ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Mobilní aplikace k nástroji MARAST
Student:	Bc. Jindřich Štěpánek
Vedoucí:	Ing. Štěpán Starosta, Ph.D.
Studijní program:	Informatika
Studijní obor:	Webové a softwarové inženýrství
Katedra:	Katedra softwarového inženýrství
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

1) Seznamte se s aplikací Matematika radostně (MARAST) a jejím aktuálním stavem a plánovanými změnami.
2) Vytvořte spolu s tvůrci aplikace MARAST seznam požadavků na mobilní verzi této aplikace. Mezi požadavky bude přihlášení uživatele, nahrání a synchronizace uživatelských dat (především poznámky k příkladům), zobrazování matematických vzorců pomocí LaTeX, zobrazování videokurzu a možnost zobrazování různých typů příkladů včetně možnosti jednoduchého přidání zobrazení nového typu příkladu.
3) Proveďte rešerši podobných mobilních aplikací a na jejím základě a na základě požadavků a vlastních nápadů vytvořte návrh mobilní aplikace MARAST včetně návrhu nutných modifikací stávající aplikace MARAST (především za účelem sjednocení rozhraní k databázi příkladů).
4) Navrženou aplikaci implementujte pro iOS, implementujte i nutné změny v původní aplikaci (především rozhraní k databázi).
5) Aplikaci otestujte a upravte ji podle výsledků testování.

Seznam odborné literatury

Dodá vedoucí práce.

Ing. Michal Valenta, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 31. října 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra Softwarového inženýrství

Diplomová práce

Mobilní aplikace k nástroji MARAST

Bc. Jindřich Štěpánek

Vedoucí práce: Ing. Štěpán Starosta, Ph.D.

8. května 2017

Poděkování

Rád bych poděkoval především Ing. Štěpánu Starostovi, Ph.D. za vedení práce a veškerou pomoc při její tvorbě. Dále pak Ing. Tomáši Kalvodovi, Ph.D. za spolupráci a důležité informace a poznatky pro tvorbu práce. Poděkování patří i každému, kdo mě v průběhu práce a celého studia podporoval.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 8. května 2017 .....

Odkaz na tuto práci

Štěpánek, Jindřich. *Mobilní aplikace k nástroji MARAST*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Práce se zabývá tvorbou mobilní aplikace k nástroji MARAST určeného pro výuku matematiky na Fakultě informačních technologií ČVUT v Praze. Součástí práce je analýza funkcionalit jeho webové verze a návrh potřebných změn pro sdílení dat s mobilní aplikací. Výstupy práce jsou návrh a implementace aplikačního rozhraní pro sdílení dat a vyvinutá funkční mobilní aplikace MARAST pro systém iOS.

Klíčová slova Matematika, MARAST, mobilní aplikace, iOS.

Abstract

This thesis focuses on creating a mobile application of MARAST which is a tool for learning math on Faculty of Information Technology of the Czech Technical University in Prague. Thesis includes analysis of MARAST's web version and specification of suggested changes needed for enabling data sharing with mobile application. Outcomes of this thesis are a design and implementation of application interface for sharing data and developed functional mobile application MARAST for iOS system.

Keywords Mathematics, MARAST, mobile applications, iOS.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Motivace

Spolu s rozvojem informačních technologií jde ruku v ruce i jejich zapojení do oblasti vzdělávání. Pro nové formy z toho plynoucí se pak ustálilo označení e-learning. Tento pojem může být vyložen mnoha různými způsoby, z pedagogického hlediska jej [1] definuje jako proces využívání multimediálních technologií, internetu a dalších elektronických médií pro zlepšení kvality vzdělávání.

V posledních letech se na popředí zájmu pohybují především mobilní technologie, přičemž adaptace pro mobilní zařízení přináší i pro e-learninové systémy řadu nových možností od samostné přenositelnosti a ovládání po efektivní upozorňování uživatelů na události spojené s daným systémem.

Na školách s technologickým zaměřením si pak přirozeně moderní systémy pro výuku rychle nacházejí uplatnění a jedním z oborů, kde je jejich využití nasnadě, je matematika.

MARAST

Pro účely výuky matematiky na Fakultě informačních technologií ČVUT v Praze vznikl systém MARAST, plným názvem MAtematika RadoSTně. Ten slouží v první řadě jako neustále rozšiřovaná sbírka příkladů a dalších studijních materiálů, které jsou využívány pro automatické generování cvičení a kvízů v jednotlivých matematických kurzech na fakultě.

Motivace a okolnosti vzniku MARASTu jsou shrnuty v úvodníku na jeho webu:

> *„Samotný webový portál byl vytvořen během zimního semestru akademického roku 2012/2013. V té době se také databáze začala plnit příklady s velkou pomocí cvičících předmětu Základy matematické analýzy. Tyto příklady pak byly použity při vytváření zápočtových písemek. Vzhledem k počtu studentů a s ním spojenému počtu kruhů*

> *cvičení byl veliký problém vytvořit dostatek podobně obtížných písemek. S MARASTem se nám s tímto problémem podařilo uspokojivě vypořádat. Vložené příklady nejsou určeny pouze pro zápočtové písemky. Velkou část z nich si může každý projít přímo na těchto stránkách. U většiny z nich najdete i výsledek, často doplněný o náš postup řešení. Cílem je vytvořit jakousi online cvičebnici.*"[2]

Ve své první verzi MARAST fungoval pouze jako jednoduchá cvičebnice s databází příkladů pro samostudium, současná druhá verze pak již nabízí generovaní online kvízů či zadání tištěných testů a další postupně rozšiřované funkcionality popsané v dalších kapitolách.

Obsah práce a specifikace cílů

Cílem práce je po analýze systému MARAST navrhnout a implementovat změny potřebné pro sdílení dat se zamýšlenou mobilní verzí a následně na základě vytvořeného aplikačního rozhraní vyvinout funkční aplikaci MARAST pro platformu iOS.

Výchozím bodem práce je seznámení se s technologiemi využitými v systému MARAST, analýza jeho funkcionality a výběr funkcí vhodných pro převod do mobilního rozhraní. Výběr bude navíc rozšířen o návrh funkcionalit nových, které může mobilní aplikace přinést coby přidanou hodnotu. Součástí této části práce je analýza mobilních aplikací, které mohou být vzorem pro navrhované funkcionality mobilní verzi MARASTu.

V další části práce bude proveden uživatelský průzkum na téma mobilní aplikace MARAST, který spolu s předchozí analýzou bude sloužit jako zdroj poznatků pro sestavení seznamu požadavků na mobilní aplikaci.

Na základě navržených požadavků bude v rámci další části práce sestaven návrh mobilní aplikace, jejího uživatelského rozhraní a aplikačního rozhraní webové verze potřebného pro sdílení a synchronizaci dat.

Výstupem práce bude implementovaná funkční aplikace pro mobilní platformu iOS. Na základě předchozího návrhu bude zároveň implementováno aplikační rozhraní webové aplikace pro sdílení potřebných dat. Výsledná aplikace a aplikační rozhraní budou následně podrobeny vhodným testům.

Kapitola 1

Analýza

V této kapitole budou popsány funkcionality MARASTu, které dále prostupují dalším textem práce. Dále budou představeny mobilní aplikace, které mohou být pro mobilní verzi MARASTu vzorem. V další části kapitoly budou popsány technologie, které jsou využity ve webové verzi MARASTu.

1.1 Funkcionality MARASTu

V následující sekci budou popsány jednotlivé logické části a funkcionality, z nichž se MARAST skládá a které budou prostupovat textem této práce.

1.1.1 Příklad

MARAST je v jádru především sbírkou příkladů, se kterými ostatní funkcionality operují. Zadání příkladu je kombinací textu a matematických formulí zadávaných ve formátu sázecího systému LaTeX[3].

Příklady jsou zařazeny do tématických okruhů a několika možných obtížností. V současnosti pak mohou být trojího typu:

- Výběrový příklad obsahuje otázku s několika možnými odpověďmi na které student odpovídá interaktivně.
- Textový příklad obsahuje pouze otázku, odpověď a případně postup řešení.
- Příklad s textovým políčkem je interaktivní a vyžaduje odpověd formou textu.

1.1.2 Cvičebnice

Cvičebnice je množinou studentům volně dostupných příkladů, ve kterých lze listovat pomocí několika dostupných filtrů. U příkladů v cvičebnici lze zobra-

zit odpověd, popřípadě i postup řešení. Dále si pak jednotliví řešitelé mohou k příkladům dopňovat vlastní poznámky a označovat stav řešení.

1.1.3 Kvíz

Kvízy jsou průběžné testy zařazené do daných kurzů v semestru a skládají se z filtru příkladů pro generování zadání a řady pravidel řídicích průběh plnění kvízu a jeho hodnocení.

1.1.4 Lekce

Lekce jsou sbírky studijních materiálů v jednotlivých kvízech. Témata materiálů jsou řazeny do hierarchické struktury a jejich text je dopňován o videopřednášky a řešené příklady z databáze.

1.1.5 Vedlejší funkcionality

Mimo výše zmíněného dále MARAST obsahuje i vedlejší funkcionality pro interakci se studenty. Blogové příspěvky informují o novinkách a dalším dění okolo MARASTu. U všech vyučovaných kvízů a také u jednotlivých příkladů je vedena diskuze.

1.2 Podobné aplikace

V následující sekci budou rozebrány aplikace podobného charakteru jako MARAST, pro který mohou sloužit jako inspirace funkcemi svých mobilních verzí a jejich uživatelským rozhraním. Jedná se o přední e-learningové aplikace nebo aplikace určené pro výuku matematiky.

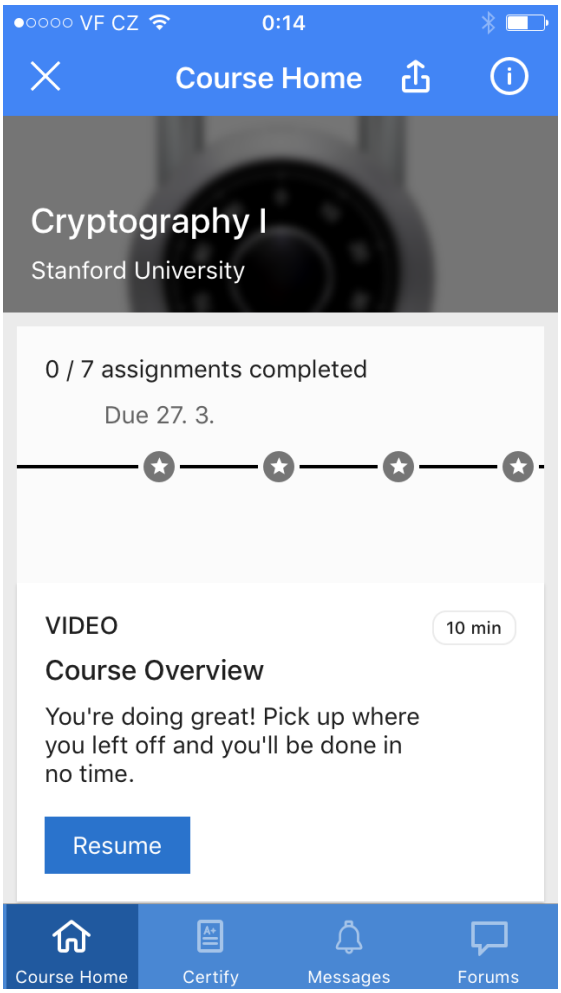
1.2.1 Coursera

Coursera je projekt, který založily v dubnu roku 2014 Daphne Koller a Andrew Ng, profesoři informatiky ze Stanford Universidy. Jejím cílem je zpřístupňování vybraných kurzů z předních světových univerzit široké veřejnosti. Tyto kurzy je možné po registraci absolvovat bezplatně, přičemž absolvent má následně možnost vystavení placeného ověřeného certifikátu. Kurzy jsou pořádány v časových obdobích odpovídajících vysokoškolským semestrům. V průběhu tohoto období jsou postupně zveřejňovány videopřednášky, úlohy, kvízy a další výukové materiály.

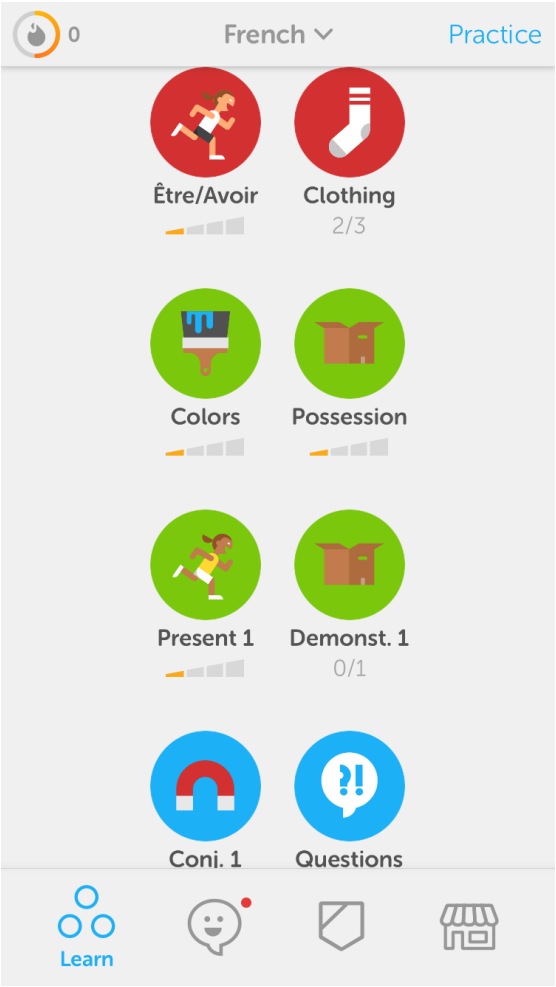
V mobilní verzi aplikace je vedle snadného prohlížení studijních materálů také možnost jejich lokálního ukládání pro použití bez nutnosti internetového připojení. Dále je k dispozici možnost nastavení notifikací o blížících se termínech v daném kurzu. Na výchozí obrazovce aplikace je po přihlášení zobrazen seznam aktuálně otevřených kurzů daného uživatele. Obrazovka jednotlivých

kurzů je členěna do čtyř záložek, z nichž výchozí a stěžejní je časová osa s informacemi o průbehu kurzu a výběrem studijních materiálů dělených do jednotlivých týdnů.

Coursera je svým principem blízko MARASTu, přičemž hlavním pojítkem je účast studenta současně v několika kurzech, ve kterých jsou dostupné studijní texty, videopřednášky, cvičení, kvízy a diskuze. Hlavní inspirací pro mobilní aplikace MARAST může být způsob rozložení obrazovky kurzu a notifikace o blížících se termínech zobrazená na obrazků 1.1.



Obrázek 1.1: Coursera, obrazovka zvoleného kurzu



Obrázek 1.2: Duolingo, výběr lekcí

1.2.2 Duolingo

Duolingo je platforma pro výuku a procvičování cizích jazyků uvedená v roce 2012 Louisem von Ahnem. Výuka zde probíhá v lekcích, které jsou zcela interaktivní. Gramatika, fráze i slovní zásoba jsou již od počátku vyučovány formou krátkých cvičení, kdy k samotnému vysvětlení problematiky slouží přímo zadání či doplňující popisek. Jednotlivé lekce jsou zde fakticky kvízem, k jehož dokončení je třeba dokončit všechna cvičení a současně se vejít do dané tolerance počtu chyb. Tato cvičení jsou několika různých typů jako je výběr správné možnosti z nabízených odpovědí či obrázků, překlad textu jedním či druhým směrem, poslech mluveného textu nebo cvičení výslovnosti, jelikož aplikace je schopná rozpoznávání řeči.

Aplikace v současné době funguje zcela zdarma a ke své monetizaci využívá technik výpočtů prováděných lidmi. Uživatelům jsou k překladu nabízeny útržky komerčních textů, které jsou následně zároveň validovány zkušenějšími uživateli.

Zásadním prvkem aplikace je gamifikace neboli zapojení herních principů a technik, které činí výuku více poutavou a motivující. Vedle zvyšování uživatelovy úrovně a sbírání virtuální měny určené ke směně za drobné doplňky, zde hraje důležitou roli možnost nastavování denních cílů. Postupné plnění těchto cílů je spojuje do nepřetržitých šňůr, které uživatele motivují k překonávání jejich délky.

Právě způsob motivace uživatele ke každodení aktivitě může být inspirací pro budoucí vývoj MARASTu. Z pohledu mobilní aplikace je zajímavá intenzita a pojetí lekcí. Uživatel zde má možnost se efektivně učit bez dlouhého čtení či poslechu opravdu téměř kdykoliv a kdekoliv.

Na obrázku 1.2 je k vidění obrazovka aplikace pro výběr lekcí.

1.2.3 Khan Academy

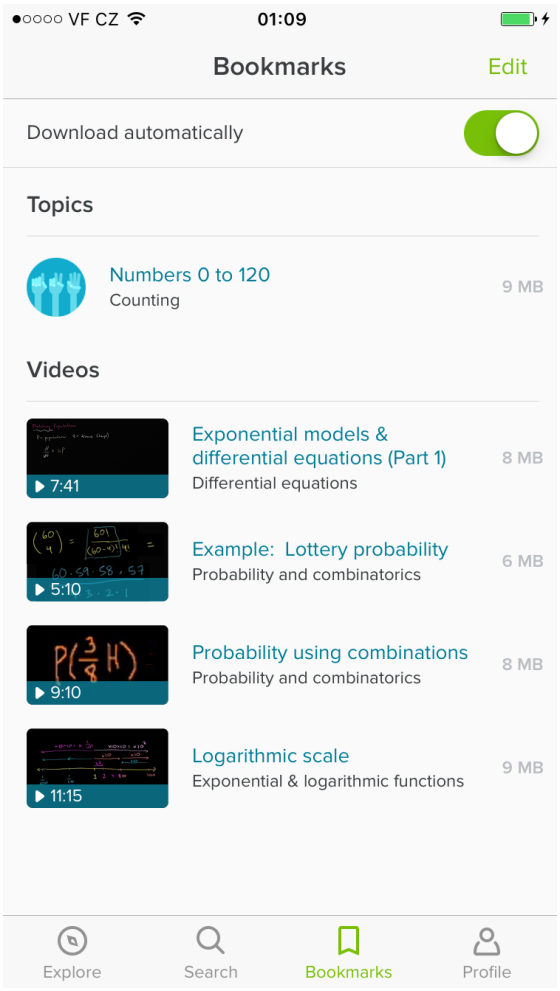
Khan Adacademy, v české verzi Khanova škola, je bezplatný vzdělávací nástroj založený američanem Salmanem Khanem. K výuce využívá série krátkých videí délkou nepřesahujících 10 minut. Videa obsahují pouze černou tabuli, na které je přednášená látka názorně vysvětlována. V současné době Khanova akademie dle udávaných statistik [4] disponuje více než deseti tisíci videi z rozličných oborů. Přes 3400 z nich již navíc bylo přeloženeno do českého jazyka v rámci české odnože s názvem Khanova škola [5].

Videolekce jsou dostupné i v mobilní aplikaci Khan Academy, přičemž zajímavou funkcí je možnost nastavení rychlosti přehrávání. Uživatel si zároveň může témata a jednotlivá videa označovat jako oblíbená s možností jejich stažení pro přehrávání offline. To je naznačeno na obrazovce zobrazené na obrázku 1.3.

1.2.4 iMathematics

iMathematics je mobilní aplikace zaměřená na výuku matematiky. Obsahuje množství lekcí pokrývajících látku od vyššího stupně základní školy po pokročilejší vysokoškolskou matematiku. Každá lekce je doplněna o několik kvízů a na základě jejich plnění je v procentech určována dovednost uživatele v daném tématu. Aplikace je v základu dostupná zdarma, přičemž platící uživatel má možnost okamžitého otevření všech lekcí a kvízů. Naopak uživatel neplatící je otevírá postupným plněním, přičemž v případě neúspěchu v daném kvízů musí pro možnost jeho opakovaní několik minut počkat.

Pro mobilní aplikace MARAST může být iMathematics inspirací především svým povedeným uživatelským rozhraním, jelikož pojetí lekcí a kvízů v iMathematics má velmi blízko k jejich obdobám v MARASTu. Lekce jsou

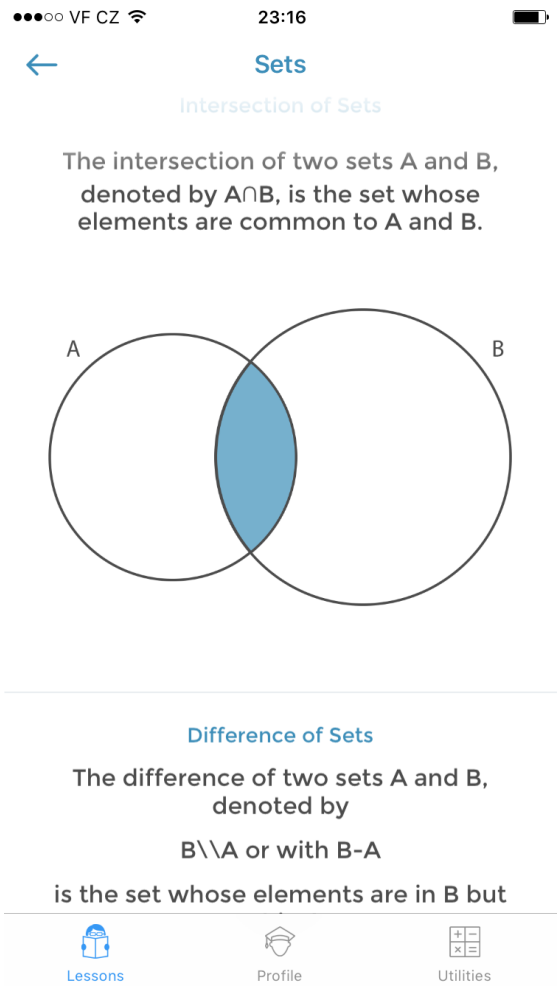


Obrázek 1.3: Khan Academy, knihovna videí

zde kombinací textu a matematických formulí, grafů a doplňujících náčrtů. Zároveň lekcemi prostupují příklady k procvičení, u nichž je původně skyto řešení, které je možné jedním klikem odkrýt. Přímo z obrazovky lekce je pak zároveň dostupný i seznam souvisejících kvízů, které jsou kombinací příkladů s volbou odpovědi z několika možností a příkladů se zadáváním textové odpovědi. Jak vidno na obrázcích 1.4 a 1.5, lekce i kvízy jsou vyvedené v jednoduchém intuitivním stylu s použitím příjemných písem a barev.

1.3 Webová aplikace MARAST

V následující sekci budou shrnuty klíčové technologie, na kterých staví webová aplikace MARAST a jejichž základní znalost je potřebná pro návrh aplikač-

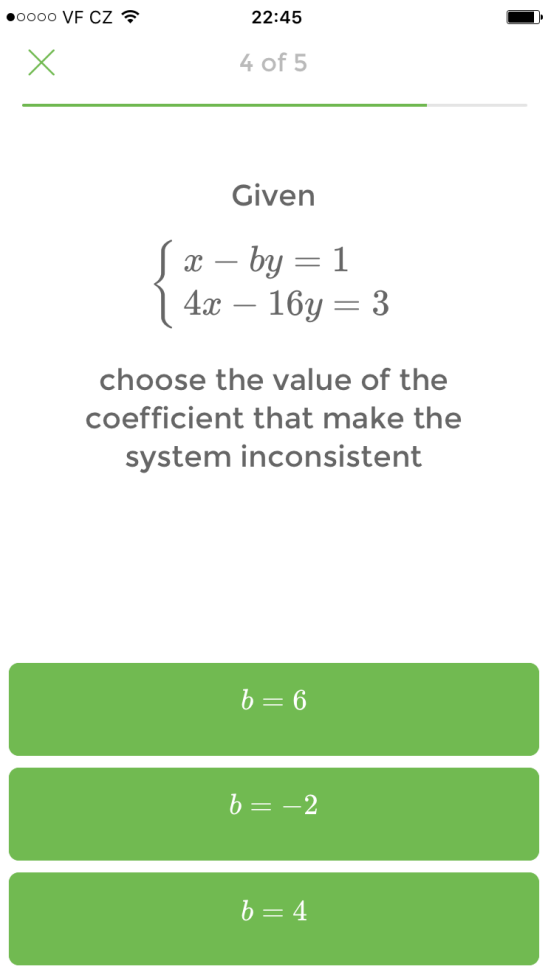


Obrázek 1.4: iMathematics, ukázka lekce

ního rozhraní a mobilní aplikace.

1.3.1 Ruby

Ruby je interpretovaný programovací jazyk, který je dynamicky typovaný a plně objektově orientovaný . Vše v Ruby je tedy objektem. Řadí se mezi vysokoúrovňové jazyky, tedy programovací jazyky s vyšší úrovní abstrakce od technických stránek počítače, přibližující zápis kódu lidskému myšlení. Je navržen k jednoduchosti a rozšiřitelnosti a jako interpretovaný jazyk je přenositelný mezi platformami.



Obrázek 1.5: iMathematics, ukázka kvízu

1.3.2 Ruby on Rails

Ruby On Rails, zkráceně Rails, je framework pro vývoj webových aplikací, který v roce 2004 publikoval dánský programátor David Heinemeier Hansson. Framework je dodnes udržován a inovován početnou komunitou uživatelů, mezi které dle [6] patří i mezinárodní společnosti jako Apple či Electronic Arts. Framework je vyvíjen v jazyce Ruby a navržen tak, aby usnadňoval programování webových aplikací na základě obecných předpokladů vývoje pro web. Je postaven na bázi architektury Model-View-Controller a zahrnuje abstraktní vrstvu pro práci s databází, zabudovanou podporu pro automatizované testování všech vrstev aplikace, generátory kódu či konzoli pro interaktivní práci s aplikací.

Filosofie Rails, jak ji uvádí [7], obsahuje několik vůdčích principů:

- „Don't repeat yourself" aneb „neopakujte se" je doporučení psát znovupožitelný kód zamezující opakování stejného či podobného kódu na více místech v rámci jedné aplikace.
- „Konvence má přednost před konfigurací" — Rails na základě obecných zvyklostí a zavedených praktik předpokládají, co vývojář požaduje a velkou část vývoje tak automatizují bez nutnosti specifikace každé drobnosti. Vývojář tak konfiguruje pouze ty části aplikace, které se liší od běžného nastavení.
- REST jako nevhodnější architektonický vzor modelování entit webových aplikací a jejich vzájemných vztahů.

1.3.3 Model-View-Controller

Model-View-Controller, zkráceně MVC, je vzor softwarové architektury rozdělující uživatelské rozhraní, datovou vrstvu a řídící logiku aplikace na tři nezávislé komponenty tak, aby změny jedné komponenty neměly dopad na funkčnost ostatních komponent. Definice architektury MVC dle [8] je následovná:

> *„Model představuje objekt aplikace, View prezentaci na obrazovce a Controller definuje způsob, jak uživatelské rozhraní reaguje na vstup od uživatele. MVC odděluje tyto objekty a tím zvyšuje flexibilitu a znovupoužitelnost celého řešení."*

V Ruby on Rails modely obsahují většinu aplikační logiky a slouží primárně k interakci s příslušnou tabulkou v databázi, kdy jedna tabulka v databázi typicky odpovídá jednomu modelu v aplikaci. Views, neboli pohledy, reprezentují výstup aplikace, většinou ve formě uživatelského rozhraní. To je v Rails typicky definované HTML zápisem s vloženými částmi Ruby kódu provádějícího úkony spojené s prezentací dat. Kontrolery pak mezi pohledy a modely fungují jako mezivrstva, jejíž hlavní zodpovědností je zpracovávání požadavků webového prohlížeče či jiného nástroje požadujícího data aplikace.

1.3.4 REST

Representational state transfer, zkráceně REST, je model softwarové architektury popisující definici a adresaci zdrojů, využívaný pro návrh webových aplikací a služeb. Jeho původ sahá do roku 2000, kdy jej Roy Fielding popsal v dizertační práci Architectural Styles and the Design of Network-based Software Architectures [9]. Aplikační rozhraní navrhované ve stylu REST, označovány výrazem RESTful, definují jednotné rozhraní komunikace mezi klientem

a serverem umožňující oběma stranám nezávislý vývoj. Každý zdroj je definován jednou specifickou URI adresou a reprezentován v daném formátu. Komunikace probíhá bezestavově, jednotlivé požadavky klienta jsou vzájemně nezávislé. Každý požadavek by měl tedy obsahovat všechny informace, které server potřebuje pro jeho úspěšné zpracování. Na straně serveru tak nejsou ukládány žádné informace o stavu klienta.

REST implementuje čtyři základní typy operací, známé pod označením CRUD, tedy vytváření dat (Create), získání požadovaných dat (Retrieve), jejich změnu (Update) a smazání (Delete). REST je navržen nezávisle na komunikačním protokolu, ale typicky bývá spojován především s protokolem HTTP, u kterého jsou CRUD operace mapovány na odpovídající HTTP metody naznačené v následujícím seznamu:

- **GET (Retrieve)**

  Poskytování dat zdroje na zadané adrese reprezentovaných v požadovaném formátu.

- **POST (Create)**

  Vytváření nového zdroje. V požadavku jsou obsažena data nového zdroje a adresa, pod kterou je nový zdroj vytvářen. Přesná adresa nového zdroje je pak součástí odpovědi.

- **PUT (Update)**

  Modifikace či vytvoření zdroje na zadané adrese.

- **DELETE**

  Smazání zdroje na zadané adrese.

1.3.5 Active Record

Ruby on Rails pro práci s databází využívá technik objektově relačního mapování, kdy tabulky relační databáze jsou automaticky mapovány na objektovou strukturu aplikačního modelu. V praxi tak programátor nepřistupuje k databázi přímo pomocí jejích metod, ale synchronizace objektů v aplikaci a jejich reprezentace v databázovém systému je zařízena automaticky. Každý řádek odpovídající databázové tabulky zde reprezentuje konkrétní instancí objektu. Framework implementující objektově relační mapování v Ruby on Rails se nazývá Active Record.

1.3.6 LaTeX

LaTeX je sázecí systém vhodný pro sazbu matematických, technických či jiných náročnějších publikací z hlediska sazby textu. Dokument v LaTeXu je zapisován

jako text s formátovacími příkazy, na základě nichž je poté výsledný dokument vysázen s využitím profesionálních standardů.

Důležitou roli v LaTeXu hraje zápis matematických formulí, k čemuž je využíván i v MARASTu. Podporu LaTeXu ve webových prohlížečích pak zajišťuje JavaScriptová knihovna MathJax. Ta je stažena spolu s webovou stránkou. Během zobrazení pak MathJax prohledá obsah stránky a postará se o vysázení matematických formulí. Nevyžaduje tedy instalaci na straně uživatele a je využitelný na všech platformách a prohlížečích s podporou JavaScriptu.

1.3.7 Markdown

Markdown je značkovací jazyk sloužící k formátování prostého textu, který může být následně převeden do HTML či jiných nástrojů. Síla Markdownu přitom tkví v jeho jednoduchosti, jelikož je snadno čitelný a zobrazitelný i bez převodu do jiného formátu.

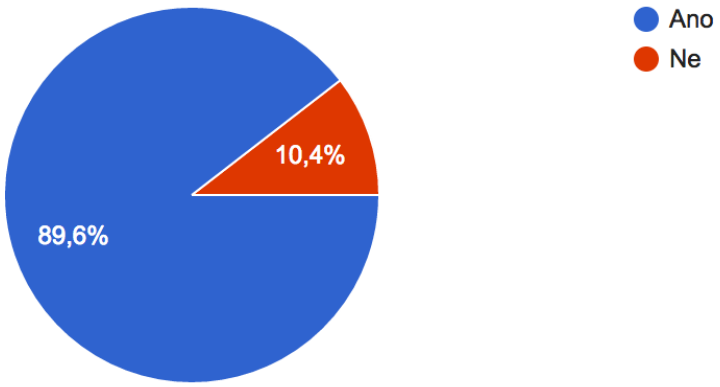
V MARASTu je Markdown využíván pro formátování textu lekcí.

Kapitola 2

Uživatelský průzkum

V následující kapitole bude rozebrán uživatelský průzkum na téma mobilní aplikace MARAST. Výzkum byl veden formou formuláře s výběrem volného množství odpovědí z nabízenýhch možností či doplnění odpovědi vlastní. Formulář byl prostřednictvím e-mailu šířen 27.10.2016 současným uživatelům MARASTu a sešlo se v něm celkem 383 odpovědí na zadané otázky a 27 doplňujících odpovědí ve volné sekci. Zbytek kapitoly bude věnován jednotlivým otázkám průzkumu a z nich vycházejícím poznatkům.

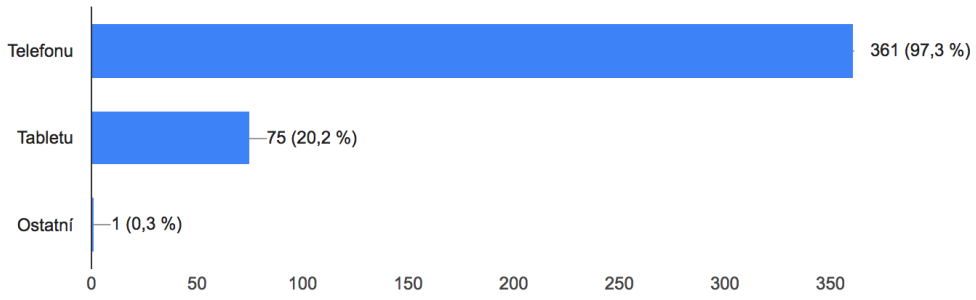
1. Je pro Vás mobilní aplikace MARAST zajímavou možností?



Obrázek 2.1: Diagram odpovědí na otázku č. 1 uživatelského průzkumu

Jak vidno z obrázku 2.1, pro bezmála 90 %, respondentů je mobilní verze zajímavou možností. Navíc pouze devět z celkové počtu čtyřiceti uživatelů se zápornou odpovědí po této otázce průzkum ukončilo. Z dalších odpovědí zbývajících respondentů lze usuzovat, že ačkoli mobilní aplikaci nepovažují za důležitou, mohli by se stát jejími uživateli.

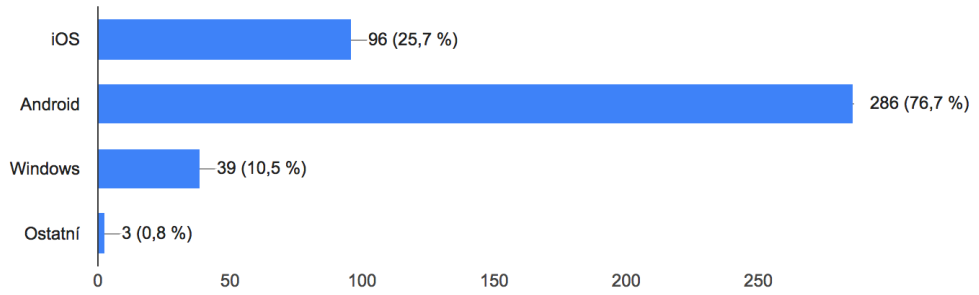
2. Mobilní aplikaci bych využil na následujících zařízeních:



Obrázek 2.2: Graf odpovědí na otázku č. 2 uživatelského průzkumu

Z odpovědí na tuto otázku je patrné, že naprostá většina tázaných, přesně 97,3 %, by aplikaci využívalo na mobilním telefonu. Oproti tomu pouhých 20 % z nich by aplikaci využívalo na tabletu. Podíl mobilních telefonů oproti tabletům je výrazně vyšší oproti původním předpokladům a je nutno ho zohlednit prioritizováním mobilních telefonů při návrhu uživatelského rozhraní aplikace. Vedle nižšího rozšíření vlastnictví tabletů lze jejich nižší podíl přikládat i předpokladu, že na tabletech je výrazně použitelnější i původní webová verze MARASTu. Přesně 65, tedy 17 %, uživatelů by pak mobilní aplikaci využilo na telefonu i tabletu současně.

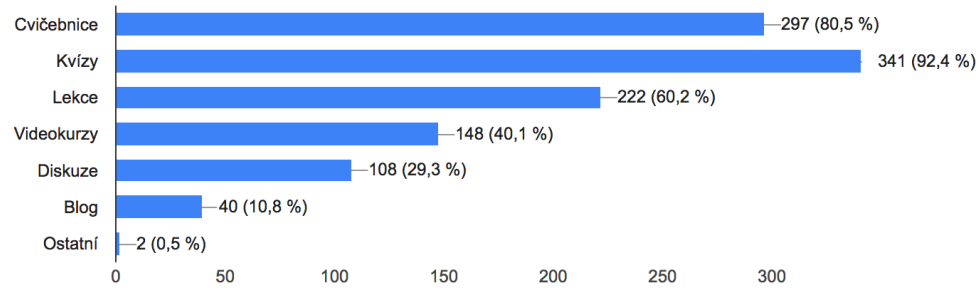
3. Mobilní aplikaci bych využil na operačním systému:



Obrázek 2.3: Graf odpovědí na otázku č. 3 uživatelského průzkumu

Na grafu 2.3 je znázorněno rozložení operačních systému, na kterých by tázaní uživatelé mobilní aplikaci využili. Z rozložení je nutné vyvodit, že je především aplikační rozhraní třeba navrhovat univerzálně tak, aby mohla být aplikace časem vyvinuta pro různé systémy. Přesně 49 uživatelů pak označilo více systémů současně, z čehož lze předpokládat, že by aplikaci využili zároveň na telefonu a tabletu s odlišnými operačními systémy.

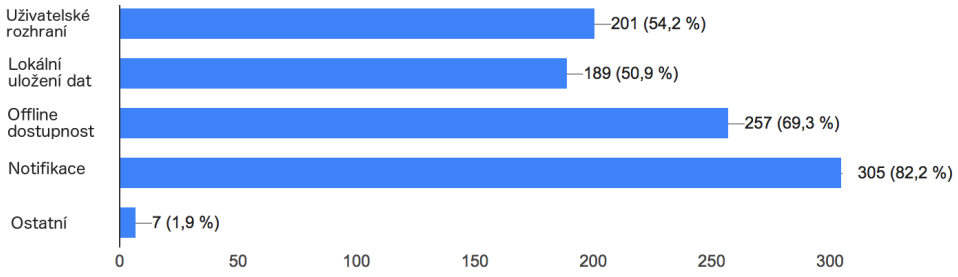
4. Označte prosím funkcionality webové aplikace MARAST, které by pro Vás byly důležité v mobilní verzi aplikace.



Obrázek 2.4: Graf odpovědí na otázku č. 4 uživatelského průzkumu

Graf na obrázku 2.4 znázorňuje funkcionality, které tázaní uživatelé označili jako zajímavé pro mobilní aplikaci. Vedle cvičebnice, lekcí a videokurzů, které jsou základními funkčními požadavky na aplikaci, by uživatelé ve většině ocenili především dostupnost kvízů. Mimo nabízených funkcionalit pak byla navhnuta možnost zobrazovaní kanálu příspěvků na sítích Facebook a Twitter souvisejících s MARASTem. Zbylé funkcionality s nižšími procenty, tedy diskuzi a blog, pak lze brát jako doplňkové s nižší prioritou.

5. Označte prosím funkcionality, které by pro Vás byly důležité coby přidaná hodnota mobilní aplikace.



Obrázek 2.5: Graf odpovědí na otázku č. 5 uživatelského průzkumu

Z odpovědí na tuto otázku vyplývá, že nejdůležitějšími funkcemi, které může mobilní aplikace oproti webové verzi nabídnout, jsou notifikace o blížících se termínech a novém obsahu a offline dostupnost. Uživatelské rozhraní aplikující zavedené standardy dané platformy a možnost lokálního ukládání osobních poznámek a dalších dat pak byly jako důležité označeny přibližně polovinou tázaných uživatelů. Mimo nabízených možností několik uživatelů uvedlo, že by ocenili označení vyhodnocených odpovědí v případě neúspěšně zodpovězené otázky v kvízu. Dále byla navržena volitelná možnost stahování videokurzů.

6. Zároveň uvítáme jakékoliv názory na funkční požadavky, vzhled a ovládání či další nápady na funkcionality, které třeba i v aktuální aplikaci MARAST chybí, ale s mobilním rozhraním by mohly být užitečné. Děkujeme.

V rámci doplňové otázky se sešlo celkem 27 odpovědí. Z těch šlo z většiny především o návrhy adresované na MARAST obecně, netýkajících se mobilní verze. Především několik uživatelů stejně jako v minulé otázce navrhuje označování špatných či správných odpovědí u kvízových otázek. Mimoto pět uživatelů, spadajících do množiny se zápornou odpovědí na otázků 2.1, rozporuje potřebu mobilní aplikace či celkově využití MARASTu na mobilních platformách.

V následující podsekci jsou citovány vybrané nejpřínosnější návrhy relevantní pro mobilní verzi.

Vybrané odpovědi

- *„Aplikace by dle mého názoru měla mít podobnou strukturu a vzhled jako webová verze MARASTu, tím se zvýší přehlednost a uživatel pak nebude mít problém přecházet plynule mezi aplikací a webem. Jako třešnička na dortu by bylo fajn, kdyby existovala možnost si vyexportovat deadliny z MARASTu do kalendáře, například na googlu.“*
- *„Responzivní vykreslovač latexu při zobrazení mobilní verze MARASTu.“*
- *„Bylo by dobré mít možnost stáhnout jen určité příklady z cvičebnice a lekcí, aby si pak člověk mohl např. prohlížet jen příklady s řešením po cestě do školy. Tedy přidat podobně inteligentní filtr jako je teď, ale s výběrem toho, co se stáhne“*
- *„Offline dostupnost videí - volitelná možnost stahovat je do zařízení). Aplikace by dle mého názoru měla mít podobnou strukturu a vzhled jako webová verze MARASTu, tím se zvýší přehlednost a uživatel pak nebude mít problém přecházet plynule mezi aplikací a webem.“*

2.1 Závěr

Odpovědi na otázky uživatelského průzkumu byly zdrojem množství poznatků shrnutých v předchozí sekci. Tyto poznatky byly reflektovány při výběru požadavků na aplikaci, které jsou obsahem následující kapitoly.

Kapitola 3

Rozbor požadavků

Na základě zadání, konzultací s tvůrci MARASTu, poznatků jeho uživatelů sesbíraných v uživatelském průzkumu a předchozí analýzy byl sestaven seznam funkčních a nefunkčních požadavků na mobilní aplikaci a aplikační rozhraní. Obsahem této kapitoly je výčet požadavků a jejich popis.

3.1 Požadavky na mobilní aplikaci

V této sekci budou popsány požadavky kladené na mobilní aplikaci.

3.1.1 Funkční požadavky

Funkční požadavky kladené na mobilní aplikaci jsou následující:

- F1. Přihlášení účtem ČVUT
- F2. Lekce
- F3. Cvičebnice
- F4. Kvízy
- F5. Notifikace
- F6. Offline režim

F1. Přihlášení účtem ČVUT

V MARASTu je k přihlášení využito účtu ČVUT autorizovaného pomocí protokolu OAuth 2.0, moderního autorizačního protokolu (resp. frameworku), který se stal de facto standardem pro zabezpečení RESTových webových služeb. Jsou jím zabezpečeny i vznikající fakultní back-end služby, jak je popsáno ve fakultní dokumentaci protokolu [10].

F2. Lekce

Mobilní aplikace bude schopna zobrazovat lekce, jejichž text je formátován značkovacím jazykem Markdown a vnořené matematické výrazy sázecím systémem LaTeX. Součástí lekcí mohou být i vnořené související příklady z cvičebnice a videa či seznami videí umístěné na službě YouTube. Ty by mělo být možno vhodnou formou přehrávat přímo v aplikaci.

F3. Cvičebnice

Mobilní aplikace bude schnopna zobrazovat cvičebnici obsahující příklady zadané pomocí systému LaTeX. Příklady mohou být několika různých typů popsaných v sekci 1.1.1. Součástí cvičebnice je možnost filtrování příkladů dle jejich obtížnosti, typu či stavu řešení. K jednotlivým příkladům v cvičebnici má uživatel možnost přidávat si vlastní poznámky. Součástí zadání je synchronizace těchto dat s webovou aplikací.

F4. Kvízy

V mobilní aplikaci bude možnost plnění kvízů, jejichž obsah a vyhodnocování bude synchronizováno s webovou aplikací. Mobilní aplikace by měla zpřístupňovat pouze omezené množství kvízů, tedy ponechat zápočtové a další významnější testy k plnění pouze ve webové aplikaci.

F5. Notifikace

Aplikace bude schnopna periodicky ze serveru aktualizovat svá data a notifikovat uživatele v případě dostupnosti nového obsahu. Zároveň by měla uživatele notifikovat o blížících se termínech, tedy především otevírání a zavírání kvízů. Možnost využití notifikací by měla být nastavitelná.

F6. Offline režim

Aplikace bude lokálně ukládat stažená data, které bude možnost prezentovat i v případě nedostupnosti internetového připojení. Zároveň by měla nabízet i možnost manuálního přechodu do offline režimu pro případ, kdy chce uživatel šetřit přenášená data či zrychlit načítání obsahu.

3.1.2 Nefunkční požadavky

Nefunkční požadavky kladené na mobilní aplikaci jsou následující:

- N1. Uživatelské rozhraní
- N2. Rozšiřitelnost
- N3. Kompatibilita s iOS 10 a vyšší

- N4. Jazyková mutace
- N5. Automatické aktualizace

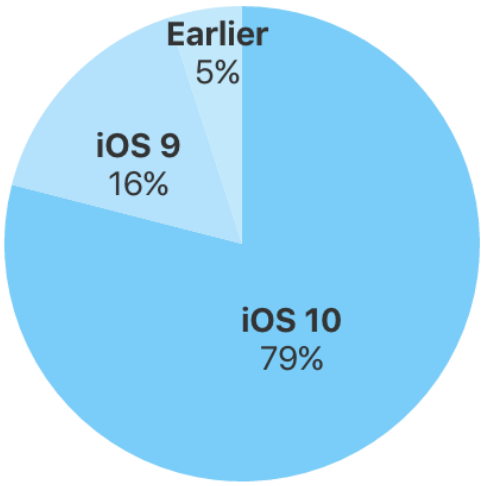
N1. Uživatelské rozhraní

Bude navrženo vhodné uživatelské rozhraní, které se bude řídit rozhraním webové aplikace a současně dodržovat zavedené standardy platformy iOS. Uživatelské rozhraní by se mělo automaticky přizpůsobovat různým velikostem zařízení se systémem iOS.

N2. Rozšiřitelnost

Aplikace by měla být vhodně navržena tak, aby mohla být rozšiřována. Především by tedy měla umožňovat snadné přidání nového typu příkladu.

N3. Kompatibilita s iOS 10 a vyšší



Obrázek 3.1: Aktuální rozložení verzí iOS, zdroj [11]

V době zadání práce je iOS 10 aktuální verzí systému, toho času používaného na 79 % aktivních zařízeních se systémem iOS, jak vidno na grafu na obrázku 3.1 z oficiálních statistik [11]. Jak demonstruje [12], pro platformu iOS je typické velmi rychlé šíření nových verzí systému a podpora maximálního množství zařízení, přičemž iOS 10 je podporován na všech iOS zařízeních novějších než iPhone 4 vyrobený v roce 2011 [13]. Verze iOS 10 byla zvolena jako minimální verze, kterou by měla aplikace podporovat.

N4. Jazyková mutace

MARAST je v současnosti ve webové verzi dostupný v české a anglické jazykové mutaci. Mobilní aplikace by tudíž měla podporovat tytéž jazyky jako webová verze.

N5. Automatické aktualizace

Aplikace by měla být schopna periodicky na pozadí aktualizovat svůj obsah tak, aby zobrazovala aktuální data i v případě spuštění při aktuální nedostupnosti internetového připojení. V souladu se standardy platformy iOS by měly být tyto aktualizace volitelné pro možnost šetření množství přenášených dat.

3.2 Požadavky na aplikační rozhraní

V této sekci budou popsány požadavky kladené na aplikační rozhraní.

3.2.1 Funkční požadavky

Funkční požadavky kladené na navrhované aplikační rozhraní vychází z funkčních požadavků na aplikaci. Aplikační rozhraní by tedy mělo zpřístupňovat potřebný obsah definovaný funkčními požadavky na aplikace, přičemž pro reprezentaci zdojů byl zvolen formát JSON [14], textový formát pro serializaci strukturovaných dat definovaný standardem [15].

3.2.2 Nefunkční požadavky

Dále jsou na aplikační rozhraní kladeny následující nefunkční požadavky:

- N6. Univerzalita
- N7. Rozšiřitelnost
- N8. Řízení přístupu

N6. Univerzalita

Aplikační rozhraní by mělo být navrženo nezávisle na vyvíjené klientské aplikaci tak, aby bylo možno využít i pro zamýšlené aplikace na dalších platformách.

N7. Rozšiřitelnost

Aplikační rozhraní by mělo být navrženo tak, aby do něj bylo možno podobným způsobem přidávat další případné funkce.

N8. Řízení přístupu

Aplikační rozhraní by mělo respektovat řízení přístupu přihlášeného uživatele k dostupnému obsahu, tedy neumožňovat uživateli přístup k obsahu, který by mu dle jeho nastavené role měl být skryt.

Kapitola 4

Návrh

Obsahem této kapitoly je návrh aplikačního rozhraní a mobilní aplikace. V kapitole budou shrnuty postupy a nástroje navržené pro realizaci kladených požadavků.

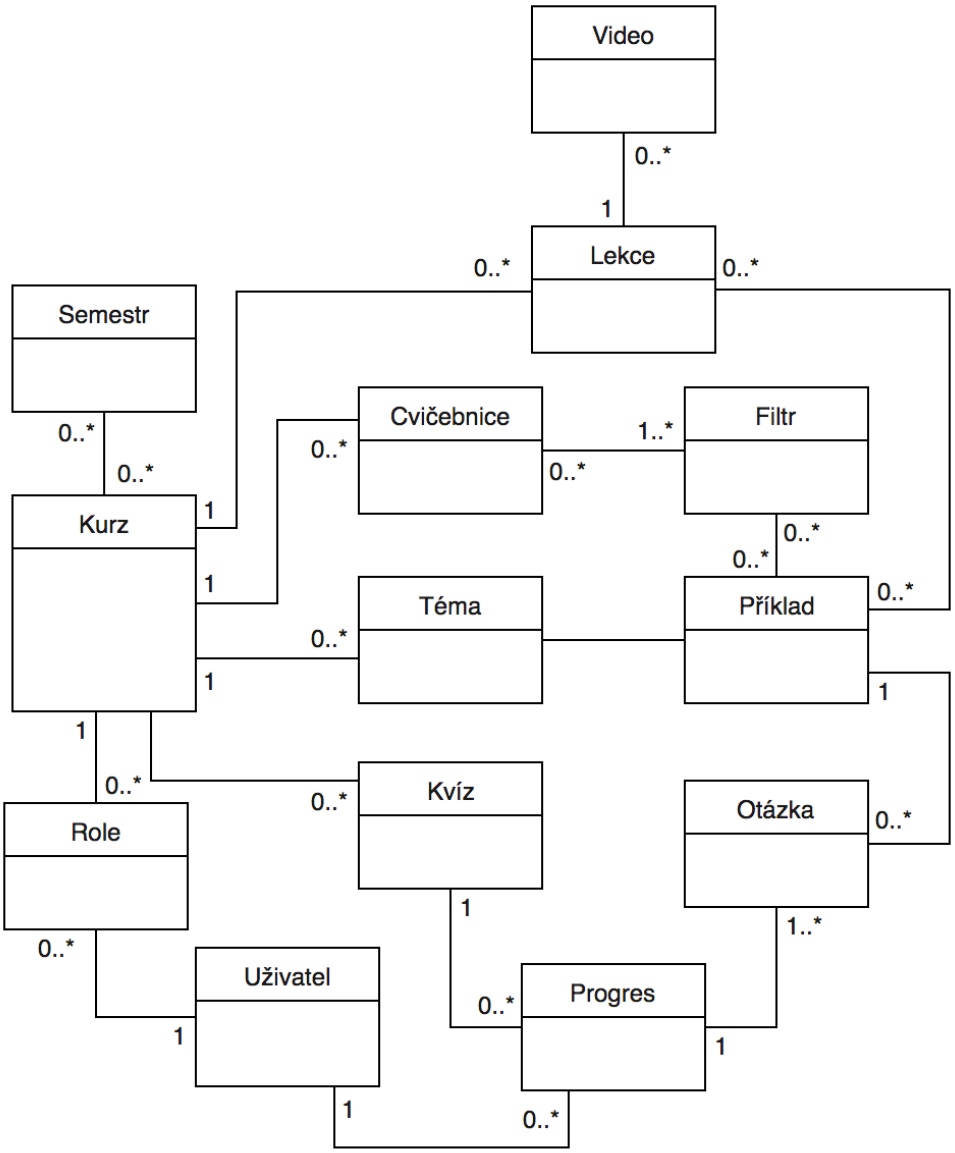
4.1 Doménový model

Doménový model zobrazený na obrázku 4.1 vychází z modelu webové aplikace MARAST a zachycuje vztahy mezi hlavními entitami aplikace vycházejícími z funkčních požadavků. Doménový model definuje logický obsah aplikace a slouží jako výchozí bod pro následující návrh.

4.2 Aplikační rozhraní

Aplikační rozhraní bylo navrženo na základě požadavků definovaných v sekci 3.2, tedy s využitím standardů REST architektury a reprezentací zdrojů ve formátu JSON. Jak popisuje [16], v kontextu aplikačních rozhraní webových služeb koncové body označují body dostupnosti definovaných zdrojů. Jak bylo zmíněno v sekci 1.3.4 popisující architekturu REST, zdroje mají čtyři následující metody přístupu odpovídající metodám protokolu HTTP.

- POST pro vytvoření zdroje
- GET pro získání zdroje
- PUT pro aktualizaci zdroje
- DELETE pro odstranění zdroje



Obrázek 4.1: Doménový model

4.2.1 Definované koncové body

V navrženém aplikačním rozhraní byly definovány následující koncové body, z nichž každý je představován unikátní URL adresou, na níž je daný zdroj dostupný. Části URL prefixované dvojtečkou zastupují proměnné specifikující konkrétní zdroj. Všechny následující zdroje jsou přístupné pouze po autorizaci přihlášeného uživatele a jejich obsah je řízen přístupem daného uživatele k danému zdroji. Data požadavků a odpovědí jsou shodně ve formátu JSON.

- GET /user

  Informace o přihlášeném uživateli. Vedle základních informací odpověď obsahuje také seznam rolí přihlášeného uživatele v dostupných kurzech.

- GET /user/:id

  Informace o uživateli s daným id.

- GET /courses/:code/lectures

  Seznam informací o dostupných přednáškách v kurzu s daným kódem. Součástí odpovědi není obsah jednotlivých přednášek.

- GET /courses/:code/exercises

  Seznam příkladů dostupných v cvičebnici kurzu s daným kódem.

- GET /courses/:code/quizzes

  Seznam informací o kvízech dostupných v kurzu s daným kódem.

- GET /courses/:code/topics

  Seznam témat příkladů v cvičebnici v kurzu s daným kódem.

- GET /lectures/:id/

  Informace o přednášce s daným id včetně jejího obsahu.

- GET /quiz/:id/progress

  Informace o progresu přihlášeného uživatele v kvízu s daným id.

- PUT /quiz_progress/:id/answer

  Aktualizace progresu přihlášeného uživatele v určitém kvízu, tedy zodpovězení aktuální otázky v progresu s daným id. Obsahem požadavku je odpověď na otázku. Její vyhodnocení je obsahem odpovědi na požadavek.

- PUT /notes/:id/

  Aktualizace poznámky k příkladu. Poznámka je určena daným id.

4.3 Autorizace aplikace

Jak bylo popsáno v sekci 3.1.1, k příhlášení uživatele je využito účtu ČVUT autorizovaného pomocí protokolu OAuth 2.0 [10]. Ve webové aplikaci MARAST je po přihlášení pro daného uživatele vytvořena session, jejíž stav je uchováván na serveru. Webový prohlížeč se na ní následně odkazuje pomocí uložené cookie, která je dle standardů HTTP protokolu zasílána s každým požadavkem [17].

Pro maximalizaci využití již vytvořených řešení je ideální využít pro autorizaci mobilní aplikace stejného způsobu, kterým je přihlášen webový prohlížeč. Samotné přihlášení uživatele je tedy navrženo pomocí vnořeného webového prohlížeče a volání následujících koncových bodů definovaných implementací protokolu OAuth 2 v Rails aplikacích jménem Doorkeeper [18]. Následující koncové body jsou již součástí webové aplikace MARAST a budou využity pro implementaci přihlášení mobilní aplikace.

- GET /auth/doorkeeper

  Tento koncový bod je volán pro iniciaci autorizace. Výsledkem volání koncového bodu ve vnořeném webovém prohlížeči je prezentace přihlašovacího formuláře, který je pro mobilní zobrazení plně responzivní.

- GET /auth/doorkeeper/callback

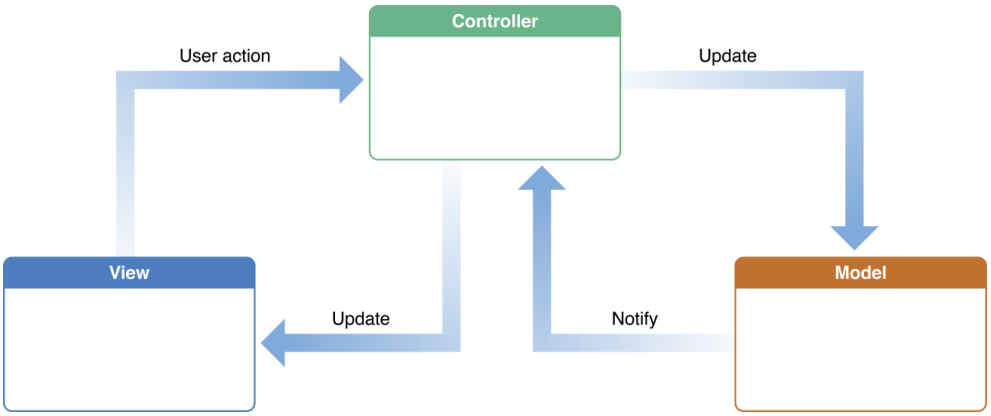
  Volání tohoto koncového bodu je výsledkem úspěšné autorizace a obsažená data v požadavku jsou využita pro inicializaci uživatelské session. Obsahem odpovědi jsou informace o cookie pro vytvořenou session.

- DELETE /logout

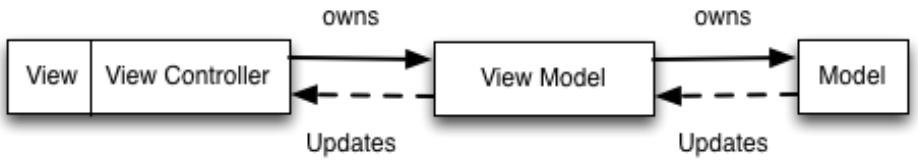
  Koncový bod určen pro zrušení aktuální uživatelské session.

4.4 Architektura aplikace

Pro architekturu iOS aplikací je samotnou firmou Apple propagován [19] návrhový vzor Model-View-Controller, kterému je uzpůsobem i návrh systémových komponent. Implementace vzoru MVC na platformě iOS, jejíž schéma je znázorněno na obrázku 4.2, však není příliš šťastná a typicky se s ní pojí velmi obsáhlé a těžko testovatelné třídy controller vrstvy, pro které se ustálilo přízvisko Massive View Controller [20]. Z toho důvodu je vhodná spíše pro opravdu minimalistické aplikace a vhodnějším vzorem architektury je z MVC odvozený vzor Model-View-ViewModel, který je navržen jako architektura navrhované aplikace.



Obrázek 4.2: Schéma vzoru MVC dle společnosti Apple, zdroj [19]



Obrázek 4.3: Schéma vzoru MVVM, zdroj [21]

4.4.1 Model-View-ViewModel

Model-View-ViewModel, zkráceně MVVM, je návrhový vzor architektury aplikací odvozený od vzoru MVC. Jeho diagram dle [21] je znázorněn na obrázku 4.3. MVVM oproti MVC upozaďuje controller, jehož odpovědnosti zčásti přebírá vrstva zvaná view model. Zodpovědnosti jednotlivých vrstev architektury MVVM nejsou zcela pevně definovány a mohou se lišit v různých výkladech a přizpůsobovat konkrétním potřebám aplikace. Dle [22] jsou vrstvy MVVM definovány následovně:

Model

Model reprezentuje datovou vrstvu aplikace a jeho zodpovědnosti jsou shodné s architekturou MVC.

View

View je vrstva reprezentující uživatelské rozhraní a jeho ovládací prvky. Neměla by přímo interagovat s datovými modely, ale pouze přijímat zpracovaná

data v prezentovatelné formě a parametry řídící jejich prezentaci. View je tak nezávislý na datové vrstvě, což přispívá k jeho zvýšené modularitě.

Controller

Controller je určen k nastavování stavu uživatelského rozhraní a zpracování uživatelských interakcí. Controller by stejně jako view neměl přímo interagovat s datovou vrstvou a namísto toho nechat zpracování dat na view modelu.

View model

View model představuje mezivrstvu mezi view a modelem a odpovídá za zpracování logiky daného view. Typicky tedy volá metody modelových tříd a poskytuje view zpracovaná data v prezentovatelné formě. Současně je také stav a business logika daného view součástí view modelu.

4.5 Návrhové vzory

V této sekci budou shrnuty další návrhové vzory využité při návrhu aplikace. Následující vzory budou využity pro strukturování kódu aplikace.

4.5.1 Koordinátory

Nedostatkem architektury MVC, respektive MVVM v mobilních aplikacích, je navigace mezi jednotlivými komponentami, kdy není definováno, jakým způsobem kód související s navigací strukturovat. Pro tento účel jsou v moderních aplikacích využívány komponenty zvané koordinátory nebo též flow controllers. Princip koordinátorů je založen na návrhovém vzoru application controller [23] a jejich využití při návrhu iOS aplikací ve formě, která je využívána v dnešních aplikacích, popsal Soroush Khanlou ve své přednášce [24]. V kombinaci s architekturou MVVM jsou koordinátory zodpovědné za inicializaci jednodlivých komponent a vztahů mezi nimi. Začlenění koordinátorů do kontextu architektury MVVM bývá též označováno zkratkou MVVM-C [25].

4.5.2 Dependency Injection

Dependency injection, zkráceně DI, je návrhový vzor pro předávání závislostí mezi jednotlivými komponentami aplikace tak, aby byly vzájemně použitelné. Dependency injection spočívá v definici závislostí ve speciální komponentě zvané kontejner, která je zodpovědná za sestavení grafu závislostí a následné předávání potřebných závislostí daným komponentám.

4.6 Datová vrstva

Vzhledem k požadavku offline dostupnosti aplikace je třeba data persistentně ukládat. Pro správu datové vrstvy aplikace byl zvolen framework Core Data [26]. Ke Core Data existuje i řada alternativních řešení jako je Realm [27] či další podobné frameworky třetích stran, které ač mohou být pro určité specifické požadavky vhodnější, CoreData má zásadní výhodu v pokročilé integraci s vývojářskými nástroji iOS SDK a je pro účely navrhované aplikace dostačující.

Ve zbytku sekce je popsán návrh datové vrstvy.

4.6.1 Datový model

Datový model aplikace, jehož diagram se nachází na obrázku 4.4, vychází z doménové modelu a struktury dat poskytovaných aplikačních rozhraním. Jedná se o Entity-Attribute-Value model, který je základem pro datovou vrstvu spravovanou frameworkem Core Data.

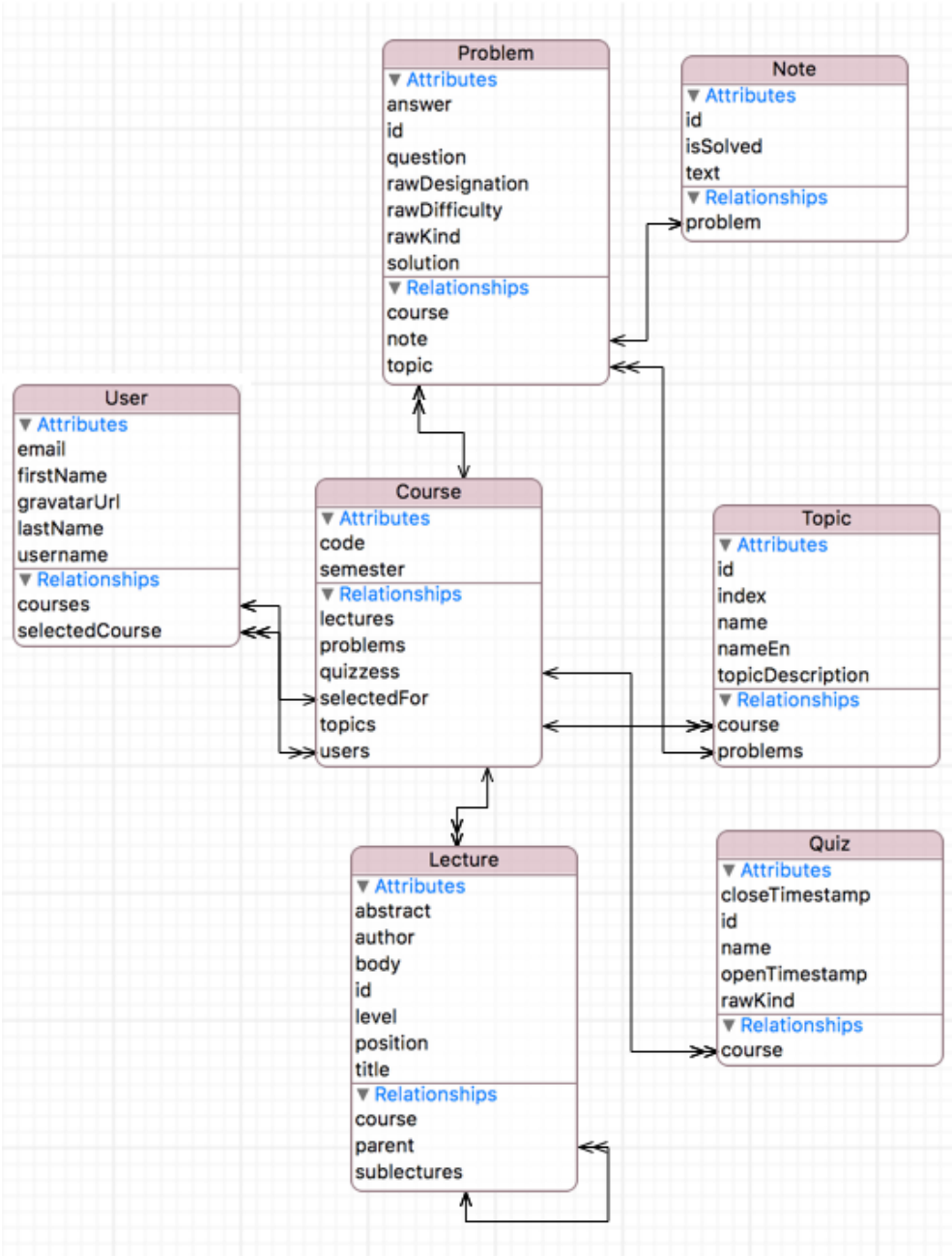
4.6.2 Entity–Attribute–Value model

Entity-Attribute-Value model (EAV), je typ datového modelu pro řízení ukládání dat. Jehož princip [28] popisuje následovně:

> *„Princip EAV spočívá v modelování vlastností objektu jako jednotlivých řádků v databázi. Každý řádek v tabulce reprezentuje jediný fakt (na rozdíl od konvenčního modelování, kdy pro každou vlastnost je vytvořen sloupec databázové tabulky a řádek pak obsahuje více faktů – jeden pro každý sloupec). Jak již z názvu Entity–Attribute–Value vyplývá, jsou data modelována jako uspořádané trojice (entity, attribute, value), kde entity představuje prvek z množiny entit (definovaných objektů), attribute je prvek z množiny atributů entity a value představuje hodnotu tohoto atributu.“*

4.6.3 Core Data

Core Data je framework pro správu datové vrstvy aplikace, jehož základem je SQLite databáze, do které jsou data persistentně ukládána. Nad databází je na základě datového modelu a definice modelových tříd vystavěna objektová vrstva, jejímž prostřednictvým programátor k datům přistupuje a je tak odstíněn od klasických databázových operací. Ač se nejedná o klasickou objektově relační databázi, je princip Core Data přirovnatelný k technologii Active Record popsané v sekci 1.3.5.



Obrázek 4.4: Schéma datového modelu

4.7 Synchronizace

Požadavkem na aplikaci je lokální dostupnost obsahu při nedostupném připojení a notifikace uživatelů na nový obsah. Pro tyto účely je třeba aplikaci periodicky na pozadí synchronizovat s databází webové verze. Časované spouštění kódu synchronizace samotnou aplikací není vhodným způsobem, jelikož běh aplikace může být v jakémkoliv okamžiku přerušen uživatelem či systémem. Právě pro účely periodické synchronizace dat na pozadí nabízí systém iOS rozhraní Background Fetch.

4.7.1 Background fetch

Background fetch je systémové rozhraní pro periodické spouštění kódu aplikace. Rozhraní je předána požadovaná délka periody a není zaručeno ve kterém okamžiku bude synchronizace vyvolána. Po dokončení synchonizace je v odpovědi očekávána informace o tom, zda-li během ní došlo ke stažení nových dat. Na základě této informace a dalších faktorů pak systém sám určuje, jak často a v jaký okamžik bude aplikace na pozadí vyvolávána. Tyto faktory a další specifika rozhraní Background fetch byly vývojáři popsány v přednášce [29] na konferenci WWDC 2013 a patří mezi ně především:

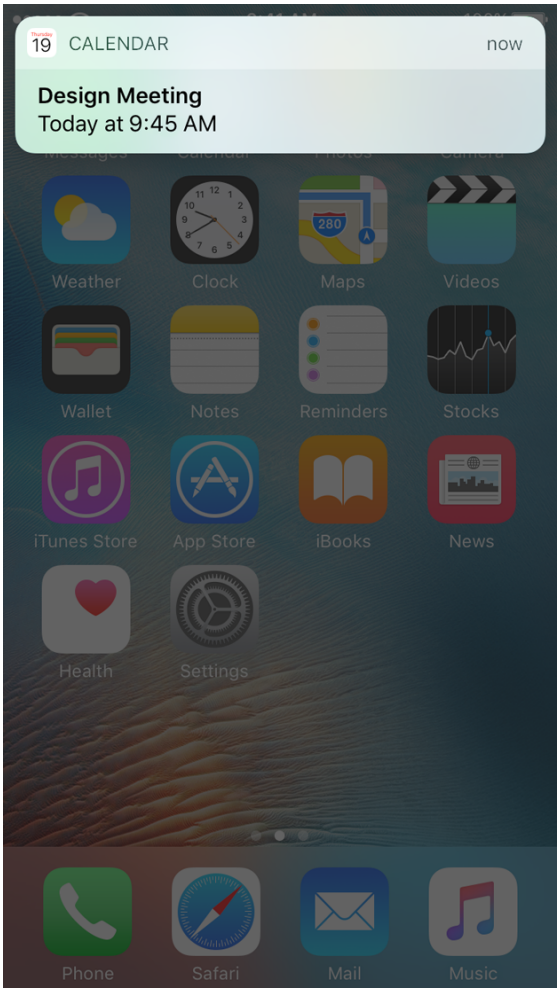
- Dostupnost připojení v daný okamžik
- Jak často aplikace úspěšně stáhne nová data
- Množství dat přenesených během předchozích synchronizací

Synchronizaci dat je tedy třeba navrhovat tak, aby byla nezávislá na době jejího spouštění a tak, aby aplikace nebyla závislá na jejím provedení. Doba synchronizace je navíc časově omezena, přičemž limit je nastaven přibližně na 30 vteřin. Background fetch tedy není vhodný pro stahování větších objemů dat.

4.8 Notifikace

Pro upozorňování uživatelů na nový obsah a další události je možno využít systémových notifikací ilustrovaných na obrázku 4.5. Součástí notifikace je nadpis, text a počínaje verzí iOS 10 lze k notifikacím připojovat také obrázky, videa a jednoduché uživatelské rozhraní.

Zobrazení notifikací lze vyvolat přímo v okamžik provádění kódu nebo lze pro ně nastavit několik typů spouštěčů jako je daný čas či dosažení určené lokace. Notifikace o novém obsahu je tedy vhodné vyvolat ve chvíli dokončení synchronizace popsané v sekci 4.7. Notifikace o blížících se termínech ve studovaném kurzu je vhodné časovat s vhodným předstihem k danému termínu.



Obrázek 4.5: Ukázka systemové notifikace, zdroj [30]

Kapitola 5

Návrh uživatelského rozhraní

V této kapitole budou shrnuta specifika uživatelského rozhraní iOS aplikací a následně popsán postup návrh obrazovek aplikace.

5.1 Specifika platformy

Firma Apple si výrazně zakládá na kontinuitě aplikací na své platformě iOS a doporučené postupy pro uživatelská rozhraní podrobně popisuje v přehledu s názvem iOS Human Interface Guidelines[31].

5.1.1 UIKit

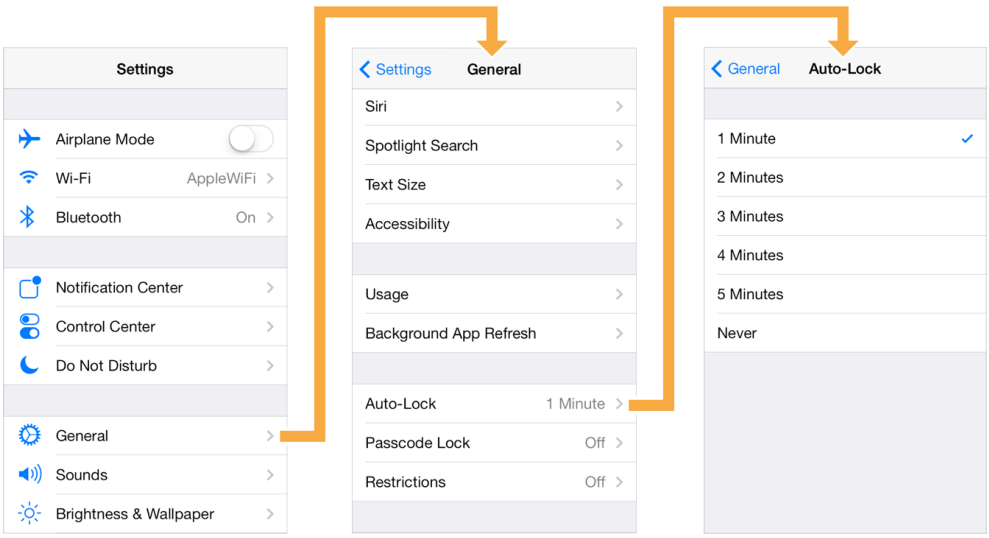
Standardní framework obsahující infrastrukturu pro budování uživatelského rozhraní na platformě iOS se nazývá UIKit. Jeho základem jsou elementární třídy **UIView** představující obdélníkovou oblast na obrazovce a **UIViewController**, kontejner spravující instance UIView. O instancích třídy UIViewController lze pro zjednodušení hovořit jako o obrazovkách aplikace.

5.1.2 Navigace

Klíčovou části uživatelského rozhraní je navigace mezi obrazovkami, pro kterou jsou typicky užívány následující prvky:

UINavigationController

UINavigationController je kontejner obsahující zásobník vnořených obrazovek. Jeho fungování dle [32] je ilustrováno na obrázku 5.1. Na něm lze vidět, že základem kontejneru je vrchní navigační lišta, jejíž součástí je v levé části tlačítko pro návrat na předchozí obrazovku a uprostřed nadpis aktuální obrazovky. V právé části lišty navíc mohou být umístěny tlačítka akcí dostupných na aktuální obrazovce.



Obrázek 5.1: Ilustrace fungování kontejneru UINavigationController na systémovém nastavení, zdroj [32]

UITabBarController

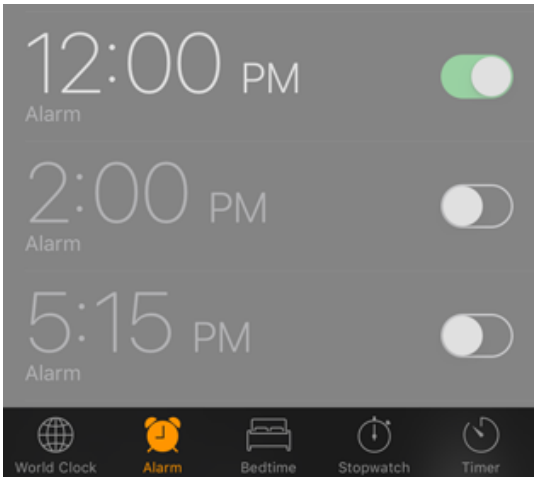
UITabBarController je dalším typem kontejneru obrazovek, které v něm tvoří záložky, mezi nimiž je možno opakovaně přepínat. Jeho základem je spodní navigační lišta pro ilustraci zobrazená na obrázku 5.2 z oficiální dokumentace [33]. Součástí lišty jsou ikony s názvy jednotlivých záložek sloužící jako tlačítka pro přepínání, přičemž tlačítko aktivní záložky je proti ostatním graficky zvýrazněno.

5.2 Navigace

Základem návrhu uživatelského rozhraní aplikace je návrh navigace mezi jejími obrazovkami. Z obsahu aplikace jak byla navržena v předchozích kapitolách vyplývá potřeba řešit několik následujících úrovní navigace:

1. Navigace mezi jednotlivými kurzy
2. Navigace mezi částmi kurzu, tedy cvičebnicí, kvízem a lekcemi
3. Navigace v rámci jednotlivých částí kurzu

Na základě těchto potřeb bylo pro navigaci v aplikaci navrženo uživatelské rozhraní zobrazené na obrázku 5.3. Na něm lze vidět využití obou navigačních lišt představených v sekci 5.1.2. Vrchní navigační lištou je řešena vnitřní



Obrázek 5.2: Navigační lišta kontejneru UITabBarController, zdroj [33]

navigace v rámci zvolené části kurzu a spodní lištou jejich volba. Navíc bylo přidáno vysouvací postranní menu pro navigaci mezi jednotlivými kurzy.

5.3 Navigační hierarchie

Z výše navržené navigace v aplikaci vychází návrh navigační hierarchie zobrazené na obrázku 5.4. Na něm jsou k vidění následující navigační komponenty:

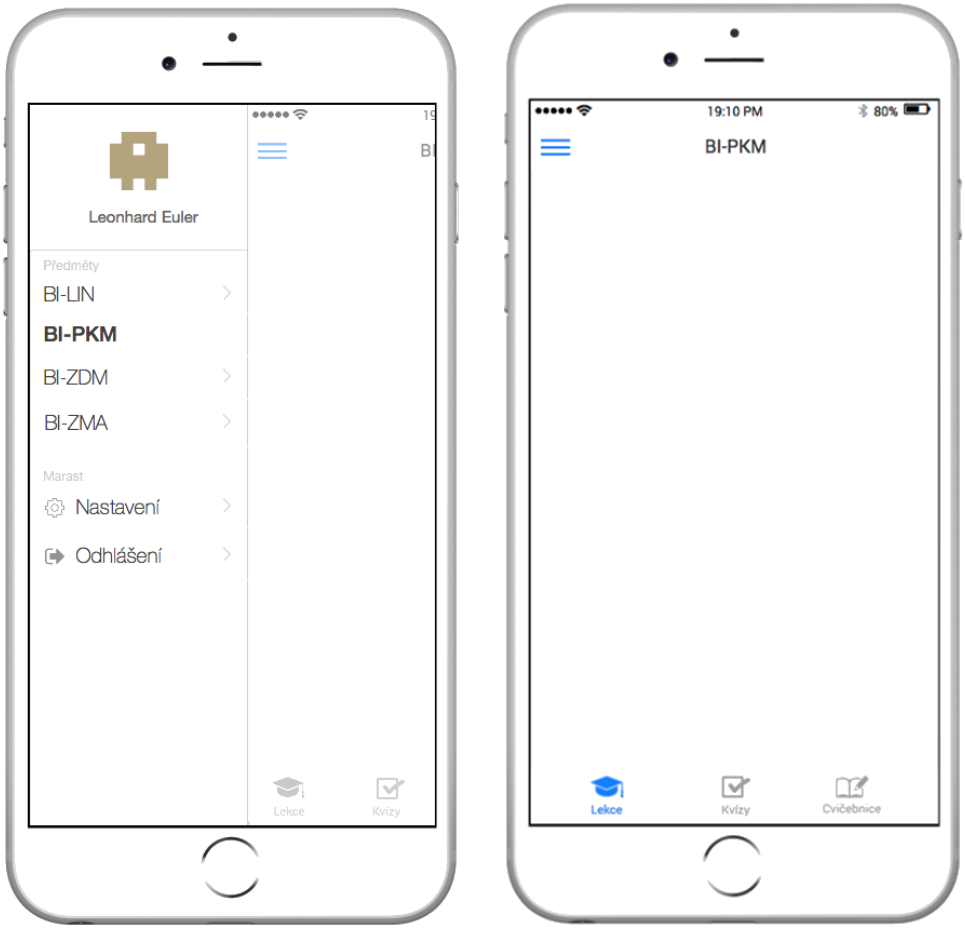
1. UINavigationController pro navigaci mezi hlavním obsahem a dalšími obrazovkami jako je nastavení aplikace.

2. SlideMenuController coby kontejner obsahující postranní menu a hlavní obsah aplikace.

3. UITabBarController - kontejnery jednotlivých kurzů. Každa záložka kurzu je navíc tvořena kontejnerem UINavigationController pro vnitřní navigaci uvnitř záložky. Z důvodu zanoření kontejnerů UINavigationController je třeba v momentě zobrazení kontejneru nižší úrovně skrýt navigační lištu kontejneru vyšší úrovně.

5.4 Návrhy obrazovek

Na základě funkčních požadavků byl sestaven návrh uživatelského rozhraní. Obsahem této sekce je popis jednotlivých navržených obrazovek aplikace. Schémata obrazovek zobrazené v této sekci jsou ve vyšším rozlišení dostupné na přiloženém médiu.



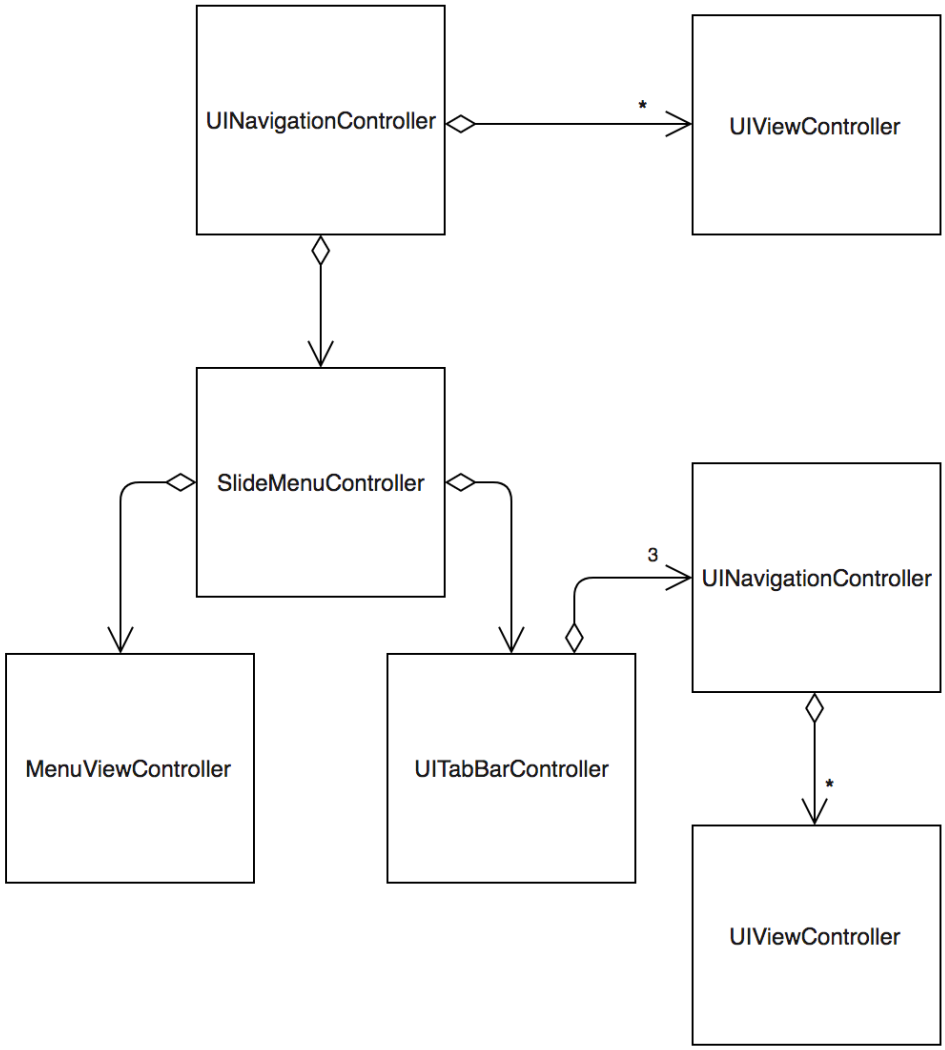
Obrázek 5.3: Uživatelské rozhraní navigace v aplikaci.

5.4.1 Přihlášení

Schéma přihlašování na obrázku 5.5 zobrazuje úvodní obrazovku aplikace ve výrazném provedení inspirovaném podobnou obrazovkou z aplikací jako Twitter, Facebook či Instagram. Z úvodní obrazovky je ve vnořeném webovém okně dostupná přihlašovací obrazovka ČVUT účtu, jejíž design je plně responzivní.

5.4.2 Lekce

Schéma uživatelského rozhraní lekcí na obrázku 5.6 zobrazuje obrazovku výběru ze stromu lekcí. Obrazovka vybrané lekce vedle textu a vnořených příkladů zahrnuje seznam přiložených YouTube videí včetně náhledového obrázku sloužícího jako tlačítko pro spuštění videa. Přehrávané video je připnuto ve



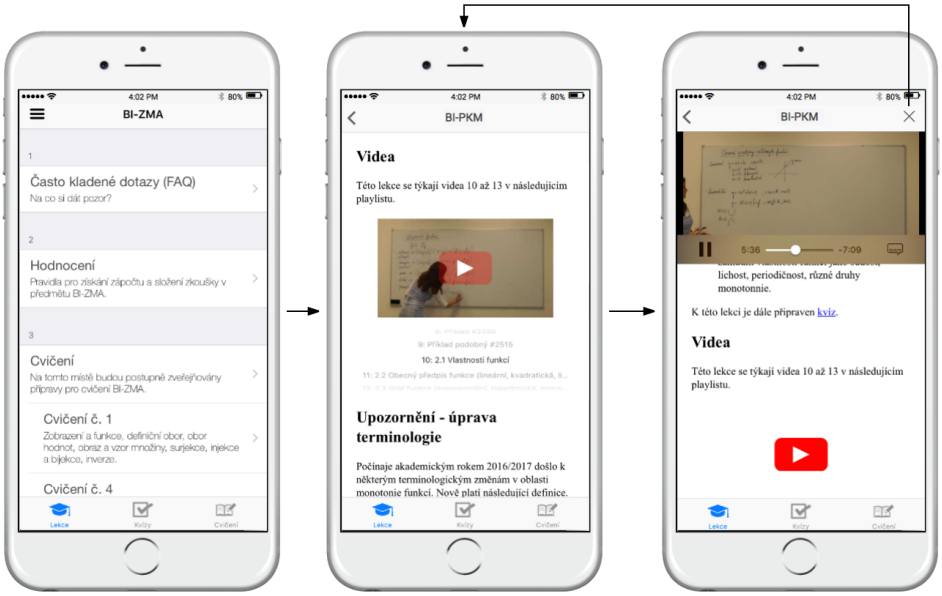
Obrázek 5.4: Schéma navigační hierarchie v aplikaci

Obrázek 5.5: Schéma uživatelského rozhraní přihlašovaní

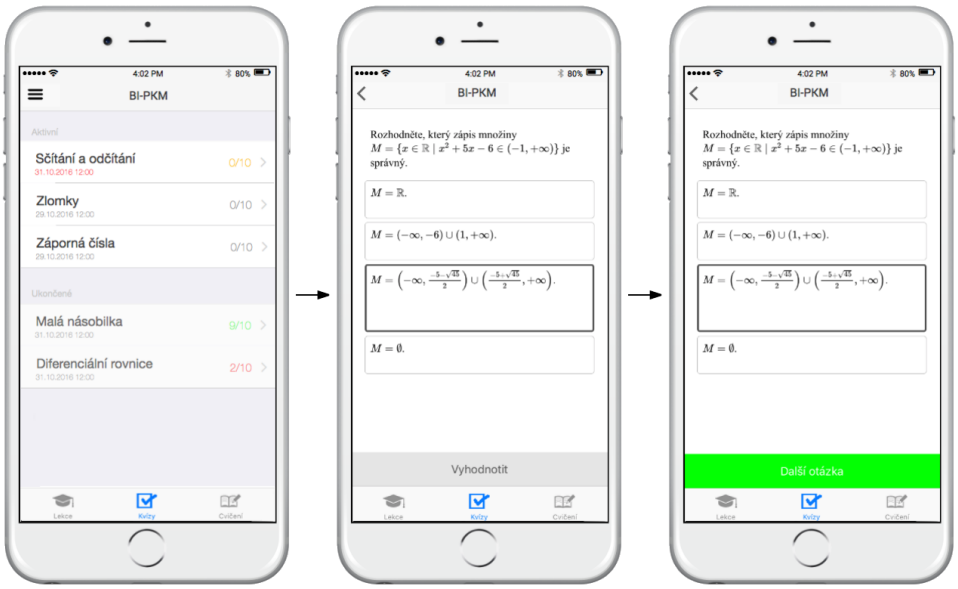
vrchní části obrazovky, přičemž je i během jeho přehrávání možno nadále procházet lekci. Okno videopřehravače navíc obsahuje ovládací prvky včetně tlačítka pro roztažení videa na celou obrazovku.

5.4.3 Kvízy

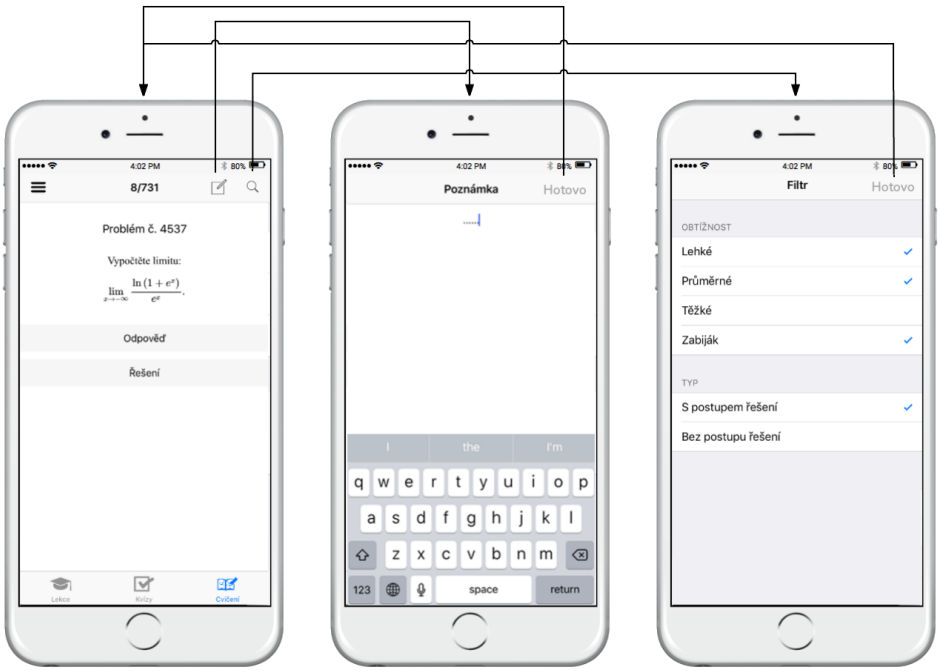
Schéma uživatelského rozhraní na obrázku 5.7 kvízů zobrazuje obrazovku zahrnující výběr dostupných kvízů a přehled hodnocení kvízů ukončených. Obrazovka zvoleného kvízu je tvořena aktuální kvízovou otázkou s výběrem odpovědí ze čtyř možností. Potvrzením odpovědí je zobrazeno vyhodnocení otázky, po němž následuje další otázka či konec kvízu.



Obrázek 5.6: Schéma uživatelského rozhraní lekcí



Obrázek 5.7: Schéma uživatelského rozhraní kvízů



Obrázek 5.8: Schéma uživatelského rozhraní cvičebnice

5.4.4 Cvičebnice

Schéma uživatelského rozhraní cvičebnice na obrázku 5.8 zobrazuje obrazovku s vybranými příklady, mezi nimiž je možno horizontálními gesty listovat. Navigační lišta obrazovky zahrnuje indikátor listování a tlačítka akcí odkazujících na obrazovku poznámky k aktuálnímu příkladu a obrazovku výběru filtru příkladů.

Kapitola 6

Realizace

V následující kapitole budou shrnuty nástroje a postupy využité při vývoji aplikace.

6.1 Vývojové prostředí

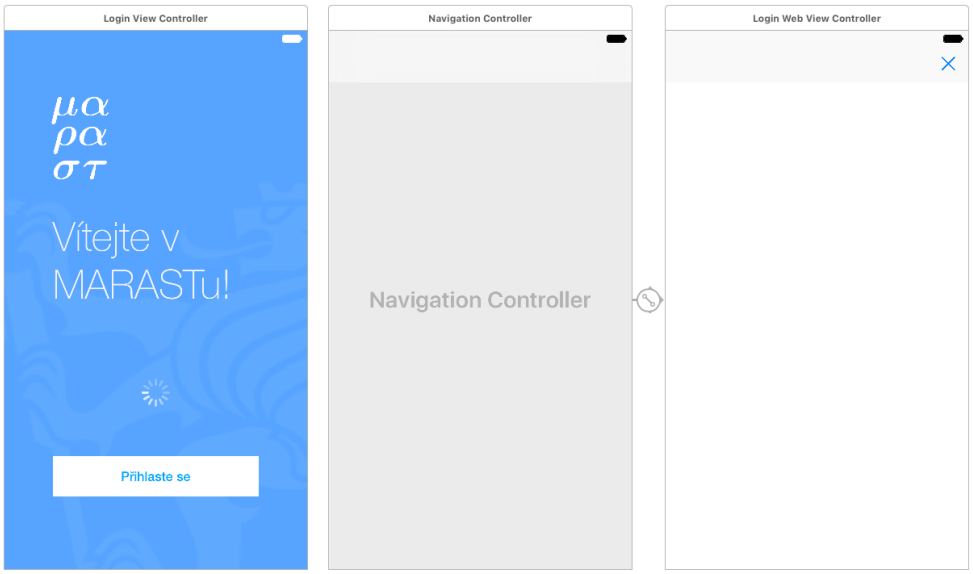
Jediné oficiálně podporované prostředí pro vývoj iOS aplikací je **Xcode** vyvíjený firmou Apple výhradně pro vlastní operační systém macOS. Součástí prostředí Xcode je simulátor aktuálních iOS zařízení, návrhář uživatelského prostředí a řada dalších podpůrných vývojářských nástrojů.

6.2 Programovací jazyk

V současné době je možno nativní iOS aplikace vyvíjet ve dvou různých jazycích, které díky vzájemné interoperabilitě mohou být využity v rámci jedné aplikace současně. Prvním z jazyků je **Objective-C**, vyvinutý počátkem osmdesátých let coby nadmnožina jazyka C. Druhým jazykem je **Swift**, přestavený firmou Apple v roce 2014. Výhodou Objective-C je dlouhodobá stabilita a odladěná podpora v programátorských nástrojích. Volba tohoto jazyka může být vhodná pro rozsáhlé aplikace z důvodu nutnosti každoroční migrace kódu na novou verzi jazyka Swift, který v současnosti stále probíhá dynamickým vývojem. Na druhou stranu Swift oplývá výrazně jednodušší a přehlednější syntaxí a práce v něm je jednoznačně rychlejší a příjemnější. Z toho důvodu byl zvolen jako primární jazyk pro vývoj aplikace.

6.3 Uživatelské rozhraní

K implementaci uživatelského rohraní iOS aplikací je vedle manuální definice v rámci kódu možno využít nástrojů zvaných **storyboard**. Jde o soubory



Obrázek 6.1: Storyboard návrhu úvodních přihlašovacích obrazovek zobrazený v režimu grafického rozhraní

ve formátu XML obsahující strukturovanou definici prvků uživatelského rozhraní, jejich vlastností, vzájemných vztahů a mapování prvků a jejich akcí na odpovídající rozhraní v kódu aplikace. V rámci prostředí Xcode je možné tyto soubory upravovat s využitím přehledného grafického rozhraní. Ukázkový storyboard zobrazený v grafickém rozhraní je zobrazen na obrázku 6.1. Jedná se o storyboard využitý pro návrh uvodních přihlašovacích obrazovek

6.4 Lokalizace

Pro implementaci lokalizace je využíváno makra **NSLocalizedString**, kterému je jako argument předáván klíč požadavaného textu. Na základě výskytů těchto maker je možno automaticky vygenerovat lokalizační soubory pro jednotlivé podporované jazyky fungující jako slovníky ve formátu klíč - hodnota. Lokalizované texty je zároveň možno exportovat do lokalizačních souborů i přímo ze storyboardů. Použitý lokalizační soubor je následně za běhu aplikace zvolen systémem iOS dle uživatelského nastavení.

6.5 Správa závislostí

Pro správu knihoven třetích stran je využito manažeru závislostí **Carthage** [34]. Jeho základem je konfigurační soubor s názvem Cartfile umístěný v koře-

novém adresáři repozitáře projektu. Součástí tohoto souboru je seznam adres repozitářů požadovaných knihoven a jejich verzí. Carthage dle tohoto seznamu sestaví graf závislostí, jejichž repozitáře stáhne a následně z nich knihovny sestaví ve formě dynamických frameworků. Dynamické frameworky, představeny na přednašce [35] na konferenci WWDC 2014, jsou typy knihoven, které nejsou přímou součástí aplikace, ke které jsou pouze připojeny a dynamicky načítány v momentě použití.

6.6 Použité knihovny a frameworky

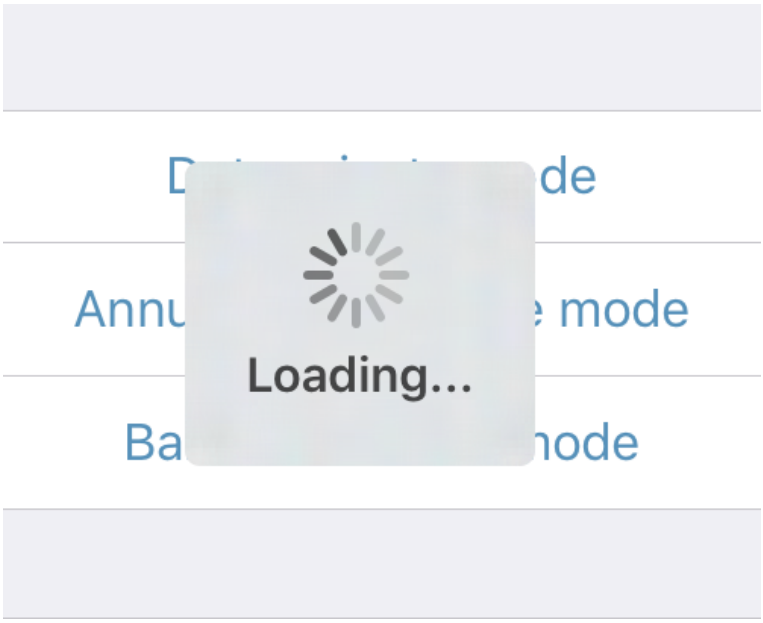
V následující sekci jsou shrnuty použité knihovny a frameworky třetích stran a jejich využití při realizaci aplikace. Jejich seznam včetně licencí a zdrojů je uveden v tabulce 6.1.

Tabulka 6.1: Použité knihovny

Název knihovny	**Licence**	**Zdroj**
MathJax	Apache License 2.0	[36]
SwiftyJSON	MIT License	[37]
MBProgressHUD	MIT License	[38]
XCGLogger	MIT License	[39]
SlideMenuControllerSwift	MIT License	[40]
Swinject	MIT License	[41]
Down	MIT License	[42]
XCDYouTubeKit	MIT License	[43]
Result	MIT License	[44]
Reachability.swift	MIT License	[45]

6.6.1 MathJax

MathJax je javascriptové rozšíření webového prohlížeče určené pro vykreslování matematických formulí zadaných ve formátu systému LaTeX. Jak bylo zmíněno v sekci 1.3.6, MathJax je pro stejný účel využit i ve webové verzi MARASTu, kde je v prohlížeči stahován společně s webovou stránkou. V mobilní aplikaci je však třeba podporovat vykreslování matematiky i v offline režimu, tudíž je třeba MathJax distribuovat v rámci aplikace. Vzhledem k výrazné velikosti distribuce MathJaxu jej bylo třeba ořezat o nepotřebné skripty, písma a další nevyužité zdroje. Ve výsledku tedy distribuce MathJaxu přidává aplikaci na velikosti 7 megabytů, což je přijatelná velikost vzhledem k zásadní funkci, který MathJax v aplikaci plní.



Obrázek 6.2: Indikátor aktivity MBProgressHUD, zdroj [38]

6.6.2 SwiftyJSON

SwiftyJSON je framework umožňující výrazně jednodušší a přehlednější práci s daty ve formátu JSON v jazyce Swift. V aplikaci je využit pro zpracování dat při interakci s aplikačním rozhraním.

6.6.3 MBProgressHUD

MBProgressHUD je jednoduchý indikátor aktivity využitý napříč aplikací v momentech načítání dat. Indikátor je možno využít v několika stylech a připojovat k němu informativní text. Ukázka indikátoru MBProgressHUD je uvedena na obrazku 6.2. Indikátor je v aplikaci zobrazován v momentech stahování a zpracování dat.

6.6.4 XCGLogger

XCGLogger je framework umožňující přehledné logování aktivity aplikace do terminálu, systémové konzole, souborů či dalších možných lokací. Framework je v aplikaci použit pro logování její aktivity především pro účely usnadnění vývoje.

6.6.5 SlideMenuControllerSwift

SlideMenuControllerSwift je hotová implementace vysunovacího postraního menu v jazyce Swift. Za programátora automaticky řeší gesta pro vysouvání a zasouvání menu a zároveň i odsazení, zatmavení či další podobné stylování spodní obrazovky v případě překrytí postraním menu. V aplikaci je tato knihovna použita pro implementaci postraního menu.

6.6.6 Swinject

Swinject je implementace kontejneru návrhového vzoru Dependency injection zmíněného v sekci 4.5.2. Swinject je implementován v jazyce Swift a jednoduše a přehledně využívá funkcí a syntaxe tohoto jazyka. Swinject je v aplikaci využit pro správu závislostí, čímž je dosaženo přehledného strukturování kódu a jeho testovatelnosti.

6.6.7 Down

Down je framework využitý pro zpracování textu ve formátu Markdown a jeho převod do formátu HTML. V aplikaci je tento framework využit pro převod formátu textu lekcí.

6.6.8 XCDYouTubeKit

XCDYouTubeKit je implementace přehravače videí ze sítě YouTube využívající systemového přehravače. Videa je možno přehrávat ve vnořeném okně s možností roztažení na celou obrazovku. Přehravač je v aplikaci využit pro přehrávání videí v rámci zobrazovaných lekcí.

6.6.9 Result

Result je minimalistický framework výrazně syntakticky zpřehledňující práci s výsledky asynchronních operací. Je využit pro přehledné a pohodlné strukturování kódu.

6.6.10 Reachability.swift

Reachability.swift je knihovna zjednodušující práci se systemovým rozhraním pro zjišťování stavu dostupnosti síťových rozhraní. Knihovna je využita pro zjišťování dostupnosti serveru.

Kapitola 7

Testování

V této kapitole bude popsáno testování mobilní aplikace a vyvíjeného aplikačního rozhraní webového MARASTu. Pro průběžné nasazování vyvíjených částí aplikačního rozhraní byl využit testovací server obsahující vzorek dat z produkční databáze. Proti testovacími serveru byla mobilní aplikace průběžně testována. Pro okamžité testování vyvíjených funkcí aplikačního rozhraní byla navíc využívána lokální instance serveru s neomezenou možností manipulace s daty v databázi.

V následujících sekcích kapitoly budou shrnuty využité metodiky testování.

7.1 Testování aplikačního rozhraní

Součástí práce byl vývoj aplikačního rozhraní webového MARASTu. Jak bylo přestaveno v sekci 1.3, MARAST je vyvíjen v jazyce Ruby a staví na technologii Ruby on Rails. Pro testování funkcí Ruby on Rails aplikací je využíván nástroj **RSpec** určený pro testování metodikou Behaviour-Driven-Development (BDD), neboli vývojem řízený chováním.

Vývoj řízený chováním je přístup odvozený od vývoje řízeného testy (Test-Driven-Development) představený Danem Northem, který ve svém článku [46] popisuje omezení termínu test a jeho nahrazení termínem chování, v němž je lépe zahrnut testovaný kontext. Vývoj řízený chováním spočívá v implementaci aplikace na základě popisu jejího chování.

Ve webové aplikaci MARAST je nástroj RSpec pro testování doplněn frameworkem **Capybara**, který jeho dokumentace [47] popisuje jako nástroj pro testování webové aplikace pomocí simulace interakcí reálného uživatele s aplikací, k čemuž Capybara umožňuje běh instance webového prohlížeče bez grafického uživatelského rozhraní. Díky tomu je Capybara využitelná pro automatizované testování aplikačního rozhraní.

Pro testování aplikačního rozhraní byla vyvinuta vlastní testovaná jednotka spec nástroje RSpec. Pro každý implementovaný koncový bod aplikačního rozhraní popsáný v sekci byly implementovány testy ověřeřující jeho

funkcionalitu. Příklad testu je demonstrován na kódu 7.1, kde lze vidět základní bloky spec jednotek. Blok **describe** specifikuje testovanou funkcionalitu, blok **before :each** zahrnuje kód vykonávaný před každým dílčím testem a blok **it** specifikuje konkrétní testované chování funkcionality. Obsahem kódu je testování chování aplikačního rozhraní pro odpověď na kvízovou otázku. Ve funkci answer lze vidět využití metody page.drive.browser.put nástroje Capybara implementující volání aplikačního rozhraní se zadanými parametry. Kód nepodstatný pro účely tohoto textu je v ukázce naznačen třemi tečkami.

7.2 Testování aplikace

V této sekci budou popsány metodiky a nástroje využité pro testování vyvíjené aplikace.

7.2.1 Testování vývojářem

Developer testing neboli testování vývojářem je označení průběžného ověřování vyvíjené funkcionality a jejího začlenění do aplikace. Toto testování klade důraz na stabilitu aplikace v každém dosažitelném stavu a testuje i oblasti aplikace nedostupné běžnému uživateli, především tedy synchronizaci s webovou verzí. Během vývoje byl pro testování využit testovací server, na kterém bylo pro dosažení různých stavů aplikace možno upravovat obsah databáze.

7.2.2 Testovaná zařízení

Součástí požadavků je korektní chování aplikace na různých podporovaných zařízení. K testování aplikace vývojářem byl v průběhu vývoje využíván simulátor systému iOS, kterým je možné simulovat chování všech současných iOS zařízení a verzí systému. Aplikace byla navíc testována na následujících fyzických zařízeních:

- Apple iPhone 6s
  - verze iOS 10.3.1
  - rozlišení displeje: 1334 × 750 pixelů
  - velikost displeje: 4.7”
- Apple iPhone 5s
  - verze iOS 10.3.1
  - rozlišení displeje: 1136 × 640 pixelů
  - velikost displeje: 4”
- Apple iPad 2
  - verze iOS 10.0.1

– rozlišení displeje: 1024 × 768 pixelů

– velikost displeje: 9.7”

7.2.3 Jednotkové testování

Součástí testování aplikace byl vývoj jednotkových testů, který se zaměřoval především na pokrytí funkčních požadavků na aplikaci. Pro testování byla využita technika mockování spočívající v nahrazování závislostí testovaného objektu objekty simulujícími funkcionalitu nahrazených závislostí. Příklad jednotkového testu je k vidění na kódu 7.2. Ukázkový test je součástí pokrytí požadavku F5 o synchronizaci obsahu s webovou verzí a upozorňování uživatele na nový dostupný obsah. Obsahem testu je zjištění přítomnosti nově přidané lekce ve stažených datech ze serveru v průběhu synchronizace. Kód je pro účely tohoto textu doplněn o české komentáře a jeho nepodstatné části jsou nahrazeny třemi tečkami.

7.2.4 Uživatelské testování

Vedle testování vývojářem byla aplikace v průběhu vývoje podrobena testování z uživatelského hlediska. Testování se účastnili zaměstnanci Katedry aplikované matematiky a byly během něj odhalovány funkční chyby a nedostatky v návrhu uživatelského rozhraní. Poznatky vyplývající z tohoto testování byly v průběhu vývoje zapracovávány do aplikace.

V současné době je aplikace ve stavu připravenosti pro testování s beta uživately, tedy vybranými studenty fakulty. Testování může proběhnout po nasazení vyvinutého aplikačního rozhraní na produkční server a byl pro něj navržen testovací scénář, který je obsahem přílohy B.

Zdrojový kód 7.1: Test aplikačního rozhraní pro odpověď na kvízovou otázku

```
describe "/quiz_progress/:id/answer" do
  before :each do
    ...
    @question.question_json["options"]
      .map { |o| o[1] }
      .each
      .with_index  { |answer, index|
        @correct_answers[index] = answer
        @wrong_answers[index] = answer == 1 ? 0 : 1
      }
  end

  def answer(answers)
    page.driver.browser.put(
      api_quiz_answer_path(id: @qp.id),
        questions: {@question.id => answers })
  end

  it "returns proper verdict for correct answer" do
    answer(@correct_answers)
    verdict = JSON.parse(page.body)
    expect(verdict["correct"]).to eq true
  end

  it "returns proper verdict for wrong answer" do
    answer(@wrong_answers)
    verdict = JSON.parse(page.body)
    expect(verdict["correct"]).to eq false
  end

  it "increments quiz score with correct answer" do
    score = @qp.score
    answer(@correct_answers)
    expect(QuizProgress.find(@qp.id).score).to eq score+1
  end

  it "does not increment quiz score with wrong answer" do
    score = @qp.score
    answer(@wrong_answers)
    expect(QuizProgress.find(@qp.id).score).to eq score
  end
end
```

Zdrojový kód 7.2: Test nalezení nově přidaných dat během synchronizace

```
func testNewData() {
    ...
    // Objekt podmínky ukončení asynchronního testu
    let expectation = XCTestExpectation()

    // Vyprázdnění mockovaného serverového uložiště dat
    session.storage.clear()

    // Synchronizace s očekáváním žádných nových dat
    fetcher.fetch() { result in
        // Ověření nedostupnosti žádných nových dat
        XCTAssertEqual(result, .noData)

        // Přidání nových dat
        session.storage.lectures.append(
            (id: 99, title: "new", abstract: "...",
             body: "...", level: 1, position: 99)
        )

        // Synchronizace s očekáváním nových dat
        fetcher.fetch() { result in
            // Ověření dostupnosti nových dat
            XCTAssertEqual(result, .newData)

           // Ukončení asynchronního testu
            expectation.fulfill()
        }
    }

    wait(for: [expectation], timeout: 10)
}
```

Závěr

Prvním cílem této práce bylo po analýze systému MARAST navhrnout a implementovat změny potřebné pro sdílení dat s mobilní aplikací. Na základě vypracované analýzy bylo navrženo a implementováno aplikační rozhraní, které bylo podrobeno potřebným testům.

Druhým cílem práce bylo na základě vytvořeného aplikačního rozhraní navrhnout mobilní aplikaci MARAST, která byla vyvinuta pro systém iOS a podrobena potřebným testům. V současné době je aplikace ve stavu první verze, která může být nasazena na distribuční kanály a zpřístupněna pro beta testování reálnými uživateli, tedy studenty fakulty.

Splnění zadání

V této sekci bude diskutováno splnění bodů zadání a odkázáno na části textu, které se dané problematice věnují. Číslování bodů vychází ze zadání citovaného v úvodu práce.

1. Výchozím bodem práce bylo seznámení s aplikací MARAST. Aplikace, její funkcionality a technologie, které využívá jsou shrnuty v sekci 1.3.

2. Navržené požadavky na mobilní aplikaci jsou součástí kapitoly 3.

3. V rámci práce byla provedena rešerše podobných mobilních aplikací, která je popsána v sekci 1.2. Následně byla aplikace navržena, přičemž návrh je obsahem kapitoly 4. Její součástí je zároveň sekce 4.2 věnovaná návrhu aplikačního rozhraní webové aplikace.

4. Aplikace byla úspěšně implementována. Realizaci aplikace je věnována kapitola 6, která shrnuje využité postupy a nástroje.

5. Mobilní aplikace a aplikační rozhraní byly podrobeny potřebným testům. Použité metodiky testování jsou rozebrány v kapitole 7. Dalším

krokem je uživatelské testování, pro jehož účely byl sestaven scénář, který je součástí přílohy B.

Možnosti rozšíření aplikace

Aplikace byla vyhotovena v rozsahu odpovídajícímu zadání práce. Pro její budoucí vývoj se pak nabízejí další možnosti rozšíření. Těmi by mohly být vedlejší funkcionality webové verze diskutované v sekci 1.1.5, tedy blog a diskuze. Především pro účely diskuze by mohla být mobilní aplikace zajímavou možností, jelikož by mohla uživatele upozorňovat na nové odpovědi.

Pro další verze aplikace by dále bylo vhodné zapracovat systém rozhodování o stažení nových dat či zobrazení dat lokálně uložených. Tím by se mohla výrazně snížit doba načítání a šetřit množství přenášených dat.

Literatura

[1] Kvetoň, K. Základy e-learningu. *Pedagogická fakulta OU*, 2007, [cit. 2017-2-24]. Dostupné z: `http://cit.osu.cz/dokumenty/elearning_kkveton.pdf`

[2] Kalvoda, T.; Klouda, K. O MARASTu. [cit. 2017-2-24]. Dostupné z: `https://marast.fit.cvut.cz/cs/about`

[3] The LaTeX Project. [cit. 2017-2-26]. Dostupné z: `https://www.latex-project.org/`

[4] Khan Academy. Khan Academy: you can learn anything. Dostupné z: `https://itunes.apple.com/us/app/khan-academy-you-can-learn-anything/id469863705?mt=8`

[5] Khanova škola. Khanova škola. Dostupné z: `https://khanovaskola.cz/`

[6] RubyOnRails.cz. Ruby on Rails. Dostupné z: `http://rubyonrails.cz/`

[7] RubyOnRails.cz. Rails Guides. Dostupné z: `http://http://guides.rubyonrails.cz//`

[8] Pecinovský, R. *Návrhové vzory*. Brno: Computer Press, 2007.

[9] Fielding, R. T. *Architectural Styles and the Design of Network-based Software Architectures*. Dissertation thesis, 2000.

[10] Jirůtka, J. OAuth 2.0. [cit. 2017-4-9]. Dostupné z: `https://rozvoj.fit.cvut.cz/Main/oauth2`

[11] Apple Inc. App Store. [cit. 2017-4-9]. Dostupné z: `https://developer.apple.com/support/app-store/`

[12] Smith, D. iOS Version Stats. [cit. 2017-4-9]. Dostupné z: `https://david-smith.org/iosversionstats/`

[13] Apple Inc. iOS 10. [cit. 2017-4-9]. Dostupné z: http://www.apple.com/ios/ios-10/

[14] Bray, T. The JavaScript Object Notation (JSON) Data Interchange Format. [cit. 2017-4-9]. Dostupné z: https://tools.ietf.org/html/rfc7159

[15] Ecma International. Standard ECMA-404. [cit. 2017-4-9]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf

[16] Rohlin, A. The Definition of Web Service EndPoint. [cit. 2017-4-17]. Dostupné z: https://www.techwalla.com/articles/the-definition-of-web-service-endpoint

[17] Mozilla Developer Network. HTTP cookies. [cit. 2017-4-17]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies

[18] Applicake. Doorkeeper. [cit. 2017-4-17]. Dostupné z: https://github.com/doorkeeper-gem/doorkeeper

[19] Apple Inc. Model-View-Controller. [cit. 2017-4-17]. Dostupné z: https://developer.apple.com/library/content/documentation/General/Conceptual/DevPedia-CocoaCore/MVC.html

[20] Peres, R. Model-View-Controller (MVC) in iOS. [cit. 2017-4-17]. Dostupné z: https://www.raywenderlich.com/132662/mvc-in-ios-a-modern-approach

[21] Furrow, A. Introduction to MVVM. [cit. 2017-4-17]. Dostupné z: https://www.objc.io/issues/13-architecture/mvvm/

[22] Buck, A. Using a Model-View-ViewModel architecture for iOS Ready Apps. [cit. 2017-4-17]. Dostupné z: https://developer.ibm.com/open/2015/12/16/using-a-model-view-viewmodel-architecture-for-ios-ready-apps/

[23] Fowler, M. *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional, 2003.

[24] Khanlou, S. Presenting Coordinators. [online], [cit. 2017-4-17]. Dostupné z: https://vimeo.com/144116310

[25] Scott, S. MVVM-C. [online], [cit. 2017-4-17]. Dostupné z: http://blog.ideveloper.co/blog/uikonf2016

[26] Apple Inc. Core Data. [cit. 2017-5-6]. Dostupné z: https://developer.apple.com/library/content/documentation/Cocoa/Conceptual/CoreData/

[27] Realm. Realm mobile database. [cit. 2017-5-6]. Dostupné z: https://realm.io/

[28] Drlík, R. Databázový systém pro správu biologických dat. 2010.

[29] Apple Inc. What's New with Multitasking. [online], [cit. 2017-4-27]. Dostupné z: https://developer.apple.com/videos/play/wwdc2013/204/

[30] Apple Inc. Notifications. [cit. 2017-4-27]. Dostupné z: https://developer.apple.com/ios/human-interface-guidelines/features/notifications/

[31] Apple Inc. iOS Human Interface Guidelines. [cit. 2017-5-2]. Dostupné z: https://developer.apple.com/ios/human-interface-guidelines

[32] Apple Inc. UINavigationController. [cit. 2017-5-2]. Dostupné z: https://developer.apple.com/reference/uikit/uinavigationcontroller

[33] Apple Inc. Tab bars. [cit. 2017-5-2]. Dostupné z: https://developer.apple.com/ios/human-interfaceguidelines-/ui-bars/tab-bars/

[34] Carthage. [cit. 2017-4-27]. Dostupné z: https://github.com/Carthage/Carthage

[35] Apple Inc. Building Modern Frameworks. [online], [cit. 2017-4-27]. Dostupné z: https://developer.apple.com/videos/play/wwdc2014/416/

[36] Consortium, T. M. MathJax. [cit. 2017-5-1]. Dostupné z: https://github.com/mathjax/MathJax

[37] SwiftyJSON. SwiftyJSON. [cit. 2017-4-27]. Dostupné z: https://github.com/SwiftyJSON/SwiftyJSON

[38] George, J. MBProgressHUD. [cit. 2017-5-1]. Dostupné z: https://github.com/jdg/MBProgressHUD

[39] Wood, D. XCGLogger. [cit. 2017-5-1]. Dostupné z: https://github.com/DaveWoodCom/XCGLogger

[40] Hato, Y. SlideMenuControllerSwift. [cit. 2017-5-1]. Dostupné z: https://github.com/dekatotoro/SlideMenuControllerSwift

[41] Tagaya, Y. Swinject. [cit. 2017-5-1]. Dostupné z: https://github.com/Swinject/Swinject

[42] phillips, R. Down. [cit. 2017-5-1]. Dostupné z: https://github.com/iwasrobbed/Down

[43] Luthi, C. XCDYoutubeKit. [cit. 2017-5-1]. Dostupné z: https://github.com/0xced/XCDYouTubeKit

[44] Rix, R. Result. [cit. 2017-5-1]. Dostupné z: https://github.com/antitypical/Result

[45] Mills, A. Reachability.swift. [cit. 2017-5-1]. Dostupné z: https://github.com/ashleymills/Reachability.swift

[46] North, D. Introducing BDD. [cit. 2017-5-6]. Dostupné z: https://dannorth.net/introducing-bdd/

[47] Nicklas, J. Capybara. [cit. 2017-5-6]. Dostupné z: http://www.rubydoc.info/github/teamcapybara/capybara

[48] Consortium, T. M. MathJax. [cit. 2017-3-20]. Dostupné z: https://www.mathjax.org

Příloha **A**

Seznam použitých zkratek

MARAST Matematika Radostně

MVC Model-View-Controller

MVVVM Model-View-ViewModel

REST Representational state transfer

SDK Software development kit

XML Extensible markup language

URI Unified resource identifier

Příloha B

Scénář uživatelského testování

Scénář uživatelského testování začíná na úvodní obrazovce aplikace ve stavu před přihlášením uživatele. Předpokladem testování je zpřístupnění takového obsahu, který umožňuje plné projití scénáře, tedy například lekce obsahující video.

Testovací scénář je tvořen následujícími kroky:

1. Proveďte přihlášení pomocí vašeho ČVUT účtu.
2. Přejděte na obrazovku výběru dostupných lekcí.
3. Zobrazte libovolnou lekci.
4. Spusťe video, které je obsahem lekce.
5. Stopněte přehrávané video.
6. Zobrazte video v režimu přes celou obrazovku.
7. Ukončete přehrávání videa v režimu přes celou obrazovku.
8. Ukončete zobrazení přehrávaného videa.
9. Přejděte na obrazovku výběru kvízů.
10. Spusťte libovolný aktivní kvíz.
11. Odpovězte na kvízovou otázku bez ohledu na správnost odpovědi.
12. Přejděte na obrazovku cvičebnice.
13. Zobrazte řešení příkladu.
14. Přejděte na další příklad.
15. Označte příklad jako vyřešený.

16. Přidejte k příkladu textovou poznámku.
17. Filtrujte zobrazené příklady výběrem pouze příkladů s vyplněným řešením.
18. Změňte zobrazovaný kurz.
19. Přejděte do nastavení aplikace.
20. Proveďte odhlášení.

Příloha C

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `app` ..... zdrojové kódy implementace
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF