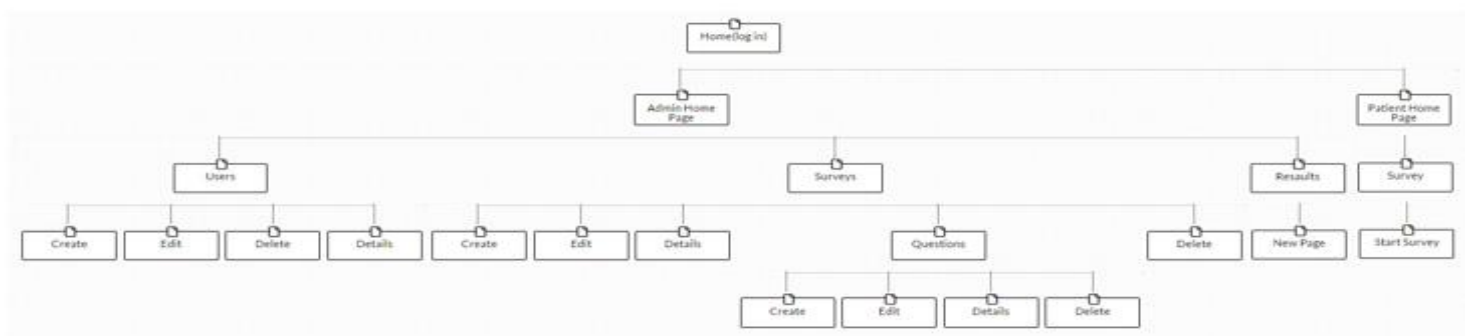


Fig. 3. Sitemap

The sitemap hierarchy consists of the login Home page with which the administrator and patient (user) gain access. The administrator should be able to create, edit, delete and view data throughout the administration page. On the patient side the user should be able to complete a survey.

It is needed to be mentioned that an early stage database model was created before we started sketching the wireframe, therefore we had a pretty solid idea of what to expect from the wireframe.

The team created low fidelity wireframes based on what logic we figured out is needed to fulfil the client requirements. We chose a low fidelity because it leaves room for improvement. During the product development we might encounter new challenges and it might result. Below we have the wireframes for the administrator.



Fig. 4. Wireframes for the administrator and user

We chose to have only simple and most required elements, this way it will ease our struggle to decide which content is not necessary for the mobile and tablet devices. (Unger & Chandler, 2012)

- **Multiplatform consideration**

The rapid growing number of connected devices, especially mobile ones, is progressively changing the way people perceive, experience, and interact with products and each other.

These experiences should focus on how the set of connected devices can best serve users' needs as they move between activities and contexts throughout the day. A set of connected devices are also referred to as an ecosystem. Such as the biologists refer to the ecosystem as the interconnection of the natural world, so do developers think of the ecosystem as a set of connected devices.

We have adopted a framework that is durable and relevant when approaching the ecosystem. This framework goes by the name of 3 C's and it stands for: Consistent, Continuous and Complementary.

Unfortunately, we could not fully benefit of this framework. Our prototype web application lacks the sophistication to truly make use of the Continuous and Complementary hallmark.

Consistent

A consistent design allows the same basic experience to be replicated between devices, keeping the content, flow, structure, and core feature set consistent across the ecosystem. In our case the admins (researchers) and the lower level users (patients) will benefit of the same experience by using different devices. Admins can advantageously use a desktop device to create users, create surveys and view results and do the same task using the mobile or the tablet device. And the same goes for the users completing a survey.

Some adjustments are made to accommodate device-specific attributes (mainly screen size and interaction model), but overall the experience can be fully consumed, in an independent manner, on any device.

Continuous

Continuous means that the experience should be passed on from one device to another, continuing the same activity or progressing through a sequence of different activities, all taking place in different contexts but all channelled toward achieving the same end goal. For our project using the continuous design approach, we make use of the single activity flow.

Although the application should be able to be fully used on all devices, the admins might find it easier to structure a complete survey on a larger screen such as a desktop computer or a tablet. He should be able to resume his activity

Complementary

Complementary design is that devices complement one another, creating a new experience as a connected group. This experience can fold two forms of device relationship: collaboration and control.

Having the administrator side optimized for multiplatform is an important task for us. The researcher should be able to check the results at any time on his mobile.

(Levin, 2014)

Database

● SQL vs MongoDB

MongoDB is an open source software that falls under the "Document" category. MongoDB helps develop applications faster because it uses tables therefor stored procedures are no longer required .It offers advantage to developers because the object model and the stored data have the same structure, similar to the JSON format, called BSON

The advantages of using MongoDB is that it enables horizontal scalability by using a technique called sharding. This technique distributes the data across physical partitions to overcome the hardware limitations. It can also be said that MongoDB is cost effective because improves flexibility and reduces cost on hardware and storage.

SQLis portable and can run in programs in mainframes, PCs, laptops, servers and even mobile phones. It runs in local systems, intranet and internet. Databases using SQL can be moved from device to another without any problems.SQL mainly consists of English statements and it is very easy to learn and understand a SQL query.

SQL can communicate with the databases and get answers to complex questions in seconds and it also supports the latest object based programming and is highly flexible.

With advantages on both sides, our team has settled on using SQL for a simple reason, most of us are more acquainted with using SQL then MongoDB. For all the team mates to experience the same understanding of the database used for our product, SQL was a fast and easy choice.

- ER diagram

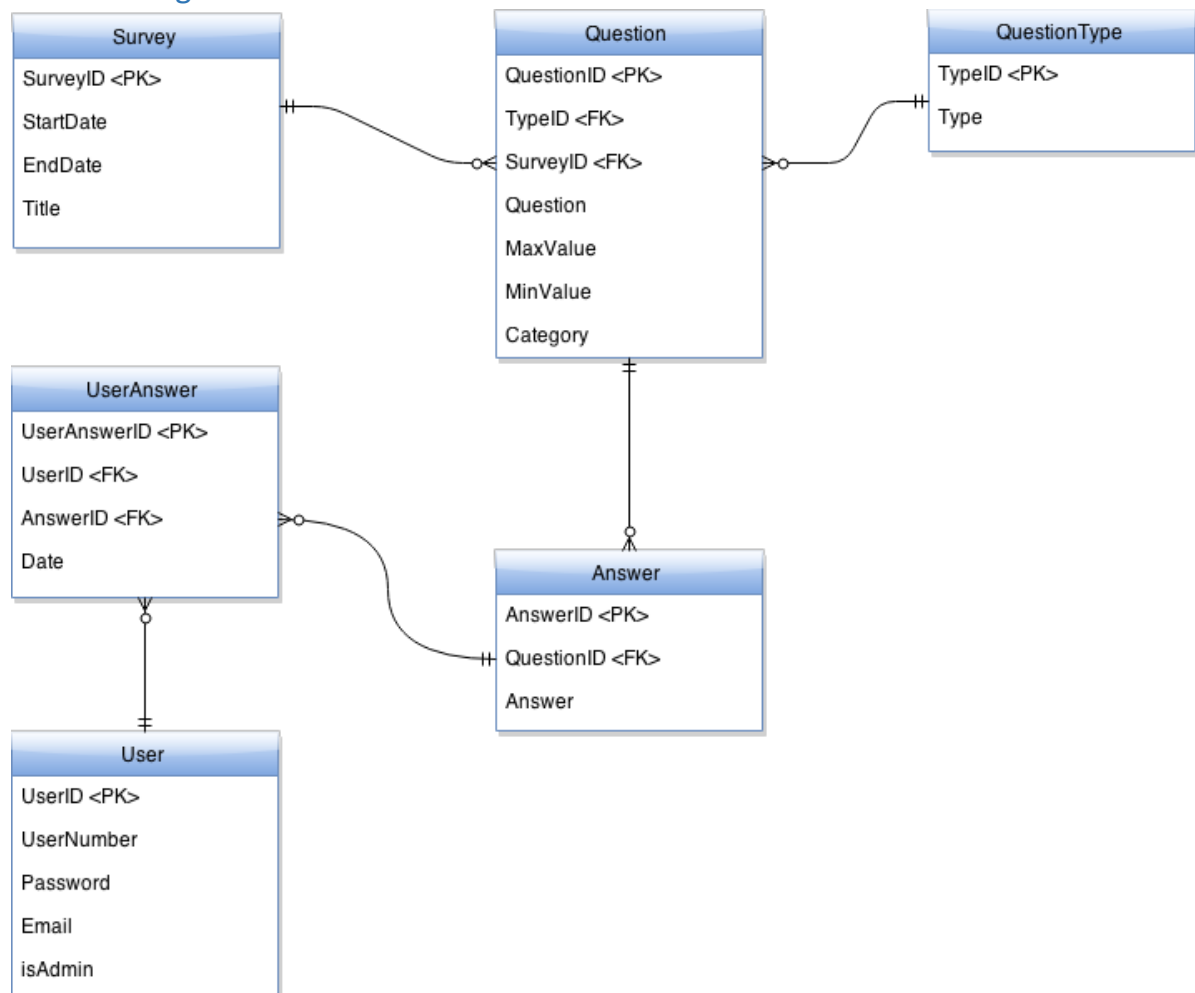


Fig. 5. ER diagram

- Stored Procedures and views

Below we have two views and two stored procedures that we think can be found useful.

It might come in handy to have this view and have a quick check on the what surveys were submitted yesterday

View 1

```
/*This view returns the users that have submitted the surveys yesterday*/
CREATE VIEW YesterdayUsersSurveys
AS
SELECT u.UserName, s.Title, ua.SubmitDate
FROM [User] as u
JOIN UserAnswer as ua on ua.UserID = u.UserID
JOIN Answer as a on a.AnswerID = ua.AnswerID AND ua.SubmitDate = CONVERT (date, GETDATE()-1)
JOIN Question as q on q.QuestionID = a.QuestionID
Join Survey as s on s.SurveyID = q.SurveyID
GO
```

Results for view 1

	UserName	Title	SubmitDate
1	one	Standard Survey 2	2015-05-27
2	one	Standard Survey 2	2015-05-27
3	one	Standard Survey 2	2015-05-27
4	one	Standard Survey 2	2015-05-27
5	one	Standard Survey 2	2015-05-27
6	one	Standard Survey 2	2015-05-27
7	one	Standard Survey 2	2015-05-27
8	one	Standard Survey 2	2015-05-27
9	one	Standard Survey 2	2015-05-27
10	two	Standard Survey 2	2015-05-27
11	two	Standard Survey 2	2015-05-27
12	two	Standard Survey 2	2015-05-27
13	two	Standard Survey 2	2015-05-27
14	two	Standard Survey 2	2015-05-27
15	two	Standard Survey 2	2015-05-27
16	two	Standard Survey 2	2015-05-27

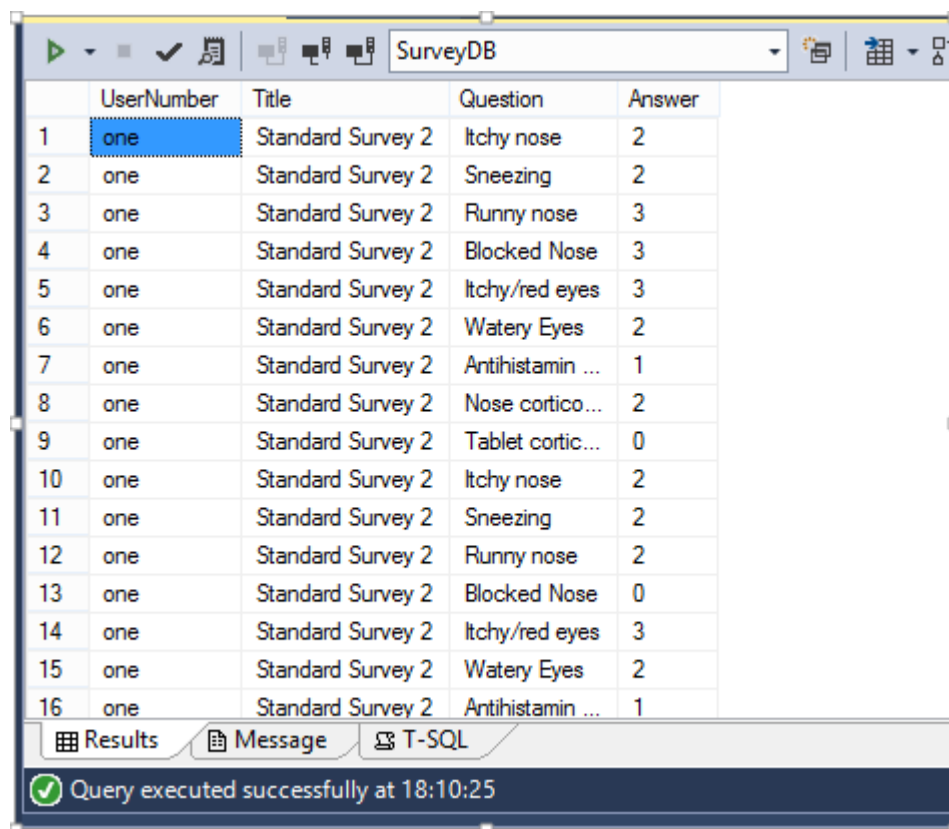
Fig. 7. Results from view 1

Or do a quick check on the answer from every user that submitted today and for a specific survey.

View 2

```
/*This view displays the answers every user that have submitted the example surveys today*/  
CREATE VIEW TodayStandarSurvey  
AS  
SELECT u.UserNumber,s.Title,q.Question, a.Answer  
FROM Answer as a  
JOIN UserAnswer as ua on a.AnswerID = ua.AnswerID AND ua.SubmitDate = CONVERT (date, GETDATE())  
JOIN [User] as u on u.UserID = ua.UserID  
JOIN Question as q on q.QuestionID = a.QuestionID  
JOIN Survey as s on s.SurveyID = q.SurveyID AND s.Title = 'Standard Survey 2'  
GO
```

Fig. 8. View 2



	UserNumber	Title	Question	Answer
1	one	Standard Survey 2	Itchy nose	2
2	one	Standard Survey 2	Sneezing	2
3	one	Standard Survey 2	Runny nose	3
4	one	Standard Survey 2	Blocked Nose	3
5	one	Standard Survey 2	Itchy/red eyes	3
6	one	Standard Survey 2	Watery Eyes	2
7	one	Standard Survey 2	Antihistamin ...	1
8	one	Standard Survey 2	Nose cortico...	2
9	one	Standard Survey 2	Tablet cortic...	0
10	one	Standard Survey 2	Itchy nose	2
11	one	Standard Survey 2	Sneezing	2
12	one	Standard Survey 2	Runny nose	2
13	one	Standard Survey 2	Blocked Nose	0
14	one	Standard Survey 2	Itchy/red eyes	3
15	one	Standard Survey 2	Watery Eyes	2
16	one	Standard Survey 2	Antihistamin ...	1

Fig. 9. Results from view 2



```

New Query
SQLQuery1.sql - not connected
USE [master]
GO
CREATE DATABASE Survey_DB
GO

/*Store procedure, creating all the tables*/
USE Survey_DB
GO
CREATE PROCEDURE sp_Survey_DB_createTables
AS
    CREATE TABLE dbo.Survey(
        SurveyID INT PRIMARY KEY IDENTITY(1,1),
        Title varchar(100) NOT NULL,
        StartDate DATE,
        EndDate DATE
    )

    CREATE TABLE dbo.QuestionType(
        QuestionTypeID INT PRIMARY KEY IDENTITY(1,1),
        QuestionType varchar(50) NOT NULL
    )

    CREATE TABLE dbo.Question(
        QuestionID INT PRIMARY KEY IDENTITY(1,1),
        SurveyID INT FOREIGN KEY (SurveyID) REFERENCES dbo.Survey(SurveyID),
        QuestionTypeID INT FOREIGN KEY (QuestionTypeID) REFERENCES dbo.QuestionType(
        QuestionTypeID),
        Question varchar(100) NOT NULL,
        Category varchar(50),
        MaxValue INT,
        MinValue INT
    )

    CREATE TABLE dbo.Answer(
        AnswerID INT PRIMARY KEY IDENTITY(1,1),
        QuestionID INT FOREIGN KEY (QuestionID) REFERENCES dbo.Question(QuestionID),
        Answer varchar(50) NOT NULL
    )

    CREATE TABLE dbo.[User](
        UserID INT PRIMARY KEY IDENTITY(1,1),
        UserNumber varchar(50) NOT NULL,
        Email varchar(50),
        [Password] varchar(50),
        IsAdmin BIT
    )

    CREATE TABLE dbo.UserAnswer(
        UserAnswerID INT PRIMARY KEY IDENTITY(1,1),
        AnswerID INT FOREIGN KEY (AnswerID) REFERENCES dbo.Answer(AnswerID),
        UserID INT FOREIGN KEY (UserID) REFERENCES dbo.[User](UserID),
        SubmitDate DATE NOT NULL,
    )
GO

```

Fig. 10. Store procedure to create all the tables

```

CREATE PROCEDURE sp_SurveyDB_EmptyTables
AS
    DELETE FROM UserAnswer
    DBCC CHECKIDENT ('[UserAnswer]', RESEED, 0)
    DELETE FROM [User];
    DBCC CHECKIDENT ('[User]', RESEED, 0)
    DELETE FROM Answer;
    DBCC CHECKIDENT ('[Answer]', RESEED, 0)
    DELETE FROM QuestionType
    DBCC CHECKIDENT ('[QuestionType]', RESEED, 0)
    DELETE FROM Survey;
    DBCC CHECKIDENT ('[Survey]', RESEED, 0)
    DELETE FROM Question;
    DBCC CHECKIDENT ('[Question]', RESEED, 0)
GO

```

Fig. 11. Store procedure for empty all the tables and reseed the primary keys

Frontend

- **Composition and colour scheme**

The colour is a really important part of the design. We had a discussion about the colour and what kind of colour scheme we should use. Unanimously we agreed that blue is the right colour for our project. We were thinking about the user experience and feelings when they are facing the symptoms of allergy. For example, if the user is facing dry eyes, swelling and gnaw eyes, it could be referenced with red colour. While filling the survey, they will think about the symptoms and those unpleasant feelings. By using the blue colour in the design we are trying to hiddenly make the users think about water, sea, and at some point ice, that would subside the unpleasant feelings caused by allergies.



Fig. 12. Adobe kuler wheel

To emphasize the content we added a darker colour for the main content and action buttons, but for the background we added the light blue colour. To avoid over colourful design we used 2 main blue colours and white to have nice contrast between them. To find the best match between the colours we used the adobe kuler wheel that is set to monochromatic colour type. (Adobe Color CC, 2015)

- **Application of Gestalt law**

Objects that are closer together are perceived as more related than objects that are further apart. By Gestalt laws that is proximity. In our design we use this law in menu bar and also in dropdown menu to show that these menu have common action. For menu it leads user to different pages, but in dropdown menu the actions are related for one row.

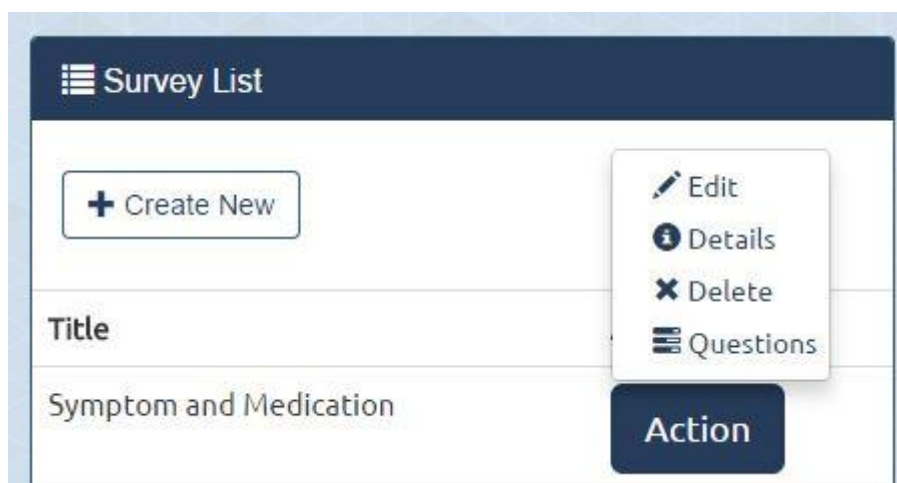
Another rule we used is parallelism. Elements that are parallel to each other are seen as more related than elements not parallel to each other. In our design it reflects in showing rows of surveys.

Survey List			
+ Create New			
Title	Start Date	End Date	Actions
Symptom and Medication	22 May 2015	22 Jun 2015	   
Standard Survey	29 Apr 2015	29 Jul 2015	   
Standard Survey 2	26 May 2015	26 Jun 2015	   
example	25 Mar 2015	30 Jun 2015	   
cool test			   

All of the surveys titles are placed parallel each to other, by showing that they are grouped and have same meaning - survey title.

Another law we used to show a connection between elements. We enclose them in group separating them with border. By Gestalt law it is called COMMON REGIONS

Everything inside the enclosure is seen as related. Everything outside the enclosure is seen as separate. The menu with border around shows that they are related by having action to specific survey.



- ## Responsive Web Design (Bootstrap)

In this chapter we will talk about what responsive web design is, and how and why we implemented it in our project. In addition to it, we will describe the possible improvements of our application for future implication.

Responsive web design and mobile first

The responsive web design is a technique for building websites that works on mobile devices, tablets, and desktop screens. This technique involves three principles: fluid grids, fluid images and media queries. Fluid grids are a simple way to define the size of an element using percentages instead of pixels. By using the fluid grids is possible to resize the content depending on the device in which it is displayed. In other words fluid grids just use relative units instead of fixed units but a web page will look the same in every kind of device. The same principle is made behind the fluid images; by using a simple line of CSS code is possible to adapt images to the screen size. In fact using this code `img{ max-width: 100%; }` and wrapping the images inside a div tag it is possible to adapt the images to the div size. The last principle is called media queries and it is the most important. Using fluid grids and fluid images is not enough to make the layout visible on every devices. With media queries it is possible to style the elements of web pages differently, depending on the device's screen resolution in which the website is displayed. There are two different approaches to use media queries: desktop first and mobile first. For both is required to use the viewport metatag `<meta name="viewport" content="width=device-width, initial-scale=1">`.

Example of media queries desktop first:

```
/*          ...desktop          styles          here...          */
@media screen and (max-width: 900px) { /* ...tablet styles here... */ }
@media screen and (max-width: 600px) { /* ...mobile styles here... */ }
```

Example of media queries mobile first:

```
/*          ...mobile          styles          here...          */
@media screen and (min-width: 600px) { /* ...tablet styles here... */ }
@media screen and (min-width: 900px) { /* ...desktop styles here... */ }
```

With both the approaches it is possible to produce a responsive website that can be adapted to different screen resolutions. The difference between these two approaches is the starting point for the layout. With the mobile first the start point is designing the smartphones experience as default style and adding the media queries to change the style for bigger screen resolutions. The mobile first approach is considered the best one due to the growing use of smartphones as devices to surf the web. Luke Wroblewski in his article [Mobile First](#) explains the reasons behind the mobile first approach. It is possible to resume his article like this:

- Mobile web browsers represent a rapidly growing demographic and will likely eclipse desktop browsing (if it hasn't happened already)
 - Small mobile screen sizes force designers to focus, because there's no room for sidebars, ads, social media buttons, and other peripheral content
 - Mobile devices typically have more capabilities than their desktop counterparts, such as touchscreens, GPS, accelerometer data, and more
- (Pettit, 2014)

Implementation of responsive web design and bootstrap

In order to implement our responsive web site we used bootstrap as our main framework. Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive websites with mobile first approach. By using the bootstrap's components it is possible to manage all the three principles of responsive web design very easily. We chose to use this framework because it is very complete and contains all that we needed to make our website responsive. In addition to the framework we override the bootstrap's classes to fit the style within our ideas.

We have used the bootstrap's grid system to style all the pages, combined with the viewport meta tag. We decided from the beginning to make a very simple layout, without useless elements that are not relevant for users in this kind of application. The challenge was to make the website responsive for the largest number of screen resolutions possible. The page that required more effort to be styled, was the page in which are displayed the questions for the users. The page was made by using a table element. Unfortunately, the tables are not really responsive for mobile devices and the bootstrap's class ".table-responsive" just add horizontal scroll for small screen size. We wanted more and our way to fix this issue was to create two different tables: one hidden for extra small and small devices, and the other one hidden for medium and large devices. In this way we had the possibility to style the elements inside the tables differently as shown in the image below.

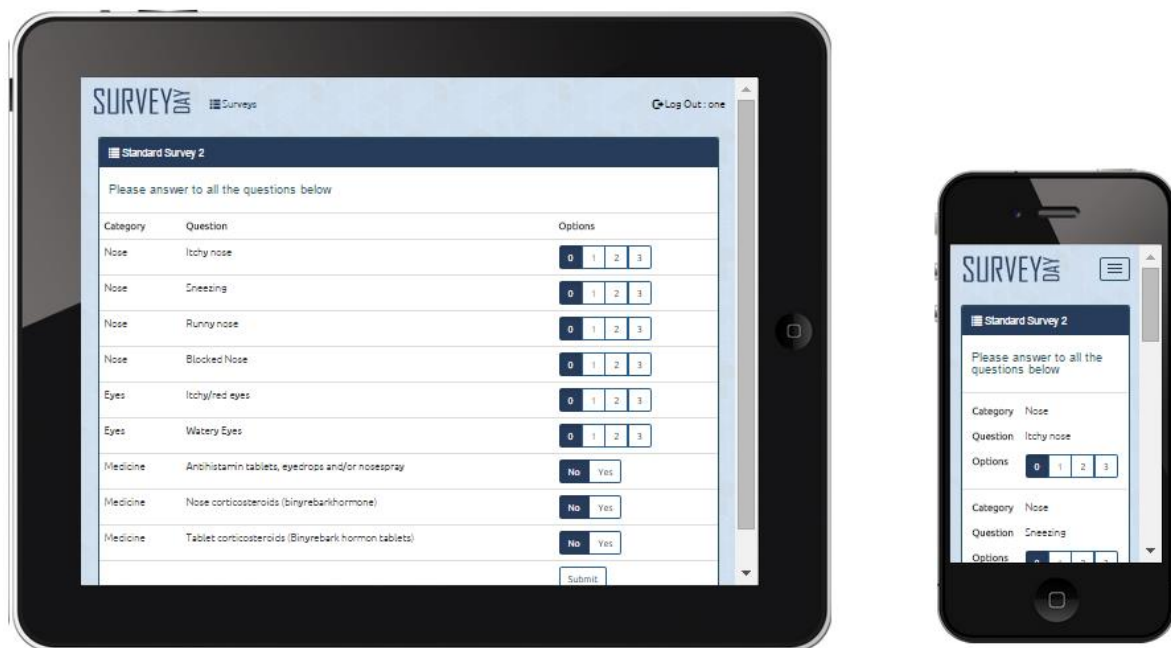
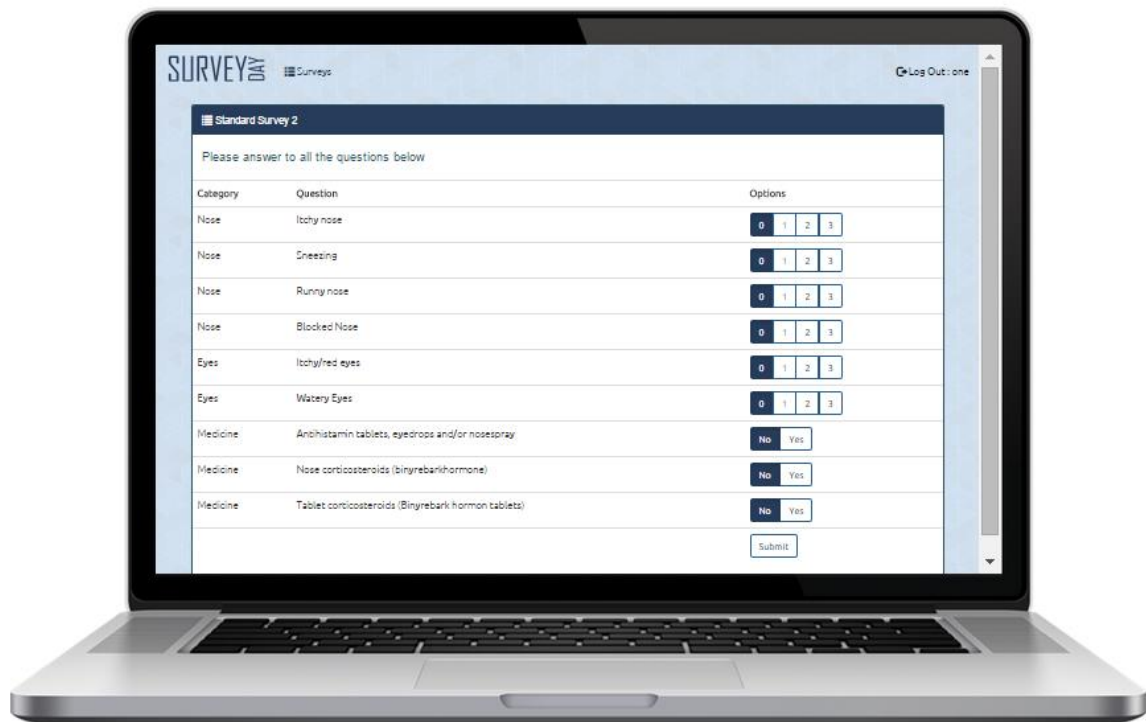


Fig. 13.

Another challenge was making the administrator's pages responsive as well.

We did it using a very simple and useful bootstrap utility such as the "drop-up buttons". With the drop-up buttons it was possible to group all the possible actions for the administrator inside one button that once is pressed shows the list of actions.

As shown in the image below, we implemented different solutions for mobile and desktop.

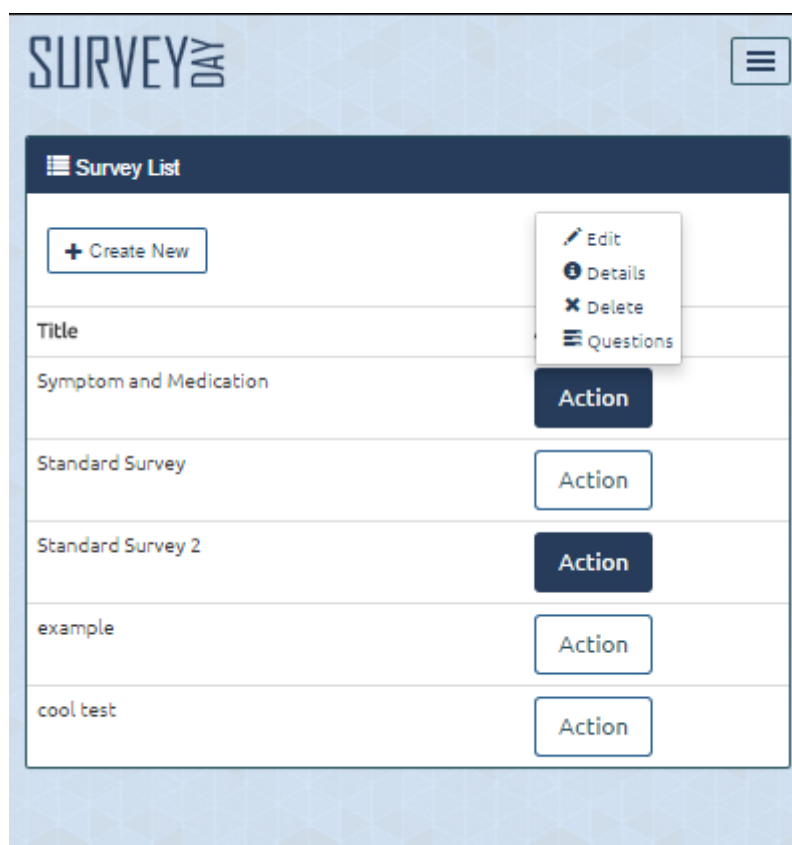
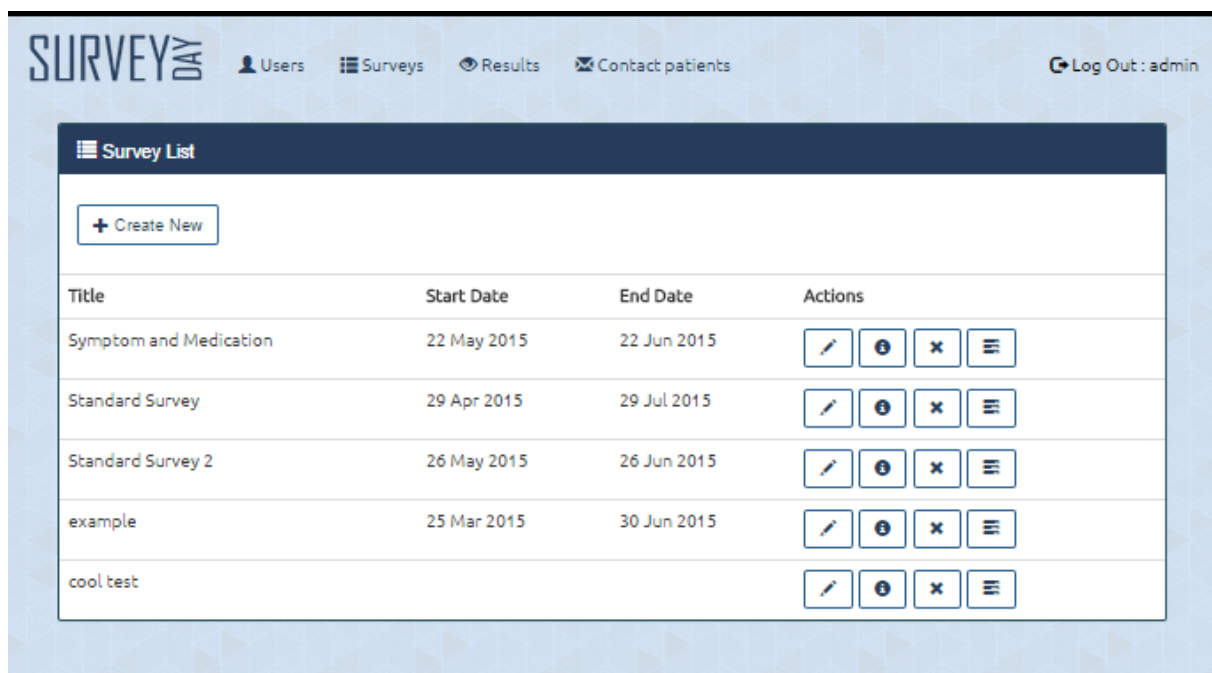


Fig. 14.

Possible improvement for the solution

Despite the fact we found a solution to make tables responsive, we didn't think about making it with standard div using the bootstrap's grid system. This could be a possible improvement of our design and probably the best solution to this problem.

- **Implementation of interactivity with JavaScript**

JavaScript and jQuery

JavaScript is known to be the language of the web. With JavaScript it is possible to add more interactivity to the pages and animate the elements in order to improve the user experience. Based on our idea of having a simple design we did not abuse of this powerful tool and, as with the CSS, we used the bootstrap built in JavaScript utilities to improve the interactivity of the design.

We also used jQuery which is a very feature-rich JavaScript library dedicated to animation, manipulation and event handling of DOM elements.

Bootstrap utilities

At the beginning inside the page in which we display the questions for a specific survey we had simple radio buttons to submit options. The radio buttons are very easy to use but they look really old and are not the best responsive choice. We decided to replace these elements with an interesting JavaScript utility made by bootstrap. With this utility it is possible to use "normal" nice looking buttons as radio buttons. The purpose of using this component was the possibility to toggle the selected active button inside the same group.

Another important utility we used from bootstrap is the tooltips for buttons. A tooltip is a small text that is displayed beside the button when the mouse is pointed on it. We decided to use tooltips because our interface does not have texts for the same buttons and the icon we used inside them could be misunderstood by users.

jQuery script

Inside the login page we used a simple jQuery function to hide and display a div that contains some information. The link to the script file is placed at the end of the page before the body's end tag. Inside the script we used an IIFE which is the acronym for *Immediately Invoked Function Expression*. This function is executed as soon as it is defined, meaning that once our page is loaded our jQuery code is running automatically.

Inside the same script we initialized the bootstrap's tooltips.

JavaScript library Chart.js for the results

Despite the fact we have a strong knowledge of JavaScript, we did not have enough time to implement a script for displaying the results with an animated chart. That is why we chose to use a ready to use library and after surfing the web for a while our choice was Chart.js.

Chart.js is a complete library that can be used to create a large number of chart types. Here are the main characteristics of this library:

- **HTML5 based**, it uses canvas elements to create charts.

- **Simple and flexible**, the script minified file is lightweight.
- **Responsive**, it resizes the charts depending on devices screen.
- **Modular**, it is possible to add only the modules needed.
- **Interactive**, it provides default data tooltips.

We used the standard bar chart type to display the results, in the image below it is possible to see the result page.

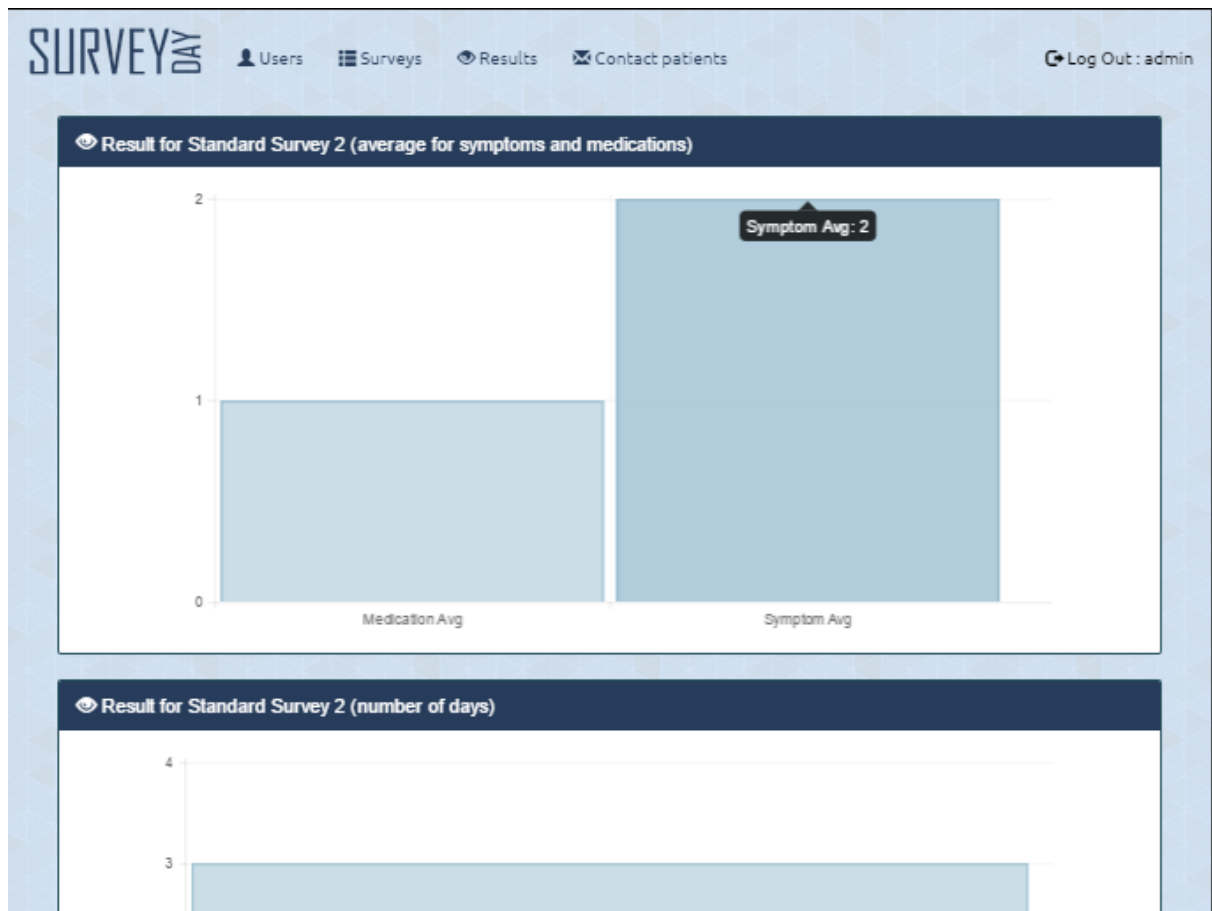


Fig. 15.

Backend

UML diagrams

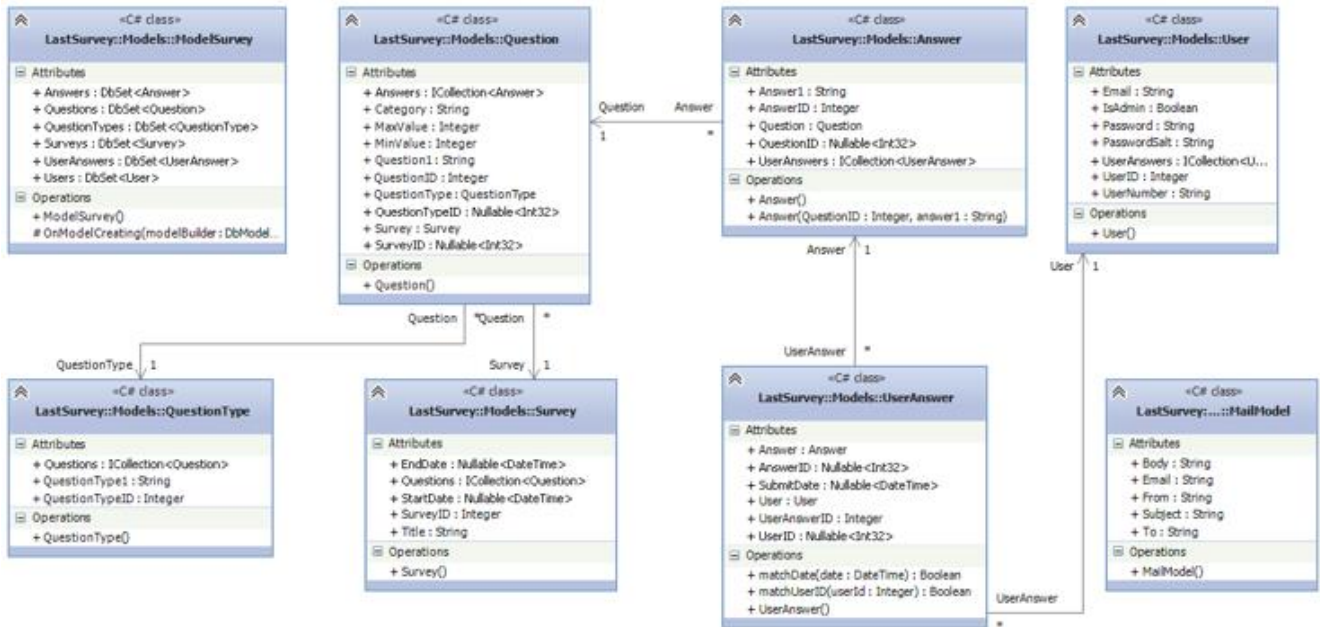


Fig. 16.

The Survey class has an ICollection of questions from the Question class, so it contains all of the questions related to the survey.

In the Question class we have a collection of answers related to the question. In our solution a question can have three different types (yes/no, point and open question) for which we have another class - QuestionType. Furthermore, we have a class Answers which is related to the Question. The Answer class is storing the answers but we need to trace which users are related to the answer, so we have a class UserAnswer. The UserAnswer class is linked to the Answer and User classes. Finally, the User class stores all of the users. We have an IsAdmin attribute with a boolean data type, since we only have two possible user roles.

The MailModel class contains attributes relevant to sending an email, it has no relationship with the other models in our solution.

Finally, we have the class ModelSurvey, which derives DbContext class. The DbContext class represents the set of tables in the database. We use it to map all the entities to the database tables.

- Validation rules

Almost all of the models were created using the code first with existing database approach, therefore, most validation annotations were already created. We did however add few Required fields where thought necessary. Error messages were handled by the views.

```
[Column("Question")]
[Required]
[StringLength(100)]
[Display(Name = "Question")]
public string Question1 { get; set; }
[Required]
[StringLength(50)]
public string Category { get; set; }
```

```
public int UserID { get; set; }

[Required]
[StringLength(50)]
[Display(Name="User Name")]
public string UserNumber { get; set; }

[Required]
[StringLength(50)]
[DataType(DataType.Password)]
[Display(Name="Password")]
public string Password { get; set; }

[StringLength(50)]
public string Email { get; set; }

public bool IsAdmin { get; set; }

[StringLength(50)]
public string PasswordSalt { get; set; }

public virtual ICollection<UserAnswer> UserAnswers { get; set; }
```

- Using the entity framework and workflow

Since day one, our group agreed that we are facing a tight deadline for the project. We considered different approaches to the problem but in the end we stuck with ADO.net Entity Framework. The framework allowed us to work at a higher level of abstraction and not be too concerned about the database. Instead, we mostly focused on the model.

We used the Entity Framework code first development with an existing database, which we created before. This approach saved us some time since we did not have to write any of the classes ourselves. The reverse engineered POCO classes seemed flexible and simple to understand. In case we would need to modify the classes, we could use Code First Migrations to sync the database. By using the code first with existing database approach, we had our clean classes and did not have to deal with .edmx model file and all of the associated auto generated code.

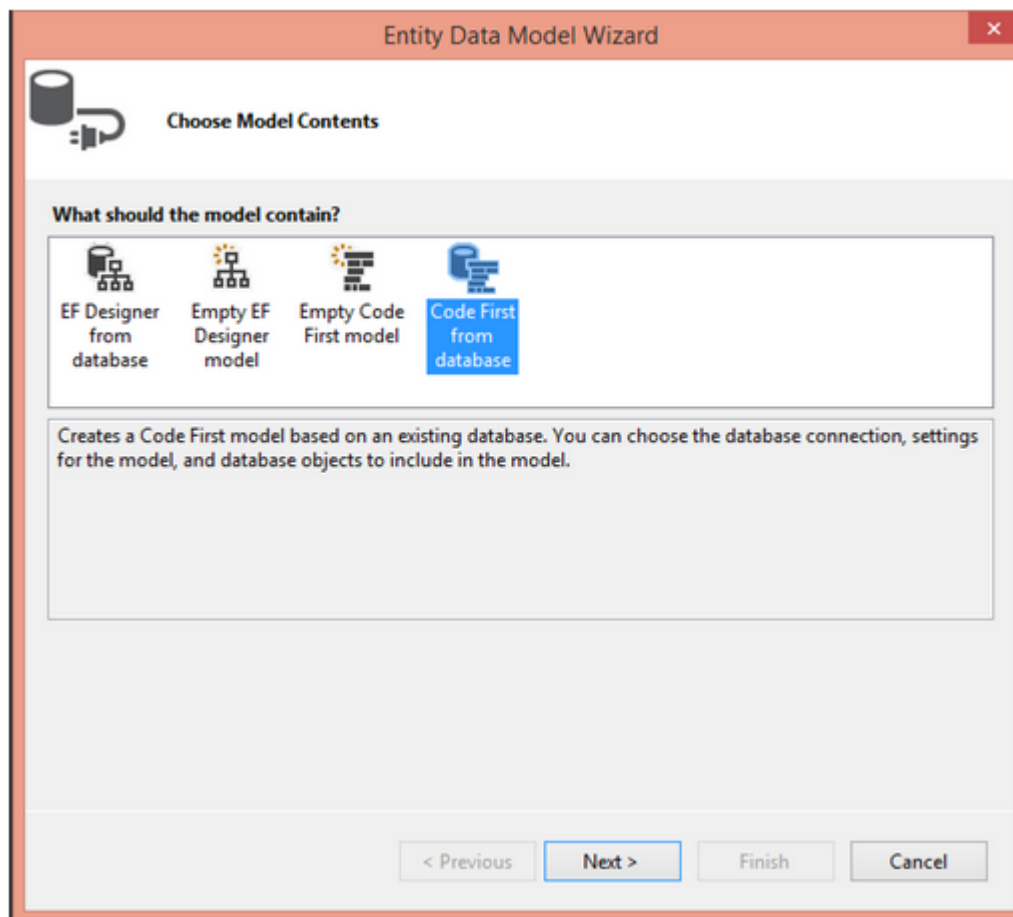


Fig. 17. Adding entity data model and choosing Code First from database option

Connection Properties

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source:
Microsoft SQL Server (SqlClient) Change...

Server name:
c5t1xnqft6.database.windows.net Refresh

Log on to the server

☐ Use Windows Authentication

☒ Use SQL Server Authentication

User name:

Password:

☐ Save my password

Connect to a database

☒ Select or enter a database name:

☐ Attach a database file:
 Browse...

Logical name:

Advanced...

Test Connection OK Cancel

Fig.18. Connecting to our existing database

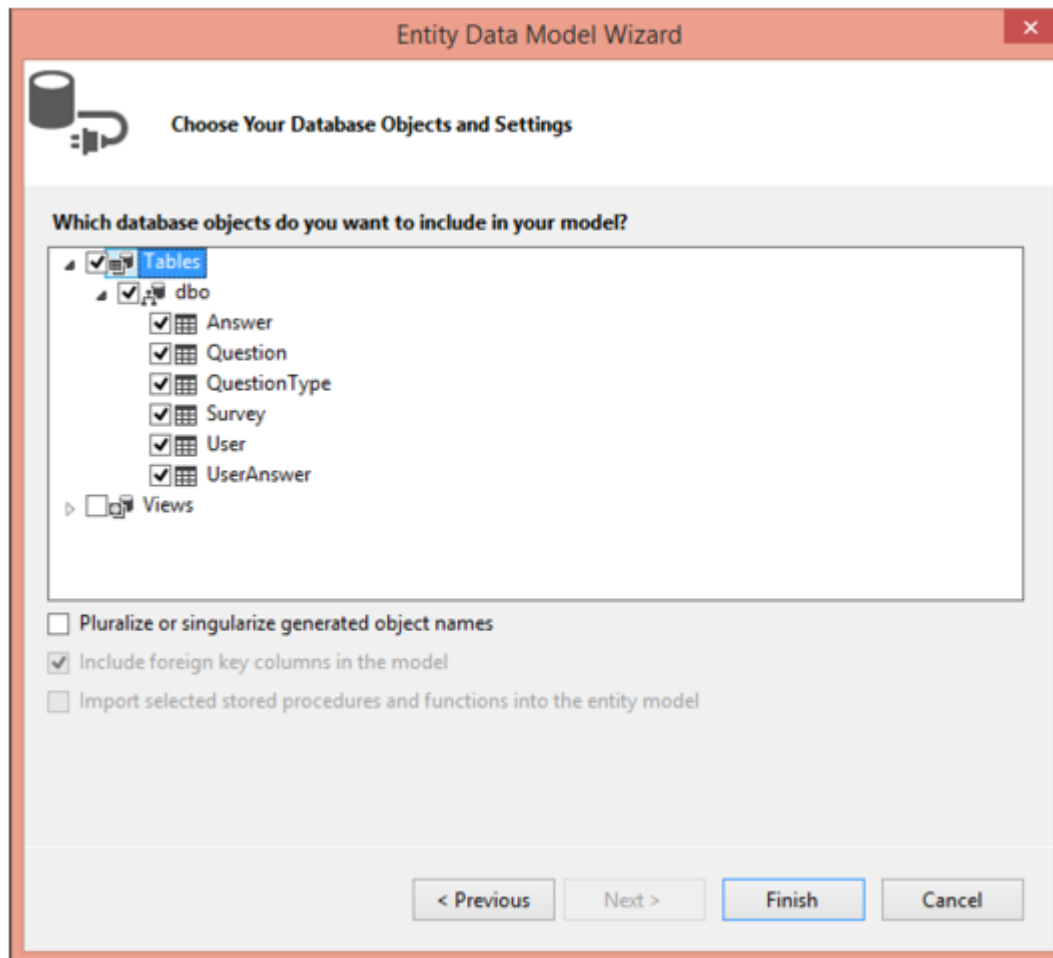


Fig. 19. Adding the database objects to reverse engineer the classes

Bibliography

Adobe Color CC. (2015). *My Color Theme*. Retrieved May 25, 2015, from Adobe Color CC: <https://color.adobe.com/create/color-wheel/>

Levin, M. (2014). *Designing Multi-Device Experiences*. O'Reilly.

Pettit, N. (2014, June 2). *The 2014 Guide to Responsive Web Design*. Retrieved May 24, 2015, from Treehouse blog: <http://blog.teamtreehouse.com/modern-field-guide-responsive-web-design>

Unger, R., & Chandler, C. (2012). *A Project Guide to UX Design, Second Edition*.