Ekonomická fakulta
Faculty of Economics

Jihočeská univerzita v Českých Budějovicích
University of South Bohemia in České Budějovice

Jihočeská univerzita v Českých Budějovicích
Ekonomická fakulta
Katedra aplikované matematiky a informatiky

Bakalářská práce

Vývoj desktopové aplikace k procvičování matematických úloh pro OS Windows

Vypracovala: Zdeňka Kolářová
Vedoucí práce: Mgr. Radim Remeš

České Budějovice 2022

JIHOČESKÁ UNIVERZITA V ČESKÝCH BUDĚJOVICÍCH
Ekonomická fakulta

Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Zdeňka KOLÁŘOVÁ**
Osobní číslo:	**E18459**
Studijní program:	**B6209 Systémové inženýrství a informatika**
Studijní obor:	**Ekonomická informatika**
Téma práce:	**Vývoj desktopové aplikace k procvičování matematických úloh pro OS Windows**
Zadávající katedra:	**Katedra aplikované matematiky a informatiky**

Zásady pro vypracování

Cílem bakalářské je vytvořit desktopovou aplikaci k procvičování matematických úloh. Aplikace bude pracovat ve dvou režimech: student (procvičování úloh) a učitel (editace a správa úloh). Program bude uživatelovi umožňovat volbu obtížnosti procvičování, bude zaznamenávat jeho historii učení a průběžně sledovat pokroky. V režimu editace úloh bude aplikace umožňovat zadávat vlastní úlohy v rámci interního editoru, bude obsahovat funkcionalitu exportu a importu matematických úloh.

Metodický postup:

1. Studium odborné literatury.
2. Návrh a popis vývoje a implementace výsledné aplikace.
3. Zhodnocení použitelnosti aplikace pro nasazení v reálném prostředí.
4. Závěr.

Rozsah pracovní zprávy:	**40 – 50 stran**
Rozsah grafických prací:	**dle potřeby**
Forma zpracování bakalářské práce:	**tištěná**

Seznam doporučené literatury:

1. Albahari , J., & Albahari , B. (2017). *C# 7.0 in a Nutshell.* Sebastopol, California (USA): O'Reilly Media.
2. Harrington, J. L. (2016). *Relational Database Design and Implementation.* Cambridge, MA, USA: Morgan Kaufmann.
3. Nagel, C. (2016). *Professional C# 6 and .NET Core 1.0.* Indianapolis, Indiana (USA): John Wiley & Sons.
4. Posadas, M. (2016). *Mastering C# and .NET Framework.* Birmingham, UK: Packt.
5. Price, M. J. (2017). *C# 7.1 and .NET Core 2.0: Modern Cross-Platform Development.* Birmingham, UK: Packt Publishing.

Vedoucí bakalářské práce: **Mgr. Radim Remeš**
Katedra aplikované matematiky a informatiky

Datum zadání bakalářské práce: **15. ledna 2019**
Termín odevzdání bakalářské práce: **14. dubna 2020**

V Českých Budějovicích dne 19. března 2019

JIHOČESKÁ UNIVERZITA
V ČESKÝCH BUDĚJOVICÍCH
EKONOMICKÁ FAKULTA
Studentská 13
370 05 České Budějovice

L.S.

doc. Dr. Ing. Dagmar Škodová Parmová
děkanka

doc. RNDr. Jana Klicnarová, Ph.D.
vedoucí katedry

Prohlášení

Prohlašuji, že svou bakalářskou práci jsem vypracovala samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své bakalářské práce, a to – v nezkrácené podobě/v úpravě vzniklé vypuštěním vyznačených částí archivovaných Ekonomickou fakultou – elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

……	……
Datum	Podpis studenta

Poděkování

Tímto chci poděkovat všem, kdo mi byli při tvorbě mé bakalářské práce nápomocni. Jmenovitě jde v prvé řadě o vedoucího mé bakalářské práce, Mgr. Radima Remeše, jehož podněty a realistický náhled na celý projekt mi umožnily najít rovnováhu mezi mou velkolepou vizí a tím, co dokážu uskutečnit. Za nenahraditelnou psychologickou pomoc vděčím PhDr. Martině Fülepové, s níž jsem o své bakalářské práci mluvila na nejednom sezení, a jejíž laskavé náhledy a praktické rady mi pomohly bojovat se strachem z neúspěchu i prokrastinací. A samozřejmě děkuji i své rodině a přátelům, kteří při mně stáli, ať už to zrovna s tvůrčím procesem vypadalo jakkoliv.

Obsah práce

1 Úvod

Prvotní myšlenka vytvořit aplikaci Nezmatematika se zrodila v roce 2018 v době, kdy jsem ve volném čase doučovala matematiku, a to převážně na úrovni druhého stupně základní školy. Příklady na samostatné procvičování mezi jednotlivými doučovacími sezeními jsem v té době studentům dávala na listech papíru a v zásadě se nijak neodlišovaly od příkladů, které dostávali za domácí úkoly ve škole.

Při vyhodnocování těchto příkladů za úkol ale opakovaně vycházelo najevo, že zvláště studenti, kteří nemají dostatečně zafixované základy, často s řešením bojují proto, že si nejsou jisti postupem, a když při sobě zrovna nemají nikoho, kdo problematice rozumí a na koho by se mohli obrátit, nevědí si rady. Při procházení daných příkladů společně ale mnohdy stačilo vyjasnit jeden jediný krok z postupu a ty zbylé už přirozeně zapadly na svá místa. Využití interaktivních učebních materiálů se přímo nabízelo.

Považovala jsem za důležité moci zadávat své vlastní úlohy, umožňovat studentům zobrazit si, co by mělo být dalším krokem k vyřešení úlohy (formulovaným jazykem, kterému studenti rozumějí), a v ideálním případě také k procvičování nepotřebovat připojení k internetu. Bývala bych upřednostnila také volně dostupný software před placeným, ale ani mezi placenými nástroji jsem nenacházela takové, které by vyhovovaly mým požadavkům.

Proto si tato práce klade za cíl zdokumentovat vývoj právě takové aplikace na procvičování matematiky, která vyučujícím umožní vytváření vlastních sad úloh včetně zobrazitelných kroků k vyřešení dané úlohy – takovým způsobem, jakým to své studenty učí v hodinách. Studentům, kterým vyučující tyto sady úloh zpřístupní, tak aplikace umožní řešit úlohy jednu po druhé (včetně možnosti vrátit se k předchozím úlohám) a v případě potřeby si zobrazit nápovědu formulovanou jazykem, na nějž jsou zvyklí.

V práci popisuji procesy analýzy, návrhu a realizace celého projektu, včetně ukázek zdrojového kódu.

2 Přehled řešené problematiky

Matematika coby vědní obor je nosným pilířem informatiky, stejně tak jako rozmanitých dalších disciplín lidského počínání. Informační technologie tuto skutečnost matematice v mnoha ohledech neméně významně oplácí, ale v oblasti výuky matematiky své místo a uplatnění stále hledá. A v současné době také úspěšně nachází.

Obzvláště během měsíců distanční výuky v období pandemie covid-19 se studenti i vyučující museli přizpůsobit naprosto odlišnému systému, než na jaký byli do té doby zvyklí. Poptávka po softwaru pro výuku matematiky vzrostla a lze předpokládat, že v blízké budoucnosti se na vývoji tohoto trendu nic nezmění.

2.1 Wolfram|Alpha

Wolfram|Alpha je ambiciózní dlouhodobé intelektuální úsilí s cílem učinit veškeré systematické znalosti okamžitě vypočitatelné a dostupné co nejširšímu možnému spektru lidí, napříč všemi profesemi a úrovněmi vzdělání (Wolfram|Alpha, b.r.-a). Poprvé byl Wolfram|Alpha veřejnosti zpřístupněn v roce 2009 jako webová stránka vypočítávající odpovědi na dotazy v přirozeném jazyce (Wolfram|Alpha, b.r.-b).

Vývoj nástroje neustále pokračuje a dotazům matematického charakteru je dnes vyhrazena celá samostatná kategorie, přičemž webová aplikace uživateli nejen najde řešení, ale také nabídne možnost zobrazit si celý postup po jednotlivých kocích. Po stránce uživatelského rozhraní je Wolfram|Alpha velmi přívětivý, jednotlivé kroky řešení jsou názorně ilustrovány, a tak se právem stal velmi oblíbeným (nejen) mezi vysokoškolskými studenty i profesory.

Mým potřebám ale nevyhovoval z několika důvodů – je dostupný pouze v angličtině, vyžaduje připojení k internetu a v neposlední řadě uživatel získává celé řešení, jakmile příklad zadá. Nejedná se o nástroj k procvičování, ale o pomůcku k pochopení.

2.2 Geogebra

Aplikace Geogebra je dynamický matematický software sdružující výuku geometrie, algebry, statistiky a počtů. Má komunitu čítající miliony uživatelů po celém světě, zdarma dostupné výukové materiály, které lze sdílet např. přes platformu Geogebra Classroom, a snadno použitelné ovládání (GeoGebra, b.r.).

Vznikla jako součást diplomové práce Markuse Hohenwartera v akademickém roce 2001/2002 a od té doby získala několik mezinárodních ocenění včetně European

Academic Software Award a German Educational Software Award (Hohenwarter & Penier, b.r.).

Geogebra je dostupná v češtině, kromě webové aplikace má i desktopovou verzi nezávislou na internetovém připojení a vytváření vlastního obsahu k ní patří zcela neodmyslitelně. Jde bezesporu o skvělou pomůcku do výuky, protože vizualizace geometrických principů pochopení podstatných souvislostí zásadně usnadňuje. V domácích podmínkách, bez doprovodu mluveného komentáře vyučujícího, ale ztrácí klíčovou část toho, co ji ve vyučovacích hodinách činí tak efektivní.

2.3 Photomath

Photomath je mobilní aplikace dostupná pro operační systémy Android i iOS, která dokáže řešit vyfocené ručně psané či tištěné matematické problémy, přičemž zároveň postup řešení rozkládá na jednotlivé kroky, které uživateli srozumitelně vysvětluje. Obsahuje také vědeckou kalkulačku, nabízí interaktivní grafy a podporuje různé způsoby řešení. První verze byla vydána v roce 2015 a nepodporovala skenování ručně psaného textu (příklady tedy musely být tištěné). Vývoj aplikace Photomath stále pokračuje a veškerý její obsah je dnes přeložen do více než 30 jazyků včetně češtiny (Photomath, b.r.).

K fungování nicméně potřebuje připojení k internetu, a spíše než o pomůcku k procvičování samotnému se jedná o nástroj ke kontrole správnosti postupu a výsledku u již spočítaného příkladu.

2.4 Mathman

Mobilní aplikaci Mathman vyvinul a publikoval v akademickém roce 2019/2020 v rámci své diplomové práce tehdejší student, dnes již absolvent Vysokého učení technického, Ing. Jan Maloušek. Mezi požadavky, které si pro svou aplikaci vytyčil, bylo, aby studentům matematickou teorii předkládala v podobě textových kroků, mezi nimiž budou muset odpovídat na otázky, a dále aby umožňovala naučenou látku procvičovat a získané znalosti ověřovat v testech. To vše navíc zpestřené gamifikačními[1] prvky (Maloušek, 2020).

[1] Gamifikace znamená použití herních prvků v mimoherním prostředí (Fiala, 2019).

Aplikace Mathman dokonce natolik zaujala porotu soutěže projektů začínajících podnikatelů, T-Mobile Rozjezdy, že se dočkala postupu do celostátního finále (VUT, 2020).

V době, kdy jsem si téma své bakalářské práce vybírala, samozřejmě aplikace Mathman neexistovala, ale považuji za důležité ji zmínit, protože jde o velmi zdařilý projekt, který s výjimkou zaměření na mobilní zařízení měl takřka stejné cíle, jaké jsem si pro svou aplikaci stanovila i já sama. Mathman je kvalitativně na velmi vysoké úrovni a k dnešnímu dni si tuto aplikaci z Google Play nainstalovalo více než 50 000 uživatelů[2].

[2] Aplikace Mathman v Google Play: https://play.google.com/store/apps/details?id=cz.musestudio.mathexplained&hl=cs&gl=US

3 Metodika

V kontextu vývoje softwaru pojem metodika označuje soubor komplexních postupů a návodů pro vývoj softwarové aplikace ve všech etapách řešení, tedy napříč všemi fázemi životního cyklu softwaru (Kadlec, 2004).

Metodik existuje celá řada a typicky se řadí do jedné ze dvou skupin: tradiční nebo (dnes stále častěji preferované) agilní. Pro tradiční metodiky je charakteristické stanovení fixního rozsahu projektu – na začátku proběhne analýza, a po odsouhlasení klientem už ke změnám nedochází, v důsledku čehož výsledný produkt často utrpí (Kodytek, b.r.). Do této skupiny patří například vodopádový model a na něj navazující spirálový model.

V případě agilních metodik je rozsahu projektu dopřána flexibilita tím, že se fixně stanoví rozpočet a termín, což vede k lepším možnostem adaptace vůči změnám vnějších faktorů v průběhu procesu vývoje. Potenciální nevýhodou agilních metodik je jisté množství chaosu způsobovaného prováděnými změnami a také požadavky na vyšší míru zapojení klienta do vývojového procesu (Kodytek, b.r.).

Metodika vhodně zvolená pro daný projekt je ve vývojářských týmech velmi cenným nástrojem, protože celému vývojovému procesu dává řád a sjednocuje komunikaci. Já se ale při vývoji aplikace, která je středobodem této bakalářské práce, dostala do pozice, kdy jsem zastávala všechny role najednou – stala jsem se návrhářem i koncovým uživatelem, programátorem i testerem a dodavatelem i zákazníkem, což s sebou přineslo i jisté neočekávané výzvy.

3.1 Souhrn použitých nástrojů

Svou aplikaci Nezmatematika jsem se rozhodla vyvíjet v programovacím jazyce C# (verze 7.3 – zvolena automaticky vůči cílové architektuře), s cílovou architekturou .NET Framework (verze 4.7.2). K napsání aplikace jsem použila software Visual Studio Community 2019 od společnosti Microsoft. Uživatelské rozhraní je řešeno s využitím WPF (Windows Presentation Foundation) rovněž od firmy Microsoft a značkovacího jazyka XAML (Extensible Application Markup Language). Ve všech situacích, kde uživatelé mohou pracovat s výběrem barev, jsem použila třídu `ColorPicker` z NuGet balíčku Extended.Wpf.Toolkit od autora Xceed.

4 Implementace kódu

4.1 Framework .NET

Framework .NET byl oficiálně uveden v roce 2002. Programovacímu prostředí společnosti Microsoft do té doby dominovaly klasické programovací jazyky jako například Visual Basic a C++ a komponentový model COM (Component Object Model), který společnost představila v roce 1993. COM měl několik nežádoucích aspektů – zejména používání registru systému Windows, které mohlo způsobovat nedostupnost komponent za běhu, pokud došlo ke změně nebo poškození fragmentu registru. Prvotní koncepce nového modelu, který se měl stát náhradou COM, začala vznikat již v roce 1995. První beta verze byla zpřístupněna v roce 2000, uvedení plné verze pak následovalo 13. února 2002. Od té doby jsou nové verze .NET pravidelně vydávány společně s novými verzemi Visual Studio (Posadas, 2016).

4.2 WPF

WPF (Windows Presentation Foundation) je moderní systém grafického zobrazování pro Windows, jehož zásadním přínosem je nezávislost na rozlišení displeje.

Oproti svému předchůdci, WinForms (Windows Forms), má WPF zejména tyto výhody:

- Rozložení jednotlivých prvků je flexibilní – nepotřebují pevně definované souřadnice ani velikosti a mohou se přizpůsobovat jak svému obsahu, tak proměnlivé velikosti okna, což činí uživatelské rozhraní responsivním.
- Má rozsáhlou škálu možností stylování textů v jakékoliv části uživatelského rozhraní včetně obtékání objektů, rozdělení do sloupců a zarovnání.
- Podporuje animace a nepotřebuje k překreslení zobrazených objektů časovač.
- Podporuje multimediální obsah – umožňuje přehrávání jakýchkoliv audio nahrávek i videí, které podporuje Windows Media Player.
- Vzhled prvků může být upraven pomocí stylů a šablon, což umožňuje snadné provádění změn a současné udržení jednolitého vzhledu napříč celou aplikací.
- Příkazy (Commands) mohou být díky abstrakci definovány na jednom místě a používány opakovaně napříč aplikací.
- XAML kód uživatelského rozhraní může být kompletně oddělen od kódu aplikační logiky a lze jej tedy i zcela nezávisle upravovat (MacDonald, 2013).

4.3 XAML

XAML (eXtensible Application Markup Language) je značkovací jazyk používaný k tvorbě instancí .NET objektů. Používá se k definování rozložení objektů v uživatelských rozhraních. XAML dokument tedy určuje uspořádání všech panelů, tlačítek a dalších prvků, z nichž se uživatelské rozhraní celé aplikace skládá.

Typicky se XAML kód nepíše ručně, ale je částečně vygenerován některým softwarovým nástrojem (např. Visual Studio či Microsoft Expression Blend). Návrháři uživatelského rozhraní tak mohou pracovat nezávisle na vývojářích píšících aplikační logiku a naopak, což bylo ostatně také hlavním důvodem, proč se Microsoft rozhodnul XAML vytvořit (MacDonald, 2013).

4.4 MVVM

Model-View-ViewModel (MVVM) je vzor softwarové architektury, jehož podstata spočívá v oddělení zodpovědností (Separation of Concerns – SoC) mezi business modelem, uživatelským rozhraním a business logikou. Při použití tohoto vzoru je každá struktura vyvíjené aplikace jednoznačně zařazena do jedné ze tří vrstev – Model, ViewModel nebo View, přičemž platí, že komponenty z vrstvy Model nikdy s komponentami z vrstvy View nekomunikují přímo. Komunikace mezi nimi probíhá prostřednictvím komponent z vrstvy ViewModel (Yuen, 2020).

4.4.1 Model

Vrstva Model uchovává data a náleží do ní i veškerá validační logika spjatá s datovým modelem. Obsahuje třídy, jejichž atributy obvykle odpovídají sloupcům v databázové tabulce či tabulkách, kam se daná data ukládají. Tato vrstva je zcela oddělena od vrstvy View (Yuen, 2020).

4.4.2 ViewModel

Komponenty ve vrstvě ViewModel obousměrně komunikují jak s komponentami z vrstvy Model, tak s komponentami z vrstvy View. Forma předávaných dat je často nějakým způsobem upravena pro jejich snazší zobrazení, např. s využitím implementací rozhraní `IValueConverter` (Yuen, 2020).

4.4.3 View

Komponentami ve vrstvě View je definováno uživatelské rozhraní. Typicky mají ve vlastnosti `DataContext` přiřazenou komponentu z vrstvy ViewModel a zobrazují data

touto komponentou poskytnutá – touto cestou je řešeno i vázání příkazů na konkrétní prvky uživatelského rozhraní (Yuen, 2020).

4.5 XML

XML (eXtensible Markup Language) je jednoduchý a flexibilní textový formát, který vyvinulo a standardizovalo konsorcium W3C pro potřeby rozsáhlého elektronického publikování. Dnes hraje významnou roli při předávání nejrůznějších dat ve webových i jiných prostředích (W3C, b.r.).

XML zobecňuje značkovací jazyk SGML, z nějž vychází. Na rozdíl od HTML nemá pevně stanované tagy, což činí možnosti jeho uplatnění takřka neomezenými. Takto formátovaný dokument popisuje výhradně logickou strukturu a může být vytvořen, přečten či upravován jakýmkoliv počítačovým systémem práce s textovými daty (Prorok, 2010).

Při vývoji své aplikace jsem se rozhodla formát XML použít pro uchovávání dat v ukládaných souborech – od nastavení, přes uživatele a data o jejich aktivitách až po kurzy.

4.6 RTF

Formát RTF (Rich Text Format) vyvinutý společností Microsoft představuje způsob kódování formátovaného textu a grafických prvků pro použití uvnitř aplikací napříč různými operačními systémy (Iqbal, 2019). Ve své aplikaci jsem jej využila k ukládání vytvářených příkladů proto, že jde o defaultní formát pro ukládání obsahu elementů `RichTextBox`. Můj editor kurzů v módu pro vyučující rovněž umožňuje příklady přímo importovat a exportovat právě jako soubory s příponou .rtf.

5 Návrh aplikace Nezmatematika

5.1 Základní charakteristika navrhovaného systému

Jak jsem zmínila již v úvodu, za zrodem mojí aplikace stálo několik požadavků vycházejících z mých osobních zkušeností s doučováním matematiky. Šlo mi především o možnost vytvářet své vlastní úlohy a umožnit studentům zobrazit si následující krok postupu k vyřešení úlohy (ale nezobrazovat celé řešení rovnou).

Už od počátku jsem pro aplikaci zamýšlela fungování ve dvou módech – mód pro vyučující, který nabídne interní editor umožňující vytváření a editaci sad úloh, a mód pro studenty, v němž pak bude možné řešení úloh procvičovat, vracet se k již vyřešeným úlohám a sledovat své pokroky.

5.2 Terminologický slovník

Názvosloví spjaté s celou aplikací Nezmatematika začalo vznikat již v prvotních rozhovorech o nápadu na vytvoření této aplikace, ale jeho vývoj dynamicky pokračoval po celou dobu, kdy jsem na aplikaci pracovala. Zde je jeho výsledná podoba:

Tabulka č. 1: Terminologický slovník

Pojem	**Vysvětlení**
Uživatel	Kdokoliv, kdo aplikaci používá.
Typ uživatele	Role, s níž uživatel do aplikace přistupuje (student nebo vyučující).
Mód - Student - Vyučující	Způsob spuštění aplikace (rozhoduje o tom, jaký obsah bude uživateli dostupný) - umožňuje procházet kurzy a procvičovat řešení úloh - umožňuje vytvářet a editovat kurzy
Student	Uživatel, který aplikaci používá pro procvičování úloh (v módu Student).
Vyučující	Uživatel, který aplikaci používá k vytváření sad úloh (v módu Vyučující).
Profil uživatele	Soubor uložených údajů o uživateli bez ohledu na jeho roli
Úloha	V editoru vytvořený objekt reprezentující matematický příklad k vyřešení, který se skládá ze zadání, otázky a sady správných odpovědí. Volitelně může obsahovat sadu kroků k vyřešení.

Zadání	Text obsahující všechny informace potřebné k vyřešení úlohy
Otázka	Text, který definuje, co by měl student zadat jako odpověď. Může obsahovat také instrukce týkající se požadovaného formátu odpovědi (např. počet desetinných míst).
Krok	Dílčí část postupu řešení, kterou si student může (ale nemusí) zobrazit.
Kurz	Sada úloh
Editor	Prostředí umožňující tvorbu kurzů a editaci úloh, z nichž jsou složeny, které je dostupné pouze v módu Vyučující.
Režim editoru - uživatelský - kódový	Jeden ze dvou režimů zobrazení editoru - umožňuje přímou editaci úloh a všech jejich součástí - neumožňuje editaci a pouze zobrazuje obsah .rtf souboru, v němž bude úloha uložena

Zdroj: vlastní zpracování (2019)

5.3 Funkční požadavky

Funkční požadavky jsem definovala zvlášť pro mód Student (tyto požadavky jsou značeny „fs“) a zvlášť pro mód Vyučující (značeny „fv“). V následujících tabulkách jsou seřazeny podle priority.

5.3.1 Mód Student

Tabulka č. 2: Funkční požadavky módu Student

Číslo	**Požadavek**	**Popis**
fs1	Vyhodnocení odpovědi	Když student zadá odpověď, zobrazí se vyhodnocení, zda byla odpověď správná.
fs2	Prozrazení správné odpovědi	Bude-li odpověď, kterou student zadal, nesprávná, zobrazí se seznam správných odpovědí.
fs3	Ukládání pokroku v kurzu	Po každém odeslání odpovědi dojde k uložení pokroku, aby student mohl pokračovat tam, kde skončil, i v případě nestandardního ukončení aplikace.
fs4	Zobrazitelné nápovědy	Student si bude moci kroky k řešení postupně zobrazovat formou nápověd

fs5	Import kurzů	Aplikace bude studentům umožňovat import kurzů jako složek komprimovaných metodou ZIP.
fs6	Prohlížení dokončených kurzů	Student si bude moci znovu zobrazit kurzy, které již dokončil.
fs7	Ukládání historie procvičování	Každá odpověď bude zaznamenána, a při opětovném procházení již dokončených kurzů se studentovi zobrazí, jak na otázku odpověděl.
fs8	Opakování, dokud to nebude dobře	Úlohy, na které student odpověděl nesprávně, se po vyčerpání ostatních příkladů kurzu zobrazí znovu (a budou se opakovat tak dlouho, dokud na ně student neodpoví správně).
fs9	Nastavitelnost opakování úloh	Student si bude moci nastavit, aby se chybně zodpovězené úlohy neopakovaly.
fs10	Statistiky	Student si bude moci zobrazit statistiky svého používání aplikace – počet vyřešených úloh, počet úloh vyřešených bez nápověd, …

Zdroj: Vlastní zpracování (2019)

5.3.2 Mód Vyučující

Tabulka č. 3: Funkční požadavky módu Vyučující

Číslo	**Požadavek**	**Popis**
fv1	Tvorba kurzů	Vyučující bude moci vytvářet nové kurzy.
fv2	Editace kurzů	Vyučující bude moci editovat dříve vytvořené kurzy.
fv3	Automatické ukládání	Editor kurzů bude při přepínání mezi jednotlivými úlohami kurzu automaticky ukládat změny (bude možno vypnout v nastavení a ukládat pouze explicitním kliknutím na tlačítko Uložit).
fv4	Správné odpovědi	Editor kurzů umožní ke každé úloze přiřadit jednu či více správných odpovědí. Bez alespoň jedné správné odpovědi u každé úlohy nebude možné kurz zveřejnit.

fv5	Stylování textů úloh	Editor kurzů umožní upravovat formátování textů úloh – zejména písmo, jeho barvu, velikost a řez.
fv6	Zveřejňování kurzů	Vyučující se bude sám moci rozhodnout, kdy chce kurz zveřejnit studentům (tj. kurzy, které jsou teprve rozpracované, nebudou studentům dostupné).
fv7	Kroky postupu	Editor kurzů umožní ke každé úloze přiřadit jeden či více kroků postupu k vyřešení úlohy (nepovinná položka).
fv8	Export kurzů	Zveřejněné kurzy bude možné exportovat ve formě složek komprimovaných metodou ZIP pro snadnou distribuci kurzů studentům.
fv9	Export úloh	Editor kurzů umožní export jednotlivých úloh coby souborů ve formátu RTF. Exportované úlohy budou nezávislé na kurzu.
fv10	Import úloh	Editor kurzů umožní do editovaného kurzu jednotlivě importovat úlohy ze souborů ve formátu RTF.
fv11	Verzování kurzů	Zveřejněním kurzu, který byl již někdy dříve zveřejněn, proběhne archivace předchozí verze. Studenti, kteří předchozí verzi kurzu zahájili, se k ní budou moci i nadále vracet, ale archivovaná verze se již nebude zobrazovat mezi novými kurzy k zahájení.
fv12	Nastavitelné výchozí formáty	Aplikace vyučujícímu umožní nastavit výchozí styl písma používaný editorem kurzů.
fv13	Statistiky	Vyučující si bude moci zobrazit statistiky svého používání aplikace – počet vytvořených a zveřejněných kurzů, …

Zdroj: Vlastní zpracování (2019)

5.4 Nefunkční požadavky

Tabulka č. 4: Nefunkční požadavky

Číslo	**Požadavek**	**Popis**
n1	Fungování off-line	Aplikace bude fungovat kompletně off-line.

n2	Jednoduché uživatelské rozhraní	Uživatelské rozhraní bude navrženo tak, aby bylo co nejsrozumitelnější a i nezkušený uživatel se v něm dokázal zorientovat.
n3	Responsivní design	Aplikace bude navržena tak, aby se dokázala přizpůsobit různým velikostem displeje.

Zdroj: Vlastní zpracování (2019)

5.5 Návrh uživatelského rozhraní

5.5.1 Úvodní obrazovka s výběrem módu

Po spuštění aplikace se uživateli nejprve zobrazí úvodní obrazovka, jejímž jediným účelem je výběr módu (Student nebo Vyučující), ve kterém má aplikace po celou zbývající dobu pracovat. Grafická podoba úvodní obrazovky tvoří přílohu č. 1.

5.5.2 Založení a editace uživatelského profilu

Bez ohledu na zvolený mód bude pro správné fungování aplikace nutné mít vytvořený uživatelský profil. Při prvním spuštění v daném módu bude tedy uživatel nejprve vyzván k vytvoření svého uživatelského profilu. Do formuláře uživatelského profilu se budou vyplňovat identické údaje bez ohledu na to, o jaký typ uživatele se bude jednat – titul před jménem, jméno, příjmení, titul za jménem, škola a třída. Titul před a za jménem jsou nepovinné položky. Všechny údaje lze později editovat a profil je možné i zcela smazat.

Profil nebude mít žádné přihlašovací údaje, protože aplikace funguje kompletně off-line. Veškerá data jednotlivých uživatelských profilů aplikace Nezmatematika se budou ukládat na úrovni uživatelského profilu OS Windows, pod kterým byla vytvořena.

Aplikace bude podporovat přepínání mezi více profily stejného typu. Oddělené profily tedy mohou mít jak studenti (např. sourozenci používající PC pod stejným Windows uživatelem), tak i vyučující (kteří mohou využívat různé profily pro výuku v různých třídách apod.).

Grafická podoba obrazovky s formulářem pro editaci uživatelského profilu v módu Vyučující je příloha č. 2.

5.5.3 Hlavní menu

Hlavní menu bude v módu pro vyučující i v módu pro studenty uspořádáno velmi podobně. V oblasti levého horního rohu obrazovky bude v obou módech umístěno tlačítko pro přepínání mezi uživatelskými profily. V levém dolním rohu obrazovky se pak budou nacházet zbývající tlačítka. Pro oba módy budou společná tlačítka statistik,

nastavení, stručných informací o programu, návrat na výběr módu a ukončení celé aplikace.

V módu Student bude hlavní menu dále obsahovat tlačítka pro výběr nového kurzu k procvičení a pro výběr z kurzů již dříve zahájených (včetně těch dokončených).

V módu Vyučující bude v menu místo dvou výše zmíněných studentských tlačítek umístěno tlačítko k vytvoření nového kurzu a tlačítko pro výběr existujícího kurzu k editaci.

Tlačítka hlavního menu zůstanou viditelná, dokud uživatel některou z dalších voleb nepotvrdí přechod do jiné části aplikace. Ke skrytí hlavního menu tedy dojde až při přechodu na vybraný kurz (v módu Student) nebo do editoru kurzů (v módu Vyučující).

Příloha č. 2 zobrazuje hlavní menu v módu Vyučující při editaci uživatelského profilu.

5.5.4 Výběry kurzů

V aplikaci bude k výběru ze seznamu kurzů docházet v módu Student při návratu ke dříve započatým či dokonce již zcela dokončeným kurzům a v módu Vyučující při volbě existujícího kurzu pro úpravu v Editoru kurzů. Ve všech těchto případech zobrazovaný seznam umožní setřídění záznamů podle zobrazovaných sloupců a výběr bude potvrzován tlačítkem.

Při volbě nového kurzu (v módu Student) bude zobrazen název kurzu, číslo zveřejněné verze, počet úloh, jméno autora a datum zveřejnění. Výběr nedokončených kurzů studenta bude zobrazovat název kurzu, číslo verze, jméno autora, informaci o počtu již vyřešených úloh, kdy student kurz zahájil a kdy jej měl naposledy otevřený. V případě výběru z dokončených kurzů pak bude počet úloh vyřešených nahrazen počtem celkovým a datum posledního otevření kurzu datem dokončení. Výběr z dříve zahájených kurzů v módu Student zobrazuje příloha č. 3.

V režimu Vyučující postrádá třídění podle autora kurzu význam (vyučující může editovat pouze ty kurzy, které sám vytvořil). Proto bude seznam kurzů k editaci možno setřídit podle názvu, verze, počtu úloh, data posledního uložení změn a data vytvoření. Z této obrazovky bude také možné kurz odstranit.

5.5.5 Procházení kurzu v módu Student

Uživatelské rozhraní procházení kurzu v módu Student bude v horní části obrazovky obsahovat tlačítko pro návrat do hlavního menu, název aktuálně procházeného kurzu a

stručnou hlášku o tom, kolik příkladů kurz obsahuje. Níže bude zobrazeno zadání aktuální úlohy včetně otázky. Prostor pod nimi bude vyčleněn pro zobrazování případných nápověd (kroků postupu) – pokud je úloha obsahuje a pokud se student rozhodne si je zobrazit.

V dolní části obrazovky se nachází pole pro studentovu odpověď s tlačítkem pro její potvrzení. Kliknutím na něj dojde k vyhodnocení správnosti a zobrazení příslušné hlášky, případně správných odpovědí. Tlačítko pro zobrazení dalšího kroku postupu je umístěno vedle tlačítka pro potvrzení odpovědi, zobrazuje se vždy, ale je aktivní pouze pokud existuje dosud nezobrazený krok postupu, který bude po kliknutí na něj odhalen.

V nejspodnější části obrazovky lze najít tlačítka „předchozí“ a „další“ pro přepínání mezi aktuální úlohou a již vyřešenými úlohami téhož kurzu. Při správném zodpovězení poslední úlohy kurzu se objeví tlačítko pro dokončení kurzu.

Procházení již dokončených kurzů se od procházení kurzem poprvé odlišuje tím, že obsah textového pole s odpovědí nelze editovat – je v něm zobrazena odpověď, kterou student zadal, když kurzem procházel poprvé (stejně tak je zobrazena i hláška o správnosti dané odpovědi). Tlačítko pro zobrazení nápověd je skryté, ale zobrazují se ty kroky řešení, které si student před zadáním odpovědi zobrazil.

Příloha č. 4 zobrazuje příklad v nedokončeném kurzu, ke kterému se student vrátil kliknutím na tlačítko „předchozí“.

5.5.6 Editor kurzů v módu Vyučující

Uživatelské rozhraní editoru kurzů se člení na 3 hlavní části: panel nástrojů v horní části obrazovky, seznam úloh v kurzu na levé straně obrazovky a oblast aktuální úlohy, která vyplňuje zbývající plochu.

Tlačítka na panelu nástrojů jsou seskupena podle svých funkcí – tlačítka upravující celý kurz (přidání a odebrání úlohy, uložení, zveřejnění), tlačítka pro formátování textu v elementech `RichTextBox` (tučné písmo, kurzíva, podtržení, přeškrtnutí, barva písma, font, velikost písma), tlačítka pro vkládání speciálních symbolů (π), tlačítka pro import/export (import úlohy, export úlohy, export kurzu a tlačítko přepínající mezi uživatelským a kódovým režimem editoru).

V seznamu úloh je každá úloha označena číslem dle pořadí v kurzu a doplněna začátkem textu zadání a začátkem textu otázky. Seznam umožňuje kliknutím přepínat mezi jednotlivými úlohami kurzu.

Obsah oblasti aktuální úlohy se mění v závislosti na režimu editoru. V uživatelském režimu tato oblast obsahuje v horní části dva elementy `RichTextBox` (do nichž lze vkládat i obrázky) – jeden na zadání, jeden na otázku – a v dolní části dvě menší oblasti, obě složené z elementů `TextBox` a `ListView` pro zadávání správných odpovědí a kroků k řešení úlohy. V kódovém režimu se veškerý obsah uživatelského režimu skryje a místo něj se objeví jediný `RichTextBox`, který nelze editovat a ve kterém je zobrazen náhled RTF souboru aktuální úlohy.

Příloha č. 5 zobrazuje editor úloh v uživatelském režimu, příloha č. 6 pak zobrazuje tutéž úlohu v kódovém režimu.

5.5.7 Nastavení

Nastavení je rozděleno do tří sekcí – nastavení vzhledu (tato sekce je společná pro studenty i vyučující), nastavení pro studenty (dostupná pouze v módu Student) a nastavení pro vyučující (dostupná pouze v módu Vyučující). Ve spodní části obrazovky se nacházejí tlačítka pro obnovení výchozího nastavení, odchod beze změn a uložení změn.

Sekce pro studenty umožňuje vypnout zařazování chybně zodpovězených úloh na konec kurzu. Sekce pro vyučující umožňuje přizpůsobení editoru kurzů nastavením automatického ukládání a výchozích hodnot rozlišování velkých a malých písmen u odpovědí, fontu a velikosti písma.

Grafická podoba nastavení v režimu Vyučující tvoří přílohu č. 7.

5.6 Datový model

Již při předchozí analýze se začaly rýsovat entity, se kterými bude celá aplikace pracovat. Nejdůležitější z nich jsou `User` (uživatel), `Course` (kurz) a `MathProblem` (úloha). Pro entitu User jsem vytvořila čtyři další dílčí entity `UserBase` (základní údaje uživatele), `UserCourseData` (uživatelova data kurzů), `UserSettings` (nastavení) a `UserStats` (statistiky uživatele). Ve vrstvě model je celkem 8 entit zakreslených na následujícím diagramu.

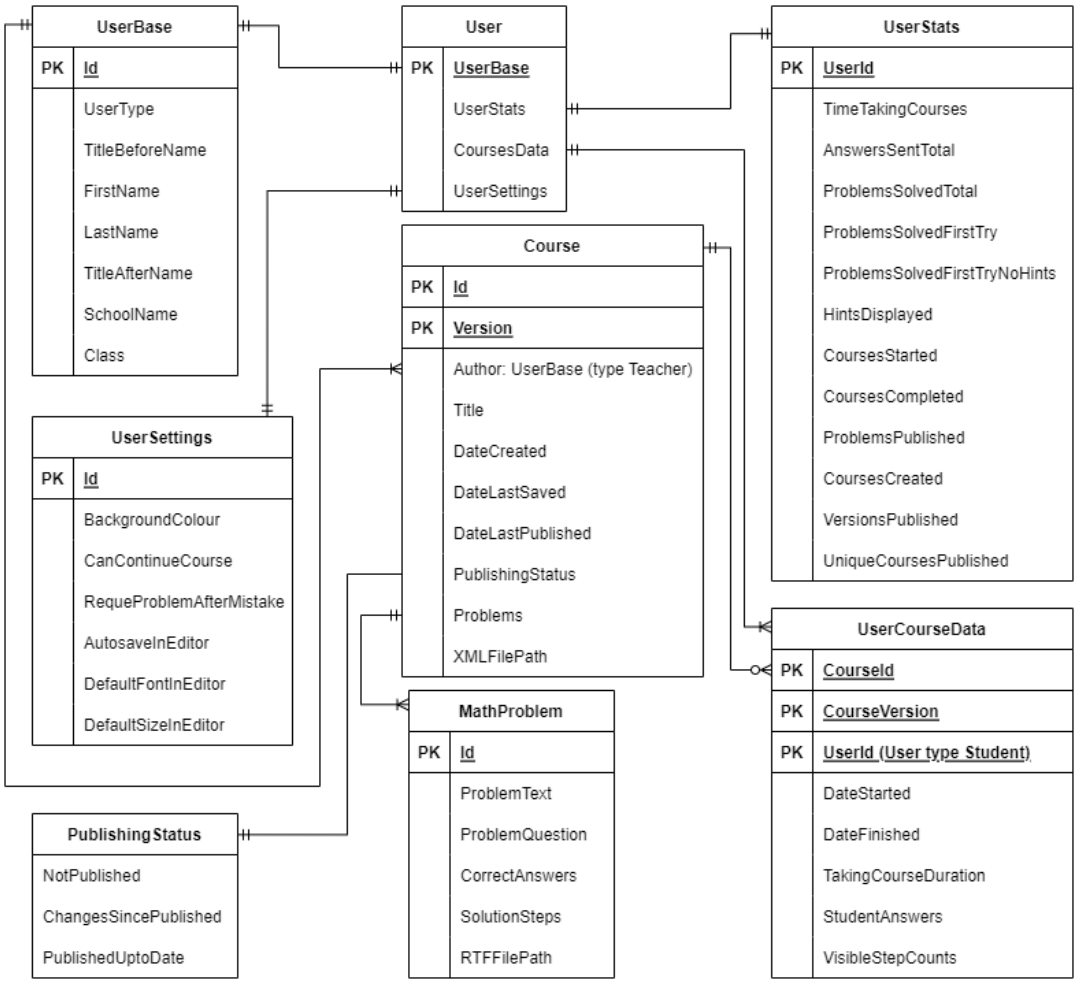
Každý uživatel má právě jeden soubor základních informací o uživateli, právě jeden soubor statistik, právě jeden soubor uživatelských nastavení a libovolné množství uživatelských dat o kurzu.

Každý kurz má právě jednoho autora (reprezentovaného třídou `UserBase`), nachází se v jednom ze stavů zveřejněnosti (nezveřejněn, obsahuje nezveřejněné změny či zveřejněn

v aktuální podobě) a obsahuje libovolný počet úloh (kurzy ve stavu nezveřejněn mohou obsahovat nula úloh). Konkrétní instance úlohy vždy patří do právě jednoho kurzu.

Uživatelská data o kurzu se vždy vztahují k právě jedné verzi právě jednoho zveřejněného kurzu a k právě jednomu uživateli, který jej absolvoval.

Obrázek č. 1: Entity relationship diagram aplikace Nezmatematika



Zdroj: Vlastní zpracování (2020)

6 Implementace aplikace Nezmatematika

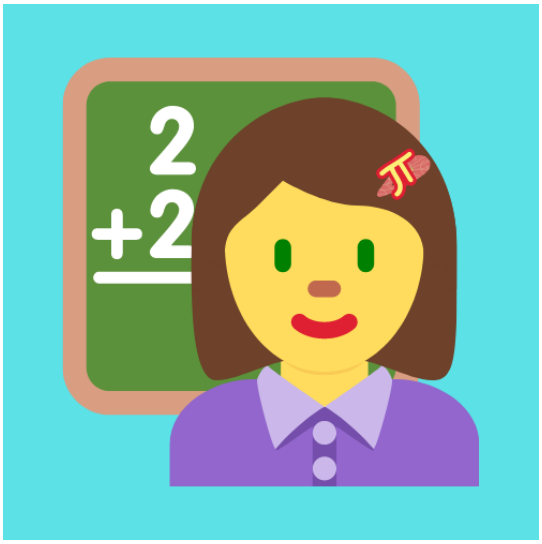
Řešení aplikace Nezmatematika obsahuje dva projekty. Prvním je samotná WPF aplikace `Nezmatematika` obsahující všechny soubory z vrstev Model, ViewModel (do níž patří také třídy typu Command, Helper a ValueConverter) i View (obsahující XAML soubory oken aplikace). Druhým projektem je její Setup Project, `NezmatematikaSetup`.

Následující podkapitoly obsahují podrobnější popis kódu celé aplikace včetně ukázek. Všechny zdrojové soubory, jakož i výslednou spustitelnou aplikaci, o níž tato práce pojednává, přikládám na datovém disku.

6.1 Logo

Logo aplikace jsem vytvořila z volně dostupných grafických prvků v nástroji Canva.com a zobrazuje usmívající se učitelku před tabulí s matematickým příkladem.

Obrázek č. 2: Logo aplikace Nezmatematika



Zdroj: Vlastní zpracování (2022)

6.2 Adresářová struktura aplikace Nezmatematika

Aplikace Nezmatematika nepracuje s databází, ale ukládá svá data do souborů, jejichž názvy začínají identifikátorem objektu, ke kterému patří. Identifikátor objektu je generován při vzniku objektu a dále se nemění.

Identifikátor kurzu v sobě obsahuje identifikátor autora, čímž je zajištěno, že žádný jiný vyučující nebude moci daný kurz editovat. Příklady ale na kurz pevně vázány nejsou, a mohou tedy být exportovány a do svých kurzů je následně mohou importovat i jiní vyučující.

Pro pojmenování XML a RTF souborů aplikace platí následující systém:

Tabulka č. 5: Systém pojmenování datových souborů aplikace Nezmatematika

Obsah souboru	**Název souboru**
defaultní nastavení	DefaultSettings.xml
seznam základních údajů všech uživatelů	UserList.xml
historie kurzů uživatele	{idUživatele}CoursesData.xml
nastavení uživatele	{idUživatele}Settings.xml
statistiky uživatele	{idUživatele}Stats.xml
kurz	{idKurzu}Course.xml
úloha	{idÚlohy}.rtf

Zdroj: Vlastní zpracování (2022)

Při instalaci si kromě uživatelem zvolené instalační složky aplikace Nezmatematika vytváří také složku \HopefulSparrow\Nezmatematika\ v systémovém adresáři pro data aplikací (`Environment.SpecialFolder.ApplicationData`), což je v případě operačního systému Windows 10 [Username]\AppData\Roaming.

Tabulka č. 6: Adresářová struktura souborů aplikace Nezmatematika

Obsah souboru	**Relativní cesta k umístění**
defaultní nastavení	./Users
seznam základních údajů všech uživatelů	./Users
historie kurzů uživatele	./Users
nastavení uživatele	./Users
statistiky uživatele	./Users
nezveřejněný kurz	./Courses/{idAutora}
úloha v nezveřejněném kurzu	./Courses/{idAutora}/{idKurzu}
zveřejněný kurz	./Courses/Published
úloha ve zveřejněném kurzu	./Courses/Published/{idKurzu}_{verzeKurzu}
archivovaný kurz	./Courses/Archived
úloha v archivovaném kurzu	./Courses/Archived/{idKurzu}_{verzeKurzu}

Zdroj: Vlastní zpracování (2022)

6.3 Singletony

Podstatou návrhového vzoru singleton je skutečnost, že pro danou třídu může vždy existovat pouze jedna instance současně. Aplikace Nezmatematika tento návrhový vzor využívá k uchování dvou významných informací. První z nich, `AppMode`, pomocí výčtového typu říká, v jakém módu je aplikace spuštěna (`Student`, `Teacher` nebo `Unselected`, pokud je uživatel právě na obrazovce s výběrem módu).

Druhý singleton, `WhereInApp`, je rovněž výčtového typu, a ukládá informaci na jaké z významných obrazovek se zrovna uživatel nachází (`ModeSelection`, `MainMenu`, `CourseForStudent`, `CourseEditor`, `Statistics`, `Settings`) – např. sekce „O programu“ není nijak svázána s daty uchovávanými vrstvou model a pro účely vyhodnocování `WhereInApp` je tedy považována za součást hlavního menu (`MainMenu`).

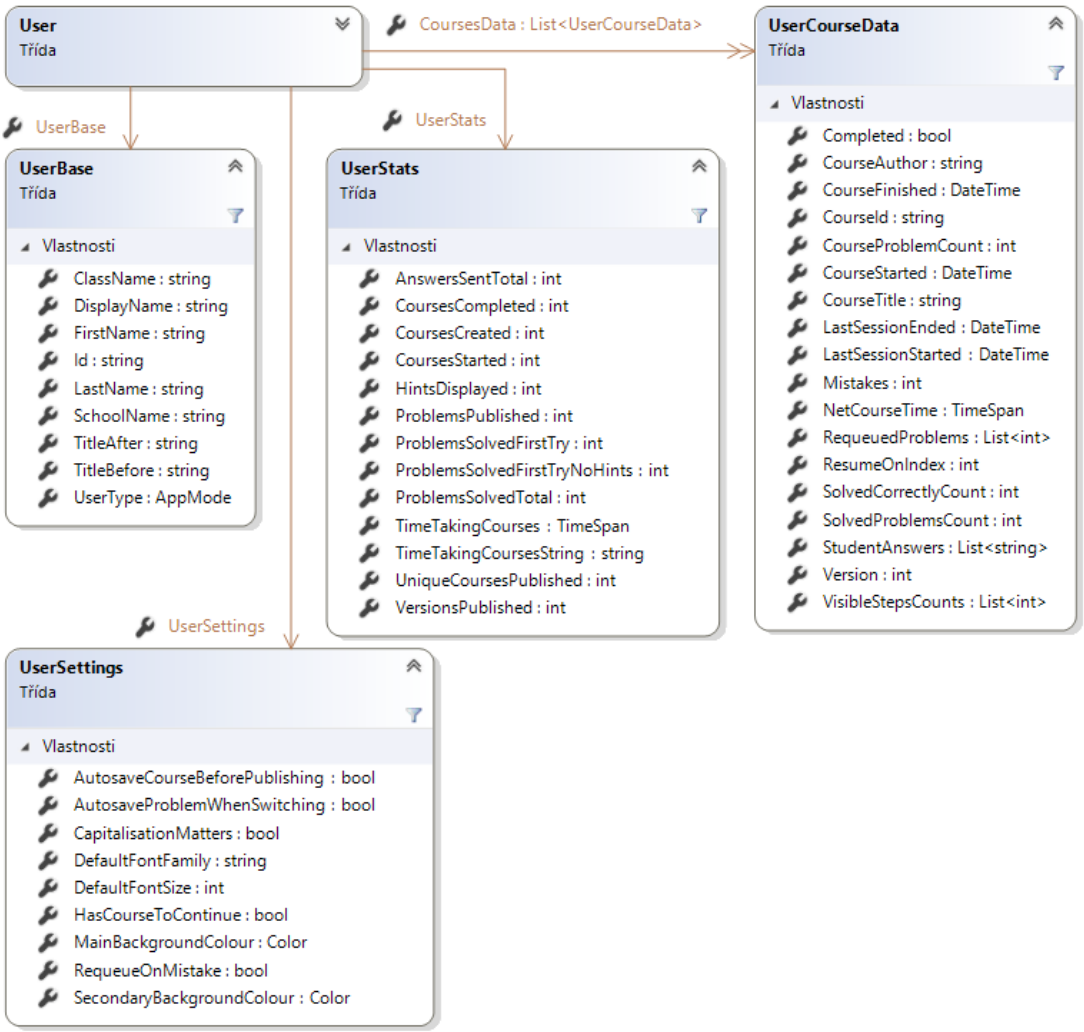
6.4 Vrstva Model

6.4.1 User (uživatel)

Třída `User` sdružuje všechny informace spjaté s uživatelem a pro větší přehlednost má svá data dále seskupena ve třídách `UserBase`, `UserSettings`, `UserStats` a kolekci typu `UserCourseData`. V `UserBase` jsou uchovávány základní údaje o uživateli – typ uživatele (student nebo vyučující), identifikátor vygenerovaný v okamžiku založení uživatelského profilu, jméno a příjmení včetně titulů, škola a třída a zobrazovací jméno používané ve výběrových seznamech. `UserSettings` obsahuje veškerá uživatelská nastavení (vzhledová i funkční). V `UserStats` jsou sdruženy veškeré údaje zobrazované ve statistikách (počet zahájených a dokončených kurzů, počet vyřešených úloh, …).

Vlastnost `UserCoursesData` drží kolekci jednotlivých `UserCourseData`, v nichž jsou v případě studenta zaznamenány veškeré jeho interakce s konkrétním kurzem – od data zahájení a informace, u kterého příkladu student skončil, přes údaje, které se později propisují do statistik, jako je například čistý čas strávený v kurzu, až po historii zobrazování nápověd a zadaných odpovědí.

Obrázek č. 3: Diagram tříd – User



Zdroj: Vlastní zpracování (2022)

6.4.2 MathProblem (úloha)

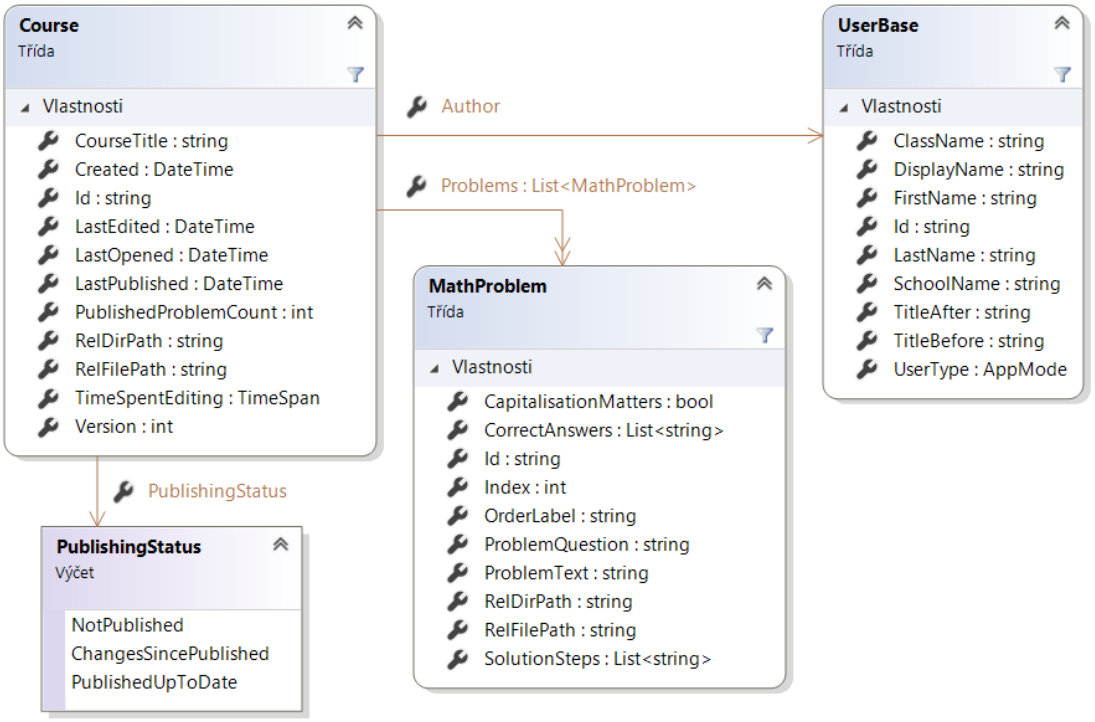
Třída `MathProblem` ukládá identifikátor úlohy, údaje využívané pro zobrazení na seznamu úloh v editoru kurzů (pořadové číslo, začátek textu zadání a otázky), relativní cesty (adresářovou i včetně názvu souboru) k RTF souboru obsahujícímu formátované zadání a otázku dané úlohy a kolekce s kroky postupu, které si student bude moci zobrazovat, a správnými odpověďmi pro vyhodnocování.

V módu Vyučující je možné v rámci editoru kurzů provádět export a import úloh. Tato funkcionalita nepracuje přímo se třídou `MathProblem`, ale s RTF souborem k úloze přidruženým, jehož prostřednictvím lze všechny klíčové údaje úlohy přenést. Exportovaný RTF soubor úlohy si může kterýkoliv vyučující importovat do libovolného ze svých kurzů.

6.4.3 Course (kurz)

Třída `Course` obsahuje jednoznačné identifikační údaje (identifikátor a verze), adresy ukládaných souborů kurzu (relativní cestu k adresáři se soubory příkladů a relativní cestu k XML souboru celého kurzu), kolekci úloh, údaje o autorovi (reprezentované třídou `UserBase`), údaje zobrazované autorovi ve výběrovém seznamu (název, datum vytvoření, datum poslední editace), údaje o stavu zveřejnění kurzu a pomocné údaje pro statistiky.

Obrázek č. 4: Diagram tříd – Course a MathProblem



Zdroj: Vlastní zpracování (2022)

Kurz může být v rámci aplikace uložen na třech různých místech (zmíněných v kapitole 6.2 Adresářová struktura). V každé z těchto situací mohou mít soubory kurzu stejný obsah, ale různá umístění jsou spojena s různými oprávněními. Nezveřejněný kurz aplikace zpřístupní pouze jeho autorovi, který jej může libovolně editovat, zveřejnit či odstranit.

Když autor kurz zveřejní, dojde ke zvýšení čísla verze kurzu a vytvoření jeho kopie do adresáře zveřejněných kurzů, odkud jej aplikace zpřístupňuje studentům (nabízí kurz mezi novými i již zahájenými, v nichž lze pokračovat). Soubory kurzu zároveň zůstávají v autorově adresáři, odkud mohou být dále editovány nezávisle na zveřejněné verzi.

Třetím místem, kde mohou být soubory kurzu uloženy, je archivační adresář. K archivaci dříve zveřejněného kurzu dochází ve dvou situacích – když autor kurz odstraňuje a když autor zveřejňuje novou verzi takového kurzu. Archivované verze kurzů nadále zůstávají přístupné těm studentům, kteří je v minulosti zahájili, ale nezobrazují se mezi novými kurzy k zahájení. Student může projít více verzí jednoho kurzu, ale v nabídce nových kurzů k zahájení se zobrazuje vždy pouze poslední zveřejněná (nebo žádná, pokud poslední zveřejněnou verzi daného kurzu student již zahájil).

Jak již bylo zmíněno dříve, kurzy mohou být exportovány a importovány. Export kurzu může provést pouze jeho autor, a to prostřednictvím editoru kurzů. Výstupem bude složka komprimovaná metodou ZIP uložená do uživatelem zvoleného adresáře. Smyslem této funkcionality je možnost předávat studentům hotové kurzy v kompaktním formátu, a proto je vždy exportována poslední zveřejněná verze kurzu.

Import kurzu je možno provést pouze v módu Student. Po kliknutí na tlačítko Importovat kurz se uživateli zobrazí dialogové okno pro výběr složky komprimované metodou ZIP, jejíž obsah aplikace extrahuje do adresáře zveřejněných kurzů.

6.5 Vrstva View (uživatelské prostředí)

Uživatelské prostředí aplikace Nezmatematika kromě automaticky vygenerované třídy `App` obsahuje dvě okna pojmenovaná `UserTypeSelectWindow` a `MainMenuWindow`. `UserTypeSelectWindow` slouží výhradně k vybrání v jakém módu má být aplikace spuštěna a `MainMenuWindow` je následně používáno po celou zbývající dobu běhu aplikace.

Komunikaci mezi uživatelským rozhraním a vrstvou ViewModel zajišťuje DataBinding. Tyto datové vazby předávají požadavky uživatele, aby mohly být dále zpracovány aplikační logikou. V ukázce kódu č. 1 je zobrazeno napojení prvku `ListView` na `ObservableCollection<UserCourseData> StudentInProgressCoursesData`, díky němuž se každý záznam v předávané kolekci zobrazuje formou požadovaných údajů v jednotlivých sloupcích. V případě sloupce s tagem `SolvedCorrectlyCount` se dokonce v jednom sloupci zobrazují dva dotažené údaje – počet správně vyřešených úloh a celkový počet úloh v kurzu oddělené lomítkem.

Ukázka kódu č. 1: DataBinding

```
<Label Content="Nedokončené kurzy:"
       Grid.Row="0" />
<ListView x:Name="lvStudentCoursesInProgress"
          Grid.Row="1"
          IsSynchronizedWithCurrentItem="True"
          ItemsSource="{Binding StudentInProgressCoursesData}"
          Visibility="{Binding StudentCoursesToContinueVis}"
          Thumb.DragDelta="Thumb_DragDelta">
    <ListView.View>
        <GridView>
            <GridViewColumn local:GridColumn.MinWidth="55"
                            Width="Auto"
                            DisplayMemberBinding="{Binding CourseTitle}">
                <GridViewColumnHeader Tag="CourseTitle"... />
            </GridViewColumn>
            <GridViewColumn local:GridColumn.MinWidth="35"...>
            <GridViewColumn local:GridColumn.MinWidth="55"...>
            <GridViewColumn local:GridColumn.MinWidth="105"
                            Width="Auto">
                <GridViewColumnHeader Tag="SolvedCorrectlyCount"... />
                <GridViewColumn.CellTemplate>
                    <DataTemplate>
                        <TextBlock>
                            <TextBlock.Text>
                                <MultiBinding StringFormat="{}{0}/{1}">
                                    <Binding Path="SolvedCorrectlyCount" />
                                    <Binding Path="CourseProblemCount" />
                                </MultiBinding>
                            </TextBlock.Text>
                        </TextBlock>
                    </DataTemplate>
                </GridViewColumn.CellTemplate>
            </GridViewColumn>
            <GridViewColumn local:GridColumn.MinWidth="130"...>
            <GridViewColumn local:GridColumn.MinWidth="130"...>
        </GridView>
    </ListView.View>
</ListView>
```

Zdroj: Vlastní zpracování (2022)

6.6 Vrstva ViewModel

Aplikace Nezmatematika má ve vrstvě ViewModel dvě třídy korespondující s okny aplikace – `UserTypeSelectVM` a `MainMenuVM` (pouze `MainMenuVM` implementuje rozhraní `INotifyPropertyChanged`), 33 tříd implementujících rozhraní `ICommand`, čtyři pomocné třídy Helper a dvě třídy implementující rozhraní `IValueConverter`.

6.6.1 INotifyPropertyChanged

Rozhraní `INotifyPropertyChanged` slouží k udržování aktuálního stavu mezi objekty uživatelského rozhraní (vrstvou View) a objekty s nimi spojenými vazbou DataBinding. Implementujeme jej deklarováním události `PropertyChanged` a metody `OnPropertyChanged(string)`, kterou budeme volat pro aktualizování zobrazené

hodnoty. Implementaci rozhraní i příklad volání metody `OnPropertyChanged(string)` zobrazuje ukázka kódu č. 2.

Ukázka kódu č. 2: InotifyPropertyChanged

```
private Course currentCourse;
Počet odkazů: 38
public Course CurrentCourse
{
    get { return currentCourse; }
    set
    {
        currentCourse = value;
        ReloadTempMathProblems();
        OnPropertyChanged("CurrentCourse");
    }
}

public event PropertyChangedEventHandler PropertyChanged;
Počet odkazů: 62
private void OnPropertyChanged(string propertyName)
{
    PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));
}
```

Zdroj: Vlastní zpracování (2022)

V aplikaci Nezmatematika je rozhraní `INotifyPropertyChanged` implementováno pouze třídou `MainMenuVM`, která slouží jako `DataContext` pro View `MainMenuWindow`.

6.6.2 ICommand

Rozhraní `ICommand` nachází uplatnění u tříd sloužících pro provádění akcí závislých na chování objektů uživatelského rozhraní (nejčastěji prvek `Button`). Implementace tohoto rozhraní vyžaduje metody `CanExecute(Object)` – ta vyhodnocuje, zda může být akce v aktuální situaci provedena – a `Execute(Object)` – ta případně akci provádí. Rovněž je pro implementaci povinná událost `CanExecuteChanged` vyvolávaná při změnách ovlivňujících spustitelnost daného příkazu.

Implementaci rozhraní `ICommand` zobrazuje ukázka kódu č. 3.

Ukázka kódu č. 3: ICommand

```
Počet odkazů: 3
public class AddNewProblemCommand : ICommand
{
    Počet odkazů: 8
    public MainMenuVM MMVM { get; set; }

    public event EventHandler CanExecuteChanged
    {
        add { CommandManager.RequerySuggested += value; }
        remove { CommandManager.RequerySuggested -= value; }
    }

    Počet odkazů: 1
    public AddNewProblemCommand(MainMenuVM vm)
    {
        MMVM = vm;
    }

    Počet odkazů: 0
    public bool CanExecute(object parameter)
    {
        return App.WhereInApp == WhereInApp.CourseEditor && MMVM.CurrentCourse != null;
    }

    Počet odkazů: 0
    public void Execute(object parameter)
    {
        var index = MMVM.CurrentMathProblem.Index;
        MMVM.CurrentCourse.AddNewMathProblem(MMVM.CurrentSettings.CapitalisationMatters);
        MMVM.PopulateTempMathProblemsFromCurrentCourse();
        MMVM.CurrentMathProblem = MMVM.CurrentCourse.Problems[index];
    }
}
```

Zdroj: Vlastní zpracování (2022)

V aplikaci Nezmatematika je toto rozhraní implementováno celkem 33 různými třídami, takže zde účel každé z nich popíši jen velmi stručně.

`ICommand` třídy přepínající mezi obrazovkami:

- `DisplayProfileSelectionCommand` – Přepne aplikaci na obrazovku výběru uživatelského profilu. Nelze provést, pokud není vybrán žádný uživatelský profil nebo pokud je obrazovka výběru uživatelského profilu již zobrazena.
- `DisplayNewCourseCreationCommand` – Přepne aplikaci na obrazovku založení nového kurzu. Nelze provést, pokud aplikace není v módu Vyučující nebo pokud není vybrán žádný uživatelský profil.
- `DisplayCourseToEditSelectionCommand` – Přepne aplikaci na obrazovku s výběrem kurzů k editaci. Nelze provést, pokud aplikace není v módu Vyučující,

pokud není vybrán žádný uživatelský profil nebo pokud uživatel nemá žádné kurzy, které by mohl editovat.

- `EditCourseCommand` – Přepne aplikaci na obrazovku do editoru kurzů a zahájí editaci zvoleného kurzu. Nelze provést, pokud aplikace není v módu Vyučující, pokud není vybrán žádný uživatelský profil nebo pokud není vybrán žádný kurz.
- `DisplayNewCourseSelectionCommand` – Přepne aplikaci na obrazovku výběru nového kurzu. Nelze provést, pokud aplikace není v módu Student nebo pokud není vybrán žádný uživatelský profil.
- `DisplayStudentCoursesToContinue` – Přepne aplikaci na obrazovku s výběrem kurzů, v jejichž procházení může student pokračovat. Nelze provést, pokud aplikace není v módu Student, pokud není vybrán žádný uživatelský profil nebo pokud student nemá žádné dříve zahájené kurzy.
- `OpenCourseForStudentCommand` – Přepne aplikaci na obrazovku studentského procházení kurzu. Nelze provést, pokud aplikace není v módu Student, pokud není vybrán žádný uživatelsky profil nebo pokud není vybrán žádný kurz.
- `DisplayStatisticsCommand` – Přepne aplikaci na obrazovku statistik. Nelze provést, pokud není vybrán žádný uživatelský profil.
- `DisplaySettingsCommand` – Přepne aplikaci na obrazovku nastavení. Nelze provést, pokud není vybrán žádný uživatelský profil.
- `DisplayAboutAppCommand` – Přepne aplikaci na obrazovku O programu. Nelze provést, pokud není vybrán žádný uživatelský profil.
- `BackToMainMenuFromAnywhereCommand` – Přepne aplikaci zpět na obrazovku hlavního menu.

`ICommand` třídy pracující s uživatelskými profily

- `CreateNewUserCommand` – Vytvoří nový uživatelský profil a nastaví jej jako aktuálního uživatele. Nelze provést, pokud nejsou vyplněny povinné údaje (jméno, příjmení, škola, třída).
- `PrepUserForEditingCommand` – Vyplní do editačního formuláře stávající základní údaje aktuálního uživatele. Nelze provést, pokud není vybrán žádný uživatelský profil.
- `EditUserCommand` – Uloží nové základní údaje pro aktuální uživatelský profil. Nelze provést, pokud není vybrán žádný uživatelský profil nebo pokud nejsou vyplněny povinné údaje (jméno, příjmení, škola, třída).

- `DeleteUserCommand` – Smaže aktuální uživatelský profil. Nelze provést, pokud je aktuální uživatelský profil jediným uživatelským profilem daného typu.
- `ApplyNewSettingsCommand` – Uloží nová nastavení pro aktuální uživatelský profil. Nelze provést mimo nastavení nebo pokud není vybrán uživatelský profil.
- `RestoreDefaultSettingsCommand` – Obnoví výchozí hodnoty nastavení pro aktuální uživatelský profil. Nelze provést, pokud není vybrán uživatelský profil.

`ICommand` třídy pro editaci kurzů:

- `CreateNewCourseCommand` – Vytvoří nový kurz a přepne obrazovku do editoru kurzů. Nelze provést, pokud aplikace není v módu Vyučující nebo je zadaný název kurzu prázdný.
- `DeleteCourseCommand` – Smaže aktuální kurz a případně archivuje jeho zveřejněnou verzi. Nelze provést, pokud aplikace není v módu Vyučující nebo není vybrán žádný kurz.
- `AddNewProblemCommand` – Do aktuálního kurzu přidá novou úlohu. Nelze provést mimo editor kurzů nebo pokud není vybrán žádný kurz.
- `RemoveProblemCommand` – Odstraní aktuální úlohu z aktuálního kurzu. Nelze provést mimo editor kurzů, pokud není vybraná žádná úloha nebo pokud je aktuální úloha v aktuálním kurzu jediná.
- `AddNewCorrectAnswerCommand` – K aktuální úloze přidá novou správnou odpověď. Nelze provést mimo editor kurzů, pokud není vybrána žádná úloh nebo pokud je nově přidávaná správná odpověď prázdná.
- `EditCorrectAnswerCommand` – Upraví aktuálně vybranou správnou odpověď aktuální úlohy. Nelze provést mimo editor kurzů, pokud není aktuálně vybraná žádná odpověď nebo pokud je nová hodnota odpovědi prázdná či shodná se stávající hodnotou.
- `RemoveCorrectAnswerCommand` – Odstraní vybranou správnou odpověď aktuální úlohy. Nelze provést mimo editor kurzů nebo pokud se hodnota aktuální odpovědi liší od hodnoty v editačním poli.
- `AddNewSolutionStepCommand` – K aktuální úloze přidá nový krok postupu řešení. Nelze provést mimo editor kurzů, pokud není vybrána žádná úloh nebo pokud je nově přidávaný krok postupu prázdný.
- `EditSolutionStepCommand` – Upraví aktuálně vybraný krok postupu řešení aktuální úlohy. Nelze provést mimo editor kurzů, pokud není vybrán žádný krok

postupu nebo pokud je nová hodnota kroku postupu prázdná či shodná se stávající hodnotou.

- `RemoveSolutionStepCommand` – Odstraní vybraný krok postupu z aktuální úlohy. Nelze provést mimo editor kurzů nebo pokud se hodnota aktuálního kroku postupu odpovědi liší od hodnoty v editačním poli.
- `SwitchBetweenUserAndCodeModeCommand` – Přepíná mezi editačním a kódovým režimem editoru kurzů. Nelze provést mimo editor kurzů.
- `PublishCourseCommand` – Provede kontrolu povinných parametrů kurzu a je-li vše v pořádku, zveřejní novou verzi aktuálního kurzu. Nelze provést mimo editor kurzů nebo pokud není vybrán žádný kurz.

`ICommand` třídy používané při studentském procházení kurzů:

- `MakeStepVisibleCommand` – Zobrazí další krok postupu řešení aktuální úlohy. Nelze provést mimo studentské procházení kurzu, pokud je aktuální úloha již vyřešena nebo pokud u aktuální úlohy již nezbývají žádné dosud nezobrazené kroky.
- `CheckIfAnswerIsCorrectCommand` – Zkontroluje správnost uživatelem zadané úlohy a zobrazí zpětnou vazbu. Nelze provést mimo studentské procházení kurzu nebo pokud je zadaná odpověď prázdná.
- `SwitchToNextProblemCommand` – Zobrazí další úlohu při studentském procházení kurzu. Nelze provést mimo studentské procházení kurzu, pokud není aktuální úlohy vyřešena nebo pokud aktuální kurz již další úlohu nemá.
- `SwitchToPreviousProblemCommand` – Zobrazí předcházející úlohu při studentském procházení kurzu. Nelze provést mimo studentské procházení kurzu nebo pokud je aktuální úloha první úlohou v kurzu.

Za povšimnutí jistě stojí, že v seznamu chybí `ICommand` implementující třída, jejímž účelem je ukládání ať už úloh či kurzů. Protože zcela klíčovou součástí jednotlivých úloh je obsah elementu `RichTextBox`, neřeším ukládání aktuální úlohy (i celého kurzu) implementací `ICommand`, ale v Code Behind `MainMenuView` a na tlačítko pro ukládání příslušnou metodu (kterou zobrazuje ukázka kódu č. 3) navazuji událostí `Click`.

Návrhový vzor MVVM klade velký důraz na princip oddělení zodpovědností a metoda `btnSaveCourse_Click` je v tomto ohledu bezesporu problematická. Yuen (2020) však zastává názor, že jsme-li si jisti, že žádný člen dané dvojice View + ViewModel

nebudeme nahrazovat jiným, můžeme si výjimku z tohoto pravidla dovolit, dává-li to v příslušném kontextu smysl.

Ukázka kódu č. 4: Ukládání kurzu v Code Behind

```
Počet odkazů: 3
private void btnSaveCourse_Click(object sender, RoutedEventArgs e)
{
    if (vM.CurrentCourse != null && vM.CurrentMathProblem != null)
    {
        vM.TeacherNotNullCurrentMathProblemAboutToChange -= ViewModelTeacher_CurrentMathProblemAboutToChange;
        UpdateCodeMode();

        var problem = new TextRange(rtbProblemText.Document.ContentStart, rtbProblemText.Document.ContentEnd);
        vM.CurrentMathProblem.ProblemText = TextRangeHelper.GetTextRangeSubstringWithoutLineBreaks(problem, 22);
        var question = new TextRange(rtbQuestion.Document.ContentStart, rtbQuestion.Document.ContentEnd);
        vM.CurrentMathProblem.ProblemQuestion = TextRangeHelper.GetTextRangeSubstringWithoutLineBreaks(question, 25);
        var contents = new TextRange(rtbCodeMode.Document.ContentStart, rtbCodeMode.Document.ContentEnd);

        vM.SaveCurrentCourse();
        vM.SaveCurrentMathProblem(contents);

        var curMathProblem = vM.CurrentMathProblem;
        vM.PopulateTempMathProblemsFromCurrentCourse();
        vM.CurrentMathProblem = curMathProblem;
        vM.TeacherNotNullCurrentMathProblemAboutToChange += ViewModelTeacher_CurrentMathProblemAboutToChange;
    }
}
```

Zdroj: Vlastní zpracování (2022)

6.6.3 Pomocné třídy Helper

Pomocné třídy v následujícím výčtu nesdílejí žádné společné rozhraní, ale spojuje je skutečnost, že jsou statické a cílem každé z nich je při psaní kódu usnadnit manipulaci s daty uloženými určitým způsobem. Jde o třídy `FilePathHelper` (pro jednotný a správný přístup k adresářům a souborům, s nimiž pracují zejména třídy `User`, `Course` a `MathProblem`), `TextRangeHelper` (obsahuje metody pro práci s datovým typem `TextRange`), `XmlHelper` (pro ukládání XML souborů) a `ZipHelper` (pro tvorbu a čtení složek komprimovaných metodou ZIP při exportu a importu kurzů).

Ukázka kódu č. 5: ZipHelper

```
Počet odkazů: 2
public static class ZipHelper
{
    Počet odkazů: 1
    public static bool UnzipDirectory(string zipPath, string destinationPath)
    {
        try
        {
            ZipFile.ExtractToDirectory(zipPath, destinationPath);
            return true;
        }
        catch (Exception e)
        {
            MessageBox.Show(e.Message);
            return false;
        }
    }
```

Zdroj: Vlastní zpracování (2022)

6.6.4 IValueConverter

Rozhraní `IValueConverter` slouží k převodu hodnot mezi dvěma datovými typy, typicky z vrstvy Model do formátu vhodnějšího pro jejich zobrazení ve vrstvě View (a naopak). Implementace tohoto rozhraní vyžaduje dvě metody, `Convert(Object, Type, Object, CultureInfo)` a `ConvertBack(Object, Type, Object, CultureInfo)`.

V aplikaci Nezmatematika jsem toto rozhraní využila ve dvou třídách. První z nich je `NullableColorToSolidColrBrushConverter` (zobrazena v ukázce kódu č. 5, převádí mezi `Color?` a `SolidColorBrush`), druhá je `SolidColorBrushToColorConverter` (pro převody mezi `SolidColorBrush` a `Color`). Obě třídy jsou velmi podobné a své role hrají při nastavování barev pozadí aplikace, poněvadž vlastnost `SelectedColor` třídy `ColorPicker` z NuGet balíčku Extended.Wpf.Toolkit je typu `Color?` a vlastnost `Background` u prvku `Grid` (a mnohých dalších) je typu `SolidColorBrush`.

Ukázka kódu č. 6: IValueConverter

```
Počet odkazů: 0
public class NullableColorToSolidColorBrushConverter : IValueConverter
{
    Počet odkazů: 0
    public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
    {
        Color? colour = (Color?)value;
        if (colour == null)
            return new SolidColorBrush(SystemColors.WindowColor);
        else
            return new SolidColorBrush((Color)colour);
    }

    Počet odkazů: 0
    public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
    {
        SolidColorBrush brush = (SolidColorBrush)value;
        Color? colour = brush.Color;
        return colour;
    }
```

Zdroj: Vlastní zpracování (2022)

7 Závěr

Informační technologie začínají ve výuce matematiky hrát stále významnější roli, k čemuž bezesporu významně přispěla a přispívá i distanční výuka spojená s pandemií covid-19. Přestože aplikace Nezmatematika začala vznikat v době, kdy její jedinou ambicí bylo zefektivnit samostatné procvičování doučovaných studentů, věřím, že by mohla najít uplatnění jako pomůcka studentů i vyučujících napříč všemi úrovněmi vzdělávání.

Teoretická část této práce se zabývá již existujícími aplikacemi pro výuku matematiky a metodikou vývoje softwaru. Seznamuje také čtenáře s platformou .NET Framework, systémem Windows Presentation Foundation (WPF), značkovacím jazykem XAML a architekturou Model-View-ViewModel. Praktická část práce zachycuje proces vývoje aplikace Nezmatematika od analýzy a návrhu až po implementaci celého projektu.

Má osobní životní situace se od zrodu projektu výrazně změnila (mimo jiné zaměstnáním na plný úvazek) a Nezmatematika tím pádem svůj prapůvodní účel (mé osobní použití pro zadávání úloh k procvičování doučovaným studentům) už zřejmě nenaplní. Z mého pohledu jde ale o aplikaci s velmi reálným potenciálem používání v běžné praxi.

Práce na vývoji aplikace Nezmatematika pro mě byla velmi náročná, a to hned z několika důvodů. Předně šlo o první opravdovou aplikaci, kterou jsem naprogramovala, takže křivka učení byla strmá. Dvojsečným ostřím byla skutečnost, že jsem vyráběla aplikaci čistě podle vlastních představ, takže jsem kromě "jednočlenného vývojového týmu" zastávala i role zákazníka a koncového uživatele. Ve své aplikaci jsem tak mohla mít cokoliv, co bych si usmyslela, pakliže to dokážu implementovat, což mě po celou dobu vývoje lákalo k ukusování si větších a větších soust.

Otevřeně přiznám, že kdybych v práci na aplikaci pokračovala, dokud s ní nebudu spokojená, pravděpodobně by vývoj Nezmatematiky neskončil nikdy. I teď, když píšu závěrečné odstavce této bakalářské práce, vidím na aplikaci, která je jejím středobodem, řadu nedostatků, které bych ráda změnila, než ji začnu ukazovat světu.

Vím ale, že to není realistický přístup, a že pokud má moje aplikace Nezmatematika svůj potenciál naplnit, nesmím svým perfekcionistickým tendencím dovolit stát tomu v cestě.

Z celého projektu si odnáším velké množství nových poznatků a zkušeností, které mi nepochybně v budoucnu dobře poslouží.

I. Summary and keywords

This bachelor thesis explores the process of creation and development of a desktop application for operating system Windows that is focused on practising Mathematics. Programming language C# and the environment Visual Studio were chosen for this project.

The application offers two modes – "Student" and "Teacher". The "Student" mode allows the user to practise solving mathematical problems. The application tracks success history of the user, allowing them to retry solving previously failed problems as well as new ones of the same type. All problems can be revisited later.

The "Teacher" mode puts emphasis on being highly customisable to suit the user's teaching style, which is provided by the integrated course editor, that supports various types of math problems. Selective questions, filling in the blanks and open questions where only the result is checked can all be created.

Key words: C#, Visual Studio, application development, mathematics, educational software

II. Seznam literatury

GeoGebra. (b.r.). O programu GeoGebra. Získáno 19. březen 2022, z GeoGebra website: https://www.geogebra.org/about

Hohenwarter, M., & Penier, J. (b.r.). Short History of GeoGebra. Získáno 19. březen 2022, z https://www.maa.org/external_archive/joma/Volume7/Hohenwarter/History.html

Iqbal, K. (2019, říjen 11). RTF - Rich Text Format. Získáno 2. duben 2022, z https://docs.fileformat.com/word-processing/rtf/

Kadlec, V. (2004). *Agilní programování* (1. vydání). Computer Press.

Kodytek, S. (b.r.). Lekce 1 - Úvod do metodologie vývoje softwaru. Získáno 9. duben 2022, z https://www.itnetwork.cz/uvod-do-metodologie-vyvoje-softwaru

MacDonald, M. (2013). *Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5* (Fourth Edition). Apress.

Maloušek, J. (2020). *Návrhové vzory architektury OS Android s využitím jazyka Kotlin* (Vysoké učení technické). Vysoké učení technické, Brno. Získáno z https://theses.cz/id/g29534/DB_Jan_Malousek.pdf

Photomath. (b.r.). Photomath – Aplikace na Google Play. Získáno 21. březen 2022, z https://play.google.com/store/apps/details?id=com.microblink.photomath&hl=cs&gl=US

Posadas, M. (2016). *Mastering C# and .NET Framework* (1st Edition). Packt Publishing.

Prorok, L. (2010, květen 14). XML – WikiKnihovna. Získáno 28. březen 2022, z https://wiki.knihovna.cz/index.php/XML

VUT. (2020, září 3). Aplikace na doučování matematiky z FEKT získá podporu od známého operátora. Získáno 22. březen 2022, z http://www.vut.cz/vut/aktuality-f19528/aplikace-na-doucovani-matematiky-z-fekt-ziska-podporu-od-znameho-operatora-d201777

W3C. (b.r.). Extensible Markup Language (XML). Získáno 28. březen 2022, z https://www.w3.org/XML/

Wolfram|Alpha. (b.r.). About Wolfram|Alpha: Making the World's Knowledge Computable. Získáno 19. březen 2022, z https://www.wolframalpha.com/about

Wolfram|Alpha. (b.r.). Historical Timeline of Computable Knowledge: 1960-2010. Získáno 19. březen 2022, z https://www.wolframalpha.com/docs/timeline/computable-knowledge-history-6

Yuen, S. (2020). *Mastering Windows Presentation Foundation: Build responsive UIs for desktop applications with WPF* (Second Edition). Packt Publishing.

III. Seznam obrázků

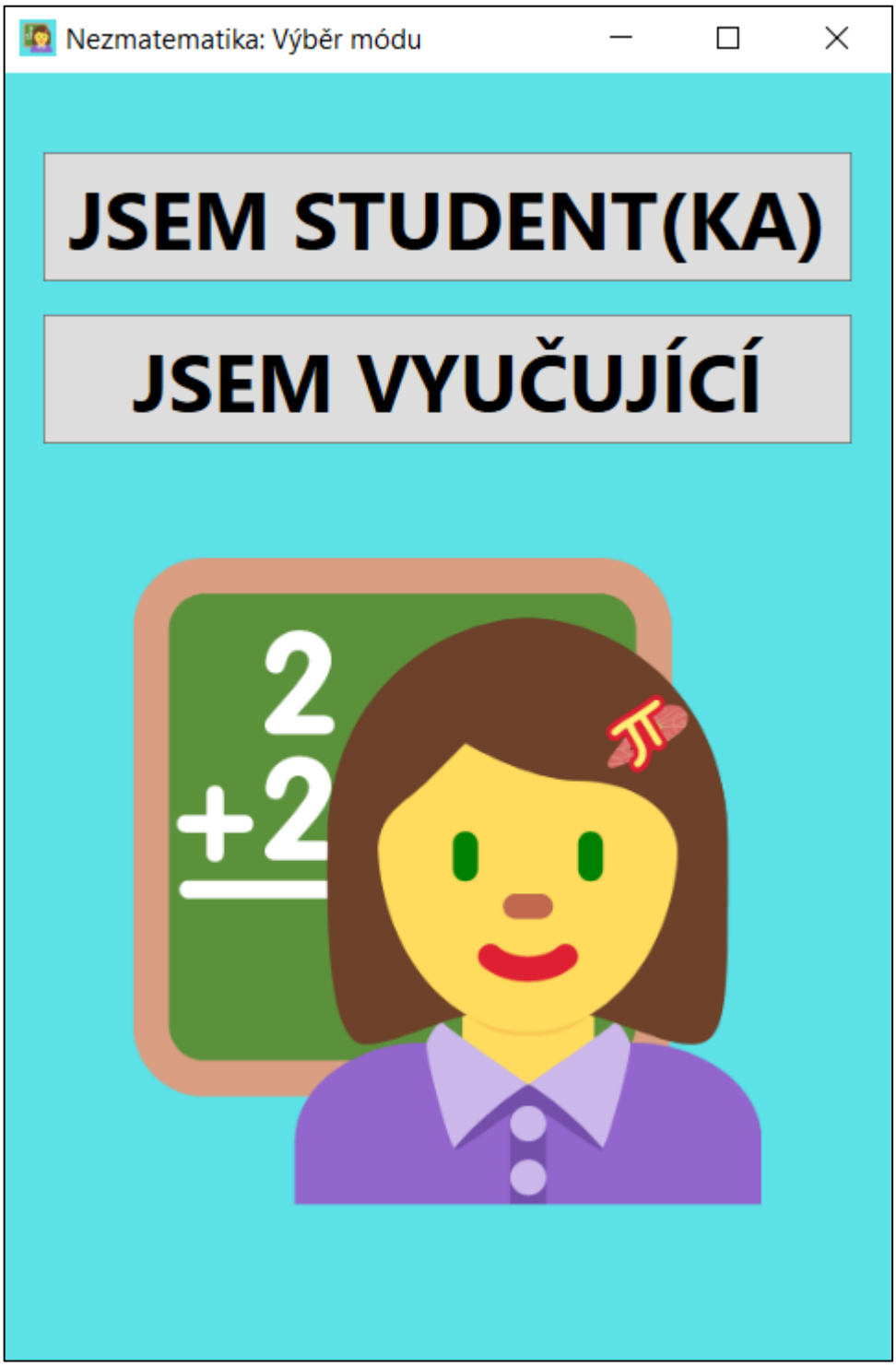
IV. Seznam tabulek

V. Seznam ukázek kódu

VI. Seznam příloh

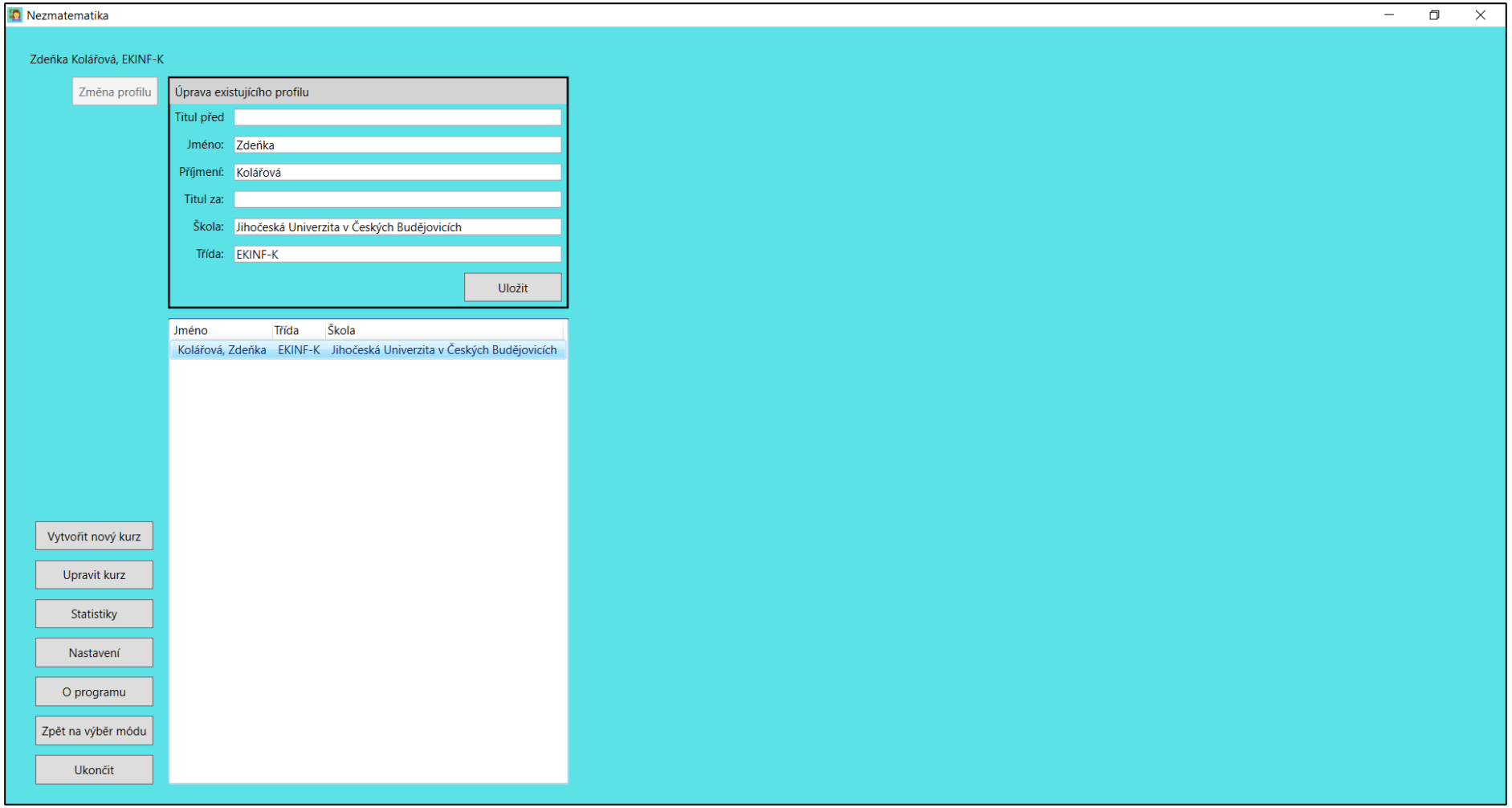
VII. Přílohy

Příloha č. 1: Úvodní obrazovka



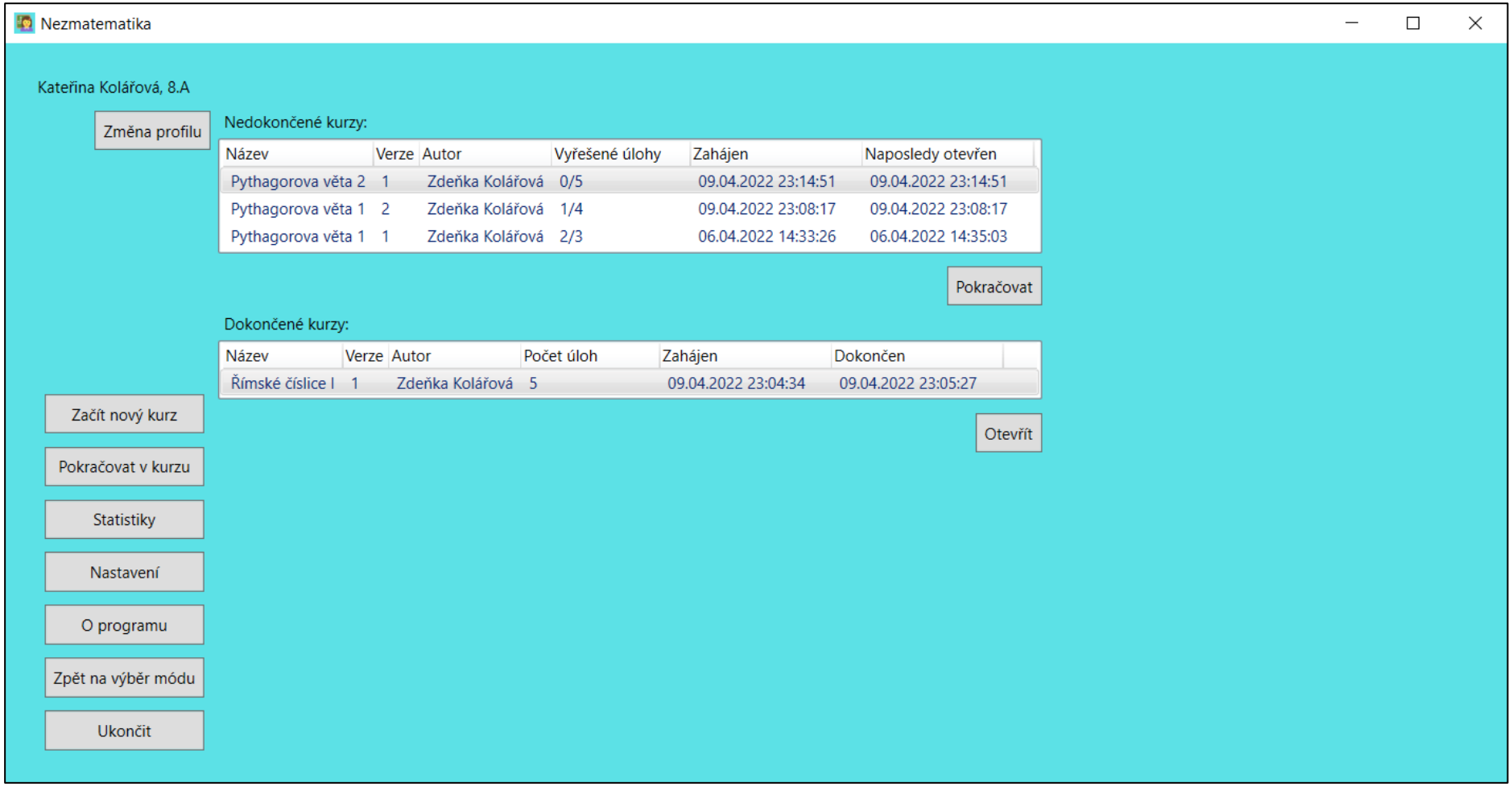
Zdroj: Vlastní zpracování (2022)

Příloha č. 2: Obrazovka s formulářem editace profilu



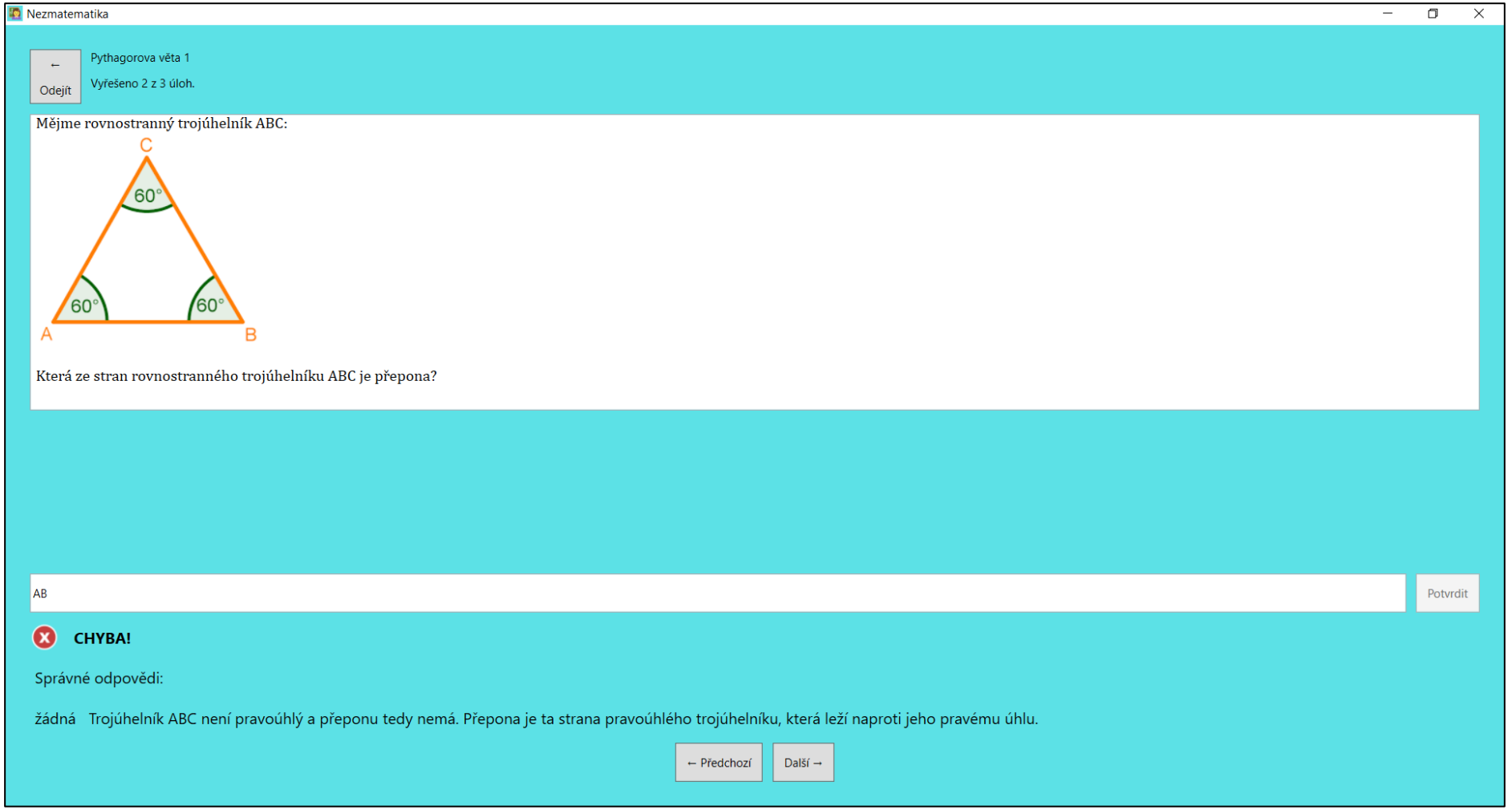
Zdroj: Vlastní zpracování (2022)

Příloha č. 3: Obrazovka výběru z dříve zahájených kurzů v módu Student



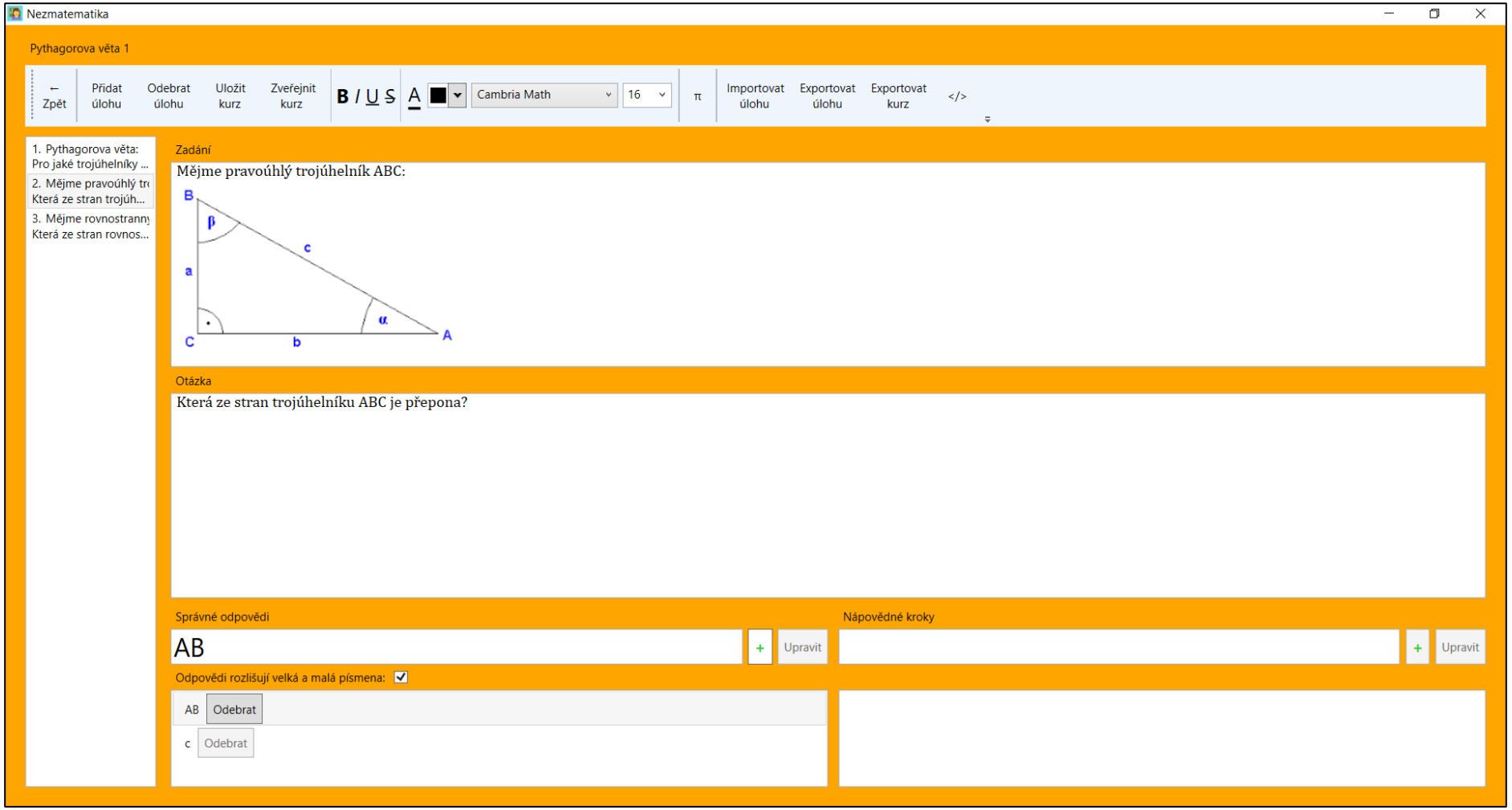
Zdroj: Vlastní zpracování (2022)

Příloha č. 4: Obrazovka procházení kurzu v módu Student



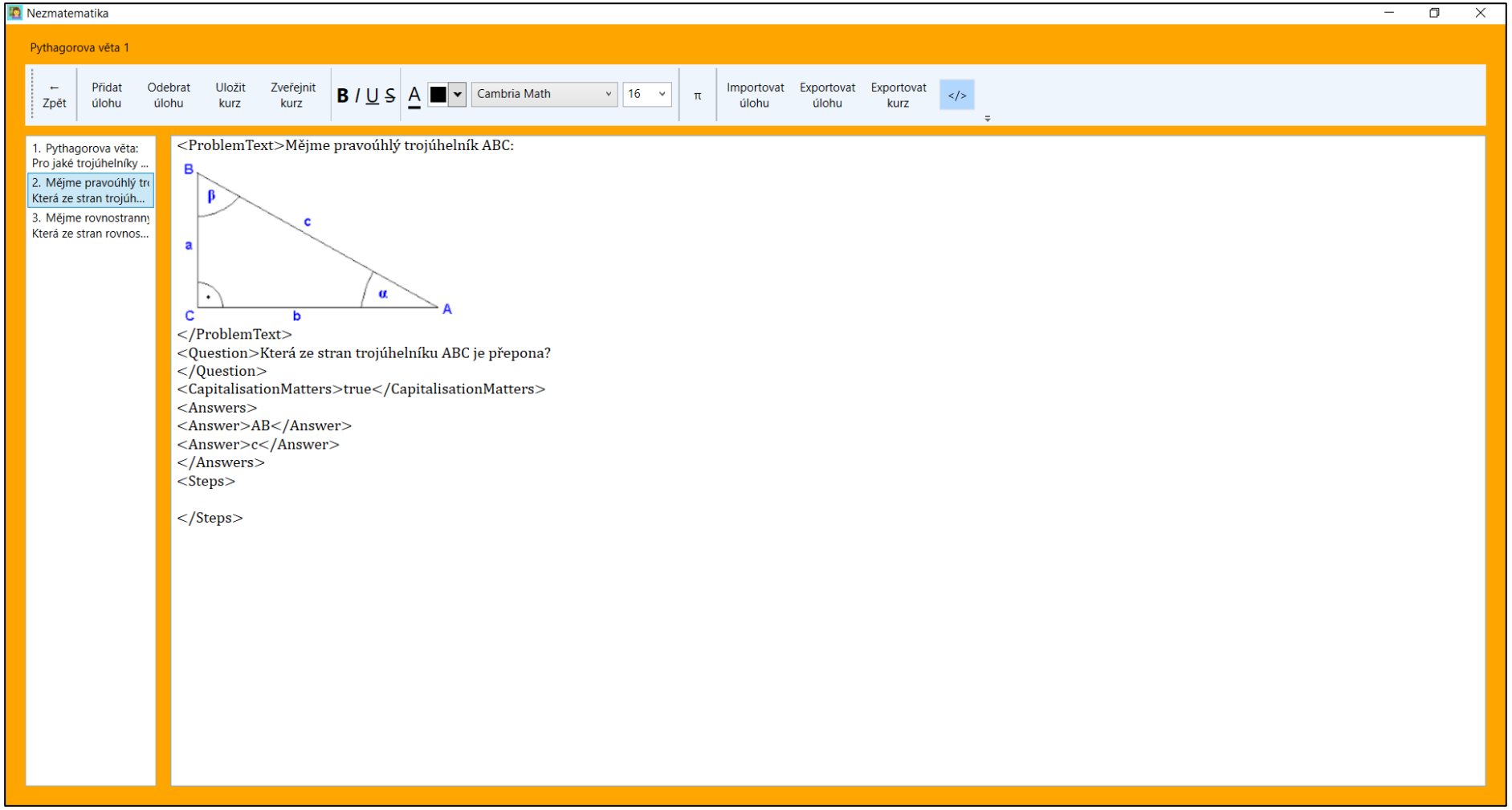
Zdroj: Vlastní zpracování (2022)

Příloha č. 5: Obrazovka editoru kurzů v uživatelském režimu



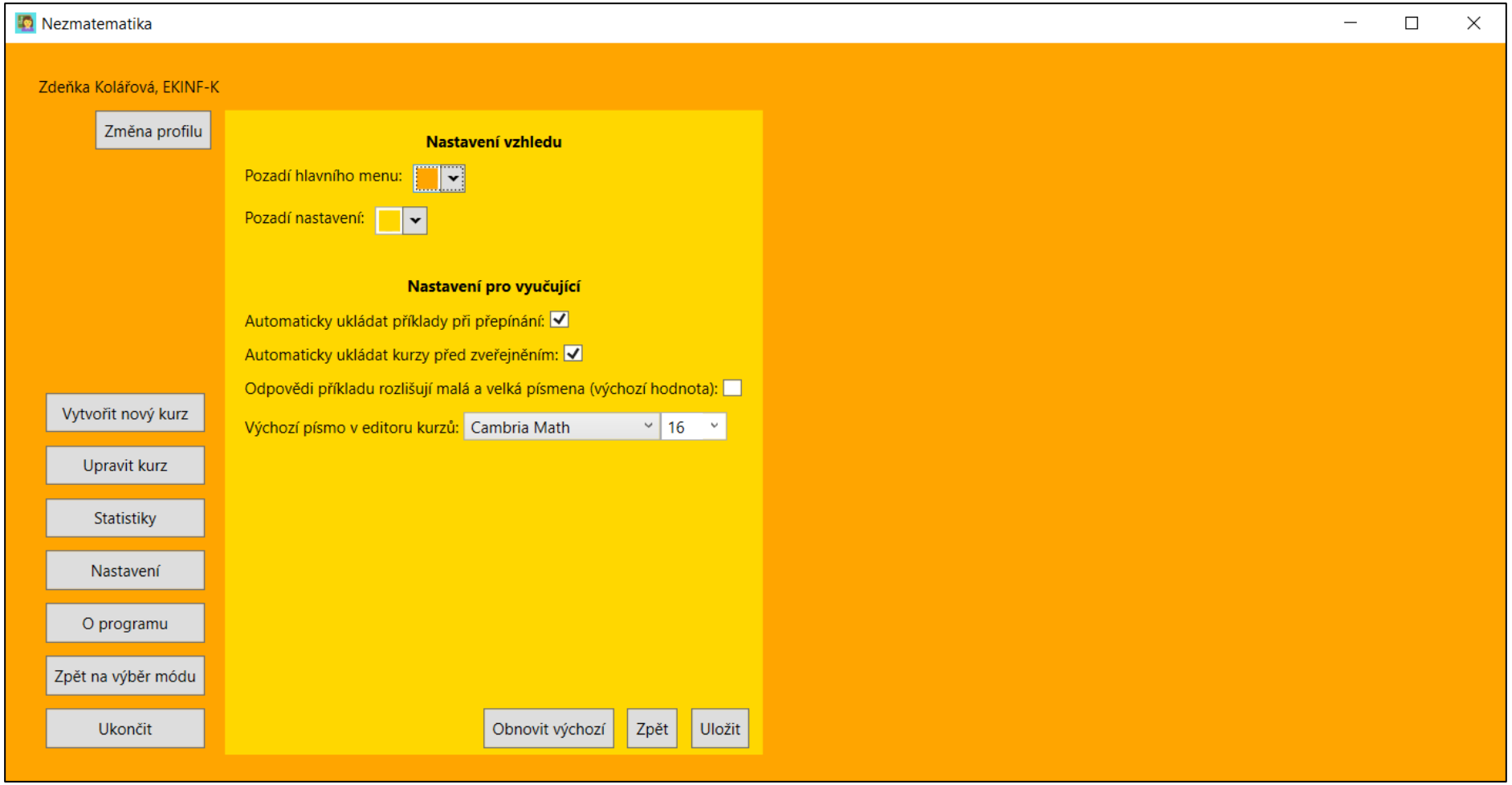
Zdroj: Vlastní zpracování (2022)

Příloha č. 6: Obrazovka editoru kurzů v kódovém režimu



Zdroj: Vlastní zpracování (2022)

Příloha č. 7: Obrazovka nastavení v módu Vyučující



Zdroj: Vlastní zpracování (2022)