VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MULTIPLATFORMNÍ PŘEHRÁVAČ ZVUKOVÝCH SIGNÁLŮ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

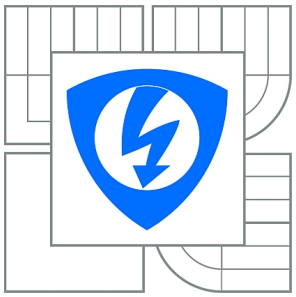
AUTOR PRÁCE
AUTHOR

Bc. JÚLIUS HENZELY

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MULTIPLATFORMNÍ PŘEHRÁVAČ ZVUKOVÝCH SIGNÁLŮ

MULTIPLATFORM AUDIO PLAYER

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

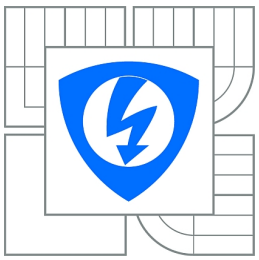
AUTOR PRÁCE — Bc. JÚLIUS HENZELY

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR SYSEL, Ph.D.

SUPERVISOR

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Július Henzely ***ID:*** 106461
Ročník: 2 ***Akademický rok:*** 2013/2014

NÁZEV TÉMATU:

Multiplatformní přehrávač zvukových signálů

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte dostupné multiplatformní knihovny pro tvorbu grafického rozhraní - Gtk, Qt, apod. S použitím vybrané knihovny poté implementujte multiplatformní přehrávač zvukových souborů použitelný v logopedických ordinacích. Přehrávač bude doplněn o jednoduchou databázi pro správu zaznamenaných osob a rozšiřitelný o analýzy řečového signálu.

DOPORUČENÁ LITERATURA:

[1] Blanchete, J.; Summerfield, M. C++ GUI Programming with Qt4. 2nd edition. Prentice Hall, 2008. 752 p. ISBN 978-0132354165
[2] Krause, A. Foundations of GTK+ Development. 1st edition. Apress, 2007. 630 p. ISBN 978-1590597934

Termín zadání: 10.2.2014 ***Termín odevzdání:*** 30.5.2014

Vedoucí práce: Ing. Petr Sysel, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ABSTRAKT

Táto diplomová práca se zaoberá problematikou multiplatformných knižníc na tvorbu grafických aplikácií. V práci sú zahrnuté aj základné teoretické znalosti o systéme relačnej databázy SQLite, ktorý bol použitý v praktickej časti. Základným pilierom praktickej časti práce je zhotovenie multiplatformného prehrávača, ktorý by mal poslúžiť v logopedickej ordinácii. Prehrávač je rozšírený o databázu pacientov a schopnosť tvorby XML súborov z dát v databáze.

KĽÚČOVÉ SLOVÁ

Qt, multiplatformná knžnica, Windows, Arch Linux, databáza, SQlite, XML

ABSTRACT

This master's thesis focuses on problematic of creating applications based on multiplatform framework. This thesis also includes fundamental theoretical knowledge about relational database system SQLite, which has been used in a practical part of the thesis. Programing of multiplatform audio player which could be used in speech-language pathology clinic is essential portion of the practical part. Player was enhanced with the database of patients and ability to generate XML files.

KEYWORDS

Qt, multiplatform framework, Windows, Arch Linux, database, SQLite, XML

HENZELY, Július *Multiplatformní přehrávač zvukových signálu*: diplomová práca. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2014. 50 s. Vedúci práce bol Ing. Petr Sysel, Ph.D.

PREHLÁSENIE

Prehlasujem, že som svoju diplomovoú prácu na tému „Multiplatformní přehrávač zvukových signálu“ vypracoval samostatne pod vedením vedúceho diplomovej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej diplomovej práce ďalej prehlasujem, že v súvislosti s vytvorením tejto diplomovej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Na tomto mieste by som rád poďakoval vedúcemu diplomovej práce panu Ing. Petru Syslovi, Ph.D. za odborné vedenie, konzultácie, trpezlivosť a podnetné návrhy k práci.

Brno . . . . . .

. . . . . .

(podpis autora)

Faculty of Electrical Engineering
and Communication
Brno University of Technology
Purkynova 118, CZ-61200 Brno
Czech Republic

http://www.six.feec.vutbr.cz

POĎAKOVANIE

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

Brno . . . . . .

. . . . . .

(podpis autora)

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

OP Výzkum a vývoj
pro inovace

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Diplomová práca sa zaoberá návrhom multiplatformného prehrávača zvukových súborov. Prehrávač má byť pomôckou na uľahčenie a spríjemnenie práce v logopedickej ordinácii. Zlepšenie má byť najmä vo forme práce s ukladaním nahrávok a ich priradení k jednotlivým pacientom. To sa dosiahne rozšírením prehrávača o jednoduchú databázu v ktorej budú tieto informácie uložené.

V teoretickej časti práce je v prvej kapitole rozobratý framework, ktorý je na prácu použitý. Z dôvodov uvedených v práci bola zvolená knižnica Qt vo verzii 4.8.5 rozšírená o Qt Creator. Ďalej sa teoretická časť zaoberá poznatkami o databázovom systéme SQLite a jeho porovnaním s dvomi open source riešeniami - MySQL a PostgreSQL. Kvôli vyhnutiu sa problémom s licenciou a kvôli ukladaniu databázy do jedného súboru a tým zjednodušeniu jej načítania v Qt bol zvolený ako databázový systém SQLite.

Druhou časťou Diplomovej práce je praktická časť. V prvej polovici je rozpísaný spôsob návrhu a vytvárania databázy a jej jednotlivých tabuliek. Celý postup je zrozumiteľne popísaný a rozšírený o obrázky pre názornejšie predstavenie štruktúry databázy a prepojení medzi jej tabuľkami. Práca pokračuje kapitolou Funkcionalita aplikácie v Qt. Kapitola v krátkosti rozpisuje funkciu jednotlivých ovládacích prvkov prehrávača a spôsob práce s ním. Ďalšie kapitoly potom detailne popisujú celú funkčnosť jednotlivých okien a funkcií, ktoré sú za pomoci ovládacích prvkov spúšťané. Kapitoly tiež obsahujú obrázky zobrazujúce vzhľad a rozmiestnenie prvkov v oknách operačného systému Windows 7. V prílohách sú potom obrázky vzhľadu okien v systéme Arch Linux s desktopovým prostredím GNOME 3.

1 KNIŽNICA QT

Qt je multiplatformný aplikačný framework, ktorý sa používa na tvorbu aplikácii s GUI - grafickým užívateľským rozhraním ale tiež na nástroje bez GUI akými sú konzolové aplikácie.[8] Qt štandardne používa C++, ale tiež značne využíva Meta Object Compiler (špeciálny generátor kódu)a makrá.[2] Funguje na väčšine desktopových platforiem a tiež niektorých mobilných platformách. Medzi využitia bez grafického rozhrania patrí prístup k SQL databázam, podpora sietí a jednotné multiplatformné API aplikačné rozgranie na prácu so súbormi. Framework je dostupný pod komerčnou licenciou a licenciami GPL v3 a LGPL v2.[6] Všetky edície podporujú veľa kompilátorov, medzi inými aj GCC a kompilátor balíka Visual Studio.

Qt je vyvýjané spološnosťou Digia, ktorá vlastní ochrannú známku QT. Qt Project je pod otvorenou kontrolou spoločnosti a podporuje individuálnych vývojárov a firmy pri rozširovaní a zlepšovaní frameworku. Pred spustením Qt Project-u bolo Qt vyvýjané Nokiou a jej oddelením Qt Development Frameworks.[8] Oddelenie vzniklo po zakúpení prvotného vývojára Qt, spoločnosti Trolltech, firmou Nokia.

Obr. 1.1: Logo Qt

Po rozhodnutí Nokie o opustení platformy Symbian a sústredení sa na platformu Windows Mobile, spoločnosť nemala ďalší záujem o podporu Qt a projekt predali spoločnosti Digia. Bezprostredným cieľom sa stala podpora Qt na platformách Android, iOS a Windows 8 bez zmien v pokračovaní podpory desktopov. Medzi aktuálne podporované platformy patria Windows, OS X, X11 Linux, Wayland, QNX / Balckberry 10, Android, iOS a VxWorks.[1] Programy postavené na frameworku Qt sú napr. Autodesk Maya, BlackBerry, Mathematica, Google Earth, Skype, VirtualBox a VLC media player.[6]

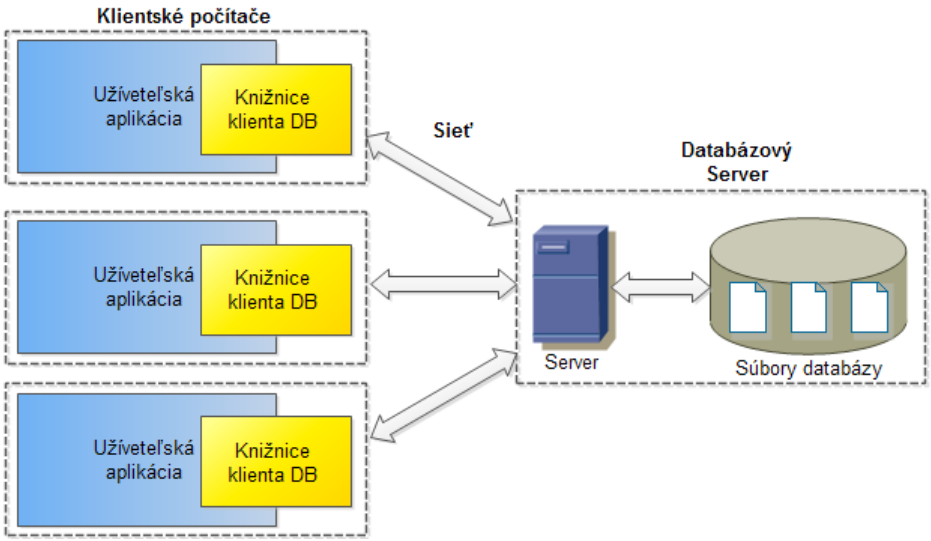
2 SQLITE

SQLite je softvérový balík, ktorý poskytuje systém organizácie relačnej databázy – RDBMS (relational database management system). Z pohľadu licencie sa jedná o public-domain (verejné vlastníctvo), takže nie je chránený autorským právom. Systém relačnej databázy sa používa na ukladanie užívateľských záznamov v rozsiahlych tabuľkách. Okrem ukladania dát databázový systém spracúva zložité query príkazy (dotazy), kde sa kombinujú dáta z viacerých tabuliek na tvorbu výslednej správy. „Lite“ v názve SQLite neodkazuje na nedostatok schopností systému, ale skôr na jednoduchosť nastavenia, administrácie a nenáročnosti na využívanie systémových prostriedkov.[4]

Medzi známe komerčné systémy správy databáz patria Oracle Database, IBM DB2 a SQL Server od Microsoftu. Čo sa týka open source riešení patria tam okrem iných veľmi populárne MySQL a PostgreSQL.

2.1 Vlastnosti SQLite

Rozsiahle databázové systémy so sebou zvyčajne prinášajú aj veľký serverový balík, ktorý tvorí databázový stroj. Databázový server pozostáva z viacerých procesov, ktoré súčasne spravujú pripojenie klientov, vstupy a výstupy súborov, optimalizáciu a vykonávanie dotazov. Inštancia databázy zvyčajne pozostáva z množstva súborov uložených v jednej alebo viacerých zložkách v súborovom systéme serveru. [4]

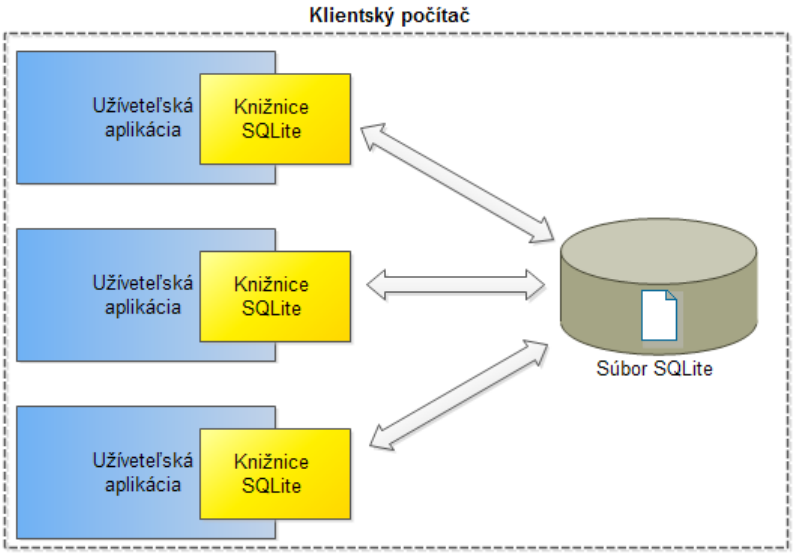


Obr. 2.1: Architektúra tradičného klient/server databázového systému

Aby bolo možné k databáze pristupovať, je nutná prítomnosť a bezchybnosť všetkých súborov, z čoho vyplýva nepohodlnosť pri zálohovaní alebo premiestňovaní databázy. Tieto súčasti využívajú prostriedky hostiteľského počítača. Na hostiteľskom systéme má byť nakonfigurovaný dedikovaný systémový účet, štartovné skripty a vyhradená pamäť na disku. To a požiadavky na výkon sú dôvody, prečo sa zvyčajne pre databázový server vyhradí celý počítač.

Na prístup k databáze potrebuje klientská aplikácia obsahovať knižnice, ktoré poskytujú aplikačné rozhranie (API – application interface) na pripojenie sa k databáze a na vykonávanie databázových dotazov a príkazov. [5]

Oproti väčšine RDBMS SQLite nemá architektúru typu klient/server. Celý databázový engine je integrovaný do aplikácie ktorá vyžaduje prístup k databáze. Jediná zdieľaná časť medzi aplikáciami je samotný súbor databázy na disku. Pri premiestňovaní alebo zálohe potom stačí vytvoriť kópiu súboru. Vylúčením serveru sa odstráni nezanedbateľná časť zložitosti. [5]



Obr. 2.2: Architektúra databázy SQLite

Absenciou serverovej časti sa zjednodušia softvérové komponenty a odstráni potreba zložitejšej podpory operačného systému, čo sa týka multitaskingu a vysokého výkonu pri komunikácii medzi procesmi. SQLite vyžaduje pre svoju funkčnosť len o trochu viac ako je schopnosť čítať a zapisovať do databázového súboru.[4] Tým sa stáva prenesenie aplikácie na iné platformy, ako napr. mobilné telefóny, multimediálne prehrávače alebo herné konzoly priamočiarym.

SQLite je navrhnutý na priamu integráciu do spustiteľného súboru, to odstráni potrebu externých knižníc. To spôsobí, že distribúcia a inštalácia sa stáva jednoduch-

šou a odstránia sa problémy s knižnicami rôznej verzie na klientovi a serveri, keďže je kód priamo vstavaný do aplikácie. Nesie to so sebou problém, že SQLite nie je veľmi dobre prispôsobený na situáciu, ktorá nabáda použitie architektúry klient/server, napr. keď viacero klientov chce pristupovať k jedinej centralizovanej databáze. [5]

Ďalšou vlastnosťou SQLite je zapuzdrenie celej databázy do jedného súboru. Tento súbor obsahuje usporiadanie tabuliek a tiež uložené dáta v týchto tabuľkách a indexy medzi tabuľkami. Formát súboru je rovnaký na všetkých platformách bez rozdielov. Dôsledkom, že je celá databáza v jednom súbore, je triviálne jej kopírovanie a zálohovanie.[4] Celá databáza môže byť presunutá, modifikovaná a zdieľaná, tak jednoducho ako súbor textového dokumentu. Databáza sa tiež neznehodnotí tým, že jeden z mnoho súborov bol náhodou presunutý alebo premenovaný.

Z pohľadu koncového užívateľa, nie je potrebné nič inštalovať alebo konfigurovať. Zatiaľ čo pre programátora existuje dostatočná flexibilita v nastaveniach databázy, vloženie databázového enginu priamo do aplikácie má za následok, že koncový užívateľ nemusí ani tušiť, že pracuje s databázou.[5]

Nízke nároky na systémové prostriedky robia z SQLite ideálnu voľbu pre systémy s obmedzeným operačným systémom. Použitie jazyka ANSI C na napísanie zdrojového kódu spĺňa nároky väčšiny kompilátorov hoci aj málo používaných vstavaných (embedded) procesorov. Pri použití predvolenej konfigurácie, skompilovaná SQLite knižnica má na väčšine platforiem menej ako 700 kB a vyžaduje menej ako 4 MB pamäte. Pri vynechaní pokročilejších funkcií sa dá knižnica orezať na menej ako 300 kB alebo a pri minimálnych zmenách konfigurácie si vystačí aj s použitím len 256 kB pamäte. SQLite vyžaduje len minimálnu podporu od hostiteľského prostredia a je napísané modulárnym spôsobom. Vnútorný alokátor pamäte môže byť jednoducho upravený alebo vymenený a prístup k súborom a diskovej pamäti sa vykonáva prostredníctvom rozhrania virtuálneho súborového systému (VFS – virtual file system), ktorý môže byť upravený k požiadavkám rozličných platforiem. Obecne SQLite dokáže bežať na čomkoľvek s 32-bitovým procesorom. [4]

2.1.1 Unikátne vlastnosti SQLite

SQLite používa systém s dynamickými dátovými typmi v tabuľkách. Preto dovoľuje uloženie ľubovoľných hodnôt no väčšiny stĺpcov tabuliek bez ohľadu na ich typ. To je významný rozdiel oproti tradičným databázovým systémom, kde je dátový typ stĺpca statický. Podobá sa to skriptovacím jazykom, ktoré často majú jediný skalárny dátový typ na prácu s ľubovoľnými dátami, či už sa jedná o INTEGER alebo znakový reťazec. Ďalšou praktickou vlastnosťou je schopnosť manipulácie s viacerými databázami súčasne. To dovoľuje spracovanie SQL príkazov rozprestrených medzi viacero databáz, spájanie tabuliek pomocou jediného query alebo kopírova-

nie objemných dát pomocou jediného príkazu. SQLite dovoľuje vytvorenie databázy uloženej výhradne v operačnej pamäti (RAM). Tieto databázy nemajú veľkú trvanlivosť, ale sú omnoho rýchlejšie (v prípade dostatku RAM) a sú dobrým riešením na ukladanie dočasných tabuliek a iných prechodných dát. Niektoré funkcie sú založené na vlastnostiach iných databázových systémov avšak s miernymi zmenami ako napr. virtuálne tabuľky. Tieto vlastnosti a rozšírenia poskytujú nástroje na prispôsobenie SQLite takmer každej situácii alebo problému.

Obr. 2.3: Logo SQLite

SQLite a jeho zdrojové kódy nemajú žiadnu užívateľskú licenciu. Nie sú kryté GNU General Public Licence (GPL) ani žiadnou podobnou open source/free source licenciou. Vývojári SQLite sa dobrovoľne vzdali práva na vlastníctvo kódu alebo odvodených produktov. Preto s kódom SQLite je možné robiť všetko okrem prehlásenia o jeho vlastníctve. Kód a knižnice môžu byť použité, modifikované, predávané a distribuované ľubovoľným spôsobom. Preto vývojári kladú veľký dôraz na vyhnutie sa potenciálnym softvérovým a algoritmickým patentom. Všetky príspevky do zdrojových kódov SQLite vyžadujú formálne uvoľnenie autorstva. V prípade komerčných prispievateľov vyžadujú aj čestné vyhlásenie podpísané prispievateľom (prípadne aj zamestnávateľom), že svoju prácu uvoľňujú ako verejné vlastníctvo. Preto je zaistené, že integrácia SQLite do ďalších produktov so sebou nesie minimálnu právnu záťaž.[4]

Účelom databázy je usporiadané uchovávanie dát, preto sa používajú databázové systémy, ktoré majú minimálnu chybovosť a sú spoľahlivé. Na zachovanie spoľahlivosti sa jadro knižnice hĺbkovo testuje pred každým uvoľnením novej verzie. Štandardné testy pozostávajú z viac ako 10 miliónov query testov. Pred uvoľnením ďalšej verzie sa spúšťa balík záťažových tzv. soak testov, ktorý je zostavený z vyše 2,5 miliardy testov. Balík testuje 100% pokrytie jednotlivých vetiev programu a tiež ich správne rozhodovanie, vrátane hraničných prípadov ako napríklad vyčerpanie pamäte alebo miesta na ukladanie dát.[4] Vysoká úroveň testovania udržuje počet chýb minimálny. Zvyčajne sa však nejedná o chyby ktoré by viedli k strate dát alebo znehodnoteniu databázy, ale skôr k zníženiu jej výkonnosti, lebo sa operácie nevykonávajú optimálnym spôsobom. S tým je spojená aj spätná kompatibilita a

aktualizácia na novú verziu so sebou málokedy prináša nejaké problémy.

2.2 Použitie SQLite

Hlavným cieľom SQLite je jednoduchosť. Veľa ľudí používa databázy SQLite kvôli ich malej veľkosti a rýchlosti. Tieto vlastnosti sú len dôsledkom toho, že menšia komplexnosť znamená existneciu menej miest, kde sa môže niečo pokaziť. Táto jednoduchosť však môže byť aj slabosťou podľa toho, čo sa pri implementácii snaží používateľ dosiahnuť. Pre dosiahnutie jednoduchosti bolo potrebné obetovať iné vlastnosti, ktoré by v určitých prípadoch boli užitočné, ako napríklad rozsiahlejší set zabudovaných funkcií a procedúr, rozšírenia pre XML alebo Javu alebo zápis dát viacerých užívateľov súčasne. V prípade nutnosti týchto funkcií sa SQLite nejaví ako najvhodnejšie riešenie a odporúča sa skôr použitie rozsiahlejšieho databázového systému.[4] Pokiaľ je dôležitejšia jednoduchosť údržby, administrácie a implementácie ako zložitejšie funkcie, tak je SQLite vhodným kandidátom.

2.2.1 Vhodné aplikácie pre databázu SQLite

SQLite sa dá použiť ako diskový formát súborov pre aplikácie ako napr. nástroje na finančné analýzy, balíčky pre CAD systémy, ukladanie a úpravu multimediálnych súborov a programy na uchovávanie záznamov. Tradičný postup Súbor - Otvoriť zavolá funkciu `sqlite_open()` na pripojenie databázového súboru, ktorý sa aktualizuje atomicky pri používaní obsahu aplikácie. Tým sa stáva operácia Súbor - Uložiť nadbytočnou. Ďalšími výhodami použitia SQLite databázy ako súborový formát sú:

1. Obsah môže byť prístupný a aktualizovaný za použitia SQL query, čím sa redukuje zložitosť kódu aplikácie.
2. Rozšírenie formátu o nové funkcie sa vykonáva pridaním tabuliek alebo pridaním ďalších stĺpcov do tabuliek existujúcich.
3. Rôznorodý obsah, ktorý by inak bol uložený ako množstvo súborov môže byť zapuzdrený do jedného súboru a jeho obsah môže byť čitateľný za použitia nástrojov tretích strán.
4. Takýto súbor je prenositeľný medzi všetkými operačnými systémami.
5. Aplikácia si načíta len dáta, ktoré potrebuje. Znižuje sa tak potrebný čas a pamäťová náročnosť aplikácie.
6. Malé úpravy prepisujú len časti súboru, ktoré sa menia a nie celý súbor, takže sa jedná o vhodnú voľbu napríklad pre SSD disky.

7. Obsah sa aktualizuje neprestajne a atomicky, tak v prípade poruchy alebo výpadku sa nestratí vykonaná práca.
8. Nedostatočná rýchlosť systému sa dá často vyriešiť použitím `CREATE INDEX` namiesto prepisovania a znovu testovania aplikácie.
9. Rozdielne programy napísané v rozdielnych programovacích jazykoch môžu pristupovať k rovnakým aplikačným súborom, bez problému s kompatibilitou.
10. Viac procesov môže pristupovať k jednému aplikačnému súboru a čítať resp. zapisovať bez toho aby si navzájom prekážali.
11. Zvyčajne je načítanie obsahu rýchlejšie z databázy ako z jednotlivých súborov.

SQLite dovoľuje databázovým súborom mať ľubovoľnú súborovú príponu. Aplikácia si teda môže podľa potreby zvoliť príponu vlastnú a môže sa s ňou asociovať v operačnom systéme.

Ďalšou možnosťou je použitie SQLite ako databázového enginu pre web stránky s nízkou alebo strednou návštevnosťou. Množstvo webovej prevádzky, ktorú SQLite zvláda záleží od toho ako veľmi stránka databázu využíva. Všeobecne sa dá povedať, že stánky okolo 100 000 návštev za deň by mali fungovať bez problémov, pričom bolo preukázané že SQLite dokáže pracovať aj so záťažou desaťkrát vyššou. [5]

Programy často využívajú funkcie `fopen()`, `fread()` a `fwrite()` na prácu so súbormi vo vlastnom formáte. Ako náhrada týchto ad hoc súborov sa dá použiť SQLite.

AK je potrebné triedenie alebo zoradenie veľkého objemu dát, je často rýchlejšie tieto dáta načítať do databázy v operačnej pamäti a použiť query s funkciou `JOIN` a `ORDER BY` na získanie výstupu dát v požadovanom formáte a poradí, ako manuálne programovanie rovnakých operácií. Použitie SQLite databázy týmto spôsobom pridá programu väčšiu flexibilitu, pretože nové indexy a stĺpce môžu byť pridané bez zmien už existujúcich query.

Skúsení používatelia môžu spustiť sqlite3 v príkazovom riadku na analýzu zmiešaných dát. Nespracované dáta sa dajú importovať vo forme CSV súborov a následne orezať a skombinovať na vytvorenie rôznych hlásení napr. analýzy návštevnosti webovej stránky, športové štatistiky alebo výsledky experimentov. To sa samozrejme dá dosiahnuť aj za pomoci iných databázových systémov, výhoda SQLite je v jeho rýchlej konfigurácii a ukladaní databázy do jediného súboru, ktorý sa dá následne jednoduchšie šíriť. Posledným veľmi dobrým spôsobom použitia SQLite je jeho využitie na pedagogické účely pri výuke jazyka SQL ale aj pri štúdiu implementácie RDBMS. Zdrojový kód SQLite,ktorý je modulárny dobre okomentovaný a zdokumentovaný môže poslúžiť ako dobrý základ. [5]

2.2.2 Situácie kde je vhodnejšie použiť iný databázový systém

V prípade, že k databáze potrebuje pristupovať viac programov prostredníctvom siete, je vhodné použiť databázu s architektúrou klient/server. SQLite dokáže pracovať aj cez sieť, ale oneskorenie spojené s väčšinou sieťových súborových systémov spôsobí, že výkon databázy bude slabší. Na sieťových súborových systémoch existujú chyby pri zamykaní súboru, preto sa môže stať že dvaja alebo viac klientov bude naraz upravovať rovnakú časť databázy a tým sa databáza znehodnotí. Pretože sa jedná o chybu súborového systému, SQLite tejto chybe nevie predísť. V prípade že k jednej databáze bude pristupovať viac počítačov súčasne je lepšie sa použitiu SQLite vyhnúť. Rovnako to je aj v prípade, že sa jedná o databázu často vyťaženej stránky, kde je vhodné databázu oddeliť a premiestniť na oddelený počítač. [5]

Veľkosť databázy SQLite je obmedzená na 128 TB. Aj v prípade, že by bolo možné ukladať väčšiu databázu, problém nastáva v tom, že je celá databáza uložená v jednom súbore. Taký veľký súbor nepodporuje väčšina súborových systémov, preto pri nutnosti ukladania takého množstva dát je lepšie využiť databázu ktorá dáta ukladá do viacerých súborov a prípadne na viacero diskov. [4]

SQLite nemá implementovaný žiaden vlastný systém riadenia prístupu k databáze, preto jediný spôsob ako regulovať prístup je závislý na súborovom systéme. Tým prakticky obmedzenia prístupu spadajú do jednej z troch kategórií: úplný prístup k zapisovaniu a čítaniu, prístup iba k čítaniu a žiaden prístup. Prístup zapisovania je absolútny a vzťahuje sa aj k modifikácii štruktúry databázy. Tým sa stáva použitie SQLite nevhodné na uchovávanie citlivých dát.[4]

Posledný problém nastáva s replikáciou databázy. SQLite nemá žiadnu vnútornú podporu na vytvorenie kópie databázy. Je možné vytvoriť kópiu databázy jednoduchým skopírovaním súboru, ale to je možné len v prípade, že nikto databázu nemodifikuje. [5]

2.2.3 Porovnanie RDBMS

V nasledujúcej tabuľke sa nachádza porovnanie niektorých vlastností často používaných databázových systémov. Hoci SQLite oproti výkonnejším systémom poskytuje len základné funkcie, na implementáciu do aplikácie je vhodný kvôli svojej jednoduchosti. Pre databázy PostgreSQL a MySQL je potrebné nastavovať aj databázový server a ich tabuľky sú vo viacerých súboroch. Nevýhodou SQLite je málo používaných dátových typov a hoci poskytuje podporu dátových typov jazyka SQL, interne si ich prevádza na jeden z piatich podoporovaných. Rozhodujúcou vlastnosťou bol typ licencie SQLite.[3]

Tab. 2.1: Porovnanie SQLite, MySQL a PostgreSLQ

	SQLite	**MySQL**	**PostgreSQL**
Správca	D. Richard Hipp	Sun Microsystems/ Oracle Corporation	PostgreSQL Global Development Group
Rok uvedenia	2000	1995	1989
Aktuálna verzia	3.8.0.2	5.6.31	9.3.2
Typ licencie	Public domain	GPL/Proprietárna	Liberálna Open Source
ACID	Áno	Áno	Áno
Referenčná integrita	Áno	Čiastočná	Áno
Databázové transakcie	Áno	Áno okrem DDL	Áno
Unicode	Voliteľné	Áno	Áno
Rozhranie	API a SQL	GUI a SQL	API, GUI a SQL
Maximálna veľkosť DB	128 TB, 2^{31} stránok, 64 KB max veľkosť stránky	Neobmedzená	Neobmedzená
Maximálna veľkosť tabuľky	Limitovaná veľkosťou súboru	MyISAM – 256 TB Innodb 64 TB	32 TB
Maximálna veľkosť riadku	Limitovaná veľkosťou súboru	64 KB	1,6 TB
Maximálny počet stĺpcov	32767	4096	250 -1600 závislé na type
Maximálna veľkosť Blob/Clob	2 GB	4 GB (`longtext`, `longblob`)	1 GB (`text`, `bytea`) 4 TB (`pg_largeobject`)
Maximálna veľkosť CHAR	2 GB	64 KB (text)	1 GB
Maximálna veľkosť NUMBER	64 bitov	64 bitov	Neobmedzená
Maximálna veľkosť názvu stĺpca	Neobmedzená	64	63

Partition	Nie	Áno	Áno
Typový systém	Dynamický	Statický	Statický
Schopnosti DB			
Zjednotenie	Áno	Áno	Áno
Prienik	Áno	Nie	Áno
Rozdiel	Áno	Nie	Áno
Blob/Clob	Áno	Áno	Áno
Common table expresions	Nie	Nie	Áno
Paralelné query	Nie	Nie	Nie
Kontrola prístupu	Nie	Čiastočná	Áno

3 PRAKTICKÁ ČASŤ DIPLOMOVEJ PRÁCE

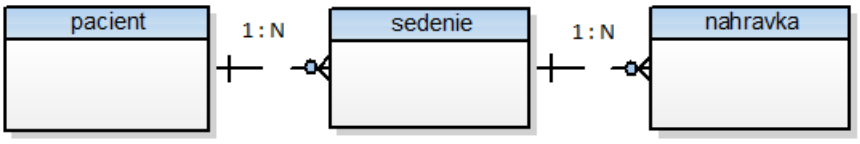
Úlohou diplomovej práce bolo rozšírenie aplikácie slúžiacej na prehrávanie a nahrávanie audio súborov v logopedickej ambulancii o jednoduchú databázu. Databáza mala obsahovať údaje o nahrávkach a pacientoch, ku ktorým sa náhravky viažu. Nasledujúci text sa bude na začiatku zaoberať návrhom tejto databázy a v druhej časti jej implementáciou do existujúcej aplikácie. Návrh databázy sa opieral o informácie obsiahnuté v opore k predmetu IDS (databázové systémy) na Fakulte informačných technológií.[9] Dôležitou časťou je tiež rozšírenie aplikácie o nové okná slúžiace práve na ukladanie a načítanie dát z databázy.

3.1 Návrh databázy v SQLite

Návrh databázy vychádzal z jednoduchej schémy v ktorej by boli použité len tri tabuľky. Jednalo sa o tabuľky na uloženie informácií o pacientovi, jeho sedeniach a nahrávkach, ktoré by sa vytvorili počas sedení. Sedenia by mali atribút dátumu a poradia. Posledným atribútom by bol identifikátor slúžiaci ako primárny kľúč v rámci tabuľky. Tabuľka nahrávok by bola navrhutá s atribúrmi o názve nahrávky, ktorý by pozostával z priezviska pacienta a dátumu nahrávania. Tento reťazec je nevhodný pre vyhľadávanie, preto bola tabuľka rozšírená index slúžiaci ako primárny kľúč. Tabuľka pacienta by obsahovala potrebný počet polí na zadanie základných údajov pacienta ako sú:

- meno
- priezvisko
- pohlavie
- rodné číslo
- dátum narodenia
- diagnóza

ER (entity relationship) diagram tejto databázy je na nasledujúcom obrázku.

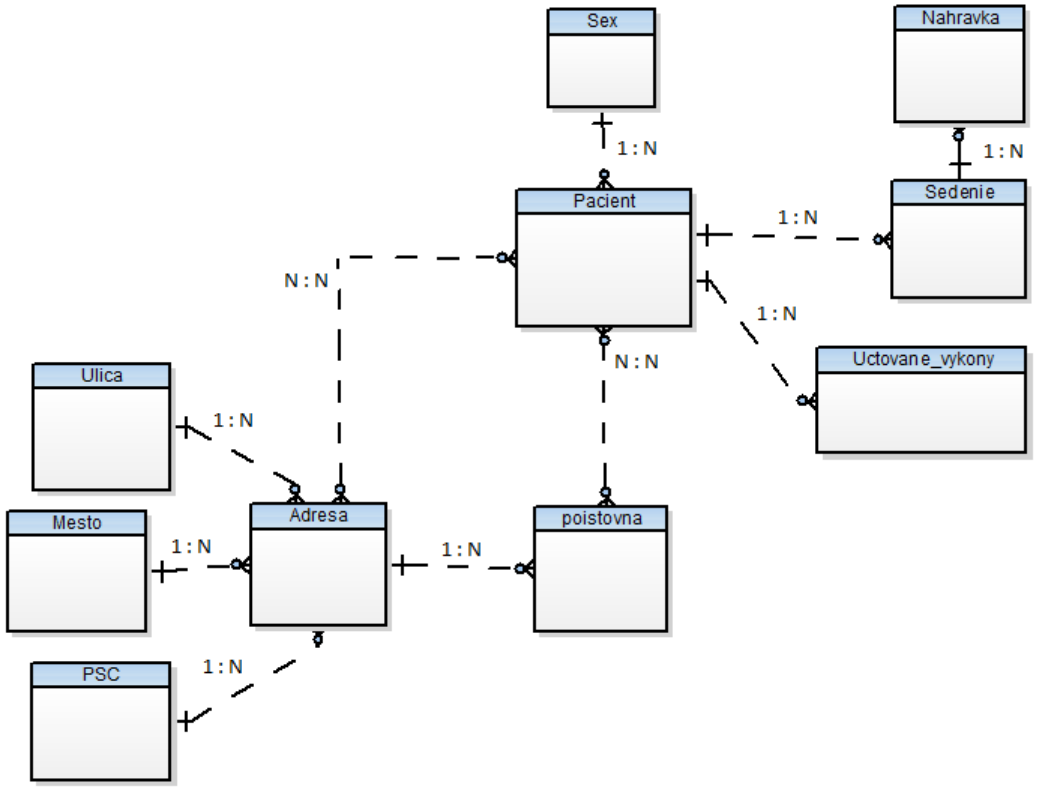


Obr. 3.1: ER diagram prvotného návrhu databázy

Za cieľom zníženia redundancie v databáze bolo vhodné vytvoriť samostatnú tabuľku pre zadávanie pohlavia pacienta. Preto, že sa jedná o údaj, ktorý by mal

mať len dve možné hodnoty, v databáze by sa nachádzalo veľa rovnakých dát.[9] Atribút pohlavia bol v tabuľke pacient nahradený cudzím kľúčom, ktorý odkazuje na tabuľku pohlavie.

V budúcnosti má byť aplikácia rozšírená aj o možnosť exportovať lekársku správu vo formáte XML. Tento fakt bol pri finálnom návrhu databázy najdôležitejší. Štruktúra a obsah tejto lekárskej spávy je určený Ministerstvom zdravotníctva Českej republiky v dokumente „DATOVÝ STANDARD MZ ČR“. Pomocou dokumentu bola tabuľka pacienta rozšírená o ďalšie polia, ktoré sú potrebné pre úplnosť tejto správy. Tým sa databáza rozšírila o tabuľky s údajmi o poisťovniach, účtovaných výkonoch a adresách. Atribúty boli pomenované tak aby zodpovedali premenným použitým v dokumente napr. urgentné data sú označené ako „u“, diagnóza ako „dg“, anamnéza „an“ alebo rodné číslo ako „rodcis“. V tabuľke adresa boli pôvodne aj informácie o ulici, PSČ a meste. To sú ďalšie atribúty, ktoré by vytvárali redundanciu v databáze, preto boli nahradené cudzími kľúčmi a samostatnými tabuľkami. Výsledný zjednodušený ER diagram je na nasledujúcom obrázku.



Obr. 3.2: ER diagram databázy s označenou kardinalitou

V dokumente sa ďalej spomína nutnosť pripojenia viacerých adries a poisťovní k pacientovi. Tým vznikájú v databáze vzťahy s kardinalitou N:N. Riešením je po-

užitie dvoch väzobných tabuliek. ER diagram sa tak rozšíril o väzobné tabuľky s cudzími kľúčmi k tabuľkám pacient a adresa resp. pacient a poisťovňa. Jednotlivým atribútom boli priradené dátové typy ukladaných dát. Ako je jasné z úplného ER diagramu, ktorý sa nachádza medzi prílohami, boli použité len dva dátové typy INTEGER a TEXT. Je to kvôli tomu, že SQLite, hoci pozná dátové typy jazyka SQL, dáta ukladá v jednom z piatich vlastných dátových typov podľa nasledujúcich pravidiel.[4]

1. Ak deklarovaný typ obsahuje reťazec „INT“, tak je priradená afinita `INTEGER`
2. Ak deklarovaný dátový typ obsahuje ľubovoľný z reťazcov „CHAR“, „CLOB“ alebo „TEXT“ tak má stĺpec priradenú afinitu `TEXT`. Numerické hodnoty priradené napr. k VARCHAR(255) sú v SQLite ignorované a maximálny rozsah je obmedzený na $2^{31} - 1$ znakov.
3. V prípade, že je dátový typ BLOB alebo dátový typ stĺpca nie je nastavený, výsledná afinita stĺpca tabuľky je `NONE`.
4. Ak deklaráciou určený dátový typ obsahuje niektorý z reťazcov „REAL“, „FLOA“ alebo „DOUB“, stĺpcu je priradená afinita `REAL`.
5. V prípade, že sa nejedná o žiadne z predošlých pravidiel je výsledná afinita `NUMERIC`

V nasledujúcej tabuľke je potom výpis všetkých bežných dátových typov jazyka SQL a ich afinita, ktorá im prislúcha na základe uvedených pravidiel. Podľa pravidiel potom dátový typ „FLOATING POINT“ získa afinitu INTEGER kvôli reťazcu „INT“ na konci slova „POINT“ a dátový typ „STRING“ bude mať afinitu NUMERIC a nie TEXT.

Tab. 3.1: Prevod dátových typov SQL na typy SQLite [4]

Typ určený pri vytváraní tabuľky pomocou príkazu `CREATE TABLE`	**Výsledný typ**	**Použité pravidlo**
`INT, INTEGER, TINYINT, SMALLINT, MEDIUMINT, BIGINT, UNSIGNED BIG INT, INT2, INT8`	`INTEGER`	1
`CHARACTER, VARCHAR(255), VARYING CHARACTER(255), TEXT, CLOB, NCHAR(55), NATIVE CHARACTER(70), NVARCHAR(100)`	`TEXT`	2
`BLOB`	`NONE`	3
`REAL, DOUBLE, DOUBLE PRECISION, FLOAT`	`REAL`	4
`NUMERIC, DECIMAL(10,5), DATE, DATETIME`	`NUMERIC`	5

V prílohách diplomovej práce sa nachádza úplný ER diagram databázy s atribútmi, ich dátovými typmi a cudzími kľúčmi, ktoré prepájajú jednotlivé tabuľky. Obrázok bol získaný z programu Toad Data Modeller verzie 5.1.[7] Dátový typ pre dátum sedenia a dátum a čas narodenia pacienta bol zvolený ako INTEGER. SQLite nemá triedu na ukladanie času a preto sa ukladá v jednom z nasledujúcich dátových typov:[4]

- TEXT - ako reťazce podľa ISO8601 („YYYY-MM-DD HH:MM:SS.SSS“)
- REAL - ukladá juliánsky dátum - počet dní od poludnia na Greenwitch 24. novembra 4714 p.n.l.
- INTEGER - ukladá UNIX Time, počet sekúnd od 1. januára 1970 00:00:00 Koordinovaného svetového času.

Pre prácu bol zvolený dátový typ INTEGER. Tabuľky databázy boli vytvorené v konozolovej aplikácii sqlite3.exe. Query na vytvorenie tabuľky pacient je nasledovné:

```
CREATE TABLE pacient
(
  id INTEGER NOT NULL,
  id_pac INTEGER NOT NULL,
  rodcis INTEGER NOT NULL,
```

```
  jmeno TEXT NOT NULL,
  prijmeni TEXT NOT NULL,
  rod_prijm TEXT,
  dat_dn INTEGER NOT NULL,
  jine_idu TEXT,
  fk_sex INTEGER NOT NULL,
  u TEXT,
  an TEXT,
  dg TEXT,
  z TEXT,
  CONSTRAINT key1 PRIMARY KEY (id),
  CONSTRAINT id_pac UNIQUE (id_pac),
  CONSTRAINT rel_sex FOREIGN KEY (fk_sex) REFERENCES sex (id_sex)
);
```

Premenné s atribútom `NOT NULL` sú povinné a potrebujú byť zadané pri vytváraní nového záznamu do databázy. Premenná „id“ je nastavená ako primárny kľúč tabuľky. „id_pac“ je nastavený ako unikátny a v prípade potreby môže byť použitý ako primárny kľúč. Teda jedná sa o kľúč kandidátny. Premenná „fk_sex“ je cudzím kľúčom do tabuľky obsahujúcej pohlavie.

Celá databáza bola vytvorená vopred a pre použitie v aplikácii sa bude len načítavať zo súboru.

3.2 Funkcionalita aplikácie v Qt

Pretože sa jedná o rozšírenie už vytvorenej aplikácie bol použitý framework Qt, v ktorom bola naprogramovaná pôvodná aplikácia. Aktuálna verzia Qt je v momente písania projektu 5.2. Bola zverejnená 12. Decembra 2013. Na programovanie však bola použitá staršia verzia z dôvodu využívania knižnice na prácu s multimediálnymi dátami - Phonon. Táto knižnica sa nenachádza v štandardnom balíku Qt od verzie 5.0. Konkrétna použitá verzia Qt preto bola 4.8.5 a Qt Creator verzie 2.8.1. Ako kompilátor bol použitý Microsoft Visual C++ Compiler 10.0. Aplikácia prehrávača bola napísaná s českými popismi ovládacích prvkov a mala by byť používaná v českých logopedických ambulanciách. Pre zachovanie jednotnosti sú preto aj na mnou navrhnutých oknách popisky v Českom jazyku.

Po spustení aplikácie sa zobrazí okno s výberom pacienta. V prípade, že s pacientom lekár ešte nemal sedenie, je možné pre nového pacienta vytvoriť záznam v databáze. Tým sa aplikácia presunie na okno Nový pacient, ktoré obsahuje polia na zadanie informácii o pacientovi a zdravotnej poisťovni, pod ktorú pacient patrí. Z úvodného okna je možné tiež vygenerovať XML súbor s inforáciami o pacientovi, jeho sedeniach a nahrávkach. Poslednou možnosťou je spustenie aplikácie v

jednoduchom móde a zadanie údajov o pacientovi až pri ukladaní súboru nahrávky. Kliknutím na niektorú z týchto možností sa aplikácie presunie do svojho hlavného okna.

Hlavné okno aplikácie slúži najmä na nahrávanie nahrávok a ukladanie informácií o nich do databázy. Nahrávky sú priradené jednotlivým pacientom a sedeniam, ktoré pacient absolvoval. V hlavnom okne je tiež možnosť prehrávania nahrávok a zobrazenie časového priebehu, resp. spektrogramu. V hlavnom menu sa dá pod položkou „O programe“ zobraziť operačný systém na ktorom bola aplikácia preložená. Poslednou funkciou hlavného okna je výber nahrávacieho zariadenia, teda mikrofónu z ktorého sa bude nahrávať audio signál.

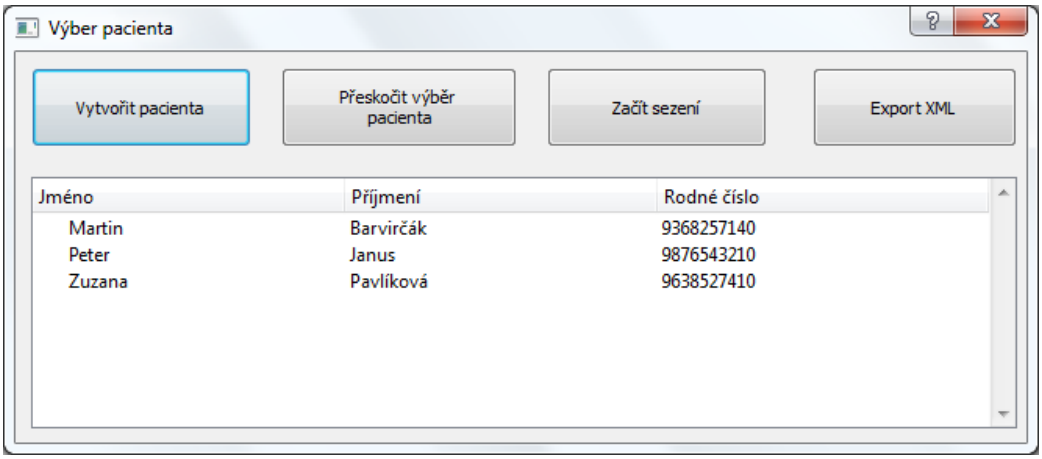
Aplikácia je rozšírena aj o schopnosť vyhľadávania nahrávok v databáze, ich otváranie, prípadne aj ich mazanie. Ku týmto možnostiam sa dá dostat prostredníctvom tlačítka „Vyhladat“. Detailnejší popis práce s aplikáciou a tiež niektorých funkcií, ktoré sa starajú najmä o spracovanie dát a údajov sa nachádza v nasledujúcich kapitolách.

3.3 Okno Výber pacienta

Po preložení a spustení programu sa ako prvé otvorí databáza. Databáza sa otvára pomocou následujúcej funkcie:

```
void Vyber_pacienta::OpenDB()
{
    db = QSqlDatabase::addDatabase("QSQLITE");
    db.setDatabaseName("database/database.db");
    db.open();
}
```

Z databázy sa načítajú dáta o pacientoch, ktoré sa zobrazia v tabuľke úvodného okna. Tabuľka obsahuje tri viditeľné stĺpce a jeden stĺpec, ktorý je skrytý. Ako viditeľné boli zvolené stĺpce obsahujúce meno, priezvisko a rodné číslo pacienta. Skrytý stĺpec tabuľky nesie informácie o „id“ z tabuľky pacient v databáze. Úvodné okno „Výber pacienta“ je zobrazené na nasledujúcom obrázku. Okno obsahuje štyri tlačítka. Tlačítko „Přeskočit výběr pacienta“ spustí hlavné okno prehrávača s tým, že si v globálnej premennej zapamatá, že pacient nebol zvolený. Táto voľba je vhodná na to aby lekár mohol hned pristúpiť k vyšetreniu a nahrávaniu nahrávok bez nutnosti zadávania údajov o pacientovi do databázy. Tým by sa mal šetriť najmä pacientov čas, pretože údaje lekár môže zapísať počas pauzy medzi sedeniami pacientov.



Obr. 3.3: Úvodné okno aplikácie s výberom pacienta

Druhým tlačítkom je „Začít sezení“. Stlačením sa spustí nasledujúca funkcia:

```
void Vyber_pacienta::BeginPac()
{
    QModelIndex index = ui->pacienti->currentIndex();
    int row = index.row();
    if (row == -1)
    {
        QMessageBox::warning(this, "Neplatná operácia", "Nebol ↩
            zvloený pacient!", QMessageBox::Ok);
        return;
    }
    int id = ui->pacienti->model()->data( ui->pacienti->model()->↩
        index(row, 3) ).toInt();
    showMainWindow = new MainWindow();
    showMainWindow->CreateSession(id);
    showMainWindow->show();
    showMainWindow->setAttribute(Qt::WA_DeleteOnClose);
    connect(showMainWindow, SIGNAL(destroyed()), this, SLOT(↩
        ShowWindow()));
    hide();
}
```

Najprv sa overí či bol zo zoznamu vybratý niektorý pacient. V prípade, že sa tak nestalo, program vypíše varovnú správu. Ak bol zvolený niektorý riadok tabuľky, do premennej „id“ sa zapíše hodnota z posledného stĺpca. „Id“ je primárny kľúč v databáze a preto sa pri znalosti jeho hodnoty dájú zistiť všetky ostatné údaje, které prislúchajú k danému pacientovi. Táto hodnota sa následne ako parameter predáva

funkciám hlavného okna. Funkcia tiež zavolá funkcie na vykreslenie hlavného okna a vytvorenie sedenia pre zvoleného pacienta.

Dalšími tlačítkami sú „Export XML“ a „Vytvoření pacienta“. Z názvov je zrejmé, čo tieto tlačítka majú za úlohu a ich detailný popis činnosti a funkcií sa nachádza ďalej v texte.

3.3.1 Generovanie XML súboru

Táto kapitola sa zaoberá funkciami na vytváranie XML súboru, ktorý obsahuje informácie o pacientovi. Súčasťou vygenerovaného XML je tiež zoznam nahrávok a udaje o sedení počas ktorého boli nahrávky vytvorené. Po kliknutí na tlačítko „Export XML“ sa spustí funkcia, ktorá získa „id“ zo zvoleného riadku tabuľky. Pred tým sa ešte overuje či bol označený niektorý z riadkov tabuľky. Pomocou id sa získajú ostatné potrebné údaje, ktoré sa uložia do príslušných premenných. Nasledujúca funkcia ukazuje zíkanie „idPacienta“ z tabuľky pacient v databáze. Získanie ostatných údajov je analogické a preto nie je query vo funkcii úplné. Nakoniec sa zavolá funkcia na vytvorenie samotného súboru XML.

```
void Vyber_pacienta::ExportToXML()
{
    if (QDir("xml").exists() == false)
            QDir().mkdir("xml");
    QString idPacienta = 0;
    QString meno;
    QString priezvisko;
    QModelIndex index = ui->pacienti->currentIndex();
    int row = index.row();
    if (row == -1)
    {
        QMessageBox::warning(this, "Neplatná operácia", "Nebol ↩
            zvolený pacient!", QMessageBox::Ok);
        return;
    }
    int id = ui->pacienti->model()->data( ui->pacienti->model()->↩
        index(row, 3) ).toInt();
    QSqlQuery query(QString("SELECT * FROM pacient WHERE id = '\%1'")↩
        .arg(id));
    while (query.next())
    {
        idPacienta = query.value(0).toString();

    }
```

```
    CreateXMLFile(meno, priezvisko);
```

Súbor XML sa vytvára funkciou „`CreateXMLFile(QString meno, QString priezvisko)`“ ktorej parametre sú meno a priezvisko pacienta:

```
void Vyber_pacienta::CreateXMLFile(QString meno, QString priezvisko)
{
    QDate date = QDate::currentDate();
    QString dateString = date.toString("yyyy-MM-dd");
    QFile file("xml/"+dateString+meno+priezvisko+".xml");
    if (!file.open(QIODevice::WriteOnly))
    {
    QMessageBox::warning(0, "Len na čtení", "Súbor je v módu len na čtení");
    }
    else
    {
        QXmlStreamWriter* xmlWriter = new QXmlStreamWriter();

        xmlWriter->setDevice(&file);

        xmlWriter->writeStartDocument();

        xmlWriter->writeStartElement("pacient");

        QMapIterator<QString,QString> a(id_pac);
    while (a.hasNext())
    {
            a.next();
        xmlWriter->writeStartElement(a.key());
        xmlWriter->writeCharacters (a.value());
        xmlWriter->writeEndElement();
    }
        xmlWriter->writeEndElement();
        xmlWriter->writeEndDocument();
        delete xmlWriter;
    }
}
```

Funkcia ako prvé vytvorí súbor pod stanoveným názvom. V tomto prípade bola za názov zvolená kombinácia údajov o pacientovi a dátum vytvorenia XML súboru. Formát názvu je „rok-mesiac-deň-meno-priezvisko.xml“. Ďalej sa tento súbor zaplní získanými dátami z databázy pomovou funkcie „`QXmlStreamWriter()`“. Pre názornosť je opäť ukázaný len zápis jednej premennej, pretože sa celý zápis vykonáva rovnakým spôsobom. Nasledujúci obrázok potom zobrazuje takto vytvorený súbor

pre pacienta s tromi nahrávkami. Za povšimnutie stojí, že XML ukazuje fakt, ako počas druhého sedenia vznikli nahrávky dve.

```
- <pacient>
    <identifikace>9368257140</identifikace>
    <jmeno>Martin</jmeno>
    <prijmeni>Barvirčák</prijmeni>
    <rod._prijmeni/>
    <titul_pred_jmenem>Ing.</titul_pred_jmenem>
    <titul_za_jmenem>PhD.</titul_za_jmenem>
    <pohlavi>Muž</pohlavi>
    <rodne_cislo>9368257140</rodne_cislo>
    <datum_narozeni>7.8.1993 4:50</datum_narozeni>
    <jine_identifikacni_udaje>Jine ident. údaje</jine_identifikacni_udaje>
    <urgentni>Urgent</urgentni>
    <anamneza>Anamnéza</anamneza>
    <diagnoza>Diagnózy</diagnoza>
    <zprava>Správa</zprava>
    <ulica>Teriakovská</ulica>
    <popisni_cislo>124</popisni_cislo>
    <orientacni_cislo>1045</orientacni_cislo>
    <mesto>Teriakovce</mesto>
    <psc>9953</psc>
    <pojistovna>Dôvera</pojistovna>
    <cislo_pojistovny>3</cislo_pojistovny>
    <ulica>Slovenská</ulica>
    <popisni_cislo>15</popisni_cislo>
    <orientacni_cislo>50</orientacni_cislo>
    <mesto>Prešov</mesto>
    <psc>12465</psc>
    <cislo_sedenia>2</cislo_sedenia>
    <datum_sedenia>24.05.2014, 17:45</datum_sedenia>
    <nazov_nahravky>2014-05-24.17-45-41BarvirčákMartin.wav</nazov_nahravky>
    <cislo_sedenia>2</cislo_sedenia>
    <datum_sedenia>24.05.2014, 17:45</datum_sedenia>
    <nazov_nahravky>2014-05-24.17-45-35BarvirčákMartin.wav</nazov_nahravky>
    <cislo_sedenia>1</cislo_sedenia>
    <datum_sedenia>24.05.2014, 17:44</datum_sedenia>
    <nazov_nahravky>2014-05-24.17-45-15BarvirčákMartin.wav</nazov_nahravky>
  </pacient>
```

Obr. 3.4: Ukážka vygenerovaného XML súboru

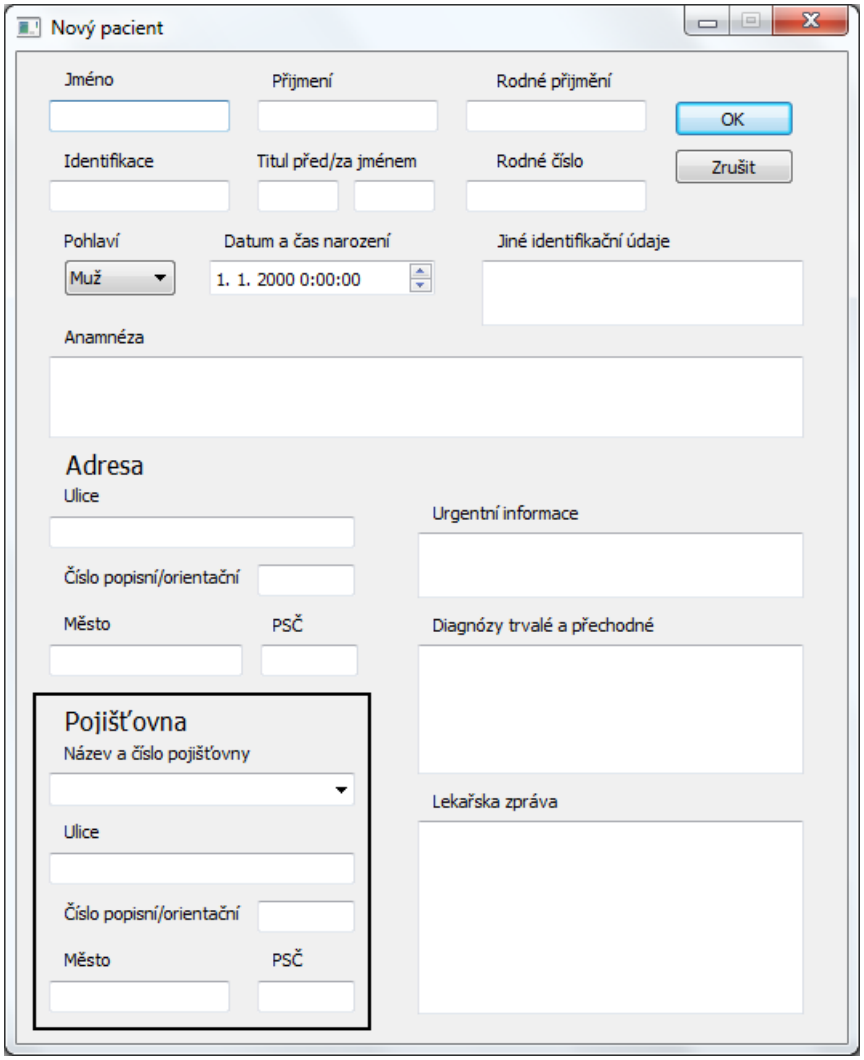
3.4 Okno Nový pacient

Do okna nový pacient sa v aplikácii dá dostať dvomi spôsoblmi. Prvým je kliknutím na „Vytvořit pacienta“ v úvodnom okne. Inak sa toto okno otvorí pri ukladaní nahrávky ak bola na úvodnom okne zvolená možnosť „Přeskočit výběr pacienta“. Okno slúži ako jednoduchý formulár na vkladanie údajov o pacientovi a poisťovni do databázy. Funkcia „`GetData()` sa stará o to, či boli údaje zadané a prípadne spracúva udaje z polí kde sa očakáva ich určitý formát. Príkladom takého poľa je zadávanie popisného a orientačného čísla, kde je medzi číslami ako deliaci znak „/“. Dáta sa do databázy vkladajú pomocou sql query.

```
void New_Pacient::GetData()
{
    Udaje Data;
    bool valid;
    Data.meno = ui->lineEdit->text();
    Data.priezvisko = ui->lineEdit_3->text();
    Data.identifikace = ui->lineEdit_2->text()
    Data.psc = ui->lineEdit_10->text().toInt(&valid);
    if (valid == false)
    {
            QMessageBox::warning(this, "Chyba", QString::fromUtf8("↩
               PSČ musí být číslo."), QMessageBox::Ok);
            return;
    }

    QStringList cpcoParse;

    if (!Data.cpco.isEmpty())
    {
        if (Data.cpco.contains("/"))
        {
            cpcoParse = Data.cpco.split("/");
        }
        else
        {
            QMessageBox::critical(this, "Chyba", QString::fromUtf8("↩
               Číslo popisní/orientační musí obsahovat znak /."), ↩
               QMessageBox::Ok);
            return;
        }
    }
```

```
 QSqlQuery queryIns;
    queryIns.exec(QString("INSERT INTO pacient (id_pac,jmeno, ↩
        prijmeni) VALUES ('\%1' '\%2', '\%3')
    .arg(Data.identifikace).arg(Data.meno).arg(Data.priezvisko));

    close();
}
```

Obrázok ukazuje rozmiestnenie jednotlivých polí v okne formulára. Samotné okno obsahuje dve štandardné tlačítka „OK“ na uloženie dát do databázy a „Zrušit“ pre návrat do úvodného okna programu.



Obr. 3.5: Okno na zadávanie údajov o pacientovi

3.5 Hlavné okno programu

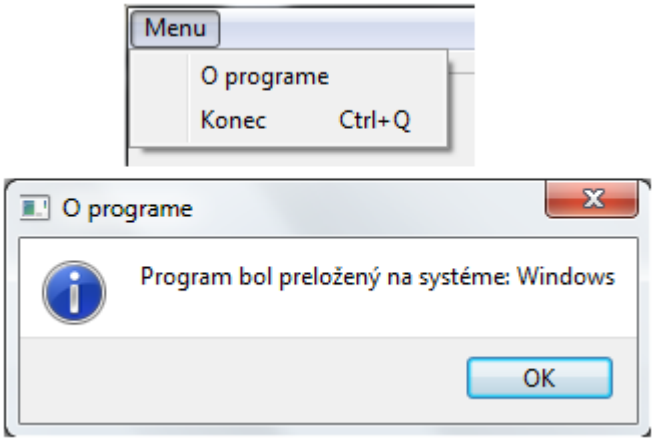
Hlavné okno prehrávača oproti povodnej verzii neobsahuje veľa zmien vo svojom vzhľade. Hlavné menu bolo rozšírené o položku „O programe“. Táto položka po kliknutí zobrazí operačný systém, na ktorom prebehla kompilácia programu. Zistenie operačného systému sa deje za pomoci makier, ktoré boli definované v úvode zdrojového kódu hlavného okna. Ich zápis sa nachádza nasledovnom boxe:

```
#ifdef Q_OS_UNIX
QString OS = "Linux";
#endif
#ifdef Q_OS_WIN32
QString OS = "Windows";
#endif
#ifdef Q_OS_MAC
QString OS = "Mac OS";
#endif
```

O samotný výpis a vykreslenie správy s obsahom aktuálneho operačného systému sa stará funkcia „`About()`“

```
void MainWindow::About()
{
    QMessageBox::information(this, "O programe", QString::fromUtf8("↩
        Program byl přeložený na systému: ")+OS, QMessageBox::Ok);
}
```

Zistenie operačného systému je diplomovej práci zmienené kvoli potenciálu, ktorý táto možnosť prináša. Vďaka týmto makrám je potom možné, prisposobiť funkcie programu pre určitý operačný systém. Tým môže jeden program vykonávať svoju funkciu optimalizovane pre knižnice toho ktorého OS. Príkladom by mohlo byť použitie knižníc DirectX v aplikácii pre Windows a použitie OpenGL pre Linux a MacOS. Ďalšou možnosťou by bolo obmedzenie funkčnosti programu pre niektorý OS. Na obrázku 3.6 sa nachádza vzhľad hlavného menu v GUI aplikácie a vukreslená správa s operačným systémom.



Obr. 3.6: Hlavné menu programu a okno s informáciou o OS

Ďalšou podstatnou zmenou okna prehrávača je rozšírenie o roletové menu (combobox) na zvolenie vstupného zariadenia pre nahrávanie. Na to bolo potrebné deklarovať premennú typu `QAudioDeviceInfo`, do ktorej sa potom vybrané zvukové zariadenie ukladá.

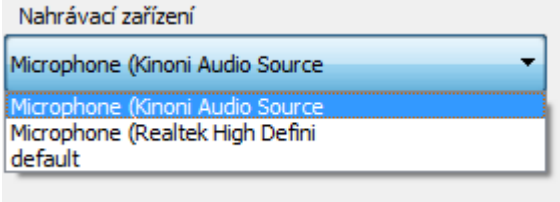
```
QAudioDeviceInfo deviceInfo = QAudioDeviceInfo::defaultInputDevice();
```

Nasledujúcim krokom bolo naplnenie roletového menu dostupnými zariadeniami. Menu sa napĺňa cyklom ktorý sa nachádza vo funkcii „MainWindow“. Daná funkcia obsahuje tiez pripojenie všetkých tlačítok v hlavnom okne a definíciu signálov, ktoré spúšťajú funkcie programu.

```
  foreach (deviceInfo, QAudioDeviceInfo::availableDevices(QAudio::↩
      AudioInput))
    ui->zvukovka->addItem(deviceInfo.deviceName());
```

Poslednou časťou výberu zariadenia je získanie hodnoty vybranej v roletovom menu a nastavenie vybranej zvukovej karty na aktuálne používanú. Nasledujúci kód sa vykonáva vo funkcii „`rec()`“, ktorá spúšťa nahrávanie zvukového sgnálu. Ukážka roletového menu sa nachádza na obrázku.

```
QList<QAudioDeviceInfo> test = QAudioDeviceInfo::availableDevices(↩
    QAudio::AudioInput);
QAudioDeviceInfo info = test.value(ui->zvukovka->currentIndex());
```



Obr. 3.7: Výber nahrávacieho zariadenia

Podstatnými zmenami prešla funkcia na ukladanie súboru „`save()`“. Funkcia kontroluje či bolo spustené nahrávanie. Ak nie, tak funkcia nemá dáta na uloženie a preto sa vráti do hlavného okna bez zmien. Ďalšou kontrolou je zistenie či už dané dáta boli uložené, to zistí podľa obsahu premennej „g_rec“. V prípade, že už uloženie prebehlo, tak sa nahrávka opätovne neukladá. Poslednou kontrolou je zistenie, že v úvodnom okne bol vybratý pacient. Ak nie, program vráti okno na vloženie údajov o pacientovi. Ukladanie sa vykonáva do predvolenej zložky „nahravky“, ktorú funkcia v prípade potreby vytvorí. Funkcia podľa „id“ pacienta získa potrebné údaje z databázy a potom nahrávku uloží pod formátovaným názvom. Názov nahrávky obsahuje údaje o mene a priezvisku pacienta a tiez o čase a dátume kedy nahrávka vznikla. Po uložení sa záznam o vzniku nahrávky zapíše do databázy, kde je uvedený názov nahrávky a nahrávka sa priradí k pacientovmu sedeniu. Posledným krokom je nastavenie parametru „g_rec“, aby sa zabránilo viacnásobnému ukladaniu rovnakej nahrávky.

```
void MainWindow::save()
{
    if (g_rec == 0 || g_rec == 2)
        return;
    if (g_selected == 0)
    {
        NewPac();
        return;
    }
    if (QDir("nahravky").exists() == false)
            QDir().mkdir("nahravky");
    QString meno = "";
    QString priezvisko = "";
    QSqlQuery query(QString("SELECT jmeno, prijmeni FROM pacient ↩
       WHERE id = '\%1'").arg(g_id));
    if (query.next())
    {
        meno = query.value(0).toString();
```

```
        priezvisko = query.value(1).toString();
    }
    QDateTime dateTime = QDateTime::currentDateTime();
    QString date = dateTime.toString("yyyy-MM-dd.H-m-s");
    QString fileName = "nahravky/"+date+priezvisko+meno+".wav";
    qDebug() << fileName;
    if (!fileName.isEmpty()) {
        m_file = new WavPcmFile(fileName, audio->format(), this);
        qDebug() << "OK";
        if(!m_file->open()) {
           // Error
            qDebug() << "Nepodarilo se otervrit soubor";
        }

        qDebug() << "opened";
        m_file->write(m_buffer.data(), m_buffer_pos);
        m_file->close();
        QSqlQuery queryIns;
        queryIns.exec(QString("INSERT INTO nahravka (nazov, fk_sed) ↩
           VALUES ('\%1', '\%2')")
                .arg(date+priezvisko+meno+".wav").arg(g_sed));
    }
    g_rec = 2;
}
```

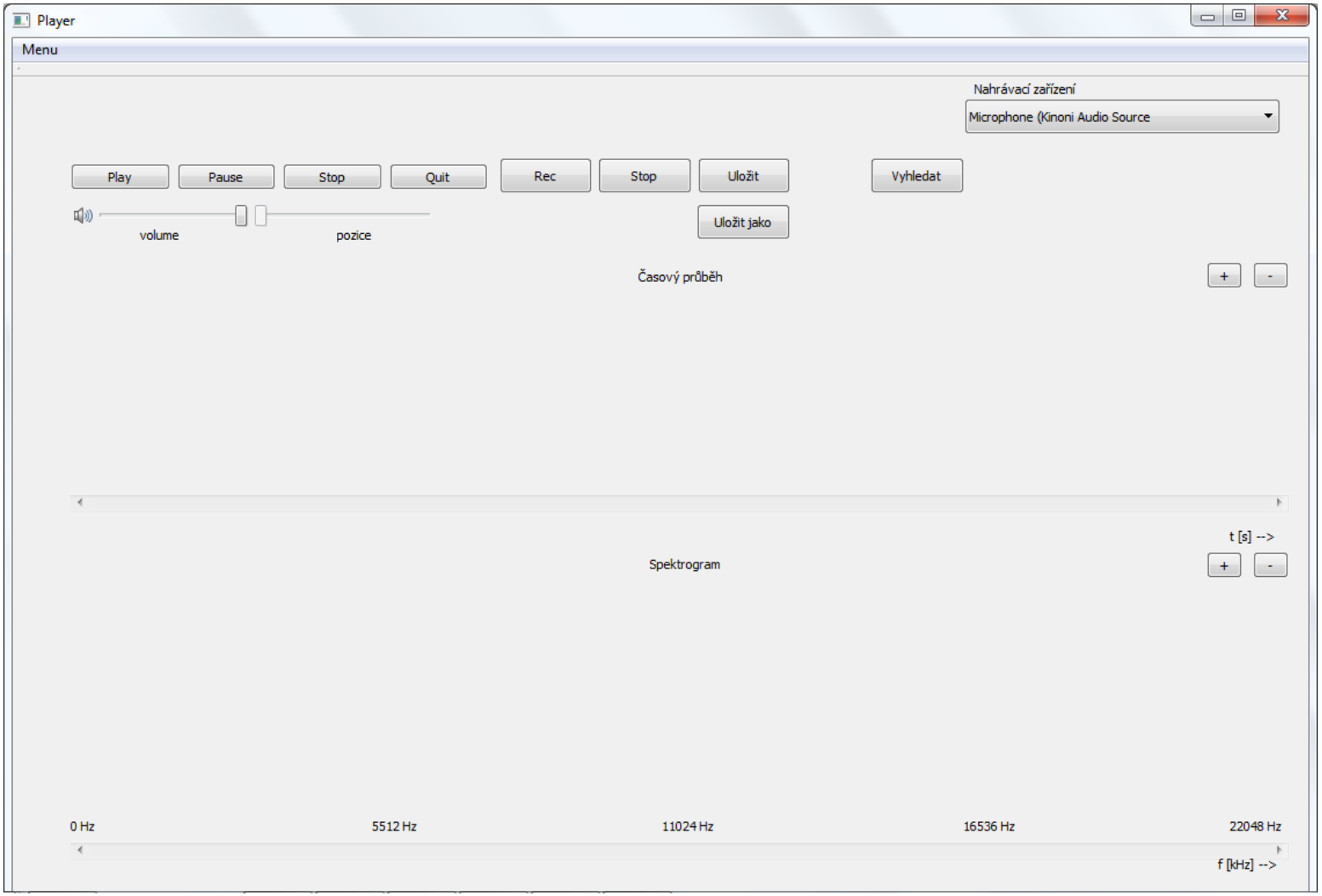
Funkciu pôvodného tlačítka „Uložit“ získalo tlačítko „Uložit jako“. V tomto prípade sa jedná o export nahrávky, kde si užívateľ sám vyberie lokáciu, na ktorú sa má kópia nahrávky uložiť. Nahrávka sa pritom ukladá aj na predom stanovené miesto ako pri stlačení tlačítka „Uložit“. Týmto sa zachová záznam o prebehnutí nahrávania v databáze a pritom sa umožní ukladanie nahrávky napríklad na prenosný USB kľúč. Rešenie týmto spôsobom bolo zvolené preto, aby správne fungovala funkcia vyhľadávania a otvárania nahrávok, o ktorých existuje v databáze záznam.

Funkcia „`SaveAs()`“ kontroluje len to, či boli predom nahrané zvukové dáta. Nasleduje zobrazenie štandardného dialógového okna, v ktorom sa vyberá umiestnenie súboru a vypíše jeho názov. Zadanie názvu súboru je nevyhnutné a vo funkcii sa kontroluje. Nakoniec sa zavolá funkcia „`save()`“, ktorá už bola vysvetlená vyššie.

```
void MainWindow::SaveAs()
{
    if (g_rec == 0)
        return;
    QFileDialog::Options options;
    QString selectedFilter;
```

```
    QString fileName = QFileDialog::getSaveFileName(this, tr("↩
        QFileDialog::getSaveFileName()"),
                    tr("C:\\"), tr("WAV Files (*.wav);;All Files (*)"), ↩
                        &selectedFilter, options);

    qDebug() << fileName;
    if (!fileName.isEmpty()) {
        m_file = new WavPcmFile(fileName, audio->format(), this);
        qDebug() << "OK";
        if(!m_file->open()) {
            // Error
            qDebug() << "Nepodarilo se otervrit soubor";
        }

        qDebug() << "opened";
        m_file->write(m_buffer.data(), m_buffer_pos);
        m_file->close();
        save();
    }
}
```

Nasledujúci obrázok zobrazuje grafické rozhranie hlavného okna prehrávača spusteného v systéme Windows 7. Hlavné okno zobrazuje tiež časový priebeh nahrávky a spektrogram, ktorý ukazuje frekvenčnú charakteristiku danej nahrávky.



Obr. 3.8: Hlavné okno prehrávača

3.6 Vyhľadávanie nahrávok

Veľkou zmenou bolo rozšírenie prehrávača o možnosť vyhľadávania v nahrávkach. Do okna vyhľadavania sa dá dostať kliknutím na tlačítko „Hledat“ v hlavnom okne prehrávača. Po kliknutí sa zobrazí dialógové okno s možnosťou hľadania podľa priezviska alebo rodného čísla pacienta. Je možné vyhľadať všetky záznamy pacienta pred alebo po stanovenom dátume. Tým je možné pomocou zadania dátumu zobrazované výsledky efektívnejšie filtrovať. Okno obsahuje tabuľku na vypísanie výsledkov hľadania. Tabuľka má päť stĺpcov, z ktorých je opäť z rovnakých dôvodov skrytý stĺpec „ID“. Okrem neho má tabuľka stĺpce „Nahrávka“ (s názvom nahrávky), „Pacient“ (kombinácia priezviska a mena pacienta), „Datum“ (dátum a čas sedenia) a „Sezení č.“ (poradové číslo sedenia).

V prvom kroku funkcie na vyhľadávanie „`FindData()`“ sa prevedie cas z okna vyhľadávania na UnixTime. Tento krok je potrebný kvoli tomu, že dátum sedenia je tiež v tvare UnixTime a porovnávajú sa potom dve celé čísla. V ďalšom kroku sa vyprázdni tabuľka v okne. To sa vykonáva kvôli tomu aby sa tabuľka nezpapĺňala dátami z predchadzajúcich vyhľadavaní. Potom následuje získanie zadaného priezvika a rodného čísla, ktoré sa porovnávajú s údajmi v databáze. Pomocou query sa z databázy vyberie „id“ ako primárny kľúč k tabuľke sedenie. Pomocou dátumu a toho v ktorej pozícii bol zaškrtnuý „radio button“ sa potom porovnáva zadaný dátum s dátumami sedenia. Podľa získaných sedení sa zistí, ktoré nahrávky zodpovedajú filtru. Tieto dáta sa potom vypíšu do tabuľky, kde je možné s nimi pracovať.

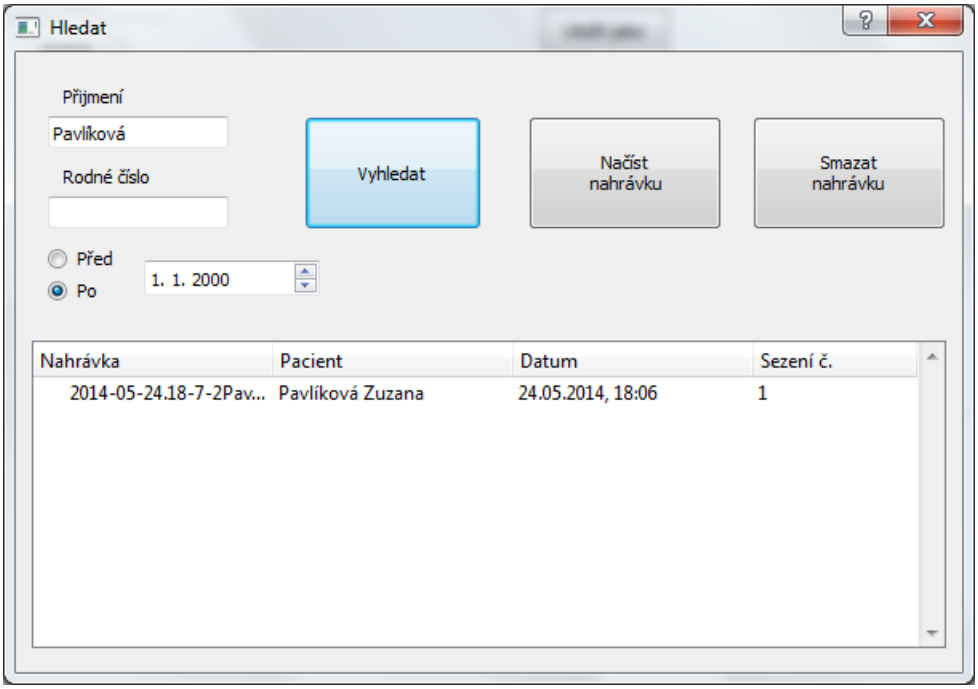
```
void Search::FindData()
{
    QDateTime date = QDateTime::fromString(ui->datum->text(), "d.M.↩
        yyyy");
    uint unixT = date.toTime_t();

    ui->nahravky->clear();
    int id = 0;
    QString meno = "";
    QString priezvisko = "";

    QString Query_s;
    Query_s = "SELECT id, jmeno, prijmeni FROM pacient WHERE prijmeni↩
        = '\%1' OR rodcis = '\%2' ";
    QString comp;
    QSqlQuery query(QString(Query_s).arg(ui->prijm->text()).arg(ui->↩
        rodc->text()));
    if (query.next())
    {
```

```
        id = query.value(0).toInt();
        meno = query.value(1).toString();
        priezvisko = query.value(2).toString();
    }
    if (ui->pred->isChecked())
        comp = "<";
    else if (ui->po->isChecked())
        comp = ">";

    Query_s = "SELECT id_sed, poradie, datum_sed FROM sedenie WHERE ↩
        fk_pacien = '\%1' AND datum_sed "+comp+" "+QString::number(↩
        unixT)+"";
    QSqlQuery query2(QString(Query_s).arg(id));
    while (query2.next())
    {
        QSqlQuery query3(QString("SELECT nazov, id_nahr FROM nahravka↩
            WHERE fk_sed = '\%1' ").arg(query2.value(0).toString()))↩
            ;
        while (query3.next())
        {
            QDateTime datum = QDateTime::fromTime_t(query2.value(2).↩
                toInt());
            QTreeWidgetItem *tempItem = new QTreeWidgetItem();
            tempItem->setText(0,query3.value(0).toString());
            tempItem->setText(1,priezvisko+" "+meno);
            tempItem->setText(2,datum.toString("dd.MM.yyyy, hh:mm"));
            tempItem->setText(3,query2.value(1).toString());
            tempItem->setText(4,query3.value(1).toString());
            ui->nahravky->addTopLevelItem(tempItem);
        }
    }
}
```

Po označení určitého riadku tabuľky je možné nahrávku otvoriť alebo vymazať. Funkcia na otvorenie nahrávky zistí názov nahrávky a v hlavnom okne nahrávku otvorí a prehrá. Je možné sa v nahrávke posúvať a zobrazovat časový priebeh a spektrogram zvukového signálu. Posledným tačítkom je „Smazatt nahrávku“. Tlačítko spustí funkciu „`DeleteData()`“, ktorá vymaže aktuálne označenú nahrávku z disku počítača a záznam o nahrávke z databázy. Následne znova prekreslí tabuľku. Ako prvé sa vymaže záznam z databázy a až potom fyzický súbor na disku. Toto poradie bolo zvolené kvôli tomu, aby nastaním chyby pri mazaní súboru neostal v databáze neplatný záznam.

```
void Search::DeleteData()
{
    QModelIndex index = ui->nahravky->currentIndex();
    int row = index.row();
    QString nahravka = ui->nahravky->model()->data( ui->nahravky->↩
       model()->index(row, 4) ).toString();
    qDebug() << nahravka;
    QString filename = ui->nahravky->model()->data( ui->nahravky->↩
       model()->index(row, 0) ).toString();
    QSqlQuery queryDel;
    queryDel.exec(QString("DELETE FROM nahravka WHERE id_nahr = '\%1'↩
        ").arg(nahravka));
    QFile::remove("nahravky/"+filename);
    ui->nahravky->clear();
    FindData();
}
```

Obrázok ukazuje okno vyhľadávania s jedným nájdeným záznamom, ktorý prináleží hľadanému pacientovi. Z obrázka je vidieť rozdiel v čase začatia sedenia (18:06)a čase uloženia nahrávky (18:07:02).



Obr. 3.9: Okno vyhľadávania nahrávok

4 ZÁVER

Diplomová práca sa zaoberá problematikou používateľských rozhraní pri návrhu multiplatformného prehrávača. Hlavný dôraz bol kladený na rozšírenie prehrávača o databázu nahrávok a údajov prislúchajúcich k nim. Výsledkom projektu je táto textová správa a projekty priložené v prílohe. V textovej správe boli spísané poznatky ohľadom výberu vhodného systému na správu databázy. Porovnaním viacerých systémov bolo zistené, že pre potreby programu sa najlepšie hodí databáza SQLite. SQLite obsahuje priamu podporu v prostredí Qt a preto je práca s dotazmi query jednoduchšia.

Výsledkom diplomovej práce je okrem textu tiež audio prehrávač so schopnosťou nahrávania. Tento prehrávač pracuje s databázou SQLite, do ktorej ukladá informácie o nahrávkach. Ukladané informácie boli vybrané na základe dokumentu „DATOVÝ STANDARD MZ ČR verze 3.01.01 a výše“ vypracovaného Ministerstvom zdravotníctva Českej republiky. Prehrávač je zostavený z viacerých dialógových okien a hlavného okna prehrávača. Práca s prehrávačom by mala byť intuitívna a popisky ovládacích prvkov sú v českom jazyku.

V úvodnom okne je možnosť vybrať už vytvoreného pacienta zo zoznamu pacientov v databáze alebo vytvoriť záznam o pacientovi novom. Ďalej je možné tento krok preskočiť a údaje o pacientovi zapísať až po začatí sedenia pri ukladaní nahrávok. Poslednou možnosťou úvodného okna je vytvorenie XML súboru o zvolenom pacientovi. XML súbor obsahuje všetky údaje o pacientovi v databáze a zoznam nahrávok, ktoré boli s pacientom počas sedení uložené. V hlavnom okne prehrávač funguje zvyčajným spôsobom. Rozšírením je vyhľadávanie v nahrávkach a možnosť výberu vstupného nahrávacieho zariadenia.

V práci sú detailne popísané funkcie, ktoré sa starajú o beh programu. Všetky okná prehrávača sú tiež v jednotlivých kapitolách zobrazené pre operačný systém Windows 7. Vzhľad okien v OS Linux sa nachádza v prílohe B.

LITERATÚRA

[1] Blanchette, J.; Summerfield, M.: *C++ GUI Programming with Qt 4*. Prentice Hall, druhé vydání, Február 2008.

[2] Ezust, A.; Ezust, P.: *An Introduction to Design Patterns in C++ with Qt*. Prentice Hall, druhé vydání, September 2011.

[3] FindTheBest.com, Inc.: Database management systems comparison [online]. Webpage, December 2013, dostupné na: http://database-management-systems.findthebest.com.

[4] Hipp, R. D.: SQLite Documentation [online]. Webpage, December 2013, dostupné na: http://www.sqlite.org/docs.html.

[5] Kreibich, J. A.: *Using SQLite*. O'Reilly Media, Inc., první vydání, August 2010.

[6] Molkentin, D.: *The Book of Qt 4 - The Art of Building Qt Applications*. 555 De Haro Street, San Francisco, CA 94107: No Starch Press, Inc., první vydání, 2007.

[7] Quest Software, Inc.: *Toad Data Modeler 3.5 - User Guide*. Marec 2010.

[8] Summerfield, M.: *Advanced Qt Programming: Creating Great Software with C++ and Qt 4*. Prentice Hall, první vydání, Júl 2010.

[9] Zendulka, J.; Rudolfová, I.: *Databázové systémy IDS - Studijní opora*. Brno: FIT VUT v Brně, 2005.

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

GTK+ The GIMP Toolkit

GUI grafické používateľské rozhranie – Graphical User Interface

SQL štruktúrovaný dopytovací jazyk – Structured Query Language

GPL GNU základná verejná licencia – GNU General Public License

LGPL GNU nižšia základná verejná licencia – GNU Lesser General Public License

GCC set GNU kompilátorov – GNU Compiler Collection

RDBMS systém správy relačnej databázy – relational database management system

API programovacie rozhranie aplikácie – application programming interface

VFS virtuálny súborový systém – virtual file system

RAM pamäť s náhodným prístupom – random-access memory

XML rozšíriteľný značkovací jazyk – extensible markup language

CAD počítačom podporovaný návrh – computer-aided design

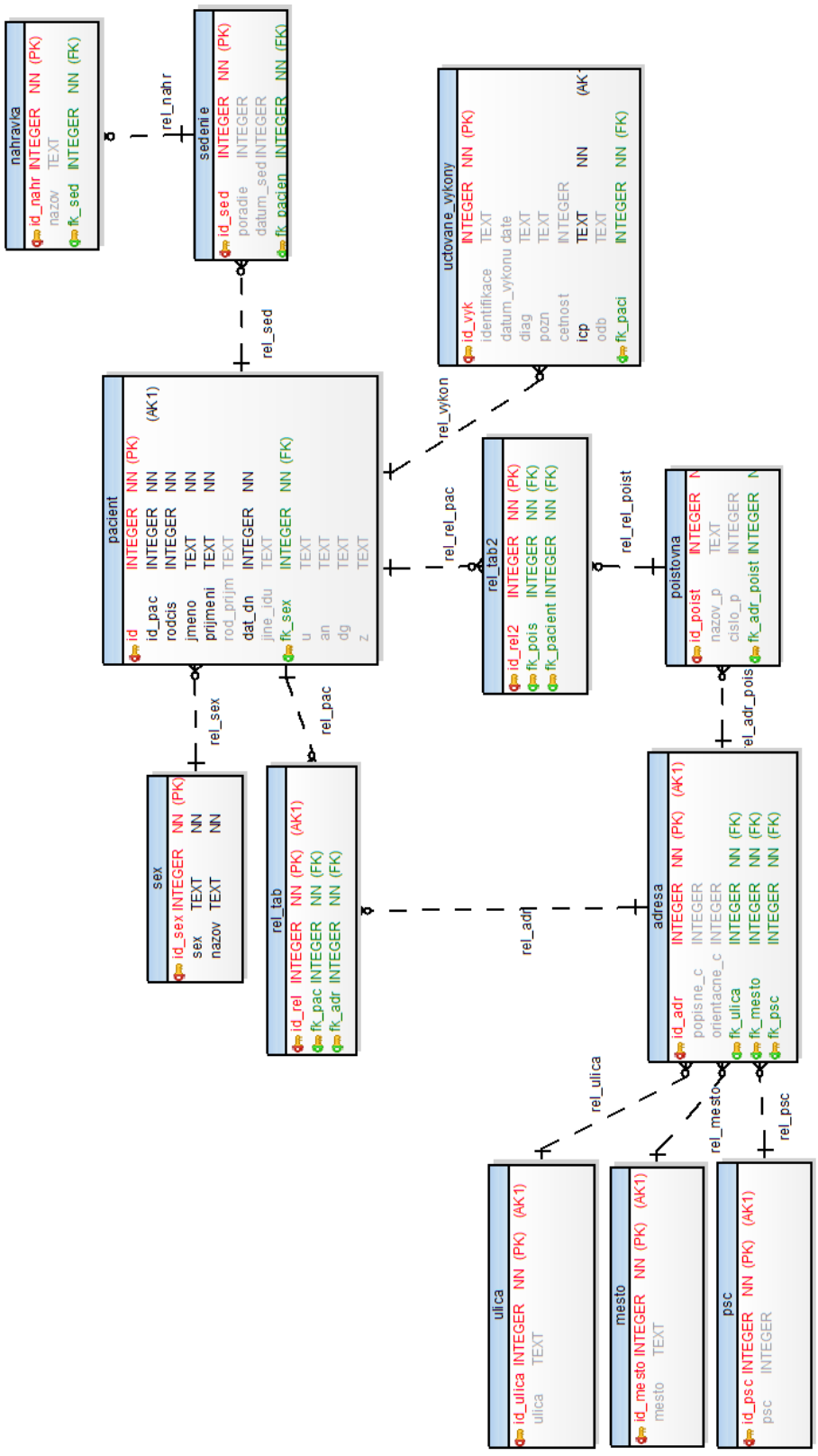
SSD mechanika s nepohyblivým médiom – solid state drive

CSV hodnoty oddelené čiarkami – comma-separated values

ER entitne-vzťahový – entity-relationship

ZOZNAM PRÍLOH

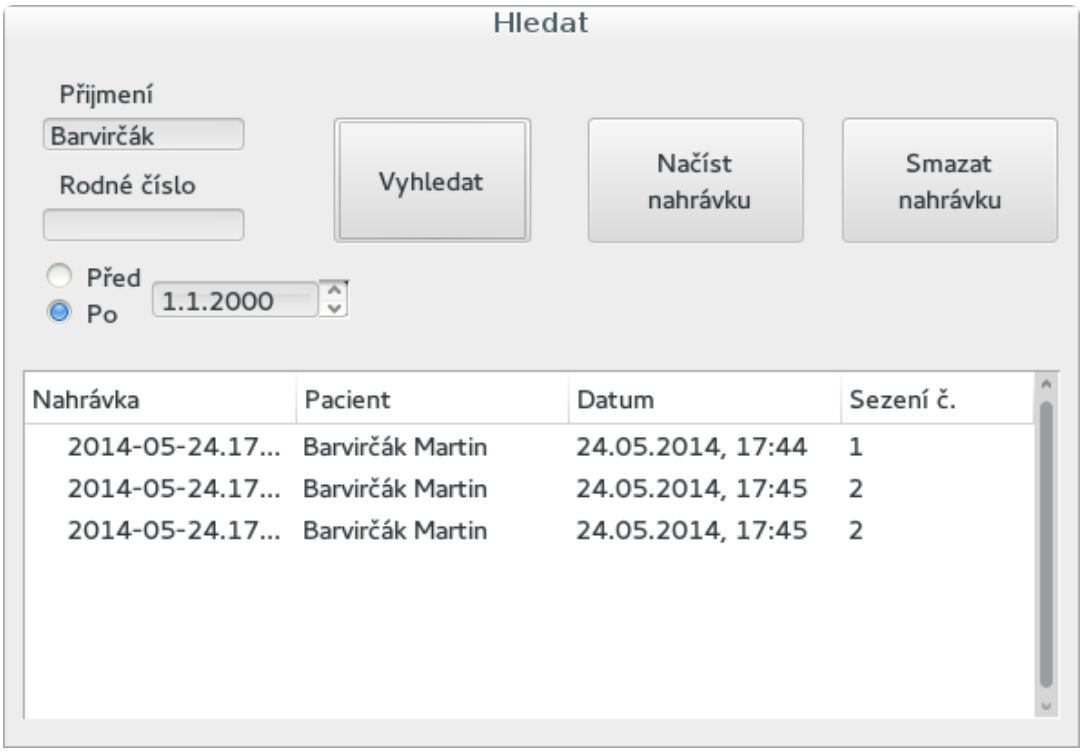
A ER DIAGRAM DATABÁZY



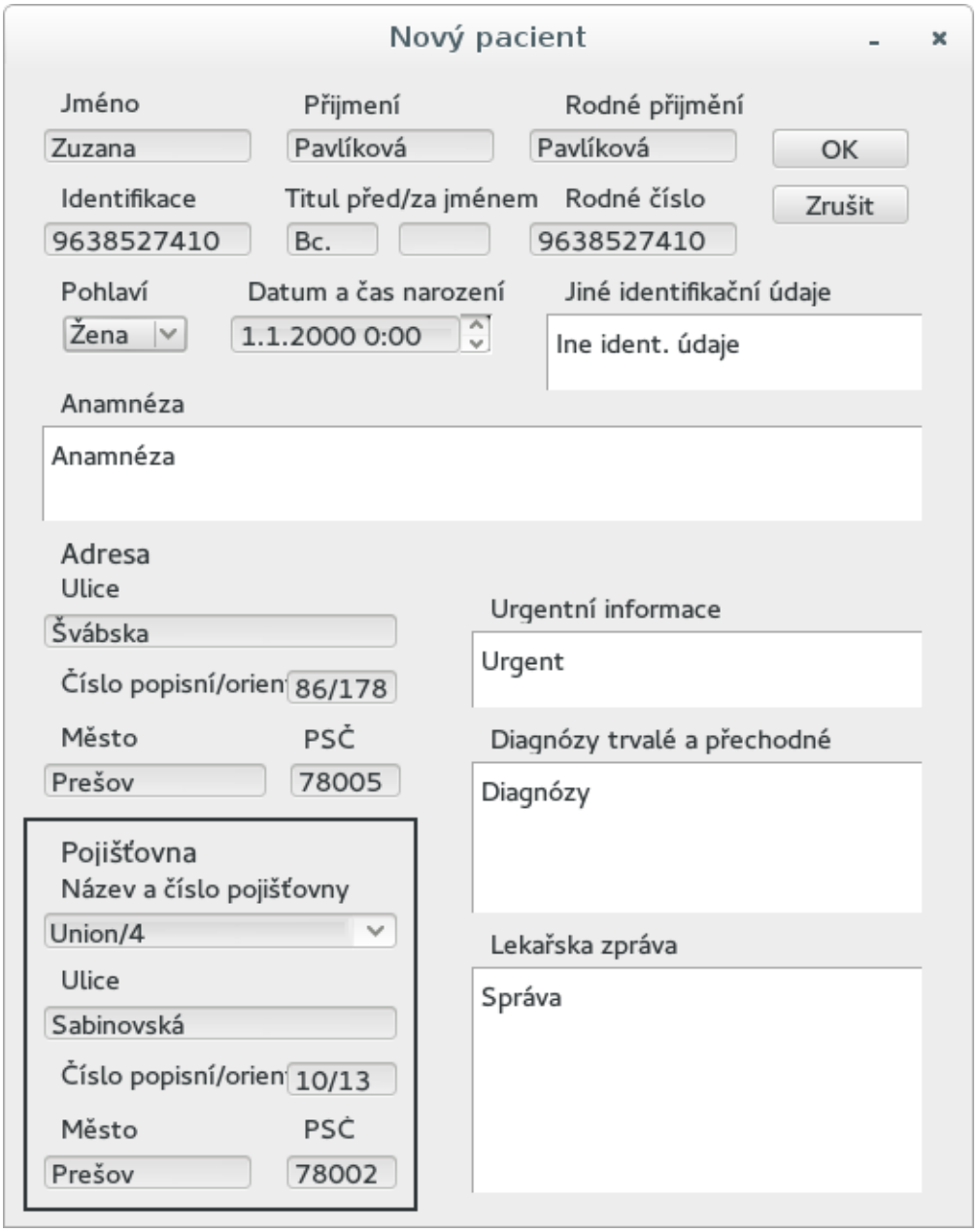
Obr. A.1: Úplný ER diagram databázy s atribútmi a ich dátovými typmi

B UKÁŽKY NAVRHNUTÝCH OKIEN V OS LINUX

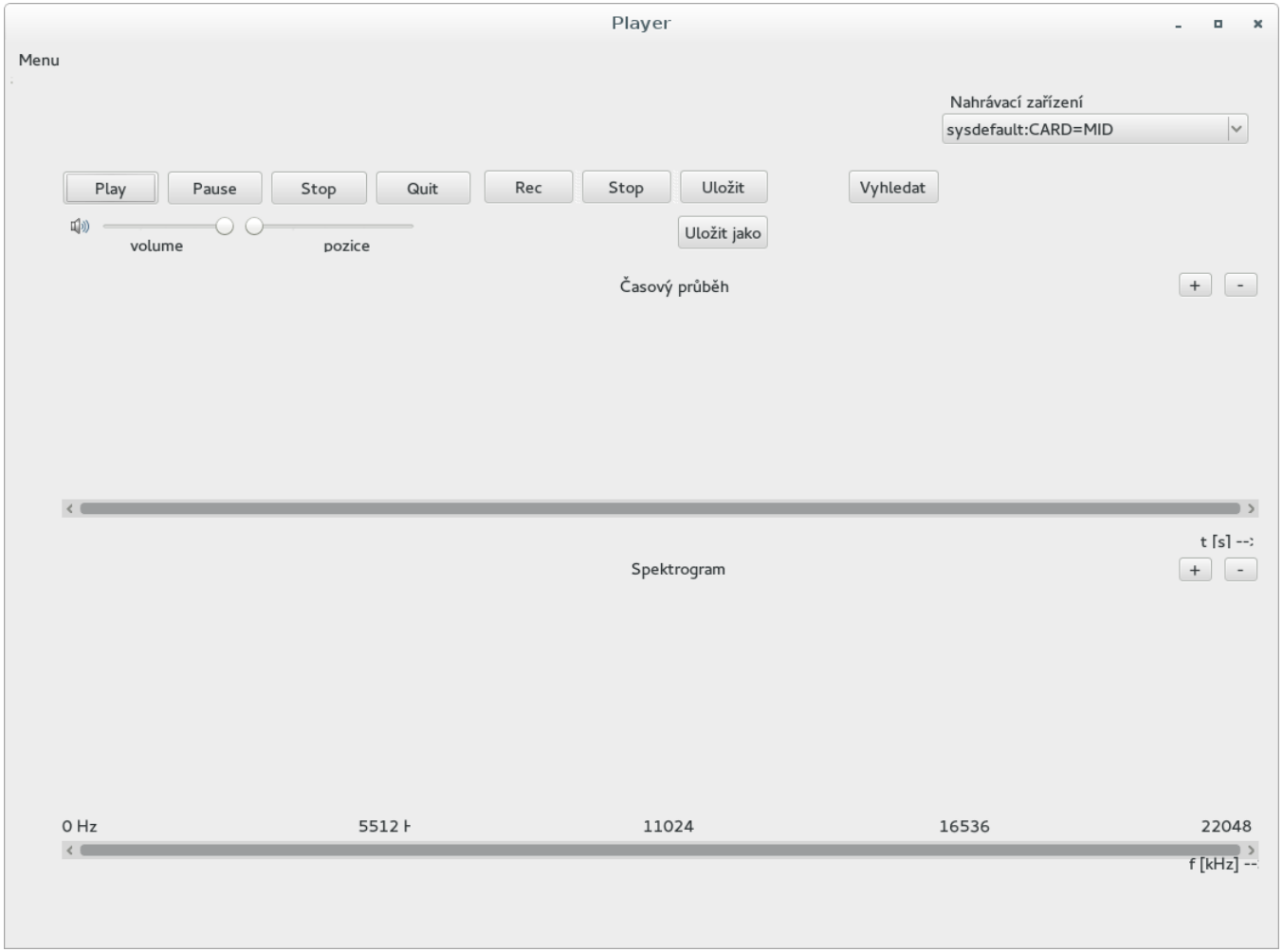
Obr. B.1: Okno výberu pacienta v OS Linux



Obr. B.2: Okno na vyhľadávanie nahrávky



Obr. B.3: Okno na zadávanie údajov nového pacienta v OS Linux



Obr. B.4: Hlavné okno prehrávača v systéme Linux

C OBSAH PRILOŽENÉHO CD

- DLLs – knižnice pre preklad v móde Release
- Empty DB – prázdna databáza
- Install
  - QT 4.8.5
  - QT Creator 2.8.1
  - Debugger a Nástroje pre Windows 7 x64
  - Debugger a Nástroje pre Windows 7 x86
- LaTex – LaTeXsúbory diplomovej práce
- Preložené projekty pre OS Windows 7
- QT projekty – projekty zdrojové kódy pre OS Windows a OS Linux
- Diplomová práca.pdf
- Read-me.txt