Mendelova univerzita v Brně
Provozně ekonomická fakulta

Vzdělávací aplikace pro Cyrilometodějské gymnázium v Prostějově

Bakalářská práce

Vedoucí práce:
Ing. Jaromír Landa, Ph.D.

Tomáš Kohoutek

Brno 2016

Rád bych poděkoval svému vedoucímu Ing. Jaromíru Landovi Ph.D. za odborné konzultace při realizaci této práce, dále Mgr. Martině Malečkové za spolupráci ze strany Cyrilometodějského gymnázia a studentům, kteří se zúčastnili testování.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Vzdělávací aplikace pro Cyrilometodějské gymnázium v Prostějově**
vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

Brno 20.05.2016 .....

Abstract

Kohoutek, T. Educational aplication for The Grammar School of Saints Cyril and Methodius in Prostějov. Bachelor thesis. In Brno: Mendel University, 2016. This thesis deals with the creation of an educational application developed for The Grammar School of Saints Cyril and Methodius in Prostějov. Application is made to test knowledge of students about german-speaking countries.The application is developed in C++ with support of the Qt Library and works with SQLite database.

Keywords

Educational application, The Grammar School of Saints Cyril and Methodius, german language, educational tool, C++, Qt, SQLite

Abstrakt

Kohoutek, T. Vzdělávací aplikace pro Cyrilometodějské gymnázium v Prostějově. Bakalářská práce. Brno: Mendelova univerzita v Brně, 2016. Tato bakalářská práce se zabývá tvorbou vzdělávací aplikace do německého jazyka pro Cyrilometodějské gymnázium v Prostějově. Aplikace slouží k testování znalostí o německy mluvících zemích a je vytvořena v jazyce C++ s podporou knihovny Qt. Pracuje s SQLite databází.

Klíčová slova

Vzdělávací aplikace, Cyrilometodějské gymnázium, němčina, vzdělávací pomůcka, C++, Qt, SQLite

Obsah

Seznam obrázků

1 Úvod a cíl práce

1.1 Úvod

V současné době jdou metody vyučování na školách stále dopředu a moderní technologie umožňují učinit výuku stále více interaktivní. Využívání prezentací je již v dnešní době na středních školách běžnou praxí. Školy poskytují materiály na internetu a žáci pracují s počítači každý den. Moderní technologie jsou zkrátka využívány v mnoha odvětvích a školství je také jedním z nich.

Jedním ze současných trendů jsou aplikace zaměřené na mobilní zařízení. Můžeme naleznout mobilní aplikaci téměř na vše a ani vzdělávací nejsou výjimkou. Od nejmenších dětí, které už používají tablety k naučení se barev a zvířat, až po dospělé, kteří se zdokonalují v logice, matematice či věcných znalostech. Jednou z vyhledávaných oblastí vzdělání jsou jazyky. Znalost jazyků je v dnešní době obrovskou výhodou a tvorba takto zaměřené aplikace se stala cílem této bakalářské práce. Nejrozšířenějším světovým jazykem je angličtina. Bez angličtiny se v dnešní době už téměř neobejdeme, ale pokud se zaměříme na Evropu, je potřeba si uvědomit, že německy mluvících zemí je hned několik. Německo a Rakousko, které zastupují procentuálně největší podíl německy mluvících občanů, jsou země sousedící s Českou Republikou. Obě jsou to vyspělé evropské země, které nabízí spoustu pracovních příležitostí. Je běžnou praxí, že české firmy spolupracují s německými a rakouskými firmami. Učit se německý jazyk je tedy velmi lukrativní volbou a při hledání práce je znalost němčiny čím dál výhodnějším faktorem. Při studiu jazyků lze nalézt pomůcek k angličtině velkou spoustu, od gramatických výukových aplikací a webů až po různé jazykové hry. Zkrátka v angličtině je trh s výukovými materiály poměrně zaplněn. U německého jazyka to tak není. Ačkoliv maturita z německého jazyka je zde běžnou záležitostí, tak při hledání německých vzdělávacích aplikací jich nenalezneme až tolik. A aplikace procvičující znalosti o německy mluvících zemích, které jsou nezbytně nutné ke složení maturitní zkoušky, je natolik specifická, že ji na trhu s aplikacemi nenalezneme.

Na základě těchto podnětů vznikl záměr této bakalářské práce a byl formulován její cíl.

1.2 Cíl práce

Cílem této práce je vytvořit vzdělávací aplikaci pro studenty Cyrilometodějského gymnázia v Prostějově. Aplikace bude sloužit k testování znalostí o německy mluvících zemích a bude využita jako výukový materiál pro poslední dva ročníky, zejména pro studenty skládající maturitní zkoušku z německého jazyka.

Aplikace bude vyvíjena v jazyce C++ s využitím podpory knihovny Qt a dále bude pracovat s databází otázek.

2 Teoretická východiska

V této kapitole jsou popsány technologie, které byly použity v této práci. Volba programovacího jazyka a prostředí, využití objektového programování a navrhnutí diagramu tříd a práce s databází.

2.1 Programovací jazyk C++

Programovací jazyk C++ vychází z jazyka C, který vznikl v 70. letech 20. století. Mezi další jazyky založené na principu C je také Java, Perl nebo PHP. Ačkoliv je C++ odvozeno z C, není s ním úplně kompatibilní. C++ má definovaná přísná pravidla pro přetypování, využívá objektového programování, ale také generické a procedurální. V současné době patří mezi nejrozšířenější a byl zvolen i pro vývoj této aplikace. Právě díky využití principů objektového programování a podpoře na mnoha platformách bude aplikace snadno upravitelná a případně rozšiřitelná.(Solter, Gregoire, Kleper, 2011)

2.2 Podpora knihovny Qt

Pro tuto práci bude vhodné využít podporu knihovny Qt. Qt je populární multiplatformní knihovna umožňující vytváření programů s grafickým uživatelským rozhraním. Její vznik se datuje k roku 1999 a tato knihovna podporuje řadu jazyků. Kromě zvoleného C++ dále například Python nebo Javu. Dále obsahuje podporu SQL, práci s vlákny, přístup k souborům a podobně. Knihovna se dále zaměřuje na mobilní platformy, což je klíčovým faktorem této práce. Poskytnutí free licence umožňuje každému programátorovi takto vyvíjet, což je i důvodem rozsáhlé komunity uživatelů a tedy kvalitní dokumentace s množstvím řešených problémů na internetu.(Blanchette, Summerfield, 2008)

2.3 Vývojové prostředí Qt Creator

QtCreator je multiplatformní vývojové prostředí, které umožňuje vyvíjet na desktop i mobilní platformy. Na internetu je k dispozici free licence, ale při tvorbě je nutné sdílet zdrojové kódy. Zakoupení komerční verze aplikace toto omezení samozřejmě nemá. Vývojové prostředí poskytuje editor kódu pro C++ a QML s našeptáváním a barevným rozlišením klíčových slov. Podporuje verzování a přepínání jednotlivých platforem lze provést snadným kliknutím na obrazovce. Prostředí Qt Designeru slouží pro vytváření grafického rozhraní jak desktopových verzí, tak pro mobilní zařízení.(Qt Documentation, 2015)

2.3.1 Android – připojení ke Qt

Android je mobilní operační systém založený na Linuxu, který je volně dostupný jako open source. Je určený pro mobilní zařízení zejména chytré telefony, ale dnes

už také tablety a hodinky. První zmínka o Androidu se datuje k roku 2003, ale největší rozvoj nastává v roce 2005 při odkoupení společností Google. Primárně se na Android vyvíjí v programovacím jazyce Java, ale v dnešní době lze využít už i jiné jazyky.(Kolomazník, 2015)

Abychom mohli vyvíjet na platformu Android je potřeba udělat několik úprav s vývojovým prostředím Qt Creatoru. První požadavek je nástroj Android SDK (Software development kit), nebo-li sada vývojových nástrojů. Druhým požadavkem je Android NDK, tedy sada pro tvorbu nativních aplikací na platformě Android. Na obě tyto sady je přehledný návod na dokumentaci Qt.(Qt Documentation, 2015)

2.3.2 iOS – připojení ke Qt

iOS je mobilní operační systém firmy Apple. Na rozdíl od Androidu není open source a tedy vyvíjet na tuto platformu je potřeba být součástí vývojového programu. Operační systém byl původně určen především pro chytré telefony, ale trendem jsou také tablety a hodinky. První generace chytrých telefonů iPhone a iPad, takzvaná první generace, vyšla v roce 2010.(Apple, 2015)

Na tuto platformu se vyvíjí v jazyce Ojective-C a dále modernější verze Swift. Jazyk také vychází ze zmíněného jazyka C. K vývoji potřebujeme pracovat v prostředí XCode. Aplikaci lze najít v Mac App Store, takže pro vývoj na tuto platformu musíme pracovat na zařízení, které má také operační systém Apple, tedy OS X. V případě, že bychom chtěli vyvíjet pouze na iPhone a iPad simulátory, pak je dosavadní postup dostačující, nicméně při testování aplikace na reálném zařízení je nutné se zapojit do iOS Developer Programu a vygenerovat si developerské certifikáty. Návod nalezneme taktéž na dokumentaci Qt.(Qt Documentation, 2015)

2.4 SQLite databáze

SQLite je relační databáze, založená na tabulkách. Každá tabulka obsahuje sloupečky, které značí jednotlivé atributy a řádky značí její záznamy. Atributy mají běžně stanovené datové typy jako číslo, textový řetězec, datum a podobně. SQLite databáze typovaná není, tedy není to nutné, přesto je vhodné typování používat. Každý záznam by měl mít svůj unikátní identifikátor, který se zpravidla stanoví jako ID a slouží a je určen jako primární klíč.(Martinek, 2016)

Ke zmíněnému spojení relační databáze. Relační vychází z anglického *relation* což značí vztah, nebo vazbu. To znázorňuje jednotlivé spojení tabulek. Hlavní výhodou databáze je RDBMS, anglická zkratka pro *Relation DataBase Management System*, česky přeloženo jako Systém řízení báze dat. Tento systém se stal postupem času velice optimalizovaným a odladěným nástrojem, který za uživatele řeší spoustu věcí jako například přístup více uživatelů. Uživatel s databází komunikuje pomocí SQL příkazů. SQL z anglického překladu *Structured Query Language* a jedná se o standardizovaný strukturovaný dotazovací jazyk.(Martinek, 2016)

Vlastnosti RDBMS se shrnují zkratkou ACID, která také z anglického překladu zastávají následující vlastnosti:

- Nedělitelnost – pokud operace selže, vrátí se databáze do původního stavu
- Validita – Stav databáze po dokončení transakce je konzistentní
- Izolace – Operace se vzájemně neovlivňují, pokud jsou volány zároveň, tak se zařadí dle pořadí do fronty
- Trvanlivost – Změny zapsání dat se provedou okamžitě

Jak jsou data uložena, si uživatel nemusí dělat starosti, jelikož databáze řeší všechno za něj. O databázi se často hovoří jako o třetí vrstvě aplikace, přičemž první je vrstva uživatelského rozhraní a druhá logika aplikace.(Martinek, 2016)

2.5 Jazyk QML

QML je jednoduchý deklarativní vysokoúrovňový skriptovací jazyk určený jak k popisu vzhledu samotné aplikace, tak i jejího chování.Soubor tvoří stromová struktura objektů, jejich vlastností a případně definovaných funkcí. Jazyk QML je založený na principech JavaScriptu a obsahuje moduly využívající XML (Extensible Markup Language) a nahrazuje tak vytváření grafického rozhraní pomocí widgetů. Důležité je si v souboru naimportovat elementy Qt Quick knihovny, v tomto případě QtQuick 2.3, QtQuick.Controls 1.2, QtQuick.Dialogs 1.1 a další. V souboru se dále nachází jednotlivé objekty s danou syntaxí. Každý objekt má název a svoji definici.(Řezník, 2012)

Pro názornou ukázku je zobrazen příklad přímo z práce:

```
ApplicationWindow {
    id: applicationWindow
    visible: true
    width: Screen.width
    height: Screen.height
    color: "blue"
Label {
        id: nazevAplikace
        width: 320
        height: 166
        text: qsTr("Název aplikace")
      }
. . .
}
```

V příkladu jsou vidět dva objekty s daným typem a sice ApplicationWindow a Label. Za složenými závorkami se nachází jednotlivé atributy. Můžeme si všimnout, že za složenou závorkou prvního objektu se nachází jiný objekt. Tento vztah lze nazvat rodič-potomek a v QML je do značné míry využíván a tvoří tak již zmíněnou stromovou strukturu.

Za složenou závorkou jsou zobrazeny jednotlivé vlastnosti objektů nebo také atributy. Každá vlastnost je popsána názvem, dále dvojtečkou, která značí takzvané binding, což je způsob jak deklarovat obsah vlastnosti. Všechny objekty by měly mít jako první atribut id, přes které se na ně dále přistupuje. Id každého objektu musí být unikátní a po prvotní deklaraci je dále nezměnitelné.

2.5.1 Funkcionalita

Jazyk QML slouží jak k definování designu, tak k obsluze funkcionality.

```
Button {
        id: spustitTest
        text: qsTr("Spustit test")
                function spusteni() { //žsloitá definice funkce
                        }
        onClicked: {
                    spusteni();
                }
```

V upraveném příkladu můžeme vidět tlačítko, které slouží ke spuštění testu. Jelikož při spouštění testu záleží na několika proměnných jak ho správně vytvořit, které prvky zobrazit a které schovat a další. Byla by implementace pouze ve vlastnosti onClicked složitá a zbytečná. Využijeme tedy další možnosti objektu a tou je funkce.

Funkci definujeme v jazyce QML snadno jako “nazevFunkce()” a následně ji definujeme. Funkce může přejímat parametry, v aplikaci je toho využito zejména číselnými a textovými. V těle funkce lze využívat podmínky if a switch, což v běžném těle například onClicked definovat nelze. V případě, že logika kliknutí na některé tlačítko není tak složitá, aby si vyžadovala vytvoření samostatné funkce, lze definovat podmínka pouze pomocí ternárního operátoru s danou syntaxí:

```
podmínka ? výraz1 : výraz2;
```

Princip tohoto ternárního operátoru je takový:

1. Vyhodnocení podmínky
2. Když je podmínka nenulová (true), vyhodnotí se první výraz
3. Když je podmínka nulová (false), vyhodnotí se druhý výraz

Jednotlivé výrazy jsou odděleny dvojtečkou a celá podmínka ukončena středníkem.

2.6 QML – Responzivní design

Další vlastností vhodnou ke zmínění je vytváření designu pro různé velikosti obrazovek, tedy responzivního designu. Velikost jednotlivých objektů lze definovat absolutně, tedy dosadit konkrétní čísla. Nicméně do vlastnosti výška a šířka aplikačního

okna lze také nastavit přímo šířka a výška zařízení, na které je aplikace právě přeložena. Dále v kódu, kde pracujeme s pozicemi jednotlivých objektů, často využíváme toto relativní zobrazení:

```
Screen.width * konstanta
```

Kde do proměnné *konstanta* jsou dosazena různá desetinná čísla. Tím je docíleno relativně stejného zobrazení velikosti na různě velkých obrazovkách.

Další relativní zobrazení se týká pozice prvků. Zde je samozřejmě možnost absolutního vyjádření, ale také relativního pomocí kotev (*anchor*). Pozice daného objektu je pak vždy určena v rámci nadřazeného, jelikož elementy tvoří hierarchii. Znázornění je například:

```
anchors.verticalCenterOffset : -150
anchors.horizontalCenterOffset : 0
anchors.verticalCenter : parent.verticalCenter
anchors.horizontalCenter : parent.horizontalCenter
```

Poslední znázorněním responzivního designu je pracování s rozlišením obrazovky. Na začátku aplikace nastaví výšku a šířku dle rozlišení aktuální obrazovky a dále všechny prvky pracují už pouze s tímto číslem. Tedy na následujícím příkladu je vyobrazeno umístění tlačítka podle relativní výšky aplikačního okna:

```
ApplicationWindow {
    id: applicationWindow
    width: Screen.width * 0.5
    height: Screen.height *0.9

        Button {
        id: button1
        anchors.verticalCenterOffset: applicationWindow.height * 0.1
        anchors.horizontalCenterOffset: 0
        }
        . . .
    }
```

2.7 UML – diagram tříd

UML (Unified Modeling Language) je jazyk sloužící k návrhu aplikace, který podporuje objektově orientovaný přístup. Využívá grafické interpretace pro snazší porozumění problému. S pomocí patřičného nástroje zaznamenává myšlenky pomocí kreslení konceptu návrhu. Je vhodné namodelovat návrh před tím, než se začne psát kód. Předností výsledných nákresů je jejich srozumitelnost. Jazykem UML lze vytvořit mnoho diagramů jako diagram tříd, aktivit, stavový diagram, sekvenční a mnoho dalších. Pro vytvářenou aplikaci bude použit diagram tříd. Je kladen důraz na analýzu problému, aby se při samotném programování myšlenky pouze opisovali. Diagram tříd patří do skupiny diagramů struktur a zobrazuje pohled na systém jako na typy objektů, obsah tříd a vztahy mezi nimi.(Čápka, 2013) Do diagramu patří prvky:

- Třídy
- Datové typy
- Vazby a násobnosti
- a další...

Třída má v horním oddílu název a pod ním seznam atributů a čarou odděleny metody. Mezi jednotlivými třídami jsou znázorněny vztahy relačními vazbami. Diagram poskytuje vizuální náhled o tom, jak aplikace bude fungovat a slouží pro jednoduchou orientaci. Grafická podoba je mnohdy bližší než text a tedy snazší pro pochopení logiky systému. UML diagram nezobrazuje všechno, ale postihne to důležité a vytvoří abstraktní pohled. Pro tvorbu tohoto diagramu využijeme nástroj Visual Paradigm, který Ústav informatiky na PEF poskytuje zdarma. Další výhodou tohoto programu je možnost vygenerování kostry kódu. To ušetří práci jako například vytváření hlavičkových a implementačních souborů.

3 Analýza současných trendů

3.1 Trendy ve školství

V následujících pěti letech by se měli elektronické pomůcky, jako jsou dataprojektory, interaktivní tabule, tablety a podobně, stát nedílnou součástí všech typů škol. Sami učitelé vidí výhody se zavedením. Interaktivní výuka žáky baví a výsledky jsou poznat okamžitě. Další ze zmiňovaných výhod je například zapojení nemocných žáků do výuky.(Djakoualno, 2012)

Moderní technologie ve výuce pomáhají v mnoha předmětech. Například virtuální simulace a trojrozměrné modely. Pro výukové účely se běžně znázorňují modely funkcí motorů, chemické reakce, principy matematických funkcí a mnoho dalších.(Tondlová, 2015)

Digitalizace učebnic je dalším z diskutovaných témat. Z několika učebnic by rázem stačilo nosit tablet poskytnutý školou. Digitalizované učebnice mají obsahovat učební hry, které zároveň rozvíjí znalosti studenta.(Tondlová, 2015)

Moderní technologie ovlivňují dnešní děti. S dřívějším používáním počítačů a mobilních zařízení ho dříve přestanou chápat jako prostředek zábavy a odpočinku, ale začnou vnímat jeho možnosti pro objevování a vytváření znalostí. Tento princip povede ke konkurenční výhodě na trhu práce, ale také na poli moderní vědy.(Filová, 2013)

Digitalizace výuky s sebou nenese pouze pozitivní vlivy. Jedním z negativních vlivů je například zhoršení dovedností u psaní a čtení. Tento následek je jistě nutné brát v potaz, nicméně je na každé škole, jakou dokáže zvolit střední cestu.(Filová, 2013)

3.2 Trendy ve vývoji

Tato sekce analyzuje současné trendy a přístupy v oblasti vývoje vzdělávacích jazykových aplikací. Motivace pro zavedení aplikací do interaktivní výuky v dnešní době je značná.

Dnes, kdy už se tablety a mobilní zařízení na trhu nějakou dobu vyskytují, je pochopitelně jejich cena stále nižší. To zapříčiňuje jejich větší odběr a tím i nové příležitosti nejen v oblasti byznysu, ale také například právě ve vzdělávání. Nastupující generace je již od útlého věku zvyklá pracovat s moderními technologiemi a je pro ně přirozené tato zařízení používat pro nejrůznější účely.(BusinessIT.cz, 2013)

Analýzy podle společnosti Gartner, které ve svém článku uvádí Čuchna (2014) tvrdí, že množství prodaných tabletů již dohnalo množství prodávaných stolních počítačů. Počet prodaných chytrých telefonů předčil stolní počítače alespoň čtyřnásobně. Do budoucna se očekává další růst. Je tedy otázkou blízké budoucnosti, kdy bude běžné zařadit mobilní telefony a tablety do přímé výuky na středních školách.(BusinessIT.cz, 2014)

Aplikace obecně rozšiřují způsoby vzdělání, učiní výuku interaktivnější, zajímavou a intenzivní. V momentě kdy se podaří získat pozornost studentů, stanou se motivovanějšími a dosahují lepších výsledků. V dnešní době je škála nabízených aplikací tak široká, že poskytuje téměř neomezené množství možností vzdělání.

V současné době se na trhu vyskytuje mnoho aplikací, které podporují výuku cizích jazyků. Jednou z nich je například Duolingo. „Duolingo funguje tak trochu jako počítačová hra, kde plníte krátké úkoly, sbíráte body, přicházíte o životy a krok za krokem postupujete dopředu. Na začátku si nastavíte svoji denní dávku, otestujete svoje znalosti a začnete."(Nádoba, 2014)

„Apple ji loni vybral jako „iPhone App of the Year" a v Americe o ní vycházejí články, slibující on-line revoluci ve vzdělávání. V Google Play skončila loni mezi „Best of the Best". Je zadarmo a po světě si ji stáhlo už skoro 50 milionů lidí."(Nádoba, 2014)

Můžeme tedy vidět, že v dnešní moderní době je výuka pomocí aplikací vítaná a vývoj v této oblasti pokračuje. Duolingo samozřejmě není jedinou aplikací na vzdělávání, ale lze na ni vhodně analyzovat postupy. Prvním faktorem úspěchu, je nebrat uživateli příliš mnoho času. Aplikace umožňuje uživateli nastavit přesné množství času které ji chce věnovat. Uživatel si nastaví určitý časový interval a pokud je i po skončení stále zaujat, jednoduše začne nové kolo. Krátké časové intervaly umožňují uživateli od aplikace odejít téměř kdykoliv a přitom za sebou nenechat nedokončenou lekci. Aplikace pokrývá úkoly v oblastech psaní, překládání, tvorby vět, poslechu i mluvení. Tedy všechny standartní postupy využívané ve školství pro výuku cizího jazyka.

Snahu zaujmout spotřebitele formou hry projevil také Goethe Institut. Aplikace kterou poskytují na Google Play s názvem „Dobrodružství s němčinou" je také pojata formou hry. Uživatel prožívá příběh hrdiny a formou audiovizuálních prvků se zdokonaluje v jazyce. Goethe Institut na svém webu popisuje více svých aplikací založených na herním principu.(Goethe Institut, 2015)

Dalším příkladem tentokrát z tuzemského prostředí je projekt Prectime.

„Vystupujeme z klasického vzdělávacího systému a snažíme se ukázat, že při výuce jazyků existuje i jiná cesta. Podcasty, návody, aplikace, žádná cesta biče a cukru, žádná teorie na dva roky. Deset až patnáct minut denně změní váš pohled a způsob práce. Naučíte se pracovat méně, avšak více efektivně, získáte mnohem více volného času a získáte velký nadhled." popisuje aplikaci její autor Jan Hovad.(Pošmura, 2015)

Můžeme tedy pozorovat obdobný přístup kratšího testování pro udržení pozornosti uživatele. Základem celého interaktivního učení je tedy motivace a snaha zaujmout studenty. V práci bude využito principů získaných na základě těchto informací a testy ve vyvíjené práci budou kratšího rozsahu s hezkým grafickým podkladem.

4 Metodika

Prvním krokem k vytvoření aplikace na míru je zjistit požadavky zadavatele. Takže před samotným zahájením návrhu aplikace proběhne konzultace, kde se stanoví, jak by měla aplikace vypadat. Tyto požadavky budou popsány v kapitole o zákazníkovi, kde bude také cílová škola stručně přiblížena.

Po sepsání požadavků proběhne návrh aplikace. Dále bude vytvořen diagram tříd v programu Visual Paradigm, ke kterému Ústav informatiky poskytuje zdarma licenci. Tento diagram vytváří vizuální přehled o aplikační logice a tím usnadňuje následnou implementaci. Zvolený program Visual Paradigm také umožňuje vygenerovat hrubé podklady souborů tříd, což ušetří čas vývoje. Aplikace bude obsahovat sadu otázek pro testování a bude použita práce s databází. Před vytvořením databáze proběhne vytvoření schéma databáze v programu Oracle SQL Developer Data Modeler. Podle schématu vznikne databáze v programu SQLite Studio a bude naplněna testovací sadou otázek, která je v souladu se zadanými požadavky. Následuje implementace. Aplikace bude naprogramována v jazyce C++ s podporou knihovny Qt. Jako vývojové prostředí poslouží Qt Creator, který je v případě open-source programování zdarma. Aplikace bude programována objektově a v průběhu implementace konzultována s cílovou školou.

Po vytvoření funkčního prototypu proběhne testování na Cyrilometodějském gymnáziu a zjištění nedostatků. Tento krok zajistí, že vývoj probíhá podle zadaných požadavků a udrží zákazníka v obraze. Zjištěné chyby budou odstraněny a vznikne výsledná verze aplikace. K výsledné verzi vznikne manuál na ovládání aplikace a bude spolu s ní poskytnut cílové škole.

5 Analýza zákazníka

Tato bakalářská práce vznikla na motiv vytvoření vzdělávací aplikace, která bude dále poskytnuta studentům střední školy. Škola projevila zájem o aplikaci vytvořenou na míru do hodin německého jazyka. Jedná se o Cyrilometodějské gymnázium v Prostějově a jednání probíhalo s vedoucí jazykových sekcí Mgr. Martinou Malečkovou.

5.1 Cyrilometodějské gymnázium v Prostějově

Jedná se o církevní osmileté gymnázium v Prostějově, které zde působí již od roku 1992. Škola sídlí na ulici Komenského 17 a v současné době rozšiřuje svoje působení rekonstrukcí vedlejší budovy. Mnoho inovačních činností, které škola pořádá, je podporováno fondy Evropské unie. Díky této iniciativě je škola technicky vybavena moderními tabulemi, dataprojektory a počítačovou učebnou. Mezi další z probíhajících projektů patří také zařazení tabletů do výuky. Využívání informačních technologií k výuce je předmětem zejména jazykových předmětů a proto škola přivítala možnost této aplikace.(Cyrilometodějské gymnázium v Prostějově, 2015)

5.2 Požadavky

Před koncem školního roku 2014/2015 bylo prokonzultováno s již zmíněnou vedoucí jazykových sekcí, co škola od takové aplikace požaduje a očekává a na základě tohoto rozhovoru vznikly následující požadavky:

- Adekvátní obtížnost
- Snadná manipulace
- Bodové ohodnocení
- Pomůcka k přípravě na maturitu

Obtížnost aplikace by měla odpovídat obtížnosti běžných testů probíhajících na škole. Tedy otázky by měly být vytvořeny dle materiálů poskytnutých školou a v případě rozmyšlení učitele, že je otázka příliš jednoduchá, by měla existovat možnost ji editovat. Studenti by neměli mít problém aplikaci ovládat, tedy její grafické prostředí by nemělo být složité. V dnešní době kdy každý mladý student používá chytrý telefon s aplikacemi, bude tedy aplikace přizpůsobena současným trendům. Dále by testy měly být podobné těm na škole, tedy forma testu nesmí být studentům neznámá. S jakýmkoliv testem v aplikaci by se student neměl setkat poprvé až při spuštění. Po odeslání testu se student dozví své skóre, aby věděl, jak si stojí, kolik udělal chyb a mohl si srovnat svoje výsledky s předešlými pokusy. Především by však testování mělo dávat smysl a poskytovat tak studentům nejen posledního ročníku průběžnou přípravu právě na závěrečnou maturitní zkoušku.

Pro vytvoření databáze otázek byly využity materiály poskytované studentům maturitního ročníku k přípravě. Aplikace se tedy velmi nesoustřeďuje na gramatické znalosti jako na reálie. Reálie jsou nutné znalosti o německy mluvících zemích ke složení státní zkoušky. Aplikace je tedy rozdělena na testy z Německa a z Rakouska. Každá země umožní studentovi psát 3 druhy testů a sice:

- ABCD test
- Pravda/Nepravda
- Doplňování vět

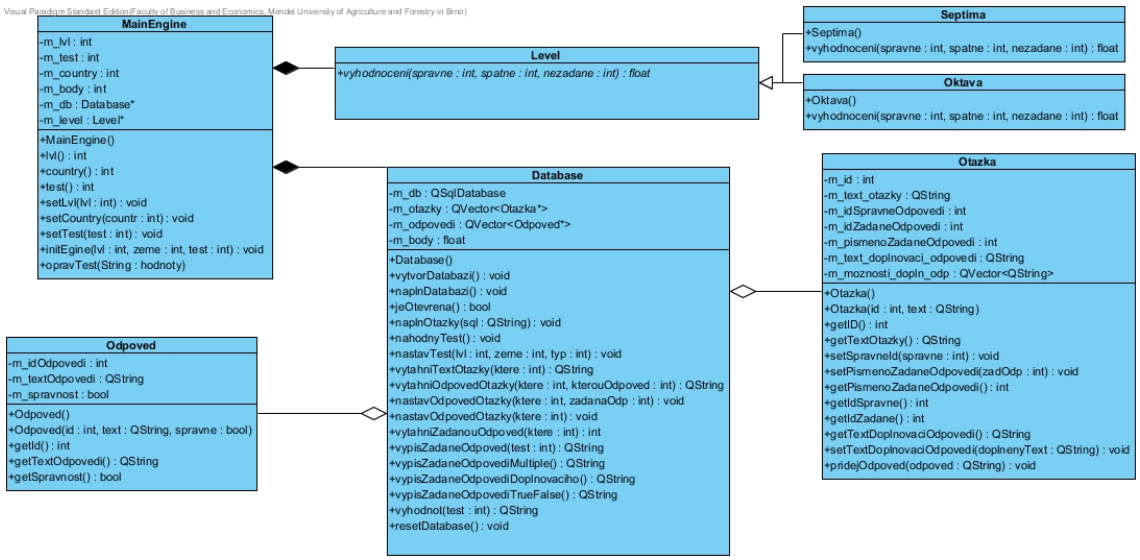
Po konzultaci bylo rozhodnuto, že tyto tři typy testů jsou dostačující svou různorodostí a obtížností. Při vyplňování správných odpovědí je samozřejmě kladen důraz také na gramatickou správnost, dodržování velkých písmen, správnost členů a podobně.

Jelikož seminář z jazyků nepostihuje pouze maturitní ročník (Oktávu), ale také předposlední (Septimu), umožňuje aplikace studentům vybrat, ve které třídě se nachází a to ovlivní hodnocení testu. Výsledkem této práce je tedy aplikace umožňující vytvoření až 6 různých testů s 2 možnostmi hodnocení výsledků. Aplikace je vytvářena jako desktopová, ale především také na mobilní zařízení.

6 Vzdělávací aplikace pro Cyrilometodějské gymnázium

V této kapitole je popsán diagram tříd výsledné aplikace, schéma databáze a popis jednotlivých tříd.

6.1 Diagram tříd aplikace



Obrázek 1: UML Diagram

Na obrázku můžeme vidět 7 propojených tříd. Hlavní a nejdůležitější třídou je MainEngine. Tato třída obstarává aplikační logiku, vytváření správných instancí a podobně. V jejím atributu *m_db* si drží pomocí kompoziční vazby instanci druhé třídy a sice Database třídy. Tato třída slouží ke korektnímu nahrání databáze do aplikace a její správě. Má definovanou spoustu metod, pomocí kterých nastavuje testy, načítá otázky, vypisuje zvolené odpovědi a tak dále. Poslední třídou, kterou má *MainEngine* v atributu na místě *m_level* je abstraktní třída *Level*. V tomto případě se jedná o příklad návrhového vzoru Abstraktní továrna(Valkovic, 2015). Instanci samotné třídy Level není možné vytvořit, nicméně do zmíněného atributu se dosadí instance jednoho z potomků této rodičovské třídy, a sice *Oktava* nebo *Septima*. Třídy jsou pojmenované podle ročníků na škole, pro kterou je tato aplikace vytvářena. Obě dědí metodu vyhodnocující test, nicméně každá ze tříd si s vyhodnocením testu a přidělením výsledného skóre poradí podle svého vzorce. Na významnou třídu spravující Databázi se pojí agregační vazbou další dvě třídy – *Otazka* a *Odpoved*. Podle názvů je zřejmé, že třída *Otazka* nese instance jednotlivých otázek. Má atributy pro

ID a text otázky, ale také například která odpověď je správná, či kterou odpověď zvolil uživatel. Nachází se v ní větší množství metod, protože aplikace poskytuje 3 druhy testů a každý má trošku jiné požadavky opravení. Testy pro možnosti abcd a nebo pravda/nepravda si mohou pamatovat pouze ID zadané odpovědi pro kontrolu správnosti, nicméně test pro doplňovací odpovědi si už musí pamatovat textový řetězec. Poslední třídou je třída *Odpoved*, která je jednoduchou třídou obsahující instance jednotlivých odpovědí v databázi. Na této třídě netřeba hledat žádné složitosti. Drží si pouze atributy ID odpovědi a její text. Logika správných odpovědí a kontroly se řeší v jiných třídách. Poslední třídou, která však není vyobrazena na tomto UML diagramu je statická třída *Settings*. Na tuto třídu nemá žádná ze zmíněných tříd vazbu, a proto není vyobrazena zde. Nicméně s ní pracujeme v Qt Creatoru v souboru *main.qml*, který se stará o grafické rozhraní aplikace, ale také funkce jednotlivých tlačítek a podobně. V těchto případech je mnohdy vhodné si držet v samostatné třídě důležité informace, jako například při procházení testu pořadí otázky, nebo velikost generovaných testů. V případě, že se zákazník rozhodne změnit velikost testu, stačí jediná změna právě v těchto statických údajích.(Čápka, 2012)

6.2 Databáze aplikace

Zvolení databáze pro tuto aplikaci bylo vhodné hned z několika důvodů, ale zejména kvůli bezpečnosti. Množství dat pro takovou aplikaci není až tak velké a jistě by šlo zpracovat i v podobě textových souborů. Nicméně se jedná o školní aplikaci, která může být použitá i k bodovanému hodnocení studentů Cyrilometodějského gymnázia a tedy uchovávání správných odpovědí v textových řetězcích je nebezpečné.

Databázi lze do aplikace dostat mnoha způsoby. Jeden z vhodných způsobů je například přes vygenerovaný XML soubor. Nicméně když takovýto soubor otevřeme, lze z něj ledacos vyčíst i pouhým okem. Pokud však k exportu databáze nedojde a pracuje se s ní přímo jako se souborem. Data jsou uložena v následujícím formátu:

```
0470 7465 7272 6569 6368 2061 6D20 3236 2E17 terreich am 26..
0480 1405 0031 0909 566F 6C6B 7373 6368 756C ...1..Volksschul
0490 6520 6461 7565 7274 2613 0500 4F09 0957 e dauert&...O..W
04A0 2E41 2E20 4D6F 7A61 7274 2077 6172 2069 .A. Mozart war i
04B0 6E20 5F5F 5F5F 5F5F 2067 6562 6F72 656E n ______ geboren
04C0 1D12 0500 3D09 09C3 9673 7465 7272 6569 ....=..Ă-sterrei
04D0 6368 2069 7374 2072 6569 6368 2061 6E6C ch ist reich anl
04E0 1006 0081 5909 094E 6163 6820 6465 6D20 ... Y..Nach dem
04F0 322E 2057 656C 746B 7269 6567 2077 7572 2. Weltkrieg wur
0500 6465 20C3 9673 7465 7272 6569 6368 2069 de Ă-sterreich i
0510 6E20 7669 6572 2042 6573 6174 7A75 6E67 n vier Besatzung
```

Obrázek 2: Formát dat v databázi

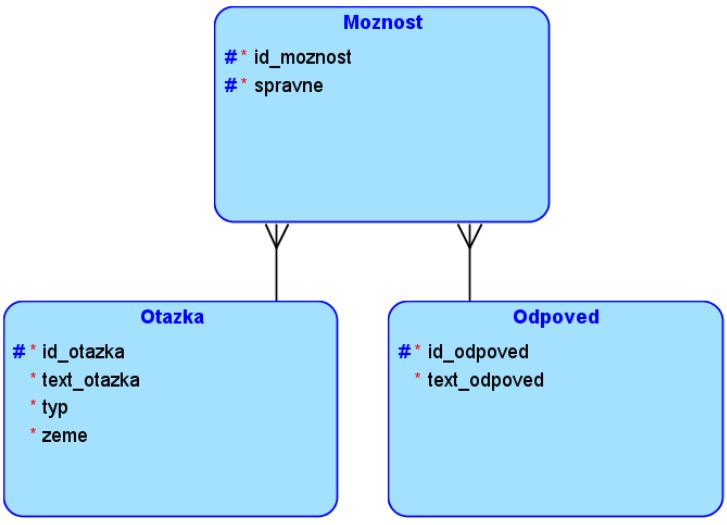
Qt Creator podporuje SQLite databázi a k ní volně stažitelné jednoduché SQLite Studio, ve které byla databáze vytvořena. Na základě poskytnutých materiálů od školy bylo sestaveno 157 otázek a vloženo do 3 jednoduchých tabulek. Databáze funguje na jednoduchém principu záznamů v tabulkách Otázka a Odpověď, spojené tabulkou Možnost pro přerušení M:N vazby.

Díky využití databáze se zbavujeme duplicitních údajů při využívání stejných odpovědí (zejména číselné údaje, data, jména měst apod.) Klíčovou je tedy spojovací tabulka Možnost, jejíž záznam si v atributech ukládá *id_otázky*, *id_odpovedi* a atribut Správně obsahuje hodnotu *TRUE* pro správnou odpověď dané otázky.

6.2.1 Příklad

Příklad záznamu v tabulce pro otázku z ABCD testu:

```
Otázka (id_otazka, text,otázka)
    ID_1 Hlavnim mestem nemecka je:ěď
Odpov (id_odpoved, text_odpoved)
    ID_10 Berlin
    ID_11 Hamburg
    ID_12 Bremen
    ID_13 Frankfurt
Moznost (id_moznost, id_otazka, id_odpoved, spravne)
    ID_21 ID_1 ID_10 1
    ID_22 ID_1 ID_11 0
    ID_23 ID_1 ID_12 0
    ID_24 ID_1 ID_13 0
```



Obrázek 3: Schéma databáze

Můžeme tedy vidět, že pouze první odpověď má pro otázku `ID_1` správný parametr. V jiné otázce může být odpověď Berlin nesprávná, ale nový záznam v tabulce odpověď už nevznikne a tyto varianty se vyřeší možnostech.

6.3 Práce s vlastními moduly a třídami v QML

V souboru lze také definovat vlastní moduly. V našem případě například *Database 1.0*. Tímto importem vytvoříme propojení mezi jazykem QML a aplikační logikou, tedy třídami psanými v C++. Nicméně pouhý import v *main.qml* nestačí, je ještě nutné přidat do *main.cpp* příkaz:

```
qmlRegisterType("Database",1,0,"Database");
```

Tato metoda má 4 parametry. Textové parametry určují název třídy a název importovaného modulu, číselné parametry zase jeho verzi. Po provedení tohoto importu lze vytvořit QML objekt ve stromové struktuře. Připojením id objektu k němu lze přistupovat a vytvářet mu dané atributy a funkce a záznam objektu vypadá následovně:

```
Database {
    id: mojeDatabaze
    ...
}
```

Dříve v této kapitole je ukázáno, jak lze definovat atributy a funkce jednotlivým objektům. Jednalo se však o funkce a atributy lokální, tedy takové, které lze využít pouze v rámci QML. Abychom se však dostali i k námi definovaným atributům a funkcím a mohli lépe zasahovat do aplikační logiky, je třeba provést následující úpravy také v hlavičkových souborech jednotlivých tříd.

```
class Database : public QObject
{
    Q_OBJECT
    Q_PROPERTY(QSqlDatabase db READ db WRITE setDb)
    Q_PROPERTY(QVector otazky READ otazky WRITE setOtazky)
    . . .
public:
    . . .
}
```

Třída, která již byla naimportována pomocí *qmlRegisterType*, musí být potomkem *QObjectu*. Objekty jsou strukturovány v objektovém stromě a tedy při principu dědičnosti přejímají vlastnosti. Výhodou objektů je možnost volat na ně signály a sloty. Makro *Q_OBJECT* na začátku definice třídy je povinné pro všechny objekty, které využívají signály, sloty a nebo atributy (*Q_PROPERTY*). Dále tedy můžeme definovat jednotlivé atributy. Každý atribut má před sebou makro *Q_PROPERTY*, dále datový typ a vlastní getr a setr. Soustřeďme se na příklad atributu *QSqlDatabase db*.

V public sekci třídy si vytvoříme atribut stejného datového typu se standardní notací.

```
QSqlDatabase m_db;
```

A dále nadefinujeme zmíněný getr a setr:

```
QSqlDatabase db() { return m_db; }
void setDb(const QSqlDatabase& newDb) {m_db = newDb;}
```

Nyní se stává kód přeložitelným a atributy lze volat i z QML souboru. Abychom mohli volat funkce, přiřadíme k jejich definici v hlavičkovém souboru makro *Q_INVOKABLE*. Například:

```
Q_INVOKABLE void vytvorDatabazi();
```

Nyní lze metody volat z různých míst v QML jen pomocí id a názvu metody. Podle zvoleného příkladu tedy:

```
mojeDatabaze.vytvorDatabazi();
```

6.4 Třídy aplikace

Aplikace je vyvíjena objektově orientovaným programováním a tedy každá třída má vlastní soubor. Kromě souborů zastupujících třídy máme v aplikaci standardně *main.cpp* pro chod aplikace a *main.qml* pro grafické rozhraní. V této kapitole je popsána logika aplikace a jejich jednotlivých tříd.

6.4.1 MainEngine

MainEngine je třída, která se stará o celkovou logiku aplikace, drží si instance obtížností a databáze a zprostředkovává vyhodnocování testů. Jak jsme zmiňovali v kapitole UML Diagram, tak využívá návrhový vzor Abstraktní továrna. Vzor spočívá ve vytvoření abstraktní třídy *Level*, jejíž instance nelze vytvořit a v jejich potomcích. Volba obtížnosti probíhá za běhu aplikace zvolením třídy *Septima*, nebo *Oktava* v náležitém *GroupBoxu*. Po zvolení následuje uložení hodnoty do statické třídy *Settings* (spočívá v návrhovém vzoru Singleton) a při volání konstruktoru instance *MainEngine* se tato hodnota předává právě odsud.(Čápka, 2012)

```
switch(lvl) {
    case 2 : m_level = new oktava();
        break;
    case 1 : m_level = new septima();
        break;
```

Pro ošetření situace jiného čísla než zmiňovaných je vytvořena defaultní možnost, při které se vytvoří snazší varianta *Septima*. Při testování k jiným dosazeným hodnotám nedocházelo, nicméně tato varianta se ponechává pro bezpečnost aplikace. Konstruktory jednotlivých dosazených instancí jsou bezparametrické.

6.4.2 Level

Další třídou je právě abstraktní třída *Level*, zmiňovaná v podsekci *MainEngine*. Tato třída má pouze jednu metodu a to na vyhodnocování testů.

```
virtual float vyhodnoceni(int spravne, int spatne, int nezadane)=0;
```

Označením funkce jako čistě virtuální zároveň říkáme překladači, že v každém potomku jeho funkce musí být tato funkce definovaná. Pokud se tak nestane, překladač ohlásí chybu. Funkce třídy jako takové je právě aby mohla být děděna, proto nevytváří konstruktor. Samotnému programu potom nezáleží na tom, jaký potomek je v instanci dosazen a zavolá pouze čistě virtuální funkci, která při správném postupu vrátí náležitý výsledek.(Solter, Gregoire, Kleper, 2011)

6.4.3 Septima a Oktáva

Septima a Oktáva jsou třídy pojmenované podle ročníků gymnázia a určují, ve které třídě se student nachází. V každé třídě musí být definovaná zmíněná funkce vyhodnocení a konstruktor.

```
float vyhodnoceni(int spravne, int spatne, int nezadane);
```

Syntaxe definice funkce už vypadá standardně a v implementačním souboru je dále provedeno vyhodnocení. Metoda přejímá 3 číselné parametry, které značí množství správných a špatných odpovědí a pak těch, které nebyly zvoleny. Hlavní výhodou tohoto principu je snadná rozšiřitelnost. Pokud se rozhodneme přidat přísnější hodnocení (například učitelé se chtějí otestovat s tvrdšími podmínkami), stačí pouze vytvořit novou třídu s touto metodou, dosadit instanci a jinak kód aplikace není třeba měnit.

6.4.4 Odpověď

Následující třída obsahuje jednotlivé instance odpovědí. U diagramu tříd je na první pohled vidět, že u odpovědí není třeba řešit velké složitosti.

```
int m_idOdpovedi;
QString m_textOdpovedi;
bool m_spravnost;
```

Každá instance odpovědi má své unikátní ID, svůj text a *boolean* hodnotu o tom, zda je správně.

6.4.5 Otázka

Třída otázka reprezentuje záznam jednotlivých otázek v aplikaci. S touto třídou pracuje třída spravující databázi. Existuje několik typů testů a tím pádem i otázek, proto třída obsahuje větší množství parametrů.

```
int m_id;
QString m_text_otazky;
int m_idSpravneOdpovedi;
int m_idZadaneOdpovedi;
int m_pismenoZadaneOdpovedi;
QString m_text_doplnovaci_odpovedi;
QVector m_moznosti_doplnovacich_odpovedi;
```

Každá otázka má své unikátní ID a znění (text otázky). Potom ale musí nastat určité dělení. Pro otázky, kde se volí možnosti, se v otázce využívají tři parametry. Je uloženo id správné odpovědi, id zadané odpovědi a také volba, kterou uživatel provedl. Tedy například otázka z ABCD testu si pamatuje, že uživatel zvolil možnost C, ta náleží odpovědi s ID 10 a správná odpověď má ID 11. Pamatování si zvoleného písmena je zde pro snazší procházení testu, aby uživatel viděl, jak odpovídal v předchozích otázkách.

Tímto principem už ale nejde vyřešit doplňovací test, neboť tam si musíme pamatovat textové řetězce, které uživatel napsal. Pro uživatelovu odpověď je tedy parametr *m_text_doplnovaci_odpovedi*. Nicméně ne vždy jsou otázky takto jednoznačné. Jedna z připomínek školy byla, že někteří žáci jsou zvyklí psát německé ostré s znakem *ß*, ale jiní používají také povolené značení *ss* a tedy obě odpovědi musí být uznané jako správné. Příklad s ostrým s není jedinou výjimkou a tedy pro možnost rozšiřitelnosti řešení je přidán bezrozměrný vektor. V tomto poli textových řetězců jsou obsaženy všechny přípustné správné odpovědi a je procházeno při vyhodnocování tohoto typu testu.(Solter, Gregoire, Kleper, 2011)

6.4.6 ComboBoxModel

Následující třída v aplikaci zastupuje prvek *ComboBox* grafického rozhraní. Je využita při učitelském rozhraní aplikace při přidávání nových otázek, odpovědí a podobně. Umožňuje uživateli rozkliknout výběr možností a zvolit právě jednu. Toto ošetření situace umožňuje dobrou kontrolu nad tím, aby uživatel nezadal nesmyslná data, které by následně mohli shazovat aplikaci. Jelikož se do *ComboBoxu* vkládají prvky z databáze, musí být zastoupen jak v aplikační logice, tak i v logice uživatelského prostředí. To je dosaženo zmíněným vytvořením vlastního modulu v jazyce QML. Třída má jeden parametr s datovým typem *QStringList*. Je to jednoduché pole textových řetězců, které jsou v *ComboBoxu* vyobrazeny. Abychom k němu snadno přistupovali, zařadíme ho mezi *Q_PROPERTY*.

```
Q_OBJECT
    Q_PROPERTY(QStringList comboList READ comboList WRITE setComboList
        NOTIFY comboListChanged)
    . . .
```

Metoda pro naplnění má v definici makro *Q_INVOKABLE* a lze tedy zavolat z grafického prostředí.

```
Q_INVOKABLE void naplnComboBox(QStringList data);
```

Přijímá parametr pole textových řetězců, které dále nastaví do příslušného parametru. Samotná třída reprezentuje model dat, které se v prvku budou zobrazovat. Jelikož si ho upravujeme dle vlastních potřeb, musíme provést import v souboru *main.cpp*:

```
ComboBoxModel combo;
QQmlContext *classContext = engine.rootContext();
classContext->setContextProperty("comboModel", &combo);
```

Podle zvoleného jména modelu ho můžeme nastavit v atributech jednotlivých prvků grafického rozhraní.

```
ComboBox {
id: comboBox
model: comboModel.comboList
}
```

Do atributu model vložíme námi definovaný *comboModel* a jeho atribut *comboList*, odkazující na pole textových řetězců.

6.5 Logika databáze

Třída spravující databázi má největší funkcionalitu a je nejrozsáhlejší. Stará se o nahrání souboru databáze do aplikace, úpravu a mazání otázek v učitelském módu, sestavování testů, výpisu a podobně. Tyto funkce jsou popsány v této sekci. Sekce také spadá pod logiku aplikace, nicméně nad databází je prováděno nejvíce operací, tak je jí vyčleněna tato sekce.

6.5.1 Nahrání databáze

Jak již bylo zmíněno databáze byla vytvořena v programu SQLiteStudio, ze kterého byla exportována pomocí souboru. Jednou z funkcí této třídy je korektní nahrání souboru do této aplikace.

V hlavičkovém souboru třídy, která se stará o databázi přidáme následující kód:

```
#include <QSql>
#include <QSqlDatabase>
```

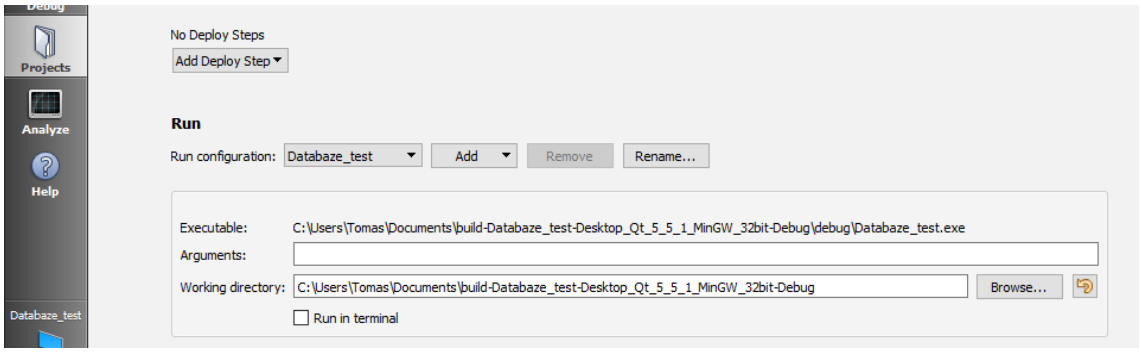
Abychom nalinkovali modul *QtSql* do projektu, musíme do projektového souboru přidat tento řádek:

```
QT += sql
```

Nyní je práce s SQL databází možná a přejdeme k metodě pro vytvoření:

```
QSqlDatabase db = QSqlDatabase::addDatabase("QSQLITE");
db.setHostName("databaze_otazek");
db.setDatabaseName("Db.db3");
m_db = db;
```

První příkaz vytváří instanci databáze *QSqlDatabase*. Následující metoda addDatabase přejíma parametr *QSQLITE*, který určuje, že se jedná o databázi typu SQLite3. Metoda *setHostName* nastavuje jméno databáze, a *setDatabaseName* cestu k souboru pro načtení. V tomto případě přejímá pouze název souboru, který se nachází v pracovním adresáři aplikace. Poslední příkaz ukládá nastavenou databázi do proměnné třídy.(Qt Documentation, 2015) U desktopové verze je nastavení pracovního adresáře v Qt Creatoru v záložce Projekt.



Obrázek 4: Pracovní adresář desktop

6.5.2 Nahrání databáze – Android

Příkazy pro vytvoření databáze jsou totožné, nicméně pracovní prostředí android zařízení se nachází v telefonu, nikoliv v počítači. Pokud kód zanecháme v původním znění, aplikace bude korektně fungovat v simulátoru, ale na konkrétním zařízení překlad způsobí problémy. V projektovém souboru s příponou *.pro* přidáme následující kód:

```
deployment.files += Db.db3
deployment.path = /assets
INSTALLS += deployment
```

Přidáním tohoto kódu nastavíme nahrání souboru databáze do složky assets. V této složce se nachází soubory, se kterými aplikace pracuje. Nyní máme soubor nalinkovaný v projektu. Abychom mohli databázi používat na zařízení, je třeba tam soubor nakopírovat. Pro provedení této operace jsou vloženy následující řádky kódu:

```
QFile dFile("assets:/Db.db3");
    if(dFile.exists()){
    dFile.copy(dir.currentPath() + "/Db.db");
    } else { qDebug() << "Soubor nenalezen"; }
```

Tento kód vytvoří instanci souboru *dFile*, který v parametru přijímá cestu k databázi. Proměnná *dir* je instancí adresáře a volaná metoda uvnitř kopírování souboru vrací aktuální pracovní adresář. Následuje podmínka pro ověření, že byl soubor do složky *assets* zkopírován korektně. Poté je soubor zkopírován do daného zařízení. Databáze je nahrána totožně jako při desktopové verzi do aplikace.

6.5.3 Vytvoření testu

V této podsekci je popsáno vytvoření testu. Jeho nastavení volíme v metodě *nastavTest*:

```
Q_INVOKABLE void nastavTest(int zeme, int typ);
```

Metoda přijímá dva parametry z grafického rozhraní. Obě tyto hodnoty jsou číselné a uloženy ve statické třídě. Již podle názvu můžeme vidět, že se jedná o zvolenou zemi a typ testu.

```
void Database::nastavTest(int pocetOtazek, int zeme, int typ){
QString pom = " SELECT id_otazka, text_otazky FROM Otazka "
              " WHERE zeme = " + QString::number(zeme) +
              " AND typ = " + QString::number(typ) +
              " GROUP BY id_otazka";
Database::naplnOtazky(pom,pocetOtazek);
}
```

V metodě se vytvoří šablona SQL dotazu, který z databáze vytáhne otázky. Přijaté parametry ovlivní provedený dotaz a ten se dále posunuje do metody k naplnění pole otázek.

```
void Database::naplnOtazky(QString sql, int pocetOtazek){
    QSqlQuery query;
    query.exec(sql);
    while (query.next()) {
        int pom_id = query.value(0).toInt();
        QString pom_text = query.value(1).toString();
        otazka* pom = new otazka(pom_id,pom_text);
        m_otazky.push_back(pom);
    }
    Database::nahodnyTest(pocetOtazek);
}
```

V této metodě je vytvořený SQL dotaz volán. Jelikož se jedná o dotaz, který vrací více než jeden řádek, přistupuje se k jednotlivým datům v cyklu. Dokud jsou nalezeny další otázky, jsou postupně nahrávány do proměnné *m_otazky*, se kterou pak databáze dále pracuje. Do proměnné se však nahrají veškeré otázky odpovídající kritériím ale velikost testu je omezená. O tento problém se stará další metoda *nahodnyTest*.

```
void Database::nahodnyTest(int pocetOtazek){
      // rozmer odkud vybírám
      int celkPocetOtazek = m_otazky.size();
      // pocet otázek ve finálním vektoru
      int pomPocet = 0;
      QVector pomVec;
      QVector cisla;
      bool stejnaOtazka = false;
      srand(time(NULL));

      while (pomPocet != pocetOtazek){
          int cislo = rand()%celkPocetOtazek;
          // poprví provedení
          if (cisla.empty()){
              cisla.push_back(cislo);
              pomVec.push_back(m_otazky.at(cislo));
              pomPocet++;
              cislo = rand()%celkPocetOtazek;
          // Pridala se prvni otazka
          } else {
              // pri druhém a~kazdém dalsím pruchodu
              for (int i=0; igetIdSpravne() == m_otazky.at(i)->
                  getIdZadane()){
. . .}
```

Každá otázka má atribut, ve kterém je uloženo ID správné odpovědi a ID odpovědi, která byla zaznačena. Jednoduchým porovnáním dvou čísel tedy dojdeme k požadované informaci.

U testu doplňovacího se může vyskytovat více správných odpovědí a algoritmus tedy musí projít všechny přípustné možnosti.

```
for (int j=0; jm_moznosti_doplnovacich_odpovedi.size();j++){
    QString pom = m_otazky.at(i)->getTextDoplnovaciOdpovedi();
    if (m_otazky.at(i)->m_moznosti_doplnovacich_odpovedi.at(j) == pom){
        spravne = true;
        }
    }
}
```

V uvedeném cyklu se projdou všechny správné odpovědi uložené v proměnné a porovnávají se s odpovědí zadanou uživatelem. V případě shody se nastaví pomocná proměnná správně na pravdivou hodnotu.

6.6 Úprava databáze – učitelský mód

V aplikaci je používaná defaultní databáze, na které už si studenti mohou testovat své znalosti. Učitel má však možnost databázi upravit, rozšířit, či promazat. Při přihlašování do aplikace lze zaškrtnout učitelskou možnost a po zadání hesla lze spravovat databázi otázek. V této části grafického rozhraní se volají funkce pracují-

cí s databází, ke kterým studentské rozhraní nemá přístup. Důležitými třídami pro tento mód je samozřejmě třída spravující databázi a prvek *comboBoxModel*. První verze této aplikace spočívala ve vyhledávání otázek a zapisování jejich ID čísel do patřičných boxů, na základě kterých se otázka například editovala. Po prvotním testování se toto řešení ukázalo jako nešťastné, neboť uživatel měl příliš mnoho možností zapisovat do aplikace, například znaky místo číslic, záporná čísla a podobně. Tyto konkrétní situace byly kódem ošetřeny, nicméně jistě ne všechny možnosti lze snadno obsáhnout. V případě, kdy uživatel zapisuje hodnoty do aplikace, dojde k chybě mnohem snáze, než když jsou mu poskytnuty předem nachystané možnosti. Druhá varianta se také ukazuje jako mnohem více uživatelsky přátelská. *ComboBoxModel* je využit pouze jeden, neboť vždy se do něj vkládá ID a text. Při mazání a editaci jde o otázku, v případě vkládání odpověď.

6.6.1 Smazání otázek

Mazání otázek patří k jednoduchým příkazům nad databází. Využitím příkazu *DELETE* v korektní syntaxi s patřičným ID otázky je otázkou jednoho řádku kódu. Při zvolení otázky v *comboBoxu* proběhne přiřazení jejího ID do pomocné proměnné a následně se zavolá metoda *smazat*.

```
QString sql = "DELETE FROM Otazka WHERE id_otazka = " + m_pomID;
QSqlQuery query;
query.exec(sql);
sql = "DELETE FROM Moznost WHERE id_otazka = " + m_pomID;
query.exec(sql);
```

V případě mazání pouze otázek z databáze by vznikalo množství zbytečných dat v tabulce možnosti. Každá otázka a odpověď je spojena spojovací tabulkou možnosti a například u ABCD testu vzniknou každé otázce možnosti hned 4. Při smazání otázky jsou data neškodná, aplikaci nijak neohrozí, ale zabírají zbytečné místo. Proto je třeba je smazat společně s otázkou na základě stejného ID. Před potvrzením smazání otázky a možností je uživatel dotázán, aby nedocházelo k nechtěnému mazání.

6.6.2 Editace otázek

U otázek, které není třeba hned mazat, ale například se vyskytuje v zadání pouze gramatická chyba, lze editovat. Princip vybrání je totožný jako při mazání, nicméně v tomto případě se volá příkaz *UPDATE*.

```
QString sql = "UPDATE Otazka "
              " SET text_otazky = '" + novyText + "'  "
              " WHERE id_otazka = " + m_pomID;
QSqlQuery query;
query.exec(sql);
```

Jediným rozdílem je, že metoda přijímá textový řetězec jako vstupní parametr, který je využit v dotazu pro upravení otázky. Parametr je vytažen z textového pole

z uživatelského prostředí. Před editací otázky je uživatel taktéž dotázán, aby potvrdil změny.

6.6.3 Přidání odpovědí

Při přidávání otázek se nám dostává výčet možných odpovědí ke zvolení. U ABCD jsou to 4, u doplňovacího testu jsou 1 až 4. Nicméně uživatel vždy vybírá z již existujících odpovědí. V případě mazání otázky jsou mazány otázky a možnosti, nikoliv odpovědi. Tedy pokud smažeme nějakou otázku, která sdílí odpovědi s jinými, žádný problém to nezpůsobí. Smazání samotné odpovědi by způsobilo pád aplikace kvůli přistupování k neexistující hodnotě, případně nechtěné smazání hned několika otázek. Z tohoto důvodu mazání odpovědí není umožněno. Pouze může nastat to, že se v databázi vyskytují nevyužité odpovědi. Nenalezne-li uživatel hledanou odpověď, přidáním se z uživatelského prostředí zavolá následující metoda:

```
void Database::pridatNovouOdpoved(QString text){
    QString sql = "SELECT max(id_odpoved) FROM Odpoved";
    QSqlQuery query;
    query.exec(sql);
    while(query.next()){
        m_pomID = query.value(0).toString();
    }
    int pom;
    pom = m_pomID.toInt();
    pom++;
    sql = "INSERT INTO Odpoved VALUES (" + QString::number(pom) +",'" +
        text + "' );";
    query.exec(sql);
}
```

Metoda přebírá textový řetězec jako parametr. Jedná se o text nové odpovědi. Následuje SQL dotaz využívající agregační funkci max, která vrací nejvyšší obsažené ID v databázi odpovědí. Typovou konverzí je tato textová hodnota reprezentující nejvyšší ID převedena na číslo a následně zvýšena o jedničku. Nové číslo slouží jako ID nově vkládané odpovědi.

6.6.4 Přidání otázek

U přidávání odpovědí bylo ukázáno, jak rozšířit výčet možných odpovědí a nastíněn princip vložení nové otázky. Při vkládání nové otázky se zobrazí volba typu testu a země. Při zvolení každého testu se zvolí textové pole pro vložení textu otázky a dále výčtové pole odpovědí. U ABCD testu se zobrazí 4 možnosti pro každé písmeno, přičemž pole pro správnou odpověď je rozlišeno zeleným textem. U typu pravda/nepravda se zobrazí dvě možnosti a při doplňovacím testu se zobrazí ještě jedna nabídka. Počet správných odpovědí k doplňovací otázce sice může být mnoho, nicméně otázky by měly být natolik jednoznačné, aby správných odpovědí bylo nej-

více 4. Vyobrazená nabídka teda umožňuje zvolit přidání až čtyř odpovědí. Každý typ má svoji vlastní metodu přidání.

```
void Database::pridatNovouMultipleOtazku(QString zadani, QString
    spravna, QString druha, QString treti,QString ctvrta, int zeme, int
     test){
    QString sql = "";
    QSqlQuery query;
    int id_otazky = pridatOtazku(zadani,zeme,test);
    int id_moznost = zjistiMaxMoznost();

    QStringList odpovedi;
    odpovedi.clear();

    odpovedi.push_back((spravna.split(")").at(0)));
    . . . // nahrani i~ostatnich id do pole

    sql = "INSERT INTO Moznost (id_moznost, id_otazka, id_odpoved,
        spravne) VALUES ( "
            + QString::number(id_moznost) + "," + QString::number(
                id_otazky)
            + "," + odpovedi.at(0) + ",1);";
    id_moznost++;
    query.exec(sql);
. . . // zbyle insert prikazy
}
```

Pro příklad je uvedena otázka pro ABCD test. V parametrech přejímá textové zadání otázky, čtyři odpovědi, typ země a testu. Na základě stejného principu jako při mazání, kdy musíme kromě otázky mazat i možnosti, zde musíme možnosti přidávat. Možnosti mají unikátní ID, a pro přidání nové možnosti si do pomocné proměnné převedeme nejvyšší současné ID. Dále vytvoříme pomocné pole textových řetězců pro shromáždění ID odpovědí. Posledním krokem je zavolání SQL příkazu *INSERT* s příslušnými parametry a zvýšení proměnné *id_moznost* o jedničku. Počet příkazů *INSERT* se liší podle typu testu.

U doplňovacího testu, kde lze přidat až čtyři odpovědi, je využito překrytí metod.

```
Q_INVOKABLE void pridatNovouDoplnOtazku(QString zadani, QString spravna
    , int zeme, int test);
Q_INVOKABLE void pridatNovouDoplnOtazku(QString zadani, QString spravna
    , QString druha,int zeme, int test);
. . .
```

Metody se jmenují totožně a liší se zde počet parametrů. Z grafického prostředí se tedy volá vždy stejný název metody a podle počtu zadaných parametrů se rozhodne, která se provede. Metody se liší množstvím příkazů *INSERT*.

7 Testování

V této kapitole je popsáno, jakým způsobem byla aplikace upravena. Během roku projevila škola zájem vidět, jak postupuje vývoj aplikace a zda jsou splněny veškeré požadavky. Po následné domluvě s vedoucí jazykových sekcí na škole se rozhodlo pro testování prototypu. V polovině března roku 2016 byla dokončena funkční verze aplikace a byla možnost ji poprvé otestovat na studentech. Před testováním byl zapůjčen tablet Nexus 7 z Provozně ekonomické fakulty a dále použit telefon Samsung Galaxy S5 mini. Tedy dvě zařízení s různými obrazovkami.

V pondělí 21.3.2016 v půl dvanácté proběhlo testování v průběhu výuky Německého jazyka na půdě Cyrilometodějského gymnázia. Testování se zúčastnilo celkem 6 studentů, vždy po dvojicích, aby nebyla přerušena výuka. Studenti dostali zařízení a stručné instrukce a dále bylo pozorováno, jak intuitivní je grafické prostředí a podobně. Každému ze studentů se podařilo spustit některý z testů a prošli jím. Po dokončení testu proběhla s každou dvojicí krátká diskuze a jejich poznatky jsou zaneseny následovně.

7.1 Otázky

V jednom z testování se vyskytly dvě otázky přímo za sebou, kde následující otázka kontextem obsahovala odpověď té předchozí. Student tedy využil příležitosti a bez dlouhého přemýšlení vyčetl správnou odpověď. Tento problém se vyřeší rozsáhlejší databází otázek, případně lépe formulovanými otázkami. Co se týče obsahové části aplikace, jedná se pouze o testovací sadu vstupních otázek.

Další připomínka byla k nahlašování otázek. V případě, že student považuje otázku za nejednoznačnou, rád by ji nahlásil. Nicméně mobilní ani desktopová verze nepracuje s internetovým připojením, ale otázku lze nahlásit učiteli. V učitelském módu byla přidána možnost otázky editovat, mazat a přidávat nové. Učitel tedy nevhodnou otázku upraví a při příští aktualizaci databáze poskytne novou verzi aplikace na své stránky, odkud si je studenti volně stáhnou. K otázkám studenti také dodali, že by chtěli přidat nějaké fotky pro oživení testu. V této verzi aplikace se otázky s fotkami vyskytovat nebudou, nicméně vzhledem k objektové povaze aplikace není problém ji v budoucnu o takovýto typ testu rozšířit.

Poslední kritika k otázkám byla o špatně zodpovězených otázkách. Studenti požadovali možnost zopakovat test pouze s otázkami, které odpověděli špatně. Vzhledem k současné velikosti databáze otázek tato možnost nebyla přidána.

7.2 Zobrazení tlačítka pro odeslání testu

V testovací verzi aplikace se při spuštění testu zobrazovala možnost test odeslat. Přestože při stlačení tlačítka odeslat poskytla aplikace dialog „Přejete si odeslat test?“, se studentům zdála možnost odeslání matoucí. Jelikož se jedná o požadavky zákazníka, bylo přidáno zobrazení odesílacího tlačítka až při zobrazení poslední

otázky. Aplikace bude stažitelná na webu školy, a tedy studenti nebudou časově omezeni, aby museli test odesílat dříve, než ho dokončí. Při zpětném procházení testu a případných opravách odpovědí už možnost odeslání zůstává zobrazena.

7.3 Správné odpovědi

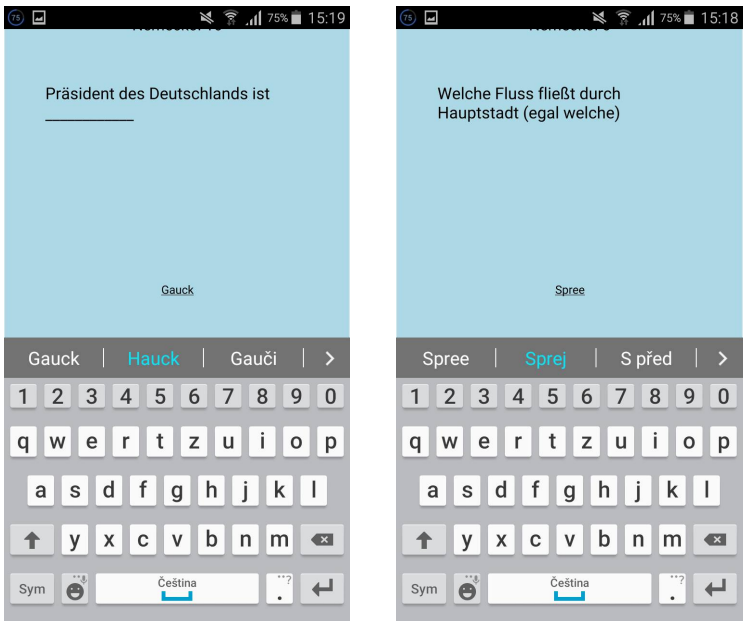
Testovací verze vyobrazovala, zda jsou zadané odpovědi správně či špatně a dále zobrazila dosažené skóre. Studenti zmínili, že by chtěli vidět svoje odpovědi a také ty, které byly správné. Ve výsledné aplikaci se při odeslání testu vypíše studentovi zadání otázky, jeho odpověď, a zda byla správně či špatně. Po bližší konzultaci s Mgr. Malečkovou bylo rozhodnuto, že se správné odpovědi zobrazovat nebudou. Studenti by si snadno nafotili výsledky a vzhledem k současné velikosti databáze, by aplikace brzo ztratila testovací smysl.

7.4 Zobrazení psaného testu

Při otázkách typu „Jaká je nejvyšší hora?“ se studenti začali ptát, jestli si zvolili Německo a nebo přeci jen Rakousko. Po vzniklé připomínce bylo přidáno do horní části testu, ze které země je test psán a na kolikáté otázce se uživatel nachází.

7.5 Nekorektní opravení doplňovacího testu

U doplňovacího testu byla objevena chyba vyhodnocování. Při zadávání odpovědi do textového pole android nabídne výchozí klávesnici zařízení. V případě, že uživatel zadá odpověď a klávesnici zavře, proběhne uložení korektně. Nicméně pokud uživatel využije funkce nápovědy, kliknutím na modře zvýrazněný text:



Obrázek 5: Nápověda u doplňovacího testu

Uloží se do výsledného textového řetězce i mezera navíc, jelikož nápověda pro psaní textu počítá s tím, že uživatel pokračuje ve větě, a tedy přidává mezeru na konec napověděného slova. Vyhodnocovací algoritmus s touto mezerou nepočítal, a tak označoval dané odpovědi jako špatné. Problém byl vyřešen funkcí, která v takovém případě odstraňuje mezeru z konce textového řetězce.

7.6 Zobrazovací chyby

- Při zopakování testu je při spuštění zaznačena první odpověď, protože neproběhlo korektní vynulování testu
- Skóre posledního testu zůstane vyobrazeno při první otázce nového testu
- Vazby a násobnosti
- Boxy s odpověďmi nejsou zarovnané

Zde jsou vypsány další chyby, kterých si studenti všimli a byly odladěny debugováním aplikace.

7.7 Přínosy

Z každého testování je důležité si odnést především zápory a kritiku. Na základě sepsaných připomínek byly nejzásadnější nedostatky opraveny a opodstatněným požadavkům bylo vyhověno. Nicméně testování ve výsledku dopadlo úspěšně. Všech 6 testovaných studentů projevilo zájem o tuto nebo podobnou aplikaci. Studenti

maturující z němčiny projevili zájem o aplikaci ještě letos, nematurující by ocenili verzi v angličtině, případně francouzštině.

7.7.1 Recenze

Zde jsou uvedeny tři pozitivní recenze po testování. Jména konkrétních studentů nejsou uvedena.

- „Na mobilu je to velmi jednoduché a intuitivní, protože mobil má dnes každý.“
- „Stejně přes den na mobilu trávím spoustu času, tak bych si alespoň procvičoval reálie k maturitě například cestou autobusem.“
- „Připomíná mi to aplikaci jako v autoškole a v té jsem se naučil mnohem více než z knížky.“

8 Diskuze

V průběhu vývoje aplikace se objevilo několik problémů. Jedním z problémů bylo rozložení prvků obrazovky na různých zařízeních. Jelikož velikosti obrazovek mají různá rozlišení, bylo u mnoha prvků využito poměrů. Tedy místo dosazených čísel byla dosazena výška nebo šířka obrazovky a vynásobena libovolným desetinným číslem. Toto řešení způsobuje, že na desktopové verzi má aplikace poměrně široké mezery mezi tlačítky a rozložení působí roztahaně. Nicméně vzhledem k počtu různých obrazovek na mobilních zařízeních bylo toto řešení zanecháno.

Dalším problémem bylo nahrání na zařízení s operačním systémem iOS. Tento problém bohužel zůstal nevyřešen. Vývoj probíhal po celou dobu také na tuto platformu, bohužel bylo naraženo na problém kopírování souboru do samotného zařízení. Při zachování stejného postupu jako pro zařízení s Androidem nedochází ke kopírování souboru. Při testování je soubor nalezen, nicméně po provedení totožných postupů, které fungují pro android, se kopírování v iOS neprovede. Tento problém zůstal nevyřešen a soustředění zůstalo na zbylých dvou platformách.

Kromě kopírování souboru databáze na zařízení je aplikace zcela funkční. Na simulátoru, který přistupuje k souborům v počítači, funguje aplikace korektně. Proběhli snahy dostat databázi do zařízení jinak než kopírováním a to stáhnutím z internetu. Pomocí přidané třídy Downloader, proběhlo korektní stažení souboru ze školního serveru akela.(Qt Documentation, 2015) Nicméně opět se vyskytl problém, že databáze nemůže být uložena rovnou do proměnné, nicméně musí být vždy nalinkovaná na soubor s patřičnou cestou. V desktopové verzi tato varianta fungovala bez problémů, ale v iOS se vyskytl opět problém přesunutí souboru do zařízení.

Pro nahrání aplikace na iPad bylo nutné vytvořit současně projekt v prostředí XCode se stejným jménem. Až po následném přeložení tohoto prázdného projektu na zařízení se začal správně detekovat projekt psaný v Qt Creatoru. Už samotná skutečnost, že pro vývoj na tuto platformu musí být uživatel registrován v developerském programu, vypovídá o celkovém zabezpečení a kopírování souboru do zařízení bude tedy větší problém, než se původně zdálo. Po čtení mnoha diskuzí a článků a konzultací jsem nenarazil na funkční řešení, aby bylo v souladu s ostatními platformami.

9 Závěr

Cílem této bakalářské práce bylo vytvořit vzdělávací pomůcku pro studenty Cyrilometodějského gymnázia v Prostějově. Konkrétně vzdělávací aplikaci do německého jazyka k procvičení znalostí o německy mluvících zemích. Cílem bylo pokrýt co nejvíce platforem a vytvořit aplikaci jak na počítač, tak na mobilní zařízení.

Prvním krokem k vytvoření aplikace bylo navštívení školy a sestavení požadavků. Během vývoje aplikace proběhlo několik konzultací s výslednou školou. Na jedné z prvních byly obdrženy výukové materiály pro maturitní ročník, na základě kterých byla vytvořena testovací databáze otázek. Na této konzultaci se také stanovily typy jednotlivých testů tak, aby korespondovaly s těmi, na které jsou studenti zvyklí. Před začátkem práce byl vytvořen diagram tříd a schéma databáze. Na základě veškerých těchto podkladů a zajištění přístupu do laboratoře virtuální reality na Provozně ekonomické fakultě, kde vývoj probíhal, byl vytvořen první prototyp práce. Vzniklá verze obsahovala funkční databázi s testy. S touto verzí proběhl test na škole, kam je aplikace cílena a byly zjištěny nedostatky a připomínky. Na základě těchto poznatků byla aplikace upravena do finální verze. Vývoj aplikace probíhal v jazyce C++ s podporou multiplatformní knihovny Qt. Toto rozhodnutí umožňovalo současný vývoj na hned několik platforem současně. Úspěšný výsledek vznikl na počítač s operačním systémem Windows i OS X. Nicméně u mobilních zařízení vznikla úspěšná verze pouze na operační systém Android. Při vývoji na operační systém iOS nastaly problémy s kopírováním databáze do zařízení. Při spuštění simulátoru ve vývojovém prostředí funguje aplikace korektně jako na ostatních platformách, ale pouze proto, že přistupuje k databázi již uložené v počítači. Při kopírování do zapůjčeného iPadu nikdy nedošlo k přenosu souboru. Po odladění ostatních platforem byly na tomto problému zkoušeny ještě jiné přístupy. Například stažení databáze ze školního serveru akela, nicméně i tento postup zarazil problém uložení souboru databáze.

Tento problém bohužel zůstal nevyřešen, tedy mobilní platforma pro iOS zůstala bez funkční databáze. Závěrem této práce lze zcela jistě říct, že v případě využití vývojových prostředí Android Studia na vývoj platformy Android a XCode na vývoj pro iOS by šlo dosáhnout lepších aplikací ať už designu nebo různých vyskakovacích upozornění funkcionalit. Důvod zastaralejšího designu je nativní chování Qt. Díky možnosti vývoje na více platforem Qt využívá nativní design výsledného zařízení. Tedy v iOS simulátoru má aplikace standartní tlačítka iOS a u Androidu zase ty svoje, které působí poměrně zastarale. Tedy uživatelské prostředí vytvořené v této aplikaci nepůsobí tak jako u nových aplikací, nicméně se to dá považovat za zanedbatelnou věc v souvislosti současného vývoje na několik platforem zároveň.

Cílem této práce není komerční prodej aplikace, ale studijní pomůcka studentům. Proto ve zhodnocení design nehraje klíčovou roli, ale pokrytí co nejvíce možností spustit aplikaci. Škole je tato aplikace poskytnuta zdarma a byla vyvíjena jako open-source, tedy náklady školy na pořízení této aplikace jsou nulové. Samotný vývoj aplikace probíhal v rámci studia, tedy mzdové náklady jsou taktéž nulové. Jediným

nákladem školy je případné rozšiřování databáze otázek, případně vytvoření obdobné databáze do jiných jazyků či jiných předmětů.

10 Seznam zdrojů

Blanchette, J. – Summerfield, M. *C++ GUI programming with Qt 4. 2.* vyd. Upper Saddle River: Prentice Hall in association with Trolltech Press, 2008. 718 s. ISBN 978-0-13-235416-5.

Čápka, D. *1. díl - Úvod do UML.* [online]. 2013. [cit. 2015-12-14]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/uml/uml-uvod-historie-vyznam-a-diagramy.

Čápka, D. *Singleton (jedináček).* [online]. 2012. [cit. 2015-12-14]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/singleton-navrhovy-vzor.

Čuchna, M. *Gartner: Světový trh PC, tabletů a mobilních telefonů vyroste v roce 2014 o 3,2 %.* [online]. 2014. [cit. 2015-05-02]. Dostupné z: http://channelworld.cz/nazory-a-analyzy/analyzy/gartner-svetovy-trh-pc-tabletu-a-mobilnich-telefonu-vyroste\{-v-roce-2014-o-3-2-12577}.

Djakoualno, L.H. *Technologie způsobí revoluci i ve školách. Je na čase.* [online]. 2012. [cit. 2015-04-21]. Dostupné z: http://zpravy.aktualne.cz/domaci/technologie-zpusobi-revoluci-i-ve-skolach-je-na-case/r~i:article:738762/.

Filová, J. *Děti a škola 21. století – výhody a rizika používání nových technologií.* [online]. 2013. [cit. 2015-04-21]. Dostupné z: http://ceskomluvi.cz/deti-a-skola-21-stoleti-vyhody-a-rizika-pouzivani-novych-technologii.

Goethe Institut *Dobrodružství s němčinou.* Článek ve formátu HTML. [online]. 2015 [cit. 2015-12-26]. Dostupné z: https://www.goethe.de/ins/cz/cs/spr/ueb/mis.html.

Kolomazník, J. *Android - Úvod.* [online]. 2016. [cit. 2016-05-05]. Dostupné z: https://akela.mendelu.cz/~xkoloma1/android/00-Uvod/prezentace.html.

Martinek, M. *Úvod do SQLite a příprava prostředí.* [online]. 2016. [cit. 2016-12-14]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/factory.

Nádoba, J. *I am going, I am playing. Nejlepší hravá aplikace na jazyky už běží i česky.* [online]. 2014. [cit. 2015-12-02]. Dostupné z: http://www.forbes.cz/i-am-going-i-am-playing-nejlepsi-hrava-aplikace-na-jazyky-uz-bezi-i-cesky.

Pecinovský, R. *OOP : naučte se myslet a programovat objektově.* 1. vyd. Brno: Computer Press, 2010. 576 s. ISBN 978-80-251-2126-9.

Pošmura, L. *Biflování jazyků končí, stačí 15 minut denně, slibují autoři aplikace*. [online]. 2015. [cit. 2015-12-02]. Dostupné z: http://hradec.idnes.cz/webove-aplikace-prectime-prichazi-s-unikatni-vyukou-jazyku-pqu.

Procházka, D. *Qt a QML*. [online]. 2014. [cit. 2015-04-04]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/318/Knihovna%20k%20projektu/Programovac%ED%20jazyk%20CPP/PDF%20soubory%20s%20p%F8edn%E1%B9kami/CPP_10_QML.pdf.

Qt Documentation. [online]. 2015. [cit. 2015-12-02]. Dostupné z: http://doc.qt.io/.

Redakce BusinessIT.cz *Mobilní zařízení ve firmě: Nové trendy, aplikace i rizika*. [online]. E-kniha ve formátu PDF. Copyright © Bispiral, s.r.o., 2014 [cit. 2015-12-27]. Dostupné z: http://www.businessit.cz/ebooks/mobilzar14.pdf.

Redakce BusinessIT.cz *Smartphony a tablety: I firmy je využívají stále více*. [online]. E-kniha ve formátu PDF. Copyright © Bispiral, s.r.o., 2013 [cit. 2015-12-27]. Dostupné z: http://www.businessit.cz/ebooks/smartphonyt.pdf.

Řezník, J. *QML – moderní uživatelská rozhraní v Qt (2)*. [online]. 2012. [cit. 2016-04-04]. Dostupné z: http://www.abclinuxu.cz/clanky/qml-moderni-uzivatelska-rozhrani-v-qt-2.

Solter, N A. – Gregoire, M. – Kleper, S J. *Professional C++*. 2nd edition. USA: Wrox, 2011. 1104 s. ISBN 0-47-093244-9.

Tondlová, K. *Moderní informační technologie mají v českém školství jasnou budoucnost*. [online]. 2013. [cit. 2015-04-21]. Dostupné z: http://www.tydenik-skolstvi.cz/archiv-cisel/2015/17/moderni-informacni-technologie-maji-v-ceskem-skolstvi-jasnou-budoucnost.

Valkovic, P. *Factory*. [online]. 2015. [cit. 2015-12-14]. Dostupné z: http://www.itnetwork.cz/navrhove-vzory/factory.

Webové stránky Cyrilometodějského gymnázia v Prostějově. [online]. 2015. [cit. 2016-04-07]. Dostupné z: http://www.cmgpv.cz/.

Webové stránky společnosti Apple. [online]. 2016. [cit. 2016-05-05]. Dostupné z: http://www.apple.com/cz/ios/what-is/.

Přílohy

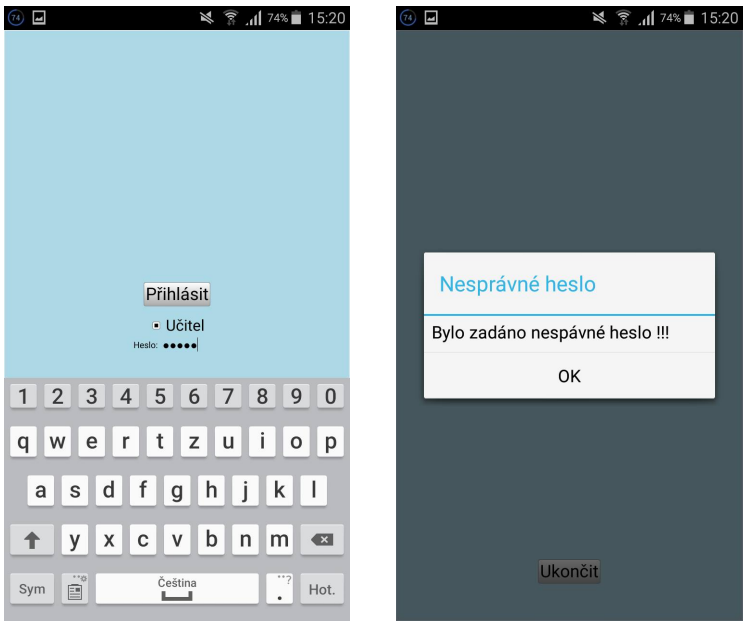
A Manuál

V této příloze je popsáno jak aplikaci správně používat. Aplikace je navržena tak, aby byla uživatelsky co nejvíce přívětivá, nicméně tvorba určité nápovědy k výsledku je standartní postup. Vzhledem k rozdělení aplikace pro studenty a pro učitele je zobrazeno přihlašování.

A.1 Úvodní obrazovka

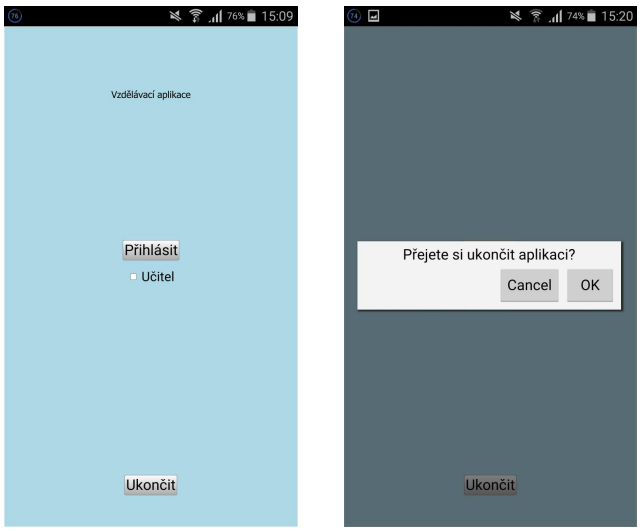
Při spuštění aplikace se jako první zobrazí úvodní obrazovka pro přihlášení. V tomto bodě lze rozlišit, zda do aplikace přistupuje učitel nebo žák. V případě přihlášení studenta stačí pouhé stisknutí tlačítka přihlásit. Vzhledem k tomu, že výsledky testů nejsou ukládány nebo odesílány, je u studenta zachována anonymita a není třeba vyplnit přihlašovací údaje. Vytvoření jednotlivých účtů studentům je možné rozšíření aplikace.

Pro přihlášení učitele je třeba zaškrtnout možnost učitel. Při stlačení boxu se objeví textová výzva pro zadání hesla, které se uživateli zobrazí skrytě. Po zadání správného hesla je uživatel přihlášen do učitelského módu.



Obrázek 6: Přihlášení učitele

Pro ukončení aplikace slouží tlačítko ve spodní části aplikace, při jehož stlačení je uživatel tázán, jestli chce aplikaci opravdu zavřít, aby nedocházelo k neúmyslnému ukončení.

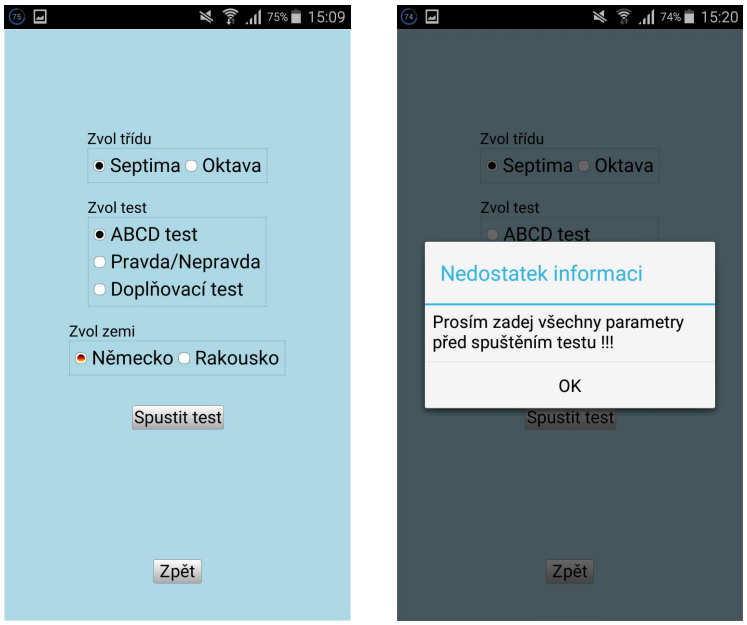


Obrázek 7: Úvodní obrazovka a ukončení aplikace

To jsou dvě možnosti přihlášení uživatele, které budou detailněji popsány v následujících sekcích.

A.2 Student

Po přihlášení je student přepnut do nového menu kde si zvolí náležité parametry testu.

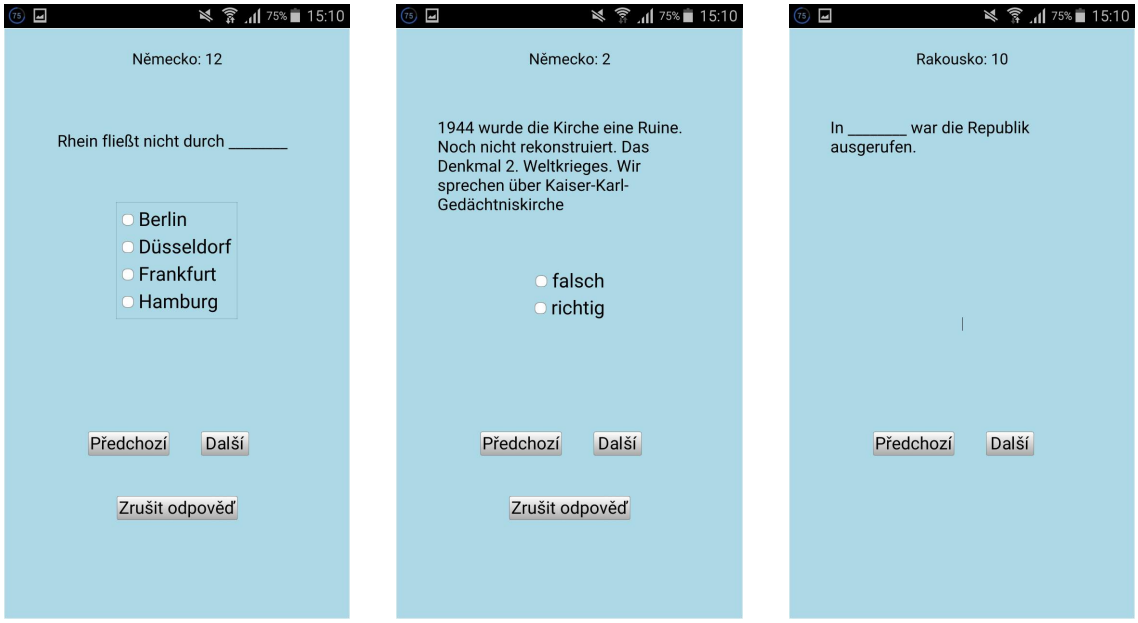


Obrázek 8: Student – volba testu

V horní části obrazovky volí, ve které třídě se nachází. Tento parametr ovlivňuje přísnost hodnocení testu. V prostřední části obrazovky zvolí test, který chce spustit a nejspodnější volba znázorňuje zemi. Na základě zvolených parametrů se vygeneruje patřičný test. Při nezadání patřičných parametrů se ukáže chybová hláška s upozorněním. Tlačítkem spustit test se potvrzují volby a tlačítko zpět vrací uživatele na úvodní obrazovku.

A.2.1 Typy testů

Na základě prostřední volby typu testu se vytvoří jedna ze tří následujících obrazovek.

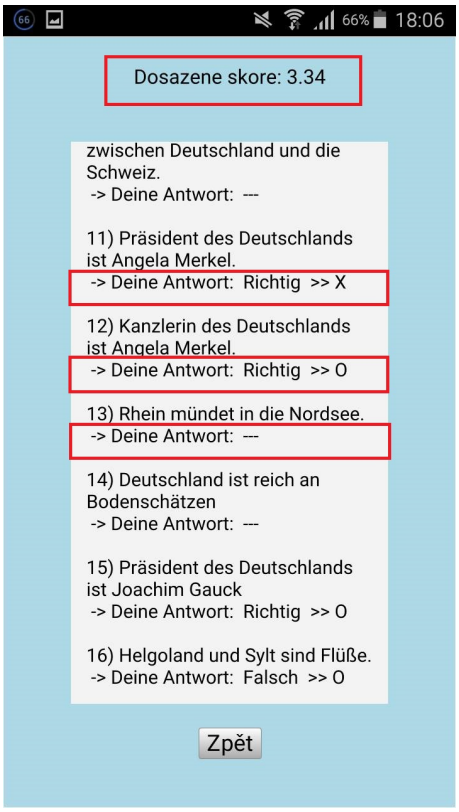


Obrázek 9: Student – volba testu

Na obrazovkách lze vidět zleva ABCD test, pravda/nepravda a doplňovací test. První test má 4 možnosti odpovědí, ve druhém je vždy pouze pravda nepravda a poslední vyžaduje od uživatele vepsat odpověď textem. V dolní části testu jsou tlačítka Další a Předchozí pro přecházení mezi otázkami. V horní části testu je uvedena země, ze které je test psán a číslo právě zobrazované otázky. Při zobrazení poslední otázky se, jak můžeme vidět na levém obrázku, ve spodní části obrazovky zobrazí tlačítko pro odeslání testu. Při odeslání je uživatel opět tázán na potvrzení. Testy, ve kterých se volí odpovědi, mají také možnost zrušení odpovědi, v případě rozmyšlení nebo neúmyslném kliknutí. Tato možnost byla přidána zejména kvůli strhávání za špatné odpovědi.

A.2.2 Vyhodnocení

Po potvrzení odeslání testu se uživatel dostane do obrazovky vyhodnocení.

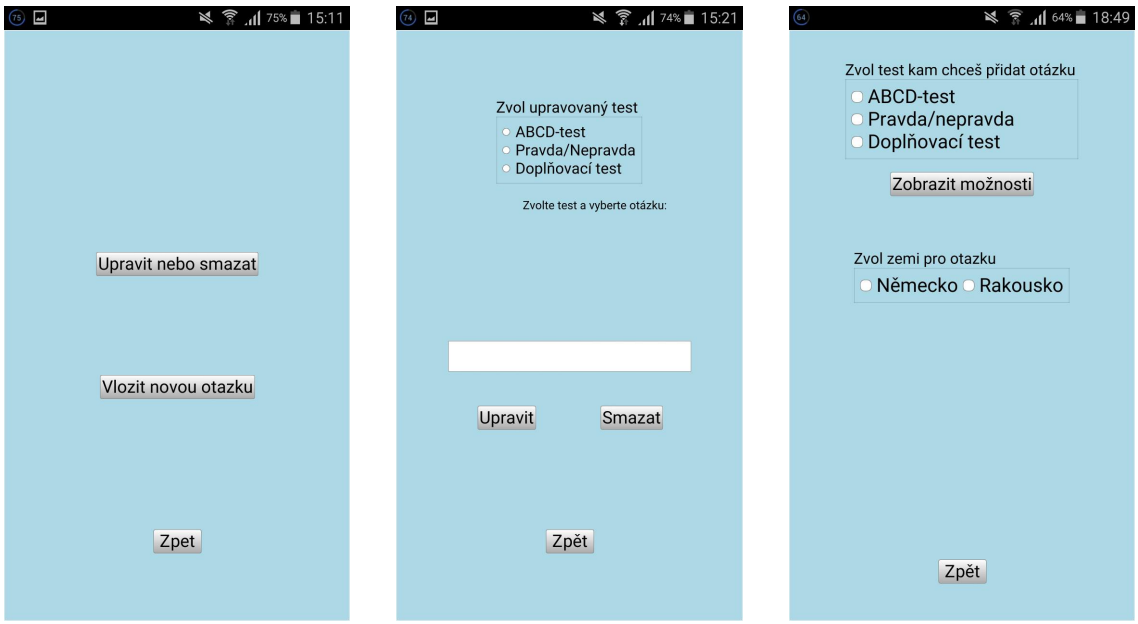


Obrázek 10: Vyhodnocení testu

Na obrázku je několik znázorněných oblastí. Obdélník v horní části obrazovky znázorňuje dosažené skóre z odeslaného testu. Pod ním se vyskytuje textová oblast obsahující výsledky. Jednotlivé záznamy obsahují číslo otázky oddělené pravou závorkou, dále text otázky. Za textem „Deine Antwort:“ se vyskytuje odpověď, kterou označil uživatel a dále znak X nebo O. Poslední znak vypovídá o tom, zda byla odpověď správně. V červeně vyznačených oblastech tedy můžeme vidět správně a špatně odpověděnou otázku. Poslední oblast ve spodní části obrazovky ukazuje vypsání otázky s neznačenou odpovědí. Tlačítko zpět vrací uživatele opět na volbu testu.

A.3 Učitel

Při korektním zadání hesla se uživatel dostane na obrazovku, kde si učitel zvolí, jak chce spravovat otázky. Při vracení se na tuto obrazovku z jednotlivých úprav zůstává učitelský mód stále přihlášen. Při opuštění na úvodní obrazovku je nutné zadat opět heslo. Jednotlivá tlačítka uživatele přesměrují na patřičné obrazovky.

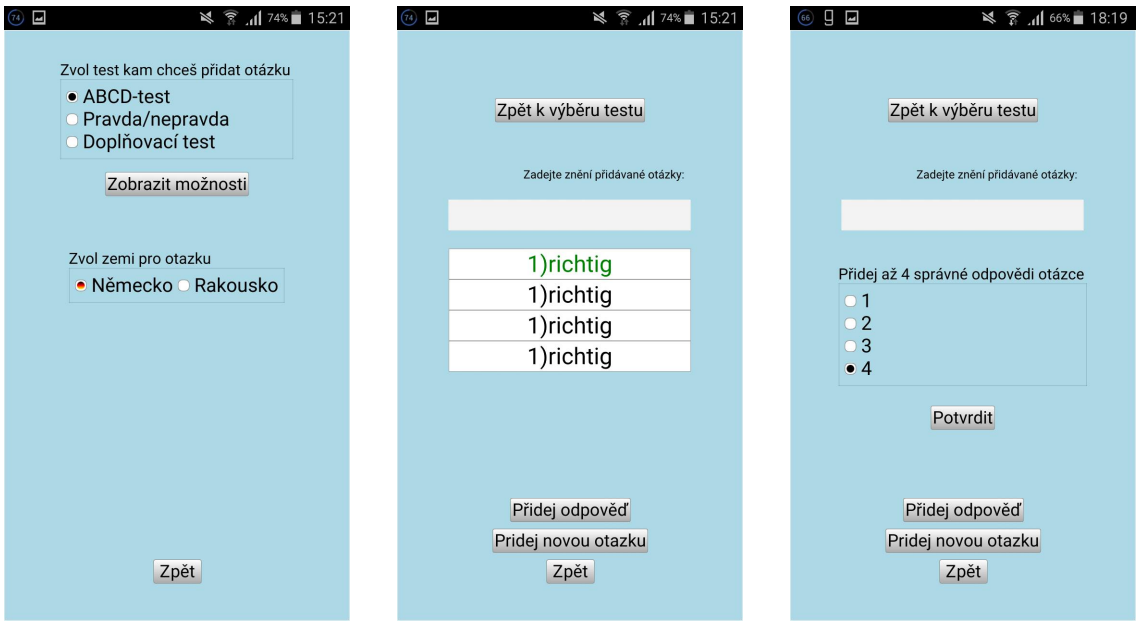


Obrázek 11: Volba učitele – možnosti

Na levém obrázku vidíme obrazovku s volbou, prostřední obrazovka se zobrazí při zvolení editace a obrazovka vpravo slouží k vkládání nové otázky.

A.3.1 Vložení nové otázky

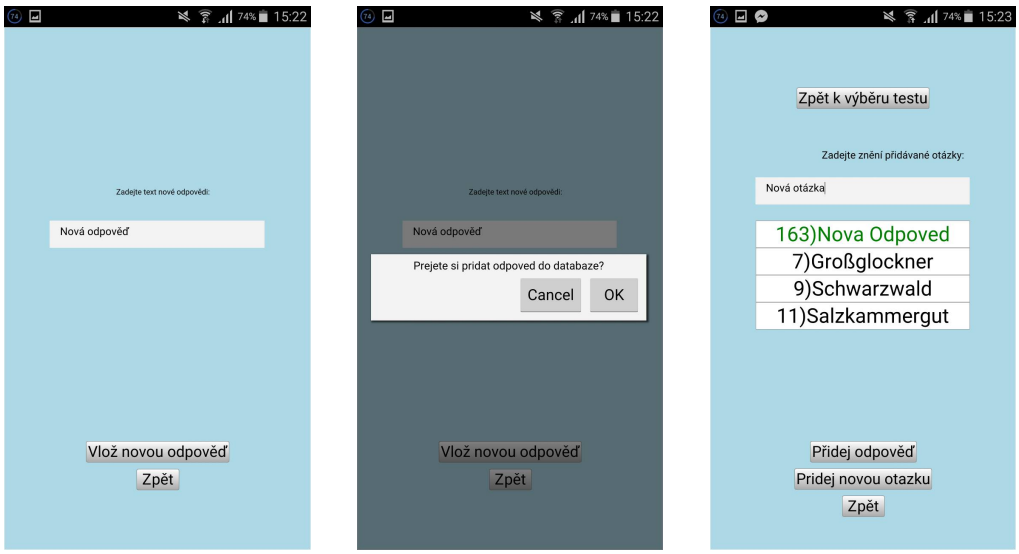
V předešlé sekci je ukázána úvodní obrazovka vkládání otázky. Při vyplnění parametrů se vykreslí standartní nabídka pro přidání otázky. V tomto příkladu byl vybrán ABCD test, ale postup je u ostatních typů testů analogický. Doplňovací test poskytuje o jednu volbu navíc, a to kolik odpovědí si přeje uživatel vložit.



Obrázek 12: Vložení otázky – ABCD příklad

Na pravém obrázku vidíme upřesňující obrazovku, která se vyskytuje pouze u přidání do doplňovacího testu.

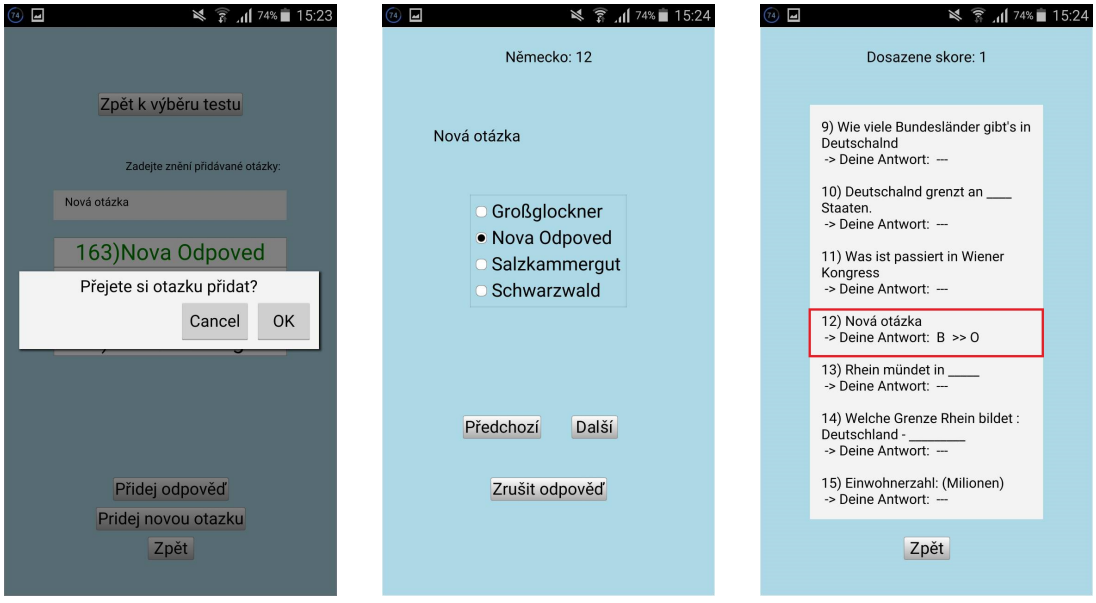
Na prostředním obrázku vidíme výslednou obrazovku, kde se zobrazí 4 možnosti, přičemž správná odpověď má zelenou barvu textu. Při kliknutí se zobrazí výčet odpovědí, kde uživatel zvolí. V případě, že uživatel nenalezne požadovanou odpověď, klikne na tlačítko pro přidání odpovědi na spodní části obrazovky a ta ho přesměruje do nové obrazovky. Zde proběhne přidání odpovědi, která se automaticky zařadí do výběru při vkládání otázky. Na testovacím příkladu vložíme novou odpověď, kterou následně využijeme jako správnou odpověď nově vkládané otázky.



Obrázek 13: Vložení nové odpovědi

Na zeleném místě přibyla uživatelem vytvořená odpověď. Text otázky byl nastaven na nová otázka a do ostatních odpovědí byly vybrány různé hodnoty, aby mohla být otázka korektně vložena. Při porušení některého z těchto pravidel se objevuje chybové hlášení.

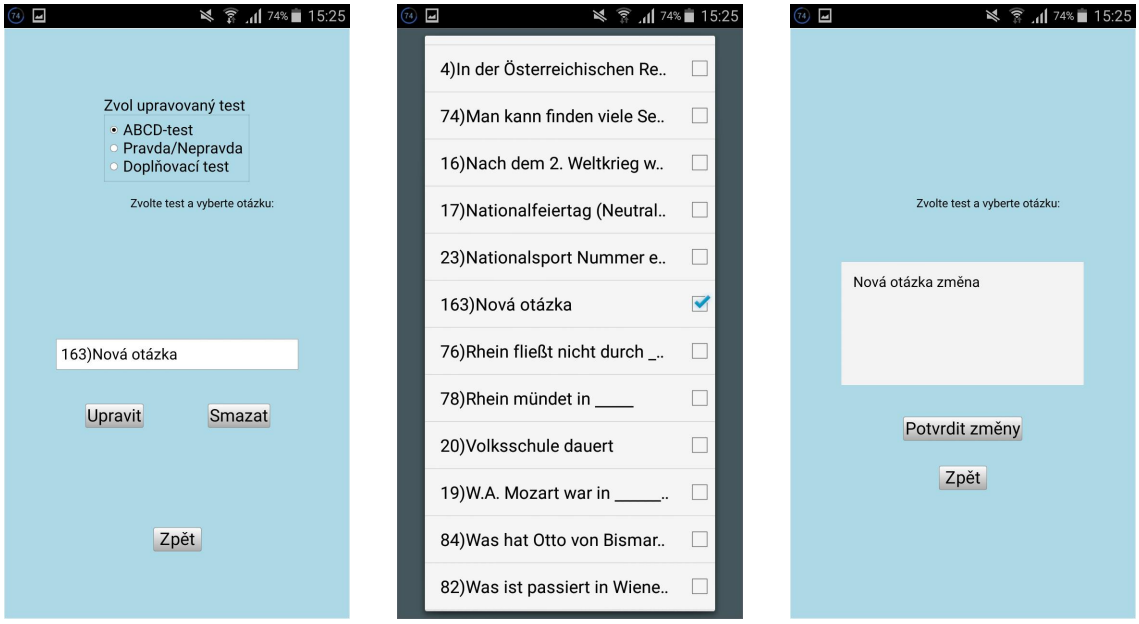
Následně je otázka vložena do databáze. Ve studentském režimu byl spuštěn ABCD test a otázka nalezena, s korektním obodováním uživatelem zvolené správné odpovědi.



Obrázek 14: Vložení nové otázky

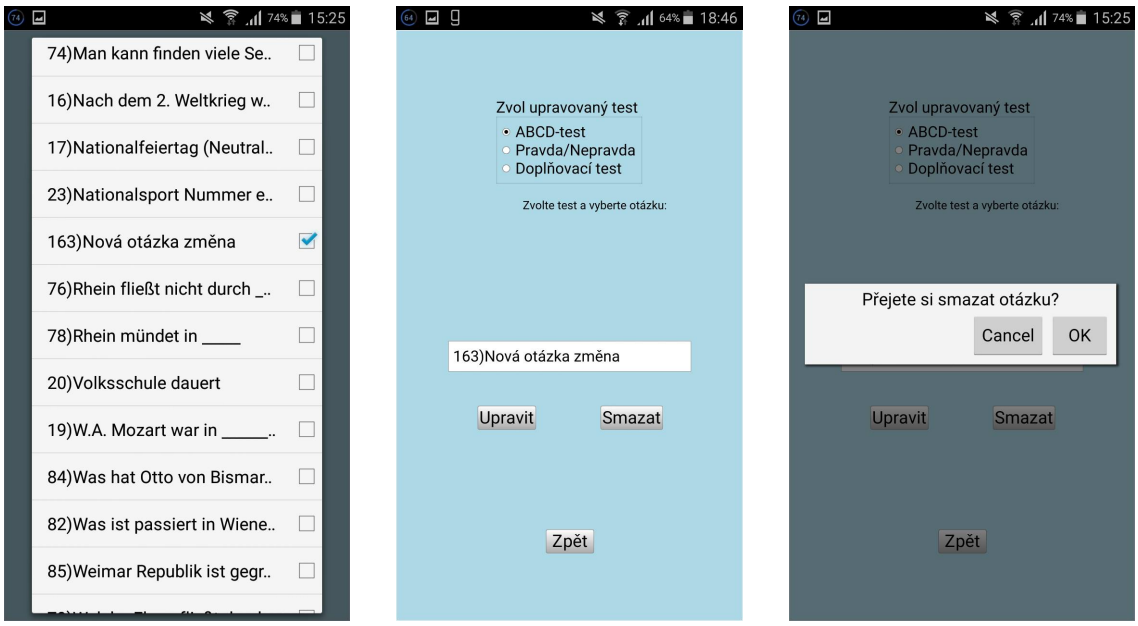
A.3.2 Editace a smazání otázky

Tyto dvě operace lze provádět na jedné obrazovce. Opět zvolíme typ testu a na zobrazeném textovém poli zvolíme ve výčtu otázku. Nalezená otázka lze upravit nebo smazat. V příkladu je uvedena úprava výše zvolené otázky a následně její smazání.



Obrázek 15: Editace otázky

Na levém obrázku je zobrazena zvolená otázka, kliknutím na ni se zobrazí výběr otázek seřazený abecedně, který je vidět na druhém obrázku. Na posledním je obrazovka, kam je uživatel přesměrován při kliknutí na tlačítko Upravit. V případě rozmyšlení uživatel vrátí na předchozí obrazovku tlačítko Zpět, v opačném případě potvrdí změny, které se okamžitě promítnou do databáze.



Obrázek 16: Smazání otázky

Smazání otázky probíhá obdobným způsobem, nyní už upravená otázka je přidána do výběru a následným stisknutím tlačítkem Smazat je odstraněna.