

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačných technológií

Tvorba užívateľských rozhraní 2021/2022

StudyDEX: Aplikácia pre memorovanie slovíčok

Technická správa k implementácii

Peter Rúček, xrucek00

Marek Miček, xmicek08

Matej Jurík, xjurik12

4.12.2021

Obsah

1	Špecifikácia výslednej aplikácie s vytvorením GUI	2
1.1	Cieľová užívateľská skupina	2
1.2	Cieľová platforma a použité nástroje	2
2	Návrh aplikácie a GUI	3
3	Popis implementácie aplikácie	4
3.1	Prepínanie Views	4
3.2	Views aplikácie	4
3.3	Controllers aplikácie	5
3.4	Repozitáre aplikácie	5
3.5	Databázové entity	6
4	Screenshots výslednej aplikácie	7

1 Špecifikácia výslednej aplikácie s vytvorením GUI

Cieľom celého tímu bolo vytvoriť desktopovú aplikáciu určenú na memorovanie cudzojazyčných/odborných slovíčok, pričom podmienkou úspešne vytvoreného produktu je prívetivé, intuitívne a ľahko použiteľné grafické užívateľské rozhranie (ďalej GUI). Toto GUI by malo zodpovedať potrebám užívateľov, ktoré boli zistené pomocou pozorovania a diskusie s užívateľmi, ktorí podobný typ aplikácie bežne používajú.

Každé slovíčko a jeho preklad/vysvetlenie je reprezentované jednou kartičkou, kde na prvej strane je dané slovíčko a na druhej jeho preklad/vysvetlenie. Užívateľ môže z ľubovoľného počtu slovíčok vytvárať jednoznačne pomenované kolekcie. V týchto kolekciách si môže prezerať jednotlivé kartičky a učiť sa ich význam. V momente keď sa cíti pripravený, môže prejsť do módu testovania, ktoré prebieha nad celou kolekciou a vyhodnotí sa až keď užívateľ prejde cez všetky kartičky. Výsledok testovania je zobrazený po každom prebehnutom teste a dáva užívateľovi predstavu o jeho priebežnom pokroku.

Takýchto kolekcií je možné vytvoriť ľubovoľný počet, pričom sa môžu priradiť do jednoznačne pomenovanej lekcie, ktorá je špecifikovaná svojím menom a oborom, ktorému patrí. Užívateľ si tak môže podľa svojich potrieb vytvárať lekcie z viacerých oborov súčasne, pričom každá lekcia ma jednoznačne roztriedené kolekcie so svojimi slovíčkami, pričom v týchto kolekciách si môže po každom teste sledovať svoj pokrok.

1.1 Cieľová užívateľská skupina

Aplikácia je určená pre ľubovoľnú skupinu užívateľov, za predpokladu, že daný užívateľ má aspoň základné znalosti o používaní a interakcií s počítačom.

1.2 Cieľová platforma a použité nástroje

Výsledným produktom je desktopová aplikácia pre operačné systémy Linux a Windows, implementovaná v programovacom jazyku Python a s využitím frameworku PyQt5 a objektovo-relačným mapovacím nástrojom (ďalej ORM) SQLAlchemy. Konkrétne verzie použitých nástrojov a postup pre spojazdnenie aplikácie sú spísané v súbore `readme.md`.

2 Návrh aplikácie a GUI

Výsledná aplikácia je implementovaná pomocou architektúry Model-View-Controller (ďalej MVC), ktorá umožňuje oddeliť frontend (ďalej FE, určený ku komunikácii s užívateľom), reprezentovaný prvkom View, a backend (ďalej BE, určený na manipuláciu a uchovanie dát) reprezentovaný prvkom Model. Controller potom mapuje akcie užívateľa na konkrétne funkcie Modelu a spätne zaistuje zobrazenie výsledku do View.

Pri implementácii sa využíva objektovo orientovaný prístup, takže každý Model, Controller a View je reprezentovaný samostatnou triedou, ktorých inštancie sa podľa potreby medzi jednotlivými prvkami MVC vymieňajú. Keďže sa v rámci BE využíva ORM, aj samotné databázové entity sú modelované ako triedy. Pre každú entitu existuje repozitár, ktorý podľa potreby s ňou komunikuje a zapisuje, aktualizuje či dotazuje sa na konkrétne dáta.

Každá lekcia, kolekcia, kartačka v StudyDEX má vlastné View, pričom majú približne rovnaké rozloženie. Pri každom je jasne viditeľný jej názov, tlačítka pre editáciu a vymazanie danej komponenty, a tlačítko pre pridanie novej (dcérskej) komponenty. Teda napríklad rovno v detaile lekcie je možné pridávať ďalšie kolekcie a v detaile kolekcie je možné pridávať ďalšie karičky so slovíčkami. Po spustení aplikácie sa zobrazí takzvané domovské View dajú pridávať nové lekcie. Užívateľ tu má tiež zobrazené svoje doteraz vytvorené lekcie po ktorých rozkliknutí sa mu zobrazí ich detail. Do módu testovania sa dá prejsť z detailu kolekcie, pričom testovanie prebieha na osobitnom View.

Užívateľ prechádza všetky slovíčka v kolekcii, pričom sa test vyhodnotí až keď prejde cez posledné slovíčko v kolekcii. Pri každej kartačke má tri možnosti:

- a) druhú stranu kartačky pozná a nepotrebuje si ju otáčať, je si tým istý
- b) druhú stranu kartačky nepozná
- c) kartačku si otočí z dvoch dôvodov:
 - aby sa uistil, že danú kartačku naozaj pozná
 - kartačku nepozná a potrebuje sa ju doučiť

Po ukončení testu sa zobrazia jeho výsledky v nasledujúcej forme. V novom View je zobrazený počet správne a nesprávne zodpovedaných slovíčok, plus koľkokrát užívateľ požiadal o otočenie kartačky. Tiež mu je ponúknutá možnosť na opakovanie testu. V prípade, že test nechce opakovať, môže sa vrátiť do detailu danej kolekcie, kde vidí, koľko slovíčok z celej kolekcie už ovláda.

3 Popis implementácie aplikácie

Vstupný bod celej aplikácie sa nachádza v module `main.py`, ktorý sa nachádza v koreňovom adresári celého projektu. Na začiatok je potrebné dodať, že pre každé `View` existuje `Controller`, ktorý si u seba vytvorí inštanciu svojho `View` a ďalej pracuje s jeho položkami a podľa potreby ho prekresľuje. Týmto spôsobom sa vytvorí aj domovské `View` a to inštanciou triedy `MainWindowController()` v funkcii `main()`. Je možné naplniť `DB` ukázkovými dátami, ktoré sa po spustení aplikácie zobrazia a to odkomentovaním riadku s volaním funkcie `seed()`.

3.1 Prepínanie Views

Keďže vo výslednom produkte sa vykresľuje viacero `View`, medzi ktorými bude užívateľ podľa potreby prepínať, je potrebné tieto `Views` ukladať na zásobník. Pridávanie/odoberanie týchto `Views` zo/na zásobník bude simulovať ich prepínanie. V `PyQt5` sa pre tento účel využíva trieda `QStackedWidget()`, ktorá poskytuje tento zásobník. Vždy keď je potrebné vykresliť nové `View`, na zásobník sa toto nové `View` uloží, pričom je viditeľné len to, ktoré bolo uložené ako posledné. V prípade, že sa užívateľ chce vrátiť na predchádzajúce `View`, je potrebné zo zásobníku odstrániť aktuálne ale aj cieľové `View`. Cieľové je potrebné vždy vykresliť na novo, aby sa dynamicky prejavili prípadné zmeny urobené v predošlom `View`. Prepínanie týchto `Views` je riadené v module `Controllers/ModeratorController.py` pomocou takzvaného moderátora, ktorý má uchované inštancie všetkých `Controllerov` aplikácie. Volaním príslušných funkcií týchto `Controllerov` sa na novo vykreslí požadované `View`.

3.2 Views aplikácie

Všetky `Views` aplikácie sa nachádzajú v adresári `/View`. Ich finálny vzhľad je pomocou screenshotov zachytený v kapitole `Screenshots` výslednej aplikácie. Každý `View` vo svojom konštruktoze inicializuje položky, ktoré bude zobrazovať a definuje ich pozície. No samotný obsah týchto položiek a akcie, ktoré sa uskutočnia po užívateľovej interakcii sa definujú v `Controlleri` manipulujúci s daným `View`.

3.3 Controllers aplikácie

Každý View má priradený Controller v adresári /Controllers. V konštruktoze každého Controllera sa vytvoria inštancie repozitárov, pomocou ktorých sa bude pristupovať k databázovej vrstve (ďalej DB). Tiež sa v ňom ukladajú inštancie zásobníku s Views a moderátora, aby mohol Controller požiadať o vykreslenie nového View. Následne sa vytvorí inštancia daného View, pomocou ktorej sa staticky vytvoria jeho základné komponenty. Tieto komponenty sa dynamicky naplnia príslušnými dátami, ktoré Controller dostal z DB pomocou repozitárov.

Tiež sa tu nachádzajú sloty, ktoré spracujú signál vyslaný užívateľom po interakcii s View. Medzi základné akcie patrí pridávanie, vymazanie, aktualizovanie lekcí, kolekcí či kartičiek. Pred pridaním či aktualizovaním lekcie je potrebné skontrolovať, či lekcja s rovnakým menom už neexistuje. Rovnaký postup sa aplikuje aj pri kolekciách a kartičkách.

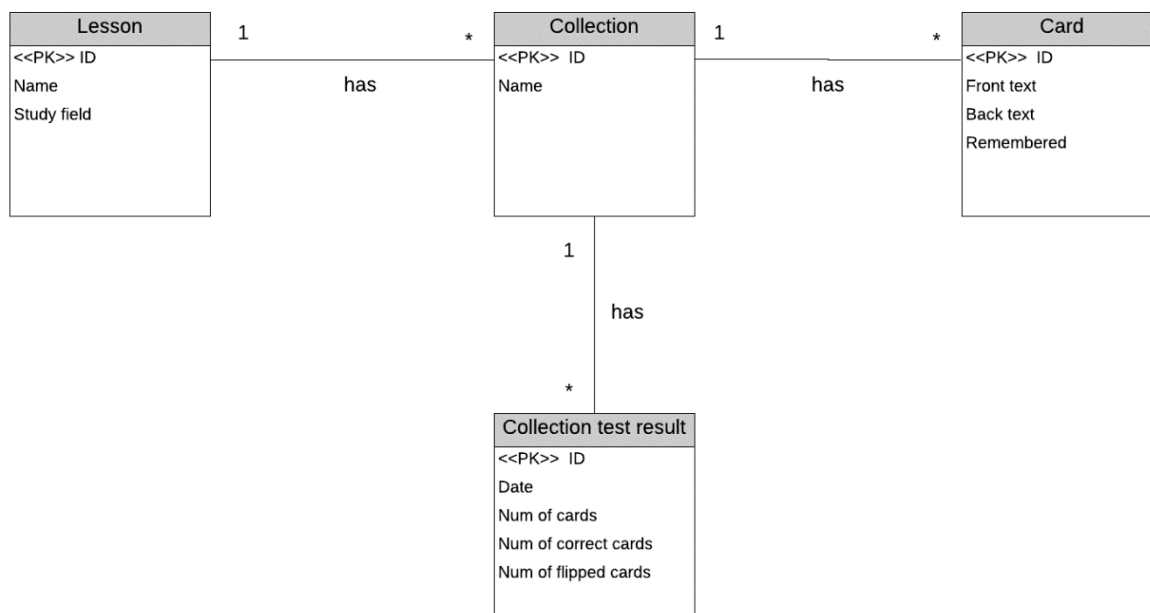
Pri zachytení signálu pre pridanie novej lekcie, kolekcie či kartičky sa vygeneruje rovnaký View ako pre zobrazenie detailu existujúcej lekcie, kolekcie a kartičky. Rozdiel je v tom, že užívateľovi sa odkryjú tlačítka pre vymazanie novo vytvorenej komponenty či pridanie novej komponenty až v prípade, že uloženie prebehne v poriadku. Teda napríklad užívateľ musí špecifikovať meno a obor novej lekcie, aby mu mohlo byť umožnené pridávať do tejto lekcie nové kolekcie. To všetko za predpokladu že zadal unikátne meno lekcie a uloženie prebehlo bez problémov. Obdobný postup sa aplikuje aj pri kolekciách a kartičkách.

3.4 Repozitáre aplikácie

Ku každej databázovej entite/triede sa pristupuje cez daný repozitár nachádzajúci sa v adresári /Models. V každom repozitári sú implementované funkcie, ktoré vracajú výsledky SQLAlchemy dotazu na danú databázovú triedu. Tieto funkcie sú volané príslušným Controllerom, ktorý takto získané dáta zobrazí vo View, ktoré má pridelené. Repozitáre samozrejme umožňujú aj ukladať nové položky do DB vždy, keď ich o to požiada Controller, ktorý obdržal signál a dáta od užívateľa.

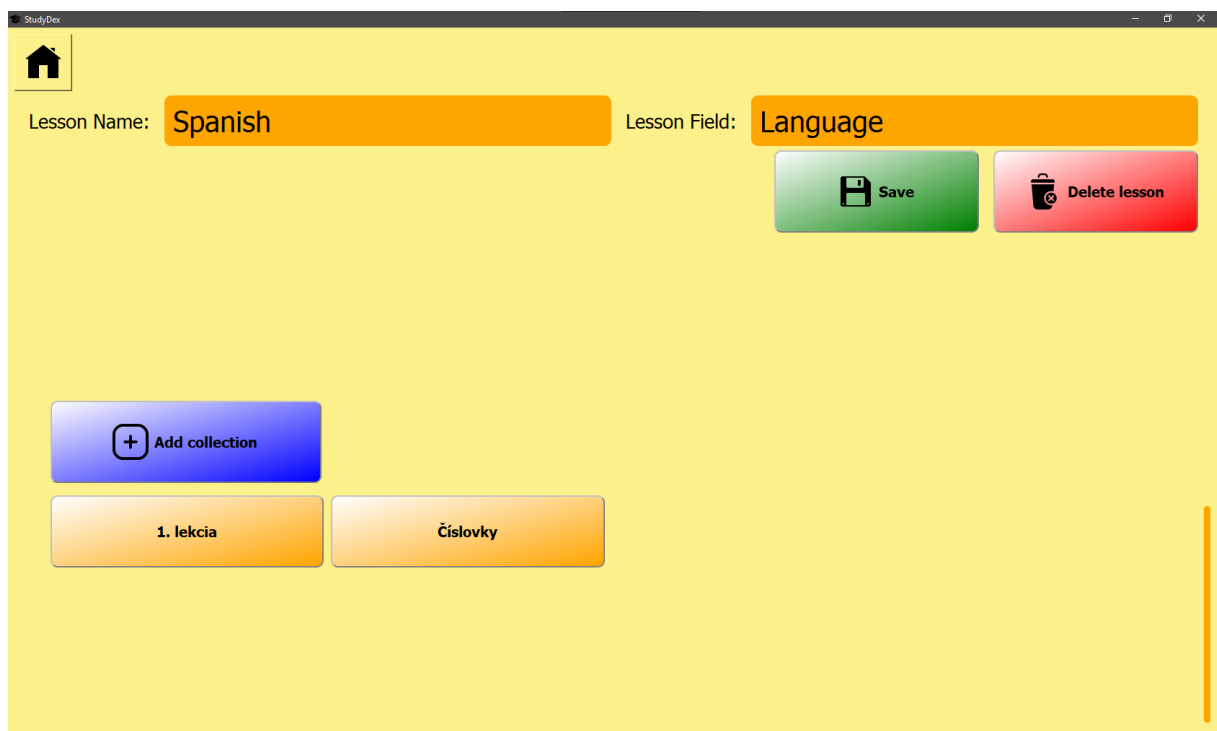
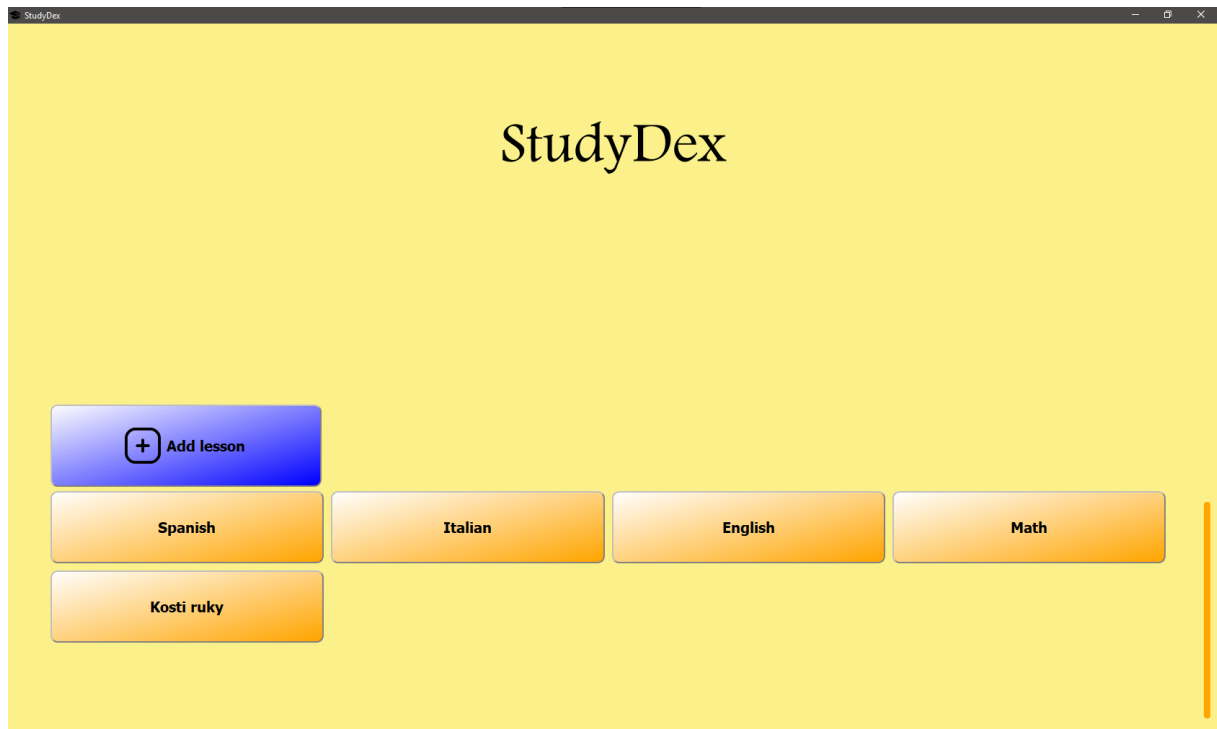
3.5 Databázové entity

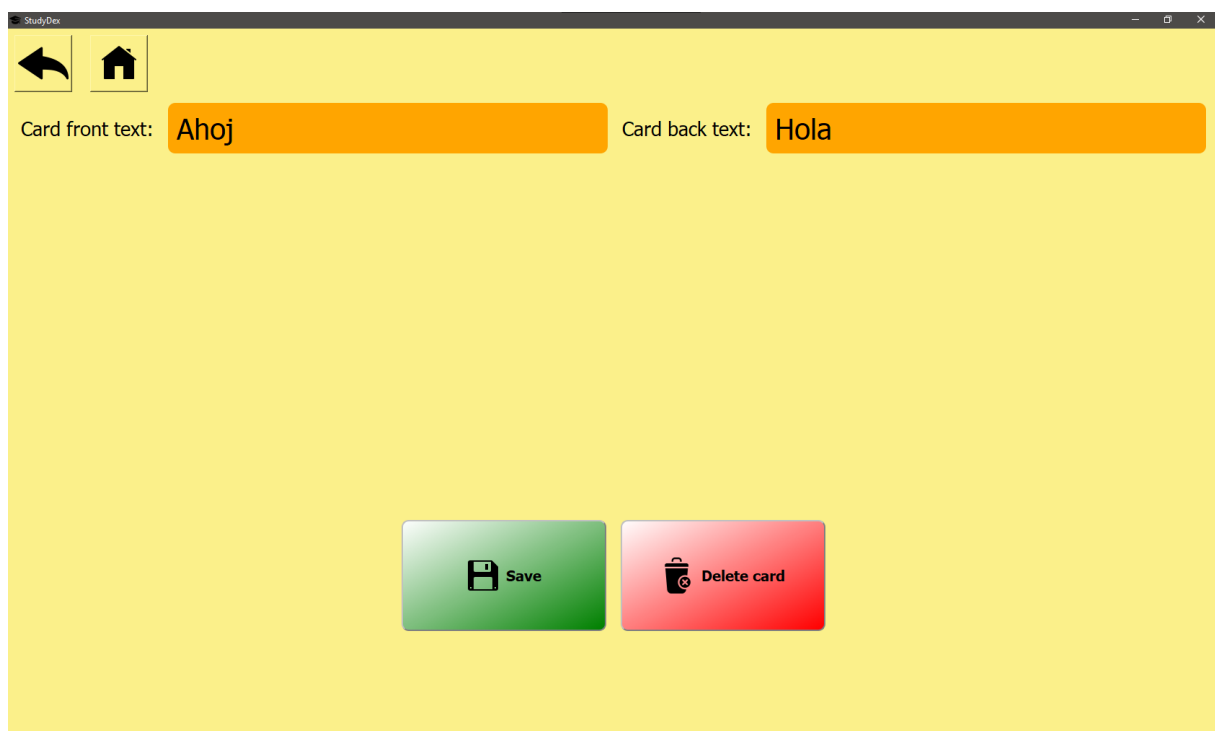
DB je tvorená troma tabuľkami (viz Obr.3.1). Keďže sa v aplikácii používa ORM, tak sú tabuľky reprezentované triedami, pričom sú implementované modulom `/Models/DbEntities.py`.



Obr.3.1: ERD StudyDEX


4 Screenshoty výslednej aplikácie









StudyDev



1. lekcja: Test Results

Correct: 3
Times flipped: 2
Incorrect: 0

 Retry test

 Finish test