BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

DENTAL CHART FOR DENTAL CLINIC

ZUBNÍ KŘÍŽ PRO STOMATOLOGICKOU AMBULANCI

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **DOMINIK KALÁB**
AUTOR PRÁCE

SUPERVISOR **Ing. VÍTĚZSLAV BERAN, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2021

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2020/2021

Bachelor's Thesis Specification

Student: **Kaláb Dominik**
Programme: Information Technology
Title: **Dental Chart for Dental Clinic**
Category: User Interfaces

Assignment:

1. Get acquainted with the processes of the dental clinic. Get acquainted with available dental information systems and evaluate their advantages and disadvantages. Study the possibilities for creating modern web applications.
2. Design your own system for the management of a dental clinic with a focus on interactive records of patient treatment (so-called dental chart).
3. Implement the proposed application using selected technology for creating modern web applications.
4. Test the proposed application with an expert on the topic and get feedback.
5. Present the key features of the solution in the form of a poster and a short video.

Recommended literature:

- Joel Marsh. *UX pro začátečníky*. Zoner Press, 2019.
- Steve Krug. *Don't make me think, revisited: a common sense approach to web usability*. San Francisco: New Riders, ISBN 978-0321965516.
- Mazánek Jiří a kolektiv. *Zubní lékařství - Propedeutika.*

Requirements for the first semester:

- Items 1., 2. and partially 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Beran Vítězslav, Ing., Ph.D.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2020
Submission deadline: May 12, 2021
Approval date: May 7, 2021

Abstract

This work aims to design and develop a prototype web application that offers a suitable environment for stomatological records keeping, with the main focus being the interactive dental chart. This prototype is based on a study of user experience, modern web application development, basic stomatology, an analysis of existing solutions and information gained by consultations with a professional dentist. My design focuses on achieving often required user goals in the fewest possible steps and displaying relevant information. It was inspired by a user interface design of editor applications. The prototype was developed using modern web technologies. It was then tested by a professional dentist, the testing results were noted, and the further development of this product was suggested on their basis.

Abstrakt

Cílem této práce je navrhnout a vyvinout prototyp webové aplikace, která bude nabízet stomatologům praktické prostředí pro evidenci stomatologické dokumentace se speciálním zaměřením na interaktivní rozhraní zubního kříže. Tento prototyp je založen na studiu user experience, technologií pro tvorbu moderních webových aplikací, základů stomatologie, analýze existujících řešení a na informacích získaných konzultací s profesionálním stomatologem. Můj návrh je zaměřen na řešení často prováděných úkonů minimu krocích, zobrazování relevantních informací. Návrh byl inspirován uživatelskými rozhraními editorových aplikací. Prototyp je vyvinut za použití moderních webových technologií. Prototyp byl otestován profesionálním stomatologem, výsledky testů zaznamenány a na jejich základě byl navržen směr dalšího vývoje této aplikace.

Keywords

user experience, user interface, user testing, information system, web application, stomatology, dental chart, dental clinic, tooth localisation, medical documentation

Klíčová slova

uživatelský prožitek, uživatelská rozhraní, uživatelské testování, webová aplikace, informační systém, stomatologie, zubní kříž, stomatologická ambulance, lokalizace zubů, zdravotní dokumentace

Reference

KALÁB, Dominik. *Dental Chart for Dental Clinic*. Brno, 2021. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vítězslav Beran, Ph.D.

Rozšířený abstrakt

Cílem této práce je navrhnout a vyvinout prototyp inovativní a moderní webové aplikace, která bude nabízet stomatologům praktické prostředí pro evidenci stomatologické dokumentace se speciálním zaměřením na interaktivní rozhraní zubního kříže. Toto rozhraní obsahuje grafickou reprezentaci chrupu pacienta, které vykresluje stav pacientova chrupu na základě jeho dokumentace. Stomatolog s ním interaguje a touto interakcí vytváří nové záznamy.

Tento prototyp je založen na průzkumu user experience designu, technologií pro tvorbu moderních webových aplikací a základů stomatologie. Tyto znalosti jsem získal studiem relevantních knih a webových článků. V této práci detailně zkoumám práci stomatologů, nejdůležitější procesy stomatologických ambulancí a své závěry konzultuji s profesionálním zubařem MUDr. Tomášem Machačem. Moje studium také zahrnuje průzkum současně dostupných řešení, hodnocení jejich zubních křížů a jejich uzpůsobení vůči identifikovaným procesům a pacientským potřebám. Konkrétně jsem podrobil bližšímu průzkumu systémy PC DENT a XDENT.

Můj prototyp je zaměřen na řešení často prováděných úkonů v minimu krocích, zobrazování relevantních informací a jejich vhodné grafické zobrazení. Návrh byl inspirován uživatelskými rozhraními editorových aplikací, jako například Photoshop či Inkscape. Moje rozhraní tedy obsahuje seznam nástrojů, kterými může uživatel upravovat „plátno“, v tomto případě samotný nákres pacientova chrupu. Prototyp je vyvinut za použití moderních webových technologií. Databáze řešení je PostgreSQL, vytvořená code-first přístupem pomocí objektově relačního mapování v technologii EntityFramework. Aplikačně programovací rozhraní (API) je vyvinuto v nejnovější verzi .NET (konkrétně verze 5). Hlavní frontend aplikace je vyvinut jako jednostránková aplikace v populárním javascriptovém aplikačním rámci Vue.js.

Poté byl prototyp otestován profesionálním stomatologem. V tomto testování bylo odhaleno několik nedostatků v designu, které způsobovaly zhoršenou orientaci v systému. Tyto chyby jsem analyzoval a navrhl jejich řešení. Celkově byl však prototyp přijat převážně pozitivně. Ohodnocen byl šesti body z deseti, což je pro dříve netestovaný prototyp aplikace kladné hodnocení, obzvláště vzhledem k triviálnosti řešení většiny identifikovaných nedostatků. Stomatolog uznal, že na současném trhu nemá tento přístup obdoby, je validní a teoreticky efektivní. Na základě identifikovaných nedostatků a stavu řešení jsem navrhl směr dalšího vývoje této aplikace. Po odstranění těchto nedostatků může být tato aplikace podrobena podrobnějšímu testování v praxi a poté integrována jako součást již existujících systémů, či jako základ pro zcela nový systém.

Dental Chart for Dental Clinic

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Vítězslav Beran, Ph.D. The supplementary information was provided by Mr. MUDr. Tomáš Machač. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Dominik Kaláb
May 10, 2021

Acknowledgements

I would like to thank my supervisor Ing. Vítězslav Beran, Ph.D., for his professional guidance, words of encouragement and his time provided in consultations. I would also like to thank MUDr. Tomáš Machač for his professional knowledge, constructive criticism and his time in consultations and user testing. Great thanks also belongs to my fiancée Klára Pisarčíková for help with editing this work and kind words of emotional support whenever a load of stress while producing this work seemed to be unbearable.

Contents

Chapter 1

Introduction

Dentist ambulances constitute roughly 30 % of all outpatient care in the Czech Republic [14]. Therefore, it is only logical that the market with information systems and electronic solutions for dental ambulances is heavily developed. These solutions need to suit legislation, implement a broad spectrum of functionalities, and accommodate institutional nuances. Therefore, they often take years to develop, are colossal, rigid, and expansive. These systems are in many cases developed using long-outdated technologies, and their application in modern ambulances is problematic.

This work aims to design and develop a prototype of an innovative and modern system of recording medical documentation in dental ambulances, with a particular focus on the interactive environment of the dental chart. I will closely observe a dentist's work, analyse the processes that are essential to a dental ambulance, and present the outcomes of my discussions about these matters with a dentist. For the sake of simplicity, I will concentrate on practical dentistry and will not take into consideration the wide range of specialised ambulances like dental surgery. My main focus will be the core process of dentists‘ appointments: examining and treating patients and recording these actions into the dental system via an interactive dental chart.

Firstly, in chapter 2, I will present the information accumulated about the functioning of dental clinics. Based on my understanding of these processes, I will evaluate the efforts of other dental systems to accommodate these processes and the general state of dental systems in the Czech republic. I will also present a short summary of user experience design and web application development, on which I grounded this thesis. Then, in chapter 3, I will further identify the requirements for this solution, followed by my solution proposal. This solution will be based on an interactive dental chart that will allow a convenient record creation and provide a visual representation of a patient's denture state. Afterwards, I will present the final solution's design, the changes made to the initial design, and the findings that lead to those changes. Chapter 4 will describe the development of the prototype, the technologies used, and some implementation details. This solution will be web-based, created with modern technologies and will be expandable. The prototype will be then tested by and discussed with a professional dentist, the results noted, analysed, and a solution to the identified issues will be purposed. Chapter 5 will be a conclusion of my work. I will summarise my efforts and suggest how my work could be continued and expanded.

Chapter 2

Theoretical Knowledge for Dental Systems Development

In this chapter, I will briefly summarise basic theoretical information from a few areas of interest. Firstly, I will present a short introduction to dentistry. Then, I will introduce some basic UX design theory. Lastly, I will outline information about today's standards of web application development.

2.1 A Short Introduction to Dentistry

This section will summarise fundamental theoretical information about how dentists and dental ambulances operate. It will present information on which I based the rest of this work, but it does by no means provide a complete knowledge base for this work nor dentistry as a whole. The whole section is based on information acquired in "Stomatologie Jan Ležovič a kolektiv"[14], "Zubní lékařství – Propedeutika"[15] and in my consultations with MUDr. Tomáš Machač.

Dentist's Work and Record-keeping

The main focus of practical dentists is to prevent, diagnose and treat dental issues. Most patients have one practical dentist who regularly checks the state of their denture and either treats issues or refers the patient to a specialist for a treatment. Therefore, dentists have to keep complete records of patients‘ denture history. These records are:

- diagnoses like teeth decay or inflammation and
- procedures to which the patient was treated, such as tooth extraction.

To treat patients correctly, dentists also have to keep a summary of patients‘ medical history known as an anamnesis. This anamnesis usually contains:

- personal anamnesis – important diseases and medical events in the patient's life,
- allergies – an overview of patient's allergies and his allergic reactions
- pharmacological anamnesis – an overview of medicaments used by a patient and

- toxicological anamnesis – a summary of patients drug usage (mainly smoking and alcohol).

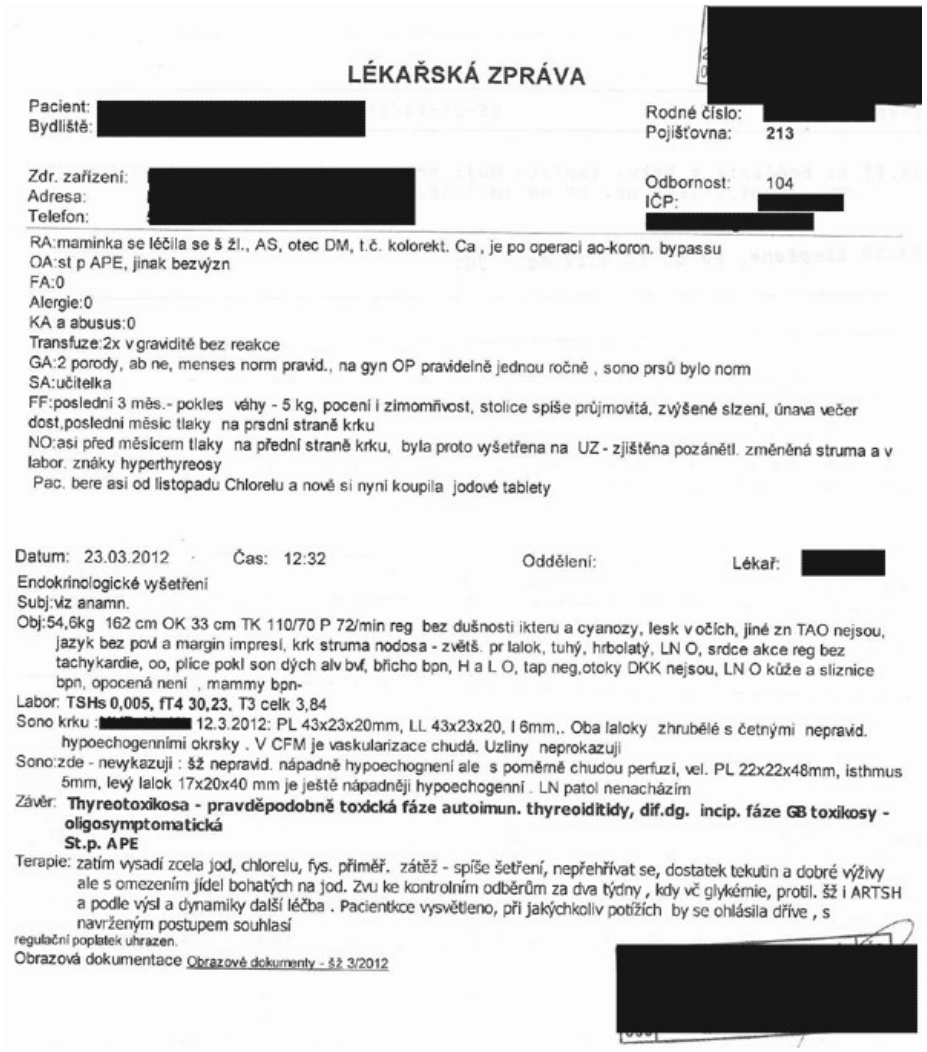
The anamnesis can also contain information about the patient's family members' medical history, their living conditions, or work environment if deemed important by the dentist. Dentists also keep basic personal information about patients, such as:

- personal identification number,
- name,
- address,
- contact information like email address and phone number and
- health insurance information.

Each dental appointment results in a medical report (as illustrated by 2.1). This medical report contains:

- transcription of the patient's subjective feelings,
- diagnoses,
- procedures,
- prescribed medication,
- recommended therapy or referrals to other specialists and
- other information deemed important by the dentist.

This medical report is then unchangeable and is a definitive transcript of the appointment.



LÉKAŘSKÁ ZPRÁVA

Pacient: [redacted]
Bydliště: [redacted]
Rodné číslo: [redacted]
Pojišťovna: 213

Zdr. zařízení: [redacted]
Adresa: [redacted]
Telefon: [redacted]
Odbornost: 104
IČP: [redacted]

RA:maminka se léčila se š žl., AS, otec DM, t.č. kolorekt. Ca , je po operaci ao-koron. bypassu
OA:st p APE, jinak bezvýzn
FA:0
Alergie:0
KA a abusus:0
Transfuze:2x v graviditě bez reakce
GA:2 porody, ab ne, menses norm pravid., na gyn OP pravidelně jednou ročně , sono prsů bylo norm
SA:učitelka
FF:poslední 3 měs.- pokles váhy - 5 kg, pocení i zimomřivost, stolice spíše průjmovitá, zvýšené slzení, únava večer dost,poslední měsíc tlaky na přední straně krku
NO:asi před měsícem tlaky na přední straně krku, byla proto vyšetřena na UZ - zjištěna pozánětl. změněná struma a v labor. známky hyperthyreosy
Pac. bere asi od listopadu Chlorelu a nově si nyní koupila jodové tablety

Datum: 23.03.2012 Čas: 12:32 Oddělení: Lékař: [redacted]
Endokrinologické vyšetření
Subj:viz anamn.
Obj:54,6kg 162 cm OK 33 cm TK 110/70 P 72/min reg bez dušnosti ikteru a cyanozy, lesk v očích, jiné zn TAO nejsou, jazyk bez povl a margin impresí, krk struma nodosa - zvětš. pr lalok, tuhý, hrbolatý, LN O, srdce akce reg bez tachykardie, oo, plíce pokl son dých alv bvf, břicho bpn, H a L O, tap neg,otoky DKK nejsou, LN O kůže a sliznice bpn, opocená není , mammy bpn-
Labor: **TSHs 0,005, fT4 30,23.** T3 celk 3,84
Sono krku : [redacted] 12.3.2012: PL 43x23x20mm, LL 43x23x20, I 6mm,. Oba laloky zhrubělé s četnými nepravid. hypoechogenními okrsky . V CFM je vaskularizace chudá. Uzliny neprokazují
Sono:zde - nevykazují : šž nepravid. nápadně hypoechogenní ale s poměrně chudou perfuzí, vel. PL 22x22x48mm, isthmus 5mm, levý lalok 17x20x40 mm je ještě nápadněji hypoechogenní . LN patol nenacházím
Závěr: **Thyreotoxikosa - pravděpodobně toxická fáze autoimun. thyreoiditidy, dif.dg. incip. fáze GB toxikosy - oligosymptomatická**
St.p. APE
Terapie: zatím vysadí zcela jod, chlorelu, fys. přiměř. zátěž - spíše šetření, nepřehřívat se, dostatek tekutin a dobré výživy ale s omezením jídel bohatých na jod. Zvu ke kontrolním odběrům za dva týdny , kdy vč glykémie, protil. šž i ARTSH a podle výsl a dynamiky další léčba . Pacientkce vysvětleno, při jakýchkoliv potížích by se ohlásila dříve , s navrženým postupem souhlasí
regulační poplatek uhrazen.
Obrazová dokumentace Obrazové dokumenty - šž 3/2012

[redacted]

Figure 2.1: Anonymised example of a czech medical report.[1]

Standard Dental Appointment

In this section, I will describe a standard course of a dental appointment. Understanding this process is vitally important for designing a convenient and intuitive system.

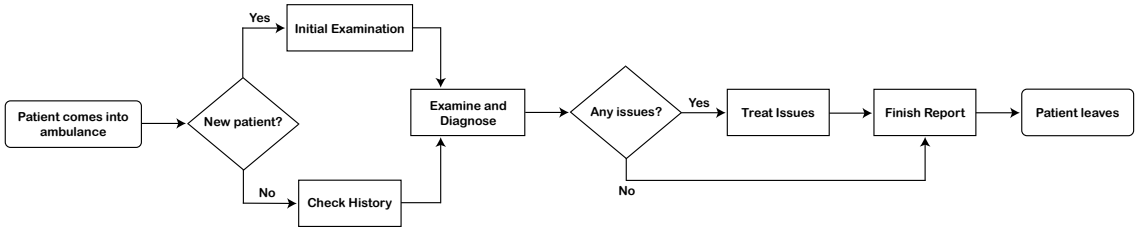


Figure 2.2: Flowchart of a typical dentist appointment.

Initial Examination

If the dentist has never examined the patient, there are a few crucial steps to be done before the examination can begin. At first, the dentist has to fill out the patient's personal information. Then, the dentist records the patient's anamnesis based on information provided by the patient. Afterwards, the dentist records the state of the patient's denture.

Check History

Suppose the patient has already been recorded in the system before. In that case, the dentist quickly checks the history of the patient's denture to get the context needed to properly examine the patient and spot potential changes.

Examine and Diagnose

The patient is seated and the examination can begin. With the use of their diagnosing tools, the dentist examines the denture from the first to the fourth quadrant (more about teeth localisation in 2.1). When an issue is found, the dentist reports it to a nurse who then immediately records it.

Treat Diagnoses

The dentist then treats the issues diagnosed in the previous step. After the treatment is finished, they record all the procedures they have done.

Finish Report

The dentist writes the report. They record the diagnoses and procedures. They also mention essential information about the appointment. They suggest therapy, including medication, and refer the patient to other specialists if needed. The report is then printed out for the patient if the patient wishes so.

Tooth Localisation

Diagnoses and procedures are often localised. For instance, in order to record a tooth decay, the dentist should mention the tooth and, if possible, also the surface on which the decay is located.

In the Czech Republic, the most widely used teeth localisation notation is the FDI notation. Other notations exist, such as Zsigmondy and Palmer notation[1] (widely spread in the UK) or ADA notation[2] (used in the USA). However, these are not commonly used among dentists in the Czech Republic, so I will not cover them in this thesis.

[1] More information about the Palmer notation can be found on Wikipedia.
[2] More information about the ADA notation can be found on Wikipedia.

The FDI notation is described as follows: The whole denture is split into four quadrants which are numbered clockwise. The numbering begins in the upper right quadrant and ends in the lower right quadrant (from the view of the patient). Numbers from 1 to 4 represent permanent denture quadrants, while numbers from 5 to 8 represent deciduous teeth (milk teeth) quadrants [15]. The quadrant numbering is indicated in the table 2.1.

	Permanent	Deciduous
Upper-Right	1	5
Upper-Left	2	6
Lower-Left	3	7
Lower-Right	4	8

Table 2.1: Teeth quadrant numbering.

Teeth numbering for each quadrant begins from the middle of the denture to the sides from 1 to 8 for permanent teeth and from 1 to 5 for deciduous teeth as shown in the picture 2.3.

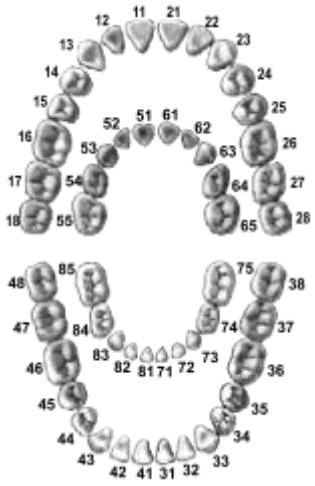


Figure 2.3: FDI notation.[12]

Each tooth is further divided into tooth surfaces, which are categorised by their relative position in the patient's mouth. These surfaces are:

- labial – the surface facing the lips, specific for teeth 1, 2, 3,
- distal – the surface facing away from the midline of the face,
- buccal – the surface facing the cheeks, specific for teeth 4, 5, 6, 7, 8,
- incisal – the biting edge of an anterior tooth, specific for teeth 1, 2, 3,
- lingual – the surface facing the tongue, specific for quadrants 3, 4 and 7, 8,
- palatal – the surface facing the palate, specific for quadrants 1,2 and 5, 6,
- mesial – the surface closest to the midline of the face and
- occlusal – the chewing surface of posterior teeth, specific for teeth 4, 5, 6, 7, 8.[17]

The division of the surfaces is illustrated in the picture 2.4.

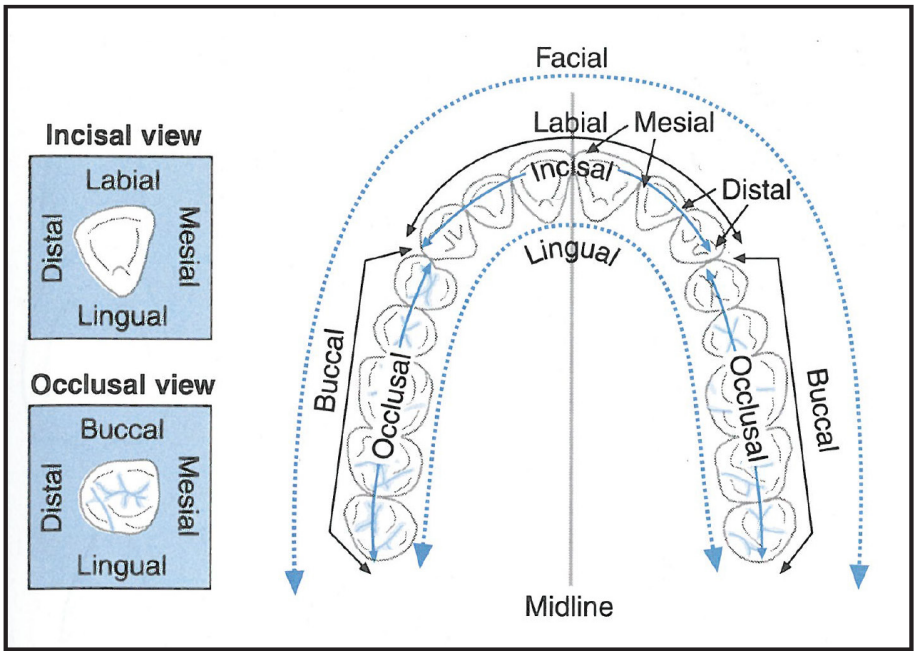


Figure 2.4: Tooth surfaces diagram.[6]

Closer localisation can be achieved by using a combination of these surfaces' names to identify edges between these surfaces, for example, distal-palatal edge.

2.2 State of Dental Systems in the Czech Republic and Evaluation

The General State of Dental Systems in the Czech Republic

Dental systems are systems that help dentists effectively execute their work and manage the ambulance. These systems usually implement sets of these functionalities:

- accounting,
- document management,
- insurance company reports,
- appointment management,
- electronic signature,
- medication administration,
- medical record keeping and
- business intelligence.

Each system contains some subset of these functionalities with variable levels of quality. The list of possible functionalities is fairly extensive. Therefore, dental systems are usually complex and expansive. Development teams have been creating these solutions for numbers of years and have very specific know-how. Many ambulances also use more than one system to cover a broader range of functionalities (for instance, a dental system in conjunction with a standalone accounting system).

Extensive research about the state of dental systems not only in the Czech Republic has been conducted by Bc. Vendula Nováková in her work "The Comparison of Software Systems in Dentists Usage". As a summary of her findings, she wrote: Dental systems contain special records unique to dentists. Some of the dental systems have been developed as dependent systems on existing ambulatory systems. The main representants of this category are solutions of CompuGroup Medical (CGM) Czech Republic s.r.o. (MEDICUS Stomatolog, PC DENT, DENTIST+). Another two Czech providers have focused on dental systems only and have developed standalone systems. Previously mentioned products by a company named CGM and system Stomatolog by a company named HoboSoft Ing. Rubáše are built to be installed directly into a PC on a workplace. System XDENT allows usage without an installation functioning on a cloud-based architecture. For accessing the system, the dentist logs into the web page of the provider. An advantage of this approach is the possibility to work from home or multi-device access. The only required condition is stable internet access.[16]

There is a broad spectrum of systems in the Czech Republic. Dentists have a variable choice of functionality sets, type of implementation and scope. The choice depends on many factors:

- How large is the ambulance? (How many system users does it have? How many patients?)
- Is the ambulance standalone or a part of a hospital? (If it is a part of a larger organisation, it needs integration with other systems.)
- What is the budget?
- Does the dentist have access to the IT department and their own server? (If so, on-premise solutions are a viable choice. Otherwise, a cloud solution is needed.)
- Does the dentist require devices (X-rays and other peripherals) to be integrated?

Choosing the right software is, therefore, a complicated process with many variables. Because of the size of some of these organisations, which may even be state-owned, politics may also affect the decision

General UI Concepts

There are two main approaches to dental system design. In this part, I will analyse them and suggest my own modification.

Classical Records-keeping System

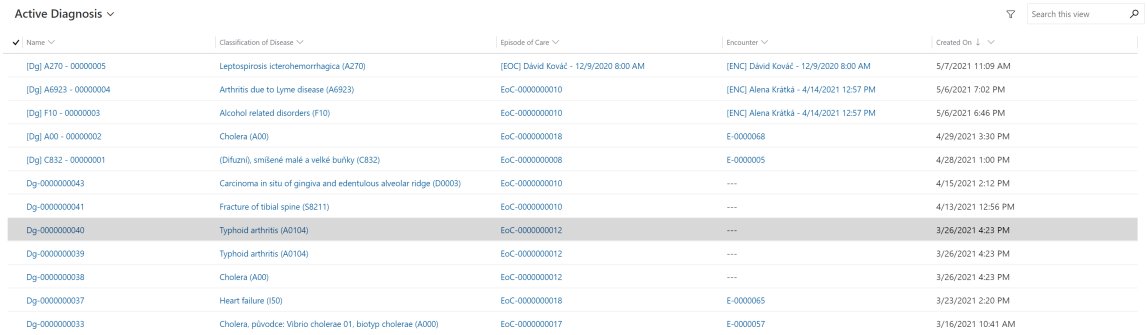
The standard lines and tables style of keeping records without any advanced graphical interface (as illustrated by the picture 2.5). Pros:

- fast to develop
- organised
- usually cheap
- can have high information density

Cons:

- slow convey of information

- limited means of information prioritising
- hard to transfer into a comprehensive overview of reality



Active Diagnosis

Search this view

Name	Classification of Disease	Episode of Care	Encounter	Created On
[Dg] A270 - 00000005	Leptospirosis icterohemorrhagica (A270)	[EOC] Dávid Kovář - 12/9/2020 8:00 AM	[ENC] Dávid Kovář - 12/9/2020 8:00 AM	5/7/2021 11:09 AM
[Dg] A6923 - 00000004	Arthritis due to Lyme disease (A6923)	EoC-0000000010	[ENC] Alena Krátká - 4/14/2021 12:57 PM	5/6/2021 7:02 PM
[Dg] F10 - 00000003	Alcohol related disorders (F10)	EoC-0000000010	[ENC] Alena Krátká - 4/14/2021 12:57 PM	5/6/2021 6:46 PM
[Dg] A00 - 00000002	Cholera (A00)	EoC-0000000018	E-0000068	4/29/2021 3:30 PM
[Dg] C832 - 00000001	(Difuzní), smíšené malé a velké buňky (C832)	EoC-0000000008	E-0000005	4/28/2021 1:00 PM
Dg-0000000043	Carcinoma in situ of gingiva and edentulous alveolar ridge (D0003)	EoC-0000000010	---	4/15/2021 2:12 PM
Dg-0000000041	Fracture of tibial spine (S8211)	EoC-0000000010	---	4/13/2021 12:56 PM
Dg-0000000040	Typhoid arthritis (A0104)	EoC-0000000012	---	3/26/2021 4:23 PM
Dg-0000000039	Typhoid arthritis (A0104)	EoC-0000000012	---	3/26/2021 4:23 PM
Dg-0000000038	Cholera (A00)	EoC-0000000012	---	3/26/2021 4:23 PM
Dg-0000000037	Heart failure (I50)	EoC-0000000018	E-0000065	3/23/2021 2:20 PM
Dg-0000000033	Cholera, původce: Vibrio cholerae 01, biotyp cholerae (A000)	EoC-0000000017	E-0000057	3/16/2021 10:41 AM

Figure 2.5: List of diagnoses in classical records-keeping system.

Needless to say, most dentists using this kind of systems use them primarily because of their lower price or because they have not yet found a suitable system to replace them.

Dental Chart

Most of the quality, dedicated dental systems use some form of a dental chart. It is a graphical representation of the state of the patient's denture. Pros:

- organised if done well
- easy to transfer into a comprehensive overview of reality
- can prioritise information
- fast in conveying information if done well
- can be interactive

Cons:

- hard to design well
- hard to implement well
- each dentist has a different perception of its ideal way of functioning
- has a learning curve (associating visual changes with information takes some time)

Some form of this approach is usually the prefered one by dentists if convenience and effectiveness are the main factors of their system choice. Some systems use a dental chart without the interactivity built-in. Records are recorded by a classical form filling, and the dental chart is used as a visualisation environment only.

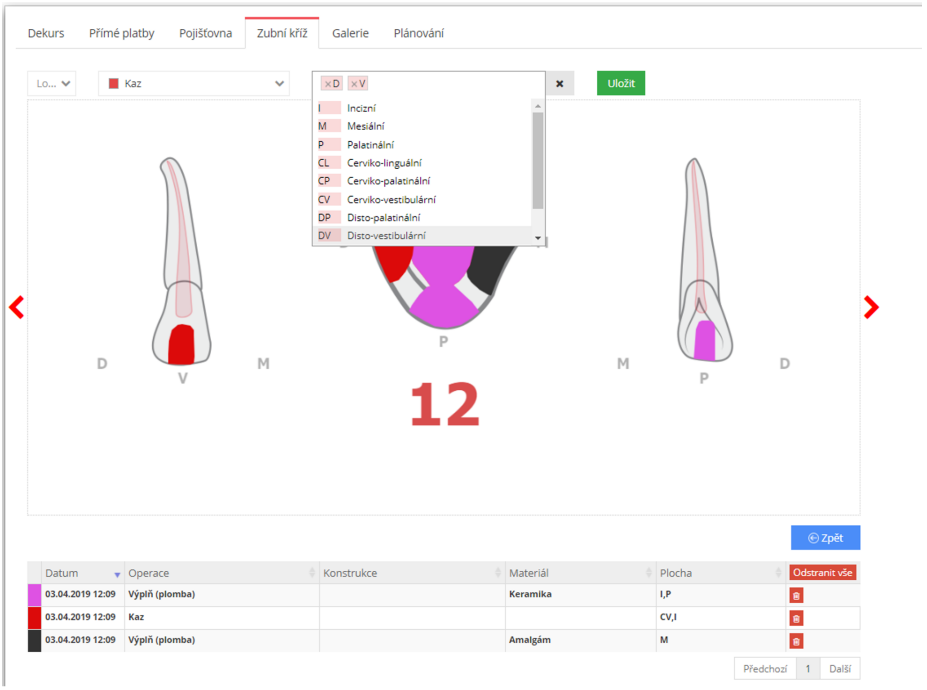


Figure 2.6: Form filling in XDENT.

If interactivity is implemented, selecting the localisation usually prefills the form for the record. Afterwards, the type of record, the classification of disease or the procedure code and other information is submitted by the dentist or the nurse (as a screenshot from figure 2.6 illustrates). This is sufficient enough but not ideal since this kind of solution limits the possibility of prioritisation and does not represent the flow of the appointment. Because the encounter can be divided into three modes of interaction, the interactivity can be optimised. The three modes are:

- information acquiring and dental chart viewing,
- recording the diagnoses – in this stage, the dentist notes all the issues identified in the patient's denture (decay being the most representative one); the dentist also needs to be able to write monitoring notes for future treatment,
- recording the procedures – in this stage, the dentist treats the issues identified prior. The most prominent procedures correspond with the most prominent diagnoses; thus, the most prominent procedures are white and amalgam fillings.

In each of these stages, the dentist is expected to repeat the same actions one after the other. Therefore some way of repeating the same actions on different localisations should be implemented.

Specific Dental Software Evaluation

To represent the possible approaches to developing dental systems, I have chosen to analyse and present two dental systems with very different approaches and system architectures. PC DENT is a desktop on-premise installed (an installation on a server owned by the client)

dental extension of the existing medical system, while XDENT is a cloud-based and purely dental software. In this section, I will examine both of them.

PC DENT

PC DENT[3] is an older software built as a modification of PC DOKTOR, a system for general practitioners. It is developed by CompuGroup Medical. It is a desktop application and allows local and also on-premise installation. The fact that this software is a modification of an already existing one is immediately visible. It presents dentists with a generous amount of not applicable tabs and a wide range of mostly impractical and unusable tools. It is also very robust and hard to navigate. The user manual has 520 pages. Its design is very old school, reminiscent of WinForms. However, the program also offers a wide range of keyboard shortcuts, a notable advantage of desktop applications.

I find the dental chart of PC DENT complicated and hard to navigate (illustrated in the screenshot 2.7). It offers three ways of recording diagnoses and procedures. The chart's interactivity is limited by the technology used.

The pricing system of PC DENT is complicated, containing one time purchases, monthly support payments and purchasable extensions.

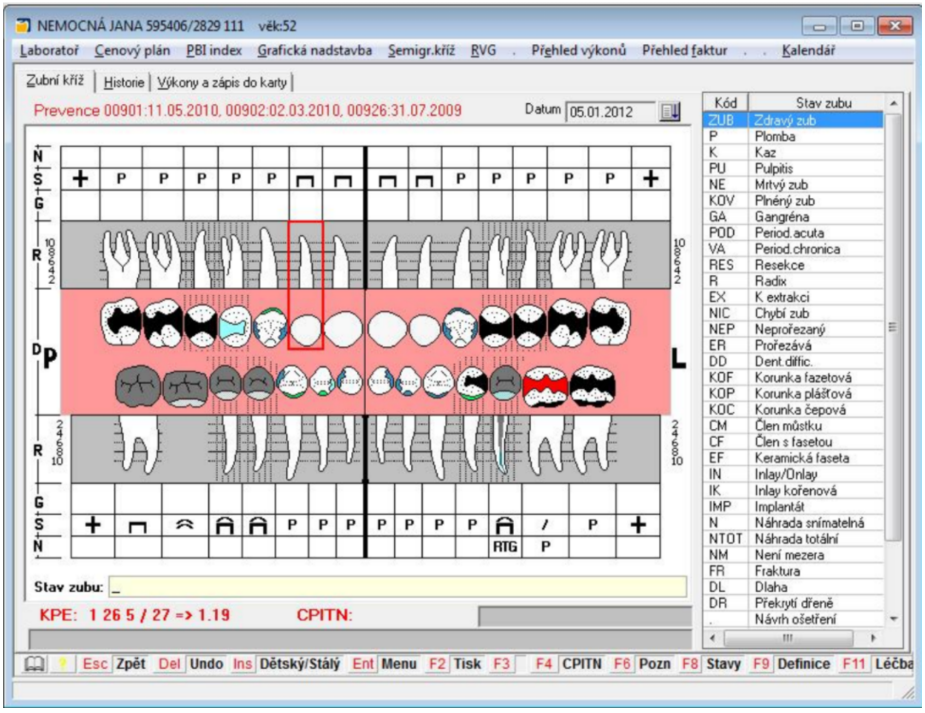


Figure 2.7: PC DENT's dental chart.[3]

In conclusion, PC DENT is a very robust on-premise system. It is complicated, outdated and relatively expensive. Nonetheless, because of its age, it is often installed in older ambulances that yet have not upgraded their systems. It can work well for ambulances that

[3]More information about PC DENT can be found on its official website https://www.pcdent.cz/

are a part of larger medical complexes (an allergologist and a dentist in one hospital can have nearly identical software installed and access the same data, which removes integration problems) with its own server and IT department.

XDENT

XDENT is a relatively new modern software designated purely for dentists. It is a strictly cloud-based web application and offers a wide variety of functions. XDENT provides a well made, interactive calendar for organising patient appointments, a simple patient form with personalised data fields and a timeline with patient's medical reports. XDENT also provides an extremely well-made accounting interface with integrated medical insurance calculations, invoice generations, patient debt tracking and other financial statistics. XDENT also offers advanced medical report generation functions and rich-text editing.

The dental chart of XDENT is anatomical but very simple. To create new records, the dentist has to open the chosen tooth record and only then can he record diagnoses and procedures. This is rather inconvenient. The dental chart has advanced graphical visualisation and good options for dental history viewing.

XDENT claims to be installed by more than 2100 dentists. The basic monthly payment is 1466 CZK (56EUR as for 25.1.2021).

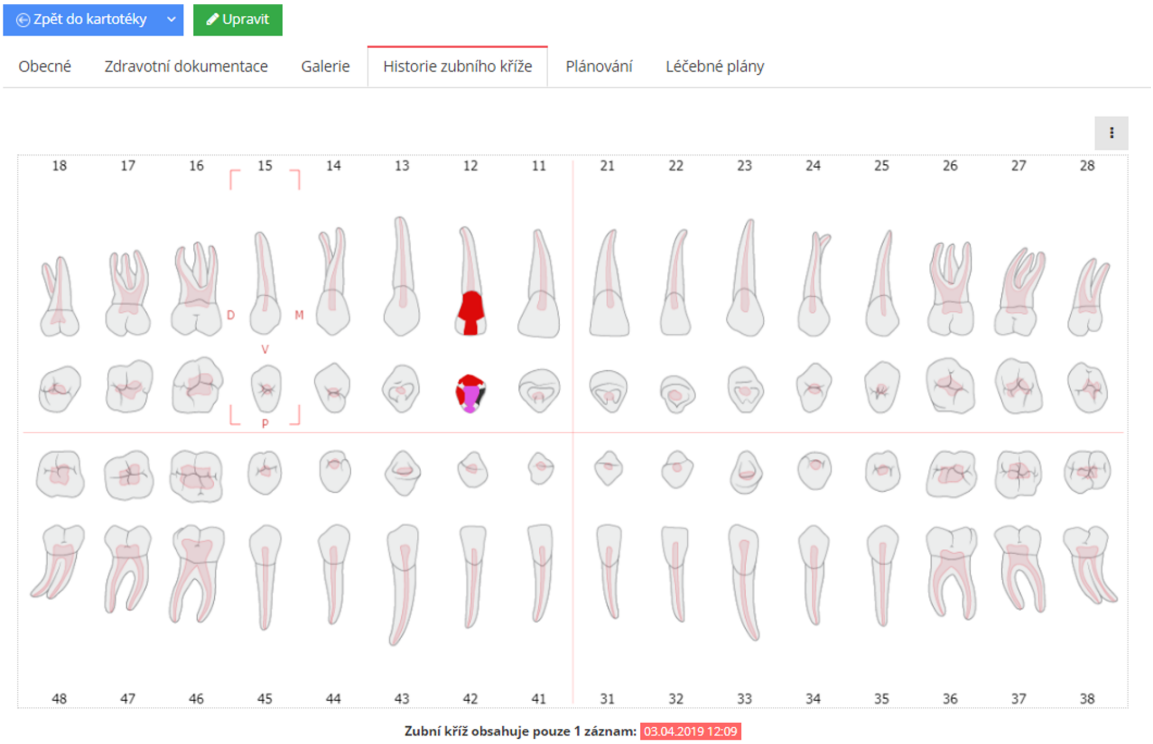


Figure 2.8: XDENT's dental chart.

To conclude, XDENT is an excellent all-around software for standalone dental ambulances with advanced capabilities amplified by a modern interface and a cloud-based service. Con-

sidering its competitive pricing, I find XDENT one of the best available systems in the Czech Republic as for the time I am writing this thesis.

However, its dental chart is, in my opinion, not optimised enough. The process of recording the diagnoses and procedure records does not completely follow the process identified in section 2.1. To record the localised records, the dentist needs to navigate onto the tooth's detail and then record the desired record into the form. The need to open a specific tooth to record a record is, in my opinion, unnecessary and hinders the user. Also, the user opens the tooth record with a specific goal in mind. Therefore, the goal definition can precede the selection of localisation.

2.3 User Experience Design

This section will be a short summary of the principles of UX design.

User experience describes how, when and why people interact with a product or service and how they feel about their experience.[4][4] User experience design involves a dozen specialities such as Interaction Design, Interface Design, Visual Design and Content Management - and, of course, Usability and Information Architecture – all under the UX umbrella.[13]

Good UX design communicates three things:

1. What is this?
   Each part of the design should clearly communicate why it exists and what is its purpose.
2. What's in it for me?
   User should be motivated to use the design. They should be aware of the advantages of using our design.
3. What do I do?
   At every moment of using our system, the user should know the steps to do to achieve their current goal.

High-quality UX design takes into consideration five main ingredients of UX:

1. Psychology – considering the user's complex mind. Asking questions such as:
   - What is the user's motivation to use our system?
   - How does the user feel using our UI?
   - What does the user expect when clicking this button?
2. Usability – analysing if all goals the user could possibly want to achieve are achievable and how effectively. Asking questions such as:
   - Is it clear and direct?
   - Are there any preventable mistakes that can be done by the user?
   - Could the job be done with less input?
3. Design – design in UX is much less artistic than in other design areas. The UX design is about how UX works and is something provable rather than subjective. Questions that should be asked when reviewing UX design:
   - Do users like the design? Do they trust it?
   - Does it communicate its purpose and function?
   - Does it represent the brand?

[4] https://copyhackers.com/2019/11/ux-copywriting/

4. Copywriting – Asking questions such as:
    - Does it motivate the user to complete their goal? Is that the desirable outcome?
    - Does it sound confident and tell the user what to do?
    - Does it reduce anxiety when using our design?
    - Is it clear, direct, simple and functional?
5. Analysis – how to use data to improve the design. When analysing, questions such as these should be asked:
    - Is the data used to prove that the design is correct or to learn the truth?
    - Is the data used to gather subjective opinions or objective facts?
    - Is the data used to look for bad results, too? Why not?
    - Is the analysis based on absolute numbers or relative improvements?
    - How can the analysis be used to make improvements?

Joel Marsh also identifies two key categories of goals that UX should follow:

- User goals are desires of users not to be bothered by the UI and desire to make their lives easier and work more efficiently. User goals vary and are difficult to identify.
- Business goals are goals of the system's owner, which describe metrics by which the system's quality is judged. These goals are usually the primary reasons for the creation of the system and are clearly defined.

The goal of the UX designer is to align these two goals‘ categories and aim to fulfil them as precisely as possible.

Furthermore, a UX designer needs to be aware of two things:

- They want things that don't matter to the user.
  UX designers need to focus on user and organisation goals. They have to empathise and control their ambition for creating overcomplicated and flashy UI, which would look good in their portfolio but would not benefit their users.
- They know things that don't matter to the user.
  Empathy is the keyword. They need to design the UX with the average knowledge of the potential user of their system in mind. They need to assume that users know less than them.

Asking the right questions, correctly identifying the goals and communicating the purposes of the design clearly, should allow me to develop quality UX and thus develop a quality system.[10]

2.4 Web Application Development

This section will provide an overview of the theory about web application development, the comparison between web applications and desktop applications, languages used to develop web applications and basic types of web applications.

Web applications are applications running on remote servers. Users can access these applications conveniently via their browsers. Web applications have numerous advantages against their counterpart desktop applications:

- They have very low hardware requirements - anything that can run a browser can access the application.

- They are universally accessible – users can access the application from any device with a browser, and there is no need to have the specific application installed.
- They are multiplatform – the application is not bound to a specific operating system nor needs to have multiple versions for different operating systems.
- They are easier to license on a subscription payment basis, and there is low to no risk of software piracy.
- Minimised version and compatibility issues – all users have the same version of the application available.

Naturally, they also have some disadvantages:

- They require internet access.
- They can use only a limited amount of resources – resource-demanding applications (such as complex games) cannot effectively run in a browser.
- They have limited integration of a keyboard.
- They are easier to exploit by hackers than desktop applications.[8][9]

Most modern web applications use client-server architecture. The client is the user's browser which displays the UI to the user and executes business logic which communicates with the server accordingly to the user's interaction with the client. The server usually contains some form of SQL database with persistent data, which it provides or modifies accordingly to the client's requests. The means of communication between the client and the server are called web API (from now on, just API). Most modern APIs are based on the Representational State Transfer (REST) standard using the HTTPS (or outdated unsecured HTTP) protocol.

Server-side code can be written in any language with the capability to receive and respond to HTTP requests. Often used server-side languages include:

- PHP,
- Java,
- Node.js (JavaScript),
- Python and
- .NET (C#).

To simplify the interaction with the database, Object-Relational Mapping (ORM) frameworks are often used. ORMs abstract the database and allow building code-first databases (the database is generated from the definition of classes and other code structures) as a part of the server-side code. Therefore, the development team has no need to interact directly with the database via SQL queries and can interact with the OMR only. The communication in the client-server application is illustrated in the figure 2.9.

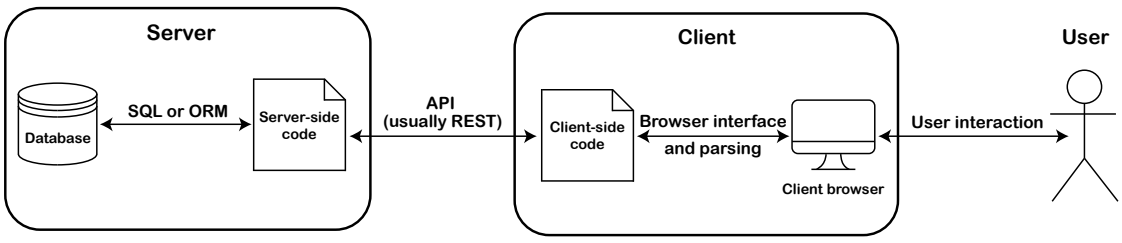


Figure 2.9: Web application client-server architecture.

The choice of client-side code languages is limited. Browsers can parse only a few languages. "At a minimum, any modern browser can parse HTML, CSS, and JavaScript." [2] HTML serves as a structure describing markup language, while CSS describes the page's appearance, layout, and design. Javascript is a language in which the functionality and interactivity of the page are programmed.

However, multiple platforms offer the possibility to write the code in other languages than Javascript and then compile into it. The role of Javascript, in these cases, is similar to the role of assembly languages in desktop applications (Javascript serves as the primary language into which other languages are compiled). Framework working on this principle is, for example, DotVVM[5], which allows developers to easily develop web applications in MVVM architecture by writing business logic in .NET. a similar tool for Python is Skulpt[6].

There are three main approaches to client-side programming categorised by the scope of interactivity and rendering.

The first and the most outdated one is the static approach. Pages rendered using this approach are fixated in appearance and contain static non-personalised information. To interact with the site, the client has to change the URL (by clicking buttons in navigation, for example), and the site has to be rerendered. Because of nearly non-existent interaction, sites written using this approach cannot be called web applications but are rather called websites.

Dynamic rendering uses server-side logic to generate dynamic HTML pages, which it then sends via HTTP response to the client. The site then reloads, and the client can render the newly generated dynamic page. This approach is also not ideal for high interactivity web applications but can be utilised in sites such as e-shops.

The most modern and advanced approach is the so-called Single Page Application (SPA). This approach relies on javascript-heavy client-side code, which dynamically changes the site's appearance based on the user interaction. The client asynchronously fetches data when required via techniques, such as AJAX[7], which are used to interact with the server without a need to reload the page. This approach is best suited for developing high interactivity applications, such as information systems or social networks. Because the process of dynamically updating the HTML page is rather complicated and would require an extensive amount of code, many frameworks have been developed over the years in order to handle the hard work for us. The three trendiest Javascript frameworks as for 2020 are:

- Vue
- React
- Angular

[5]More information about DotVVM on https://www.dotvvm.com/
[6]More information about Skulpt on https://skulpt.org/
[7]More about AJAX on https://www.w3schools.com/xml/ajax_intro.asp

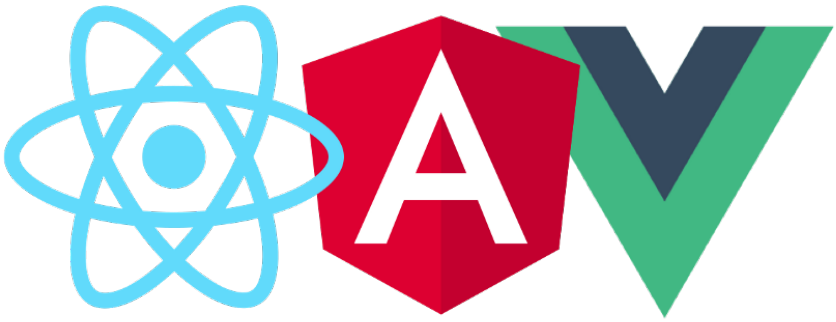


Figure 2.10: Logos of React, Angular and Vue (from left).[5]

Each of them has its advantages and disadvantages and is better suited for different tasks. Javascript framework relevancy is also heavily affected by trends.

Chapter 3

Requirements and Suggested Solution

In this chapter, I will identify the needs the system should fulfil. I will purpose my innovative dental chart design concept. Then, I will present an initial suggestion of this concept's execution in the form of wireframes. I will also present its data model. Lastly, I will present the design of the finished implementation of the solution, the functionality of the finished prototype and present the changes I have made to the initial design and data model.

3.1 Identified Dentist Needs

Before I suggest a solution, the details of a dental appointment must be first closely observed and analysed. In this part, I will aspire to define the goals of the dentist throughout such an appointment.

In this case, the user and business goals closely correspond. This is caused by the fact that the user and the business owner are the same person; the dentist himself. For example, an effective recording of a diagnosis does not only fulfil the dentist's desire to provide patients with proper care but also allows them to sustain more patients and thus achieve better financial results.

An important consideration of the solution is its intended usage. The system will be used by highly proficient users trained in the usage of the particular system. Therefore, unlike with projects such as websites, a certain amount of unintuitiveness or unusual but consistent behaviour is acceptable when redeemed by gains in effectiveness of trained users.

This section will expand on section 2.1. I will inspect the dentist's needs of interaction with the system, discovering what they expect to do and see and their goal at each part of the process.

Initial Examination

The dentist needs to be able to create a patient record quickly, ideally accessing the new record form in one click. The crucial information in the form has to be immediately accessi-

ble and has to be required by the system. A possibility of submitting optional information should also be available, but such information does not have to be as accessible. The new patient form should contain two logical sections:

- the general information containing the patients' name, phone number, email and other essential information as mentioned in section 2.1 and
- the anamnesis consisting of personal anamnesis, allergies, pharmacological anamnesis and toxicological anamnesis, which are all critical pieces of medical information and should be recorded prior to any patient treatment. They are, however, categorically different from general information. Therefore, they should be optically divided.

Check History

The dentist needs to find the record of the patient instantaneously. This would be usually achieved by searching for the patient's national ID number found on their insurance card (which is borrowed by the nurse when the patient enters the waiting room or is scanned by a specialised device in the waiting room). On the patient's record, the dentist has to be presented with the patient's general information, anamnesis or the current state of the patient's denture.

The visualisation of the current state of the patient's denture is one of the main focuses of this work. To effectively display all the important information, the classic simple table of records will not suffice. The display should be:

- graphic,
- familiar – for the user to adapt well and quickly to the UI, the user should feel comfortable in the system,
- representative of the real denture – so the view on the screen and the view in the patient's mouth are effortlessly comparable,
- prioritising the most crucial information – the dentist should first notice the most critical information to be able to quickly and precisely analyse the situation and
- interactive – the amount of information the dentist could need greatly exceeds the space available. The dentist should therefore be able to fetch detailed information by interacting with the UI.

Examine and Diagnose

The diagnoses need to be precisely localised to ensure the quality of the treatment. Dentists usually diagnose a handful of diagnoses types (tooth decay being the most prominent). Therefore, some frequently used diagnoses should be easier to diagnose than other rarer ones.

Treat Issues

After treating the issues, the dentist records the treatment. The treatment can be connected to a previous diagnosis. As with diagnoses, there are some procedures which the dentist records regularly. Therefore, some procedure codes have to be easier to access than others.

Finish Report

When writing the report, the dentist should not be bothered with submitting information recorded in previous steps of the process. All such records should automatically transfer into the report (more about recorded information in 2.1). The dentist should then write additional information into a simple text field and print the entire record on the patient's demand. Afterwards, the whole encounter record has to be archived to remain unchanged since the dentist's documentation has to fully correspond with the data submitted to the insurance company.

3.2 Design Concept - Editor Approach

Before I suggest my solution concept, I will closely assess the role of the dentist. The dentist is not just observing, examining and diagnosing as a practical practitioner does. They do not usually treat the patient by prescribing medications, referring to other specialists, recording feedback and suggesting other actions. Most of the time, they diagnose and then actively treat the state of the patient's denture in a relatively quick and irreversible manner. More than an observer and a coordinator, they actively fix and edit the state of the patient's denture.

Because the dentist's role can be described as one of an editor of the denture, the concept that I am suggesting is to design the system as an editor similar to the ones used for photo editing or graphic design, such as Photoshop or Inkscape. As illustrated in the figure 3.1, the layout of these editors usually contains some kind of a toolbox, with a list of the tools that can be used to edit a canvas. The canvas is traditionally situated in the centre of the application. The layout also usually contains a component, which is displaying the objects created in the application. I propose that the dental chart could serve as a canvas and the diagnoses and procedures creating actions as the editor's toolbox.

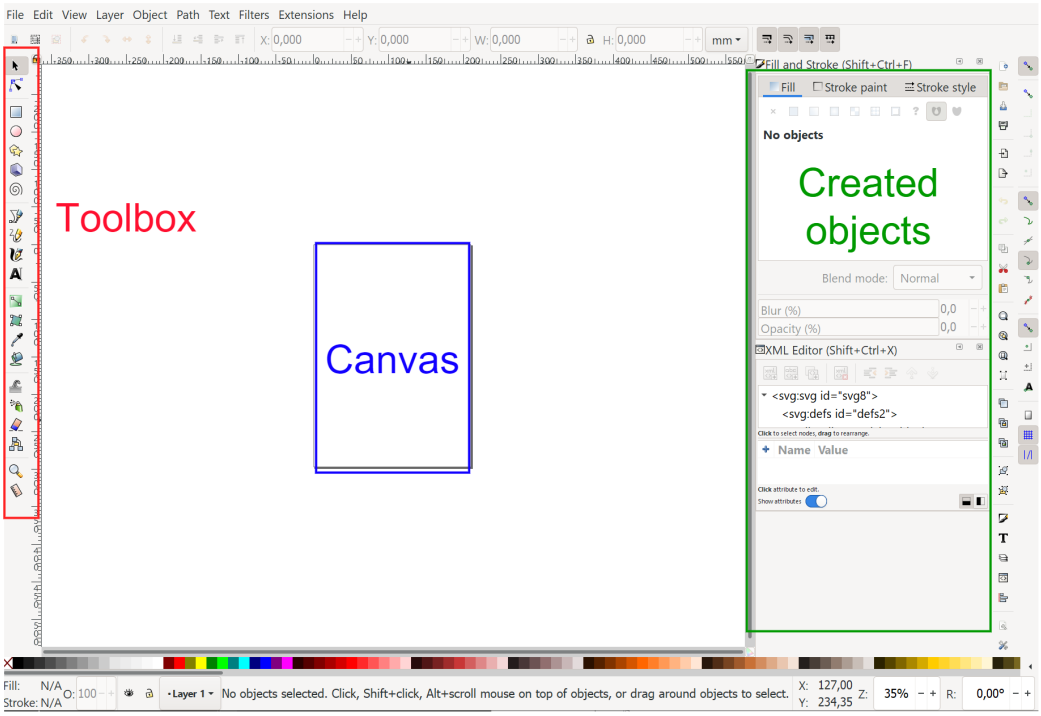


Figure 3.1: Typical editor application layout, as illustrated on a layout of Inkscape - a vector graphics editor software.

Implementing this idea will not only allow me to group many consecutive identical actions into fewer clicks (the dentist will select the tool and then apply it as many times as they like). It will also enable me to prioritise some actions over others. Furthermore, it will logically correspond with the role of the dentist. The system will not be presented as a simple recording system, but rather it will be conceptualised as a denture editor.

3.3 Design Suggestion

In this part, I will focus on the design accomodating the requirements identified prior in section 3.1, following the concept of my innovative editor approach defined in section 3.2. The main focus will be the dental chart view, but I will also design the patient detail view and the medical report view. The design will be presented as a series of wireframes with a description of the ideas behind them.

Dental Chart View

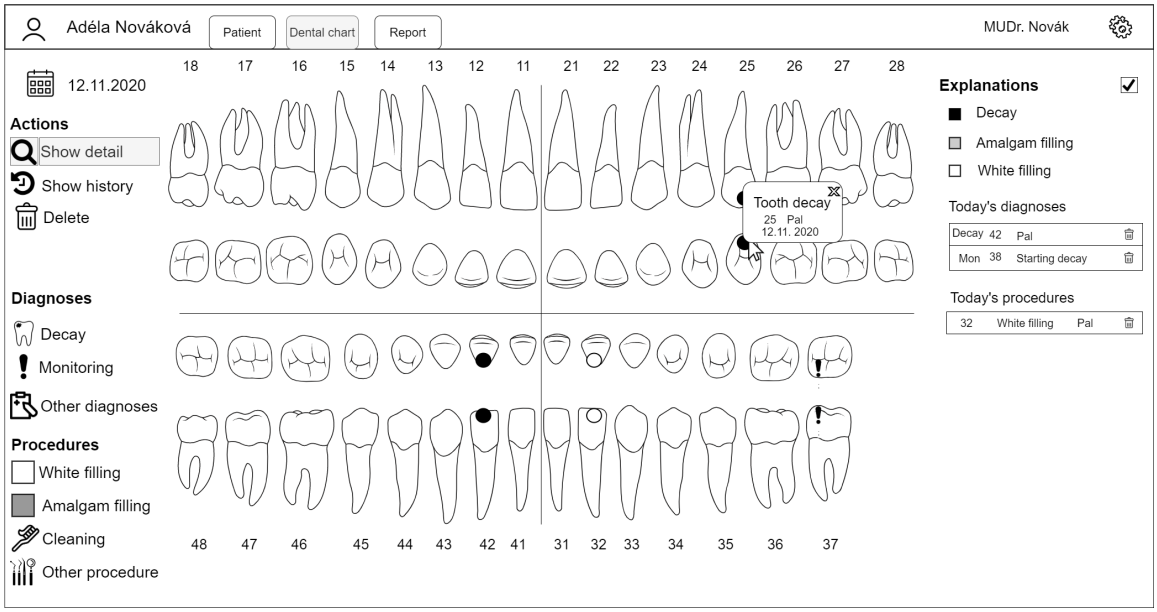


Figure 3.2: Wireframe of the Dental Chart View with the "Select" tool used.

In the wireframe 3.2, the dental chart editor with the "Show detail" tool used is visible. The editor's main feature is the dental chart itself. The dental chart is divided into four quadrants and displays all the tooth records and surfaces of the patient. Each surface (or a whole tooth) has its state displayed by the chart (for example, tooth 38 is not visible because it was extracted). Left of the chart, the toolbox of the dentist is placed. For faster orientation, it is divided by modes of interaction into actions, diagnoses and procedures sections.

- Actions serve to fetch information and general tools.
- Diagnoses serve for creating diagnoses.
- Procedures serve to create procedure records.

Right of the chart, the summary of all the diagnoses and procedures done on this encounter and the explanations to the graphical displays of the chart are present. The explanations can be hidden once the dentist becomes familiar with the system. The encounter does not have to be explicitly created but is rather created automatically when either a diagnosis or a procedure record is recorded. If no diagnoses or procedure records have been created, the encounter is generated with the report creation. This will spare one unnecessary user's action. On hover of the diagnoses and the procedure records rows, the records are highlighted in the dental chart for a better visual association of the position in the dental chart with the hovered record.

The "Show detail" tool is the default tool selected, because before any diagnosing or treatment, the dentist is expected to fetch information about the patient's denture. On click, it will open details about the tooth, the surface, the diagnosis or the procedure clicked.

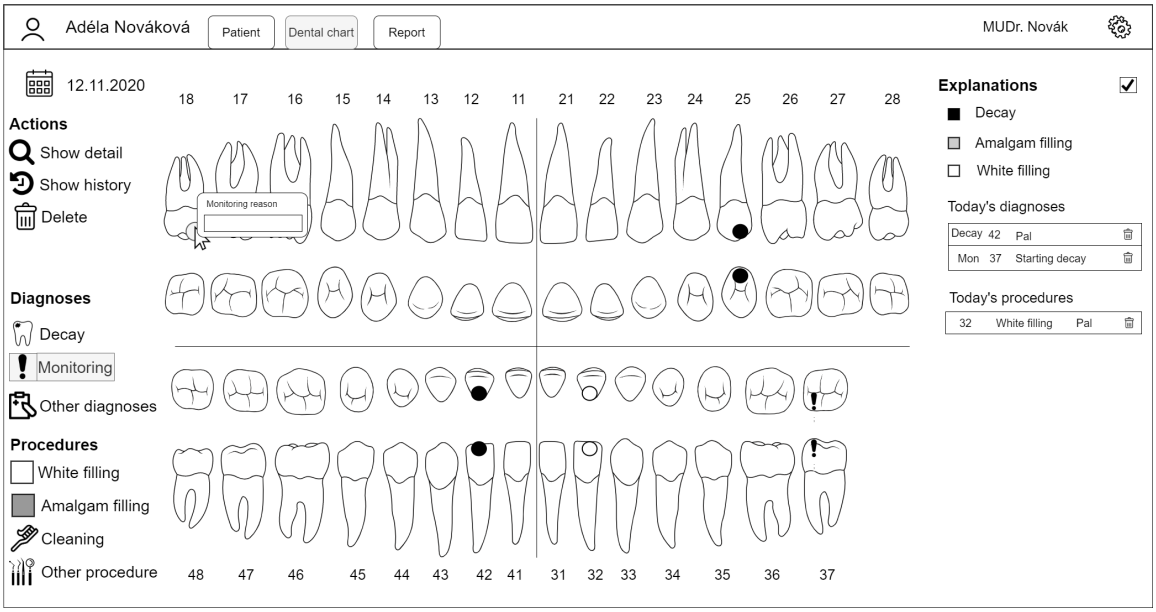


Figure 3.3: Wireframe of the Dental Chart View with the "Monitoring" tool used.

Using the "Monitoring" tool requires an additional note. Therefore, after selecting the localisation with this tool chosen, the dentist is presented with a prompt asking to submit additional information (as demonstrated in wireframe 3.3). If no further information is required when using a tool, the change is immediately performed. This is done to minimise the user's actions needed.

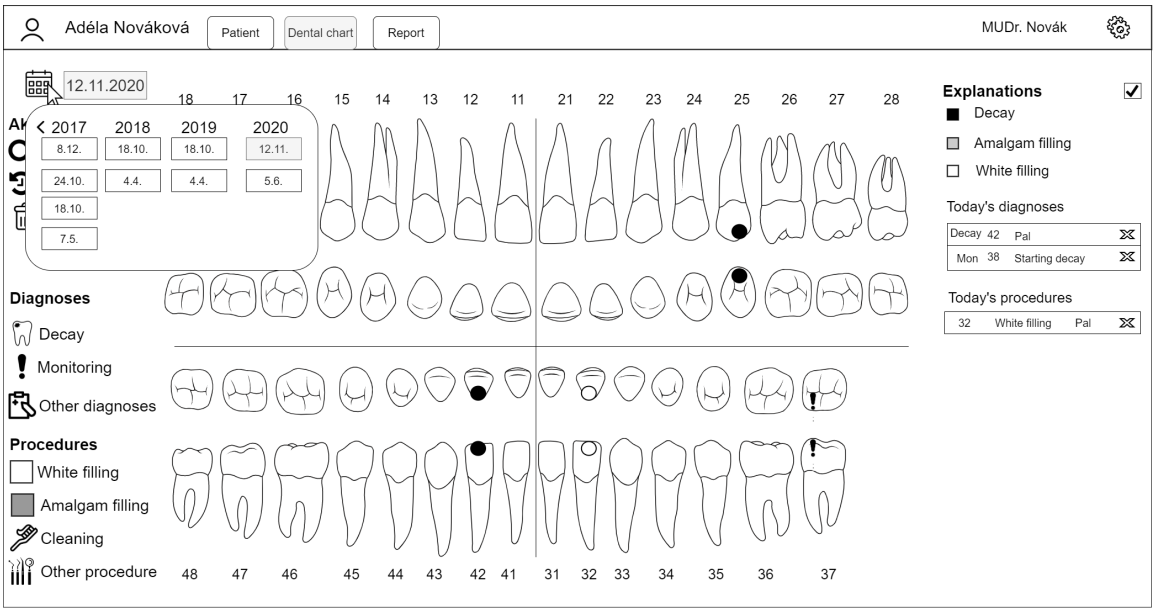


Figure 3.4: Wireframe of the Dental Chart View with the encounter picking component expanded.

By clicking on the calendar icon (exemplified in wireframe 3.4), the dentist can easily switch to a different encounter and view old patient's encounters and resulting medical reports in read-only mode (the encounters are already archived and cannot be changed). Reading old reports is vital for gaining the information unique to these reports (more about information recorded in medical reports in 2.1).

Medical Report View

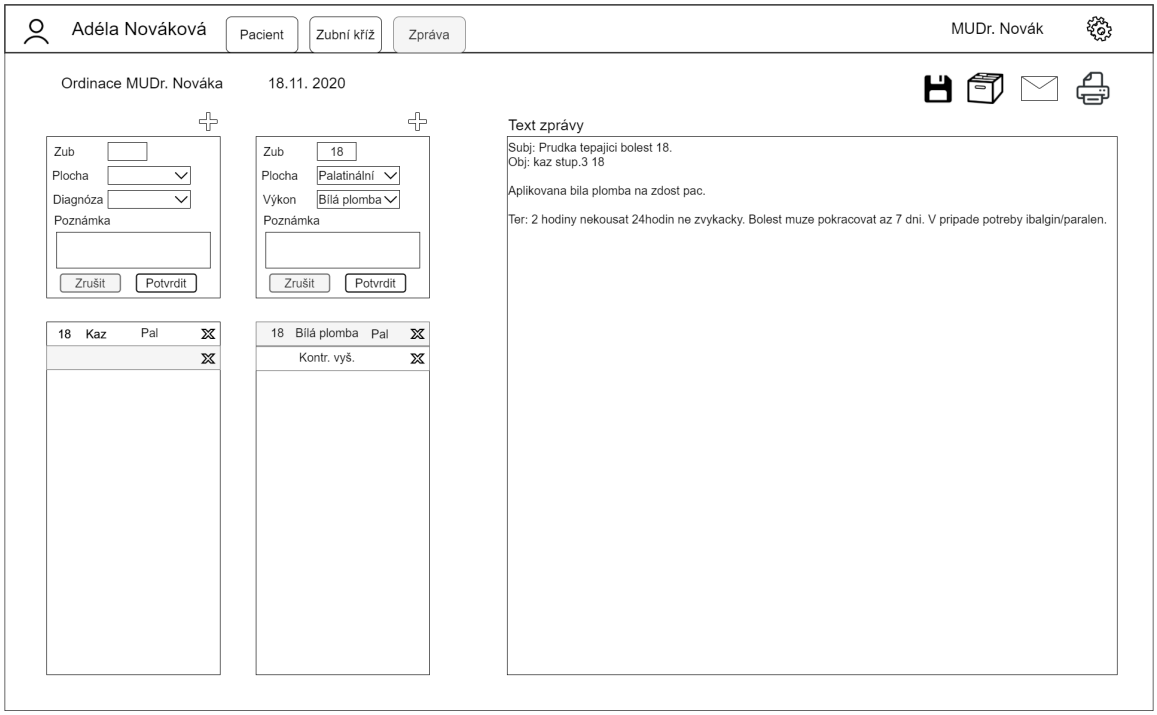


Figure 3.5: Wireframe of the Medical Report View.

In the report view, all the diagnoses and the procedure records created on this encounter are visible. The dentist can edit them, remove them and also add new ones. They can also write the text of the report into the text field. After the report is written, the dentist can save the report, archive the report (which locks the encounter), email the report to the patient, or print the report.

Patient Detail View

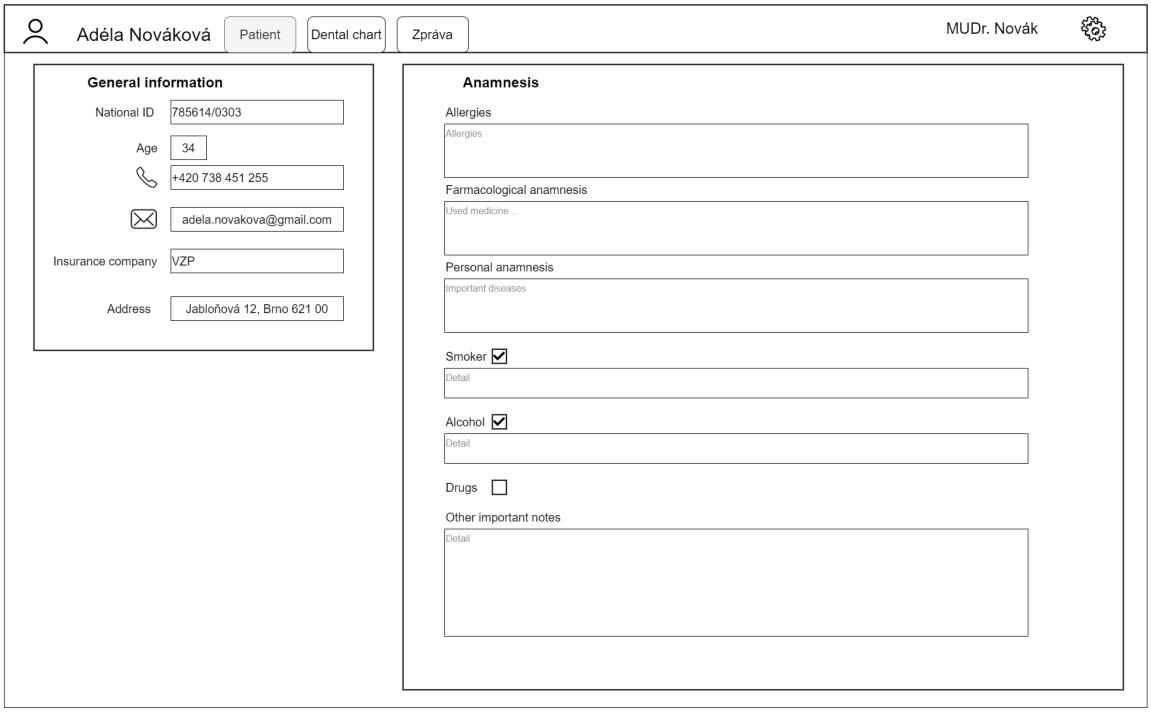


Figure 3.6: Wireframe of the Patient Detail View.

The patient view is simple. It does not need to contain much information (more about the information recorded in 2.1). Only basic general information and text blocks for the anamnesis are required. If the patient is not a smoker, does not drink or does not use drugs, the text fields for these specific pieces of information can be hidden not to clutter the user's view.

3.4 Data Model

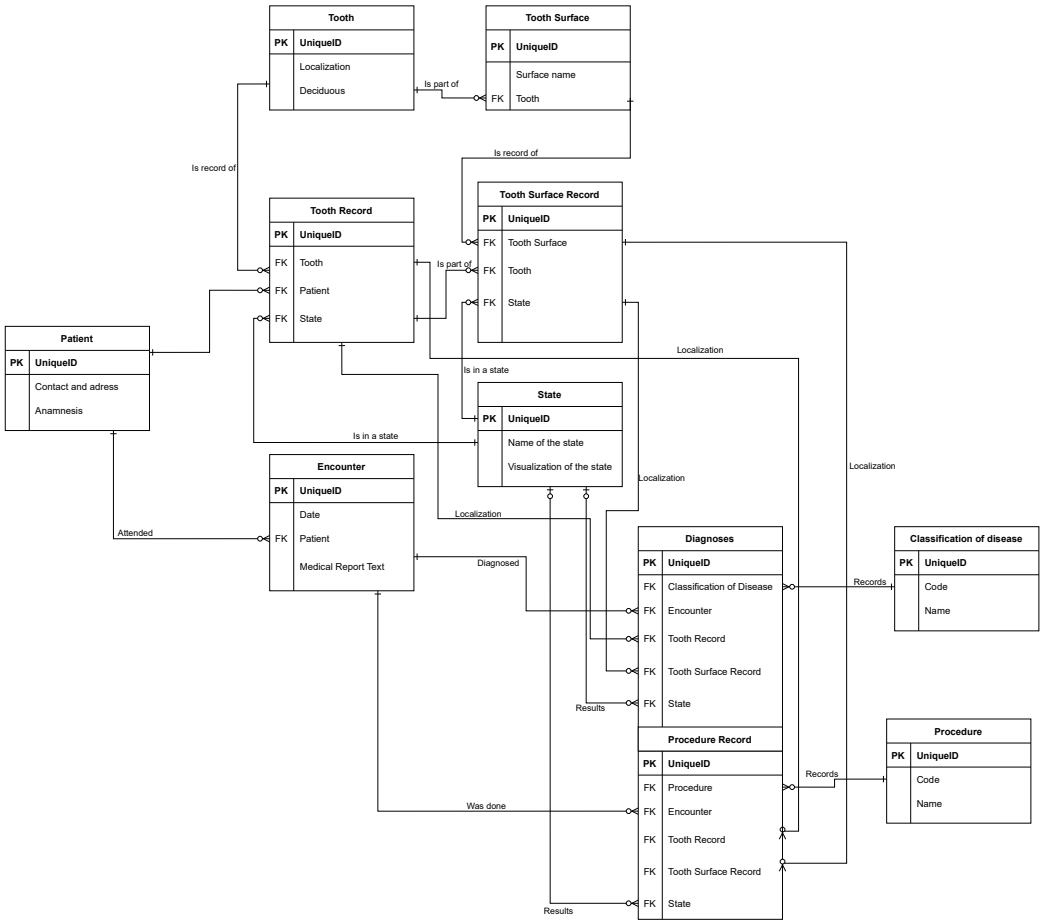


Figure 3.7: Entity relationship diagram of the data model.

Encounter

An entity representing the patient's visit. It groups records under a single date.

Procedure, Procedure Record

"Procedure" is a code entity that contains all codes of possible procedures in the healthcare system of the dentist. "Procedure record" is the record of the code with associated localisation and encounter.

Classification of Disease, Diagnoses

"Classification of disease" is a list of codes that are valid as diagnoses in the healthcare system of the country. Dentists usually use a limited subset of these classifications. "Diagnosis" records the disease code with associated localisation and encounter.

State

An entity that will result in a visual change of the dental chart (a change of colour or texture). It is a system entity for the dental chart functionality. Diagnoses or procedure records can result in a change of the state of their localised surface or tooth.

Tooth, Tooth Surface

These entities list all existing teeth and associated surfaces of each tooth. They serve to model a general denture according to which all the other dentures are generated.

Tooth Record, Tooth Surface Record

An entity representing tooth and tooth surfaces associated with a patient. It should be generated from the "Tooth" and "Tooth Surface" configuration.

Data Model Improvement

In the course of the solution development, I have improved the data model. I have made some minor changes.

First of all, I have decided to remove the "State" entity because it interfered with the concept of database design. Since the "State" is a direct indicator of the visualisation in the dental chart, the visualisation of the records should be decided by the frontend. Therefore, rather than having one state for each tooth, each tooth's visualisation is determined by the front end's logic. This resulted in a much more sustainable and logical design.

I have also decided to add the "DisplayName" field to the code entities (procedure and classification of disease) because the records for these entities are derived from the official code lists. The official namings can be hard to use because of their extensive lengths. Also, dentists are used to speaking in a slur. For example, the term "White Filling" is used for the procedure "Treatment of a permanent tooth with photo composite filling". The "DisplayName" field can also be used for translations.

3.5 The Final Design of the Application

After many iterations and many reworks of the design concept while developing the solution, the resulting design was slightly different from the initial design. The goal of the design was for it to be uncomplicated, information-dense and intuitive. Each action should have one definitive process of completing, and the completion should be as practical as possible without any extra unnecessary steps.

The colour scheme is intentionally simple. Each spur of colour has its meaning and signals a message to the user. Light grey signals selection. The selected tool, the selected encounter and the expanded form are highlighted by a light grey background. The dark red colour in the dental chart signals the localisation of the record expanded in the records list on the

right. Blue colour signals the localisation of the hovered record in the records list. Other colours serve as a visualisation of records in the dental chart.

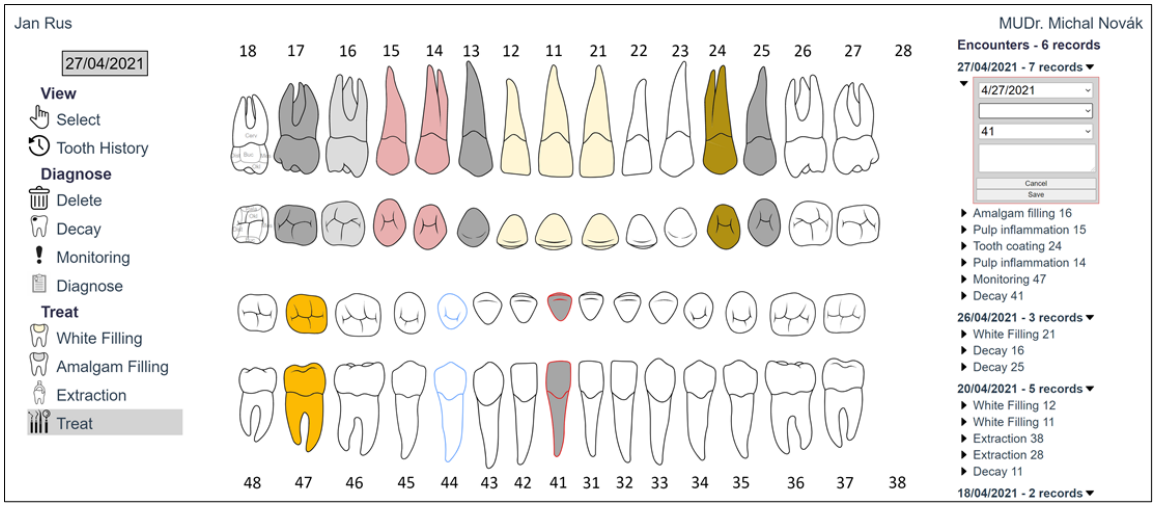


Figure 3.8: The final design of the solution.

Contrary to the initial design, I have entirely removed explanations from the application. The dentists using this system will be informed users, and by using the system for a while, they will quickly associate the colours with their meaning. There is no need for an uninformed user to be able to read the chart comfortably. I have also renamed some parts of the design to closer fit their purpose.

There are four main components of the implementation.

Toolbox Component

On the left is the toolbox (observable in 3.9), which is the component that serves for the dentist's choice of action, which will be performed on the chart. Only one tool can be selected at a time. As mentioned earlier in 3.3, the process of the dental appointment can be divided into three main phases. These are viewing, diagnosing and treatment (slightly renamed from the original suggestion). The tools are therefore divided into the same categories. By clicking on the teeth in the dental chart, the selected tool will be applied. The selected tool is highlighted in the toolbox by a light grey background. The tools available are:

Select

This tool expands the record visualised into the record form on the clicked tooth in the records list component. It also sets focus onto the note field.

Tooth History

This tool filters the records list component to the list of all the records made on the clicked tooth. This is a useful tool for quickly finding specific old records.

Delete

This tool deletes the visualised record on the clicked tooth. To ensure the record is really meant to be deleted, the user is prompted for a confirmation first. Only records created today can be deleted for consistency.

Decay, Monitoring

These tools are a shortcut for the more general tool "Diagnose". They create the desired record in one click. If the dentist wishes to add a note to the record, they can press and hold the Ctrl key while they click to automatically expand the record with a focus on the note field to start writing the note immediately. Any number of these tools can be rapidly added to fit the needs of the dentist.

Diagnose

This tool serves to record any desired diagnosis for the patient. On click of the tooth, a form in the records list component is expanded with a focus on the classification of disease dropdown. There, the dentist can pick any classification of disease available on the list. The record can then be saved by either pressing Enter or clicking the save button. The creation of the record can also be cancelled at any time by pressing the Escape key or clicking the cancel button.

White filling, Amalgam filling

These tools are a shortcut to the more general tool "Treat". They create the desired record in one click. If the dentist wishes to add a note to the record, they can press and hold the Ctrl key while they click to automatically expand the record with a focus on the note field to start writing the note immediately. Any number of these tools can be rapidly added to fit the needs of the dentist.

Treat

This tool serves to record any desired procedure record for the patient. On click of the tooth, a form is expanded with a focus on the procedure dropdown. There, the dentist can pick any procedure available on the list. The record can then be saved either by pressing Enter or clicking the save button. The creation of the record can also be cancelled at any time by pressing the Escape key or clicking the cancel button.

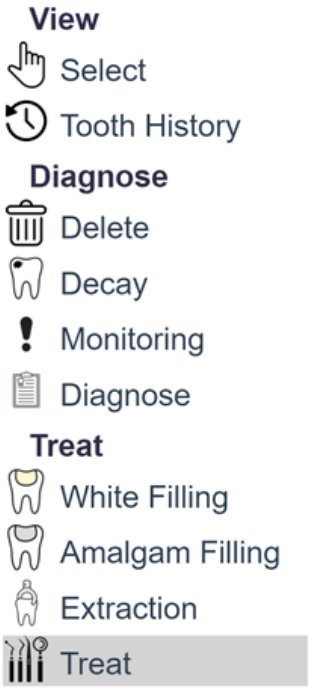


Figure 3.9: The toolbox component with the "Treat" tool selected.

Dental Chart Component

The dental chart is the main workspace. It displays the patient's denture and serves the tool usage. It is the dentist's canvas. For purposes of this prototype, the dental chart is only divided into teeth. For better orientation, the currently hovered tooth is highlighted by blue colour.

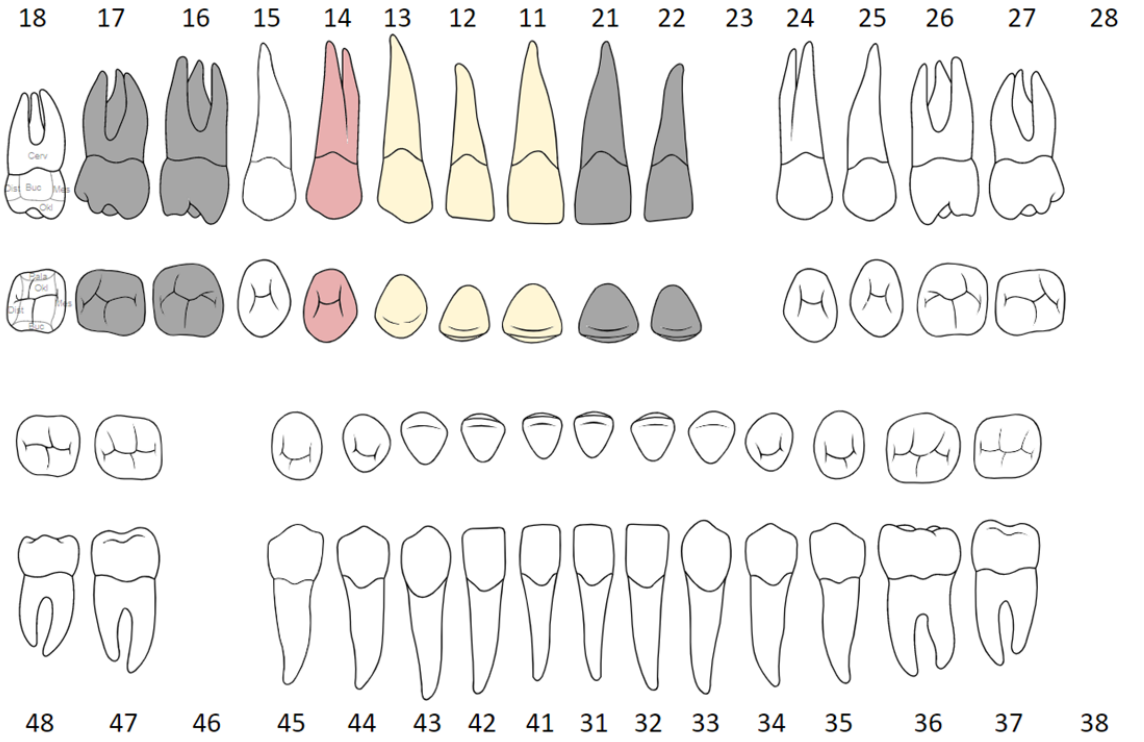


Figure 3.10: The dental chart component with visualised records.

In the final system, the teeth should be divided into surfaces, and multiple surfaces could be selected for a single record.

Records List Component

The records list displays all the procedure records and the diagnoses of the patient. Each record is expandable into a small and compact form. By default, the dentist can see records aggregated by the encounter in which they were created from the latest to the oldest. The total number of patient's encounters is displayed in the header of this component for better orientation in the records. Each aggregated encounter is followed by the number of records recorded in order to ease the navigation through the compressed aggregated records. The procedure records are shown first. The diagnoses are displayed afterwards because the procedure records usually follow the diagnoses and are more important outcomes of the encounter. Using the "Tooth History" tool, the dentist can filter the records to aggregate by the clicked tooth record. In the final system, more options of aggregation and filtering should be added.

Figure 3.11: The records list component.

Contrary to the initial design, I have decided to show all the records instead of only today's records. This will allow much denser information transfer. The aggregation by encounter will also ensure better readability, and the collapse feature will ensure good information prioritisation. I have found out that for most encounters, there are only a few, if any, records. Therefore, displaying only the records made today is a waste of space.

Records Forms Components

These forms display details of the created records and their editing. The expanded record is highlighted in the dental chart by dark red colour. The form is displayed on the expansion of the record on the list. Only the records created today are editable. The records created in the past are not editable (except for the note field) because they should be already recorded by an insurance company. Therefore, changing them would cause inconsistencies. For saving the records, the user can use the dedicated button or Enter key. For cancelling the edit, the user can use the dedicated button or Escape key. Deleting of records is rare. Therefore, a quickly accessible delete button is not needed in the forms.

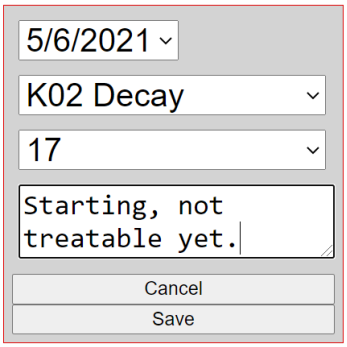


Figure 3.12: The diagnosis form component of a decay record.

In the final system, a multi-select control of tooth surfaces should be added. Contrary to the initial design concept, I have decided to nest the records form in the records list component instead of displaying them as floating prompts. This turned out to be a more consistent UX design since the appearance of the form does not hinder any other usage of the dental chart. If the form would appear as a floating prompt, it would overlay the dental chart and restrict its usage.

Encounter Picker

This component displays the currently chosen encounter to which the currently displayed context is bound. On the expansion of this component, the dentist can see all the patient's past encounters. By selecting a different encounter, they can switch the context and view the development of the patient's denture in time. This component will be important, especially when medical reports will be implemented, because encounter switching will be the only way to view older medical reports.

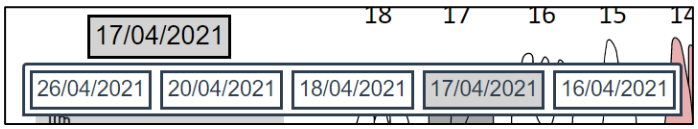


Figure 3.13: Expanded encounter picker with the encounter from 17/04/2021 picked.

Chapter 4

Development of the Solution and Testing

Because the entirety of the system proved to be impossible to develop in such a short period of time, I have implemented a prototype version of this system with a focus on the dental chart itself. This enabled me to thoroughly evaluate each and every design step and develop an incomplete yet intentional solution. This section will describe the product I developed, its design, tools and technologies used, data model, and some of the implementation details.

4.1 Backend

In this section, I will explain my backend application mechanics, share information about its development and show some implementation details.

For the purposes of the prototype, the database is a simple local PostgreSQL database used with a code-first approach managed by the EntityFramework ORM.

The backend was developed using .NET 5.0 API and EntityFramework ORM. It was developed in a Visual Studio 2019 integrated development environment (IDE). I have chosen these technologies because they are well documented, effective, and EntityFramework is one of the best ORMs for building code-first databases in the Visual Studio environment. I also utilised a technology called Swagger (as illustrated in figure 4.1), which allowed me to conveniently test, debug and document the function of my API. The project is called "DentistAPI" and consists of a few main directories and files:

- Models – These are classes representing entities of the database. All strong entities inherit from EntityBase.cs which contains the ID primary key property.
- Controllers – These are classes of functions that receive and handle API requests.
- `DentistAPIContext.cs` – This is a class of database context. It configures the entity framework ORM, entities in the database and their relationships. It also handles the creation of sample data for the database (also known as seeding).
- Seeds – These are sample data and lists of codes which will be seeded into a freshly created database via `DentistAPIContext.cs`.

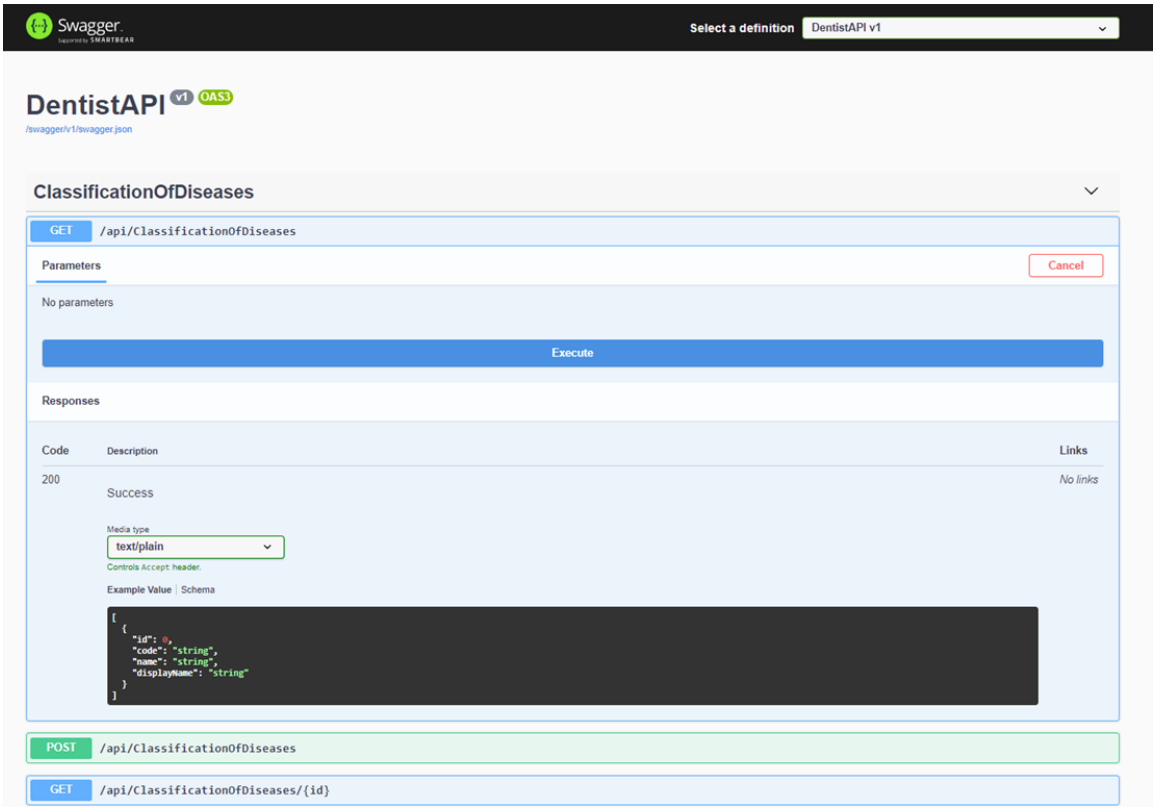


Figure 4.1: Swagger API testing and documentation environment.

Controllers

The controllers handle all the essential CRUD (Create Read Update Delete) operations for the desired entities. Their basis was generated with the help of EntityFramework ORM. By standards of REST APIs, CRUD operations are handled by specific HTTP request methods:

- POST submits new data to the server. (Create)
- GET retrieves resources. (Read)
- PUT updates existing data. (Update)
- DELETE removes data. (Delete)

All the data is transferred in the form of JSON objects, as it is a best practice for REST API. [11] To retrieve all the needed information, it is important to join related tables in the controller. To achieve this, the convenient function `Include()` is provided by EntityFramework. The usage of multiple chained includes is demonstrated in the figure 4.2.

```
// GET: api/Patients/5
[HttpGet("{id}")]
[EnableCors(origins: "*", headers: "*", methods: "*")]
0 references
public async Task<ActionResult<Patient>> GetPatient(int id)
{
    var patient = await _context.Patients
        .Include(p => p.ToothRecords).ThenInclude(t => t.ToothSurfaces).ThenInclude(ts => ts.ToothSurface)
        .Include(p => p.ToothRecords).ThenInclude(t => t.Tooth)
        .Include(p => p.Encounters).ThenInclude(e => e.Diagnoses).ThenInclude(d=> d.ClassificationOfDisease)
        .Include(p => p.Encounters).ThenInclude(e => e.Diagnoses).ThenInclude(d => d.ToothRecord).ThenInclude(t=>t.Tooth)
        .Include(p => p.Encounters).ThenInclude(e => e.Diagnoses).ThenInclude(d=>d.Encounter)
        .Include(p => p.Encounters).ThenInclude(e => e.ProcedureRecords).ThenInclude(p => p.Procedure)
        .Include(p => p.Encounters).ThenInclude(e => e.ProcedureRecords).ThenInclude(p => p.ToothRecord).ThenInclude(t => t.Tooth)
        .Include(p => p.Encounters).ThenInclude(e => e.ProcedureRecords).ThenInclude(p => p.Encounter)
        .FirstOrDefaultAsync(p => p.Id == id);

    if (patient == null)
    {
        return NotFound();
    }

    return patient;
}
```

Figure 4.2: `GetPatient` function – one of the core functions of the API utilizing multiple includes.

The most challenging function of the API was by far the `PostPatient()` function (figure 4.3. For each new patient, a whole assessed denture has to be created. For this, the entities "Tooth" and "ToothSurface" are used. Once the database is seeded, the structure of standard denture is created between these two entities. Then, by cycling through this already created structure, the "ToothRecords" and "ToothSurfaceRecords" for the patient are generated. This way, all the newly created patients have their dentures ready for visualisation in the dental chart and recording their medical history.

```
// POST: api/Patients
// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754
[HttpPost]
0 references
public async Task<ActionResult<Patient>> PostPatient(Patient patient)
{
    List<Tooth> teeth = await _context.Teeth.Include(t => t.ToothToothSurface).ThenInclude(t => t.Surface).ToListAsync();
    patient.ToothRecords = new List<ToothRecord>();
    foreach (Tooth tooth in teeth)
    {
        ToothRecord toothRecord = new ToothRecord()
        {
            Tooth = tooth,
            Patient = patient,
        };
        List<ToothSurfaceRecord> toothSurfaceRecords = new List<ToothSurfaceRecord>();
        foreach (ToothToothSurface toothToothSurface in tooth.ToothToothSurface)
        {
            ToothSurfaceRecord toothSurfaceRecord = new ToothSurfaceRecord()
            {
                ToothSurface = toothToothSurface.Surface,
                ToothRecord = toothRecord
            };
            toothSurfaceRecords.Add(toothSurfaceRecord);
        }
        toothRecord.ToothSurfaces = toothSurfaceRecords;
        patient.ToothRecords.Add(toothRecord);
    }
    _context.Patients.Add(patient);
    await _context.SaveChangesAsync();

    return CreatedAtAction("GetPatient", new { id = patient.Id }, patient);
}
```

Figure 4.3: PostPatient function.

4.2 Frontend

The frontend is developed using Vue.js version 3 with the router dependency installed. The code was written in the Visual Studio Code integrated development environment (IDE). This section will provide some basic information about the framework used, the project structure and some implementation detail of its components.

Vue.js

"Vue (pronounced /vju:/, like view) is a progressive framework for building user interfaces. Unlike other monolithic frameworks, Vue is designed from the ground up to be incrementally adoptable. The core library is focused on the view layer only and is easy to pick up and integrate with other libraries or existing projects. On the other hand, Vue is also perfectly capable of powering sophisticated Single-Page Applications when used in combination with modern tooling and supporting libraries (opens new window)."[18][1]

Figure 4.4: The logo of Vue.js.[7]

The central concept of Vue.js is based around building applications from building blocks called "components" or "vues". Each component has its:

- template – the display HTML of the component,
- Vue instance represented by JSON object and containing attributes, such as:
  - name,
  - properties – input values set by the parent component,

[1]Official documentation to Vue.js version 3 can be found on https://v3.vuejs.org/

  - methods – similar to object-oriented programming class methods, accessed by "this" keyword and
  - data – similar to class properties, accessed by "this" keyword and
- style – CSS styling for the component. In the Vue instance, the component lifecycle functions, such as "OnMount" or "Created", can be overwritten. Watcher functions for properties or data that are triggered by their change can be defined.

The router dependency handles the URL changes for the application. Even though the application has to function as a single page application, the URL modification is vital for convenient application usage, for example, using links.

Vue is a very powerful, convenient and modern framework, providing all the tools needed to develop a modern and user-friendly application.

Project Structure

Views

This folder contains the router Vue pages. Specifically, the page containing the list of patients and the page containing the dental chart interface for a specific patient.

Components

This folder contains all the child Vue components used in the parent routing pages.

Services

This folder contains all the functions handling API requests, helper functions and global variables and enums.

Assets

This folder contains all the non-code assets used in the project, such as SVG files and images.

Component Hierarchy

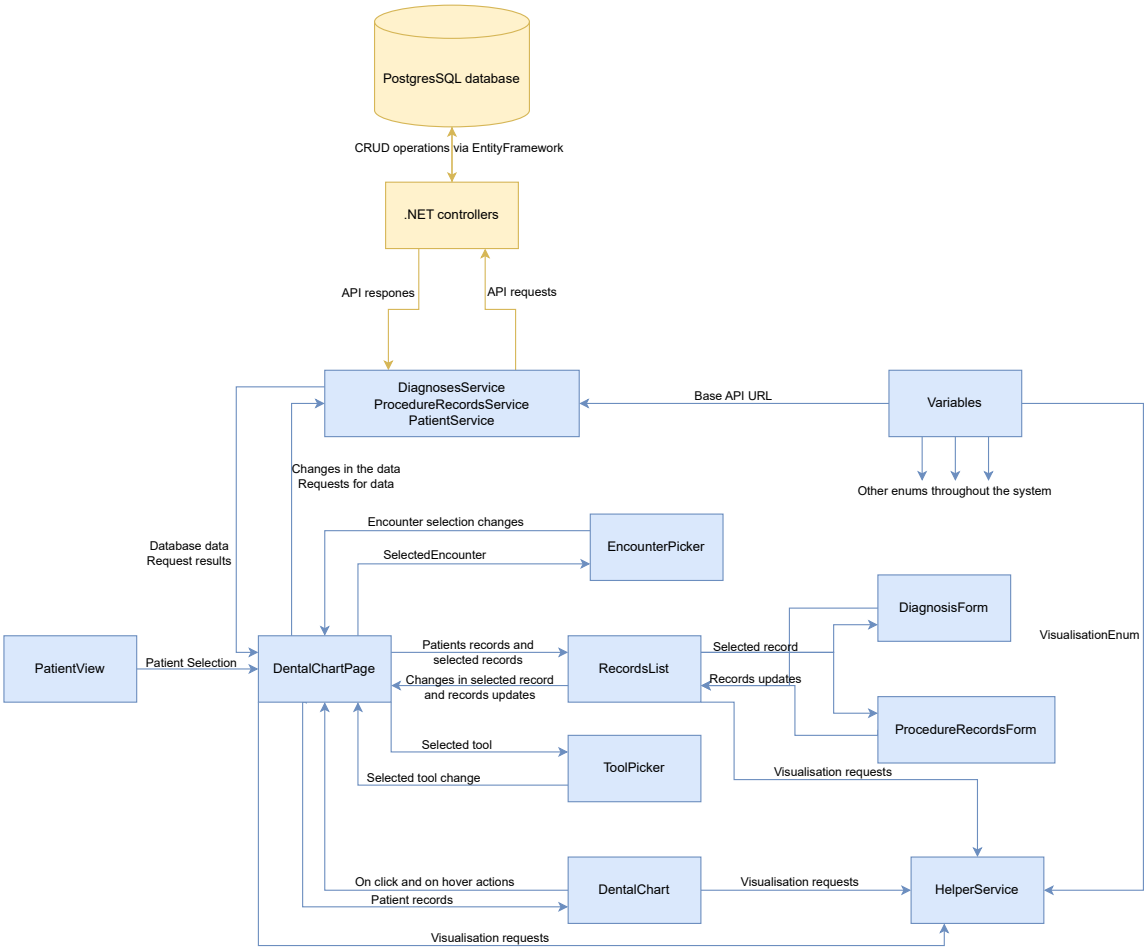


Figure 4.5: The diagram of the solution, components' hierarchy and data flow(the yellow colour signifying parts of the backend, the blue colour signifying parts of the frontend of the solution).

The "PatientList" view is only a simple list of patients for navigating into their dental charts. Therefore, I will not mention it any further.

The main view of the application is the "DentalChartPage". It is a root vue for all the other components. It also handles most of the interactions with the API. It contains the full context of the application, and the changes in its data are passed onto the children components. Therefore, by rewriting the parent's data, Vue rerenders all the child components using this data. The child components pass the information about their activity to the parent with the help of the `emit()` function. This function enables the child components to raise events to which the parent component can react with the help of the `v-on` keyword. This event can pass multiple components until it reaches the desired parent.

```
save: async function(){
    this.updateProcedureRecord.toothRecord = this.selectedTooth
    this.updateProcedureRecord.procedure = this.selectedProcedure
    this.updateProcedureRecord.encounter = this.selectedEncounter
    var result = await ProcedureRecordsService.UpdateProcedureRecord(this.updat
    this.updateProcedureRecord = result.data
    this.$emit('updateProcedureRecord', this.procedureRecord,this.updateProcedu
},
```

The event is raised via the emit function in the child component (procedure record is updated).

```
<procedure-record-form
  v-on:updateProcedureRecord="updateProcedureRecord"
```

This event then triggers a method in the parent component with the help of the `v-on` keyword

```
updateProcedureRecord: function (record,updateProcedureRecord) {
  record.expanded = !record.expanded;
  this.$emit("updateProcedureRecord", updateProcedureRecord);
},
```

and the method again raises the event for its parent component.

```
<records-list
  v-on:updateProcedureRecord="updateProcedureRecord"
```

Then, the method is once again triggered in the parent component

```
updateProcedureRecord:function(procedureRecord){
  var encounter = this.Encounters.find((e) => e.id == procedureRecord.encoun
 encounter.procedureRecords[
    encounter.procedureRecords.findIndex((d) => d.id == procedureRecord.id)
  ] = procedureRecord   //find the record in the encounter and substitute it
  this.Encounters[this.Encounters.findIndex((e)=>e.id==encounter.id)] = enco
  this.procedureRecords = HelperService.unpackProcedureRecordsFromEncounters
  HelperService.toothUnexpanded(this.formRecord.toothRecord.tooth.localizati
  HelperService.visualizeRecord(procedureRecord)
},
```

and because the core data is available and editable in the context of the root component, they can be updated. After the data is edited, Vue.js handles the rest.

Services

`DiagnosesService.js`, `ProcedureRecordsService.js` and `PatientsService.js`

These libraries handle all the connections to the API. `DiagnosesService.js` handles the creation, updates and deletion of Diagnoses. It also meditates fetching of classifications of disease. `ProcedureRecordsService.js` does the same for procedure records and procedures. `PatientsService.js` handles fetching and manipulating of patients and encounters. The third-party Axios library is used to handle the API requests.

`Variables.js`

This library contains all the used enums and dictionaries as well as the API base address. This modular approach enables simple expansion of the application and also improves readability. It also enables switching the API in order to accommodate existing systems or different versions and endpoints of the API whenever needed.

`HelperService.js`

This service handles all visualisation in the dental chart. Elements of the dental chart's SVG have element IDs set accordingly to the tooth they represent. Therefore, if the localisation of the record to be visualised is known, the dedicated element can be easily found by the `document.getElementById()` and its children styled accordingly (as visible in the figure 4.6).

```
static colorTooth(localization, resultVisualization) {
    try{
        document.getElementById('t' + localization).children.forEach(child => {        //get tooth of the localitazation
            if (child.style.fill != VisualisationEnum.SelectedTooth && child.style.fill != VisualisationEnum.NonSelectedTooth) {
                child.style = resultVisualization;
            }
        })
    }
    catch{
        null
    }
}
```

Figure 4.6: The function which handles the change of tooth's colour.

Components

Encounter Picker

This component takes two input properties, an array of patient's encounters and the date of the currently selected encounter. The list of the encounters is then displayed as an array of buttons by `v-for`. These buttons are labelled with the date of their encounter. On click of any of these buttons, an event is raised, and the clicked encounter is then passed to the parent. The parent then filters the context in order to display only the records created up to the date of the selected encounter.

Tool Picker

The list of available tools is handled by an enum in the `Variables.js` file. On click of each tool button, an event along with the enum value of the selected tool is emitted for the parent. The parent then records this selected tool into his `toolSelected` data field and then interprets the user's actions dependent on its value.

Dental Chart

This component contains the SVG of the dental chart itself. Since the dental chart is represented by an SVG element, the SVG child elements can be easily mapped and dynamically

interacted with by changing their CSS properties. On the mount of the component, event listeners are added to each element. These event listeners handle the on hover and on click functionalities by emitting events to the parent component. The component receives, as input properties, the list of records (diagnoses and procedure records) and tooth records by the parent component. The core method of this component is the `refreshDentalChart` method (figure 4.7). This function iterates through each tooth record and for each of them, it finds the record that is supposed to be visualised (the last procedure record or the last diagnosis if no procedure record exists on the tooth). This record is then passed to the `HelperService.visualizeRecord()`, which handles the visualisation.

```
static colorTooth(localization, resultVisualization) {
    try{
        document.getElementById('t' + localization).children.forEach(child => {        //get tooth of the localitazation
            if (child.style.fill != VisualisationEnum.SelectedTooth && child.style.fill != VisualisationEnum.NonSelectedTooth) {
                child.style = resultVisualization;
            }
        })
    }
    catch{
        null
    }
}
```

Figure 4.7: The `refreshDentalChart` method.

Records List

As an input property, this component takes the list of all patient's encounters along with the associated medical records. Two nested `v-for` components then display these. The expansion is handled by adding temporary `expanded` property to each record on mount. This property is read by the application, and the visibility along with the rotation of the expansion icon are handled based on it. Only one record can be expanded at a time. The expanded record is decided based on the "SelectedRecord" property. On the hover of each row, the localisation associated with this row is highlighted in the chart via the `HelperService.selectTooth()` function. To implement the "Tooth History" tool, this component also watches the `HistoryToothRecord` property. When this property is null, the list is displayed normally. When this record is not null, the records list aggregated by the encounters is hidden, and the records list for the tooth record is displayed.

Diagnosis/Procedure Record Form

These components handle the record reading and editing. The editing is handled by HTML forms. There are two modes, edit and display. The current mode of the form is decided by whether the record is recorded for today's date or not. If it is, the form is displayed in its entirety, and all the properties are editable. If not, the key properties (the code and the localisation) are display only. Therefore, the properties are displayed only as labels. On click of the save button or on press of the Enter key, an event is emitted for the records list component along with the updated record. The records list then passes it onto the root component. Then, an API service is called, and the update is handled. The form then collapses itself. On click of the cancel button or on press of the Escape key, the form is collapsed, and the update data is thrown away.

4.3 Testing

This section is an output of the short user testing and the discussion about the product with MUDr. Tomáš Machač. First, the user was given a minute-long application layout overview. Then, he was asked to perform certain tasks in the application. Afterwards, we discussed his opinions on the product. The whole debate and the record of the user testing in the Czech language can be found on the DVD provided with this thesis.

Usability Testing

Usability testing (also known as simply user testing) is testing in which the users are asked to perform some tasks in the tested application. The testing serves to retrieve information about how the user interprets the user interface. By user testing our application, we can uncover problems with unintuitiveness and unclear, conflicting or confusing messaging. Identifying these isuess is vital for further improvement of the user experience. The transcript of the course of the usability testing can be found in appendix A.

Discussion and Rating

The testing was followed by a discussion about the system and a rating, the shortened transcript of this discussion can be found in appendix B. In this discussion, the dentist was initially critical of the approach. He deemed it unintuitive and monotone. However, a while into the discussion, after understanding more about the system, the feedback grew more positive. The dentist deemed the design innovative. He has never seen such an approach to the dental chart. He also agreed that the design is more effective than the existing ones if the ideal course of the appointment was considered. He also rated the prototype in three categories. Functionality, intuitiveness and overall rating. The scores were as follows:

Considering the fact that the prototype was previously untested and most issues with the unintuitiveness can be easily fixed with a bit of styling, I find the outcome of the testing positive. The critique was also well formulated and helped me formulate multiple ideas for the system's improvement (more on that in 4.3). Therefore, the purpose of the testing was fulfilled.

Analysis of the Testing Outcome

The main problem was the lower intuitiveness of the solution and the monotone colour scheme. This was apparent in the usability testing and was confirmed in the discussion afterwards. However, a steep learning curve was noticeable even in such a short usage of the system. The time between actions shortened significantly. In the end of the discussion, the dentist did understand the purpose and value of my approach and even deemed it valid. Because the application is designed for everyday use by trained users, some unintuitiveness measures are excusable, or even expected, if redeemed by gains in usability or effectivity. However, several problems should be addressed and are avoidable. Main issues with the intuitiveness I have identified:

- The selection of the tool was not noticeable enough. The light grey background was not strong enough in contrast to the significance the selected tool has in the system overall.
- The records list component was too monotonously coloured. The encounter records were not differentiated enough from the diagnoses and procedure records, and thus, fused together.
- The expanded record was harder to find in the records list. The user did not know where to expect its appearance.
- The expanded encounter picker was not enough differentiated from the background, therefore fused with it, and thus, the dates of the encounters were harder to read.
- Division lines between the quarters of the dental chart were missing.
- The extracted tooth should not completely disappear from the chart. It makes navigating the chart harder.

Often repeated theme was the problem with a different approach. The dentist did not understand the approach to the application. He was unfamiliar with it and was used to a very different workflow. This was an anticipated problem because dentists use specific dental systems daily for years. Therefore, some confusion when using systems different from the ones they are used to is unavoidable. This could have been partially avoided by a more extensive introduction to the system or, even better, by a more extended time using the system. The (at least theoretical) validity of my approach was admitted.

Another valid point of the the critique was the translation of professional terms. More research about the terminology should be conducted to improve the overall professionality of the product.

Overall, I was genuinely pleased by the results of the testing. From the issues identified, most are easily fixable with minor changes. The critique was constructive and very well articulated. My approach was admitted to be innovative and at least theoretically viable. The ratings were also quite high for a previously untested prototype. I would once again like to thank MUDr. Tomáš Machač for agreeing to review this prototype and giving his feedback based on his professional know-how.

Improvement Suggestion Based on the Feedback

After identifying the key issues in the section 4.3, I will suggest actions to improve the product.

The selection of the tool was not evident enough. The light grey background was not strong enough in contrast to the significance the selected tool has in the system overall.

First of all, the selection should be made more visible. A simple way to achieve this would be by adding a border to the selected tool. I have chosen the same blue colour the teeth outlines are coloured with when hovered over to correspond with the colour scheme. This optically connects the hover over a tooth with the tool used on click. The underlying problem with the tool selection is the need to look at the toolbox to learn which tool is selected. This is an unnecessary eye movement. Therefore, I would suggest changing the mouse cursor to display the selected tool next to the cursor constantly. Both of these changes are illustrated in the figure 4.8.

Figure 4.8: Delete tool selected together with the delete cursor.

The records list component was too monotonously coloured. The encounter records were not differentiated enough from the diagnoses and procedure records, and thus, fused together.

I would suggest adding a background and a border to the encounter rows. This will create a more recognisable header than a bold text only. As shown in figure 4.9.

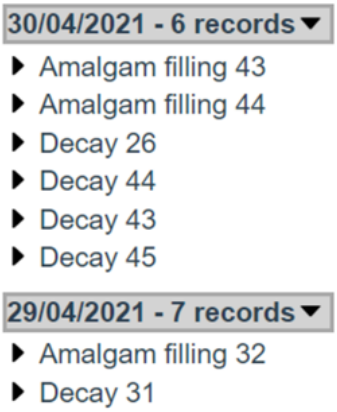


Figure 4.9: The records list control with improved design.

The expanded record was harder to find in the records list. The user did not know where to expect its appearance.

I suggest moving the expanded record form to a fixated position above the records list. Therefore, the records will not be "expandable" per se, but the user will choose which record they want to display in the record form. This will make the form appear in the same place every time, and the user will have a fixated place for reading and editing information.

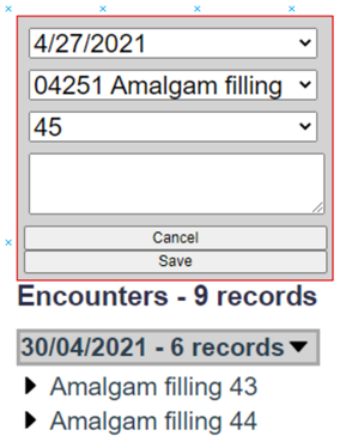


Figure 4.10: The expanded record form moved above the records list.

The expanded encounter picker was not enough differentiated from the background, therefore fused with it, and thus, the dates of the encounters were harder to read.

First of all, the selected encounter should be made more recognisable by changing its border to a blue colour, signifying selection. Then, I would suggest aggregating the encounters by the years in which they were made in. This would ease the navigation in the control. Next, I would suggest adding more margin between the encounters, so they are easier to differentiate from one another. I would also add a grey overlay to the rest of the system to bring the component visually forward. This is illustrated in the figure 4.11.

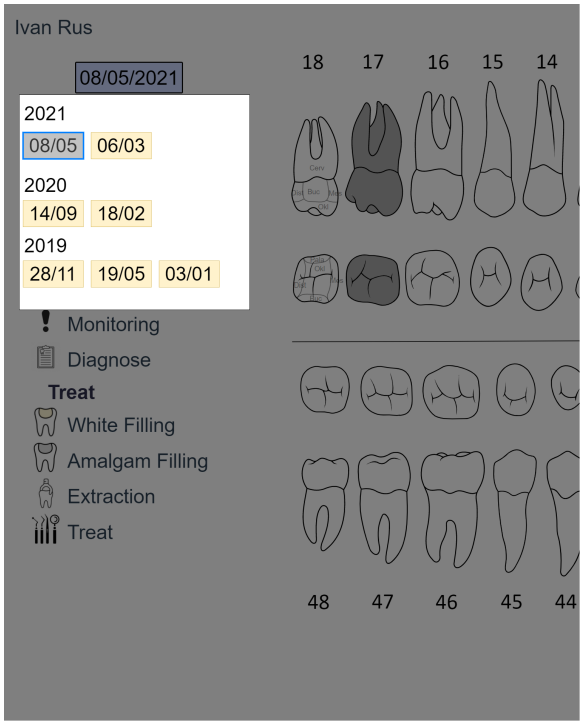


Figure 4.11: The encounter picker control fixed.

Division lines between the quarters of the dental chart were missing.

This is an easy fix. The chart without the lines was more visually appealing to me. However, the lines proved to be truly significant in navigation. They should be added. I have underrated their importance.

The extracted tooth should not completely disappear from the chart. It makes navigating the chart harder.

Instead of completely removing the tooth, I would suggest a dotted or dashed line for the tooth's outline as demonstrated in figure 4.12.

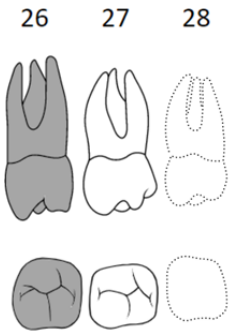


Figure 4.12: Extracted tooth 28 in the dental chart.

With all the minor issues fixed, the final design would look as 4.13:

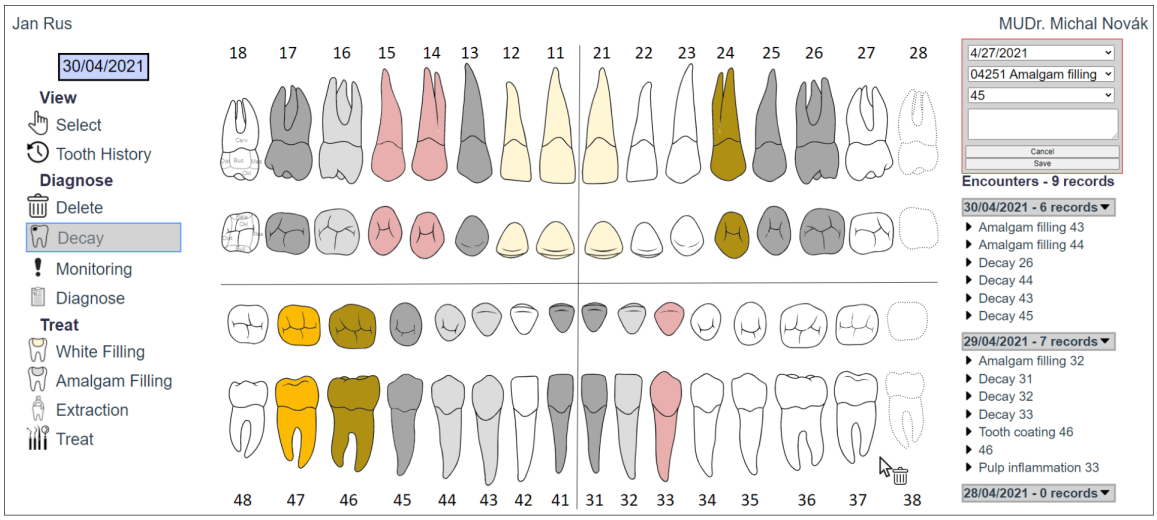


Figure 4.13: Final design with all the minor issues fixed.

Chapter 5

Conclusion

This work aimed to design and develop a prototype web application that would offer a suitable environment for dental records keeping, with the main focus being the interactive dental chart.

In the course of writing this thesis, I have researched the functioning of dental clinics and the process of treating their patients. This was achieved by studying relevant textbooks and consulting a professional in this field of study. I have also acquired knowledge in the field of UX design and web application development by studying written sources in the form of books and online articles. An overview of my accumulated information is summarised in this thesis.

Based on this knowledge, I have then designed, in the form of wireframes, a system for dental clinics with an innovative approach to dental record keeping via a dental chart. This design was inspired by a graphic editors' layout. The dental chart serves as a canvas and the diagnoses and procedures creating actions as the editor's toolbox.

I have implemented a prototype of the dental chart as a web application using .NET, EntityFramework and Postgres database as the backend and Vue.js as the frontend of the application, with the two communicating via REST API.

Then, I have consulted a professional dentist MUDr. Tomáš Machač who has tested the application and provided valuable feedback which can be utilised for further improvement of the design. I have analysed the feedback and purposed solutions to the identified issues.

My application was rated 6/10 by MUDr. Tomáš Machač, with the main issues being minor design flaws leading to worsened user experience and the missing areas of implementation due to the application still being a prototype. My approach to the design was deemed innovative, theoretically valid and potentially effective in practice.

To continue developing the product, the solutions I have proposed to the identified problems should be implemented. Afterwards, the division of the teeth into tooth surfaces can be added as suggested in my design. In this stage of implementation, the application can be subsided to thorough testing in an actual dental practice. Suppose the application and my approach proves to be practical. The application can then be embedded into an existing system, or an entire dental system can be created around it, as proposed in my initial design.

Bibliography

[1] [cit. 6.5.2021]. Available at: https://www.dub.cz/ohlasy/509-01.gif.

[2] 2 Steps To Parsing Code. *Websitebuilders.com*, 12. september 2015 [cit. 6.5.2021]. Available at: https://websitebuilders.com/how-to/glossary/parsing/.

[3] Ambulantní software, C. M. Česká republika s.r.o.divize. *Počítačový program pro lékařské ambulance Uživatelská příručka*. Jeremiášova 1422/7b155 00 Praha 5.

[4] Armstrong, C. What Is UX Copywriting? *COPYHACKERS*, 6. november 2019 [cit. 6.5.2021]. Available at: https://copyhackers.com/2019/11/ux-copywriting/.

[5] Barros, A. Angular / React / Vue pros and cons. *Afonso Barros*, 11. april 2018 [cit. 30.4.2021]. Available at: https://medium.com/@afonsobarros/angular-react-vue-pros-and-cons-75e161311e86.

[6] Bland, K. Dental Anatomy: A Review. *American Dental Assistants Association*. [cit. 6.5.2021]. Available at: https://adaa.cdeworld.com/courses/21595-dental-anatomy-a-review.

[7] Bo, H. D. Vue.js as an enterprise solution. *Positive thinking company*. [cit. 6.5.2021]. Available at: https://www.positivethinking.tech/wp-content/uploads/2021/01/Logo-Vuejs.png.

[8] Gibb, R. What is a Web Application? *STACKPATH*, 31. may 2016 [cit. 6.5.2021]. Available at: https://blog.stackpath.com/web-application/.

[9] Hussey, E. Web Apps vs. Native Apps: Advantages & Disadvantages. *Jācapps*, 21. may 2019 [cit. 6.5.2021]. Available at: https://jacapps.com/web-apps-vs-native-apps-advantages-disadvantages/.

[10] Joel Mars, J. M. *UX for beginners*. Sebastopol: O'Reilly, 2016. ISBN 978-1-491-91268-3.

[11] John Au Yeung, R. D. Best practices for REST API design. *THE OVERFLOW*, 2. march 2020 [cit. 6.5.2021]. Available at: https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/.

[12] Klepáček, I. *Klinická anatomie ve stomatologii*. Praha: Grada, 2001. ISBN 8071697702.

[13] Krug, S. *Don't make me think, revisited: a common sense approach to web usability*. San Francisco: New Riders, 2014. ISBN 0-321-96551-5.

[14] Ležovič, J. *Stomatologie*. 2nd ed. Praha: Avicenum, 1990. ISBN 80-201-0048-2.

[15] Mazánek, J. *Zubní lékařství*. Praha: Grada, 2014. ISBN 978-80-247-3534-4.

[16] Nováková, B. V. *POROVNÁNÍ SOFTWAROVÝCH SYSTÉMŮ PRO ZUBNÍHO LÉKAŘE*. Kladno, 2017. Master's thesis. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE FAKULTA BIOMEDICÍNSKÉHO INŽENÝRSTVÍ, Katedra biomedicínské techniky.

[17] Vickie Parrish Foster, M. Surfaces of the Teeth. *Dentalcare.com*. [cit. 6.5.2021]. Available at: https://www.dentalcare.com/en-us/professional-education/ce-courses/ce500/surfacesurl-of-the-teeth.

[18] You, E. Introduction. *Vue.js*. [cit. 6.5.2021]. Available at: https://v3.vuejs.org/guide/introduction.html.

Appendices

Appendix A

Transcript of the Usability Testing

The user was given following tasks:

Read the note of the White Filling procedure record on the tooth 12 and then edit it.

The user clicked on the tooth 12 with the pre-selected "Select" tool. The record was expanded in the records list. The user did not understand that he was already seeing the record and clicked the tooth three times more. Then, he overwrote the note and saved it by pressing Enter.

Display only records for the tooth 12.

The user was thinking for a while and then selected the tool "Tooth history". He did not notice the tool was already selected and tried to click it a few more times. Afterwards, he clicked the tooth requested, and the records were filtered.

Create a decay record for the teeth 31, 32, 33 and then treat it by extraction, amalgam filling and white filling.

The user tried to click the tooth 31 with the "Select" tool chosen. Since this tooth had no associated records, nothing happened. This confused the user for a while. Then, he selected the "Decay" tool. Next, he created the decay on the tooth 31. The tooth was coloured, and a new record was created in the records list. The user searched for the newly created record. It took him a while. Then, he created the other two decays. He then treated all the teeth without any problem.

Create tooth coating diagnosis for the tooth 47.

The user selected the "Diagnose" tool. He became confused because he could not find a "Tooth coating" tool. I explained that the tooth coating diagnosis must be created via the "Diagnose" tool since it is an unusual diagnosis. The user then used the tool, selected the "Tooth coating" classification of disease and saved the record by clicking the "Save" button.

Delete white filling on the tooth 33.

The user clicked the tooth 33 with the "Diagnose" tool selected from the previous step. This created a new empty diagnosis. The user searched for the delete button in the new form. I explained that he had the wrong tool selected. He then picked the tool "Delete" and deleted the record.

Display the dental chart in the context of the 20/04/2021 encounter.

The user remembered the encounter picker component from the introduction, struggled for a while to find the correct encounter and then selected it.

Appendix B

Transcript of the Discussion After Usability Testing

This is an edited shortened transcript of the discussion about the system which followed the testing. I will summarise the main points of the discussion.

Me: *Could you imagine using this tool in your everyday practice?*

MUDr. Machač: *Not yet.*

Me: *Why not?*

MUDr. Machač: *It is not very clear. It might be a problem with my habit. I am used to first clicking the tooth and then creating the record. The user interface is not guiding me enough. Without tooth surfaces implemented, it cannot be used. It is monotone and not colourful enough. I am a dyslectic, so it fuses together for me. I am a bit lost in the design. I must focus and search where the following action is supposed to be done. I would add quadrant division lines to ease the navigation.*

Me: *So the main problem is a difference in approach and monotonousness?*

MUDr. Machač: *Yes, and the workflow does not suit me.*

Me: *there any critical aspect missing? Rate the functionalities the solution offers from 1-10.*

MUDr. Machač: *Except for the surfaces, not really. I would appreciate more diagnosing tools. However, the division of the diagnoses and the procedure records into often used ones and rare ones makes sense. I think it is pretty good. I would say 7.*

Me: *Was the design intuitive? Rate it from 1-10.*

MUDr. Machač: *Not that much. But the more I am using it, and the more we are discussing it, it starts to make more sense. It needs some work with guidance, but it is not terrible. I would say 5.*

Me: *How significantly is this tool different in approach from the tools you are using in your everyday practice? Does it bring any innovative thoughts?*

MUDr. Machač: *It is indeed innovative with its approach. I do not know of any other tool using this approach. I can imagine the approach being useful when put into practice, but it still needs some polishing.*

Me: *Would you implement any part of this system into the system you are already using? Rate your experience from 1-10.*

MUDr. Machač: *The more we discuss it, the more the workflow makes sense. Hypothetically, if we think about the ideal case, it saves time. The question is if that would prove in practice. It is not finished, and there is room for improvement, so I would give it an overall score of 6.*

Me: *You may add any other notes you would like to say.*

MUDr. Machač: *I am not fond of the tooth completely disappearing when extracted. It is then easily overlooked. I would use the word "caries" instead of "decay". I am used to using Latin terms more than English in the international field. Especially the records list is too monotone. I would like to see more colour separation. It could use more synoptical.*

Appendix C

Contents of the DVD

- `DentistAPI` - this folder contains the source codes for the .NET Visual Studio backend API project
- `DentistUI` - this folder contains the source codes for the Vue.js frontend application
- `Dental Chart For Dental Clinic` - LaTeX source files for the thesis
- `Dental Chart For Dental Clinic.pdf` - this thesis
- `UserTesting.mp4` - recording of the usability testing and the following discussion with MUDr. Tomáš Machač in Czech language
- `Poster.pdf` - presentation poster for my solution
- `Presentation.mp4` - short presentation video for my solution

Appendix D

Installation Guide

This is the installation guide for my prototype. The installation includes the guide to setting up the API server and setting up the Vue.js application. This prototype requires you to have .NET 5 SDK, PostgreSQL v12, Visual Studio 2019 and Node.js installed. This guide works only for the Windows 10 operating system. Due to the prototype nature of this application, this installation guide is not guaranteed to work on every device.

API Server Configuration

1. Install .NET 5 SDK from https://dotnet.microsoft.com/download/dotnet/5.0.
2. Install PostgreSQL v12 database, along with the pgAdmin4 (useful for viewing and managing the database) from https://www.postgresql.org/download/windows/. Set all passwords to "postgres".
3. Install Visual Studio 2019 from https://visualstudio.microsoft.com/vs/community/. Make sure to install packages for ASP.NET along with it.
4. Start the PostgreSQL service, for example, by starting pgAdmin4 installed along with the PostgreSQL.
5. Open `DentistAPI/DentistAPI.sln` in Visual Studio 2019. Enter command `update-database` into the NuGet package manager console (`Tools/NuGet Package Manager/Package Manager Console`). This should create the Postgres database on your device.
6. Start the IIS Express launch profile (by clicking the IIS Express next to the green play button). Allow adding SSL certificate if prompted. This should open a Swagger interface in your default browser.
7. Run the `GET` function on `ClassificationOfDisease`. If JSON data is returned, the server was set up and is running correctly.

Frontend Vue.js application instalation

1. Install Node.js from https://nodejs.org/en/download/. Make sure that it is added to the path system variable.
2. Open the command line or PowerShell in the `DentistUI` folder.
3. Enter command `npm run serve`. This should build and start the project. Open the site http://localhost:8080/.

4. If the server is set up correctly and is running, you should be presented with the record of a patient "Jan Rus".
5. Enjoy using the prototype.