FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Rozšíření možností Czech Salivary Gland Database pro klinickou praxi a pro analýzu dat

Vojtěch Jelínek

PLZEŇ

2024

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Bakalářská práce

Rozšíření možností Czech Salivary Gland Database pro klinickou praxi a pro analýzu dat

Vojtěch Jelínek

Vedoucí práce
Ing. Petr Brůha

PLZEŇ 2024

Citace v seznamu literatury:
JELÍNEK, Vojtěch. *Rozšíření možností Czech Salivary Gland Database pro klinickou praxi a pro analýzu dat*. Plzeň, 2024. Bakalářská práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Ing. Petr Brůha.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vojtěch JELÍNEK**
Osobní číslo:	**A21B0154P**
Studijní program:	**B0613A140015 Informatika a výpočetní technika**
Specializace:	**Informatika**
Téma práce:	**Rozšíření možností Czech Salivary Gland Database pro klinickou praxi a pro analýzu dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Seznamte se s aplikací Czech Salivary Gland Database vyvíjenou na ZČU.
2. Ve spolupráci s vedoucím práce a lékaři z FN Motol analyzujte stávající nedostatky této aplikace a vyberte (dle stupně jejich závažnosti) minimálně tři, pro které navrhněte a realizujte řešení.
3. Prostudujte problematiku tvorby zdravotnických databází a jejich využití pro klinické a výzkumné účely.
4. Rozšiřte dále aplikaci o vizualizaci tzv. křivek přežití (Kaplanovou–Meierovou metodou) pro vybrané skupiny pacientů v databázi k hodnocení úspěšnosti léčby.
5. Výsledné řešení nasaďte na pracovišti FN Motol.
6. Otestujte výsledné řešení na reprezentativním vzorku pacientů a zhodnoťte dosažené výsledky práce.

Rozsah bakalářské práce: **doporuč. 30 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

Dodá vedoucí bakalářské práce.

Vedoucí bakalářské práce: **Ing. Petr Brůha**
Katedra informatiky a výpočetní techniky

Datum zadání bakalářské práce: **2. října 2023**
Termín odevzdání bakalářské práce: **2. května 2024**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 25. října 2023

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Plzni dne 30. dubna 2024

. . . . . .

Vojtěch Jelínek

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Cílem této bakalářské práce je rozšířit možnosti využití aplikace Czech Salivary Gland Database v klinické praxi a pro analýzu dat pacientů. Práce se soustředí na identifikaci stávajících nedostatků aplikace a navrhuje opatření pro jejich řešení. Dále se zabývá problematikou tvorby zdravotnických databází a rozšířením aplikace o vizualizaci dat pomocí Kaplan-Meierovy metody. Výsledná aplikace je testována na reálných datech a nasazena na pracovišti Fakultní nemocnice v Motole, což poskytuje lékařům efektivní nástroj pro shromažďování a základní analýzu dat.

Abstract

The aim of this bachelor thesis is to extend the possibilities of using the Czech Salivary Gland Database in clinical practice and for patient data analysis. The thesis focuses on the identification of the current shortcomings of the application and proposes measures to address them. It also deals with the issues of creating medical databases and extending the application with data visualization using the Kaplan-Meier method. The resulting application is tested on real data and deployed at the University Hospital in Motol, providing physicians with an effective tool for data collection and basic data analysis.

Klíčová slova

Electron.js • ReactJS • Czech Salivary Gland Database • zdravotnická databáze • Kaplan-Meierova metoda

Poděkování

Rád bych vyjádřil svou upřímnou vděčnost svému vedoucímu, Ing. Petrovi Brůhovi, a panu doktorovi MUDr. Davidu Kalfeřtovi, Ph.D., za jejich neocenitelné vedení a podporu během tvorby této bakalářské práce. Také hluboce děkuji své rodině a přítelkyni za jejich neochvějnou povzbuzující podporu a porozumění.

Obsah

1 Úvod **5**

2 Seznámení s aplikací Czech Salivary Gland Database 7
- 2.1 Hlavní účel aplikace . . . 7
- 2.2 Uživatelské rozhraní . . . 7
  - 2.2.1 Seznam pacientů . . . 7
  - 2.2.2 Přidání pacienta . . . 9
  - 2.2.3 Seznam studií . . . 10
  - 2.2.4 Přidání studie . . . 10
  - 2.2.5 Exportovat data . . . 10
- 2.3 Analýza zdrojového kódu aplikace . . . 11
  - 2.3.1 Struktura aplikace . . . 11
  - 2.3.2 Vstupní bod aplikace . . . 12
  - 2.3.3 Základní funkce souboru main.js . . . 12
  - 2.3.4 Modul renderer . . . 13

3 Analýza současných nedostatků **19**
- 3.1 Identifikace nedostatků . . . 19
- 3.2 Nedostatky functionality aplikace . . . 19
  - 3.2.1 Vysoký stupeň závažnosti . . . 19
  - 3.2.2 Střední stupeň závažnosti . . . 22
  - 3.2.3 Nízký stupeň závažnosti . . . 22
- 3.3 Nedostatky v implementaci a struktuře kódu . . . 23
  - 3.3.1 Front-end . . . 23
  - 3.3.2 Back-end . . . 24
  - 3.3.3 Struktura kódu . . . 24

4 Problematika tvorby zdravotnických databází **27**
- 4.1 Definice zdravotnických databází . . . 27
- 4.2 Cíle tvorby zdravotnických databází . . . 27
- 4.3 Proces tvorby zdravotnické databáze . . . 27

4.3.1 Definice cílů a požadavků . . . 27
4.3.2 Získávání dat . . . 28
4.3.3 Normalizace a Standardizace . . . 28
4.3.4 Správa kvality dat . . . 28
4.3.5 Návrh a implementace databázové struktury . . . 28
4.3.6 Zabezpečení a ochrana dat . . . 28
4.3.7 Rozhraní a přístupová práva . . . 28
4.3.8 Aktualizace a údržba . . . 29
4.4 Výzvy při tvorbě zdravotnických databází . . . 29
4.4.1 Bezpečnost . . . 29
4.4.2 Neúplnost dat . . . 29
4.5 Využití zdravotnických databází . . . 29
4.6 Zabezpečení a ochrana osobních údajů aplikace Czech Salivary Gland Database . . . 30
4.6.1 Způsob zajištění ochrany dat a práce s nimi v rámci aplikace 30
4.6.2 Přidávání nově diagnostikovaných pacientů . . . 30
4.6.3 Retrospektivní přidávání pacientů . . . 30

5 Rozšíření aplikace o vizualizaci dat 31
5.1 Kaplan-Meierova metoda . . . 31
5.1.1 Matematické odvození . . . 31
5.1.2 Cenzurování dat . . . 32
5.1.3 Předpoklady pro cenzuru . . . 32
5.1.4 Ukázka výpočtu . . . 33
5.1.5 Log-rank test . . . 33
5.1.6 Příklad výpočtu . . . 33
5.1.7 Vyhodnocení log-rank testu . . . 34
5.2 Návrh implementace vizualizace . . . 34

6 Implementace kódu 37
6.1 Rozdělení aplikace do implementačních fází . . . 37
6.2 První iterace . . . 38
6.2.1 Inicializace aplikace . . . 38
6.2.2 Inicializace CI/CD . . . 40
6.2.3 Kostra uživatelského rozhraní a základní menu . . . 41
6.2.4 Komponenta add-patient . . . 41
6.2.5 Výsledek první iterace . . . 42
6.3 Druhá iterace . . . 42
6.3.1 Komponenta name-gland-form . . . 42
6.3.2 Výsledek druhé iterace . . . 44

6.4 Třetí iterace . . . 44
6.4.1 Komponenta PatientsList . . . 44
6.4.2 Výsledek třetí iterace . . . 46
6.5 Čtvrtá iterace . . . 46
6.5.1 Komponenta AddStudy . . . 46
6.5.2 Komponenta StudiesList . . . 47
6.5.3 Výsledek čtvrté iterace . . . 48
6.6 Pátá iterace . . . 49
6.6.1 Instalace databáze . . . 49
6.6.2 Připojení k databázi . . . 49
6.6.3 Vytvoření tabulek databáze . . . 49
6.6.4 Definice základních operací s databází . . . 50
6.6.5 Komunikace s jednotlivými tabulkami . . . 50
6.6.6 Propojení uživatelského rozhraní s databází . . . 51
6.6.7 Využití IPC komunikace ve vykreslovacím procesu . . . 52
6.6.8 Filtrování pacientů . . . 53
6.6.9 Export a import pacientů . . . 54
6.6.10 Výsledek páté iterace . . . 55
6.7 Šestá iterace . . . 55
6.7.1 Zabezpečení aplikace . . . 55
6.7.2 Implementace vizualizace křivek přežití . . . 58
6.7.3 Oprava nalezených chyb . . . 60
6.7.4 Výsledek šesté iterace . . . 60
6.8 Sedmá iterace . . . 60
6.8.1 Změna pořadí tlačítek pro vytváření nových pacientů a studií 60
6.8.2 Přejmenování položek formuláře . . . 60
6.8.3 Výsledek sedmé iterace . . . 61
6.9 Testování aplikace . . . 61
6.9.1 Manuální testování . . . 61
6.9.2 Jednotkové testy . . . 61

7 Nasazení aplikace 63
7.1 Distribuce aplikace . . . 63
7.1.1 Distribuce pro macOS . . . 63
7.1.2 Distribuce pro Linux . . . 63
7.2 Sdílení instalačního souboru . . . 63
7.3 Instalace aplikace . . . 64
7.3.1 První spuštění aplikace . . . 64

8 Zhodnocení výsledků **65**
8.1 Zpětná vazba k finální verzi aplikace 65
8.1.1 Testování na reálných datech 65
8.2 Dosažené výsledky 66
8.2.1 Oprava nedostatků 66
8.2.2 Přidání vizualizace dat 66
8.3 Rozšíření aplikace mezi další lékaře 66
8.3.1 Možnosti budoucích úprav aplikace 66

9 Závěr **69**

A Uživatelský manuál **71**

B Obsah přílohy 77

Bibliografie **79**

Seznam obrázků **81**

Seznam tabulek **83**

Seznam výpisů **85**

Seznam zkratek **87**

Úvod 1

Rakovina slinných žláz je velmi závažné a nebezpečné onemocnění, které má zásadní vliv na kvalitu života postižených pacientů. Vzhledem ke komplexnosti léčby je nezbytné, pro lékaře zabývající se touto problematikou, mít k dispozici efektivní nástroj pro shromažďování, sledování a následnou analýzu dat pacientů s tímto onemocněním. Jedním z těchto nástrojů je Czech Salivary Gland Database (CSGDB), který je vyvíjen na Západočeské univerzitě v Plzni.

Czech Salivary Gland Database je desktopová aplikace, která zjednodušuje lékařům systematické shromažďování nezbytných klinických dat o pacientech s nádory slinných žláz. Toto usnadnění sběru dat umožňuje lékařům vynaložit větší úsilí samotné analýze a péči o pacienty.

Cílem této bakalářské práce je rozšíření možností využití aplikace Czech Salivary Gland Database v klinické praxi a pro analýzu nashromážděných dat. Důraz bude kladen na pečlivou analýzu aplikace s cílem identifikovat stávající nedostatky. Následně budou navržena a implementována opatření, která vyřeší tyto nedokonalosti.

Práce se také zabývá problematikou tvorby zdravotnických databází a jejich využití pro klinické a výzkumné účely.

Důležitou součástí bude rozšíření aplikace o vizualizaci tzv. křivek přežití pro vybrané skupiny pacientů pomocí Kaplan-Meierovi metody.

Dosažená výsledná aplikace bude nasazena na pracovišti Fakultní nemocnice v Motole, kde bude sloužit jako nástroj pro lékaře při diagnostice a léčbě pacientů s nádory slinných žláz. Zároveň bude přínosem pro další výzkum v oblasti onkologie hlavy a krku.

Práce se v první části nejprve zabývá seznámením s aplikací, analýze současných nedostatků a návrhů jejich řešení. Následně je přiblížena problematika tvorby zdravotnických databází a jejich využití pro klinické a výzkumné účely. V závěru je popsán princip vizualizace dat pomocí Kaplan-Meierovi metody a návrh implementace.

Druhá část práce se zabývá implementací navržených řešení a nasazení aplikace. Na závěr je provedena analýza dosažených výsledků a zhodnocení celé práce.

Seznámení s aplikací Czech Salivary Gland Database — 2

2.1 Hlavní účel aplikace

Aplikace vznikla ve spolupráci s panem doktorem MUDr. Davidem Kalfeřtem, Ph.D., v rámci semestrální práce z předmětu Základy softwarového inženýrství (KIV/Z-SWI). Byla vytvořena studenty Mikulášem Machem, Viktorem Havlíkem a Vojtěchem Jelínkem. Jejím hlavním účelem bylo ulehčení a urychlení shromažďování klinických dat pacientů trpících nádory slinných žláz.

2.2 Uživatelské rozhraní

Uživatelské rozhraní bylo strukturováno do dvou hlavních částí, což zajišťuje snadnou navigaci a rychlý přístup k nezbytným funkcím.

V levé části obrazovky se nachází menu obsahující klíčové funkce aplikace. Zde mohou uživatelé přistupovat k základním operacím, jako je zobrazení seznamu pacientů, přidání nového pacienta, zobrazování studií, přidání nové studie a exportování dat o pacientech.

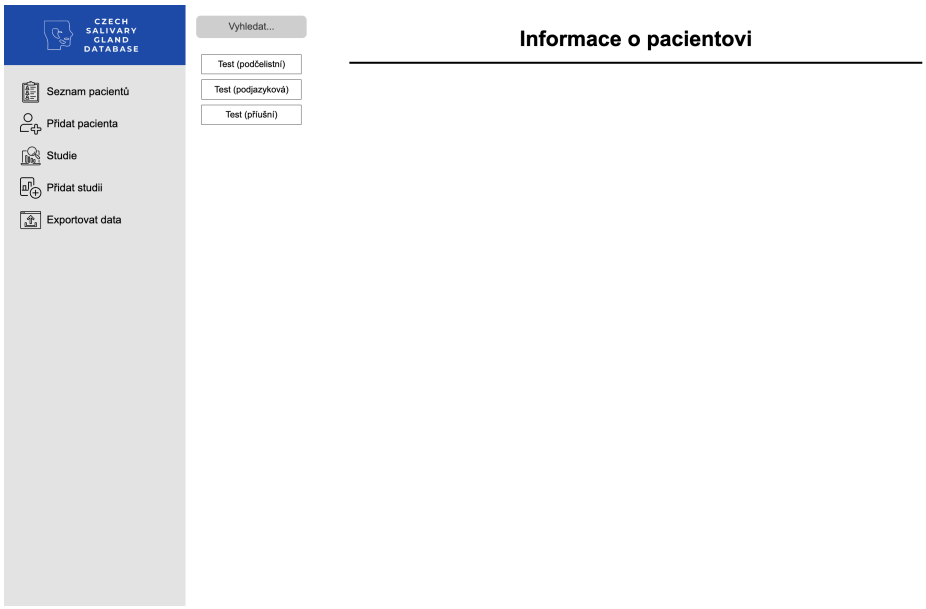
Centrální část obrazovky poskytuje detaily týkající se aktuálně vybrané části z menu. Zde jsou prezentovány všechny další položky a informace, které souvisejí s danou funkcí.

Zmíněné uživatelské rozhraní je vidět na obr. 2.1 na str. 8.

2.2.1 Seznam pacientů

Tato část menu představuje seznam obsahující jména, příjmení a typ postižené žlázy všech pacientů, kteří jsou uloženi v databázi. Zadáním jména a příjmení do vyhledávacího pole na vrcholu seznamu je možné vyhledat konkrétního pacienta.

Po jednoduchém kliknutí na rámeček obsahující jméno a příjmení pacienta jsou uživateli zobrazeny detailní informace ve formuláři.



Obrázek 2.1: Základní rozložení uživatelského rozhraní

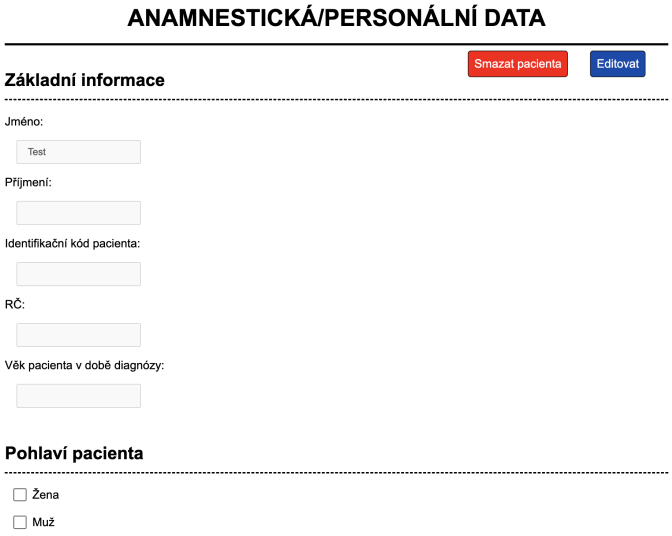
Detail pacienta ze seznamu

V detailu pacienta se otevírá rozsáhlý pohled na všechny relevantní aspekty sledované žlázy, kde každá položka formuláře má svůj definovaný prostor. Pokud je ke konkrétní položce v databázi uložen záznam, je tato informace zobrazena. V opačném případě zůstává příslušná položka prázdná, což poskytuje jednoduchý přehled o tom, které údaje jsou již v databázi uloženy. Detail je vidět na obr. 2.2 na str. 9.

Uživatel má v této části uživatelského rozhraní možnost aktivně spravovat záznamy pacientů. Během zobrazení detailu pacienta je zřetelně vizualizováno, zda probíhá editace, či nikoliv, díky zašedlým prvkům formuláře, což je také výchozí stav formuláře.

Tlačítko pro editaci odstraňuje zašedlé efekty, což signalizuje připravenost k editaci informací o pacientovi. Uživatel může provedené změny potvrdit stisknutím tlačítka **Uložit změny**, což zajišťuje aktualizaci záznamu v databázi. V případě, že uživatel nechce uložit provedené změny, existuje možnost stornovat je pomocí tlačítka **Zrušit editaci**.

Při rozhodnutí smazat pacienta se uživateli otevírá vyskakovací okno, které provádí kontrolu, zda je smazání skutečně zamýšleným krokem. Až po potvrzení odstranění je pacient definitivně vymazán z databáze, což zajišťuje bezpečnost a neomylnost této akce.



Obrázek 2.2: Detail pacienta s jménem Test

2.2.2 Přidání pacienta

Část menu určená k přidávání pacientů podle postižené žlázy poskytuje uživatelům možnost výběru ze tří specifických typů žláz: příušní, podjazyková a podčelistní. Každá z těchto žláz je reprezentována příslušným tlačítkem.

Po výběru konkrétní žlázy se na obrazovce zobrazí formulář, který slouží k detailnímu vyplňování všech relevantních informací o pacientovi s postižením zvolené žlázy.

Formulář pro přidání pacienta

Každý z formulářů je systematicky strukturován do osmi klíčových částí, které zajišťují komplexní a důkladné zaznamenání informací, jednotlivé částí jsou uvedeny v tab. 2.1.

Tabulka 2.1: Části formuláře pro přidání pacienta

název	**popis**
Anamnestická/Personální data	Základní informace o pacientovi.
Diagnostika	Informace o diagnostice nádoru.
Terapie	Informace o léčbě nádoru.
Histopatologie	Popis charakteristik nádoru.
TNM klasifikace	Stádium rakoviny podle T, N, M.
Přílohy	Doplňující informace k pacientovi.
Poznámky	Záznamy a poznámky o pacientovi.
Studie	Zařazení pacienta do studií.

2.2.3 Seznam studií

Výpis studií je strukturován obdobně jako seznam pacientů, viz obr. 2.1 na str. 8. S výjimkou toho, že první sloupec obsahuje jména a typy všech vytvořených studií. Po rozkliknutí konkrétní studie se v dalším sloupci zobrazí seznam všech pacientů zařazených do této studie. U každé studie má uživatel možnost upravit její název a také ji odstranit.

2.2.4 Přidání studie

Při vytváření nové studie nejdříve uživatel vybere, o který typ studie se jedná. V aplikaci jsou k dispozici čtyři typy studií: příušní, podjazyková, podčelistní a speciální.

Po vybrání typu studie uživatel definuje její název a vybírá pacienty, kteří budou do této studie zařazeni. Zobrazeni jsou pouze pacienti s nádorem daného typu žlázy. Speciální studie jsou flexibilní a umožňují začlenit pacienty s libovolným typem nádoru.

2.2.5 Exportovat data

Proces exportu dat probíhá následujícím způsobem:

1. Uživatel zahájí proces exportování dat kliknutím na tlačítko **Seznam pacientů** v hlavním menu.

2. V zobrazeném seznamu pacientů uživatel vybere specifické položky pomocí kombinace držení klávesy Shift a následného klikání na pacienty, které chce zařadit do exportu.

3. Následně uživatel zahájí export stisknutím tlačítka **Exportovat data**.

4. V poslední fázi exportu má uživatel možnost specifikovat soubor, do kterého budou data exportována.

Tento postup zajišťuje uživateli flexibilitu a kontrolu nad exportem, umožňující selektivní výběr pacientů a specifikaci cílového umístění dat.

Formát exportovaných dat

Data jsou exportována ve formátu `.xlsx` a to takovým způsobem, že pro každou žlázu je vytvořen jeden soubor, který obsahuje všechny informace o každém pacientovi s nádorem dané žlázy.

2.3 Analýza zdrojového kódu aplikace

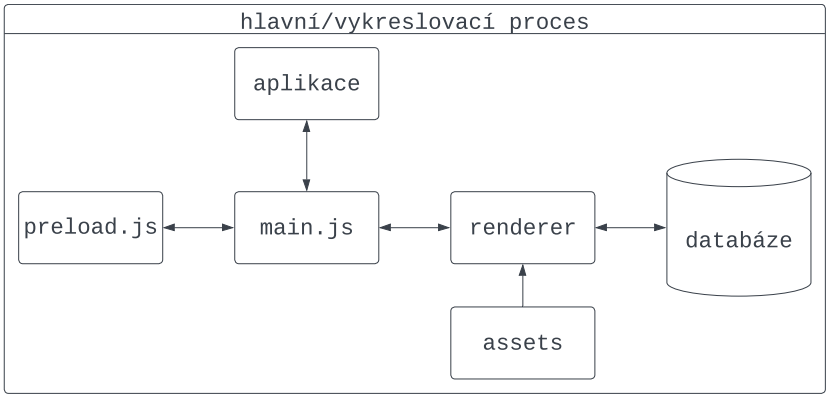
Aplikace využívá skriptovací jazyk JavaScript ve spojení s frameworkem Electron.js, což umožňuje integraci webových technologií, jako HTML, CSS a JS, pro vývoj desktopových aplikací. Tato moderní technologie nabízí několik výhod, z nichž jednou z klíčových, je rychlý vývoj a snadná tvorba multiplatformních aplikací. To umožňuje jednoduché nasazení aplikace na různé operační systémy s minimálním úsilím.

Rychlost vývoje je jednou z výrazných předností této technologie. Díky využití webových technologií je možné efektivně využívat široké spektrum nástrojů, knihoven a frameworků dostupných pro webový vývoj. To přispívá k rychlé implementaci nových funkcí a snadné údržbě kódu.

Další významnou výhodou je jednoduchá tvorba multiplatformních aplikací. Framework Electron.js umožňuje zapouzdření webových technologií do spustitelných souborů, které lze nasadit na různé operační systémy. Tím dochází k efektivnímu snížení nákladů na vývoj a údržbu, neboť stejný kód lze opakovaně využít napříč různými platformami.

Přestože tato technologie nabízí mnoho výhod, je důležité zdůraznit některé její nevýhody. V porovnání s jinými nástroji, které jsou optimalizované pro vývoj na konkrétní platformy operačních systémů, může Electron.js vyžadovat vyšší paměťové a výpočetní prostředky. Toto může být zásadní pro aplikace, které vyžadují vysoký výkon nebo jsou závislé na efektivním využití systémových prostředků. Kromě toho může být kontrola nad hardwarovými zdroji a systémovými voláními omezenější ve srovnání s nativními jazyky. [Mar24]

2.3.1 Struktura aplikace



Obrázek 2.3: Struktura aplikace Czech Salivary Gland Database

Aplikace je strukturována do jednoho hlavního modulu `renderer` a několika pomocných souborů a adresářů. Struktura je znázorněna na obr. 2.3 na str. 11.

2.3.2 Vstupní bod aplikace

Vstupním bodem každé aplikace postavené na Electronu je její hlavní skript s názvem `main.js`. Tento skript ovládá hlavní proces, který běží v prostředí Node.js a je zodpovědný za řízení životního cyklu aplikace, zobrazování nativních rozhraní, provádění privilegovaných operací a správu procesů vykreslovače [Ele23].

Během interpretace kódu aplikace framework Electron.js vyhledá tento skript podle názvu nastaveném v atributu `main`, který se nachází v souboru obsahujícím konfiguraci aplikace `package.json`. Tato vlastnost umožňuje flexibilní pojmenování hlavního skriptu, což přispívá k větší adaptabilitě aplikace..

V aplikaci Czech Salivary Gland Database je hlavní skript pojmenovaný podle klasické konvence jako `main.js`. Jeho hlavním úkolem je vytvoření okna, ve kterém jsou vykresleny všechny prvky obsažené v aplikaci. Dále komunikuje s modulem `renderer`, který obsahuje veškerou logiku a zajišťuje komunikaci s databází.

2.3.3 Základní funkce souboru main.js

Hlavním účelem souboru `main.js` je řízení základních operací aplikace, především vytváření hlavního okna, exportování dat a přidávání příloh.

Funkce createWindow()

Funkce `createWindow()` slouží k vytvoření hlavního okna aplikace. Toto okno má nastavené rozměry a povoluje integraci s Node.js API, což umožňuje modulu **renderer** přistupovat k funkcím Node.js. Dále je určena cesta k souboru, který přednačte potřebné funkce pro modul **renderer**, a nakonec je načten soubor `index.html`, obsahující všechny prvky uživatelského rozhraní aplikace. Ukázka této funkce je uvedena v kódu 2.1.

Zdrojový kód 2.1: Funkce pro vytvoření okna

```
1 const createWindow = () => {
2     win = new BrowserWindow({
3         // šířka okna
4         width: 1000,
5         // výška okna
6         height: 600,
7         webPreferences: {
8             // povolení integrace NodeJS
9             nodeIntegration: true,
```

```
10              // cesta k souboru pro přednačtení funkcí
11              preload: path.join(__dirname, 'preload.js'),
12          }
13      })
14      // načtení HTML souboru
15      win.loadFile('renderer/index.html');
16  };
```

Funkce exportData()

Úkolem této funkce je exportování dat vybraných pacientů. Po stisknutí tlačítka **Exportovat data** v menu je vyvolána. Nejprve získá všechny vybrané pacienty a následně je exportuje do souboru ve formátu `.xlsx`. Tento soubor je uložen na cestě, kterou uživatel specifikuje v dialogovém okně.

Funkce attachmentsWindow()

Poslední klíčovou funkcí v souboru `main.js` je `attachmentsWindow(data)`. Funkce zajišťuje vytvoření okna, prostřednictvím kterého může uživatel vybírat soubory ze svého úložiště a následně je přidávat jako přílohy k informacím o pacientovi.

2.3.4 Modul renderer

Všechny prvky vizuálního designu, komunikace s databází a obsluha událostí, jsou soustředěny v modulu `renderer`, který se skládá z souboru `index.html` a tří složek: `css`, `img` a `js`.

Soubor `index.html` definuje strukturu grafického uživatelského rozhraní aplikace, zatímco složky `img` a `css` přidávají stylové prvky, které ovlivňují celkový vizuální dojem. Programová logika aplikace je obsažena ve složce `js`.

V následujících kapitolách budou podrobněji popsány submoduly a soubory obsažené v modulu `renderer`, které jsou uvedeny v tab. 2.2.

Tabulka 2.2: Moduly obsaženy v modulu `renderer`

název	**typ**	**popis**
`calculations`	modul	Výpočet TNM klasifikace a balíčkoroků.
`formHandlerer`	modul	Získávání dat z formulářů.
`listeners`	modul	Reagování na události.
`models`	modul	Komunikace s databází.
`processing`	modul	Vykreslování formulářů.
`renderer.js`	soubor	Načítání prvků z `index.html` a přidávání obsluhy událostí k nim.

Soubor renderer.js

Soubor napsaný v jazyce JavaScript má za úkol načítat všechny prvky z `index.html`, které vyžadují nějakou formu obsluhy událostí při interakci s nimi (např. stisknutí tlačítka v menu). Po úspěšném načtení všech prvků jsou k nim přidány tzv. „event listenery", což jsou funkce sloužící k naslouchání, zda nad kterýmkoliv prvkem došlo k vyvolání libovolné události a jsou uvedeny v tab. 2.3. Tyto naslouchače událostí byly importovány prostřednictvím již dříve zmíněné funkce `createWindow`, která je zobrazena v ukázce zdrojového kódu 2.1 na straně 12. Zmíněné naslouchače jsou součástí modulu `listeners`.

Tabulka 2.3: Importované listenery do `renderer.js`

název listeneru	**pozorovaná změna**
`startDbListener`	Komunikace s databází.
`startAttachmentsChangeListener`	Přídávání příloh.
`glandListener`	Přidávání pacienta.
`addMenuButtonsListeners`	Tlačítka v menu.
`exportListener`	Exportování pacientů.
`addSearchListener`	Vyhledávání pacientů.
`showStudiesListener`	Zobrazení studií.

Submodul calculations

Při vyplňování informací o pacientovi jsou zaznamenány obecné rizikové faktory, jako je kouření a nadužívání alkoholu. Při stanovení intenzity kouření pacienta se používá jednotka nazývaná balíčkorok. Balíčkorok představuje termín, který se používá při odhadování množství cigaret vykouřených jednotlivcem během delšího časového úseku. Například, 20 balíčkoroků značí, že osoba kouřila (nebo kouří) po dobu 20 let jeden balíček cigaret denně [ČR]. Automatický výpočet této hodnoty podle počtu cigaret denně a doby, po kterou pacient kouřil, je řízen funkcí `calculateCigars()`. Funkce implementuje vztah 2.1, kde číslo 20 představuje průměrný počet cigaret v jednom balíčku.

$$\text{balíčkoroky} = \frac{\text{počet cigaret}}{20} \cdot \text{doba užívání (roky)} \tag{2.1}$$

Další informací, u které je vhodné provádět automatický výpočet, je TNM klasifikace. Tato hodnota určuje stádium rakoviny v daném okamžiku.
Funkce `calculateTNMClassification` realizuje výpočet podle tab. 2.4 na str. 15.

Klasifikace TNM je systém určený pro hodnocení rozsahu nádorového onemocnění. Symbol T reprezentuje velikost nebo rozsah primárního nádoru a může nabývat hodnot 0, 1, 2, 3, 4a, 4b, kde každá hodnota odpovídá určitému stupni šíření

Tabulka 2.4: TNM klasifikace

Stádium	Hodnota T	Hodnota N	Hodnota M
0	0	0	0
I	1	0	0
II	2	0	0
III	3	0	0
	1, 2, 3	1	0
IVa	1, 2, 3	2	0
	4a	0, 1, 2	0
IVb	4b	jakákoliv	0
	jakákoliv	3	0
IVc	jakákoliv	jakákoliv	1

nádoru. Symbol N označuje přítomnost nebo nepřítomnost metastáz v regionálních mízních uzlinách, a to s hodnotami 0, 1, 2a, 2b, 3a, 3b. Symbol M zaznamenává přítomnost nebo nepřítomnost vzdálených metastáz, nabývajíc hodnot 0, 1. Pan doktor MUDr. David Kalfeřt, Ph.D., poskytl zmíněné informace ohledně TNM klasifikace.

Pro ilustraci, hodnota T4 v klasifikaci T znamená, že pod ní spadají varianty označené T4a i T4b. Toto pravidlo platí analogicky i pro ostatní části klasifikace TNM, kde specifické hodnoty označují specifické charakteristiky nádorového růstu a metastázování.

Submodul formHandlerer

Při tvorbě nebo úpravě záznamů o pacientech je nezbytné získat všechna vyplněná data z formulářů. Získávání dat je řešeno pomocí modulu `formHandlerer`, konkrétně v souboru `formGetrs.js`, který obsahuje základní funkce pro získávání dat.

V uvedeném souboru se nachází několik funkcí, z nichž každá je zaměřena na získání dat specifického typu položky ve formuláři, například ze zaškrtávacího pole nebo textového vstupu. Ukázka této funkcionality bude představena na příkladu zdrojového kódu 2.2. I když je zde reprezentována pouze jedna funkce, je třeba poznamenat, že ostatní jsou koncipovány podobným způsobem.

Zdrojový kód 2.2: Funkce pro získání dat z formuláře

```
1 function getCheckBoxValue(form, id){
2     // inicializace výsledné hodnoty
3     let value = "";
4     // získání divu s daným id
5     const div = form.querySelector('#' + id);
6     // získání divu se zaškrtávacími boxy
7     const checkboxDivs = div.querySelectorAll('div');
8
```

```
9       checkboxDivs.forEach(checkBoxDiv => {
10          if(checkBoxDiv.classList.contains('nestedDivs')) {}
11          else {
12              // získání jednoho boxu
13              const checkBox = checkBoxDiv.querySelector('input');
14
15              if(checkBox.checked){
16                  // uložení hodnoty
17                  value = checkBoxDiv.querySelector('p').textContent
                        ;
18              }
19          }
20      });
21
22      return value;
23  }
```

Funkce `getCheckBoxValue(form, id)` identifikuje a načte do proměnné `div` element formuláře označený specifikovaným `id`. Následně provádí výběr všech prvků v daném elementu a iteruje přes ně pomocí smyčky. Během této iterace získává požadované hodnoty a ukládá je do proměnné `value`. Nakonec vrací tuto sestavenou hodnotu.

Další velmi důležitou části tohoto modulu je `formPodcelistni.js`, který využívá funkce nadefinované v `formGetrs.js`. Nejprve extrahuje hodnoty z formuláře příslušícího k podčelistní žláze a ukládá je do příslušných proměnných. Následně jsou tyto hodnoty předány funkci `insertPodcelistniPromise()`, která nese zodpovědnost za vložení získaných dat do databáze. Analogicky jsou vytvořeny další dva soubory pro získávání dat z formulářů patřících podjazykové a příušní žláze.

Obdobným způsobem jsem vytvořeny funkce pro aktualizování dat v databázi. Tyto funkce jsou obsaženy v souborech `updateForm<název_žlázy>.js`.

Submodul listeners

Aplikace vyžaduje odpovědi na události, které vznikají v důsledku různých interakcí s prvky. Správu těchto událostí zajišťuje modul `listeners`, který obsahuje několik funkcí, z nichž ty nejdůležitější budou popsány v následujícím textu.

Funkce `addMenuButtonListeners` přidává obsluhu událostí vyvolaných stiskem tlačítek v menu a definuje, které elementy mají být zobrazeny. Při stisku tlačítek **Seznam pacientů** a **Studie** jsou také zavolány funkce, které provádějí načítání pacientů nebo studií z databáze. Funkce pro načítání pacientů je vidět ve zdrojovém kódu 2.3 na str. 17. Analogicky je implementována i funkce pro načítání studií.

Zdrojový kód 2.3: Funkce pro načítání pacientů

```
1 listPatient.addEventListener("click", async(e) => {
2     await loadAndDisplayPatients();
3 })
```

Druhou velmi zásadními funkcí v tomto modulu je `exportListener()`, která potřebuje pro vykonání svého kódu komunikovat s již zmíněným hlavním procesem celé aplikace `main.js`. Tato komunikace je realizována prostřednictvím objektu `ipcRenderer`, který je součástí frameworku Electron.js. Pomocí metody `send()` je možné poslat zprávu hlavnímu procesu, který na ni následně může reagovat. V tomto případě je zpráva odeslána v okamžiku, kdy uživatel stiskne tlačítko **Exportovat data**. Zpráva obsahuje informace o tom, kteří pacienti mají být exportováni. Hlavní proces následně zavolá funkci `exportData()`, která je obsažena v souboru `main.js`. Funkce zajišťuje samotný export dat do souboru.

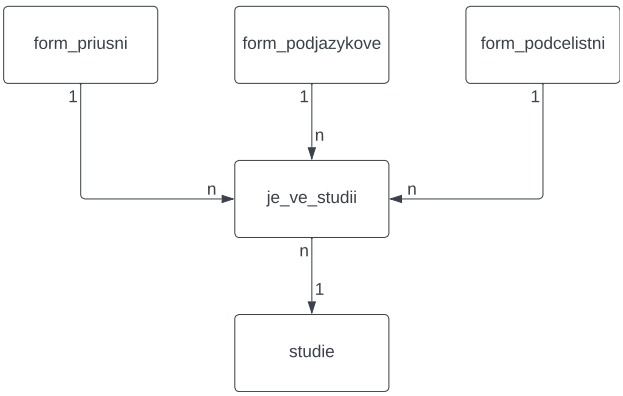
Nejdůležitější funkcí je `addGlandListener()`, která je obsažena v souboru `addGlandListener.js`. Funkce zajišťuje vykreslení formuáře pro přidání pacienta do databáze. Při vykonávání této funkce je načten formulář reprezentující žlázu, pro kterou chce uživatel přidat pacienta. Formulář je načten z modulu `assets`. Následně je formulář vykreslen do hlavního okna aplikace.

Submodul models

Hlavním úkolem modulu `models` je komunikace s databází a získávání relevantních pacientských dat. Aplikace využívá SQLite databázi pro úschovu a správu informací o pacientech. Klíčovými vlastnostmi této databáze jsou její malé rozměry, vysoká rychlost a hlavně to, že ke svému běhu nepotřebuje externí server.

Pro komunikaci s danou databází je využita knihovna `node-sqlite3`, která je importována do souboru `dbConnection.js`. Po importu této knihovny je vytvořeno pět tabulek: **from_podcelistni**, **from_podjazykovy**, **from_priusni**, **studie** a **je_ve_studii** zjednodušený ER model databáze je vidět na obr. 2.4 na str. 18. Tabulky slouží k ukládání a strukturování informací o pacientech a studiích.

V rámci modulu jsou také obsaženy soubory `studieManager.js` a `<nazev_zlazy>Manager.js`. Soubory mají za úkol usnadnit manipulaci s daty v databázi a zahrnují operace pro vkládání, aktualizaci, získávání a mazání dat. Při získávání informací o pacientovi z databáze jsou data reprezentována formou Data Transfer Object (DTO) neboli přepravka, která je definována třídou `Patient` v souboru `Patient.js`. Třída obsahuje všechny společné informace o pacientovi. Dále od ní dědí třídy `Patient<nazev_zlazy>.js`, které obsahují specifické informace o pacientovi v kontextu dané žlázy.



Obrázek 2.4: Zjednodušený ER model databáze

Submodul processing

Modul obsahuje soubory a funkce, která mají za úkol vykreslování formulářů. Každý formulář je detailně reprezentován samostatným souborem, jenž je strukturován ve formátu `.json`. Soubor následně organizuje obsah formuláře do sekcí a subsekcí, přičemž zpracování těchto částí je realizováno skrze specializované funkce `processSection()` a `processSubSection()`.

Funkce `processSection()` postupně prochází jednotlivé sekce uvedené v souboru s příponou `.json`, přičemž každou sekci předává k dalšímu zpracování funkci `processSubSection()`. Následná funkce má za úkol vykreslit individuální prvky formuláře na základě specifikací dané subsekce a typu formuláře.

Analýza současných nedostatků — 3

3.1 Identifikace nedostatků

Nedostatky aplikace jsou rozděleny do dvou hlavních skupin. První skupina obsahuje nedostatky týkající se funkcionality aplikace, uživatelského rozhraní a celkového zážitku z používání aplikace, druhá skupina se zaměřuje na nedostatky v implementaci a struktuře kódu.

3.2 Nedostatky functionality aplikace

Nedostatky ve funkcionalitě a uživatelském rozhraní jsou rozděleny do tří kategorií podle stupně závažnosti. Kategorie jsou uvedeny v tab. 3.1.

Tabulka 3.1: Kategorizace nedostatků

stupeň závažnosti	**popis**
Vysoký	Aplikace by byla obtížná až téměř nevyhovující pro další používání.
Střední	Aplikace by se dala nadále používat, ale s jistými komplikacemi.
Nízký	Aplikaci by bylo možné dále používat téměř bez komplikací.

3.2.1 Vysoký stupeň závažnosti

V této části textu budou uvedeny všechny nedostatky s vysokým stupněm závažnosti a následně bude navrženo jejich řešení. Realizace řešení se poté bude nacházet v kapitole věnující se implementaci kódu.

Filtrace pacientů

V současné verzi aplikace není možné filtrovat pacienty podle různých položek, které jsou vyplněné v jejich formuláři. To znamená, že uživatelé nemají možnost

vyhledávat pacienty podle jejich data narození, diagnózy a dalších důležitých parametrů. Tudíž je velmi obtížné hledat pacienty, kteří splňují několik určitých kritérií zároveň. To velice komplikuje práci s daty pacientů.

Například pokud by chtěl uživatel vyhledat všechny pacienty s nádorem slinné žlázy, pod třicet let a následně je exportovat do souboru, musel by procházet všechny pacienty ručně a hledat ty, kteří splňují tato kritéria.

Po konzultaci s panem doktorem MUDR. Davidem Kalfeřtem, Ph.D. bylo navrženo řešení tohoto problému. Aplikace bude mít v části seznamu pacientů nad vstupním polem pro vyhledávání tlačítko **Filtrovat**, které po stisku zobrazí nové okno, ve kterém bude možné nastavit jakým způsobem budou pacienti filtrování. Při domluvě s panem doktorem bylo rozhodnuto, že filtrování bude probíhat na základě typu žlázy, terapie a histopatologického typu nádoru. Tato varianta byla zvolena z důvodu, že tyto parametry jsou nejdůležitější pro budoucí uživatele aplikace a eliminuje se takto komplikace se zbytečně složitým filtračním menu, ve kterém by se filtrace podle jiných parametrů využívala minimálně.

Export celých studií

Uživatelé mohou při používání aplikace vytvářet studie, které obsahují pacienty s nějakými specifickými vlastnostmi. V aktuální verzi programu není možné pomocí jedné operace (například stisknutí tlačítka) exportovat všechny pacienty, kteří jsou v dané studii obsaženy. To znamená, že uživatelé následně nemohou jednoduchým způsobem vytvořit soubory ve formátu `.xlsx`, pro další analýzu nasbíraných dat, což výrazně potlačuju celkové využití aplikace, jelikož vytváření a exportování studií je jednou z nejdůležitějších funkcí.

Pro vyřešení tohoto problému bylo navrženo násedující řešení. V seznamu studií, při rozkliknutí konkrétní studie, bude na vrcholu seznamu pacientů zobrazeno tlačítko **Vybrat vše**, které po stisku vybere všechny pacienty v dané studii. Následně bude uživatel mít možnost stisknout tlačítko **Exportovat**. Navržené řešení poté bude kompatibilní s řešením úpravy způsobu vybíraní pacientů k exportu, které je zmíněno v kapitole s nedostatky středního stupně závažnosti.

Odebírání pacientů pouze ze studie

Současná funkcionalita aplikace umožňuje pouze mazání pacientů z databáze. To znamená, že pokud chce uživatel smazat pacienta pouze ze studie není mu to umožněno. Zmíněná vlastnost není příliš praktická, protože neumožňuje uživatelům upravovat obsah studií, což je velmi důležité pro dlouhodobé užívání aplikace.

Řešení problému spočívá v tom, že pokud uživatel vybere nějakého pacienta, který je ve studii tak v části okna, kde se nyní vyskytuje tlačítko **Smazat pacienta**,

přibude tlačítko **Odebrat pacienta ze studie**. Po stisku tlačítka bude pacient smazán pouze z dané studie a ne z celé databáze.

Anonymizovaný export

Při sdílení dat mezi lékaři je velmi důležité, aby osobní údaje pacientů, jako je jméno, příjmení a rodné číslo, byly anonymizovány. V současné verzi aplikace není tato funkcionalita implementována, což může způsobit problémy s GDPR.

Uvedený nedostatek bude vyřešen přidáním nového tlačítka **Exportovat anonymizovaně**, které se bude nacházet vedle tlačítka **Exportovat**. Po stisku tlačítka budou všechny osobní údaje pacientů anonymizovány a následně budou data exportována do souboru ve formátu `.xls`.

Zabezpečení aplikace

V aplikaci se nachází velice citlivé osobní informace o pacientech, které by měly být chráněny před neoprávněným přístupem. Vzhledem k tomu, že program bude používán pouze na osobních počítačích lékařů a nebude připojen k internetu, tak jediný způsob, jakým by mohlo dojít k neoprávněnému přístupu k datům, je fyzický přístup k počítači. Z tohoto důvodu je nutné, aby aplikace vyžadovala heslo při jejím spouštění a zároveň musí být uložená data v databázi šifrována. Avšak v současné verzi aplikace není funkcionalita implementována.

Pro zabezpečení aplikace bude vytvořeno vyskakovací okno, které se zobrazí při spuštění aplikace. Zmíněné okno bude vyžadovat heslo k odemknutí aplikace a klíč k dešifrování dat v databázi. Po správném zadání hesla a klíče bude aplikace odemknuta a uživatel bude moci pokračovat v práci. Samotná inicializace zabezpečení bude probíhat při úplně prvním spuštění aplikace, kdy uživatel bude vyzván k zadání hesla a bude pro něj vygenerován dostatečně dlouhý a bezpečný klíč k dešifrování dat v databázi. Vygenerovaný klíč si musí uživatel zaznamenat na bezpečné místo, jelikož bez něj nebude možné přistoupit k osobním údajům o pacientech v databázi. Data budou zašifrována pomocí AES šifrování, které je jedním z nejbezpečnějších symetrických šifrovacích algoritmů.

Zadávání hesla a klíče může být velmi zdlouhavé a nepohodlné, proto bude mít uživatel možnost při vytváření zabezpečení vybrat možnost, že nechce používat zabezpečenou verzi aplikace a nebude po něm vyžadováno heslo a klíč při spuštění aplikace. Použití nezabezpečené verze bude nedoporučován a mělo by se používat pouze v případě, kdy databáze nebude obsahovat osobní údaje pacientů.

3.2.2 Střední stupeň závažnosti

V této části textu budou uvedeny všechny nedostatky se středním stupněm závažnosti. Následně bude navrženo jejich řešení a realizace bude uvedena v kapitole věnující se implementaci kódu.

Úprava způsobu vybírání pacientů k exportu

Vybírání pacientů k exportu je nyní implementováno pomocí ne zcela intuitivní metody. Pokud je uživatel v části aplikace, která obsahuje seznam pacientů a chce vybrat nějaké pacienty k exportu, musí držet klávesu Shift a jednotlivě je zaklikávat. Takové řešení není vhodné, pokud chce například uživatel exportovat všechny pacienty, kteří jsou zobrazeny v seznamu. V takovém případě musí uživatel držet klávesu Shift a klikat na všechny pacienty, které chce vybrat.

Řešením zmíněného problému je vytvoření zaškrtávacího boxu u každého pacienta v seznamu pacientů. Pokud bude uživatel chtít vybrat jen některé pacienty k exportu, bude moci zaškrtnout příslušné zaškrtávací boxy a následně stisknout tlačítko **Exportovat**. V případě exportu všech pacientů, kteří jsou zobrazeni v seznamu bude na jeho vrcholu tlačítko **Vybrat vše**, které označí všechny pacienty. Dále může uživatel pokračovat s exportem stejně jako při výběru pouze některých. Řešení bude kompatibilní s řešením exportu celých studií, které je zmíněno v kapitole s nedostatky vysokého stupně závažnosti.

Přidávání pacienta do studie při jeho editaci

Dosavadní verze aplikace neumožňuje uživateli přidat pacienta do studie při jeho editaci, tato operace je možná pouze při vytváření nového pacienta, což vede k jednomu zásadnímu problému. Pokud studie vznikne až po vytvoření pacienta, uživatel musí pacienta smazat a znovu vytvořit, aby ho mohl přidat do studie. Takový nedostatek ovšem velmi omezuje uživatele, jelikož může nastat situace, kdy v době vytvoření pacienta studie ještě neexistuje, ale vznikne až po nějakém čase.

Nedostatek bude opraven přidáním zaškrtávacích boxů do formuláře, které umožňují volbu studie při editaci pacienta.

3.2.3 Nízký stupeň závažnosti

Jako poslední jsou uvedeny nedostatky s nízkým stupněm závažnosti. Nedostatky neovlivňují funkcionalitu aplikace, ale spíše její uživatelskou přívětivost. Navržená řešení budou uvedena v této kapitole a jejich realizace bude uvedena v kapitole věnující se implementaci kódu.

Změna zobrazení náhledu příloh

Aktuálně aplikace zobrazuje přílohy jako tlačítka, které obsahují název přílohy. Což relativně zhoršuje orientaci uživatele v přílohách a to zvlášť v případě, že je jich velké množství.

Pro zjednodušení orientace uživatele v přílohách budou jednotlivé druhy příloh zobrazeny jako ikony, které budou indikovat, o jaký druh souboru se jedná a pod nimi bude napsán název přílohy.

Absence instalačního souboru

Aplikace v současné verzi neobsahuje instalační soubor a sdílí se pouze prostřednictvím komprimovaného archivu, který je třeba extrahovat a následně spustit aplikaci. Distribuce aplikace tímto způsobem je nevhodná, jelikož může způsobit problémy s uživatelskou přívětivostí.

Pro zlepšení uživatelské přívětivosti bude vytvořen instalační soubor, který umožní rychlou a jednoduchou instalaci aplikace.

3.3 Nedostatky v implementaci a struktuře kódu

Zdrojový kód aplikace obsahuje mnoho nedostatků, které nesplňují základní principy čistého kódu, jako je například oddělení zodpovědností, DRY (don't repeat yourself) princip, který říká, že by se stejný kód neměl objevovat na více místech, a mnoho dalších. Nedostatky způsobují, že kód je těžko čitelný, obtížně udržovatelný a náchylný k chybám.

Při abstraktnějším prohlédnutí kódu byli identifikovány dvě oblasti, do kterých je aplikace jako celek rozdělena. První oblast se věnuje hlavně tzv. „front-end" části aplikace, která je zodpovědná za uživatelské rozhraní a interakci s uživatelem. Druhá oblast se zabývá „back-end" částí aplikace, která zajišťuje komunikaci s databází a zpracování dat. Obě oblasti obsahují zmíněné nedostatky a jejich další prozkoumání je provedeno v následujících podkapitolách.

Dalším signifikantním problémem je špatné strukturování kódu aplikace do modulů. Všechna logika zabývající se jak front-endem, tak back-endem je obsažena v jednom modulu, což výrazně komplikuje navigaci napříč kódem aplikace.

3.3.1 Front-end

Do této části aplikace patří submoduly `listeners`, `formHandler`, `processing` a `calculations` z modulu `renderer`. Jejich nejzásadnější problémem je snaha implementovat funkce, které jsou součástí moderních frameworků pro tvorbu uži-

vatelských rozhraní, jako je například React.js nebo Angular. Tento přístup vede k velkému množství kódu, který je těžko čitelný a hlavně se snaží tzv. znovu vynalézat kolo.

Vyřešení nedostatku, který má velký vliv na celkovou kvalitu a dlouhodobou udržitelnost kódu, je přepsání těchto submodulů tak, aby využívaly moderní frameworky pro tvorbu uživatelských rozhraní. Pro tyto účely byl zvolen framework React.js, který je velmi oblíbený a má velkou komunitu, což znamená, že je velmi pravděpodobné, že bude podporován i v budoucnu a zároveň pro něj existuje nespočet dostupných online materiálů. Přepsání submodulů s využitím technologie React.js by mělo za následek zjednodušení kódu, zvýšení jeho čitelnosti a udržitelnosti a zároveň by mělo zvýšit efektivitu vývoje nových funkcí a oprav chyb.

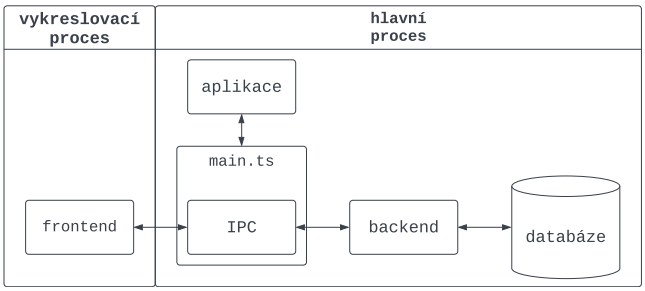
3.3.2 Back-end

Submodul spadající do části back-end je `models`, který obsahuje funkce a třídy zprostředkovávající komunikaci s databází a zpracování dat. Jeho hlavním problémem, je redundantní repetetivnost funkcí pro práci s jednotlivými tabulkami v databázi. Většina souborů v submodulu obsahuje téměř identické funkce, které se liší pouze názvem tabulky, se kterou pracují. Zmíněný přístup vede k těžko udržitelnému kódu a zvýšené pravděpodobnosti chyb.

Řešením problému je vytvoření univerzálních funkcí, které budou schopny pracovat s libovolnou tabulkou v databázi. Využití univerzálních funkcí zjednoduší kód, zvýší jeho čitelnost a udržitelnost. Zároveň bude zvýšena efektivita vývoje nových funkcí a oprav chyb.

3.3.3 Struktura kódu

Celá logika původní aplikace je obsažena v jednom modulu, což výrazně komplikuje navigaci kódem a jeho dlouhodobou udržitelnost.



Obrázek 3.1: Nová struktura aplikace Czech Salivary Gland Database

Nově navržená struktura viz obr. 3.1, se dělí do modulů `backend` a `frontend`. Modul `frontend` je začleněn do vykreslovacího procesu frameworku Electron.js

a má za úkol zobrazovat uživatelské rozhraní a interagovat s uživatelem. Zprostředkování práce s databází obstarává modul `backend`, který vykonává svůj kód v hlavním procesu. Komunikaci mezi `frontend` a `backend` zajišťuje modul `IPC`, jenž je importován pomocí souboru main.ts. Výhodou této nové architektury je zřetelné oddělení vykreslovacího a hlavního procesu.

Problematika tvorby zdravotnických databází

4

4.1 Definice zdravotnických databází

Zdravotnické databáze představují centralizované úložiště citlivých zdravotních dat pacientů. Jejich cílem je shromažďovat, uchovávat a spravovat informace relevantní pro poskytování péče, výzkum a sledování zdravotních trendů. [Lee94]

4.2 Cíle tvorby zdravotnických databází

Vytváření zdravotnických databází má několik cílů. Zdravotnické databáze umožňují lékařům a zdravotnickým pracovníkům přístup k důležitým informacím o pacientech, jako jsou jejich zdravotní záznamy, diagnózy, léky, které užívají, a další. Uložené informace mohou pomoci lékařům a zdravotnickým pracovníkům lépe porozumět zdravotnímu stavu pacienta a poskytnout jim lepší péči. Zdravotnické databáze také umožňují sledovat trendy v oblasti zdraví a pomáhají výzkumníkům a epidemiologům studovat různé nemoci a hledat způsoby, jak je lépe léčit.

4.3 Proces tvorby zdravotnické databáze

Proces tvorby zdravotnické databáze je komplexní a klíčový krok směřující k efektivnímu shromažďování, ukládání a správě zdravotnických informací. Samotný proces zahrnuje několik klíčových fází, které společně zajišťují vytvoření robustní a spolehlivé databáze pro podporu zdravotní péče a výzkumu. [Sto+06]

4.3.1 Definice cílů a požadavků

Prvním krokem je pečlivá definice cílů a požadavků, ke kterým databáze bude sloužit. To zahrnuje specifikace druhů zdravotních informací, které budou zaznamenány,

formáty dat, bezpečnostní normy a požadavky na přístupnost. Při definici cílů a požadavků je důležité úzce spolupracovat s lékaři a zdravotnickými pracovníky, kteří budou databázi používat [CCN18].

4.3.2 Získávání dat

Pro úspěšnou tvorbu databáze je nezbytné získávat data z různých zdravotnických zdrojů. To může zahrnovat elektronické zdravotní záznamy, laboratorní výsledky, lékařské zprávy a další relevantní informace. Automatizované procesy a standardy pro interoperabilitu jsou klíčové pro efektivní sběr dat.

4.3.3 Normalizace a Standardizace

Získaná data musí být normalizována a standardizována, aby byla dosažena konzistence. Normalizace zahrnuje sjednocení formátů, standardizaci kódů pro diagnostiku a procedury a další normalizační postupy.

4.3.4 Správa kvality dat

Kontrola kvality dat je nezbytným krokem pro eliminaci chyb a zajištění přesnosti a spolehlivosti informací v databázi. Automatické kontroly, auditní sledování a manuální revize jsou součástí procesu správy kvality dat.

4.3.5 Návrh a implementace databázové struktury

Na základě definovaných požadavků je navržena optimální struktura databáze. To zahrnuje vytvoření tabulek, vztahů mezi entitami a definici klíčů. Následně je struktura implementována do databázového systému.

4.3.6 Zabezpečení a ochrana dat

Bezpečnost dat je klíčovým hlediskem zdravotnických databází. Zahrnuje implementaci robustních bezpečnostních opatření, šifrování, omezení přístupu a sledování aktivit.

4.3.7 Rozhraní a přístupová práva

Uživatelská rozhraní a práva přístupu jsou navrženy tak, aby umožňovala oprávněným uživatelům snadný a bezpečný přístup k potřebným informacím. To může zahrnovat různé úrovně oprávnění a přizpůsobitelné možnosti filtrování dat.

4.3.8 Aktualizace a údržba

Zdravotnická databáze musí být pravidelně aktualizována, aby reflektovala aktuální zdravotnické trendy a normy. Pravidelná údržba zahrnuje také kontrolu integrity dat, opravy chyb a aktualizace bezpečnostních opatření.

4.4 Výzvy při tvorbě zdravotnických databází

Vytváření zdravotnických databází je proces plný výzev, které odrážejí složitost a citlivost zdravotních informací. V následujícím textu budou uvedeny nejčastější výzvy, které se vyskytují při tvorbě zdravotnických databází.

4.4.1 Bezpečnost

Bezpečnost je jednou z největších výzev při tvorbě zdravotnických databází. Zdravotnické informace jsou velmi citlivé a mohou obsahovat osobní údaje, jako jsou jméno, příjmení, datum narození, adresu, telefonní číslo, pojišťovnu, diagnózu, léky, které pacient užívá, a další. Osobní informace mohou být zneužity, pokud se dostanou do nesprávných rukou. Proto je důležité, aby byla zdravotnická data chráněna před neoprávněným přístupem a zneužitím.

4.4.2 Neúplnost dat

Zdravotnická data jsou často neúplná a obsahují chybějící informace. Neúplnost dat může být způsobena nesprávným a neúplným procesem zaznamenávání dat, nebo neschopností získat určité informace. Neúplná data ovlivňují kvalitu a přesnost zdravotnických databází a mohou vést k nesprávným závěrům a rozhodnutím.

4.5 Využití zdravotnických databází

Zdravotnické databáze představují významný nástroj pro hlubší porozumění zdravotních stavů a léčebných postupů. Jejich využití může být demonstrativně ilustrováno studií provedenou na národní populační kohortě 9388 pacientů s karcinomem hlavy a krku (HNSCC), léčených radioterapií. Zmíněná studie nikoliv pouze identifikovala výskyt komorbidit a jejich vliv na celkové přežití, ale také poskytla základ pro vytvoření specifického komorbiditního indexu pro oblast hlavy a krku (HN-CCI) [Bøj+14]. Tímto příkladem je patrné, jak zdravotnické databáze hrají klíčovou roli při formování informovanějšího přístupu k diagnostice a léčbě pacientů.

Podobně lze demonstrovat význam zdravotnických databází prostřednictvím výzkumu zaměřeného na klinický audit. Studie hodnotila nasazenou databázi jako

nástroj pro zlepšování kvality péče. Hlavním cílem bylo posoudit účinnost zdravotnické databáze při usnadňování procesu klinického auditu[Mar+20]. Závěrem studie bylo, že databáze je efektivním nástrojem pro klinický audit.

4.6 Zabezpečení a ochrana osobních údajů aplikace Czech Salivary Gland Database

Po komunikaci s odpovědnou osobou za GDPR (General Data Protection Regulation) ve Fakultní nemocnici v Motole bylo definováno několik bodů, které je nutné v rámci práce zpracovat.

- Způsob zajištění ochrany dat a práce s nimi v rámci aplikace.
- Přidávání nově diagnostikovaných pacientů.
- Retrospektivní přidávání pacientů.

4.6.1 Způsob zajištění ochrany dat a práce s nimi v rámci aplikace

Aplikace je společně s databází uložena lokálně na osobních počítačích lékařů, které jsou zabezpečeny přístupovým heslem. Pro přístup do aplikace je vyžadováno heslo a veškerá lokálně uložená data jsou šifrována pomocí symetrického protokolu AES.

4.6.2 Přidávání nově diagnostikovaných pacientů

Při přidávání nově diagnostikovaného pacienta do aplikace je nutné, aby před jeho vložením do databáze podepsal souhlas o zpracování osobních údajů. V souhlasu musí být jasně definováno, k jakému účelu budou jeho osobní informace použity a po jakou dobu budou uchovávány. Podoba souhlasu je v kompetenci Fakultní nemocnice v Motole.

4.6.3 Retrospektivní přidávání pacientů

U pacientů, kteří budou přidáváni retrospektivně, je nutné přesně definovat, proč jsou vybrané osobní údaje uloženy v databázi. Výběr jednotlivých osobních údajů a odůvodnění jejich uložení spadá do kompetence jednotlivých lékařů, kteří budou aplikaci používat.

Rozšíření aplikace o vizualizaci dat — 5

5.1 Kaplan-Meierova metoda

Kaplan-Meierova metoda je neparametrický odhad funkce přežití, který je definován následovně:

$$S(t) = \prod_{t_i \leq t} p_i = \prod_{t_i \leq t} \left(1 - \frac{d_i}{n_i}\right), \tag{5.1}$$

kde t_i jsou časy událostí (např. úmrtí pacienta), d_i je počet událostí (např. úmrtí) v čase t_i a n_i je počet osob (např. kteří nezemřeli) v čase t_i.

5.1.1 Matematické odvození

Odvození vychází z předpokladu, že pacient se dožije události v čase t, pokud událost (např. úmrtí) nenastala v žádném čase t^*, pro který platí $t^* < t$. Pro odhadnutí pravděpodobnosti, že u subjektu nenastala v čase t daná událost je potřeba odhadnout odpovídající pravděpodobnosti pro všechny předcházející časy t^*. [Šná]

Předpokládejme n různých časů přežití takových, že pro ně platí $t_1 < t_2 < ... < t_n < t$. Poté lze pravděpodobnost výskytu vyjádřit vztahem

$$\begin{aligned} S(t) &= P(T > t) \\ &= P\left((T > t_1) \cap (T > t_2) \cap \ldots (T > t)\right) \\ &= P\left(T > t_1\right) P\left(T > t_2 \| T > t_1\right) \\ &\quad \ldots P\left(T > t_n \| T > t_{n-1}\right). \end{aligned} \tag{5.2}$$

Odhad $S(t)$ přežití v daném čase je poměr celkového počtu pacientů na konci časového úseku t_i ku celkovému počtu pacientů na začátku úsek t_i.

$$p_i = \frac{R_i - d_i}{R_i} = 1 - \frac{d_i}{n_i} \tag{5.3}$$

Následným vynásobením odhadů přežití přes všechny pozorované časové úseky získáme vztah 5.1.

5.1.2 Cenzurování dat

V průběhu sledování pacientů mohou nastat události, které neočekávaně dokážou změnit jejich stav, ale nejsou cílem pozorování dané studie (např. pacient s nádorovým onemocněním zemře při autonehodě). Takové skutečnosti je nutné v rámci datového souboru cenzurovat.

5.1.3 Předpoklady pro cenzuru

Prvním předpokladem je, že pacienti, kteří jsou cenzurováni kdykoli během sledování, mají stejné šance na přežití jako ti, kteří jsou nadále sledováni. Druhý předpoklad je ten, že pravděpodobnosti přežití pacientů, kteří byli rekrutování na začátku i na konci studie jsou stejné.

Cenzura je zařazena do výpočtu tím, že ve vztahu 5.1 je nahrazena hodnota d_i nulou.

5.1.4 Ukázka výpočtu

V tab. 5.1 je vidět jak probíhá výpočet hodnot pomocí Kaplan-Meierovy metody. Je to část tabulky 1 z článku [Jag+08]. Výpočet využívá vztahu 5.1 a na řádku 5 je vidět, jak probíhá výpočet při cenzurování dat.

Tabulka 5.1: Tabulka ukazující výpočet hodnot pomocí Kaplan-Meierovy metody

i	Čas (dny)	Pacienti n_i	Úmrtí d_i	Cenzurování	Výpočet $p_{i-1} \times \left(1 - \frac{d_i}{n_i}\right)$	Pravděpodobnost přežití p_i	Pravděpodobnost úmrtí
1	0	50	0	0	$1 - \frac{0}{50}$	1.0000	0
2	34	50	1	0	$1.0000 \times \left(1 - \frac{1}{50}\right)$	0.9800	0.0200
3	35	49	1	0	$0.9800 \times \left(1 - \frac{1}{49}\right)$	0.9600	0.0400
4	44	48	1	0	$0.9600 \times \left(1 - \frac{1}{48}\right)$	0.9400	0.0600
5	57	47	0	1	$0.9400 \times \left(1 - \frac{0}{47}\right)$	0.9400	0.0600

5.1.5 Log-rank test

Při porovnávání přežití mezi různými skupinami není dostačující pouze porovnávat kumulativní pravděpodobnost přežití ve stejných časových okamžicích. Tímto přístupem je pouze zjištěn rozdíl šance přežití v daném časovém okamžiku. Pro lepší porovnání výsledků mezi dvěma skupinami se používá **log-rank test**. Ten je definovaný vztahem[GKK10],

$$P = \frac{(O_1 - E_1)^2}{E_1} + \frac{(O_2 - E_2)^2}{E_2}, \tag{5.4}$$

kde O_1 a O_2 je celkový počet pozorované události (např. smrti) v první a druhé skupině. A potom E_1, E_2 je celkový počet očekávaných událostí ve skupině.

5.1.6 Příklad výpočtu

V tab. 5.2 jsou vypočítána všechna potřebná data pro určení log-rank testu. Hodnoty E pro jednotlivé dny so počítají podle vztahu

$$\frac{\text{počet úmrtí v dané skupině}}{\text{celkový počet živých pacientů}} \cdot \text{počet živých pacientů ve skupině.} \tag{5.5}$$

Hodnoty E_1, E_2, O_1, O_2 jsou poté sumy daných řádků. Po dosazení do vztahu 5.4 získáme

$$P = \frac{(3 - 2.501)^2}{2.501} + \frac{(2 - 2.499)^2}{2.499} = 0.199. \tag{5.6}$$

Tabulka 5.2: Příklad výpočtu log-rank testu

Den	1	10	20	30	40	
Úmrtí v obou skupinách	1	1	1	1	1	
Úmrtí v 1. skupině O_1	1	0	0	1	1	3
Úmrtí v 2. skupině O_2	0	1	1	0	0	2
Živých pacientů v obou skupinách	46	45	44	43	42	
Živých pacientů 1. skupině	23	22	22	22	21	
Živých pacientů 2. skupině	23	23	22	21	21	
Pravděpodobnost očekávané události v 1. skupině E_1	$\frac{1}{46} \cdot 23 = 0.5$	$\frac{1}{45} \cdot 22 = 0.489$	$\frac{1}{44} \cdot 22 = 0.5$	$\frac{1}{43} \cdot 22 = 0.512$	$\frac{1}{42} \cdot 21 = 0.5$	2.501
Pravděpodobnost očekávané události v 2. skupině E_2	$\frac{1}{46} \cdot 23 = 0.5$	$\frac{1}{45} \cdot 23 = 0.511$	$\frac{1}{44} \cdot 22 = 0.5$	$\frac{1}{43} \cdot 21 = 0.488$	$\frac{1}{42} \cdot 21 = 0.5$	2.499

5.1.7 Vyhodnocení log-rank testu

Získaná hodnota z rovnice 5.4 se nazývá P-hodnota a udává zda jsou rozdíly mezi skupinami statisticky signifikantní. V praxi se využívá hladina významnosti α podle, které se následně provede rozhodnutí o signifikantnosti. Pokud je hodnota $P > \alpha$ neexistují dostatečné důkazy na to, aby bylo potvrzeno, že pozorovaný efekt je statisticky signifikantní tzn. není dostatek důkazů, aby bylo řečeno, že pozorovaný efekt je reálný. V případě, že hodnota $P \leq \alpha$, existuje dostatek důkazů na to, aby byl pozorovaný jev považován za statisticky signifikantní, tzn. existuje dostatek důkazů, aby se dalo říct, že pozorovaný efekt je reálný.

Při vyhodnocení P-hodnoty z uvedené příkladu, kde hodnota $\alpha = 0.05$ (v praxi často používaná hodnota), lze tedy říci, že výsledek není statisticky signifikantní, protože $0.199 > 0.05$.

5.2 Návrh implementace vizualizace

Při konzultaci s panem doktorem MUDr. Davidem Kalfeřtem, PhD., bylo specifikováno, že při zobrazení dat pomocí Kaplan-Meierovy metody bude možné vybrat, zda zobrazené křivky reprezentují pravděpodobnost přežití nebo pravděpodobnost recidivy v závislosti na čase. Mezi těmito možnostmi si bude možné vybírat pomocí tlačítek v uživatelském rozhraní. Dále bude možné vybrat skupiny pacientů, pro které budou křivky vykresleny podle jejich histopatologického typu nádoru.

V grafu se tedy budou zobrazovat křivky pro všechny zvolené skupiny a každá křivka bude reprezentována jinou barvou.

Pro samotnou implementaci bude zvolena knihovna **recharts**, která poskytuje React.js komponenty reprezentující jednotlivé grafy. Pro vizualizaci dat pomocí Kaplan-Meierovy metody bude využita komponenta `LineChart`, která umožňuje vykreslení grafu. Pro zobrazení pravděpodobností přežití bude využita komponenta `Line`, která umožňuje vykreslení dat v grafu.

Implementace kódu 6

6.1 Rozdělení aplikace do implementačních fází

Zdrojový kód aplikace byl rozdělen do několika fází, které postupně implementují jednotlivé části aplikace. Jednotlivé fáze, které jsou seřazeny podle pořadí, v jakém byly implementovány jsou uvedeny v tab. 6.1.

Tabulka 6.1: Implementační fáze a jejich popis

název fáze	**popis fáze**
Inicializace aplikace	Založení GitHub repositáře a instalace všech potřebných knihoven.
Inicializace CI/CD	Inicializace automatického vytvoření instalačních balíčků, testování a zveřejnění.
Kostra uživatelského rozhraní	Vytvoření kostry uživatelského rozhraní aplikace a základního menu.
Komponenta add-patient	Vytvoření komponenty pro přidávání pacientů.
Komponenta name-gland-form	Vytvoření komponenty vykreslující formulář pacienta.
Komponenta patients-list	Vytvoření komponenty zobrazující seznam všech pacientů.
Komponenta add-study	Vytvoření komponenty pro přidávání studií.
Komponenta studies-list	Vytvoření komponenty zobrazující seznam všech studií.
Napojení na DB	Napojení aplikace na databázi.
Filtrace	Implementace filtračního menu pacientů.
Export dat	Implementace exportování vybraných pacientů do `.xlsx` souborů.
Import dat	Implementace importování dat z `.xlsx` souborů.
Kaplan-Meier	Implementace Kaplan-Meierových křivek.
Zabezpečení	Implementace zaheslování aplikace a šifrování osobních údajů.

První dvě fáze se zabývají inicializací aplikace, nastavení adresářové struktury, založení GitHub adresáře, instalací všech potřebných knihoven a přípravě kontinuální integrace a nasazení (CI/CD). Následujících 5 fází se věnuje implementaci základních prvků uživatelského rozhraní. Zbývající fáze se věnují zejména funkcím pro práci s daty, jako je napojení na databázi, filtrace, export a import dat a implementace Kaplan-Meierových křivek. Poslední fáze se zaměřuje na zabezpečení aplikace.

Uvedené fáze byli následně přiřazeny do sedmi týdenních iterací, kde každá iterace měla v průměru padesát hodin. Na konci jednotlivých iterací byla pokaždé poskytnuta výstupní verze aplikace určena pro testování panem doktorem MUDr.

Davidem Kalfeřtem, Ph.D. Pokud byly zjištěny nějaké nedostatky, byly opraveny v následující iteraci. Všechny iterace jsou uvedeny v tab. 6.2.

Tabulka 6.2: Iterace a jejich popis

název iterace	**provedené fáze**
Iterace 1	Inicializace aplikace, CI/CD, komponenta add-patient, kostra UR.
Iterace 2	Komponenta name-gland-form.
Iterace 3	Komponenta patients-list.
Iterace 4	Komponenty add-study, studies-list.
Iterace 5	Napojení na DB, filtrace, export a import dat.
Iterace 6	Zabezpečení, Kaplan-Meier.
Iterace 7	Oprava chyb ve finální verzi aplikace.

6.2 První iterace

Hlavním cílem první iterace byla inicializace aplikace, založení GitHub repositáře, instalace všech potřebných knihoven a vytvoření základní komponenty pro přidávání pacientů.

6.2.1 Inicializace aplikace

Vzhledem k tomu, že v kapitole 3.3 bylo zjištěno, že aplikace obsahuje velké množství nedostatků ve zdrojovém kódu, bylo by velice obtížné všechny problémy odstranit a s vysokou pravděpodobností by to bylo časově náročnější, než vytvořit novou aplikaci od základu. Proto bylo rozhodnuto vytvořit úplně nový projekt, který bude obsahovat všechny funkcionality, jenž byli v původní aplikaci a bude obsahovat všechny nově definované požadavky, které byli navrhnuty v kapitole 3.

Instalace základních knihoven

Pro instalaci základních knihoven byl využit nástroj **Electron Forge**[For24], který umožňuje rychlou inicializaci projektu, s využitím frameworku Electron.js ve spojení s Webpackem a TypeScriptem. Webpack je nástroj jehož hlavním účelem je sloučení všech zdrojových souborů front-endové části aplikace do jednoho nebo více balíčků, které mohou být snadněji načteny do okna aplikace[Web]. TypeScript je nadstavba jazyka JavaScript, která přidává statické typování a další funkce, které zvyšují čitelnost a udržovatelnost kódu. Instalace byla provedena pomocí příkazu:

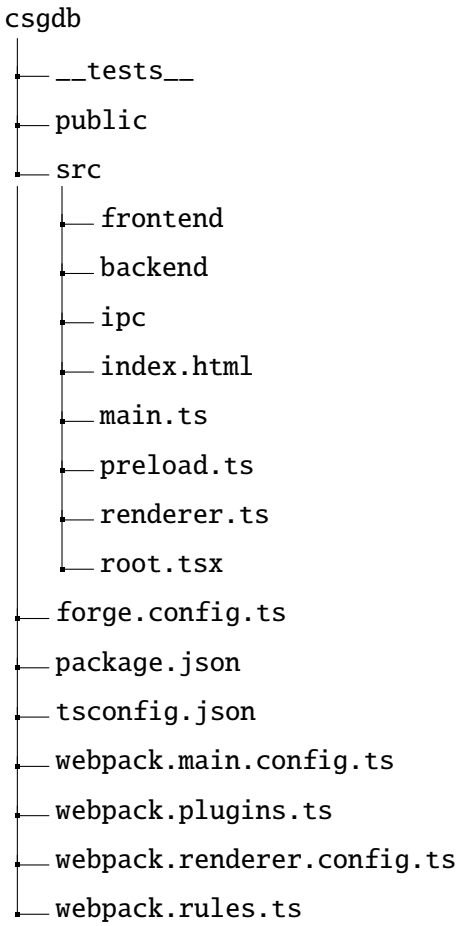
```
> npm init electron-app@latest my-new-app -- --template=webpack-
    typescript
```

Pro provedení prvotní instalace balíčků byl nainstalován framework React.js pomocí příkazů:

```
> npm install --save react react-dom
> npm install --save-dev @types/react @types/react-dom
```

Vytvoření základní adresářové struktury

Značná část adresářové struktury byla automaticky vygenerována pomocí nástroje Electron Forge. Následně byla doplněna o adresáře, které realizují architekturu navrženou v kapitole 3.3.3. Výsledná adresářová struktura je následující:



Popis jednotlivých adresářů je uveden v tab. 6.3.

Tabulka 6.3: Popis adresářové struktury

adresář	obsah
__tests__	Jednotkové a E2E testy.
public	Obrázky využité v aplikaci.
src	Zdrojový kód aplikace.
frontend	React.js komponenty tvořící uživatelské rozhraní.
backend	Funkce pro komunikaci s databází.

Vytvoření GitHub repositáře

Veškeré změny v kódu byly verzovány pomocí nástroje Git a byly pravidelně ukládány do GitHub repositáře. Samotné vytvoření repositáře bylo provedeno pomocí webového rozhraní GitHub, které obsahuje návod kompletní návod tvorby nového repositáře [Git23].

6.2.2 Inicializace CI/CD

Vzhledem k tomu, že aplikace byla po každé iteraci poskytována k testování panem doktorem MUDr. Davidem Kalfeřtem, PhD., bylo nutné vytvořit automatický proces, který bude zajišťovat vytvoření instalačních balíčků, testování a zveřejnění. Pro tento účel byl využit nástroj GitHub Actions, který umožňuje vytvoření vlastního CI/CD procesu. Označení CI/CD vychází z anglického názvu Continuous Integration/Continuous Deployment, což znamená neustálé integrování a nasazování. Ve své podstatě se jedná o procesy, které zajišťují automatické testování a nasazování aplikace.

Všechny CI/CD akce byly definovány v adresáři `.github/workflows`, kde byly následně vytvořeny tři soubory realizující jednotlivé části CI/CD procesu.

První soubor `build.yml`, který je spuštěn při každém provedení pull-request (PR), což je požadavek na začlenění změn do hlavní větve repositáře, má za úkol otestovat, zda je možné aplikaci sestavit pro předem definované operační systémy, které jsou v případě této aplikace Windows, MacOS a Linux (Ubuntu).

Dalším souborem, který je spuštěn při provedení každého PR, je `test.yml`. Jeho hlavním úkolem je otestovat zda aplikace prochází všemi jednotkovými testy, definovanými v adresáři `__tests__`. Další kontroly prováděné touto akcí, vykonávají statickou analýzu kódu pomocí, nástroje ESLint a kontrolu formátování pomocí nástroje Prettier.

Poslední soubor `release.yml` je spuštěn při vytváření instalačního souboru pro danou verzi aplikace. Akce je automaticky spuštěna při vytvoření nového vydání

(release) v GitHub repositáři a má za úkol vytvořit instalační balíček pro operační systém Windows. Zvolený operační systém byl vybrán na základě konzultace s panem doktorem, a to z toho důvodu, že většina jeho spolupracovníků s tímto systémem pracuje a je pro ně nejvhodnější. Dalším důvodem volby pouze tohoto operačního systému byla nutnost placení vývojářského účtu pro MacOS a vzhledem k malému počtu uživatelů využívajících MacOS bylo rozhodnuto, že financování tohoto účtu by bylo neekonomické. Pro Linux problém s financováním nenastává, ale vzhledem k tomu, že téměř žádný z uživatelů operační systém nevyužívá bylo rozhodnuto, že bude také vynechán.

6.2.3 Kostra uživatelského rozhraní a základní menu

Hlavním cílem této fáze bylo vytvoření kostry uživatelského rozhraní (UR) a základního menu, které obsahuje tlačítka pro přidání pacienta, přidání studie, zobrazení seznamu pacientů, zobrazení seznamu studií, vizualizace Kaplan-Meirových křivek a import dat.

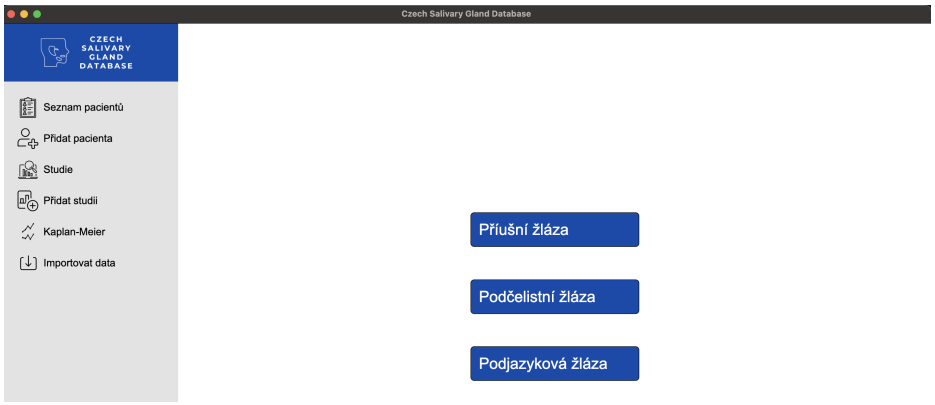
Implementace této části aplikace byla provedena v adresáři `frontend`, ve kterém byl vytvořen podadresář `components`. V tomto adresáři byly vytvořeny komponenty `App` a `Menu`. Komponenta `App` definuje základní kostru aplikace a mechanismus, kterým se překresluje obsah hlavní části okna podle zvolené položky v menu. Komponenta `Menu` obsahuje seznam tlačítek, které umožňují přepínat mezi jednotlivými částmi aplikace.

6.2.4 Komponenta add-patient

V této fázi byla vytvořena komponenta `AddPatient`, která vykresluje tři tlačítka: **Příušní žláza**, **Podčelistní žláza** a **Podjazyková žláza**. Úkolem tlačítek je zobrazit formulář pro přidání pacienta s daným typem nádoru.

6.2.5 Výsledek první iterace

Výsledkem první iterace bylo vytvoření základní adresářové struktury aplikace, inicializace GitHub repositáře, definice CI/CD akcí, instalace potřebných knihoven, vytvoření kostry uživatelského rozhraní a základního menu viz obr. 6.1.



Obrázek 6.1: Kostra aplikace s menu a vykreslenou komponentou `AddPatient`

6.3 Druhá iterace

Druhá iterace se zaměřila na vytvoření komponent, které umožňují vykreslit formulář specifický pro daný typ žlázy. Ačkoli je název prováděné fáze **Komponenta name-gland-form**, ve skutečnosti se jedná o tři komponenty, kde každá reprezentuje jeden formulář. Vzhledem k tomu, že jsou formuláře velice rozsáhlé, bylo nutné vytvořit několik dalších subkomponent, které realizují jednotlivé části. Aby byla udržena jednoduchá navigace napříč kódem, byl vytvořen adresář `components/forms`, ve kterém se nacházejí všechny subkomponenty formuláře.

6.3.1 Komponenta name-gland-form

Úkolem této fáze bylo vytvoření tří komponent `ParotidGlandForm`, `SubmandibularGlandForm` a `SublingualGlandForm`, které reprezentují formuláře pacienta s nádorem v příušní, podčelistní a podjazykové žláze. Každý formulář obsahuje několik polí, které uživateli umožňují zadat informace o pacientovi, jako je jméno, příjmení, rodné číslo a další.

Elementární subkomponenty formuláře

Elementární subkomponenty reprezentují jednotlivé prvky, zaznamenávající vstup od uživatele a jsou uvedeny v tab. 6.4 na str. 43.

Tabulka 6.4: Elementární komponenty formuláře

název komponenty	**popis**
`NumberInput`	Vstupní pole pro číselné hodnoty.
`TextInput`	Vstupní pole pro text.
`SimpleCheckboxes`	Skupina zaškrtávacích boxů.
`ConditionalCheckboxes`	Skupina podmíněných zaškrtávacích boxů.

Komponenty `NumberInput` a `TextInput` obalují základní značky HTML `<input type='text'>`, `<input type='number'>` a přidávají k nim funkcionality, umožňující kontrolu vstupu, podmíněné umožnění zápisu a schopnost vykreslení předem definovaných dat z databáze, což je využito při následném vygenerování již existujících pacientů.

Komponenta `SimpleCheckboxes` umožňuje vykreslit skupinu boxů, které uživateli dovolují vybrat jednu nebo více možností. To, zda uživatel může vybrat více možností, závisí na nastavení komponenty a konkrétně to zařizuje tzv. „hook“ s názvem `useSingleSelection`.

Poslední elementární komponenta `ConditionalCheckboxes` vykresluje skupinu zaškrtávacích boxů, ve které mohou existovat zanořené vstupní prvky, jež se zobrazují pouze v případě, že byl zaškrtnut nějaký konkrétní box. Opět komponenta využívá „hook“ `useSingleSelection`.

Společné části formuláře

Každý z formulářů obsahuje několik společných částí, které bylo vhodné kvůli zamezení repetetivnosti kódu extrahovat do samostatných komponent, které skládají elementární komponenty do jednoho celku. Komponenty reprezentující jednotlivé části formuláře, jsou uvedeny v tab. 6.5. Každá z těchto částí opět umožňuje zobrazení načtených hodnot z databáze.

Tabulka 6.5: Společné části formuláře

název komponenty	**popis**
`PersonalData`	Osobní údaje o pacientovi.
`Histopathology`	Histopatologický typ nádoru.
`TNMClassification`	TNM klasifikace pacienta.
`Dispensarization`	Dispenzarizace pacienta.
`Attachments`	Přílohy k pacientovi.
`Notes`	Poznámky k pacientovi.
`AddPatientButton`	Tlačítko pro přidání pacienta.
`EditButtons`	Tlačítka pro editaci a smazání pacienta.

Specifické části formuláře

Jednotlivé formuláře se liší v částech, ve kterých je definována diagnóza a terapie pacienta. Pro každou část byla vytvořena samostatná komponenta, specifická pro daný typ žlázy, která byla následně opět složena z elementárních komponent. Pro zjednodušení navigace v kódu byly vytvořeny adresáře `components/forms/parotid`, `components/forms/submandibular` a `components/forms/sublingual`, ve kterých se nacházejí jednotlivé specifické komponenty.

Kompletní formulář

Posledním krokem implementace formuláře bylo složení jednotlivých komponent, které reprezentují části formuláře do jednoho celku. Po provedení této akce byl formulář kompletní a bylo možné jej zobrazit v uživatelském rozhraní.

6.3.2 Výsledek druhé iterace

Po dokončení druhé iterace byly vytvořeny tři kompletní formuláře pro jednotlivé typy žláz. Každý formulář realizuje jedna komponenta, která obsahuje další subkomponenty, reprezentující jeho části. Vzniklé komponenty jsou univerzální a umožňují jak zapisování dat, tak i jejich načtení z databáze.

6.4 Třetí iterace

Třetí iterace se věnovala implementaci komponenty `PatientsList`, která umožňuje zobrazit seznam pacientů. Komponenta byla vytvořena v již zmíněném adresáři `components`. Vzhledem k tomu, že seznam pacientů se vyskytuje na třech místech v aplikaci a pokaždé v něm mohou být zobrazeni jiní pacienti, bylo nutné, aby vzniklá komponenta dokázala zobrazit seznam pacientů podle zvolených parametrů.

6.4.1 Komponenta PatientsList

Při implementaci komponenty `PatientsList` bylo nutné přihlížet na to, že seznam pacientů bude zobrazován v několika částech aplikace, ale pokaždé s jinými daty a zároveň bude umožňovat zobrazení pacientů podle zvolených filtrů. Avšak konkrétní implementace těchto vlastností byla provedena až v pozdějších iterací a v této iteraci se pro tuto komponentu implementovaly pouze základní funkcionality, jako je samotný seznam, jednoduché vyhledávání a možnost označení pacientů.

Základní funkcionality

Hlavní základní funkcionalitou komponenty `PatientsList` je zobrazení seznamu s tlačítky, kde každé tlačítko reprezentuje jednoho pacienta a po jeho stisknutí se zobrazí formulář, který obsahuje všechny informace o daném pacientovi. K dosažení této vlastnosti byla využita HTML značka <table>, která vykresluje tabulku. Do tabulky byla následně vložena tlačítka, které reprezentuje komponenta s názvem `PatientButton`. Jejím primárním úkolem je zobrazit tlačítko, které obsahuje jméno, příjmení a typ nádoru pacienta. Sekundárním úkolem je reakce na stisknutí samotného tlačítka, kdy se zobrazí formulář s informacemi o daném pacientovi.

Druhou základní funkcionalitou je možnost vyhledávání pacientů podle jména, příjmení nebo rodného čísla. Toto vyhledávání je realizováno pomocí textového vstupního pole, které umožňuje zadat hledaný výraz. Vyhledávání je prováděno v reálném čase, což znamená, že se seznam pacientů překresluje v závislosti na zadaném výrazu. Samotné prohledávání seznamu pacientů je prováděno funkcí `handlePatientSearch`, která filtruje pacienty pomocí JavaScript funkce `filter` viz zdrojový kód 6.1.

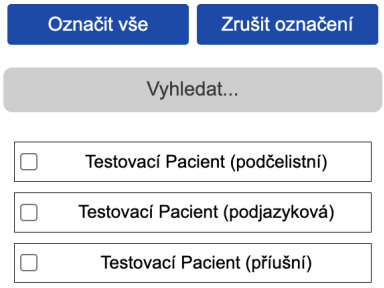
Zdrojový kód 6.1: Funkce pro vyhledávání pacientů

```
1  const foundPatients = allPatients.filter((patient) => {
2    // jméno a příjmení pacienta
3    const fullName =
4        `${patient.jmeno || ''} ${patient.prijmeni || ''}`.
             toLowerCase()
5    // rodné číslo pacienta
6    const rodneCislo = patient.rodne_cislo || ''
7    // vrať true pokud je pacient v poli jinak false
8    return (
9        fullName.includes(search.toLowerCase()) ||
10       rodneCislo.toLowerCase().includes(search.toLowerCase())
11   )
12 })
```

Poslední základní funkcionalitou je možnost označení pacientů pomocí zaškrtávacích boxů. Uživatel má možnost označit jednoho nebo více pacientů, a to buď klikáním na příslušné zaškrtávací boxy nebo pomocí tlačítka **Označit vše**. Dále je uživateli umožněno odoznačit všechny pacienty pomocí tlačítka **Zrušit označení**.

6.4.2 Výsledek třetí iterace

Ve třetí iteraci byla vytvořena komponenta `PatientsList`, která umožňuje zobrazit seznam pacientů a při kliknutí na některého z nich, je vykreslen formulář. Dále je možné v seznamu vyhledávat pacienty podle jména, příjmení nebo rodného čísla. Vykreslená komponenta je vidět na obr. 6.2.



Obrázek 6.2: Komponenta PatientsList

6.5 Čtvrtá iterace

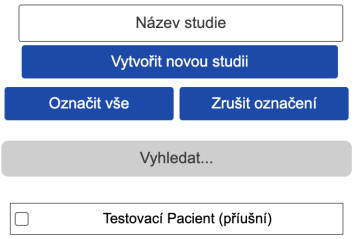
Cílem čtvrté iterace bylo vytvoření komponent `AddStudy` a `StudiesList`, které umožňují přidávat studie a zobrazovat seznam studií. V aplikaci existují čtyři typy studií, jeden pro každý druh žlázy a speciální studie pro všechny druhy.

6.5.1 Komponenta AddStudy

Komponenta `AddStudy` umožňuje přidat novou studii do databáze. Nejprve vykreslí tlačítka, která reprezentují jednotlivé typy studií. Stisknutí jednoho z tlačítek zobrazí komponentu `StudyCreation` obsahující seznam pacientů a vstupní pole. Seznam obsahuje pouze pacienty, které je možné do studie zařadit. Vstupní pole slouží k zadání názvu studie.

Pomocná komponenta StudyCreation

Komponenta `StudyCreation` využívá již existující komponenty `PatientsList`. Umožňuje vybrat pacienty, kteří budou zařazeni do dané studie. Komponenta `PatientsList` musela být upravena přidáním vstupního pole určeného k zadání názvu studie. To znamená, že pokud byl zobrazen seznam pacientů v rámci komponenty `StudyCreation` bylo možné vybrat pacienty a zároveň zadat název studie viz obr. 6.3 na str. 47.



Obrázek 6.3: Komponenta StudyCreation

Zmíněná úprava byla implementována přidáním parametru `studyType` do komponenty `PatientsList`. Parametr určuje, zda se jedná o zobrazení pacientů v rámci vytváření studie nebo v rámci seznamu pacientů.

6.5.2 Komponenta StudiesList

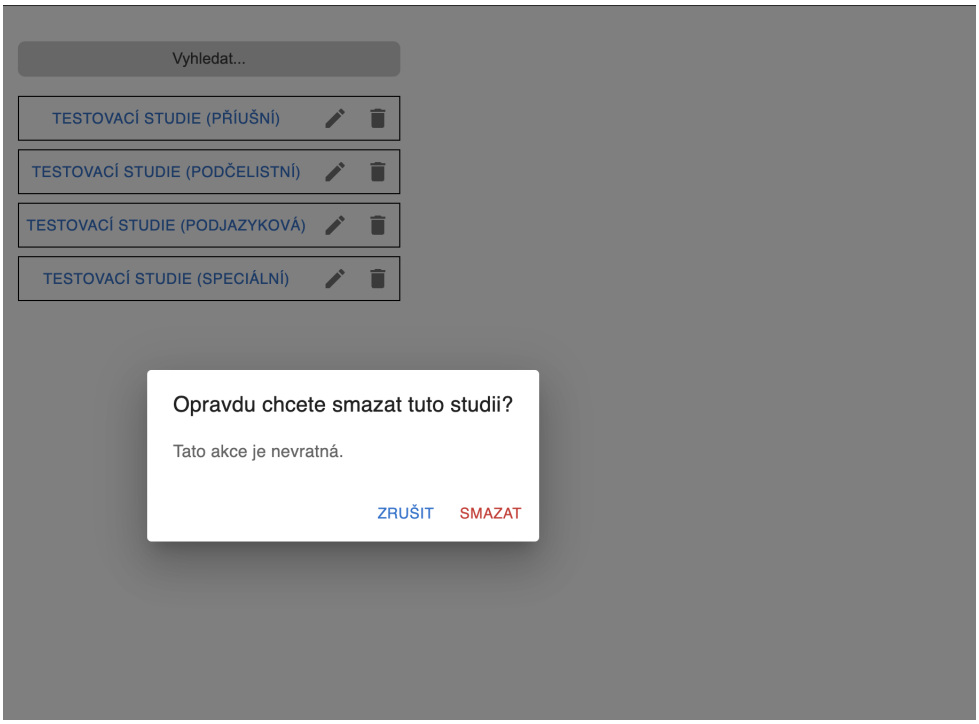
Komponenta `StudiesList` zobrazuje seznam všech studií (viz obr. 6.4) v databázi, ve kterém je možné vyhledávat pomocí textového pole dle názvu studie. Každá studie je reprezentována tlačítkem, které obsahuje název a typ studie. Po stisknutí tlačítka se zobrazí seznam pacientů, kteří do dané studie patří.

Obrázek 6.4: Komponenta StudiesList

Mazání a editace studií

Každá studie v seznamu má vedle názvu dvě tlačítka viz obr. 6.4. První tlačítko s ikonou tužky slouží k editaci názvu, druhé s ikonou koše k odstranění studie z databáze. Po stisknutí tlačítka s ikonou koše se zobrazí dialogové okno viz obr. 6.5, které se ptá uživatele, zda si je jistý, že chce danou studii smazat. Po potvrzení se studie smaže z databáze.

Stisknutím tlačítka s ikonou tužky se promění název studie na vstupní pole, které umožňuje editaci názvu. Po stisknutí tlačítka s ikonou „fajfky“ nebo stlačení klávesy Enter se změny uloží do databáze. Pokud však uživatel nechce provést žádné změny může stisknout tlačítko s ikonou křížku a změny nebudou uloženy.



Obrázek 6.5: Dialogové okno pro smazání studie

Odebrání pacienta ze studie

Pokud uživatel stiskne tlačítko reprezentující studii, zobrazí se seznam pacientů, kteří do dané studie patří. Každý pacient je reprezentován tlačítkem, které obsahuje jméno, příjmení a typ nádoru. Po stisknutí tlačítka se zobrazí formulář obsahující data daného pacienta. V zobrazeném formuláři se v jeho pravé horní části nachází tlačítka **Editovat**, **Smazat pacienta** a **Odebrat ze studie**. Stisknutím tlačítka **Odebrat ze studie** je pacient odebrán ze studie a není nutné dalšího potvrzování. Důvodem je možnost opětovného vrácení pacienta do studie.

6.5.3 Výsledek čtvrté iterace

Po dokončení čtvrté iterace vývoje byly implementovány komponenty `AddStudy` a `StudiesList`. Komponenty umožňují přidávat nové studie a procházet seznam existujících studií. Dále komponenty umožňují zobrazit seznam pacientů zařazených do dané studii a provádět základní operace, jako je editace, mazání studií a odebírání pacientů ze studie. Výsledné komponenty jsou vidět na obr. 6.3 na str. 47 a 6.4 na str. 47.

6.6 Pátá iterace

Úkolem páté iterace bylo propojit prvky uživatelského rozhraní s databází a umožnit uživatelům filtrovat a exportovat pacienty. Model databáze byl převzat z původní verze aplikace. Veškeré funkce pro komunikaci s databází byly implementovány v adresáři `backend`.

6.6.1 Instalace databáze

Nová verze aplikace využívá stejnou databázi jako původní, kterou je SQLite. Instalace databáze byla provedena pomocí příkazu:

```
> npm install sqlite3
```

Vzhledem k tomu, že kód aplikace využívá TypeScript bylo nutné nainstalovat i typy pro SQLite pomocí příkazu:

```
> npm install --save-dev @types/sqlite3
```

Těmito příkazy byla nainstalována databáze i s jejími typy. Což umožňuje využívání TypeScriptu v kombinaci s SQLite.

6.6.2 Připojení k databázi

Připojení k databázi zajišťuje soubor `dbManager.ts` konkrétně pomocí příkazu:

```
const db = new sqlite3.Database(getDBPath('db'))
```

Příkaz využívá funkci `getDBPath`, která vrací cestu k databázi. Důvod implementace funkce spočívá v tom, že nástroj ElectronForge při vytváření distribučních souborů balí veškerý obsah zdrojového kódu do souboru `app.asar` a všechny obsažené soubory jsou pouze pro čtení. Pokud by byla databáze umístěna v tomto souboru nebylo by možné do ní zapisovat. Proto byla vytvořena zmíněná funkce, která vrací cestu k databázi mimo soubor `app.asar`.

6.6.3 Vytvoření tabulek databáze

Tvorbu jedné tabulky databáze zajišťuje funkce `createTable`, která přijímá jako své argumenty název tabulky a TypeScript rozhraní (interface) reprezentující sloupečky tabulky. Zmíněné rozhraní je odlišné od rozhraní běžných programovacích jazyků jako je například Java zejména tím, že neslouží k deklaraci chování třídy ale pouze k určení struktury proměnné. Funkce po přijetí argumentů vytvoří SQL příkaz, který vytvoří tabulku s danými sloupečky.

Všechny tabulky byly vytvořeny pomocí funkce `db.serialize`, kterou poskytuje SQLite viz zdrojový kód 6.2 na str. 50.

Zdrojový kód 6.2: Funkce pro vytvoření tabulek

```
db.serialize(() => {
    createTable(TableNames.podcelistni, podcelistniColumns)
    createTable(TableNames.podjazykove, podjazykoveColumns)
    createTable(TableNames.priusni, priusniColumns)
    createTable(TableNames.studie, studieColumns)
    createTable(TableNames.jeVeStudii, jeVeStudiiColumns)
    createTable(TableNames.password, passwordColumns)
})
```

6.6.4 Definice základních operací s databází

Veškeré základní operace s databází jako je vkládání, mazání, editace a získávání dat, byly implementovány v souboru `basicOperations.ts`. Základní operace jsou implementovány takovým způsobem, aby dokázaly pracovat s libovolnou tabulkou a to díky tomu, že přijímají jako své argumenty název tabulky a data, která mají být vložena, smazána nebo editována. V případě získávání dat je nutné předat konkrétní sloupec nebo sloupce podle kterých se má vyhledávání provést. Funkce implementující zmíněné operace jsou uvedeny v tab. 6.6 na str 50.

Tabulka 6.6: Základní operace s databází

název funkce	**popis**
`insertRow`	Vložení dat do tabulky.
`getRow`	Získání jednoho záznamu z tabulky.
`getRows`	Získání všech záznamů z tabulky.
`getRowsBy`	Získání záznamů z tabulky podle zadaných sloupců.
`updateRow`	Editace záznamu v tabulce.
`saveRow`	Uložení nebo editace záznamu podle jeho existence.
`deleteRow`	Smazání záznamu z tabulky.
`deleteRowsBy`	Smazání záznamů z tabulky podle zadaných sloupců.

6.6.5 Komunikace s jednotlivými tabulkami

Každá tabulka má vlastní soubor s názvem `<nazev_tabulky>Manager.ts`, který slouží k implementaci funkcí pro komunikaci s danou tabulkou. Jedinou výjimkou jsou tabulky `form_priusni`, `form_podcelistni` a `form_podjazykove`, které sdílí pouze jeden soubor `patientsManager.ts`. Všechny tři tabulky sdílí jeden soubor, protože obsahují pacienty uložené v databázi. Pokud by byly tyto tabulky implementovány v samostatných souborech, bylo by nutné duplikovat kód, který se týká práce s pacienty.

Kvůli komunikaci s tabulkami byli vytvořeny soubory, které jsou uvedené v tab. 6.7. Každý z těchto souborů obsahuje funkce pro práci s konkrétní tabulkou.

Tabulka 6.7: Soubory pro komunikaci s tabulkami

název souboru	**popis**
`patientsManager`	Komunikace s tabulkami form_priusni, form_podcelistni a form_podjazykove.
`studiesManager`	Komunikace s tabulkou studie.
`jeVeStudiiManager`	Komunikace s tabulkou jeVeStudii.
`passwordManager`	Komunikace s tabulkou password.

6.6.6 Propojení uživatelského rozhraní s databází

Propojení uživatelského rozhraní s databází bylo provedeno pomocí mechanismu Inter Process Communication (IPC), který je součástí frameworku Electron.js. Mechanismus umožňuje komunikaci mezi hlavním a vykreslovacím (render) procesem aplikace. Úkolem hlavního procesu je komunikace s operačním systém. Vzhledem k tomu, že databáze je soubor uložený na disku, provádí i samotnou komunikaci s databází. Vykreslovací proces zajišťuje vykreslení uživatelského rozhraní a zobrazení dat načtených z databáze.

Implementace IPC komunikace

Ačkoliv je komunikace mezi hlavním a vykreslovacím procesem součástí souboru `main.ts` byl vytvořen modul s názvem `ipc`, který je následně importován do hlavního souboru aplikace. Hlavním důvodem je udržení přehlednosti kódu.

Modul `ipc` obsahuje soubory, které jsou uvedeny v tabulce 6.8. Soubory obsahují

Tabulka 6.8: Soubory modulu ipc

název souboru	**popis**
`ipcAPIHandles.ts`	Funkce pro komunikaci s databází z vykreslovacího procesu.
`ipcChannels.ts`	Kanály pro komunikaci mezi hlavním a vykreslovacím procesem.
`ipcExportHandles.ts`	Funkce pro export dat z databáze.
`ipcImportHandles.ts`	Funkce pro import dat do databáze.
`ipcEncryptionHandles.ts`	Funkce pro zabezpečení aplikace a databáze.

funkce, které vyvolává vykreslovací proces po některém z definovaných kanálů. Po provedení příslušné reakce je vrácena patřičná návratová hodnota pro vykreslovací proces.

Posledním krokem implementace IPC komunikace, bylo zpřístupnění funkcí `ipc` modulu, vykreslovacímu procesu pomocí souboru `preload.ts`. Zpřístupnění umožňuje modul `contextBridge` z frameworku Electron.js. Konkrétně pomocí funkce `exposeInMainWorld`. Ukázka použití této funkce je uvedena v zdrojovém kódu 6.3 na str. 52.

Zdrojový kód 6.3: Ukázka použití funkce exposeInMainWorld

```
// zpřístupnění funkce save vykreslovacímu procesu po kanále api
contextBridge.exposeInMainWorld('api', {
  save: (channel: ipcAPISaveChannels, data: JSON) => {
      return ipcRenderer.invoke(channel, [data])
  },
  ...
})
```

Takto poskytnuté funkce je následně možné volat z vykreslovacího procesu pomocí globální proměnné `window.api`. Analogicky jsou zpřístupněny všechny funkce z modulu `ipc`.

6.6.7 Využití IPC komunikace ve vykreslovacím procesu

Po dokončení implementace IPC komunikace bylo umožněno vykreslovacímu procesu zobrazovat seznam pacientů, studií, vytvářet nové pacienty, studie, filtrovat, exportovat a importovat pacienty.

Vytváření nových pacientů a studií

Pro vytváření nových pacientů bylo přidáno tlačítko **Přidat pacienta** do komponenty reprezentující formulář. Tlačítku byla přidána funkce, která po stisknutí zavolá pomocí IPC komunikace funkci pro uložení nového pacienta do databáze viz zdrojový kód 6.4. Analogicky byla vytvořena funkce pro ukládání studií.

Zdrojový kód 6.4: Funkce pro přidání pacienta

```
const handleButtonClick = async (
    e: React.MouseEvent<HTMLButtonElement>
) => {
    e.preventDefault()
    // převedení dat do JSON formátu
    const JSONdata = JSON.parse(JSON.stringify(formData))
    // uložení dat do databáze
    const result = await window.api.save(
        ipcAPISaveChannels.savePatient,
        JSONdata
    )
    ...
}
```

Zobrazení seznamu pacientů a studií

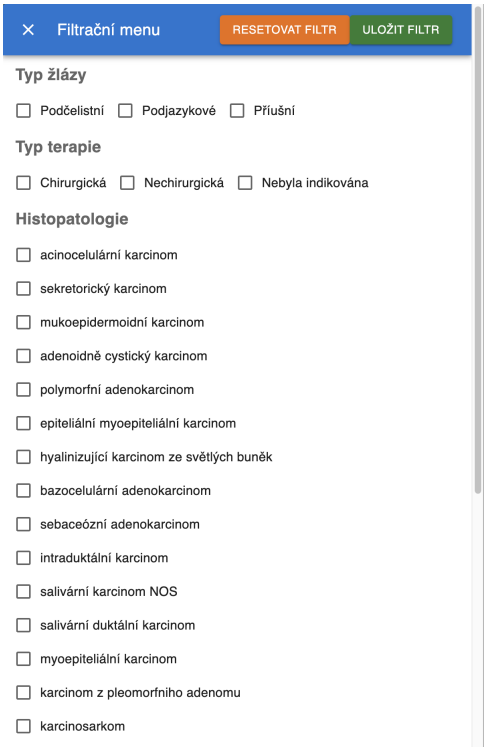
Komponenty `StudiesList` a `PatientsList` byli rozšířeny o novou funkci, která je volána při jejich prvním vykreslení a následně po provedení jakékoliv změny zobrazených dat. Funkce využívá IPC komunikaci k tomu, aby načetla aktuální data z databáze.

Edtiace a mazání studií a pacientů

Tlačítkům, která vyvolávají akci editace a mazání, byly přidány funkce, které po jejich stisknutí zavolají pomocí IPC komunikace funkci pro editaci nebo mazání dat z databáze.

6.6.8 Filtrování pacientů

Pro filtrování pacientů byla vytvořena komponenta `FiltrationMenu`, která zobrazí seznam atributů dle, kterých je pacienty možné filtrovat viz obr. 6.6.



Obrázek 6.6: Komponenta FiltrationMenu

Po stisknutí tlačítka **Uložit filtr** jsou dle uložených filtrů zobrazení pacienti. Filtrování pacientů provádí funkce `getFilteredPatients`, která je vidět v ukázce zdrojového kódu 6.5 na str. 54.

Zdrojový kód 6.5: Funkce pro filtrování pacientů

```
1 export const getFilteredPatients = async (filter, idStudie ) => {
2     if (idStudie) {
3         // pokud je definováno idStudie jsou filtrováni pouze
              pacienti, kteří jsou součástí dané studie
4         return getFilteredPatientsFromStudy(filter, idStudie)
5     } else {
6         // filtrace všech pacientů
7         return await getFilteredPatientsFromAllPatients(filter)
8     }
9 }
```

Funkce využívá dalších dvou pomocných funkcí, které zajišťují získání pacientů dle zvolených filtrů. První funkce na řádku čtyři ve zdrojovém kódu 6.5 získává pacienty dle zvolených filtrů z aktuálně zobrazené studie. Druhá funkce na řádku sedm získává pacienty dle zvolených filtrů ze všech pacientů.

6.6.9 Export a import pacientů

Export pacientů zařizují tlačítka **Exportovat** a **Exportovat anonymizovaně**, které byly přidány do komponenty `PatientsList`. Volbu pacientů pro export může uživatel provést, buď pomocí zaškrtávacích boxů, nebo stisknutím tlačítka **Označit vše**. Po vybrání pacientů určených pro export, je vyvolána exportní operace pomocí zmíněných tlačítek. Uživatel má možnost si vybrat jaký druh exportu chce použít a následně cestu kam budou exportovaná data uložena.

Importování pacientů je realizováno pomocí tlačítka **Importovat data**, které se nachází v hlavním menu aplikace. Po jeho stisknutí se zobrazí dialogové okno, ve kterém je možné vybrat soubor ve formátu `.xlsx`, ze kterého se budou data importovat.

Pro práci se soubory ve formátu `.xlsx` bylo nutné nainstalovat knihovnu **exceljs**, pomocí příkazu:

```
> npm install exceljs
```

Implementace exportu

Export je vyvolán tlačítky **Exportovat** a **Exportovat anonymizovaně**. Po stisknutí těchto tlačítek je vyvolána funkce `exportPatients`, která přijímá jako své argumenty seznam vybraných pacientů a zda mají být data anonymizována. Anonymizace dat provádí přepsání jména, příjmení a rodného čísla na řetězce s textem „Anonymizováno“. Po zavolání funkce je vytvořeno dialogové okno pomocí funkce `dialog.showSaveDialog`, která je součástí frameworku Electron.js. Využití funkce je vidět v ukázce kódu 6.6 na str. 55.

Zdrojový kód 6.6: Vytvoření dialogového okna pro uložení exportovaných dat

```
const result = await dialog.showSaveDialog({
  // titulek dialogu
  title: 'Export Patients',
  // výchozí cesta k souboru
  defaultPath: 'patients.xlsx',
  // filtrované soubory
  filters: [{ name: 'Excel', extensions: ['xlsx'] }],
})
```

Dialogové okno využívá filtru, který dovoluje uživateli vybrat pouze soubory s koncovkou `.xlsx`. Po vybrání cesty kam budou data exportována, jsou vytvořeny jeden až tři soubory, kde každý soubor obsahuje pouze data pacientů s jedním typem nádoru.

Implementace importu

Po vyvolání importu pomocí tlačítka **Importovat data** je zavolána funkce `import`, která zobrazí filtrované dialogové okno. V tomto okně je možné vybrat soubor, ze kterého budou data importována do databáze.

6.6.10 Výsledek páté iterace

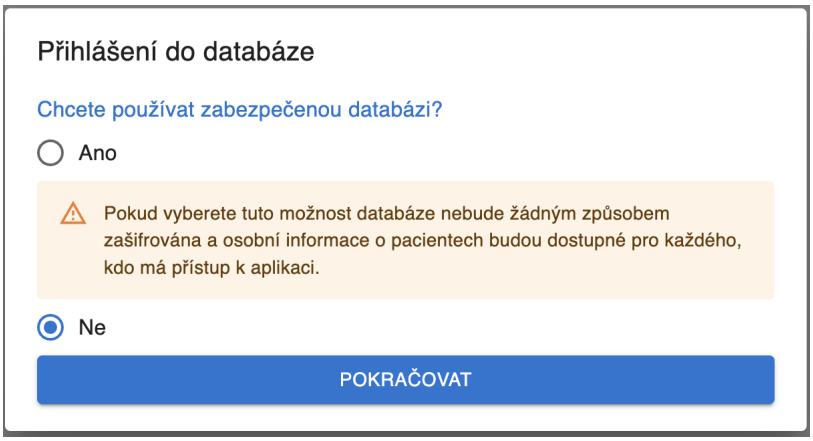
Implementací všech fází páté iterace byla aplikace napojena na lokální databázi. Díky tomu bylo umožněno uživatelům ukládat, mazat, editovat a zobrazovat data pacientů a studií. Poté byla implementována filtrace pacientů podle typu žlázy, terapie a histopatologického typu nádoru. Nakonec byl realizován export a import pacientů ve formátu `.xlsx`.

6.7 Šestá iterace

Šestá iterace se zaměřila na zabezpečení aplikace. Zejména pomocí přihlašovacího hesla a šifrování citlivých osobních údajů. Následně byli implementovány křivky přežití využitím Kaplan-Meierovy metody.

6.7.1 Zabezpečení aplikace

Zabezpečení aplikace bylo provedeno ve třech krocích. Prvním bylo vytvoření dialogového okna, které po uživateli požaduje heslo a šifrovací klíč. Druhým byla implementace autentizace uživatele pomocí hesla a posledním bylo šifrování a dešifrování citlivých osobních údajů.



Obrázek 6.7: Dialogové okno první fáze zabezpečení s výběrem nezabezpečené verze

Implementace dialogového okna

Dialogové okno bylo implementováno v komponentě `LoginForm` a má tři různé fáze ve kterých se může nacházet.

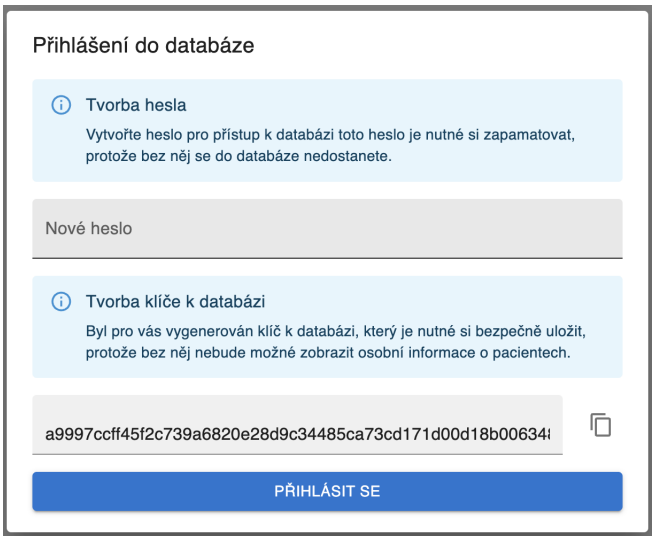
První fáze je zobrazena při úplně prvním spuštění aplikace a nabízí uživateli výběr toho, zda chce používat zabezpečenou verzi aplikace, nebo ne. Možnost byla poskytnuta po dohodě s panem doktorem MUDr. Davidem Kalfeřtem Ph.D., z důvodu, že někteří uživatelé mohou chtít aplikaci pouze testovat a vědí, že nebude obsahovat žádné citlivé údaje. Při reálném využití není tato varianta doporučována. Dialogové okno první fáze, ve kterém je vybrána nezabezpečená verze je vidět na obr. 6.7.

V případě že uživatel zvolí zabezpečenou verzi aplikace je přesunut do druhé fáze dialogového okna, ve které je požádán o vytvoření hesla pomocí, kterého se bude do aplikace přihlašovat a je pro něj vygenerován dostatečně dlouhý a náhodný šifrovací klíč. Dialogové okno je vidět na obr. 6.8 na str. 57. Ke každému vstupnímu poli je vykreslen informační box, který říká k čemu je požadovaný údaj a jakým způsobem s ním pracovat.

Po vyplnění druhé fáze dialogového okna je dokončena inicializace zabezpečení databáze. Nyní je uživateli po každém opětovném spuštění aplikace vykreslena třetí fáze, které požaduje zadání hesla a šifrovacího klíče. Pokud uživatel zadá špatné heslo není mu umožněn přístup. V případě, že zadá správné heslo, ale špatný klíč, je aplikace otevřena bez osobních údajů pacientů.

Implementace autentizace uživatele

Autentizaci uživatele zajišťuje tabulka v databázi s názvem `password` a její příslušný ovladač s názvem `passwordManager.ts`.



Obrázek 6.8: Dialogové okno druhé fáze zabezpečení

Jak již bylo zmíněno uživatel při přihlašování prochází přes tři fáze dialogového okna. Kvůli tomu, aby byla první a druhá fáze vykreslena pouze při prvním spouštění aplikace bylo nutné implementovat funkci `isPasswordSet`, která vrací zda již bylo heslo uloženo do databáze. V případě, že ne, je zobrazena první a druhá fáze dialogového okna. V ní uživatel nastaví požadované heslo, které je uloženo do tabulky `password` ve formě hašovaného řetězce. Nyní pokud dojde k opětovnému přihlášení je provedena kontrola zda hodnota, hašovací funkce zadaného hesla odpovídá uložené hodnotě. Kontrola je provedena pomocí funkce `validatePassword`, která vrací hodnotu `true` pokud si hesla odpovídají, jinak `false`. Po provedení kontroly je uživatel, buď přihlášen do aplikace, nebo požádán o opětovné zadání hesla.

Implementace šifrování a dešifrování

Veškeré šifrovací a dešifrovací operace jsou prováděny v souboru `encryption.ts`, který je součástí modulu `backend`. Šifrování je prováděno pomocí symetrického šifrovacího protokolu Advanced Encryption Standard (AES), který je poskytován modulem `crypto`, jenž je součástí Node.js.

První volaná funkce v `encryption.ts` je `generateEncryptionKey`. Funkce je zavolána při druhé fázi dialogového okna a slouží pro vygenerování šifrovacího klíče pomocí modulu `crypto` viz ukázka zdrojového kódu 6.7.

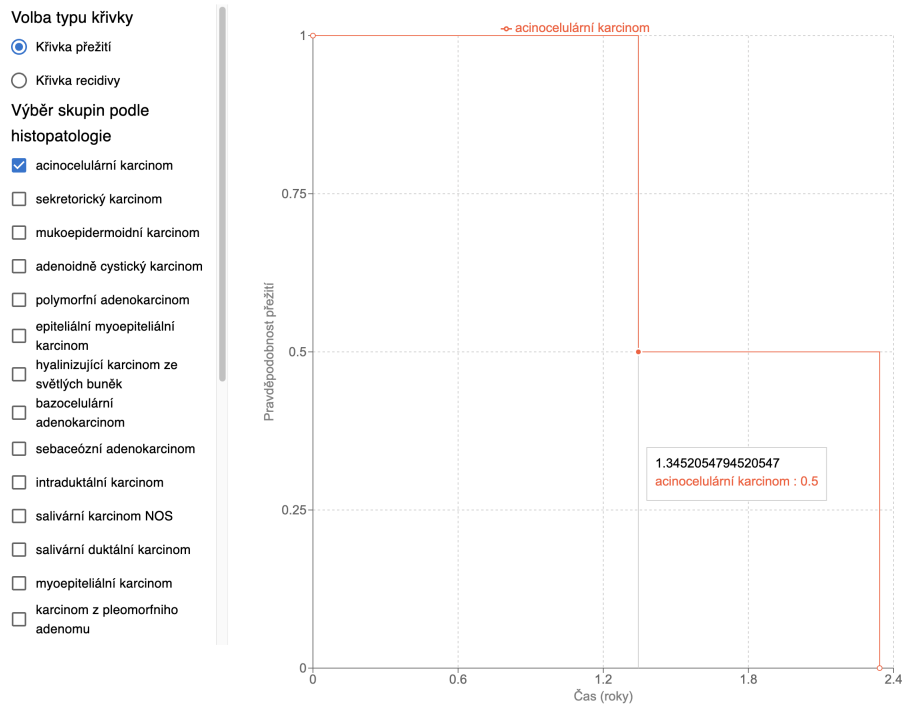
Zdrojový kód 6.7: Funkce pro generování šifrovacího klíče

```
1 export const generateEncryptionKey = (): string => {
2   return crypto.randomBytes(32).toString('hex')
3 }
```

Samotné šifrování a dešifrování provádějí funkce `encrypt` a `decrypt` pomocí modulu `crypto`. Funkce `encrypt` je volána při každém ukládání pacienta do databáze zatímco `decrypt` při načítání pacientů.

6.7.2 Implementace vizualizace křivek přežití

Vizualizaci křivek přežití realizuje komponenta `KaplanMeier` společně s pomocnými komponentami `KaplanMeierChart` a `KaplanMeierFilter`.



Obrázek 6.9: Vizualizace křivek přežití

Komponenta `KaplanMeier` byla implementována tak, aby zobrazovala výběrové menu, které slouží k volbě typu zobrazené křivky a skupin pacientů, pro které je křivka zobrazena. Po výběru těchto parametrů je zobrazena patřičná křivka. Náhled na jednoduchou vizualizaci křivek přežití pro dva pacienty se stejným histopatologickým typem nádoru je vidět na obr. 6.9.

Komponenta KaplanMeierFilter

Úkolem komponenty `KaplanMeierFilter` je zobrazit výběrové menu pomocí, kterého může uživatel vybrat mezi vizualizací křivek přežití nebo recidivy. Dále menu slouží k určení skupin pacientů dle histopatologického typu nádoru, pro které

mají být křivky vykresleny. Potom co uživatel provede výběr je zavolána funkce `getKaplanMeierData`, která načte z databáze všechny potřebné informace pro vykreslení křivek přežití nebo recidivy.

Komponenta KaplanMeierChart

Komponenta `KaplanMeierChart` je zodpovědná za vykreslení křivek podle načtených dat z výběrového menu. Vykreslení křivek je realizováno pomocí knihovny **recharts**, která poskytuje komponenty pro vizualizaci dat.

Získaná data z výběrového menu jsou nejprve zpracována pomocí funkce `calculateKaplanMeierCurveData`, která pro jednotlivé skupiny pacientů vypočítává hodnoty pro křivku přežití nebo recidivy, pomocí vztahu 5.3, který byl odvozen v kapitole zabývající se Kaplan-Meierovou metodou. Při výpočtu jednotlivých pravděpodobností jsou data pro každou skupinu seřazeny podle doby od určení diagnózy do události, kdy nastalo úmrtí nebo recidiva (podle zvoleného typu křivky). Po seřazení dat je proveden samotný výpočet pravděpodobností v jednotlivých časových bodech. Výpočet provádí funkce `calculateKaplanMeierCurveRecords` a její implementace je vidět v ukázce zdrojového kódu 6.8.

Zdrojový kód 6.8: Funkce pro výpočet pravděpodobností křivky přežití (recidivy)

```
1  const calculateKaplanMeierCurveRecords = (
2      kaplanMeierPatientsData: KaplanMeierPatientData[]
3  ): KaplanMeierCurveRecord[] => {
4      // načtení pacientů, kteří mají definované potřebné atributy
5      kaplanMeierPatientsData = kaplanMeierPatientsData.filter(...)
6      // seřazení událostí podle času
7      const sortedEventTimesMap = getSortedEventTimesMap(
           kaplanMeierPatientsData)
8      const records: KaplanMeierCurveRecord[] = []
9      // v čase nula mají všichni pacienti 100% šanci na přežití
10     records.push({ time: 0, probability: 1 })
11     // počet pacientů v ohrožení
12     let numberOfPatientsAtRisk = kaplanMeierPatientsData.length
13     // průchod přes všechny pacienty a výpočet pravděpodobnosti
14     for (const [time, eventsAtTime] of sortedEventTimesMap.entries
           ()) {
15         numberOfPatientsAtRisk -= eventsAtTime
16         const probability =
17             numberOfPatientsAtRisk / kaplanMeierPatientsData.
                   length
18         // přidání dvojice času a pravděpodobnosti
19         records.push({ time, probability })
20     }
```

```
21      // vrácení dat pro vizualizaci křivky
22      return records
23  }
```

Vzhledem k tomu, že v některých případech nemusí mít všichni pacienti definovaný čas události (úmrtí nebo recidivy). Bylo nutné tyto pacienty vyřadit z vzorku dat pro, který se výpočet provádí. Akce vyřazení pacientů je provedená na řádku tři v ukázce zdrojového kódu 6.8 na str. 59.

6.7.3 Oprava nalezených chyb

Poté co byla aplikace v páté iteraci napojena na databázi, byl nalezen problém s vyhledávání pacientů dle jména, příjmení a rodného čísla v komponentě `PatientsList`. Problém nastával v momentě, kdy chtěl uživatel vyhledávat mezi pacienty, kteří jsou součástí studie. Místo toho aby bylo vyhledávání provedeno pouze mezi nimi provedlo se mezi všemi pacienty.

Problém byl vyřešen úpravou funkce `handlePatientSearch` v komponentě `PatientsList`, která nyní zajišťuje vyhledávání pouze mezi pacienty, kteří jsou v patřičném seznamu zobrazeni.

6.7.4 Výsledek šesté iterace

V šesté iteraci byla aplikace zabezpečena pomocí hesla a šifrování citlivých osobních údajů. Dále byla implementována vizualizace křivek přežití a recidivy pomocí Kaplan-Meierovy metody.

6.8 Sedmá iterace

Cílem sedmé iterace bylo opravení chyb, které byly nalezeny ve finální verzi aplikace. Všechny opravené chyby se týkají zejména detailů uživatelského rozhraní.

6.8.1 Změna pořadí tlačítek pro vytváření nových pacientů a studií

Podle informace od pana doktora MUDr. Davida Kalfeřta Ph.D. se nádory slinných žláz vyskytují nejčastěji v žlázách příušních, poté podčelistních a nakonec podjazykových. Proto bylo změněno pořadí tlačítek pro vytváření nových pacientů a studií tak, aby reflektovali tuto skutečnost.

6.8.2 Přejmenování položek formuláře

V části formuláře věnující se diagnostice bylo objeveno chybné pojmenování jedné z možností adjuvantní terapie. Tato možnost byla pojmenována jako „CT“ místo

„CHRT“. Oprava této chyby byla provedena přejmenování této možnosti v komponentách `FormPriusni`, `FormPodcelistni` a `FormPodjazykove`.

Další špatně pojmenovaný atribut formuláře byl „Velikost nádoru - největší rozměr (cm)“ v komponentě `Histopathology`, kde jednotka cm je příliš veliká a byla nahrazena jednotkou mm.

6.8.3 Výsledek sedmé iterace

V poslední iteraci byly opraveny chyby týkající se detailů uživatelského rozhraní. Bylo změněno pořadí tlačítek pro vytváření nových pacientů a studií a byly přejmenovány chybně pojmenované atributy formuláře.

6.9 Testování aplikace

Aplikace byla testována dvěma způsoby. Prvním způsobem bylo manuální testování prováděné panem doktorem MUDr. Davidem Kalfeřtem, Ph.D. po dokončení každé iterace. Druhým způsobem bylo provedení jednotkových testů při každém sloučení kódu do hlavní větve GitHub repozitáře pomocí CI/CD operací.

6.9.1 Manuální testování

Během vývoje byl výstup každé iterace odeslán panu doktorovi, který prováděl manuální testování aplikace. Pan doktor kontroloval, zda jsou jednotlivé části uživatelského rozhraní pojmenovány správně a zda vykonávají operace podle jeho požadavků.

6.9.2 Jednotkové testy

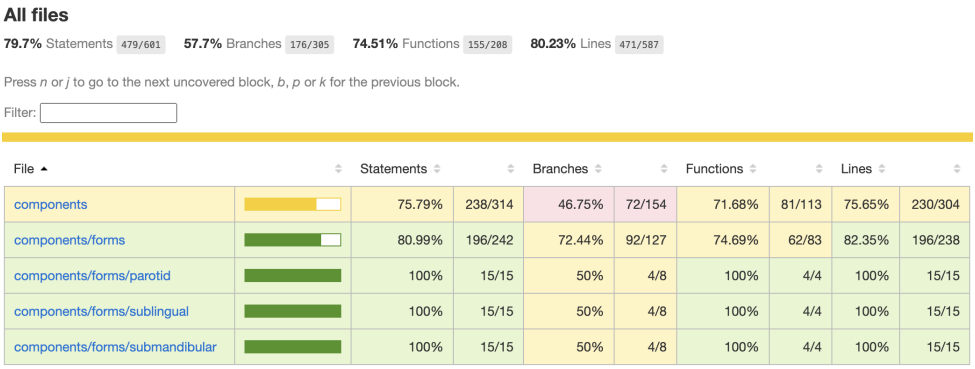
Jednotkové testy byly provedeny pomocí nástroje Jest, který byl integrován do aplikace podle návodu na oficiálních webových stránkách[Jes23]. Shrnutí pokrytí jednotkovými testy je vidět v tab. 6.9.

Tabulka 6.9: Shrnutí pokrytí jednotkovými testy

adresář	příkazy	rozhodovací větve	funkce
`components`	75%	46%	71%
`forms`	81%	72%	75%
`parotid`	100%	50%	100%
`sublingual`	100%	50%	100%
`submandibular`	100%	50%	100%

Testy pokrývají základní funkcionality komponent uživatelského rozhraní a ověřují jejich správné vykreslení. Vzhledem k velkému rozsahu formulářů pro jednot-

livé typy žláz bylo velmi obtížné otestovat všechny možné kombinace zadaných dat. Z tohoto důvodu mají jednotkové testy nižší pokrytí rozhodovacích větví. Celkové pokrytí kódu uživatelského rozhraní aplikace je 80%, a detailnější informace jsou zobrazeny na obr. 6.10, který byl automaticky vygenerován nástrojem Jest.



Obrázek 6.10: Detailní statistika výsledků jednotkových testů

V tabulce na obr. 6.10 je vidět pokrytí jednotlivých modulů uživatelského rozhraní. V levém horním rohu je uvedeno celkové pokrytí adresáře. Významy jednotlivých anglických názvů jsou uvedeny v tab. 6.10.

Tabulka 6.10: Význam anglických názvů ve výsledné statistice pokrytí

anglický název	**význam**
Statements	Pokrytí jednotlivých příkazů.
Branches	Pokrytí rozhodovacích větví.
Functions	Pokrytí funkcí.
Lines	Pokrytí řádků.

Nasazení aplikace 7

7.1 Distribuce aplikace

Aplikace byla vyvíjena pomocí frameworku Electron.js, který umožňuje tvorbu multiplatformních desktopových aplikací. Při vývoji bylo testováno, zda je možné aplikaci sestavit na operačních systémech Windows, macOS a Linux. Avšak kvůli důvodům, které jsou popsány v následujících dvou podkapitolách, bylo rozhodnuto aplikaci distribuovat pouze pro operační systém Windows.

7.1.1 Distribuce pro macOS

V případě distribuování aplikace pro operační systém macOS, by bylo nutné mít k dispozici certifikát od společnosti Apple pomocí, kterého musí být aplikace podepsána. Bez tohoto podpisu není možné aplikaci spustit na macOs. Získání tohoto certifikátu vyžaduje registraci v Apple Developer Program, která je spojena s ročními náklady ve výši 99 dolarů.

Po obeznámení pana doktora MUDr. Davida Kalfeřta Ph.D. s tímto faktem, bylo rozhodnuto, že v této fázi vývoje aplikace nebude distribuována pro macOS.

7.1.2 Distribuce pro Linux

Při konzultaci s panem doktorem MUDr. Davidem Kalfeřtem Ph.D. bylo zjištěno, že uživatelé, kteří budou aplikaci používat, pracují hlavně s operačním systémem Windows. Z tohoto důvodu bylo rozhodnuto, že aplikace nebude distribuována ani pro Linux.

7.2 Sdílení instalačního souboru

V současném stavu je instalační soubor pod názvem `csgdb-setup-1.0.0.exe` dostupný na GitHub repositáři[1].

[1] `https://github.com/vjelinekk/CzechSalivaryGlandDB_v2/releases`.

7.3 Instalace aplikace

Po stažení instalačního souboru `csgdb-setup-1.0.0.exe`, je možné zahájit instalační proces spuštěním tohoto souboru. Po spuštění se uživateli zobrazí okno, které si uživatele ptá, zda důvěřuje aplikaci a chce pokračovat.

Obrázek 7.1: Potvrzení důvěry v aplikaci.

Poté co uživatel potvrdí, že důvěřuje aplikaci viz obr. 7.1, proběhne instalace a aplikace je připravena k použití.

7.3.1 První spuštění aplikace

Při prvním spuštění aplikace má uživatel možnost zvolit mezi zabezpečenou a nezabezpečenou verzí aplikace.

V případě, že uživatel zvolí nezabezpečenou verzi, je mu ihned umožněno využívat aplikaci. Tento způsob není doporučován. Slouží hlavně pro testovací účely nebo pro případy, kdy uživatel nebude ukládat osobní údaje pacientů.

Pokud uživatel zvolí zabezpečenou verzi je požádán o vytvoření hesla, kterým se do aplikace bude přihlašovat. Následně je vygenerován šifrovací klíč, který je nutné si bezpečně uložit. Poté uživatel může začít využívat aplikaci. Základní funkce aplikace jsou popsány v uživatelském manuálu viz příloha A.

Zhodnocení výsledků 8

8.1 Zpětná vazba k finální verzi aplikace

Dle zpětné vazby od pana doktora MUDr. Davida Kalfeřta, Ph.D. program přesně odpovídá jeho zadaným požadavkům viz následující text.
„Program v aktuální verzi je velmi dobře připraven, uživatelsky velmi přehledný a odpovídá přesně zadání. Graficky velmi dobře zpracovaný. V průběhu vytváření programu byla velmi dobrá spolupráce se studentem Vojtěchem Jelínkem. Program bude dále testován s implementací dalších nástrojů (možnost jednoduché deskriptivní statistiky souboru), práce s přílohami, možnost zálohování a zpětného obnovení ze zálohy přiložených příloh k jednotlivým pacientům a další. “

8.1.1 Testování na reálných datech

Veškeré funkcionality finální verze aplikace byly testovány panem doktorem na souboru 20 pacientů. Po provedení testování byla sestavena zpráva, která popisuje, jak jednotlivé části aplikace fungují a zda odpovídají očekávání pana doktora. V následujících několika pododstavcích je uveden zpětná vazba.

Seznam pacientů

Seznam pacientů umožňuje jednoduše prohlížet zadaná data u jednotlivých pacientů. Rozšíření o filtraci velice zvyšuje uživatelskou přívětivost programu a usnadňuje následnou analýzu dat. Vyhledávání mezi pacienty pomocí jména také velmi pozitivně ovlivňuje uživatelskou přívětivost. Způsob výběru paceintů pro export je přirozenější než v předchozí verzi a přidání anonymizace exportovaných dat zajišťuje ochranu osobních údajů pacientů při sdílení nasbíraných dat.

Přidat pacienta

Proces přidávání pacientů velmi usnadňuje provádění automatických výpočtů u TNM klasifikace. Přidávání příloh a jejich zobrazení také pozitivně ovlivňuje celko-

vou uživatelskou přívětivost tvorby pacientů.

Kaplan-Meier

Možnost zobrazit křivku přežití a recidivy v závislosti dle histopatologického typu představuje velmi užitečný nástroj pro hodnocení výsledků léčby.

8.2 Dosažené výsledky

Jedním z největších úspěchů této práce bylo kompletní přepracování aplikace s využitím moderních technologií, především frameworku React.js. Tento krok zásadně zlepšil možnosti rozvoje aplikace, zejména díky eliminaci složitého a chybně navrženého kódu předchozí verze. Nová struktura zdrojového kódu výrazně zjednodušuje rozšíření a úpravy jednotlivých částí aplikace.

8.2.1 Oprava nedostatků

Během implementace nové verze aplikace byly důkladně odstraněny všechny nedostatky v uživatelském rozhraní i ve struktuře zdrojového kódu. Tato oprava nejen zlepšila využitelnost aplikace Czech Salivary Gland Database v klinické praxi, ale také položila pevné základy pro budoucí rozšiřování o další funkcionality.

8.2.2 Přidání vizualizace dat

Pomocí Kaplan-Meierovy metody byla aplikace rozšířena o možnost vizualizace křivek přežití a recidivy pro vybrané skupiny pacientů dle histopatologického typu nádoru.

8.3 Rozšíření aplikace mezi další lékaře

Vzhledem k velké spokojenosti s výslednou aplikací se pan doktor MUDr. David Kalfeřt, Ph.D rozhodl, že bude vytvořenou aplikaci prezentovat na 10. československým kongrese otorinolaryngológie a chirurgie hlavy a krku, a 85. kongres České společnosti otorinolaryngologie a chirurgie hlavy a krku ČLS JEP a 70. kongres Slovenskej spoločnosti pre otorinolaryngológiu a chirurgiu hlavy a krku, který se bude konat od 18. do 20. září roku 2024.

8.3.1 Možnosti budoucích úprav aplikace

Finální verze aplikace poskytuje lékařům vizualizaci křivek přežití a recidivy podle histopatologického typu nádoru. V budoucnu je možné rozšířit aplikaci tak, aby

dokázala generovat křivky na základě více parametrů. Další potenciální vylepšení by mohlo zahrnovat:

1. **Rozšíření možností personalizace**: Umožnění lékařům nastavit si své preference pro zobrazení dat a grafů podle jejich potřeb a pracovních preferencí.
2. **Integrace s existujícími systémy**: Zahrnutí funkcí, které umožní snadnou integraci aplikace s dalšími informačními systémy, které lékaři již používají.
3. **Zlepšení uživatelského rozhraní**: Na základě zpětné vazby od uživatelů průběžně vylepšovat uživatelské rozhraní, aby bylo používání aplikace co nejpříjemnější.
4. **Implementace funkcí pro monitorování a sledování vývoje pacientů**: Vytvoření možností pro sledování dlouhodobého vývoje pacientů a automatické generování upozornění na případné změny nebo potřebné intervence.
5. **Komplexnější analýza dat**: Umožnění komplexnější analýzy nasbíraných dat a přidání mechanismu, který umožní jednoduché sdílení získaných výsledků.

Tato vylepšení by mohla přinést další hodnotu aplikaci a poskytnout lékařům ještě lepší nástroj pro analýzu dat a péči o pacienty.

Závěr 9

Cílem této bakalářské práce bylo rozšíření možností využití aplikace Czech Salivary Gland Database pro klinickou praxi a analýzu dat. Při důkladné analýze předchozí verze aplikace byly identifikovány problémy v uživatelském rozhraní i v kódu. Jednotlivé nedostatky byly rozděleny do dvou skupin dle povahy. Dále byly kategorizovány dle úrovně závažnosti, přičemž pro každý problém bylo navrženo řešení.

V textu byla rovněž rozebrána problematika tvorby zdravotnických databází. Byly představeny konkrétní příklady jejich využití pro klinické a výzkumné účely. Současně byly definovány požadavky na aplikaci a uživatele, které musí být splněny v průběhu využívání a implementace aplikace. Během implementace byl kladen důraz zejména na lokální uložení databáze, zabezpečení dat pomocí šifrování a ochranu aplikace heslem.

Předchozí verze aplikace neobsahovala nástroje pro analýzu nasbíraných dat, proto byla navržena implementace nástroje umožňujícího vizualizaci křivek přežití a recidivy pomocí Kaplan-Meierovy metody. Po identifikaci nedostatků a popsání teorie o vizualizaci křivek přežití, byla provedena implementace s využitím agilních metodik, rozdělením do sedmi týdenních iterací, které byly postupně realizovány. Výstup každé iterace byl testován. Dle zpětné vazby byly provedeny požadované změny v následující iteraci.

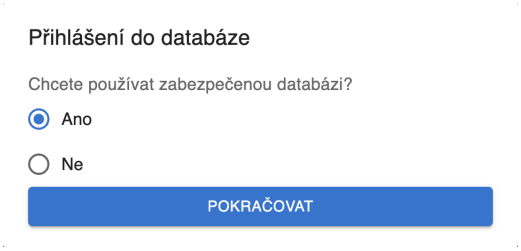
Výsledná aplikace byla podrobena testování na souboru dvaceti reálných pacientů. Na základě výsledků testování a zpětné vazby bylo vyhodnoceno, že aplikace obsahuje všechny požadované funkcionality a byla nasazena na pracovišti Fakultní nemocnice v Motole.

Uživatelský manuál

Kompletní manuál je dostupný v `README.md` na GitHub[1].

Nastavení hesla a zabezpečení

Při prvním spuštění aplikace se otevře dialogové okno, které Vám umožní výběr mezi zabezpečenou či nezabezpečenou verzí aplikace, jak je vidět na obrázku A.1.

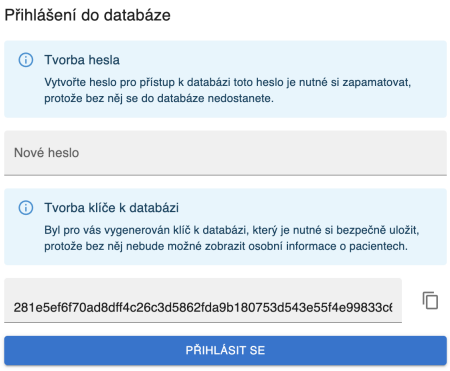


Obrázek A.1: Okno pro výběr zabezpečené databáze

Pokud chcete využívat aplikaci s reálnými daty pacientů, vyberte možnost „Ano“. V případě, že aplikaci chcete pouze testovat se smyšlenými daty, můžete zvolit volbu „Ne“.

Po zvolení zabezpečené verze je zobrazeno další okno viz obr. A.2 na str. 72. V okně na obr. A.2 si nastavte heslo, které budete používat při otevírání aplikace. Toto heslo si pečlivě uchovejte, v případě jeho ztráty nebude možné se dostat k uloženým datům. Dále je pro Vás vygenerován klíč, pomocí kterého jsou šifrována data. Tento klíč si zkopírujte a uložte na bezpečné místo. V případě, že o tento klíč přijdete, nebude možné zobrazit osobní údaje o uložených pacientech.

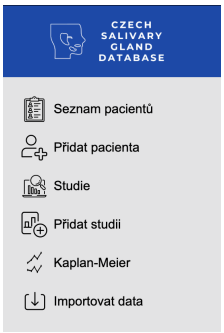
[1] `https://github.com/vjelinekk/CzechSalivaryGlandDB_v2`.



Obrázek A.2: Okno pro nastavení hesla a vygenerování šifrovacího klíče

Navigace aplikací

V levé části okna aplikace se nachází menu, pomocí kterého můžete vykonávat jednotlivé úkony s aplikací viz obr. A.3.



Obrázek A.3: Menu aplikace

V menu se nachází tlačítka, která po stisknutí vykreslují jednotlivé funkcionality aplikace.

- Seznam pacientů – zobrazení seznamu všech pacientů
- Přidat pacienta – přidání jednoho pacienta
- Studie – zobrazení všech vytvořených studií
- Přidat studii – přidání nové studie
- Kaplan-Meier – zobrazení křivek přežití a recidivy
- Importovat data – importování dat z Vašeho počítače

Přidávání pacientů

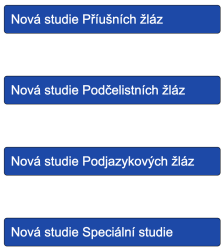
Pokud chcete přidat nového pacienta, klikněte na tlačítko **Přidat pacienta** v menu aplikace. Následně jsou zobrazeny tři tlačítka, pomocí kterých můžete vybrat místo postižené nádorovým onemocněním viz obr. A.4.

Obrázek A.4: Tvorba nového pacienta

Po zvolení žlázy je vykreslen formulář, do kterého můžete ukládat jednotlivé informace o pacientovi a jeho nádoru. Na úplném konci tohoto formuláře se nachází tlačítko **Přidat pacienta**. Po jeho stisknutí je vytvořen nový pacient a zobrazí se seznam pacientů, ve kterém je zvolen Vámi přidaný pacient.

Přidávání studií

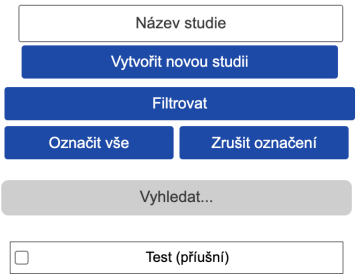
Přidávání nových studií se vyvolá kliknutím na tlačítko **Přidat studii** v menu. To Vám zobrazí okno s tlačítky pro definici typu žlázy viz obr. A.5, které se bude daná studie věnovat.



Obrázek A.5: Výběr typu nové studie

Do speciální studie spadají všechny tři typy žláz. Po zvolení, pro kterou žlázu chcete studii vytvořit, je zobrazeno okno viz obr. A.6 na str. 74.

Nyní zde můžete nadefinovat název studie pomocí vstupního pole a vybrat jednotlivé pacienty, kteří mají být součástí dané studie. Pokud budete chtít studii vytvořit, stačí stisknout tlačítko **Vytvořit novou studii**. Po úspěšném vytvoření budete přesunuti do seznamu studií a nově vytvořená studie bude vybrána.



Obrázek A.6: Výběr pacientů a nastavení názvu nové studie

Exportování pacientů

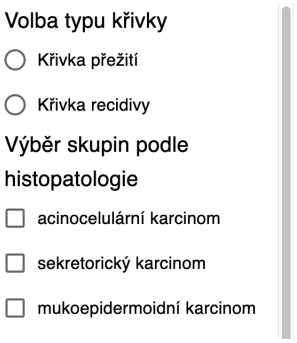
Exportování pacientů můžete provést z libovolného seznamu pacientů. Tím je myšlen seznam vykreslený v části „Seznam pacientů“ nebo v části „Studie“.

Obrázek A.7: Seznam pacientů s tlačítky pro export

Jednotlivé pacienty pro export si můžete zvolit klikáním na zaškrtávací políčka vedle jejich jmen. Pokud chcete zvolit všechny pacienty, stačí kliknout na tlačítko **Označit vše**. Jakmile máte vybrány pacienty, stačí kliknout na tlačítko **Exportovat** nebo **Exportovat anonymizovaně** viz obr. A.7. Pokud zvolíte anonymizovaný export, budou skryty osobní údaje pacientů. Po stisknutí jednoho ze zmíněných tlačítek je zobrazeno okno, ve kterém zvolíte pod jakým názvem budou uložena exportovaná data. Následně je pro každý typ žlázy vytvořen jeden soubor ve formátu `.xlsx`, který obsahuje data všech vybraných pacientů.

Zobrazení křivek přežití nebo recidivy

Pro zobrazení křivek přežití nebo recidivy zvolte položku menu **Kaplan-Meier**. Následně je vykresleno menu, pomocí kterého můžete zvolit typ vykreslené křivky a jednotlivé skupiny pacientů podle histopatologického typu viz obr. A.8.



Obrázek A.8: Menu pro nastavení parametrů vizualizovaných křivek

Po výběru typu křivky a skupiny pacientů dojde k vizualizaci dat. Pro výpočet se berou v potaz jen ti pacienti, kteří mají definovaný histopatologický typ nádoru, rok diagnózy a v případě křivek přežití datum úmrtí a pro křivky recidivy datum prokázání recidivy.

Importování dat

Importování dat vyvolá stisknutí tlačítka **Importovat data**. Následně vyberte soubor s daty pacientů, které chcete importovat. Vybraná data jsou poté vložena do aplikace.

Obsah přílohy B

Vytvořená aplikace a text této bakalářské práce jsou uloženy v přiloženém souboru `A21B0154P_prilohy.zip`. Jednotlivé adresáře jsou uvedeny jako nadpisy v následujícím textu.

Text_prace

Adresář obsahuje přeložený soubor `BP.pdf` s celým textem bakalářské práce. Dále se zde nachází soubor `main.tex` a všechny potřebné soubory pro překlad nástrojem TEX.

Aplikace_a_knihovny

Adresář obsahuje veškerý zdrojový kód aplikace, jednotkové testy a instalační soubor.

- `__tests__` – adresář obsahující jednotkové testy aplikace
- `babel.config.js` – konfigurační soubor pro překlad zdrojového kódu pomocí nástroje Babel
- `db.sqlite` – soubor, který se využívá pro vytvoření databáze při balení aplikace pro nasazení
- `forge.config.js` – konfigurační soubor pro nástroj Electron Forge
- `jest.config.js` – konfigurační soubor pro jednotkové testy
- `package.json` – soubor obsahující definici všech externích knihoven, které aplikace využívá a skripty, které lze spustit v rámci projektu
- `public` – adresář obsahující veškeré využité obrázky v rámci aplikace

- `setupTests.ts` – soubor obsahující základní konfiguraci všech spustitelných testů
- `src` – adresář, který obsahuje front-end a back-end modul aplikace
  - `frontend` – adresář obsahující veškeré komponenty a funkce spojené s uživatelským rozhraním aplikace
  - `backend` – adresář obsahující veškeré funkce pro komunikaci s databází
  - `ipc` – adresář obsahující veškeré funkce pro komunikaci mezi front-end a back-end modulem
  - `index.html` – soubor se základní HTML kostrou aplikace
  - `main.ts` – vstupní bod aplikace
  - `preload.ts` – soubor sloužící k přednačtení potřebných funkcí
  - `renderer.ts` – vstupní bod vykreslovacího procesu
  - `root.tsx` – vstupní bod pro ReactJS komponenty
- `tsconfig.json` – konfigurační soubor pro TypeScript
- `webpack.main.config.ts` – hlavní konfigurační soubor pro Webpack
- `webpack.plugins.ts` – soubor obsahující pluginy využívané nástrojem Webpack
- `webpack.renderer.config.ts` – konfigurační soubor pro vykreslovací proces v rámci nástroje Webpack
- `webpack.rules.ts` – soubor obsahující pravidla pro Webpack
- `csgdb-1.0.0-Setup.exe` – instalační soubor aplikace
- `README.md` – uživatelská příručka a návod pro vlastní sestavení aplikace

Readme.txt

Soubor, který obsahuje text uvedený v této kapitole.

Bibliografie

[Bøj+14] BØJE, Charlotte Rørth et al. Evaluation of comorbidity in 9388 head and neck cancer patients: a national cohort study from the DAHANCA database. *Radiotherapy and oncology : journal of the European Society for Therapeutic Radiology and Oncology*. 2014, roč. 110, č. 1, s. 91–97. Dostupné z DOI: 10.1016/j.radonc.2013.11.009.

[CCN18] C., Amaechi; C., Agbasonu; NWAWUDU, Sixtus Ezenwa. Design and Implementation of a Hospital Database Management System (HDMS) for Medical Doctors. *International Journal of Computer Theory and Engineering*. 2018, roč. 10, s. 1–6. Dostupné z DOI: 10.7763/IJCTE.2018.V10.1190.

[ČR] ČR A ÚSTAV ZDRAVOTNICKÝCH INFORMACÍ A STATISTIKY ČR, Ministerstvo zdravotnictví. *Národní zdravotnický informační portál* [online]. nzip.cz. [cit. 2024-01-05]. ISSN 2695-0340. Dostupné z: https://www.nzip.cz/rejstrikovy-pojem/5375.

[Ele23] ELECTRON.JS. *Quick Start* [online]. OpenJS Foundation, 2023. [cit. 2024-03-12]. Dostupné z: https://www.electronjs.org/docs/latest/tutorial/quick-start.

[For24] FORGE, Electron. *Getting Started* [online]. Electron Forge, 2024. [cit. 2024-03-20]. Dostupné z: https://www.electronforge.io/.

[Git23] GITHUB. *Create a new repository* [online]. GitHub, 2023. [cit. 2024-03-12]. Dostupné z: https://github.com/new.

[GKK10] GOEL, Manish K; KHANNA, Pardeep; KISHORE, Jugal. Understanding survival analysis: Kaplan-Meier estimate. *International journal of Ayurveda research*. 2010, roč. 1, č. 4, s. 274–278. Dostupné z DOI: 10.4103/0974-7788.76794.

[Jag+08] JAGER, Kitty J.; VAN DIJK, Paul C.; ZOCCALI, Carmine; DEKKER, Friedo W. The analysis of survival data: the Kaplan–Meier method. *Kidney International*. 2008, roč. 74, č. 5, s. 560–565. ISSN 0085-2538. Dostupné z DOI: https://doi.org/10.1038/ki.2008.217.

[Jes23] JEST. *Testing React Apps* [online]. OpenJS Foundation, 2023. [cit. 2024-04-01]. Dostupné z: https://jestjs.io/docs/tutorial-react.

[Lee94] LEE, J. Y. Uses of clinical databases. *The American journal of the medical sciences*. 1994, roč. 308, č. 1, s. 58–62. Dostupné z DOI: 10.1097/00000441-199407000-00012.

[Mar24] MARCIN DRYKA, Olga Gierszal. *What Is Electron.js? Pros and Cons* [online]. Brainhub, 2024. [cit. 2024-01-05]. Dostupné z: https://brainhub.eu/library/what-is-electron-js#a-little-bit-of-history-of-electron-framework.

[Mar+20] MARINONE LARES, Stefanie G.; CLARK, Suzanne; MATHY, Jennifer A.; CHAPLIN, Jonathan; MCIVOR, N. Evaluation of a novel database for quality assurance at a head and neck service in New Zealand: an audit of free flap head and neck reconstruction. *ANZ journal of surgery*. 2020, roč. 90, č. 7-8, s. 1386–1390. Dostupné z DOI: 10.1111/ans.15974.

[Sto+06] STOW, Peter J. et al. Development and implementation of a high-quality clinical database: the Australian and New Zealand Intensive Care Society Adult Patient Database. *Journal of Critical Care*. 2006, roč. 21, č. 2, s. 133–141. ISSN 0883-9441. Dostupné z DOI: https://doi.org/10.1016/j.jcrc.2005.11.010.

[Šná] ŠNÁBL, Ivo. *Kaplanův-Meierův odhad funkce přežití* [online]. [cit. 2024-02-10]. Dostupné z: https://portal.matematickabiologie.cz/index.php?pg=aplikovana-analyza-klinickych-a-biologickych-dat--aplikovana-analyza-preziti--neparametricke-odhady--kaplanuv-meieruv-odhad-funkce-preziti.

[Web] WEBPACK. *Getting Started* [online]. Webpack. [cit. 2024-03-20]. Dostupné z: https://webpack.js.org/guides/getting-started.

Seznam obrázků

2.1 Základní rozložení uživatelského rozhraní 8
2.2 Detail pacienta s jménem Test 9
2.3 Struktura aplikace Czech Salivary Gland Database 11
2.4 Zjednodušený ER model databáze 18

3.1 Nová struktura aplikace Czech Salivary Gland Database 24

6.1 Kostra aplikace s menu a vykreslenou komponentou `AddPatient` . . . 42
6.2 Komponenta PatientsList 46
6.3 Komponenta StudyCreation 47
6.4 Komponenta StudiesList 47
6.5 Dialogové okno pro smazání studie 48
6.6 Komponenta FiltrationMenu 53
6.7 Dialogové okno první fáze zabezpečení s výběrem nezabezpečené verze 56
6.8 Dialogové okno druhé fáze zabezpečení 57
6.9 Vizualizace křivek přežití 58
6.10 Detailní statistika výsledků jednotkových testů 62

7.1 Potvrzení důvěry v aplikaci. 64

A.1 Okno pro výběr zabezpečené databáze 71
A.2 Okno pro nastavení hesla a vygenerování šifrovacícho klíče 72
A.3 Menu aplikace 72
A.4 Tvorba nového pacienta 73
A.5 Výběr typu nové studie 73
A.6 Výběr pacientů a nastavení názvu nové studie 74
A.7 Seznam pacientů s tlačítky pro export 74
A.8 Menu pro nastavení parametrů vizualizovaných křivek 75

Seznam tabulek

2.1 Části formuláře pro přidání pacienta 9
2.2 Moduly obsaženy v modulu `renderer` 13
2.3 Importované listenery do `renderer.js` 14
2.4 TNM klasifikace 15

3.1 Kategorizace nedostatků 19

5.1 Tabulka ukazující výpočet hodnot pomocí Kaplan-Meierovy metody . 33
5.2 Příklad výpočtu log-rank testu 34

6.1 Implementační fáze a jejich popis 37
6.2 Iterace a jejich popis 38
6.3 Popis adresářové struktury 40
6.4 Elementární komponenty formuláře 43
6.5 Společné části formuláře 43
6.6 Základní operace s databází 50
6.7 Soubory pro komunikaci s tabulkami 51
6.8 Soubory modulu ipc 51
6.9 Shrnutí pokrytí jednotkovými testy 61
6.10 Význam anglických názvů ve výsledné statistice pokrytí 62

Seznam výpisů

2.1 Funkce pro vytvoření okna 12
2.2 Funkce pro získání dat z formuláře 15
2.3 Funkce pro načítání pacientů 17
6.1 Funkce pro vyhledávání pacientů 45
6.2 Funkce pro vytvoření tabulek 50
6.3 Ukázka použití funkce exposeInMainWorld 52
6.4 Funkce pro přidání pacienta 52
6.5 Funkce pro filtrování pacientů 54
6.6 Vytvoření dialogového okna pro uložení exportovaných dat 55
6.7 Funkce pro generování šifrovacího klíče 57
6.8 Funkce pro výpočet pravděpodobností křivky přežití (recidivy) . . 59

Seznam zkratek

CSGDB – Czech Salivary Gland Database

HTML – Hypertext Markup Language

CSS – Cascading Style Sheets

JS – JavaScript

API – Application Programming Interface

GDPR – General Data Protection Regulation

AES – Advanced Encryption Standard

IPC – Inter Process Communication

CI/CD – Continuous Integration/Continuous Deployment

TS – TypeScript

PR – Pull Request

UR – Uživatelské Rozhraní

DB – Databáze