A Design Application for physician-patient communication

Lazar Slavković-Raco

Bachelor's thesis
2020

Tomas Bata University in Zlín
Faculty of Applied Informatics

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lazar Slavković-Raco**
Osobní číslo:	**A16446**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Softwarové inženýrství**
Forma studia:	**Prezenční**
Téma práce:	**Návrh webové aplikace pro komunikace mezi lékařem a pacientem**
Téma práce anglicky:	**A Design Application for Physician-Patient Communications**

Zásady pro vypracování

1. Proveďte rešerši aplikací zaměřených na komunikaci „lékař-pacient".
2. Analyzujte požadavky na aplikaci.
3. Vypracujte stručný rozbor technologií, které budou použity k návrhu.
4. Realizujte navrženou aplikaci ve zvolené technologii.
5. Věnujte pozornost zabezpečení aplikace.

Rozsah bakalářské práce:
Rozsah příloh:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. GALLOWAY, Jon, Brad WILSON, K. Scott ALLEN a David MATSON. Professional ASP.NET MVC 5. Indianapolis, IN: Wrox, a Wiley brand, 2014. Wrox professional guides. ISBN 978-1118794753.
2. JOHNSON, Glenn. Programming in HTML5 with JavaScript and CSS3: training guide. Redmond, Wash.: Microsoft, 2013. ISBN 978-0735674387.
3. MUNRO, Jamie. ASP.NET MVC 5 with bootstrap and knockout.js: building dynamic, responsive web applications. Sebastopol: OReilly Media, 2015. ISBN 978-1491914397.
1. ANDERSON, Elijah. *Sql: The Ultimate Beginners Guide To Learn SQL In Less Than 24 Hours*. CreateSpace Independent Publishing Platform, 2016.
2. ASP.net MVC with entity framework and CSS. New York, NY: Springer Science+Business Media, 2016. ISBN 978-1484221365.

Vedoucí bakalářské práce: **doc. Ing. Petr Šilhavý, Ph.D.**
Ústav počítačových a komunikačních systémů

Datum zadání bakalářské práce: **28. listopadu 2019**
Termín odevzdání bakalářské práce: **15. května 2020**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky

L.S.

doc. Mgr. Milan Adámek, Ph.D.
děkan

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

I hereby declare that:

- I understand that by submitting my Diploma thesis, I agree to the publication of my work according to Law No. 111/1998, Coll., On Universities and on changes and amendments to other acts (e.g. the Universities Act), as amended by subsequent legislation, without regard to the results of the defence of the thesis.
- I understand that my Diploma Thesis will be stored electronically in the university information system and be made available for on-site inspection, and that a copy of the Diploma/Thesis will be stored in the Reference Library of the Faculty of Applied Informatics, Tomas Bata University in Zlin, and that a copy shall be deposited with my Supervisor.
- I am aware of the fact that my Diploma Thesis is fully covered by Act No. 121/2000 Coll. On Copyright, and Rights Related to Copyright, as amended by some other laws (e.g. the Copyright Act), as amended by subsequent legislation; and especially, by §35, Para. 3.
- I understand that, according to §60, Para. 1 of the Copyright Act, TBU in Zlin has the right to conclude licensing agreements relating to the use of scholastic work within the full extent of §12, Para. 4, of the Copyright Act.
- I understand that, according to §60, Para. 2, and Para. 3, of the Copyright Act, I may use my work - Diploma Thesis, or grant a license for its use, only if permitted by the licensing agreement concluded between myself and Tomas Bata University in Zlin with a view to the fact that Tomas Bata University in Zlín must be compensated for any reasonable contribution to covering such expenses/costs as invested by them in the creation of the thesis (up until the full actual amount) shall also be a subject of this licensing agreement.
- I understand that, should the elaboration of the Diploma Thesis include the use of software provided by Tomas Bata University in Zlin or other such entities strictly for study and research purposes (i.e. only for non-commercial use), the results of my Diploma Thesis cannot be used for commercial purposes.
- I understand that, if the output of my Diploma Thesis is any software product(s), this/these shall equally be considered as part of the thesis, as well as any source codes, or files from which the project is composed. Not submitting any part of this/these component(s) may be a reason for the non-defence of my thesis.

I herewith declare that:

- I have worked on my thesis alone and duly cited any literature I have used. In the case of the publication of the results of my thesis, I shall be listed as co-author.
- That the submitted version of the thesis and its electronic version uploaded to IS/STAG are both identical.

In Zlin; dated: 10.08.2020

Lazar Slavkovic-Raco v.r.
Student´s Signature

ABSTRAKT

Bakalářská práce se zabývá návrhem webové aplikace pro komunikaci mezi pacientem a lékařem. V první části této práce vysvětlíme, co je přesně ochrana dat, jaké data můžeme od uživatele získat a již existující řešení na českém trhu. V další části je rozebraná technologie použita na tuto práci při návrhu aplikace. Následně je zpracován rozbor a analýza požadavků na funkčnost aplikace, z kterého se rozvíjí aplikace a vypracování pro uživatele.

Klíčová slova: webová aplikace, webová technologie, analýza požadavků

ABSTRACT

The bachelor thesis deals with the design of a web application for communication between patient and doctor. In the first part of this work we will explain what exactly data protection is, what kind of data we can receive from user and existing solutions on the Czech market. In the next part, the analyzed technology is used for this work in the design of the application. Subsequently, the analysis and analysis of the requirements for the functionality of the application is processed, from which the application and elaboration for users are developed.

Keywords: web application, web technologies, requirement analysis

I want to thank to Veronika Vyvleckova for moral support and help with a grammar correction. I hereby declare that the print version of the Bachelor's thesis and the electronic version of the thesis deposited in the IS/STAG system are identical, worded as follows:

I hereby declare that the print version of my Bachelor's/Master's thesis and the electronic version of my thesis deposited in the IS/STAG system are identical.

CONTENTS

INTRODUCTION

The first web page in HTML started to exist somewhere at late 90's and since then web technology started to improve gradually overtime. In addition to web technology, a few years later came CSS (Cascade Sheet Style), which improved the visualization of HTML page with a range of styles and properties. JavaScript became standard alongside of HTML and CSS and it's job was to provide interaction to the web page for the user. Over the last 20 years HTML, CSS and JS (JavaScript) improved a lot. Nowadays, many frameworks exist exists for creating either simple web page or simple functional web application.

They are some medical clinics that don't use any type of app for creating appointments, instead they use the old fashioned way: calling phone number of the clinic where nurse picks up the call and writes it down to the appointment diary. Some people do not like to call by phone due to communication troubles or they have hard time organizing their free time where they could book an appointment.

This thesis will focus on design of such application and how should an application that enables patients to book an appointment with a physician look like, with simplistic design and functions.

I. THEORY

1 DATA PROCESSING

Before digital age, data were written on paper form, for example birth or death certificate, contracts, transactions, medical history, ownership's. When technology slowly integrated in daily lives, laws of data privacy started to emerge to give a person a control of his own data.

Council of Europe signed the agreement *Convention for the protection of individuals with regard to automatic processing of personal data* that was signed on 28th of January 1981 [1], then Czech Republic signed same agreement on 28th of January 2001, which went in to full effect since 1st of November 2001 [2]. Czech Republic had already an Act regarding protection of personal data, which was in full power on 1st of June 1992. Czech Republic had created an *Act on the protection of personal data in information systems* (256/1992 Sb.) that was active till 1st of June 2000 that was replaced with *Personal Data Protection Act* (101/2000 Sb.). PDPA was active till 24th of April 2019 due to existing data regulation from European Union (EU for short) that introduced GDPR (General Data Protection Regulation).

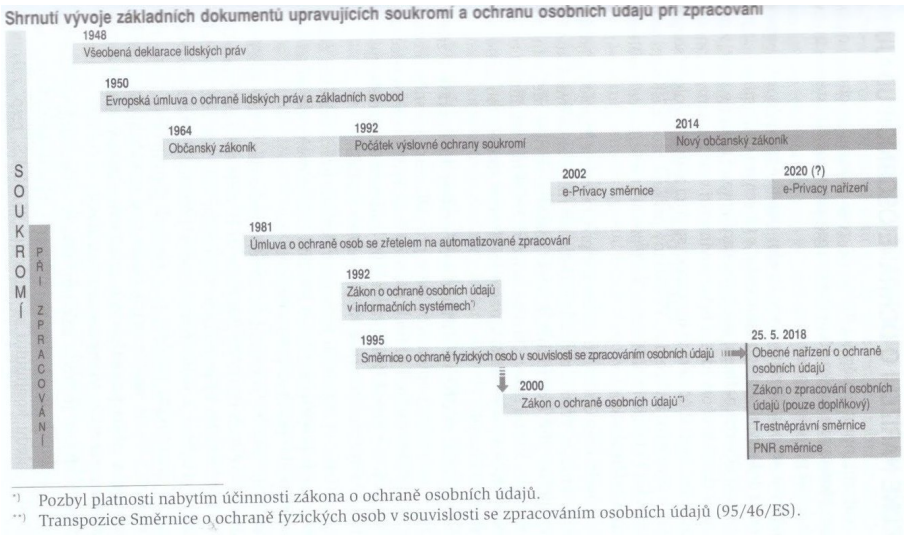


Figure 1: Graphs presenting history of development for user's right and privacy [1]

1.1 What is General Data Protection Regulation

General Data Protection Regulation (GDPR) is a regulation of data protection and privacy in the European Union(EU) and European Economics Area(EEA). This regulation also addresses when data is being collected outside of EU and EEA. Main goal

of GDPR is to give the user control of his data and increase his privacy.

User has a right:

- to edit
- to remove (to be forgotten)
- to limit data that is being processed

1.1.1 Type of user's data

GDPR can split those data into two categories [3]:

- personal data
  - e-mail
  - name (either as username or first and last name)
  - IP address
  - ...
- sensitive personal data (most companies tend to avoid to collect such information due of possible discrimination)
  - religion
  - race
  - sexual orientation
  - criminal past
  - ...

Personal data can be expanded beyond basic information about user:

- biological data
- genetic data
- location

1.2 Data leak scandals

There has been two major scandals when it comes to data leakage of a large amount of users, most famous one was Facebook – Cambridge Analytica in early 2018 when it was used for Trump's political campaign at 2016, data was used to target specific user to vote for him during American elections. [4]

Another occurred same year when Google rushed to shut down their product Google+ in mid 2018 when data was leaked on separate occasions, first one leaked 500,000+ users data but it was never proved that those data were misused and two months later over 50,000,000+ user's data were leaked again due to security bug that came with an update, data such as leaked e-mail, name's and corporation were leaked [5].

1.3 User's concerns about collection of data

There are some companies that collected bio-metrical data even before GDRP was introduced in 2018. Most known one is Fitbit. Fitbit was offering to remove data about user on their website and such deletion took of maximum 7 days.

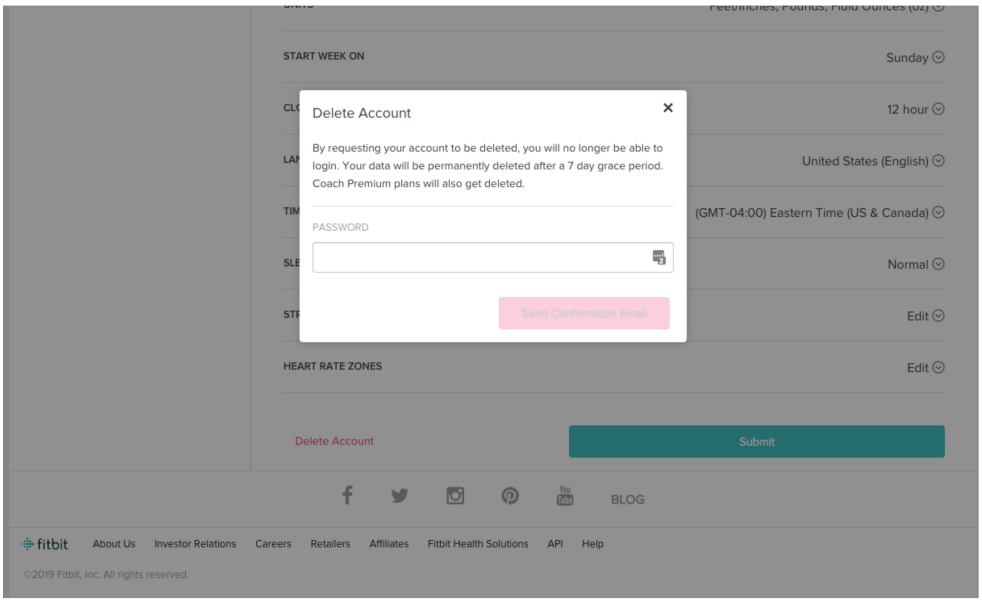


Figure 2: Fitbits notification of removing user's data

Users started to have concerns when Google announced that they are going to buy Fitbit somewhere at 2020. A lot of users were not happy about this announcement and most of them started to request their data to be deleted from Fitbit. Some of them even threw Fitbit tracker to the trash can. [7]

2 EXISTING SOLUTIONS ON THE MARKET

text

2.1 E-health Moravskoslezky kraj

There is only one web application that exists on Czech market for Moravskoslezky county. Besides appointment booking, it also offers to see medical history and transport arrangements to one of hospital's clinics. This application works for 6 hospitals that are in Moravskoslezky county.

2.1.1 Advantages

This app's interface is very simple to use, it also offers to book appointment to one of hospitals clinics without login information. This app was created by Ministry for Regional Development for Moravskoslezky county. It display's nicely on any mobile's browser.

2.1.2 Disadvantages

When user wants to register a new profile, a lot of information is required about the new user. It requires birth number, number of medical insurance and contacts information. Another author's biggest concern is possible security risk of the medical history of the user, however access to the user's medical history requires login with either information that user has to access to or with login info e-identita.cz or „datova schranka".

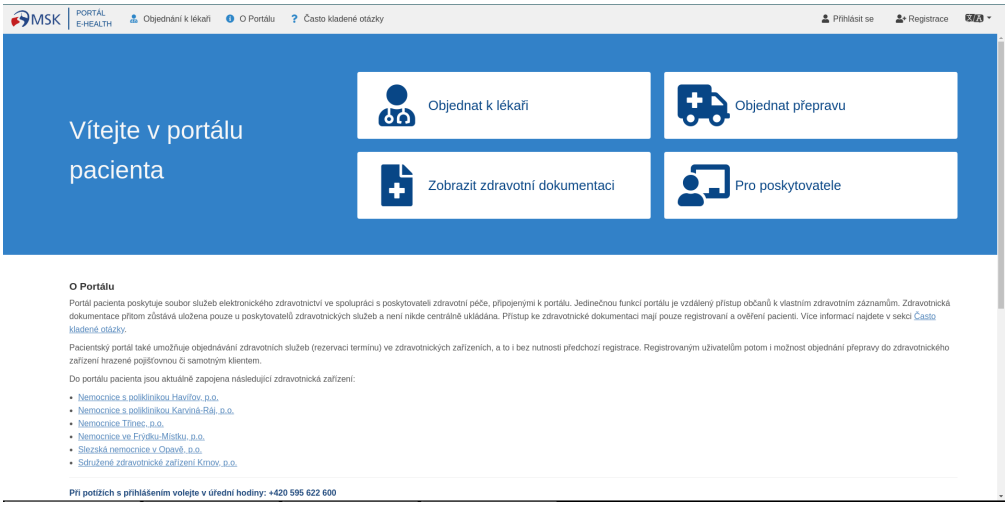


Figure 3: Main page of web app e-health

3 USED TECHNOLOGIES

For creating web application we have always front-end and back-end. Logic behind such solution is that front-end helps to display to the user what we want them to see. Back-end consists of logic that is behind the front-end part of the application.

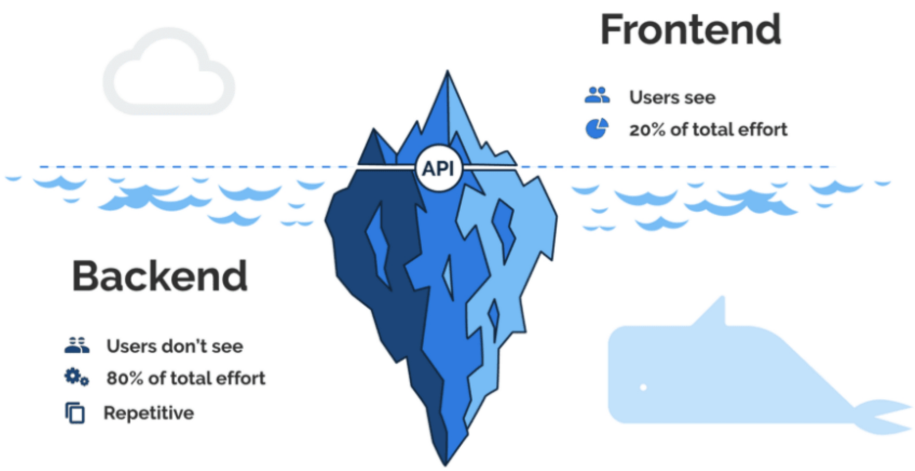


Figure 4: Difference between front-end and back-end of what user can see

There are three core technologies that we use for front-end: HTML, CSS and JavaScript

When we want to work with the logic behind web application we need a programming language that can work with HTTP (HyperText Transfer Protocol) requests/responses and which can communicate with database. Another requirement for the language is that it must have asynchronous code availability.

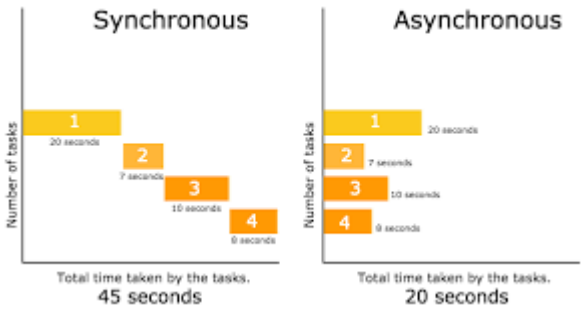


Figure 5: Syncronous vs asyncronous code

Today we have a lot of frameworks and languages that work with back-end coding. Most famous ones are ASP.NET, Ruby On Rails, Laravel (PHP framework) and Python.

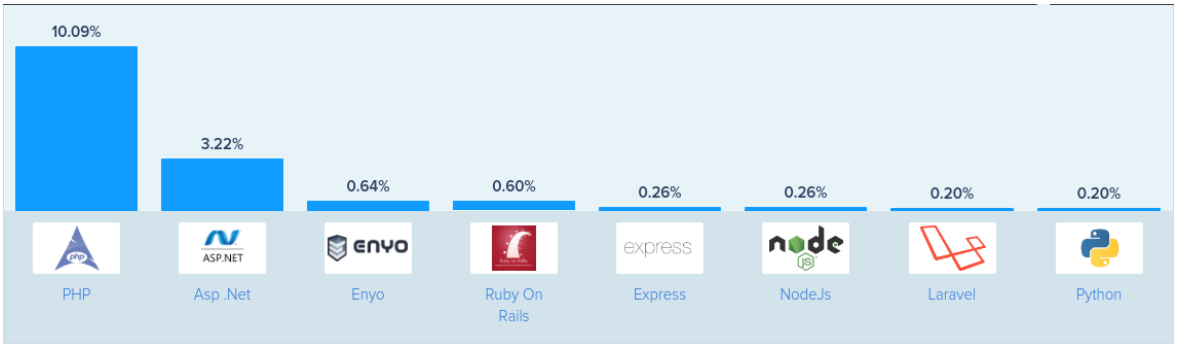
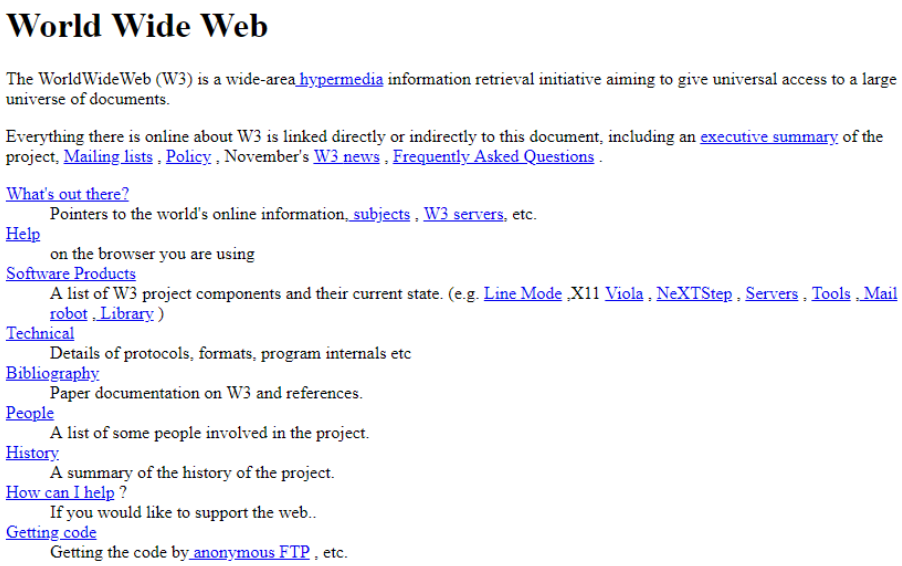


Figure 6: Top framework that are being used in applications from entire Internet [10]

3.1 Front-end

3.1.1 HTML

The cornerstone of any website is HyperText Markup Language (HTML). It is a markup language that defines elements on a web page. This language started in the 90‘s. Current version of HTML is HTML5.



World Wide Web

The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Everything there is online about W3 is linked directly or indirectly to this document, including an executive summary of the project, Mailing lists , Policy , November's W3 news , Frequently Asked Questions .

What's out there?
Pointers to the world's online information, subjects , W3 servers, etc.
Help
on the browser you are using
Software Products
A list of W3 project components and their current state. (e.g. Line Mode ,X11 Viola , NeXTStep , Servers , Tools , Mail robot , Library)
Technical
Details of protocols, formats, program internals etc
Bibliography
Paper documentation on W3 and references.
People
A list of some people involved in the project.
History
A summary of the history of the project.
How can I help ?
If you would like to support the web..
Getting code
Getting the code by anonymous FTP , etc.

Figure 7: First website [8]

Each HTML document consists of a basic structure containing a document type (DOC-TYPE), which is used by browsers to recognize the HTML version, as well as the page header, where we can define imports (cascade sheets, scripts, meta tags) and body of web-

page, where you can define what will be displayed to the user. There are a lot of tags for its usage, for example:

- header tags - <h1>, <h2>, ...<h6>
- paragraphs -
- section or division -

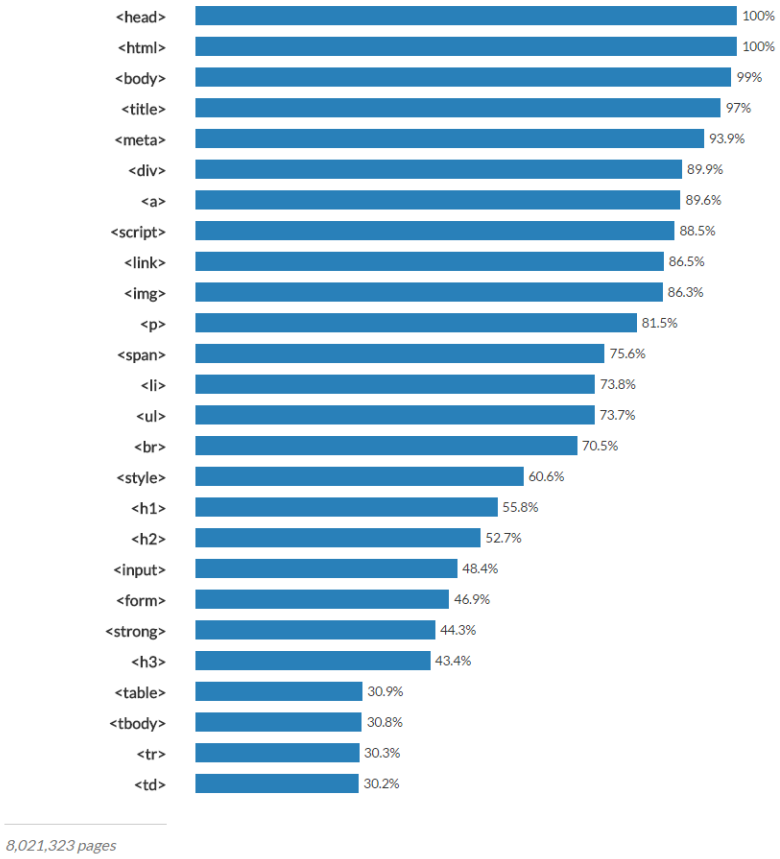


Figure 8: Statistic of tag usage for HTML [9]

3.1.2 CSS

Cascade Style Sheet describes how HTML will be displayed to the user on their screen. Before CSS existed, styles had to be inserted to HTML tag as attribute.

Nowadays a separate file that enables easier styling of HTML web page exists. This also eases the job if global *.css file for all webpages is needed. A problem can occur when there is a lot of classes/id in file and are not structured and therefore they can be lost in the file. The structure when typing new classes or id to the file is called BEM (Block-Element-

```html
<h1 style="color:blue;">A Blue Heading</h1>

<p style="color:red;">A red paragraph.</p>
```

Figure 9: Attribute inside of HTML tag

Modifier). This method teaches how to properly name classes or id and how to properly write attributes of such named instance.

HTML

```html
<form class="form form--theme-xmas form--simple">
  <input class="form__input" type="text" />
  <input
    class="form__submit form__submit--disabled"
    type="submit" />
</form>
```

Figure 10: HTML example for BEM method [11]

CSS

```css
.form { }
.form--theme-xmas { }
.form--simple { }
.form__input { }
.form__submit { }
.form__submit--disabled { }
```

Figure 11: CSS example for BEM method [11]

3.1.3 JavaScript

JavaScript is programming language that existed since early times of web pages. It is designed to work with the logic of HTML pages. It is not to be confused with programming language Java or island Java, as they are not the same.

Java is so called “weak type”, which means it has not data types even if we use numbers, strings or dates. We use following keywords to declare variables that we can use later on in development:

- var
- let
- const

Problem with JavaScript that it has no debugger, only way you can debug something is to print it to the console of the browser.

Another disadvantage is lack of browser support. While running, code is being run on client-side and each browser engine can interpret it differently. Therefore it can be exposed due to error in security and used for malicious purposes. Some people prefer disabling it completely on their browser.

Main usage of JavaScript is DOM (Document Object Model), which is able to access and change elements in HTML document. With this we can remove, change, add or delete specific elements of DOM either by marking the element with class or the first chosen tag.

```
let id = document.querySelector('#something');
let class = document.querySelector('.something');
let tag = document.querySelector('h1');
```

Figure 12: Example of DOM in JavaScript

3.1.4 Bootstrap

Bootstrap is free and open-source CSS framework that we can use to create „mobile-first“ graphical responsive interface on web page.

“Mobile-first” means that the goal of Bootstrap is to aim for mobile devices, such as smartphones or tablets. Because of this feature, Bootstrap is very popular among front-end developers to create web application. There are 4 suffixes that Bootstrap uses to determine display size.

- Without suffix – display to 567px („mobile-first“)
- Sm – small displays over 567px
- Md – medium to large displays over 768px
- Lg – large displays over 997px
- Xl – extra large displays over 1200px

These suffixes can be used in combination with other classes, where the change of HTML is depending on size of a display.

In Bootstrap we can create a Grid-view system that can help us to create a layout of our application. Those grids will change based on he size of the device's display.

One of three columns	One of three columns	One of three columns

Copy

```
<div class="container">
  <div class="row">
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
    <div class="col-sm">
      One of three columns
    </div>
  </div>
</div>
```

Figure 13: Example of Grid display

Bootstrap also offers huge variety of prepared components that we can use for our HTML page. Most useful components are for creating navigation bar, forms, pop-up and sliders.

Only disadvantage of using Bootstrap is that it needs to have correctly set up dependencies, such as Popper.js and jQuery.

3.1.5 Node.js

Node.js is open-source, JavaScript runtime engine, that runs JavaScript code outside of web browser. It is a cross-platform, which means that it can be used on any other operating system such as Windows 10, MacOs or on any Linux distribution. It is a non-blocking asynchronous I/O.

Meaning of non-blocking operation is that the code is not being blocked by execution. It does not wait for other previous blocks to finish execution, which is the case for JavaScript that has a blocking operations [12]. Advantage of asynchronous non-blocking code is that it is using single thread to execute all requests. We can see the example on Figure 13.

When requests arrive to the server, they are serviced one at a time. But when the code that is requesting service requires DB query, it sends the callback to a second queue and the main thread will continue running [12].

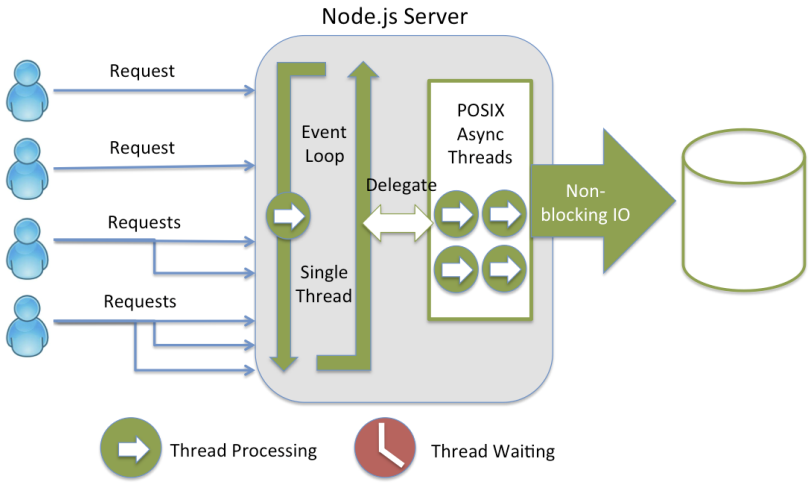


Figure 14: Example of asyncronous I/O with Node.js [13]

3.1.6 Angular

Angular is a TypeScript-based open-source front-end web framework that is being maintained by Google, Angular team and community. With Angular, we can easily develop applications, if developer knows basics of JavaScript. However, if we want to work with JavaScipts DOM‘s (Document Object Model) or jQuery, it demands complex development, maintaining the code and following patterns of design. That being said, using Angular as front-end, rather than using JavaScript or jQuery, saves a lot of time and money.

It is being used to develop single-page (SPA) applications. With Angular we can build not only web-application, but we can use it to build mobile application as well. With Angular, we develop front page with components, that have templates, which will be shown to the user. In components we have two main templates, one that is showed to the user (*.component.html) and another that is using data to bind to the view (*component.ts). Advantage of Angular is that it does not call every single page when it is being called, it is using Routing to do that job for you, it will show unique view with calling of URL, that is being written in the file for routing.

3.1.7 Ng-Bootstrap

Ng-Bootstrap is set of components and directives that is designed for Angular. It needs Bootstraps CSS. It offers easier usage of Bootstrap for Angular application.

Advantage of this set of components is that it is easy to install and does not require to be installed manually. Another advantage is that it does not require to use Popper.js and

jQuery as dependency, which is important for classic Bootstrap. It has very good documentation for each components including examples and API.

3.2 Back-end

3.2.1 C#

C# is high-level object-oriented programming language that is being developed by Microsoft. It can be used to develop mobile applications and desktop applications, either as WPF or UWP and many other things.

This language is inspired by C++ language, because it has data types such as int, long, float, etc.

It is built to run on CLI, which is known as Common Language Infrastructure and it can interact with other languages that are built on same architecture [14]

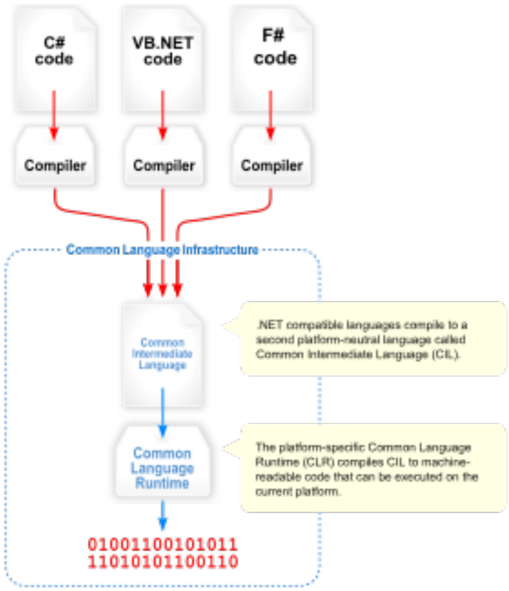


Figure 15: Visual overview of the Common Language Infrastructure

Over the course of beginning of the language existence, it is slowly starting to compete with other programming languages. It received major enhancements that put it forward including Generics, LINQ (Language Integrated Query), Dynamics and async/await pattern.

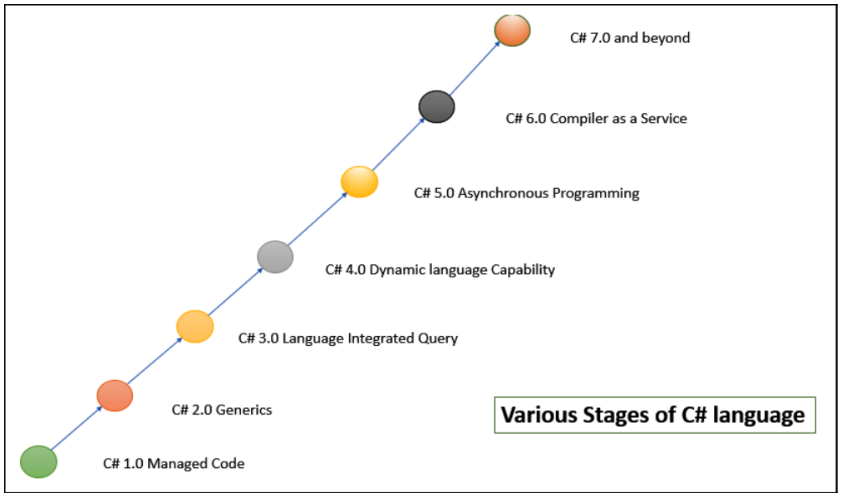


Figure 16: Evolution of C# [14]

When C# code is run, the code is firstly transformed to Intermediate Language (IL) and it is being saved in executable file (*.exe). To execute the code, it needs to use Common Language Runtime (CLR) to interpret it from IL with Just-in-Time compiler.

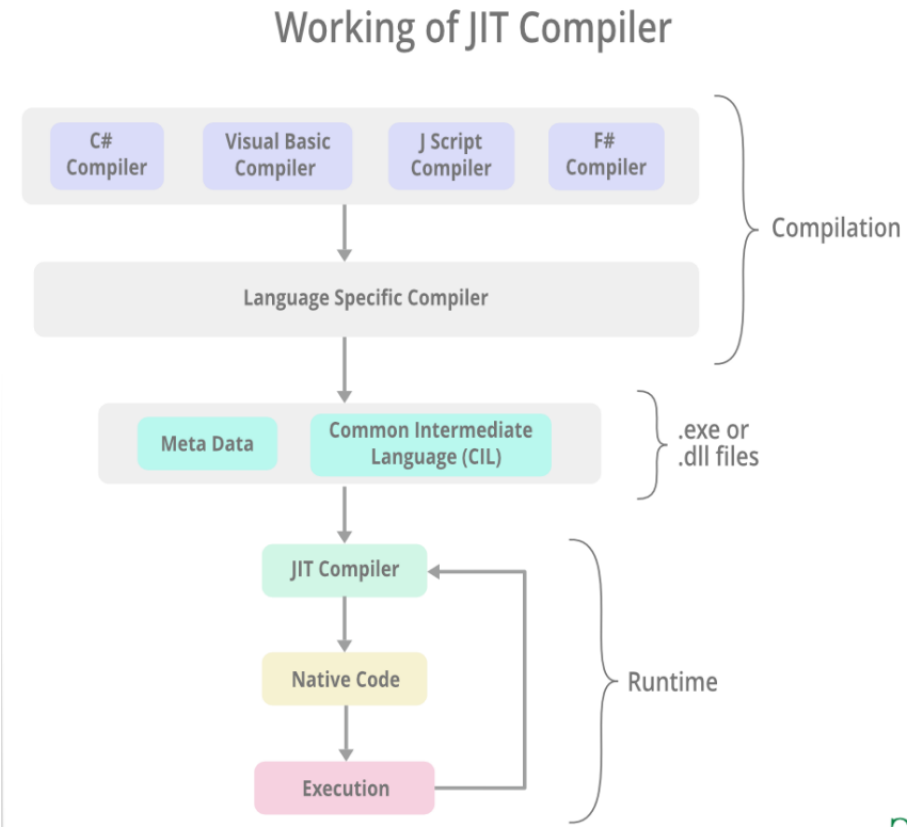


Figure 17: Overview of how JIT Compiler works in .NET Framework

3.2.2 ASP.NET Core

ASP.NET Core is a open-source cross-platform C# framework designed for web development. It is designed by Microsoft to allow programmers to create dynamic websites, applications and web services.

It is a modular of Window's .NET Framework and for cross-platform .NET Core. At version 3.0.x of ASP.NET Core dropped support for .NET Framework.

In ASP.NET Core we can start creating from modules for web development:

- Main modules for web development
  - ASP.NET Web Forms
  - ASP.NET MVC – allowing to build web pages using model-view-controller design pattern
  - ASP.NET Web Page
  - ASP.NET. Web API – framework to build Web API on server-side
  - SignalR – real-time communication framework for communication between client and server
- Other modules
  - ASP.NET Handler
  - ASP.NET AJAX
  - ASP.NET Dynamic Data

Framework also offer CLI (Command Line Interface) for creation of projects if user's code is using text editor like Vim.

3.2.3 Entity Framework Core

Entity Framework Core is open-source Object Relation Mapping framework (ORM) that was part of .NET Core but since version 6 its has detached. Since then it has own release schedule.

Entity Framework makes it easier to create database with C# instead of writing SQL queries when programming an app that uses databases. With this, we can create models and it's relationships with other models, we can call from database to get specific data from it using LINQ or Collection. EF Core is compatible with today's major open-source and

commercial SQL and NoSQL engines, all thanks to the official and third-party packages that are available through NuGet package manager [15]. EF Core also offers command line interface (CLI) to create migrations, databases and database contexts

EF Core support three types of creation of data modeling for database:

- Database-first
- Model-first
- Code-first

Each of these data modeling has pros and cons, but we will focus rather on code-first approach that is being used for the thesis.

```
> $ dotnet ef

                      _/\__
               -----==/    \\
         ___  ___   |.    \|\
        | __|| __|  |  )   \\\
        | _| | _|   \_/ |  //|\\
        |___||_|       /   \\\/\\

Entity Framework Core .NET Command-line Tools 3.1.4

Usage: dotnet ef [options] [command]

Options:
  --version        Show version information
  -h|--help        Show help information
  -v|--verbose     Show verbose output.
  --no-color       Don't colorize output.
  --prefix-output  Prefix output with level.

Commands:
  database    Commands to manage the database.
  dbcontext   Commands to manage DbContext types.
  migrations  Commands to manage migrations.

Use "dotnet ef [command] --help" for more information about a command.
```

Figure 18: Example of CLI command for EF Core "dotnet ef" in Linux's terminal

Code-first is more popular approach due to the simplicity of database design. Database is being created based on model that is defined using standard classes being created in the code. With this, we do not need any XML mapping or design tool [15].

As mentioned before, advantage of code-first approach is that we do not need design tools to create database and such approach is ideal for small to medium sizes projects because it helps to easily to maintain the code and saves a lot of time when designing database.

```
public class Appointment
{
    1 usage
    public int Id { get; set; }
    1 usage
    public Patient Patient { get; set; }
    1 usage
    public Physician Physician { get; set; }
    3 usages
    [ForeignKey( name: "Patient")] public int PatientFKId { get; set; }
    public string? PatientFullName { get; set; }
    1 usage
    [ForeignKey( name: "Physician")] public int PhysicianFKId { get; set; }
    public string? PhysicianFullName { get; set; }
    5 usages
    public DateTime StartOfAppointment { get; set; }
    1 usage
    public DateTime EndOfAppointment { get; set; }
    public string? Description { get; set; }
```

Figure 20: Example of class for creating table with code-first approach

Disadvantage is that it is not suitable for large projects, because the code needs to be constantly maintained and having a large amount of models can cause spending more time checking the code. It also requires to have good amount of knowledge of C# and EF to be able to successfully create tables.

```
public class DataContext : IdentityDbContext<
    User, Role, int, IdentityUserClaim<int>, UserRole, IdentityUserLogin<int>, IdentityRoleClaim<int>,
    IdentityUserToken<int>>
{
    public DataContext(DbContextOptions<DataContext> options) : base(options)
    {
    }

    4 usages
    public DbSet<Appointment> Appointments { get; set; }
    3 usages
    public DbSet<Physician> Physicians { get; set; }
    4 usages
    public DbSet<Patient> Patients { get; set; }
```

Figure 21: Example of declaring class for database context

3.3 Other used technologies

3.3.1 Git

Git is open-source tool for handling projects from small to large size. It is used to track changes in a source code during development. Creator of Git is Linus Torvald, who created it for development Linux kernels since 2005. Before Git, BitKeeper was being used, a proprietary source-control management (SCM).

Git controls changes in documentation and it is initialized in specific directory called working directory where programmer will be working on it during development.

There are services that offer Git repositories online. The most popular one is GitHub and GitLab (mainly for bussines).

3.3.2 GitKraken

GitKraken is multi-platform GUI (graphical user interface) for Git, it was developed as an alternative to the command line. In GitKraken we can visually see our commits, branches, easy sync with Git services (either with Github or Gitlab).

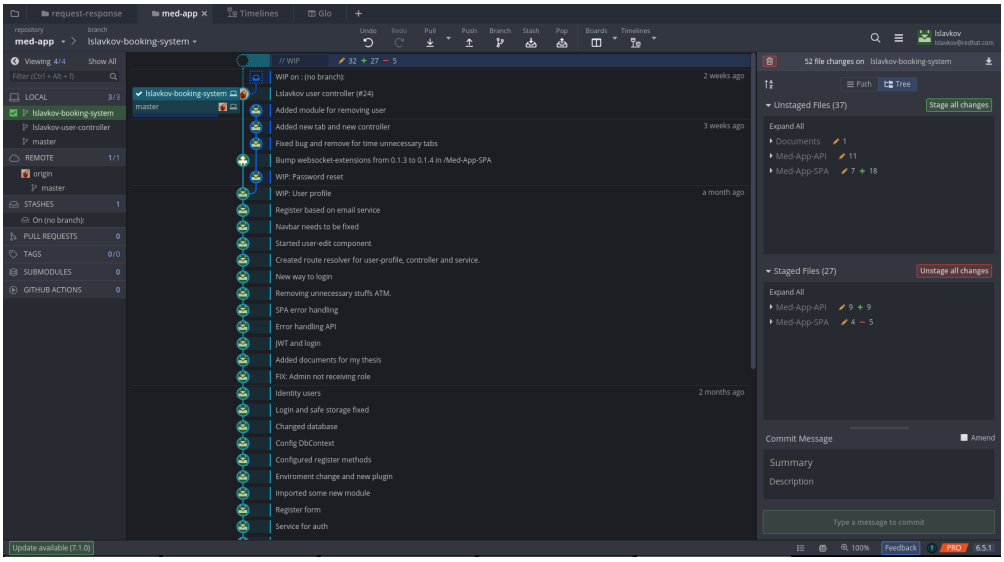


Figure 22: GitKraken for Git

GitKraken later created other products and changed it's name to GitKraken Git. Other products that are free if user signs as a student through universities/high schools e-mail is GitKraken Boards (Kanban board) and GitKraken Timeline.

3.3.3 Rider

Rider is a cross-platformed IDE (Integrate Developing Environment) from the company JetBrains. It is a complex IDE for development in C# language for applications, web applications, web API's, desktop applications, mobile development, and so on.

Rider offers extending IDE with plugins from JetBrains or from community that created them for Rider and other products.

This product's version is payed but it is possible to get it for free as a student either with e-mail from university/high school or international student's ISIC card.

Rider offers IntelliSense that whispers auto-completion and giving tips on code (naming the variable for example). It also offers great deal of other tools such as Database viewer, debugger, integrated terminal, it's own Git's GUI tool.

4 WEB APP VULNERABILITIES

Web apps contain a lot of user's information that can be exposed to potential security risk. In this chapter we will present some of the most common web app attacks than can happen.

4.1 SQL Injection

SQL Injection (or as commonly known as SQLi) is one of most common attacks on web application. It attacks vulnerabilities with queries to the database through form. It allows attacker to see data from database that is not normally visible to the user. Attacker can see data that belongs to other users, see other data that is in database or delete data. There are some situations that can come to compromise to the server or DOS (denial-of-service) attack [16].

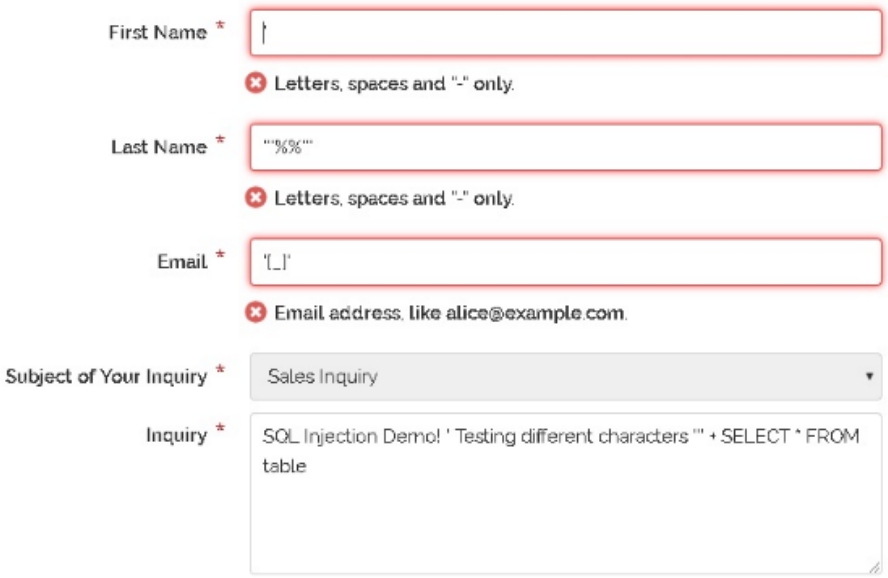


Figure 23: Example of SQL Injection attack using form [17]

If attack is successful, attacker can get sensitive data of user such as passwords or user's details (which can lead to identity theft). Most of the time, breaches in recent years are due to SQL Injection attacks. There are some cases where attacker can gain access to backdoor authorization of the system, where it can lead to large damage to the organization's application, if it is unnoticed[16].

4.2 Cross-site scripting (XSS)

Cross-site scripting is another type of an attack that uses Javascript to execute commands in another user's browser [18].

This attack does not directly attacks victim but instead, this attack exploits a vulnerability in a website that every victim visits. Once the victim visits that website, it can collect some of user's sensitive information, get user's key logging (registering user's keyboard using *addEventListener* and return to the attacker's server information) or phishing (creating fake login form to get sensitive data from user) [18].

There are three types of XSS attacks:

- Persistent – malicious string originates from web's database
- Reflected – originates from victim''s request
- DOM-based – in client-side code rather than on server-side code

4.3 Broken Authentication a Session Management

This type of attack creates a session cookie and session ID for each time where there is a valid session, and these cookies will collect data like username and password. When user ends his session either by a logout or browser, these cookies should be invalidated. This means that for every new session there should be new cookie. If this cookie is not invalidated, user's sensitive data will exist in the system [19].

We can secure against this type of an attack with [20]:

- setting up properly application's timeout
- properly hashing and salting a password
- forcing user to create a strong password policy
- never exposing credentials in URL's or in logs.

4.4 Cross Site Request Forgery (CSRF)

Cross Site Request Forgery, or commonly known as CSRF, is an attack that forces the victim to execute malicious requests on a web app from where they are authenticated. Attacker can trick victims of web application to perform an action of attacker's choosing. It can be two main scenarios. If the victim is a regular user, an attack can force the user to perform requests like transferring funds, change password or email, etc. If the user is an administrator, this attack can compromise an entire web application [21].

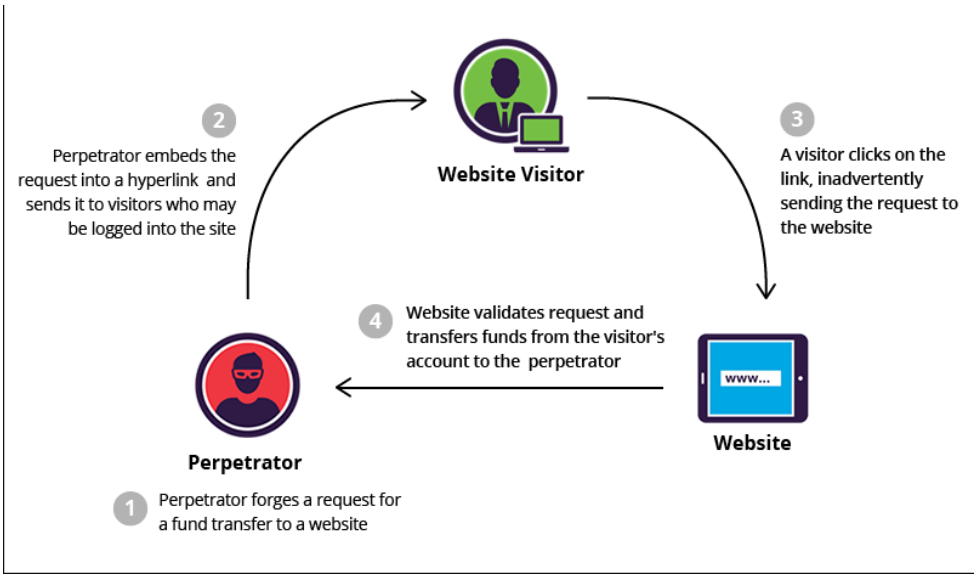


Figure 24: Overview how CSRF attack works

II. ANALYSIS

5 DESIGN FOR APPLICATION

In this section of the thesis, author will show requirements for the app, use-case models with scenarios and implementation of the front-end and back-end.

5.1 Analyzation of requirements

In this part, author will display the requirements for the application that are needed for this application. These requirements are split into two parts:

- functional requirements
- nonfunctional requirements

5.1.1 Functional requirements

With functional requirements, the functionality of the system is defined. These requirements are split into packages:

- Patient
- Physician
- System
- User

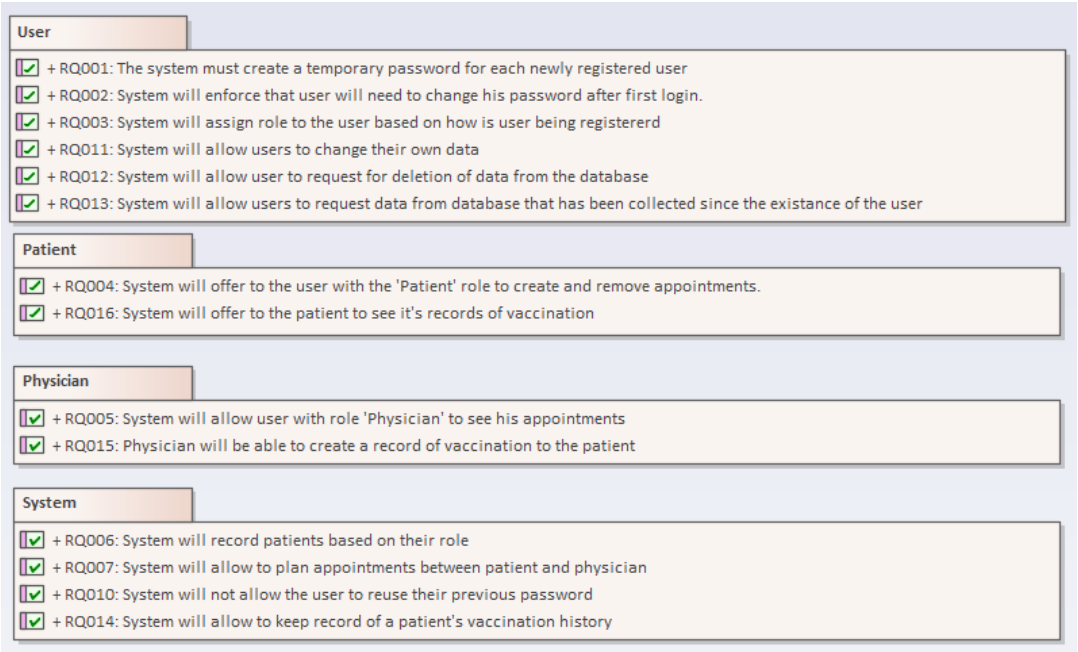


Figure 25: Functional requests

RQ001: The system must create a temporary password for each newly registered user

- The system will create a temporary password for each new customer account, which will be valid until the first login. The password will respect the password strength requirements.

RQ002: System will enforce that user will need to change his password after first login.

- If a new user logs in for the first time, they will need to change password according to the security rules on creating passwords.

RQ003: System will assign a role to the user based on how was the user registered

- System will assign a role to the user based on how has the user been registered. If a user is being registered on sign-up form, it will be registered with role 'Patient'. If user is being registered through admin's register form it will be registered with role 'Physician'

RQ004: System will offer to the user with the ‚Patient‘ role to create and remove appointments.

- When patient wants to create appointment, he can choose:
  - physician of his choice
  - type of appointment of his choice
  - date
  - time
  - and if needed, description can be added

RQ005: System will allow user with role ‚Physician‘ to see his appointments

- This user can see the name of the patient and date of an appointment with some details in description upon inspection.

RQ006: System will record patients based on their role

- System will give users role ‚Patient‘ after being registered. Patient will be recorded on two tables:
  - ‘UserRoles’
  - ‘Patient’

RQ007: System will allow to plan appointments between patient and physician

- Patient can choose from available physicians to create an appointment, they can also add notes into the description of the appointment so the chosen physician will know ahead of time.

RQ010: System will not allow the user to reuse their previous password

- System will not allow the user to have their new password to be identical to their previous one, it needs to be unique

RQ011: System will allow users to change their own data

- User can change their own first and last name

RQ012: System will allow user to request deletion of data from the database

- All data relating to that user will be deleted, that includes tables, where data is being recorded such as what role they have and their appointments

RQ013: System will allow users to request data from database that has been collected since the existence of the user

- User can request only his data, not data of entire system.

RQ014: System will allow to keep record of a patient's vaccination history

- Patient will be able to have a record of his vaccination history in the application for him to see and keep up with

RQ015: Physician will be able to create a record of vaccinations of the patient

- When physician enters new record to the patient he will:
  - choose a patient to create a new record
  - type of vaccine
  - dosage of vaccine
  - date of vaccination
  - if needed, description

RQ016: System will offer to the patient to see it's records of vaccination

- User with role 'Patient' can see his own records of vaccination.

5.1.2 Non-functional requirements

Non-functional requirements are part of the application that define how the system is supposed to be.

RQ008: System will force user, based on time cycle, to change their password

- For extra layer of safety, users will have to change password based on time cycle (every 3 months)

RQ009: System will enforce security rules upon new passwords after user requests its creation

- Rules for creating a password are:
  - minimum length is 8 characters
  - maximum length is 16 characters
  - it requires a digit in a password

5.2 Use-case models

Use-case models describe basic functionality of the application. In the application there are 4 basic actors:

- *User*
- *Admin*
- *Patient*
- *Physician*

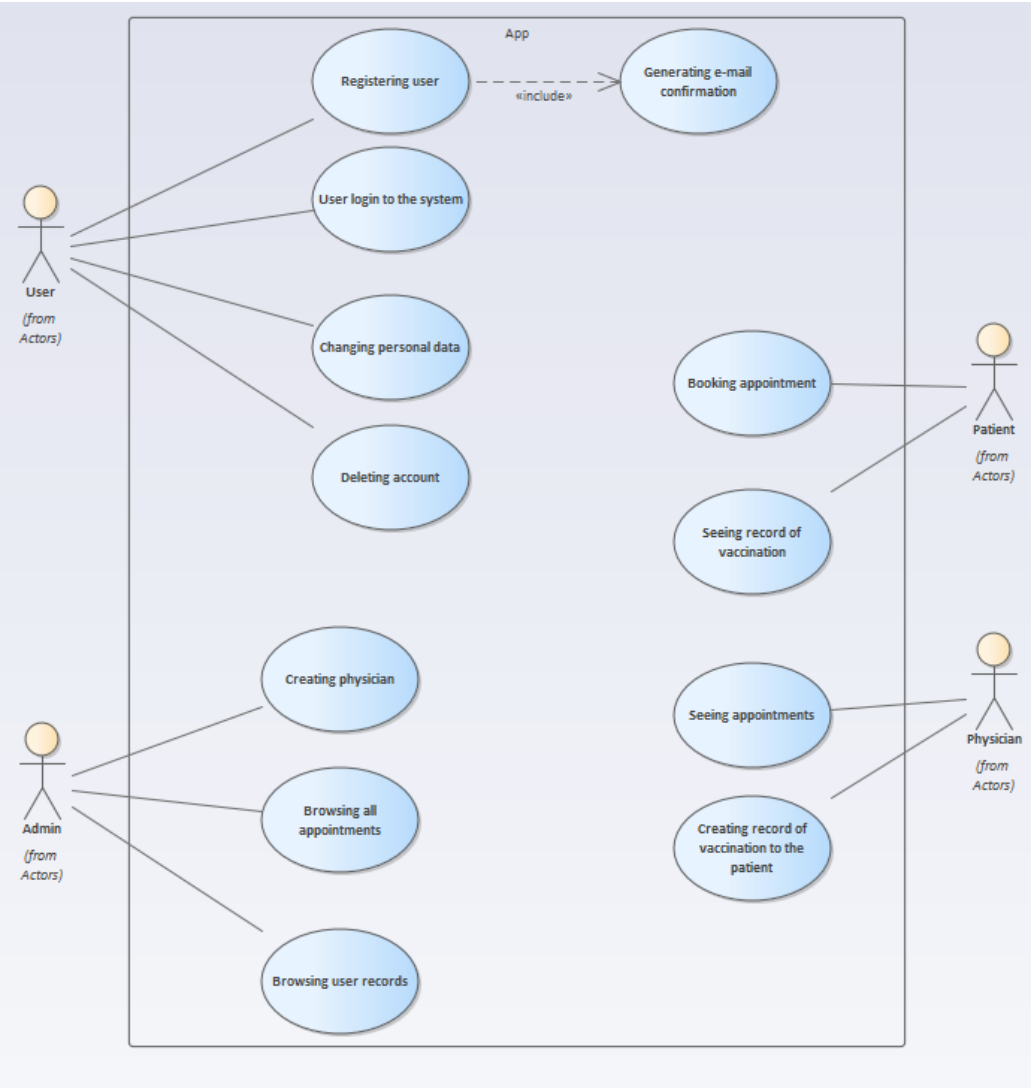


Figure 26: Use-case model

5.2.1 Scenarios

In this section, we will show the scenarios that are part of the use-case model. There are two types of scenarios:

- Main scenario
- Alternative scenario

Table 1: Scenario for registering user

Name: Registering user		
Characteristic: If user wants to use the app for booking an appointment, he has to register.		
Primary actor: User		
Secondary actor: None		
Input condition: User wants to register		
Output condition: Generating validation e-mail		
Main scenario:		
Steps	Actor/System	Description
1	User	User enters the registration page
2	System	System will show the User register form
3	User	User fills it out
4	System	System creates record of the new User
5	User	<include> Generating e-mail confirmation
Alternative scenario: None		

Table 2: Scenario for user to log in

Name: User login to the system		
Characteristic: User will log in with login data they created during registration		
Primary actor: User		
Secondary actor: None		
Input condition: User logs into the application		
Output condition: User is successfully logged in		
Main scenario:		
Steps	Actor/System	Description
1	User	User logs into the app
2	System	System checks if User exists and if it is verified
3	User	User gains access to the app
Alternative scenario: 2a - System denies access to the app because the user is not verified.		

Table 3: Alternative scenario for user to login

Name – System denies access to the app because the user is not verified.		
Characteristic: System requires from user to be verified by generated e-mail he received during registration		
Alternative scenario:		
Step	Actor/System	Description
1	System	System denies login details of the User
2	User	User is returned to the login page

Table 4: Scenario for booking appointment

Name: Booking appointment		
Characteristic: Patient can book an appointment with the physician		
Primary actor: Patient		
Secondary actor: None		
Input condition: User starts to book appointment		
Output condition: User's appointment is successfully booked		
Main scenario:		
Steps	Actor/System	Description
1	User	Patient creates appointment
2	System	System processes creation of the appointment
3	User	Patient receives confirmation from the system about existence of the appointment
Alternative scenario: 3a - User receives a message that says that the appointment cannot be created due to the time being already booked or outside office hours		

Table 5: Alternative scenario for booking appointment

Name – User receives a message that says that the appointment cannot be created due to the time being already booked or outside office hours		
Characteristic: System requires from the user to be verified by generated e-mail he received during registration		
Alternative scenario:		
Step	Actor/System	Description
1	System	System rejects booking a appointment
2	User	Patient is redirected to the booking form

Table 6: Scenario for updating personal data

Name: Changing personal data		
Characteristic: User can change his personal data in account settings		
Primary actor: User		
Secondary actor: None		
Input condition: User is changing his personal data		
Output condition: User has successfully changed his personal data		
Main scenario:		
Steps	Actor/System	Description
1	User	User requests his own personal data
2	System	System processes request for the user's data
3	User	User changes his personal data
4	System	System processes new changed information
5	User	User sees new data displayed

Table 7: Scenario for deleting user account

Name: Deleting account		
Characteristic: User can delete his account data in account settings		
Primary actor: User		
Secondary actor: None		
Input condition: User is deleting his account		
Output condition: User is successfully deleted his account		
Main scenario:		
Steps	Actor/System	Description
1	User	User requests his data to be deleted from the app
2	System	System processes user's request for deleting their account
3	User	User is deleted
Alternative scenario: 3a - System shows error during deletion		

Table 8: Alternative scenario for deleting user

Name – System shows error during deletion		
Characteristic: System denies deletion of account		
Alternative scenario:		
Step	Actor/System	Description
1	System	System rejected deletion of user's profile
2	User	User is being redirected to the account's settings

Table 9: Scenario for browsing appointments

Name: Browsing all appointments		
Characteristic: Admin can browse all data that is being recorded in database by the system.		
Primary actor: Admin		
Secondary actor: None		
Input condition: Admin is requesting data of all appointments		
Output condition: Admin has successfully received data		
Main scenario:		
Steps	Actor/System	Description
1	User	Admin requests data about appointments
2	System	System process request about appointments
3	User	Admin is browsing data
Alternative scenario: None		

Table 10: Scenario for browsing user's

Name: Browsing user records		
Characteristic: Admin can browse recorded users that are saved in database		
Primary actor: Admin		
Secondary actor: None		
Input condition: Admin is requesting data of all users		
Output condition: Admin has successfully received users data		
Main scenario:		
Steps	Actor/System	Description
1	User	Admin requests records about a user in database
2	System	System processes request
3	User	Admin is browsing
Alternative scenario: None		

Table 11: Scenario for requesting records of vaccination

Name: Browsing records of vaccination		
Characteristic: Patient is browsing his records of vaccination		
Primary actor: Patient		
Secondary actor: None		
Input condition: Patient request records of his vaccination		
Output condition: User has successfully received records		
Main scenario:		
Steps	Actor/System	Description
1	User	Patient request to see his records of vaccination
2	System	System processed request
3	User	Patient receives his record from the system
Alternative scenario: None.		

Table 12: Scenario for creating record of vaccination for the patient

Name: Creating record of vaccination for the patient		
Characteristic: Physician will create a record of vaccination for the patient		
Primary actor: Physican		
Secondary actor: None		
Input condition: Physician enters record of vaccination to the patient		
Output condition: Physician has successfuly added record of vaccination		
Main scenario:		
Steps	Actor/System	Description
1	User	Physician creates a record of vaccination for the patient
2	System	System processed request for recording
3	User	Physician succesfully receives recordings of vaccination for the patient
Alternative scenario: None.		

Table 13: Scenario for creating user with role ‚Physician‘

Name: Creating physician		
Characteristic: Admin can create another user with role 'Physician'		
Primary actor: Admin		
Secondary actor: None		
Input condition: Admin creating user with role ‚Physician‘		
Output condition: Admin has created new physician		
Main scenario:		
Steps	Actor/System	Description
1	User	Admin request sregister form
2	System	System shows register form to the Admin
3	User	Admin fills registration details about Physician
4	System	System creates new user with role 'Physician'
Alternative scenario: None.		

6 IMPLEMENTATION OF THE APPLICATIONS

In this chapter we will show what requirements are fulfilled and client-side and server-side of application.

6.1 Fulfilled requirements

In this section we will show what requirements have been fulfilled.

6.1.1 Fulfilled requirements

Requirements that are fulfilled:

- **RQ003**: System will assign role to the user based on how is user being registered
- **RQ001:** The system must create a temporary password for each newly registered user
- **RQ004**: System will offer to the user with the ‚Patient‘ role to create and remove appointments.
- **RQ005**: System will allow user with role ‚Physician‘ to see his appointments
- **RQ006**: System will record patients based on their role
- **RQ007**: System will allow to plan appointments between patient and physician
- **RQ011**: System will allow users to change their own data
- **RQ012**: System will allow user to request deletion of data from the database
- **RQ009**: System will enforce security rules upon new passwords after user requests its creation
- **RQ014**: System will allow to keep record of a patient's vaccination history
- **RQ015**: Physician will be able to create a record of vaccination for the patient
- **RQ016**: System will offer to the patient to see their records of vaccination

6.2 Client (Front-end)

In this section we will present front-end of the web application. Note this is a prototype, therefore the design of the application has according looks to mainly show functionality of the app.

When creating application author tried to create as simple as possible front-end that does not have a lot of visual elements on the application.

On application we can see from two points of view:

- as patient or physician
- as administrator

There are error messages that are same across application to the user. Alerts of either successful message or error message will pop up. It is implemented with AlertifyJS that allowed us to create pop up messages on bottom right screen.

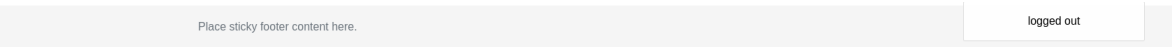


Figure 27: Example of AlertifyJS pop up message

Main page

On the main page, navigation bar with ‚Sign In‘ and ‚Sign Up‘ buttons was created, that allow user to either sign-in or allow a new user to be created. We can also see information about working hours.

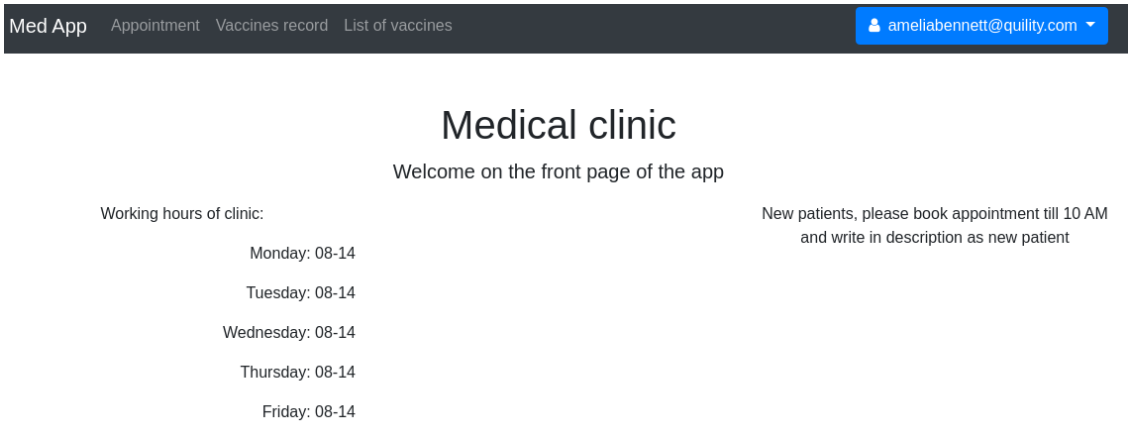


Figure 28: Main page of application

Registering form is very simple, it does not require a lot of personal information and does not require any sensitive personal information to register. It only requires first name, last name and e-mail to register.

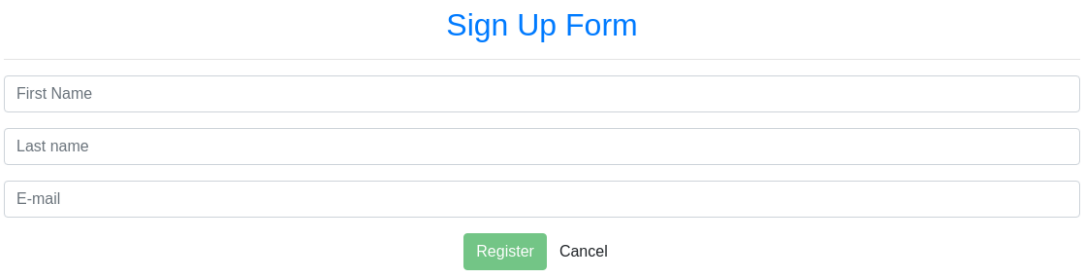


Figure 29: Sign up form

6.2.1 View from patient/physician

When user signs in as a patient, he can see his upcoming appointments that they have recorded in database. Information about his upcoming appointments show to which physician is patient going, at what time is patient booked with physician and if patient added description during booking, it will also show description about appointment.

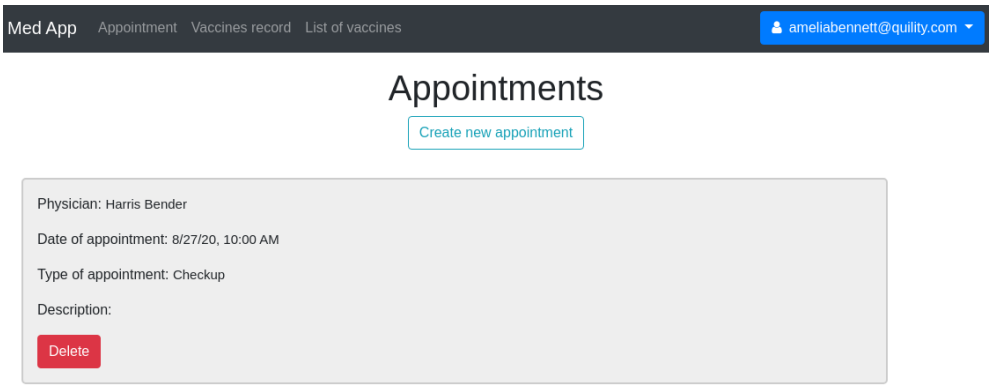


Figure 30: Appointment page from patient side

This goes similar for physician, that can also see his appointments, that he has in upcoming days, but difference is, that physician cannot delete those appointments neither can he create a new ones.

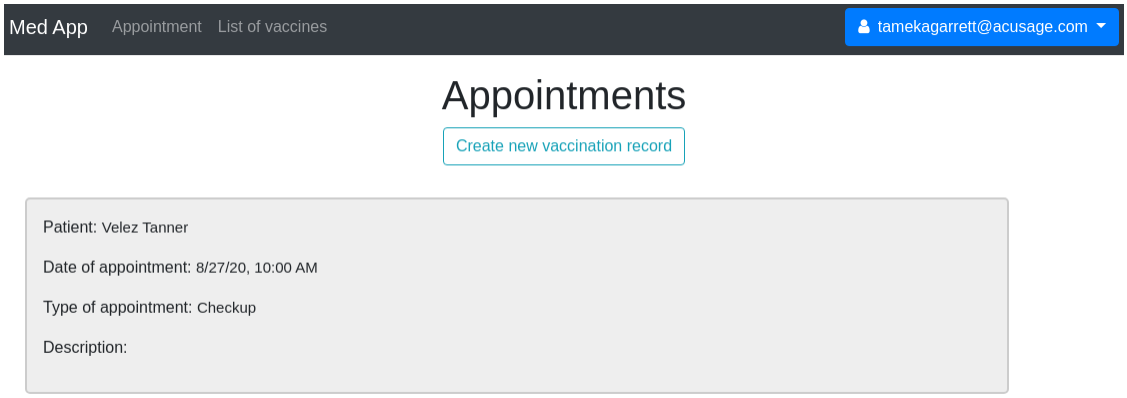


Figure 31: Appointment page from physician side

Physician can add details to the patients vaccinations records. He can add such information at any time.

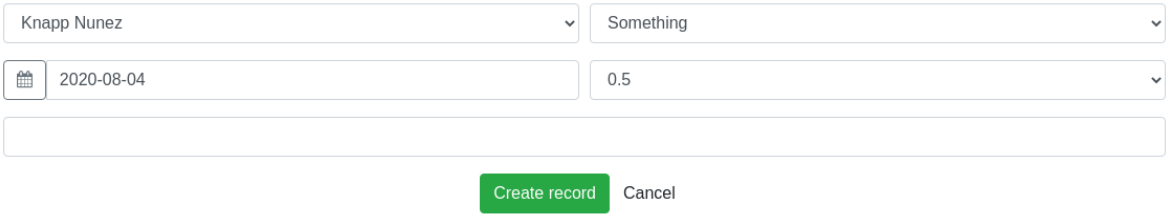


Figure 32: Vaccination record form

Physician can choose to which patient, type of vaccine, date of vaccination, dosage in ml and, if needed, a description to the record.

Vaccination record can look something like this to the patient upon inspection.

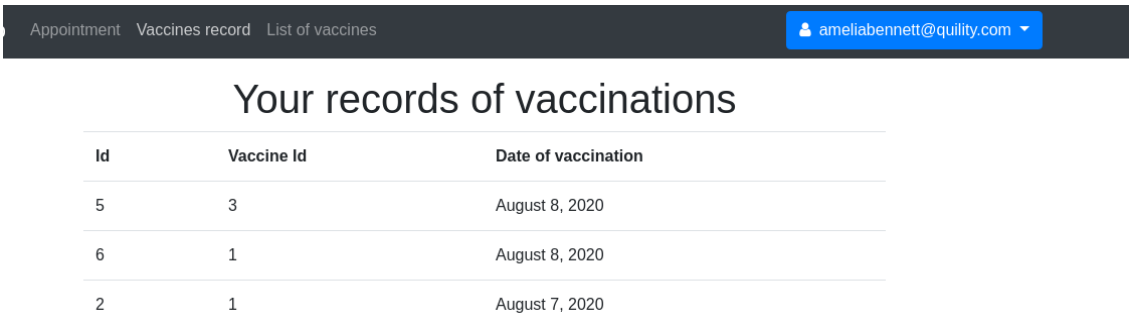


Figure 33: List of vaccinations patient had

To see ids of vaccination, there is a list of vaccines in navigation bar, where patient or physician can see.

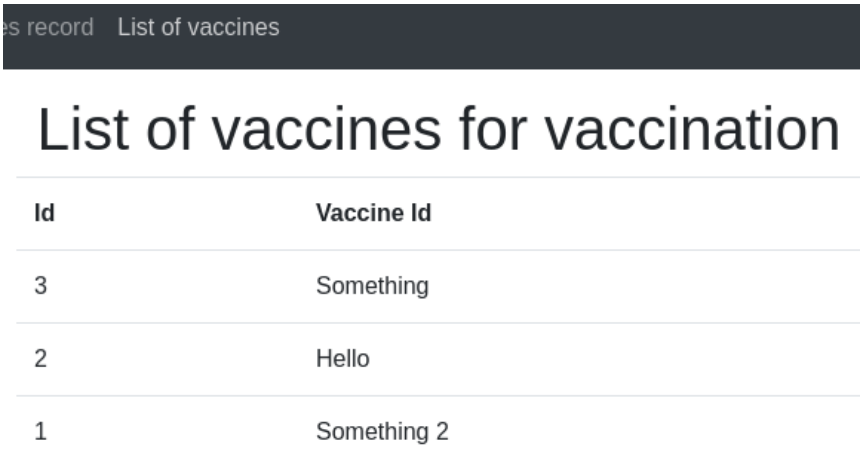


Figure 34: List of available vaccines

The profile's settings is the same for patient and physician. They can edit their first and last name and set a new password for their account. It also shows details about user in the profile settings.

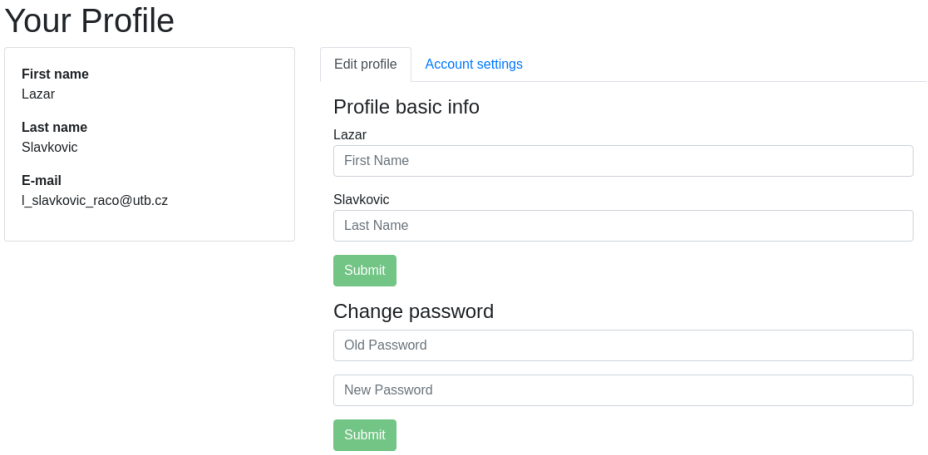


Figure 35: Profile settings from user's

User can also delete their account in tab ‚Account settings', where they can find button for deletion of their account.

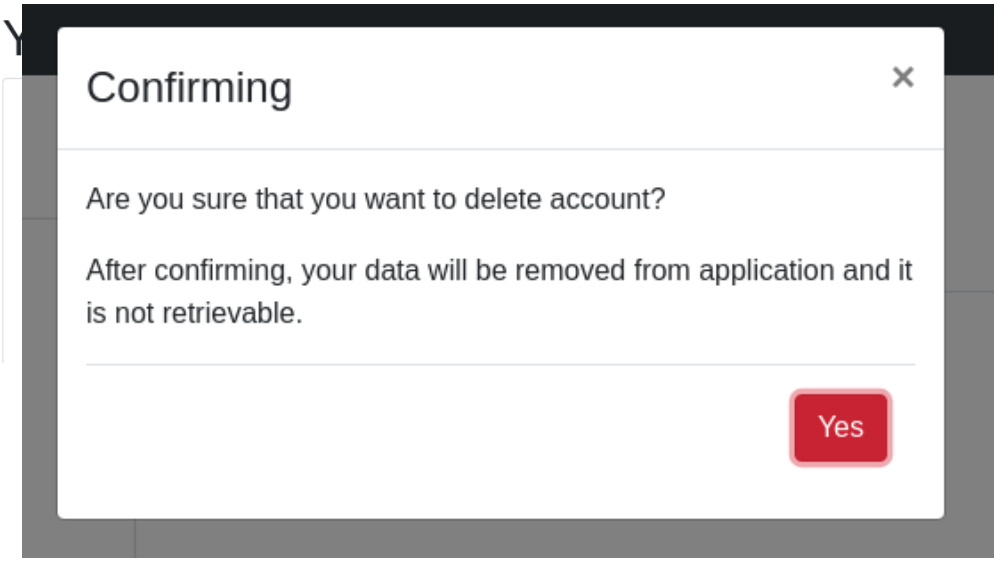


Figure 37: Confirmation if user wants to be deleted.

After user clicks button, it will be needed to confirm if the user really wants to be „forgotten“. If he does confirm, his relevant account information will be deleted, such as what roles he got, his appointments and record of being either a patient of physician.

6.2.2 View from administrator

Administrator has access to data that are stored. He can see users, appointments, but he can also create new physician from his side and add to the list of available vaccines on his page. He can also change his password, first name and last name in profile's settings. Admin can also delete user in the same way as a regular user would request deletion. Only difference is, that it does not require any confirmation whether he is sure. Administrator also cannot delete himself.

Admin panel

User management | Appointment management | Vaccine management | Create physician

Id	E-mail	First Name	Last Name	Roles	
14	admin@example.com	Admin	Admin	Admin	Delete user
2	sarahwalter@pyrami.com	Terrell	Leach	Patient	Delete user
10	fitzgeraldlindsey@plasmox.com	Clements	Cantu	Patient	Delete user
11	tamekagarrett@acusage.com	Harris	Bender	Physician	Delete user
9	brendawhite@isosphere.com	Howe	Wynn	Patient	Delete user

Figure 38: Admin side for viewing users that are stored in database with their role displayed

In another tab of administrator's page, he can also see all past and upcoming appointments with names of patients and physician with date and description. Admin can't manipulate these data.

Admin panel

User management | Appointment management | Vaccine management | Create physician

Id	Physician	Patient	Start of appointment	End of appointment	Type of appointment	Description
2	Harris Bender	Velez Tanner	8/27/20, 10:00 AM	8/27/20, 10:15 AM	Checkup	

Figure 39: Admin side for viewing appointments

Administrator can also see and add new vaccine that clinic offers for the patients to get vaccinated.

Admin panel

User management | Appointment management | Vaccine management | Create physician

Add vaccine

Id	Name
3	Something
2	Hello
1	Something 2

Figure 40: Admin side for viewing list of vaccines in the offer

If administrator wants to add new vaccine in the offer, he can do it with simple form of just adding a name of the vaccine.

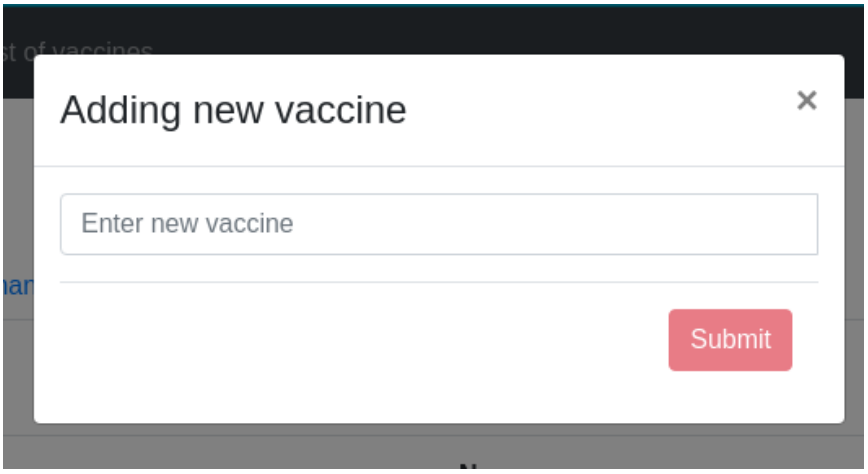


Figure 41: Form for adding new vaccine to the list

In another tab, administrator can add new physicians if needed. This way admin can create, if necessary, new user with role ‚Physician‘ to the database. Once created they will also receive on their e-mail verification and password.

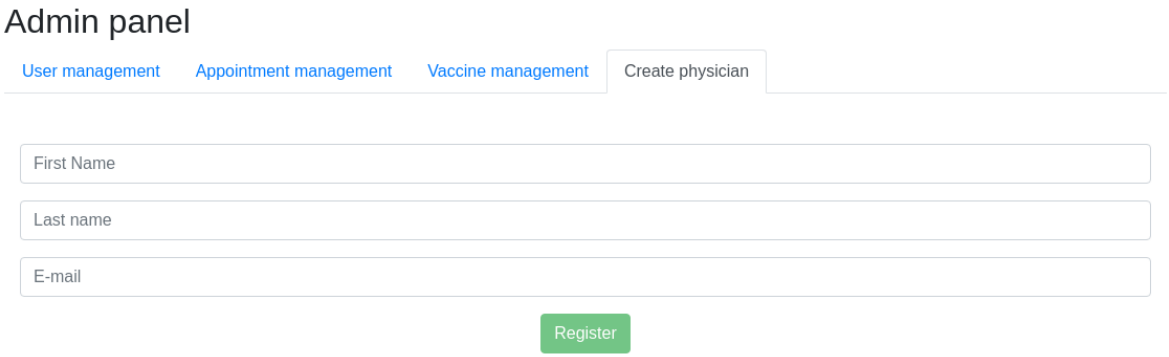


Figure 42: Form of registering a new physician through admins control panel

6.3 Server (Back-end)

6.3.1 Data Transfer Objects (DTO)

To work with data that are being sent from client-side, we used Data transfer object, also known as DTO. Task of DTO is to minimize requests to the server when we are communicating. There are cases, where we need to get complex data, that are made from multiple objects, with that we need to create multiple requests. DTO makes it simple to create one model, which requests will be sorted rather than having multiple models requesting.

```
1 usage
public class UserForLoginDto
{
    1 usage
    public string Email { get; set; }
    1 usage
    public string Password { get; set; }
}
```

Figure 43: Example of DTO for User's login

There are several NuGet packages that are working with DTO, but most popular one is AutoMapper, a lightweight and simple to use library for mapping DTO models in C#. To create mapping you need two things:

- Source
- Destination

Once you determine these two things, you can create simple model that you can use during development.

```
public class AutoMapperProfile : Profile
{
    public AutoMapperProfile()
    {
        CreateMap<User, UserForListDto>();
        CreateMap<User, UserForDetailedDto>();
        CreateMap<UserForRegisterDto, User>();
        CreateMap<UserForUpdateDto, User>();
        CreateMap<Patient, PatientForListDto>();
        CreateMap<Physician, PhysicianForListDto>();
        CreateMap<AppointmentForCreatingDto, Appointment>().ReverseMap();
        CreateMap<Appointment, AppointmentPatientForListDto>();
        CreateMap<Appointment, AppointmentPhysicianForListDto>();
    }
}
```

Figure 44: Example of AutoMapper's mapping for DTO

6.3.2 Database

As mentioned before, for creation of the database, author used approach of code-first (see chapter 3.2.3) that makes it simple for development. Each model created in application basically represents a table in the database.

Tables for Users, roles and user roles are created with AspNet library Identity, the rest of the tables that are created are Patients, Physicians and Appointments.

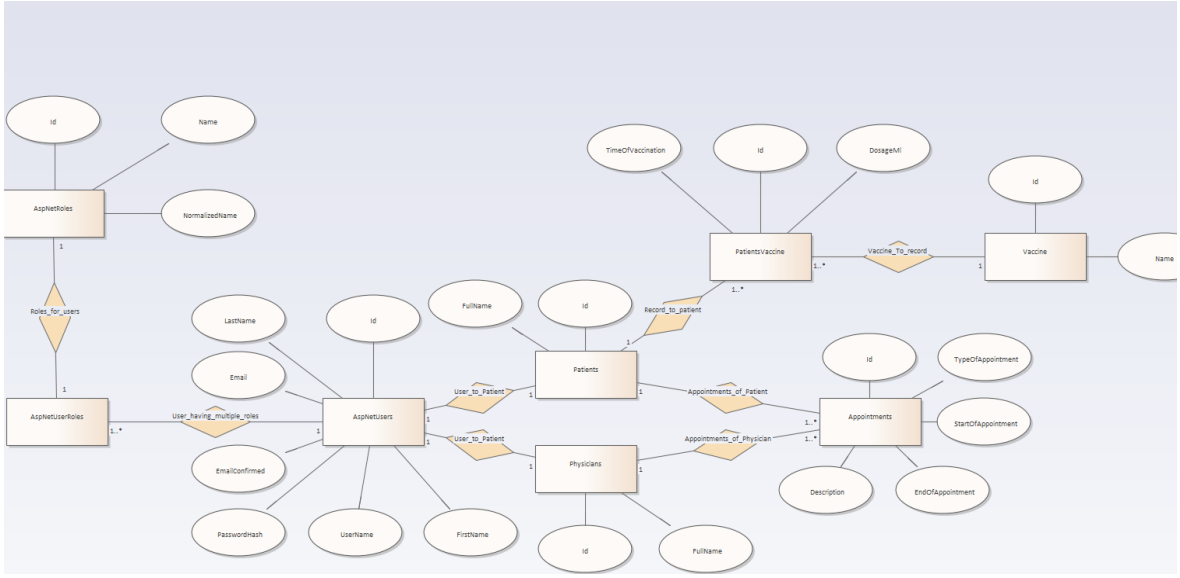


Figure 45: Entity relationship diagram

Table AspNetUsers

This table represents user's account. Generated table contains basic information about user such as username, email, phone number, password in hash form. It can be expanded with more attributes, such as first name and last name.

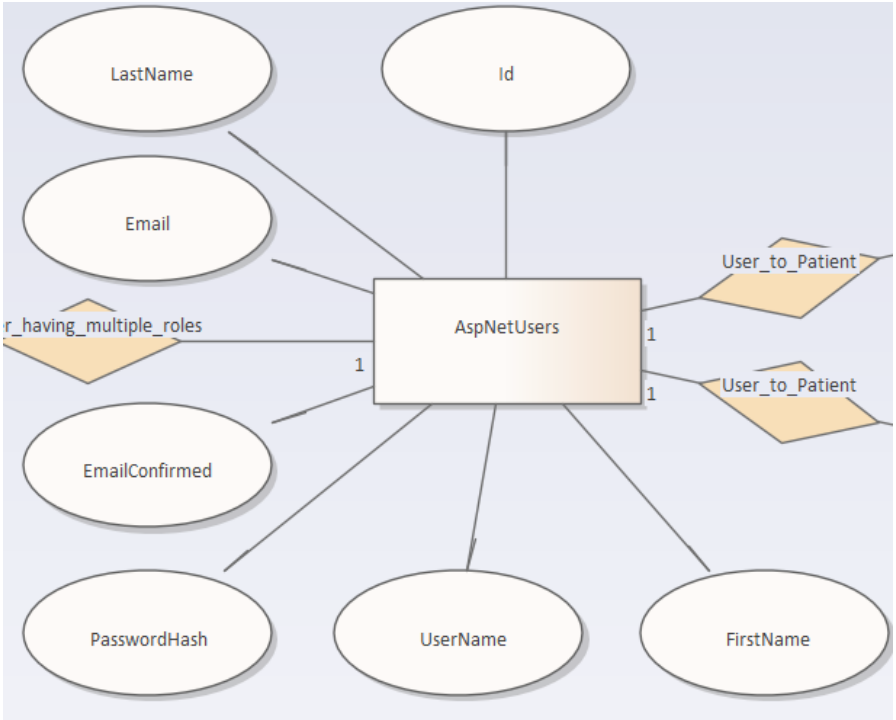


Figure 46: Entity attributes of AspNetUsers

Table AspNetRoles

This table contains roles that will be available for the user.

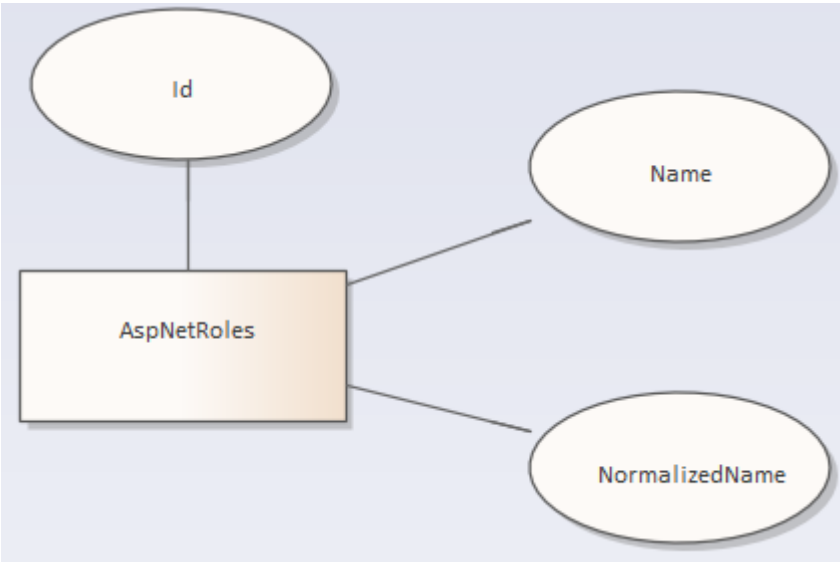


Figure 47: Entity attributes of AspNetRoles

Table AspNetUserRoles

This table shows what users have roles from table AspNetRoles.

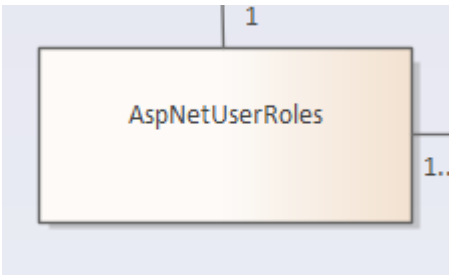


Figure 48: Entity attributes of AspNetUserRoles

Table Physicians

This table is to keep record of users that have role ‚Physician' in AspNetUserRoles.

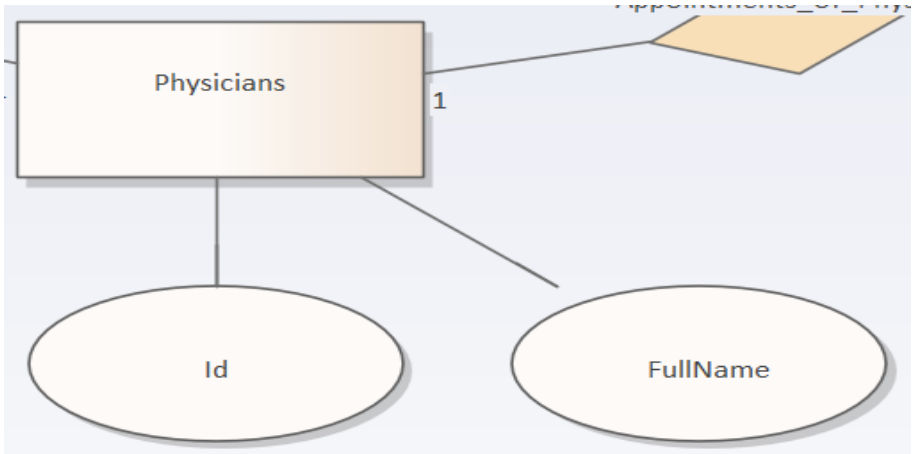


Figure 49: Entity attributes of Physicians

Table Patient

This table is to keep record of users that have role ‚Patient‘ in AspNetUserRoles.

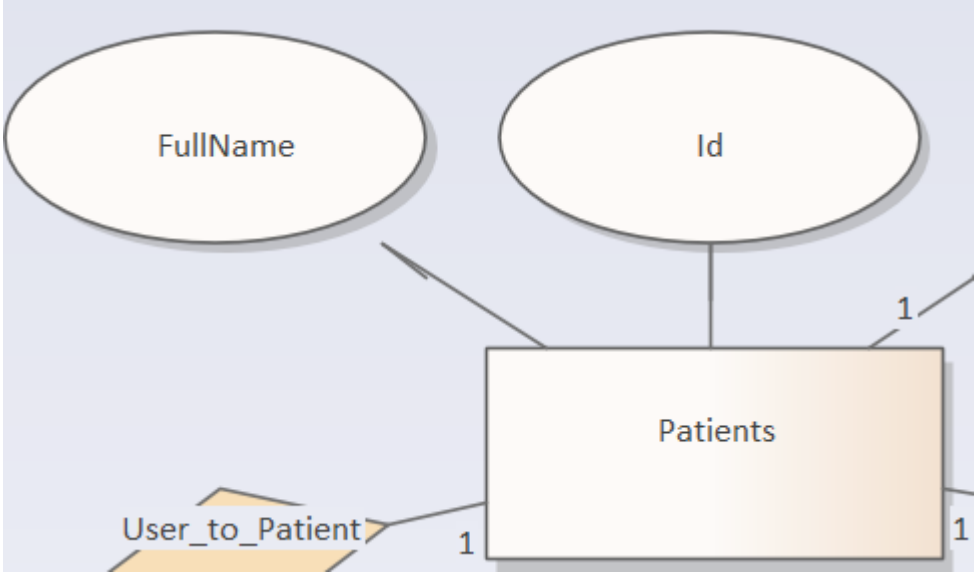


Figure 50: Entitty attributes of Patients

Table Appointments

This table is to keep record between physician and patient.

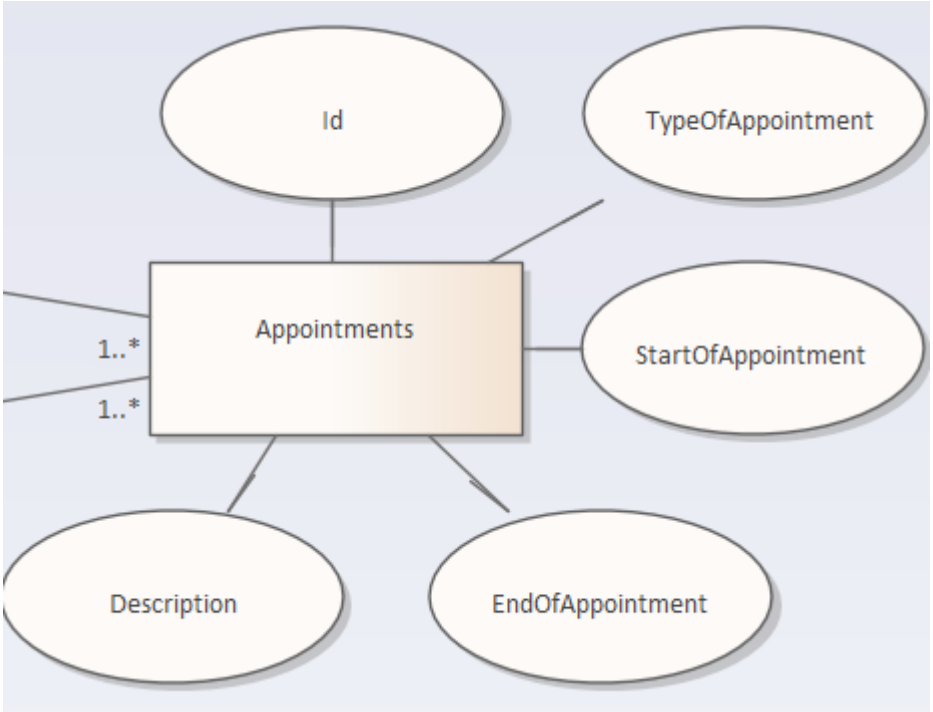


Figure 51: Entitty attributes of Appointments

Table Vaccine

This table is to keep record of vaccines to offer for vaccination.

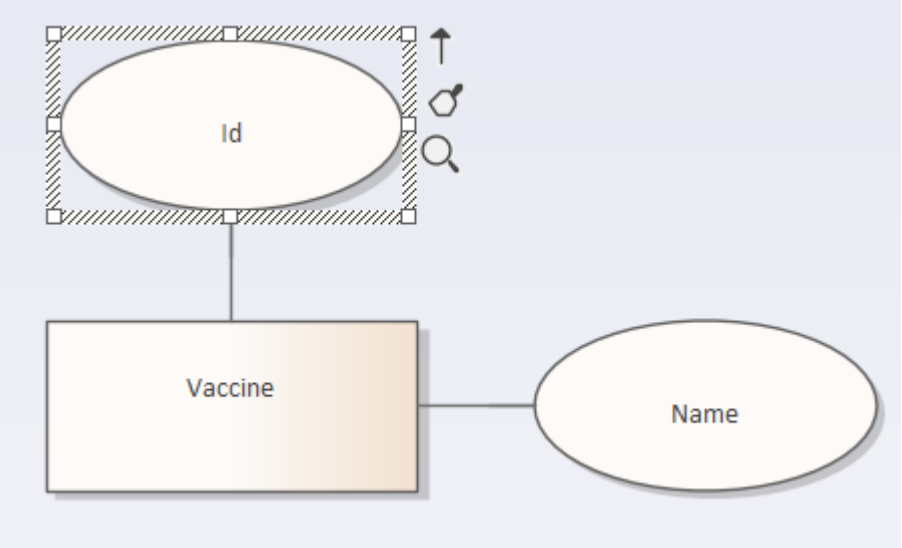


Figure 52: Entitty attributes of Vaccine

Table PatientsVaccine

This table is to keep record of patients history of vaccination.

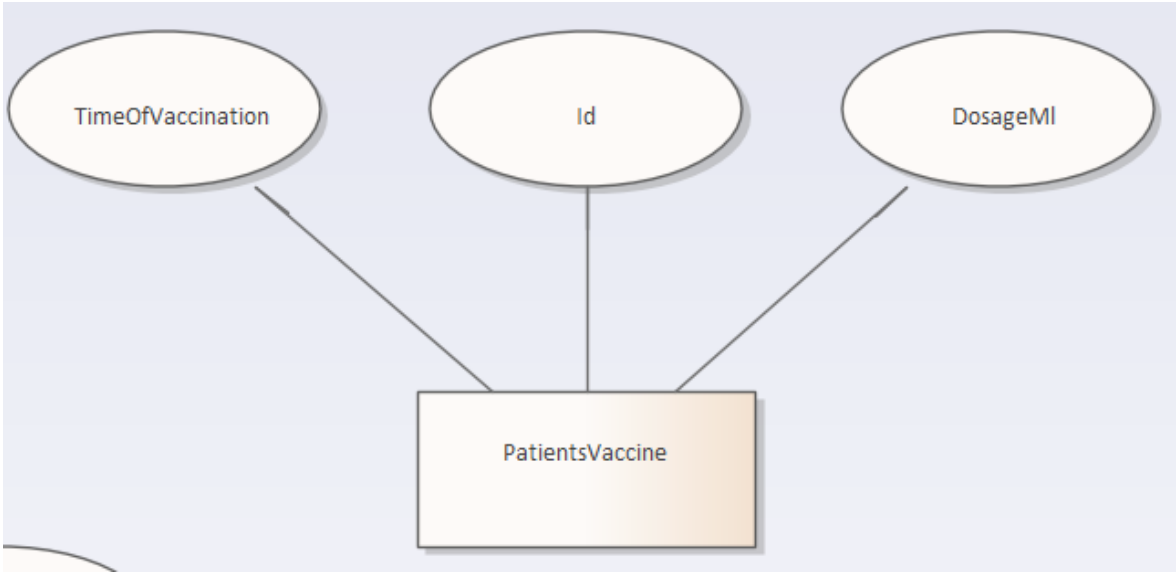


Figure 53: Entitty attributes of PatientsVaccine

7 SECURITY IMPLEMENTATION

Application is protected against Cross-site scripting with implementation Content-Security-Policy. Thanks to that implementation any foreign links will be blocked on site and marked as violation of Content Security Policy.

```
public async Task<IEnumerable<Appointment>> GetPatientsAppointments(int id)
{
    var patientAppointments :List<Appointment> =
        await _context.Appointments //DbSet<Appointment>
                    .Where(p :Appointment => p.PatientFKId == id && p.StartOfAppointment > DateTime.Now) //IQueryable<Appointment>
                    .OrderByDescending(d :Appointment => d.StartOfAppointment) //IOrderedQueryable<Appointment>
                    .ToListAsync(); //Task<List<...>>

    return patientAppointments;
}
```

Figure 54: Example of using LINQ to get data from database

Protection against SQL Injection does not need any special protection method since we are communicating with the database directly. Against SQL Injection we have Language Integrated Query (LINQ) that helps us to send data to database as SQL parameters, that prevents attack.

To prevent attack against CSRF application is using JWT (JSON web token) rather than cookie. With this JWT is in a way a ‚cookie‘ that allows user to be identified. CSRF targets mainly cookies but JWT is placed in ‚Authorization‘ header that allows user to sign in and use it during usage of web application.

eyJhbGciOiJIUzI1NiIsInR5
cCI6IkpXVCJ9.eyJzdWIiOiI
xMjM0NTY3ODkwIiwibmFtZSI
6IkpvaG4gRG9lIiwiaWF0Ijo
xNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6y
JV_adQssw5c

Figure 55: Example of JWT token

When user creates account, his password is being hashed in application to protect his data when database is being exposed. This is due to AspNet Core’s class *UserManager* that is creating new account to the table of users. Thus passwords are unreadable by an average user.

CONCLUSION

Realization of this bachelor thesis was accomplished by creating simple web application that allows to communicate between patient and physician in form of booking appointment. Application needs to follow guideline of GDPR when it comes to collecting personal data, permission of editing personal data and user's choice of „being forgotten".

In theory part of the thesis, author explained what data can be collected, existing solutions and technologies that were used for creating web application. Author tried to explain in detail technologies, that will be used, so that the reader can understand after reading.

In practical part of the thesis, author shows requirements for application, use-case model and implementation of the requirements into application.

REFERENCES

[1] ŽŮREK, Jiří. *Praktický průvodce GDPR*. Olomouc: ANAG, [2017], 223 s. Právo. ISBN 9788075540973.

[2] ČESKO. sdělení č. 115/2001 Sb. m. s., *Ministerstva zahraničních věcí o přijetí Úmluvy o ochraně osob se zřetelem na automatizované zpracování osobních dat.* Zákony pro lidi.cz [online]. c AION CS 2010-2020 [cit. 24. 7. 2020]. Dostupné z: https://www.zakonyprolidi.cz/ms/2001-115

[3] KOUBA, *Tomáš. Jak na GDPR na webu – praktický návod* . Net Magnet [online] [cit. 2020-07-25]. Dostupné z: https://www.netmagnet.cz/blog/gdpr/

[4] CHAN, Rosalie. *The Cambridge Analytica whistleblower explains how the firm used Facebook data to sway elections. Business Insider* [online]. 5 Oct 2019, , 1 [cit. 2020-07-24]. Dostupné z: https://www.businessinsider.com/cambridge-analytica-whistleblower-christopher-wylie-facebook-data-2019-10

[5] SNIDER, Mike. *Google sets April 2 closing date for Google+, download your photos and content before then*. USA Today [online]. 14 Dec 2019 [cit. 2020-07-24]. Dostupné z: https://eu.usatoday.com/story/tech/talkingtech/2019/02/01/google-close-google-social-network-april-2/2741657002/

[6] OSTERLOH, Rick. *Helping more people with wearables*: Google to acquire Fitbit. *Google* [online]. 1 Nov 2019 [cit. 2020-07-24].

[7] K. Paul, *`Tossed my Fitbit in the trash´: users fear for privacy after Google buys company*, The Guardian, lis. 06, 2019.

[8] *Tim Berners-Lee created the first website*. Web Design Museum [online]. [cit. 2020-07-27]. Dostupné z: https://www.webdesignmuseum.org/web-design-history/tim-berners-lee-created-the-first-website-1991

[9] *Average web page data analyzing 8 million websites*. CSS tricks [online]. [cit. 2020-07-27]. Dostupné z: https://css-tricks.com/average-web-page-data-analyzing-8-million-websites/

[10] *Top Frameworks from entire Internet*. Similar Tech [online]. [cit. 2020-07-27]. Dostupné z: https://www.similartech.com/categories/framework

[11] *CSS example of BEM method. Get BEM* [online]. [cit. 2020-07-27]. Dostupné z: http://getbem.com/naming/

[12] *What is non-blocking or asynchronous I/O in Node.js?* [online]. In: . StackOverflow [cit. 2020-07-27]. Dostupné z: https://stackoverflow.com/a/10570261/7574597

[13] ROTH, Issac. *What Makes Node.js Faster Than Java?* [online]. Strongloop by IBM, 30th Jan 2014 [cit. 2020-07-27]. Dostupné z: https://strongloop.com/strongblog/node-js-is-faster-than-java/

[14] TAHER, Rainan. *Hands-On Object-Oriented Programming with C#*. I. edition. Packt Publishing, 2019. ISBN 978-1-78829-622-9.

[15] DE SANCTIS, Valerio. *ASP.NET Core 3 and Angular 9*. III. edition. Packt Publishing, 2020. ISBN 978-1-78961-216-5.

[16] *What is SQL Injection?* [online]. Port Swigger [cit. 2020-07-29]. Dostupné z: https://portswigger.net/web-security/sql-injection

[17] POLLACK, Ed. *SQL Injection: Detection and prevention* [online]. SQLShack, 30th August 2019 [cit. 2020-07-29]. Dostupné z: https://www.sqlshack.com/sql-injection-detection-and-prevention/

[18] KALLIN, Jakob a Irene Lobo VALBUENA. *What is XSS?* [online]. Excess XSS [cit. 2020-07-29]. Dostupné z: https://excess-xss.com/#:~:text=Cross%2Dsite%20scripting%20(XSS),JavaScript%20in%20another%20user's%20browser.&text=Instead%2C%20he%20exploits%20a%20vulnerability,the%20malicious%20JavaScript%20for%20him.

[19] EATI, Prasanthi. 1*0 Most Common Web Security Vulnerabilities* [online]. Guru99 [cit. 2020-07-29]. Dostupné z: https://www.guru99.com/web-security-vulnerabilities.html

[20] BLAZQUEZ, Daniel. *What is Broken authentication and session management?* [online]. HDVI Security, 19th May 2020 [cit. 2020-07-29]. Dostupné z: https://hdivsecurity.com/owasp-broken-authentication-and-session-management

[21] *Cross Site Request Forgery (CSRF)* [online]. [cit. 2020-07-29]. Dostupné z: https://owasp.org/www-community/attacks/csrf

LIST OF ABBREVIATIONS

GDPR	General Data Protection Regulation
HTML	HyperText Markup Language
CSS	Cascade Style Sheet
REST	Representational state transfer
API	Application programming interface
HTTP	HyperText Transfer Protocol
BEM	Block-Element-Modifier
DOM	Document Object Model
WPF	Windows Presentation Foundation
UWP	Universal Windows Platform
CLI	Common Language Interface
LINQ	Language Integrated Query
IL	Intermediate Language
CLR	Common Language Runtime
EF	Entity Framework
GUI	Graphical User Interface
SQL	Stuctured Query Language
IDE	Integrated Developing Enviroment
XML	Extensible Markup Language
SCM	Source-Control Management
SQLi	SQL Injection
XSS	Cross-site scriptin
URL	Uniform resource locator
CSRF	Cross-site request forgery

LIST OF FIGURES

LIST OF TABLES

APPENDICES

Appendix P 1: CD Content

APPENDIX P 1: CD CONTENT

Content that CD has:

- Text
  - bachelor thesis in format *.odt
  - bachelor thesis in format *.pdf
- Source code
  - med-app.zip – source code of the thesis
  - thesis.eapx – UML