VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁSTROJ NA PROCVIČOVÁNÍ VĚDOMOSTÍ K MATURITĚ

A TOOL FOR PRACTISING KNOWLEDGE FOR GRADUATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **SAMUEL MOŘKOVSKÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

147011

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Mořkovský Samuel**
Program: Informační technologie
Specializace: Informační technologie

Název: **Nástroj na procvičování vědomostí k maturitě**
Kategorie: Uživatelská rozhraní
Akademický rok: 2022/23

Zadání:

1. Prostudujte tématiku UX a moderní nástroje pro tvorbu webových a mobilních aplikací s GUI.
2. Proveďte průzkum a zmapujte potřeby uživatelů (žáci, ale i tvůrci obsahu).
3. Navrhněte UI procesy a GUI, vytvořte makety, otestujte a vyhodnoťte.
4. Navrhněte klíčové datové struktury a implementujte základ informačního systému pro správu obsahu k procvičování znalostí k maturitě.
5. Vytvořte experimentální aplikace s GUI pro výuku i správu obsahu. Řešení otestujte a vyhodnoťte z pohledu UX.
6. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- Steve Krug: Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability, ISBN: 978-0321965516
- Steve Krug: Rocket Surgery Made Easy: The Do-It-Yourself Guide to Finding and Fixing Usability, ISBN: 978-0321657299
- Joel Marsh: UX for Beginners: A Crash Course in 100 Short Lessons, O'Reilly 2016

Při obhajobě semestrální části projektu je požadováno:
Body 1., 2., 3. a z větší části bod 4.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 31.10.2022

Abstrakt

Tato práce se zabývá vytvořením edukační mobilní aplikace pro maturitní předmět uměleckých středních škol Dějiny výtvarné kultury. Při práci byl proveden průzkum mezi studenty umělecké SŠ pro získání informací o jejich postoji ke zmíněnému předmětu a zkušenostech s edukačními aplikacemi. Po průzkumu byla provedena realizace mobilní aplikace pro systémy iOS a Android. Serverová aplikace pak realizuje veškerou logiku a za pomocí HTTP komunikuje s mobilní aplikací na zařízeních uživatelů. Data jsou uchovávána v relační databázi MySQL. Pro správu obsahu je poskytnuto webové rozhraní s přihlášením. Po realizaci byl otestován UX se studenty pro získání zpětné vazby. Výsledkem byly pozitivní reakce a pokud by aplikace obsahovala větší množství obsahu, studenti by ji aktivně používali. Výsledky práce ukazují, že ji studenti považují za přínosnou a další vývoj má význam.

Abstract

The goal of this thesis is to create an educational mobile application for a subject History of arts and culture. The main target group are high schools. Firstly, initial research with students was done about their experience with mobile educational appliactions and opinion about mentioned subject. After the research a mobile appliaction was implemented for Android and iOS systems. Then, a server application processes the logic and communicates through HTTP with mobile applications on users devices. Data are stored in a relational databaze MySQL. The content is managed through a web interface with login. After the implementation, an UX test was performed with students to obtain feedback. The result was a positive reaction and if there was more content, students would actively use the application. The results of this thesis show that students find the application useful and is worthy for further development.

Klíčová slova

UX, edukační aplikace, mobilní aplikace, dějiny výtvarné kultury, dějiny umění, Android, iOS, React Native, Expo, klient-server, PHP, Laravel

Keywords

UX, educational application, mobile application, history of arts, history of culture, Android, iOS, React Native, Expo, client-server, PHP, Laravel

Citace

MOŘKOVSKÝ, Samuel. *Nástroj na procvičování vědomostí k maturitě*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Nástroj na procvičování vědomostí k maturitě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Samuel Mořkovský
10. května 2023

Poděkování

Děkuji panu Ing. Vítězslavu Beranovi Ph.D. za vedení mé práce, vstřícnou komunikaci a cenné rady.

Obsah

Kapitola 1

Úvod

Cílem této práce je navrhnout, implementovat a otestovat aplikaci, která se zaměřuje na studenty středních škol. Měla by jim umožnit si příjemnou a méně seriózní formou procvičit, osvěžit a případně doplnit znalosti z určitých maturitních předmětů. Výběr těchto předmětů bude záležet na nalezení pedagogů středních škol ke spolupráci. Výsledek práce nemá nahrazovat vzdělání potřebné k úspěšnému zvládnutí maturitní zkoušky, má studentům pomoci si informace nabyté při studiu příjemnou a pohodlnou formou připomínat.

Uživatel by měl mít možnost otevřít aplikaci, kdykoli si chce procvičit nějakou konkrétní maturitní otázku nebo vybraná témata z celého předmětu. Dále by měl být uživatel aplikací motivován, kromě vlastní iniciativy si zopakovat učivo, k opětovnému využívání a tím i častějšímu opakování znalostí.

K přiblížení se efektu motivovaného používání aplikace bude potřeba nastudovat principy UX (z angl. User Experience) a co vhodně je zakomponovat. V kombinaci s vhodně navrženým uživatelským rozhraním by tak měla aplikace být již při prvním použití jednoduchá, přehledná a intuitivní, aby studentům nic nebránilo začít v jejím používání. Lze také využít prvky gamifikace, které studentům pomohou si proces učení odlehčit odměňováním, překvapením a třeba i sdílením svých výsledků. Velkou roli bude hrát také dostupnost aplikace – měla by být dostupná co největšímu počtu studentů středních škol, aby ji mohli bezpodmínečně využívat. S ohledem na trendy a standardy dnešní doby lze předpokládat, že se bude jednat o aplikaci mobilní.

Pro určení konkrétního rozsahu aplikace bude třeba provést uživatelský průzkum a zjistit, co od edukační aplikace studenti očekávají, jakou formou by si chtěli informace opakovat a jaké typy informací jim ve vybraných předmětech dělají největší obtíže. Zároveň bude vhodné získat ke spolupráci pedagoga, který může poskytnout obsah, dát zpětnou vazbu k formě prezentování informací a případně posoudit, zda užívání aplikace studenty má dopad na výsledky v jeho předmětu.

Text sleduje postupně vývoj celé aplikace. Začíná uživatelským průzkumem, kde je kromě shrnutí potřebných technik práce s uživatelem prezentován i výsledek průzkumu a specifikace výsledné aplikace a tvorba jeho grafického uživatelského rozhraní
(kapitoly 2 a 3).

V kapitolách 4 a 5 jsou pak představeny nástroje a techniky navržené pro realizaci aplikace. Je zde vysvětlen i datový model a některé důležitější implementační detaily, které se ve výsledné realizaci nacházejí.

V neposlední řadě kapitola 6 popisuje testování s uživateli a jejich zpětnou vazbu. V rámci této kapitoly jsou i navržená či provedená řešení a vysvětleny přetrvávající problémy.

Kapitola 2

Počáteční průzkum

Na počátku práce je třeba zjistit a specifikovat, co je vlastně žádoucí výstupní řešení. V této kapitole budou popsány cíle této výukové aplikace a jakým způsobem proběhl výběr předmětu a počáteční uživatelský průzkum, který určuje faktory jako typ aplikace, forma cvičení atd. Dále budou popsána podobná nebo související aktuální řešení na trhu, co jsou jejich hlavní přednosti a čím konkrétně jsou tyto informace podstatné pro tuto práci. Závěrem této kapitoly bude specifikace požadavků pro každou část řešení.

2.1 Cíle výukové aplikace

Cílem této aplikace je, že si ji student pustí sám od sebe, bude v jeho prostředí chtít chvíli zůstat a procvičovat si obsah, který nabízí. Student by neměl cítit, že si pouští aplikaci na vyzvání učitele, proto se řešení zaměří primárně na používání ze strany studentů. Dále by procvičování mělo být rychlé, jednoduché a pokud možno zábavné pro udržení uživatelovy pozornosti.

Před začátkem uživatelského průzkumu je třeba pochopit jaké předměty mohou být na maturitní zkoušce v ČR.

Maturitní předměty v ČR

Aktuálně se maturitní zkouška v ČR dělí na 2 části – společná a profilová. Společná část je povinná pro všechny maturanty a povinné zkoušky jsou z předmětu matematika nebo cizí jazyk a český jazyk a literatura. Z cizích jazyků je na výběr z anglického, německého, ruského, francouzského a španělského. V této části se lze přihlásit k dalším 2 nepovinným zkouškám. Všechny zkoušky v této části se konají formou didaktického testu.

V profilové části mají všichni studenti povinnou zkoušku z českého jazyka a literatury a pokud si maturant zvolil ve společné části cizí jazyk, je pro něj povinná profilová zkouška z tohoto jazyka. Maturant se dále přihlašuje ke 2 nebo 3 povinným profilovým zkouškám ze školní nabídky povinných zkoušek. [13]

Pro přípravu na maturitu mají zpravidla studenti dostupné školní materiály, vlastní poznámky a také mohou hledat informace na internetu. Přestože studování těchto materiálů je běžný a dostatečně efektivní způsob přípravy na maturitní zkoušku, lze proces učení různými prostředky a přístupy zpříjemnit a ozvláštnit. Mezi některé způsoby patří použití audiovizuálních pomůcek (videa, filmy, virtuální třídy), jazykových laboratoří nebo počíta-

čem asistovaného učení (podcasty, blogy, DVD, e-knihy, webové aplikace...). Tato práce se zaměří na vytvoření nástroje pro počítačem asistované učení[1].

2.2 Uživatelský průzkum

Pro zjištění uživatelských potřeb a konkretizování požadavků na aplikaci je nutné provést uživatelský průzkum. V průběhu průzkumu bude třeba zjistit, jak se uživatelé nejraději učí, jaký typ opakování preferují a kolik času by cvičení mělo zabrat. Je také namístě prozkoumat jaké mají zkušenosti s existujícími aplikacemi, co se jim na nich líbilo a jakými prvky se lze inspirovat v tomto řešení. Uživatelský průzkum se provádí pro lepší porozumění uživatelů, kteří by potenciálně mohli využívat daný produkt. Je možné jej provést téměř kdykoli, protože sesbíraná data a informace z nich získaná nám zůstanou. Nicméně je vhodné prozkoumávat zájmy uživatelů co nejdříve, aby se vyvíjený produkt mohl co nejvíce (a nejlevněji) přizpůsobit těmto potřebám.

Otázky na uživatele mohou být subjektivní (oblíbená barva) a objektivní (odkud přišli na naši stránku). Podobný subjektivní názor mezi velkým počtem lidí se může tvářit objektivně, tzn. je třeba dbát na to, že některý subjektivní názor může být populární, i když jej to nutně nečiní pravdivým. Počet uživatelů na průzkum záleží od informace, kterou se snažíme získat. Pokud se průzkum týká např. problémů na webu, na některé problémy narazí 1 ze 20 uživatelů, průzkum na 10 uživatelích je nemusí vůbec objevit a je třeba s tímto počítat. Mezi nejběžnější metody uživatelského průzkumu patří[8]:

- Pozorování - zadání požadavku a sledování chování uživatele
- Interview - tázání se na otázky jeden na jednoho
- Focus group - několik lidí pohromadě odpovídá na otázky a diskutuje
- Dotazníky - na papíru nebo online, prvek anonymity, nelze navázat na odpovědi

Pozorování

V průběhu pozorování je vhodné si poznatky ihned zaznamenávat – lze do poznámek nebo videem. Dále je také důležité vzít v potaz řeč těla a výrazy obličeje, které odhalují uživatelovy pocity při testování. Pokud uživatel provede nějaké rozhodnutí, je v rámci UX klíčové znát kromě výsledku, i proč uživatel zvolil tuto možnost. Uživatel může být při testování zmatený či se cítit nepříjemně, protože něčemu nerozumí. V tomto případě by se nemělo uživateli pomáhat, značně se tím sníží přínosná hodnota testování – i neúspěšný test je pro UX užitečná zpětná vazba. UX designer by měl brát v potaz hlavně, když je s uživatelem při pozorování v místnosti, že se uživatel nechce „ztrapnit“, a aby k tomu nedošlo, může podávat zavádějící informace. Případně se může chtít zavděčit, a tím částečně zamlčet, co si opravdu myslí[8].

Dotazníky

Dotazník je sada otázek, které uživatel vyplňuje na papíru či online, často anonymně. Hlavní výhody dotazníků spočívají v tom, že jsou soukromé, a tím je uživatel upřímněji vyplňuje,

[1] Definici počítačem asistovaného učení (z angl. Computer assisted learning) lze nalézt v abstraktu práce na https://pubmed.ncbi.nlm.nih.gov/8714892/

všichni uživatelé dostanou naprosto stejně podané otázky a jsou levné i při nárustu účastníků.

Dotazník má zároveň své nevýhody – není možné se dále doptat na odpovědi a výsledky lze nechtěně ovlivnit způsobem napsání otázky. Čím delší dotazník, tím méně lidí jej dokončí. Dotazníky je vhodné používat, pokud je potřeba se zeptat mnoha lidí, kontrolovat, kdo konkrétně bude dotazník vyplňovat, nebo porovnávat odpovědi uživatelů[8].

2.3 Průzkum

Hlavní cílovou skupinou aplikace jsou studenti uměleckých oborů na středních školách. Dalšími potenciálními uživateli mohou být studenti vysokých škol, kteří si chtějí otestovat své vědomosti nebo jim je jednoduše aplikace tematicky blízká. Kromě studentů mohou aplikaci využít i lidé se zájmem o dějiny či umění.

Aby bylo možné lépe cílit na studenty středních škol a jejich potřeby, byl proveden průzkum pomocí dotazníku, který měl zodpovědět základní otázky potřebné pro návrh aplikace:

1. V jaké formě by aplikace měla být dostupná, aby byla studentům nejbližší?
2. Jak by mělo typově vypadat cvičení a kolik času by mělo zabrat?
3. Odkud bude brán obsah a jaký by měl být jeho rozsah?
4. Které typy informací (kalendářní data, životy autorů, názvy uměleckých děl,...) se studentům učí hůře než jiné?

Při tvorbě edukačního nástroje je vhodné si ke spolupráci zvolit nějakou vzdělávací instituci, v lepším případě takovou, která vyučuje na maturitní úrovni předmět, jenž bude náplní aplikace. Na dotazník odpovědělo celkem 61 studentů z různých ročníků stejné SŠ.

Výběr předmětu a Dějiny výtvarné kultury

Ke spolupráci byla vybrána Střední škola filmová, multimediální a počítačových technologií s.r.o. ve Zlíně, která byla ochotna spolupracovat a má s autorem práce vstřícné vztahy. Při komunikaci se školou bylo vyhodnoceno, že pro předměty společné části maturitní zkoušky existuje značné množství volně dostupných materiálů i aplikací. Studenti by zároveň raději upřednostnili nástroj na učení některého předmětu z profilové části.

V rámci oboru **82-41-M/17 Multimediální tvorba** nabízí škola 3 zaměření – každé z nich má v profilové části svůj specifický předmět. Jako umělecká škola však do profilové části zařazuje předmět DVK (Dějiny výtvarné kultury), který je pro všechna zaměření společný. Učitel toho předmětu byl zároveň nakloněn spolupráci a sám hledá způsob, jak předmět studentům lépe přiblížit.

Předmět se, také pod názvy *Dějiny umění* nebo *Dějiny kultury*, vyučuje ve všech uměleckých oborech pro střední školy v ČR[2], ať už se jedná o obor zakončený výučním listem či maturitní zkouškou. Obsah předmětu je typicky rozdělen do období, která následně více či méně odpovídají maturitním otázkám (jednotlivé školy si rozsah maturitních otázek upravují dle svých potřeb).

[2]Dle portálu https://atlasskolstvi.cz se v ČR nachází 125 středních uměleckých škol.

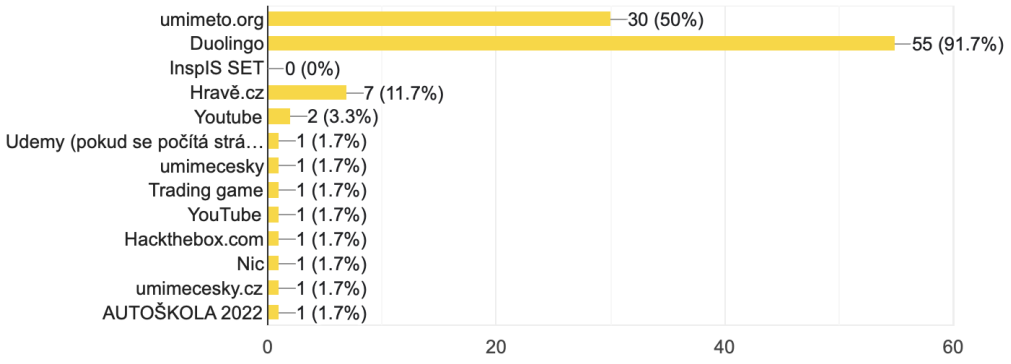
Jedná se o předmět typově podobný dějepisu s rozdílem v tom, že je zaměřen na vývoj výtvarného umění – malba, architektura, sochařství, užité umění, související autoři a události atd. V RVP SOV[3] je tento předmět zahrnut v rámci obsahového okruhu Uměleckohistorická a výtvarná příprava, který je částečně popsán tak, že: „*Objasňuje žákům základní souvislosti dějin umění a vývojových etap společnosti, seznamuje je s tvorbou významných osobností konkrétních historických období a jednotlivými složkami scénické tvorby*“[10].

Zkušenosti s edukačními aplikacemi

První část dotazníku byla zaměřena na zkušenosti studentů s existujícími edukačními aplikacemi. Šlo hlavně o zjištění, v jaké formě je používali, proč je používali a proč je případně používat přestali. Z dotazníku vyplynulo, že 70 % studentů používali webovou aplikaci, téměř nikdo se neučil za pomocí desktopové aplikace, a edukační aplikace na tabletu spustili jen 4 studenti. Dle očekávání byly v silné převaze mobilní zařízení, na kterých studenti využili nějakou edukační aplikaci v 93 %.

Vzhledem k tomu, že studenti jsou nejvíce zvyklí na edukační aplikace na mobilních zařízeních a zároveň 99 % lidí ve věku 16 - 24 let vlastní mobilní telefon[9], bude mít největší dosah aplikace vytvořená právě pro mobilní zařízení a do budoucna lze případně zvažovat webové rozhraní.

Z obrázku 2.1 je zřejmé, jaké aplikace jsou nejznámější. Překvapivé je, že *InspIS SET* od České školní inspekce nabízí nejvíce obsahu přizpůsobeného maturitní zkoušce v ČR (pro několik předmětů) a dle výsledků jej nikdo z dotázaných nezná.



Obrázek 2.1: Odpovědi ve formě grafu na dotaz, jaké znají studenti edukační aplikace.

Duolingo, které bude více popsáno v sekci 2.4, vyučuje jazyky a je nejznámější edukační aplikací – tento fakt se potvrdil i ve výsledcích průzkumu. Studenti mezi největší výhody zařadili flexibilitu používání aplikace (kdykoli a kdekoli), rychlá cvičení, jednoduché používání a gamifikační prvky, které je motivovaly aplikaci využívat každý den. K aplikaci se vraceli po delším nepoužívání ve chvíli, kdy si chtěli procvičit jazyk, který již umí nebo se naučit nový. Naopak se jim nelíbilo řešení obtížnosti související s postupem ve zvoleném jazyku – uživatel si nemůže sám obtížnost korigovat, a tím se aplikace stává pro některé jednoduchá, nudná či repetitivní, a pro jiné těžká a demotivující. Dále je pro ně, jakožto

[3]Rámcové vzdělávací programy středního odborného vzdělávání. Jednotlivé RVP lze nalézt na https://www.edu.cz/rvp-ramcove-vzdelavaci-programy/

studenty SŠ často bez vlastních příjmů, překážkou omezený počet chyb, kterých se mohou dopustit předtím, než se zregenerují životy.

Druhou nejznámější aplikací mezi dotazovanými je webový portál umimeto.org[4], který zná 53 % z dotázaných a bude také více rozepsán v kapitole 2.3. Studentům se líbila jednoduchost a svižnost cvičení, zpětná vazba na chyby, gamifikační prvky a možnost zábavnější formy plněních domácích úkolů. I když aplikaci může využívat kdokoli a je do určitého počtu odpovědí zdarma, využívali ji převážně na výzvu učitele a poté přestali, chyběla jim motivace.

Jen 11 % studentů zná výukovou webovou aplikaci Hravě[5], která nabízí studentům maturitní obsah z českého a anglického jazyka a matematiky – demo zadarmo, zbytek kurzu je možné si dokoupit. Studenti se k Hravě v dotazníku ale více nevyjadřovali.

Dle zmíněných aplikací a zkušeností mají studenti rádi aplikace se svižným a jednoduše ovladatelným procvičováním, které si mohou kdykoli zapnout či odložit. Zároveň se jim líbí motivační prvky (u Duolinga například hlášky a počítadlo dnů s provedeným cvičením), které je přimějí pokoušet se v učení vytrvat a často reflektují jejich postup. Proto bude výsledkem této práce mobilní aplikace, čímž nabídne flexibilitu použití a nejpřívětivější formu pro cílovou skupinu studentů SŠ.

Způsob a rozsah učení DVK

Další otázka, kterou je třeba zodpovědět je, jak by mělo typově vypadat cvičení a kolik času by mělo zabrat. Z průzkumu vyplývá, že studenti používali aplikace spíše s kratšími cvičeními (v rámci jednotek minut) než s delšími testy (o takových aplikacích ve výsledcích dotazníku nebyla žádná zmínka). Přesto je však vhodné se ujistit, jestli opravdu preferují kratší cvičení, která si mohou spustit vícekrát po sobě, nebo delší testy například jednou za pár dní. Ve výsledcích 80 % studentů bylo pro kratší a častější cvičení, zbytek pro dlouhá, případně pro kombinaci – průběžně krátká cvičení a na konci tématu shrnující test.

Dle dalších otázek v dotazníku mají studenti největší problém se zasazením informací do širšího kontextu. Konkrétně se může jednat například o zařazení specifických uměleckých prvků do správného období, případně celého období do historického kontextu (co bylo předtím, potom, jaké historicky důležité události se v období udály, atd.). Na toto bude vhodné myslet při návrhu jednotlivých cvičení.

2.4 Průzkum podobných řešení

Na trhu se aktuálně nachází nespočet různých edukačních aplikací, které nabízejí vyučování či procvičování různých vědomostí. Některé z nich byly zmíněny samotnými studenty v průzkumu, a je proto vhodné se na ně více zaměřit a rozebrat hlavní výhody, kterými se lze inspirovat v rámci tohoto řešení.

Duolingo

Duolingo je aktuálně nejpopulárnější existující mobilní aplikace v edukační kategorii na Google Play i App Store. Obsahuje více než 40 jazyků, které se uživatelé mohou učit. Měsíčně aplikaci navštíví přibližně 45 milionů uživatelů a dohromady má přes 500 milionů stažení[4].

[4] https://umimeto.org
[5] https://hrave.cz

Silnými přednostmi aplikace jsou[2]:

- Učení děláním – uživatel od prvního spuštění aplikace interaguje s bohatým obsahem. Lekce jsou navrženy s úmyslem přimět uživatelovu pozornost na to, co se potřebuje naučit bez nutnosti si číst explicitní úvodní přehledy.
- Personalizovaná cvičení – Duolingo používá vlastní reakce a chování uživatele, aby mu pomohla se zlepšit. Jedná se například o poskytování více cvičení, kde to uživatel potřebuje, nastavování obtížnosti a pořadí cvičení nebo zobrazování feedbacku aj. K tomuto Duolingo využívá své vlastní modely strojového učení, které interně označuje jako Birdbrain.
- Motivace – pomocí gamifikace zajišťuje, že se uživatelé k aplikaci vracení a tráví v ní čas učením se materiálů. Některé gamifikační prvky jsou odemykání dalších materiálů, získání XP[6] nebo herní měny. Poskytování častých odměn udržuje mozek v zapojení. Dalším nástrojem jsou odměny za každodenní použití – uživatelé dostávají odměny za to, že se každý den k aplikaci vrátí a vyplní cvičení. Krom toho se v aplikaci nachází sociální aspekt – žebříček uživatelů, u kterých lze vidět jak moc se aktuálně učí.
- Práce s prostředím – aplikace se snaží vytvořit příjemné prostředí pro uživatele humorem, vyprávěním příběhů, emocionálními momenty, postavičkami (např. maskotem Duo), animacemi atd. Příjemné prostředí pomáhá uživateli se cítit otevřeněji a komfortněji, což je viditelné na obrázku 2.2

.

Duolingo je velmi propracovaný edukační systém, který má za sebou nespočet výzkumů, experimentů a měření. Kromě toho je na trhu již od roku 2011 a je stále vylepšován a rozšiřován. Pro tuto práci se lze inspirovat prací s prostředím (např. maskotem) a gamifikačními prvky (každodenní úkoly, odměňování). Inspirací může být také zapojení uživatele do procesu učení se co nejdříve po spuštění aplikace.

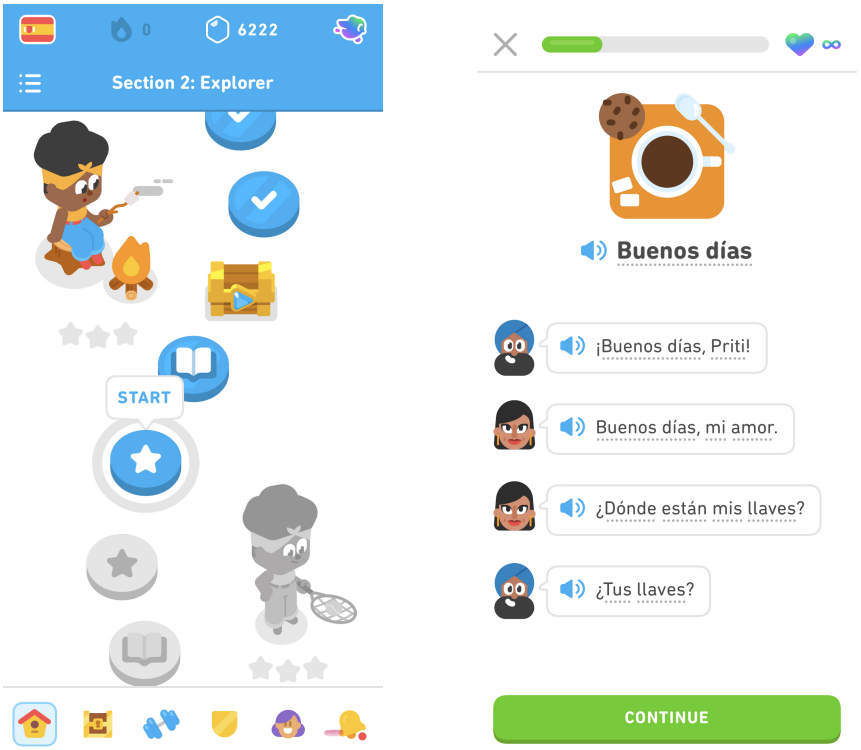
Umíme.to

Umíme.to je česká webová aplikace, která nabízí možnost procvičování několika modulů dle různých témat (česká jazyk, matematika, dějepis...). Obsah v modulech se rozsáhlostí různí; v českém jazyce se jedná o první stupeň ZŠ až po 4. ročník SŠ. Cvičení jsou vytvořena gamifikovanou formou, jedná se například o rozřazovací (obr. 2.3) a doplňovací cvičení, minihry, pozorumění textu a úlohám atd. Obsahuje speciální minihry pro více hráčů, takže spolu mohou uživatelé interagovat a společně plnit úkoly. Uživatelé se dále mohou registrovat a přihlašovat do tříd, kde jim může učitel zadávat úkoly k řešení a sledovat jejich postup a výsledky, případně statistiky v rámci celé třídy.

Aplikace poskytuje pouze webové rozhraní, které je však přehledné a příjemně zpracované. Omezení plyne z povahy webové aplikace, že se uživateli lépe používá na zařízeních s velkým displejem (počítače, laptopy...) a uživateli se tedy omezují možnosti kdy si aplikace spustí. Toto částečně potvrzuje výsledek průzkumu mezi studenty 2.3, kde studenti tvrdili, že si tuto aplikaci spustí převážně na vyzvání učitele a nejsou příliš motivováni si ji spustit sami, přestože obsahuje řadu motivačních prvků. Výukový systém podložen výzkumem, který je vedený v rámci Fakulty informatiky MU v Brně.[7]

[6] XP – zkratka pro anglický výraz *experience points*, přeložen jako zkušenostní body. Jedná se o herní prvek, který nejčastěji značí hráčův pokrok.

[7] https://www.umimeto.org/text-based-on-research



Obrázek 2.2: Ukázka příjemně vytvořeného prostředí Duolingem. Vlevo je obrazovka s výběrem lekcí, postavičky mají animaci, uživatel chce pokračovat dále. Vpravo je cvičení na pochopení textu, opět doplněné postavičkami, ikonkami a obrázky.

Inspirací pro tuto práci může být návrh složitějších cvičení a jejich vyhodnocování a případně modul, který slouží k rozlišení uživatelů na žáka a učitele.

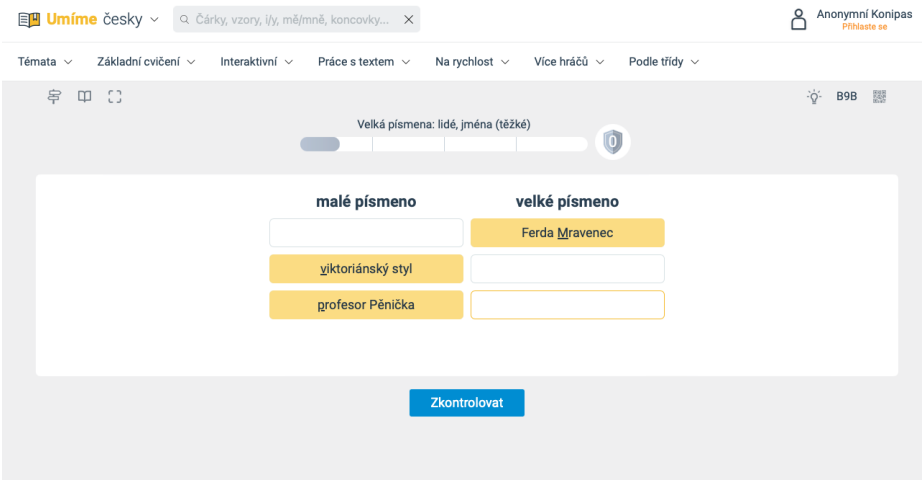
Google Arts & Culture

Jedná se o webový portál s kolekcemi umění, kultury a historie. V rozsáhlé škále obsahu lze vyhledávat dle umělců, historických událostí, geografické polohy nebo také tématu či barev. Cílem aplikace je zpřístupnit a přiblížit umění a kulturu široké veřejnosti, přestože si díla nemohou prohlédnou fyzicky. Uživatel může prozkoumávat, vyhledávat a obsah je podáván více formou velmi přehledné databáze než aplikací, jejímž cílem je přimět uživatele k zamyšlení a odpovídání na otázky. Obsahuje i minihry, virtuální prohlídky, obrázky ve vysokém rozlišení s možností přiblížení a prohlédnutí si detailů.

Tento portál je typově odlišný od cíle této práce, ale je vhodné jej zmínit, protože ve vztahu k dějinám umění je to jeden z největších dostupných edukačních nástrojů pro veřejnost.

2.5 Specifikace uživatelských potřeb

V závěru kapitoly o počátečním průzkumu jsou dle výsledků odvozeny požadavky na řešení a jeho jednotlivé části.



Obrázek 2.3: Příklad cvičení typu *rozřazovačka* v aplikaci Umíme.to. Možnosti lze myší přesouvat do volných políček a tím sestavit správnou odpověď.

Rozdělení řešení

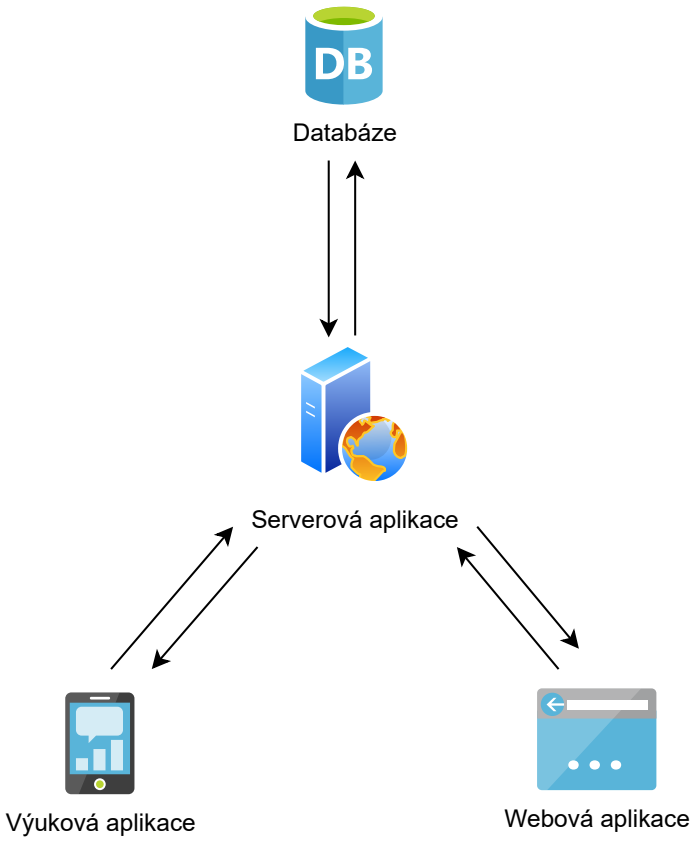
Z povahy aplikace vyplývá, že uživatele lze rozdělit na 2 skupiny – uživatele mobilní aplikace a uživatele plnící obsah. Celkově bude tedy řešení rozděleno na 3 části, které budou manipulovat s obsahem a dalšími daty uchovanými v databázi. Komunikace mezi všemi částmi řešení je naznačena na obrázku 2.4. Jednotlivé části jsou:

1. Výuková aplikace pro mobilní zařízení, kterou budou používat běžní uživatelé pro získávání a opakování vědomostí z předmětu Dějiny výtvarné kultury.
2. Webová aplikace pro správu obsahu, ve které budou vytvářeny a editovány tematické okruhy a otázky se zadáním, odpověďmi a vysvětlujícími informacemi.
3. Serverová aplikace, která bude obsluhovat požadavky přicházející z výukové a webové aplikace.

Požadavky na výukovou aplikaci

Nejviditelnější část řešení je klientská aplikace, kterou mají uživatelé využívat k učení se a opakování si obsahu, který se může objevit u maturitní zkoušky. Je potřeba ji zpracovat tak, aby byla co nejjednodušší na použití, protože cílová skupina uživatelů jsou sice mladí lidé zkušení s používáním mobilních aplikací, nicméně u nich nelze předpokládat pokročilou technickou zdatnost. Používání edukační aplikace již samo o sobě vyžaduje zvýšenou pozornost a je tedy vhodné na jiných místech uživateli zjednodušit interakci, například intuitivním ovládáním, vysvětlivkami a chybovými hláškami v nutných případech.

Primárními funkcemi aplikace jsou: možnost zobrazení otázky, vybrání některé z odpovědí, následné získání zpětné vazby pro uživatele a případné zobrazení odlišnosti jeho odpovědi od správné. Doplňující informace k otázce si uživatel bude moci prohlédnout hned po zodpovězení a budou zahrnovat text i obrázky. Obrázky jsou pro téma umění a historie klíčový učící prvek. Sekundární funkce jsou pak všechny, které nemají přímou souvislost s učením, ale přidávají aplikaci specifické prvky, které uživateli ozvláštňují a zpříjemňují



Obrázek 2.4: Schéma reprezentující rozdělení aplikace. Serverová aplikace komunikuje se všemi částmi řešení a realizuje veškerou logiku a datové úložiště.

zážitek. Základem toho je uživatelský účet, ke kterému se budou ukládat výsledky zodpovězených otázek a z těchto záznamů lze agregovat informace o tom, jak se uživateli daří v jednotlivých tématech a jaké jsou například otázky (či témata), ve kterých nejvíce chybuje. Sesbírané statistiky lze uživateli následně prezentovat formou umístění v žebříčku mezi dalšími uživateli, získávání různých typů trofejí nebo výpisu na obrazovce s uživatelským profilem.

Je žádoucí, aby aplikace byla interaktivní (prvky obratem reagují na gesta uživatele), informovala uživatele o probíhající činnosti v pozadí (pokud je třeba, aby o ní uživatel věděl – například musí vyčkávat), byla responzivní a uživatelé s různými rozlišeními telefonů měli co nejpodobnější zážitek. Dalším požadavkem je funkčnost aplikace na operačních systémech Android i iOS a na obou poskytnutí stejné nebo maximálně podobné funkce. Mezi další požadavky, které se k této aplikaci vztahují stejně jako k drtivé většině mobilních aplikací na trhu, patří šetrné zacházení s mobilními daty (načítání jen relevantních dat a vyhnutí se opakovaného dotazování na stejná data, načítání obrázků v komprimované formě atd.) a zajištění dostatečného výkonu aplikace pro pozitivní uživatelský zážitek.

Požadavky na aplikaci pro plnění obsahu

Zde jsou požadavky podstatně jednodušší než u mobilní aplikace. Zatímco mobilní aplikace je část, která bude dostupná pro nejvíce uživatelů, rozhraní pro plnění obsahu budou využívat jen tvůrci obsahu. Rozhraní bude dostupné z webového prohlížeče jako webová

aplikace a budou se do ní uživatelé moci přihlásit pomocí přihlašovacích údajů. Přestože se dá předpokládat, že tvorba obsahu bude probíhat primárně ze zařízení s klávesnicí pro snadnější manipulaci s textem, bude rozhraní optimalizováno i pro telefony, aby bylo možné provést nezbytné úpravy i z mobilního zařízení.

Rozhraní bude nabízet přehled témat se základními informacemi (zda je téma zveřejněné a množství vytvořených otázek k tématu) společně s formulářem pro vytvoření nového tématu. Dále bude v rozhraní možnost prohlížet, modifikovat a vytvářet otázky. K otázce náleží zadání, odpovědi a doplňující informace. Budou se zobrazovat pouze ta pole, která jsou relevantní k danému typu otázky, aby uživatel viděl jen vstupy, které je potřeba vyplnit. Délka a obsah doplňujících informací k různým otázkám mohou být různě dlouhé a různého typu (někdy jen text, jindy mnoho obrázků), proto budou tyto informace skládány v blocích za sebe, které budou buď textové či obrázkové. U otázek bude vhodné zobrazit, kdo ji naposledy upravil a případně i historii úprav pro případnou revizi.

Požadavky na serverovou aplikaci

Serverová aplikace bude zpracovávat API požadavky přicházející z jednotlivých instancí mobilní aplikace. Jelikož mobilní aplikace může být využívána z různých sítí, musí být API veřejně dostupné. API bude napsáno v programovacím jazyce, který je vhodný pro serverové technologie, v tomto případě půjde o PHP s použitím frameworku Laravel.

Aplikace musí zpracovávat všechny API dotazy odesílané z klientské aplikace a při případném neúspěchu operace dle potřeby informovat o chybě.

Primární funkce API bude obsluhovat požadavky na kvíz, který si bude uživatel spouštět – vygenerování otázek, odeslání další otázky, vyhodnocení po ukončení kvízu, obsluha nápověd, nakupování témat. Dále bude autentifikovat uživatele odesílající požadavky a ukládat do databáze jeho výsledky a změny v účtu.

Kapitola 3

GUI výukové aplikace

Ze sekce o uživatelském průzkumu 2.2 vyplývá, že podstatnou a viditelnou částí práce bude výuková aplikace, kterou budou uživatelé využívat k interakci s obsahem a učení se. Tato kapitola popisuje, jaké jsou cíle a potřeby pro uživatelské rozhraní, základní principy UX, jak a s jakými nástroji vzniká GUI (graphical user interface). Dále bude popsáno, jak konkrétně vznikalo GUI k mobilní aplikaci v tomto řešení, jaké k tomu byly využity nástroje a jakých cílů mělo dosáhnout.

3.1 UX a GUI

V této práci bude vytvářeno GUI pro mobilní aplikaci a pro systém na vkládání obsahu.

User Experience je téma zabývající se zážitkem uživatele používáním produktu. Uživatel má z užívání produktu zážitek v každém případě; prací UX designera je udělat tento zážitek co nejlepší v kombinaci s požadavky a cíli produktu. Proces zlepšování UX se nazývá User Experience Design a kromě jiného zahrnuje výzkum pro porozumění požadavků a potřebám uživatelů a podniku, který produkt vytváří. Po specifikaci požadavků lze vytvářet řešení a následně je měřit [8].

UI (uživatelské rozhraní, z angl. User Interface) je bod interakce a komunikace mezi člověkem a počítačem. To může zahrnovat obrazovky, klávesnice, myš a aktuální stav plochy. Uživatelská rozhraní mohou nabývat několika typů, například CLI (rozhraní příkazového řádku), TUI (textové či dotykové UI), VUI (hlasové UI), GUI (grafické UI) a další.

GUI zahrnuje elementy jako okna, vysouvací menu, tlačítka, ikony a posouvací prvky. Speciálním případem jsou pak mobilní rozhraní, jejichž primární zájem je vytvoření použitelných a interaktivních rozhraní na menších zařízeních, smartphonů a tabletů[3].

K navrhování GUI existuje řada nástrojů, některými z nich jsou Adobe XD nebo Figma. V těchto nástrojích lze navrhovat webové stránky, desktopové či mobilní aplikace, loga nebo třeba vytvářet wireframy.

3.2 Funkce

Mobilní aplikace by měla obsahovat následující funkce:

- **Cvičení** – jedná se o obrazovku, na které uživatel bude vybírat odpověď na zadání či otázku. Těchto cvičení bude několik, a to z důvodu zvýšení angažovanosti uživatele. Cvičení budou krátká (vyplnitelné v rámci jednotek či nižších desítek sekund), aby uživatel udržel pozornost a motivaci.

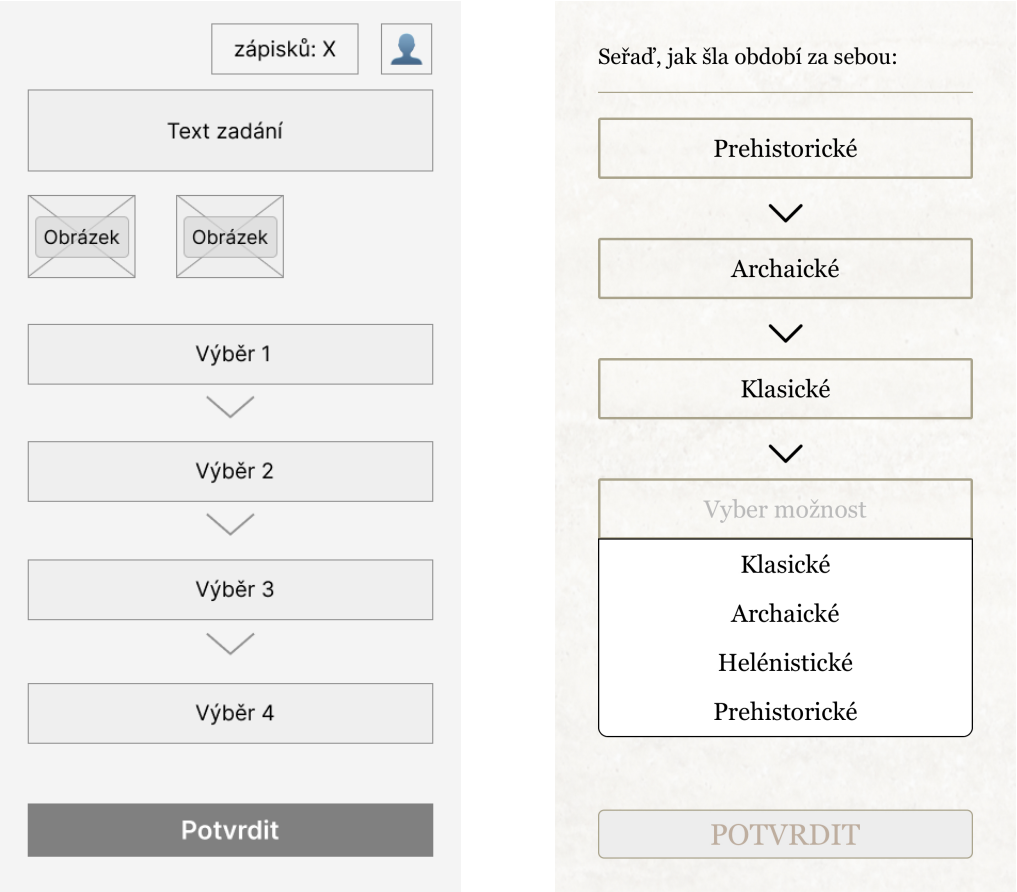
- **Výběr témat** – uživatel si bude moci vybrat, jakou část předmětu si chce procvičovat, tedy která témata. V závislosti na kvantitě obsahu lze uvažovat nad situací, kdy by bylo vhodné obsah rozdělit na podčásti – při DVK lze uvažovat nad dělením tématu na podtémata podle druhu umění, tedy architektura, malba, užité umění, sochařství atd. Uživatel by si pak mohl zvolit kombinaci celých témat nebo specifických podčástí (v tomto případě by pak bylo namístě přidat možnost filtrace).
- **Získávání zpětné vazby** – po zodpovězení otázky bude vyhodnoceno, zda uživatel odpověděl správně (případně částečně správně) a pokud ne, jaká odpověď byla správná. Dále bude uživateli předána zpětná vazba ve formě správně odpovědi a doplňujícího textu a obrázků souvisejících s otázkou, které si uživatel může dle libosti prohlédnout.
- **Náhodná fakta** – v načítací obrazovce mezi otázkami se budou zobrazovat náhodná fakta týkající se témat, které si uživatel pro kvíz zvolil
- **Získávání odměn** – za správné (nebo částečně správné) zodpovězení otázky bude uživateli připsána odměna ve formě tzv. *zápisků*. Zápisky se budou chovat jako herní měna a uživatelé si za ně budou moci kupovat nápovědy.
- **Nákup témat** – jedním ze způsobů útraty zápisků bude nákup dalších témat. Tento mechanismus by měl uživatele přimět k častějšímu opakování, čímž získá dost zápisků na to, aby si mohl koupit další téma
- **Využití nápovědy** – přímo v otázce si bude uživatel moci pořídit za zápisky jednu z nápověd, která mu pomůže s řešením. Nápovědy budou následující:
  - *50 na 50* – ve cvičení se z poloviny omezí možnosti, jak může uživatel zodpovědět.
  - *Tahák od spolužáka* – bude stát nejméně zápisků (nebo žádné), ale je šance, že je tahák napsaný špatně. Přidá prvek napětí a pokud uživatel nebude mít ponětí, jaká je správná odpověď, může alespoň zkusit spolužákův tahák.
  - *Rada od učitele* – bude stát nejvíc zápisků a cvičení se z větší části samo vyplní.
- **Statistiky v profilu** – na profilu bude mít uživatel shrnutí o tom, kolikrát dokončil kvíz v daném tématu a s kolika chybami bylo posledních 10 kvízů.
- **Denní mise** – uživatel bude každý den dostávat úkol, který bude plnit pro získání odměny navíc (zápisků).

3.3 Postup návrhu GUI

Při návrhu GUI bylo dbáno na správný postup pro jeho vytváření. V první fázi byly analyzovány cíle aplikace, potřeby uživatelů a specifikován rozsah funkcí aplikace. Na základě těchto parametrů začaly nejdříve vznikat hrubé skici a z nich později tzv. *wireframy*. *Wireframe* je plánovací dokument obsahující veškeré prvky, které mají být ve výsledném GUI. Rozdíl mezi wireframem a GUI spočívá v tom, že ve wireframu nejsou prvky nikterak stylizovány a esteticky upravovány. Je to technický dokument určen pouze pro rozložení prvků, specifikaci jejich významu a zdůraznění důležitosti v kontextu okolí[8].

Pro tvorbu wireframů i výsledného GUI byl po zvážení dalších alternativ vybrán nástroj Figma. Ten lze využít na rychlé skicování, tvoření wireframů i tvorbu detailního GUI.

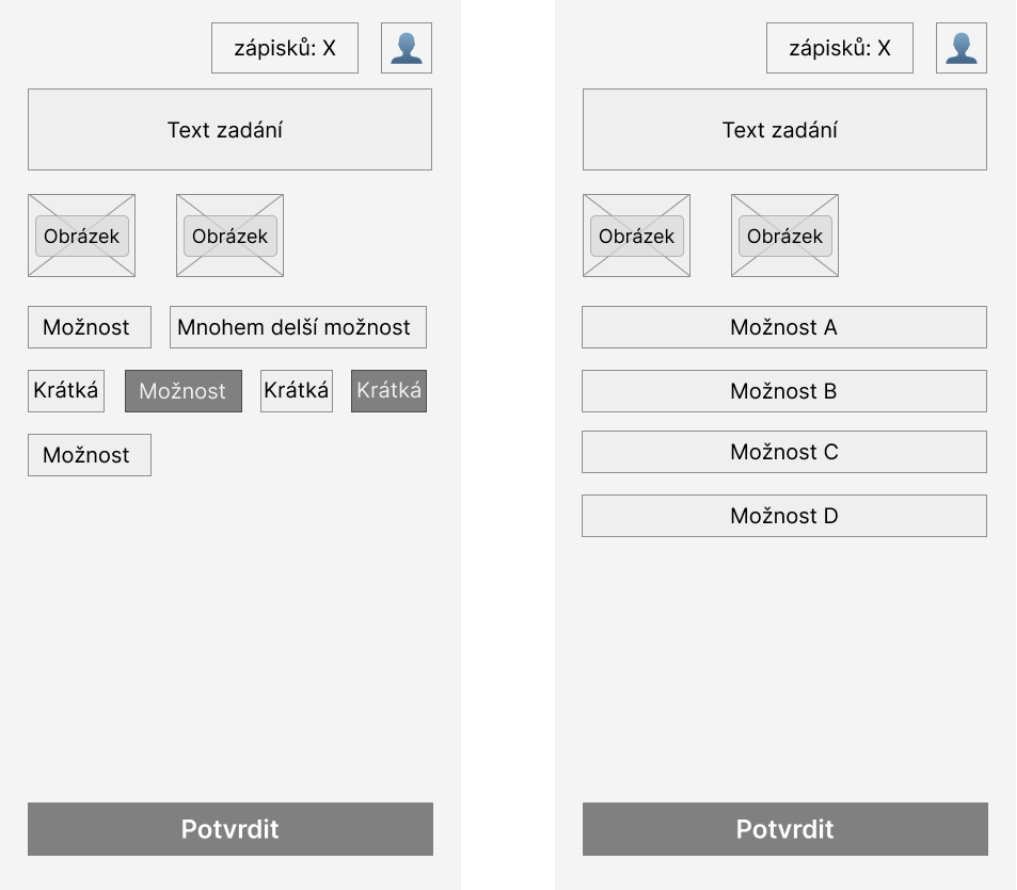
Při návrhu wireframu pro konkrétní cvičení ve kvízu byl kladen důraz na to, aby byla jednoduše pochopitelná, vyplnitelná v krátkém čase a zároveň poskytovala dostatečný prostor na možnost způsobit chybu při vyplňování. Další inspiraci pro cvičení poskytly i aplikace rozebrané v kapitole 2.



Obrázek 3.1: Vlevo wireframe pro cvičení typu *seřazení*. Vpravo hotové GUI. Lze si všimnout, že prvky jsou na stejném místě, změnily se barvy a stylování.

Navržená cvičení jsou:

- **ABCD** – Čtyři možnosti, z nichž je jedna správnou odpovědí. Do budoucna lze upravit do odpovědi skládajících se z více možností. Jedná se o základní a známý typ cvičení.
- **Seřazení** – Několik polí s rozbalovací nabídkou, ve které se nacházejí možnosti, které je potřeba správně zvolit za sebe. Toto cvičení vzniklo specificky pro seřazení např. uměleckých období či dat narození autorů. Vychází z výsledků průzkumu v kapitole 2.3, že pro studenty je problematické seřazení událostí v čase za sebou.
- **Vícenásobný výběr** – Vícero položek, ze kterých lze vybrat libovolný počet a tím odpovědět na otázku. Cvičení je mířené hlavně na přiřazení prvků k pojmu – například techniky malby k autorovi nebo architektonické prvky k období. Wireframe lze vidět na obr. 3.2.



Obrázek 3.2: Vlevo je wireframe pro otázku typu **Vícenásobný výběr**. Vpravo se nachází wireframe pro obrazovku se cvičením **ABCD**.

Vzniklé wireframy byly ověřovány se 2 absolventy SŠ s maturitní zkouškou z DVK. Při tomto ověřování bylo dbáno primárně na to, aby byl jasný význam jednotlivých prvků a uživatel věděl, kam ho přenese kliknutí na jednotlivá tlačítka. Po zakomponování změn začalo vznikat GUI.

Hlavním námětem pro GUI byl maturitní předmět, pro který je tvořeno – v tomto případě dějiny výtvarné kultury. Vzhled byl navrhován tak, aby aplikace na první pohled působila, že se týká historie. Z toho vycházela i volba pozadí, kde je zvolena jen lehce viditelná a nerušící mramorová textura doprovázející uživatele napříč aplikací. Hlavními barvami jsou světlé odstíny hnědé barvy, téměř bílá a černá pro text. Vypracování GUI podle wireframu zhlédnout na obrázku 3.1.

Uživatele počas používání aplikace doprovází maskot zobrazený na obr. 3.3. Ten vznikl za účelem poskytování uživateli vizuální zpětné vazby na situaci v aplikaci – například při dokončení kvízu je delfín pozitivní, při zobrazení faktu u načítání se tváří vážně. Maskot navíc vytváří emocionální vazbu mezi uživatelem a aplikací, což může zvýšit chuť uživatele si aplikaci znovu spustit. Inspirací pro maskota je tzv. *Freska delfínů*, která vznikla mezi rokem 1800-1400 př.n.l. a byla nalezena v paláci ve městě Knósos[1].

[1] https://theartwolf.com/masterworks/dolphins-of-knossos/
[3] https://www.mundo.cz/sites/default/files/images/kreta/kreta-knossos-delfini.jpg

Obrázek 3.3: Vlevo dílo *Freska delfínů*[3], vpravo dílem inspirovaný maskot projevující emoci nápadu, případně úspěchu. Delfín je také symbolem inteligence.

Kapitola 4

Výuková aplikace

Tato kapitola kompletně popisuje realizaci mobilní aplikace. Nejprve jsou definovány cíle a potřeby a dále shrnuta teorie a nástroje k tvorbě mobilních aplikací. Z cílů a teorie dále vychází návrh funkcí a technických vlastností aplikace, které má výsledná realizace aplikace splňovat. V neposlední řadě se v kapitole nachází implementační detaily, které zdůrazňují důležité nebo zajímavé části implementace.

4.1 Nástroje pro vývoj mobilních aplikací

V návaznosti na kapitolu 2.5 bude k vyhovění požadavkům potřeba realizace mobilní aplikace s uživatelským rozhraním. Dnes na trhu dominují 2 operační systémy, na které je mířená většina nově vznikajících aplikací. Jedná se o operační systém Android od společnosti Google a iOS od společnosti Apple. Dle statistik z prosince 2022[1] aktuálně využívá Android přibližně 72 % českých uživatelů a iPhone 27,5 %. Z toho lze usoudit, že implementování aplikace pouze pro Android by znamenalo, že stále čtvrtina studentů středních škol nemá jak aplikaci využít.

Při vývoji na mobilní zařízení je důležité porozumět architektuře jejich operačních systémů. V případě vývoje nativní i hybridní aplikace tyto informace pomohou např. při ladění chyb a optimalizaci výkonu.

Vývoj pro Android

Android je operační systém založený na upravené verzi Linuxového jádra a dalšího open-source softwaru navrženého primárně pro zařízení s dotykovým displejem. Jeho nejpopulárnější verze je vyvíjena společností Google.

Pro vývoj nativních aplikací na Android se používá programovací jazyk Java a Kotlin a jako nejznámějším vývojovým prostředím je Android Studio. Zmíněné jazyky lze dle potřeby kombinovat v rámci jednoho projektu.

Architektura se dělí na[1]:

- **Linux Kernel** – základní stavební blok platformy. Použití tohoto jádra umožňuje Androidu využít základní bezpečnostní funkce a umožňuje výrobcům zařízení vyvíjet hardwarové ovladače pro již dobře známý kernel.
- **Hardware Abstraction Layer (HAL)** – poskytuje standardní rozhraní, které zpřístupňuje hardwarové možnosti Java API frameworku. HAL se skládá z několika mo-

[1] https://gs.statcounter.com/os-market-share/mobile/czech-republic/#monthly-202012-202212

dulů knihoven a každý z nich implementuje rozhraní pro specifický typ hardware komponenty. Když framework API zažádá o přístup k hardwaru, Android načte modul knihoven pro práci s tímto zařízením.

- **Android Runtime** – od verze 5.0 se spouští každá aplikace ve svém vlastním procesu a s vlastní instancí ART (Android Runtime). ART je vytvořeno pro spouštění několika virtuálních strojů na nízkopaměťových zařízeních.
- **Nativní C/C++ knihovny** – mnoho komponent Androidu (jako ART nebo HAL) jsou sestaveny z nativního kódu, který vyžaduje nativní knihovny napsané v C a C++. Některé tyto nativní knihovny jsou přístupy z Java framework API (např. Java OpenGL API pro kreslení a manipulaci s 2D a 3D grafik).
- **Java API Framework** – celá sada Android OS funkcí je přístupná skrze API napsaná v jazyce Java. Tyto API vytváří stavební bloky, které jsou potřeba pro vývoj aplikací, a to zjednodušením opětovného použití modulárních komponent a služeb systému, což zahrnuje systém zobrazení (pro stavbu UI), správce zdrojů (zpřístupňuje zdroje jako lokalizované řetězce a grafiku), správce upozornění (umožňuje aplikacím zobrazit upozornění ve stavovém řádku) a správce aktivit (spravuje životní cyklus aplikace a poskytuje navigační stack).
- **Content providers** – umožňuje aplikacím přistupovat k datům jiných aplikací nebo sdílet vlastní data.

Vývoj pro iOS

Označení iOS znamená iPhone operating system. Je to uzavřený mobilní operační systém společnosti Apple pro svá handheld zařízení (volně přeloženo jako příruční zařízení). Po Androidu se jedná o druhý nejpopulárnější operační systém pro mobilní zařízení a kromě mobilních telefonů běží na tomto operačním systému například i iPad nebo iPod od společnosti Apple.

Pro svůj vývoj podporuje jazyky C, C++, Objective-C a Swift. Nejznámější vývojové prostředí je Xcode. Vrstvy architektury operačního systému jsou následující[12]:

- **Core OS** – nachází se na nejnižší vrstvě u samotného hardwaru. Nízkoúrovňové funkce, které formují základ všech iOS funkcí, jsou zpřístupněny touto vrstvou. Kromě standardních funkcí základního operačního systému (správa paměti, souborového systému, vláken atd.) nabízí řadu služeb jako nízkoúrovňovou obsluhu sítí a přístup k externímu příslušenství.
- **Core Services** – touto vrstvou jsou abstrahovány služby nabízené Core OS. Poskytuje základní služby, které mohou použít všechny aplikace, a sadu frameworků (např. Account Framework pro přístup k externímu účtu přímo z aplikace či Foundation framework pro základní funkce aplikace jako ukládání dat, zpracování textu či počítání dat, atd.).
- **Android Runtime** – od verze 5.0 se spouští každá aplikace ve svém vlastním procesu a s vlastní instancí ART (Android Runtime). ART je vytvořeno pro spouštění několika virtuálních strojů na nízkopaměťových zařízeních.

- **Media Layer** – vrstva pro používání multimédií. Umožňuje systému používat grafiku, audio a video technologie. Vývojáři poskytuje možnost pracovat s animacemi, fotografiemi, audiem a videi. Také obsahuje řadu frameworků, které lze využít na práci s médii.
- **Cocoa Touch** – poskytuje abstraktní vrstvu, která různé knihovny pro iOS zařízení dělá přístupné pro programování. Nejzákladnější skupina Objective-C frameworků vytvořena pomocí Mac OS X Cocoa API jsou zahrnuty v Cocoa Touch vrstvě. Tato vrstva poskytuje například podporu notifikací, multitaskingu a specifických vstupů na dotyk.

Hybridní řešení

Jednou z možností, jak aplikaci spustit na obou zmíněných operačních systémech, je za použití takzvaného webview. Webview se rozumí aplikace, jejíž účel je promítat předem zvolenou webovou stránku a chovat se jako prohlížeč, který je však zproštěn záložek a ovládacích prvků pro prohlížeč specifické. Tento způsob umožňuje vývojářům vytvořit pouze jednoduchou aplikaci, která bude pomocí webview prvků promítat obsah existující na webu, optimalizovaný pro zobrazení na mobilním zařízení. Nevýhodou tohoto řešení je menší výkonnost, dlouhá odezva na akci uživatele a stálé zobrazování webové stránky, která se může na pomalejším připojení nahrávat postupně, což vede k horšímu uživatelskému zážitku a menší kontrolou nad jednotlivými prvky aplikace.

Dalším způsobem, jak umožnit aplikaci fungovat na více operačních systémech, je kombinace webového a nativního přístupu – tzv. hybridní. Hybridní aplikace je psána ve webových technologiích (HTML, CSS, Javascript) a běží zapouzdřená v nativní aplikaci. Největší výhodou samozřejmě je dostatečnost jedné aplikace pro více operačních systémů a oproti webview je rychlejší a interaktivnější. Tento přístup je dnes populární a existuje množství technologií, které například nabízí běžné UI prvky, které hybridní aplikace, narozdíl od nativní, nemá v základu k dispozici. Nevýhodou hybridní aplikace je, že se zvyšující se programovou kontrolou je třeba v průběhu vývoje stále dbát ohled na rozdílné chování některých prvků a funkcí na různých operačních systémech a případně i zařízeních. Mezi známé technologie patří například React Native, Xamarin nebo Ionic[5].

4.2 SW architektura mobilní aplikace

Mobilní aplikace bude implementována ve frameworku React Native, který zajistí fungování na operačních systémech Android a iOS. Dále bude použit nástroj Expo poskytující, kromě jiného, možnost bezplatně zveřejnit vyvíjený projekt a stáhnout si jej pomocí aplikace *Expo Go* s použitím QR kódu nebo odkazu.

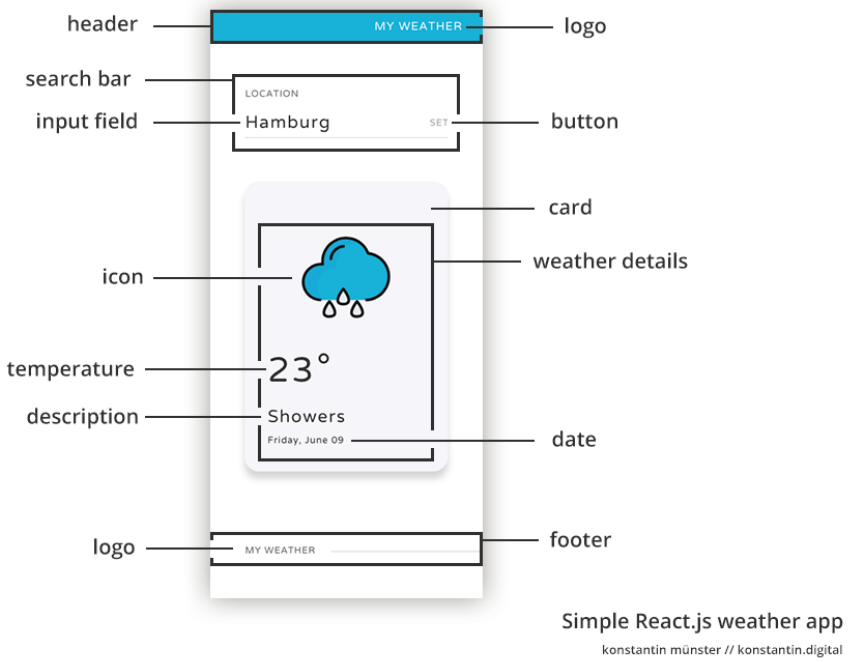
Pro verzování kódu bude použit verzovací systém *Git*. Tento systém ukládá historii změn v kódu a je dnes již standardem při vývoji projektu v týmu. Nicméně možnost revidovat změny a verzovat kód je užitečné i pro jednotlivce.

React Native

React Native je populární framework pro Javascript a slouží k vývoji hybridních mobilních aplikací pro iOS a Android. Je vyvíjen společností Facebook a byl zveřejněn v roce 2015. Zdrojové kódy jsou dostupné na GitHubu. React Native vychází z frameworku React.js, který je užíván pro vývoj frontendových částí webových aplikací.

Hlavními prvky Reactu jsou:[2]

- State – proměnné, které reprezentují aktuální stav komponenty, mohou se měnit v závislosti na vstupu uživatele nebo třeba obdržení odpovědi na HTTP dotaz.
- Props – parametry, které jsou předány komponentě při jejím vytvoření. Tyto parametry jsou pouze pro komponentu pouze pro čtení.
- Komponenty – znovupoužitelné bloky, ze kterých se skládá výsledné UI. Komponenty mohou i nemusí mít vlastní funkcionalitu, čímž se pak stávají pouze vizuálními prvky. Komplexní komponenty mohou být složeny z několika jednodušších nebo atomických komponent, což ve výsledku tvoří hierarchii, která zpřehledňuje zdrojový kód a snižuje jeho duplicitu. Na obrázku 4.1 je příklad komponent tvořící dohromady kompletní rozhraní.



Obrázek 4.1: Příklad komponent v mobilní verzi webové aplikace. Lze vidět komponenty s funkcí (*input field* pro vyhledávání), bez funkcí (*header*) a komponenty obsahující jiné komponenty (*card* obsahuje *weather details*). Obrázek byl převzat[4].

- JSX (JavaScript XML) – slouží pro deklaraci vizuálních prvků. Jedná se o syntaktické rozšíření pro JavaScript, které od pohledu připomíná standardní zápis HTML. Protože se stále jedná o JavaScript, lze podmíněně vykreslovat prvky, vkládat atributy či používat elementy jako proměnné. Rozdíl v JSX mezi Reactem a Reactem Native spočívá v tom, že React využívá typické HTML elementy (<p>, , <img>, <button> aj.) zatímco React Native využívá tzv. core elementy (např. <Text>,

[2] https://reactnative.dev/docs/
[4] https://konstantin.digital/blog/how-to-...

<View>, <Image> a <Button>) na pozadí se vykreslí jako nativní elementy daného systému (např. <Text> se vykreslí jako prvek TextView pro Android a `UITextView` pro iOS).

Kombinace technických vlastností, popularity (v roce 2022 kolem milionu stažení týdně[5]) vázající se k množství dostupných materiálů, návodů a knihoven dělají z React Native vhodného kandidáta na vývoj mobilní aplikace k realizaci cíle práce.

Expo

Expo je sada nástrojů a služeb vytvořených pro React Native usnadňující vývoj aplikací. Vývojář si nemusí instalovat Android Studio pro vývoj na Android či XCode pro iOS. K vytvoření a manipulaci s projektem slouží Expo CLI a Expo Go zase pro spuštění aplikace v mobilním zařízení. Mezi další výhody patří jednoduché sdílení vyvíjené aplikace za pomocí odkazu nebo QR kódu – stačí, aby na daném zařízení byla nainstalována aplikace Expo Go dostupná z Google Play a App Store.

Pro přístup k systémovým funkcím zařízení (kamera, gyroskop, souborový systém atd.) Expo poskytuje vlastní rozhraní. Pokud je však třeba přistoupit k rozhraní, které aktuálně není nabízeno (například bluetooth) nebo jednoduše použít nativní kód, omezí to některé funkce Expa (například zmíněné sdílení pomocí Expo Go) a není to doporučováno[6].

Z povahy realizované aplikace v rámci práce lze odhadnout, že pravděpodobně nebude třeba konkrétního rozhraní, které Expo nenabízí. Výhoda jednoduchého sdílení aplikace bude klíčová při provádění experimentů a testování mezi studenty. Zároveň Expo nabízí snadný způsob publikování a vydávání aktualizací aplikace na App Store a Google Play, kam bude v pozdější fázi aplikaci vhodné umístit a nechat ji otestovat širší veřejností.

4.3 Implementační detaily

Implementace proběhla v JavaScriptovém frameworku React Native s použitím platformy Expo a vývojového prostředí Visual Studio Code. Pro vytvoření projektu a správu závislostí byl použit nástroj pro správu balíčků a závislostí *NPM* (Node Package Manager).

Struktura projektu

Po vytvoření Expo projektu se vytvoří pouze několik souborů (mezi nimi App.js, který je výchozím vstupním bodem aplikace) a adresářů `assets` a `node_modules` pro assety, respektive knihoven. Je žádoucí si již na začátku vytvořit základní adresářovou hierarchii podle toho, jak chceme, aby byl projekt organizován a v průběhu vývoje tuto strukturu dodržovat a rozvíjet. V této práci se téměř všechny zdrojové soubory nachází v hlavním adresáři *src*, který obsahuje tyto podadresáře:

- components – zde se nacházejí komponenty, které se nevztahují ke konkrétní funkci aplikace.
- features – obsahuje adresáře, které společně tvoří kompletní funkci. Obvykle se adresář funkce dělí na *components* s komponentami a *screens* s obrazovkami pro danou funkci.
- infrastructure – zde jsou zdrojové kódy definující téma a navigátory aplikace.

[5] https://www.npmjs.com/package/react-native
[6] https://docs.expo.dev/

- services – obsahuje kód, jehož primární účel je dotazování se na data a jejich předávání za použití kontextu.
- utils – adresář se zdrojovými soubory definující různé konstanty použité skrze aplikaci.

Důležité technologie

V projektu jsou využity nástroje a knihovny usnadňující implementaci některých běžně se vyskytujících funkcí nebo zlepšující organizaci kódu.

Jednou z těchto technologií jsou *Styled-Components*[7]. Je to nástroj, který přináší vizuální primitivy pro komponenty a více zpřístupňuje CSS stylování pro systém komponent Reactu. V projektu je stylování řešeno téměř výhradně s použitím styled-components. Tyto vizuální komponenty jsou nejčastěji definovány ve stejném souboru, ve kterém jsou i používány – například v souboru `items-grid.component.js` je definován vizuální primitiv `Item`, který je v souboru použit v komponentě `ItemsGrid`. Některé vizuální komponenty jsou definovány zvlášť, protože se používají skrze celý projekt – jedná se například o `Text`.

Další podstatnou technologií je `React Navigation`. Je to modul poskytující řešení pro navigaci po aplikaci. Umožňuje využít běžnou zásobníkovou a záložkovou navigaci, což poskytuje uživateli přirozený pohyb po aplikaci gesty a prvky, které již zná. Pro použití tohoto modulu bylo vytvořeno několik navigátorů, které se navzájem přepínají dle toho, v jaké části aplikace se uživatel nachází. Jedním z navigátorů je `QuizNavigator`, který se používá v průběhu kvízu na přepínání mezi obrazovkami cvičení, načítání a dokončení.

Komunikace pomocí Promises a cyklus kvízu

Komunikace se serverem probíhá s použitím objektů typu Promise[8], který reprezentuje eventuální dokončení nebo selhání asynchronní operace a její výsledné hodnoty. U obou zmíněných výsledků lze zvlášť definovat chování a zachytit tak případné chyby, uživateli o tom podat informaci, případně spustit jinou akci na pozadí. V projektu se Promises používají pro každé odeslání požadavku na API a přijetí odpovědi.

Aplikace se po přihlášení dotáže na témata, které se po přijetí uloží do lokálního kontextu a poté se podle nich vykreslí položky v obrazovce s výběrem témat. Jakmile si uživatel vybere témata a klikne na tlačítko „SPUSTIT“, cyklus kvízu funguje následovně:

1. Odešle se požadavek na vygenerování kvízu a čeká se na obdržení první otázky s odpověďmi.
2. Při čekání na otázku se změní obrazovka na načítání a zobrazí se fakta.
3. Po přijetí odpovědi ze serveru načítací obrazovka vyzve uživatele k pokračování a otevře se otázka.
4. Až uživatel zodpoví, odešle se požadavek na vyhodnocení otázky a získání zpětné vazby, která má být uživateli zobrazena. Při tomto čekání se nemění obrazovka na načítací, pouze se spustí jednoduchá animace v tlačítku, na které v tuto chvíli nelze znovu kliknout.
5. Po získání dat je zobrazena obrazovka s dodatečnými informacemi a zpětnou vazbou. Nyní uživatel pokračuje na další otázku a pokud otázka není poslední, odesílá se

[7] https://styled-components.com/
[8] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

požadavek na další otázku a cyklus se opakuje od 2. bodu dokud nebylo dosaženo poslední otázky.

6. Jakmile uživatel dosáhne konce kvízu, je zobrazena finální obrazovka s informacemi o počtu správně a špatně zodpovězených otázek z témat a získaných zápiscích. Uživatel může nyní buď zapnout kvíz se stejným výběrem témat nebo odejít do menu.

Kapitola 5

Serverová aplikace

Požadavky mobilní aplikace budou zpracovávány serverou aplikací, o které pojednává tato kapitola. Nejprve jsou popsány cíle a potřeby, ze kterých vychází nástroje a přístupy používané k vývoji serverových aplikací.

Kapitola zahrnuje i databázi, SQL a způsob uchování dat, aby mohla být v budoucnu znovu využita. K tomu se vztahuje výběr konkrétního systému řízení báze dat.

5.1 Nástroje a technologie

Druhou částí práce je serverová aplikace, která bude zpracovávat HTTP požadavky z klientské mobilní aplikace a zároveň poslouží pro správu obsahu z webového rozhraní. K této aplikaci bude potřeba databáze k uchovávání dat.

Požadavkem na serverovou aplikaci je schopnost obsluhovat všechny HTTP požadavky mobilní aplikace týkajících se funkcí, které jsou definované v kapitole 3.2. V souvislosti s těmito funkcemi bude třeba práce s databází a uchovávanými daty.

K postavení serverové aplikace existuje řada přístupů – jedním z nich je BaaS – jedná se o model cloudové služby, kam mohou vývojáři outsourcovat všechny backendové[1] aspekty webové či mobilní aplikace. Poskytovatelé BaaS[2] služeb nabízejí, kromě cloudového úložiště a hostingu, předpřipravený software na aktivity, které se obvykle na serveru dějí – autentifikace uživatele, správa databáze, push notifikace atd.

Výhodou těchto služeb je, že se vývojáři mohou soustředit na klientskou část aplikace a lze rychle navrhnout backend s využitím zmíněných služeb. Nevýhodou je, že z hediska dlouhodobého vývoje mohou vzniknout situace, kdy je třeba implementovat funkci, kterou daný BaaS poskytovatel nepodporuje. V tom případě je potřeba tuto funkci přizpůsobit možnostem poskytovatele nebo zcela zavrhnout.

Pro zmíněné důvody a osobní preferenci autora práce bude zvolen přístup s vytvořením výukové aplikace a využitím REST API. Kód bude v jazyce určeného pro serverové prostředí nebo s rozšířením umožňující psát serverově orientovaný kód. Mezi tyto jazyky patří například Python, PHP, C# a Ruby nebo třeba JavaScriptové prostředí NodeJS. Pro tuto aplikaci byl zvolen jazyk PHP, jehož přednostmi jsou stabilita, rychlost, řada open-source frameworků a knihoven a dostupnost materiálů. Dle webu w3techs.com[3] je k lednu

[1] Backend – označuje „zadní část“ – aplikace. Jedná se o část systému, která není přímo přístupná uživatelem a typicky se používá pro manipulaci a uchovávání dat. Opakem backendu je frontend, což je „přední část“ aplikace, se kterou interagují uživatelé.

[2] Backend-as-a-Service

[3] https://w3techs.com/technologies/details/pl-php

2023 PHP používané na přibližně 77.8 % všech webových stránek, u kterých je znám jazyk na straně serveru. Pro zefektivnění práce bude využit open-source framework *Laravel*. Při vývoji serverové aplikace bude také použit verzovací systém *Git*.

Laravel

Laravel je PHP framework pro tvorbu webových aplikací. Nabízí širokou knihovnu předpřipravených modulů pro funkce běžné (autentifikace uživatele, routing[4], validace přijatých dat, mailing...) i pokročilé (lokalizace, cache, fronty, práce s lokálními i vzdálenými souborovými systémy...) a silně nabádá k psaní organizovaného a strukturovaného kódu.

Jedná se o framework užívající Model-View-Controller (MVC) architekturu. Model je zodpovědný za data aplikace s celou logikou. Konkrétně se může jednat například o vytvořené entity. View je bod interakce s uživatelem. Zde jsou data z Modelu poskytovány uživateli ke čtení a modifikacím. Do View patří například šablony. *Laravel* konkrétně využívá šablonovací engine Blade, který však není k této práci relevantní, protože cílem je na HTTP požadavky odesílat formátovaná data JSONu, nikoli ve stylovaném HTML.

Vrstva Controller propojuje Model a View. Jinými slovy, Controller přijme požadavek, získá a zpracuje patřičná data z Modelu a vloží do View, které odešle jako odpověď. Mezi další užitečné funkce, které framework nabízí, patří například routování API požadavků, autentifikace studentů, migrace pro postupné modelování databáze a *Eloquent ORM*[5] pro práci s databází a entitami.

Kromě zmíněných funkcí *Laravel* disponuje příkazovým řádkem nesoucí název *Artisan*, který nabízí spoustu užitečných příkazů různě použitelných po celou dobu vývoje. Jedním z běžných příkazů je artisan `make:model User`, který vytvoří soubor pro model s názvem dle parametru, v tomto případě *User*[7].

Twill

Protože se bude v této práci průběžně přidávat a aktualizovat obsah, je nutné poskytnout správci obsahu způsob jak toho docílit. Jedním pohodlným způsobem může být webové rozhraní, do kterého se uživatel pomocí účtu přihlásí a bude mít přístup ke správě obsahu.

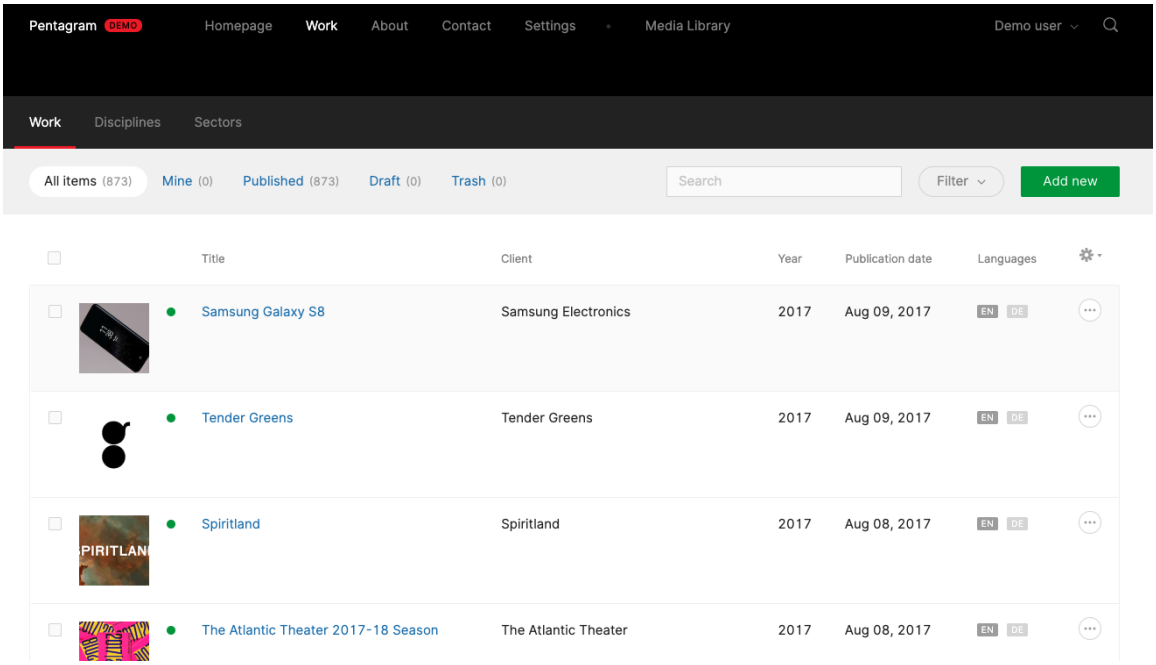
Twill je open-source balíček pro *Laravel*, který umožňuje snazší vytvoření vlastního CMS[6]. Kromě běžných administračních funkcí (vkládání a úprava obsahu) nabízí i autentifikaci uživatelů, předpřipravené přiřazování mediálních souborů k entitám či blokový editor, kde se uživateli při úpravě v administraci zobrazí náhled výsledného obsahu u klienta. *Twill* s sebou nese vlastní knihovnu UI komponent a lze se tedy zaměřit pouze na konfiguraci menu a přehledu s formulářem pro úpravu entit. Při běžném použití není třeba psát vlastní HTML, výchozí rozhraní lze vidět na obrázku 5.1.

Obsah CMS je dělen na moduly, kdy každý z nich pracuje s jinými entitami – například při správě blogu s kategoriemi by byly zvlášť moduly pro články, kategorie a dle potřeby i pro samotnou stránku blogu (například na správu SEO nebo určení, který článek bude doporučovaný v úvodní části). Moduly jsou založeny na běžných principech *Laravel*, a proto je ve většině případů poměrně bezproblémové i již existující entity začlenit do CMS. Pro nově vytvářené entity (a k nim běžné zdrojové soubory) nabízí *Twill* příkaz `twill:make:module`, který připraví modul se specifikovanou entitou připravenou na použití v CMS.

[4] Routing - nasměrování požadavku dle URI k příslušné akci, která požadavek zpracuje

[5] Eloquent Object-Relational Mapper (ORM) - každá databázová tabulka má korespondující model, který s tabulkou interaguje

[6] Content Management System (CMS) - systém pro správu obsahu



Obrázek 5.1: Ukázka demo rozhraní *Twillu* pro modul pojmenovaný jako *Work*[8]. Nahoře lze vidět navigaci s jednotlivými moduly, níže je výběr sekcí a základní filtr upravující výpis záznamů. Pro záznamy jsou v přehledu zobrazeny nejdůležitější informace. Při rozkliku záznamu se uživatel dostane do formuláře pro editaci.

Jedná se o tzv. *headless* CMS, což znamená, že *Twill* nevytváří technické požadavky na frontendovou část aplikace. K datům, která se ukládají pomocí *Twillu*, lze v *Laravel* přistupovat běžnými způsoby, a proto i data vytvořená před integrací *Twillu* do projektu lze později napojit pouze pár základními úpravami[9].

5.2 Datový model a jeho realizace

Jedním z požadavků na řešení je realizace datového modelu a trvalé uchování dat z důvodu jejich opětovného využití v budoucnosti.

Uchování dat

Vhodný způsob uchování závisí na povaze dat, které je třeba ukládat. V tomto případě se jedná primárně o tato data:

- Obsah – jednotlivá témata, ke kterým se vážou otázky. Zadání otázky, které se může skládat z textu a obrázků. K otázce patří několik odpovědí a také vysvětlující informace, obojí může obsahovat text i obrázky.
- Výsledky – záznamy o odpovědích uživatele – na jaké otázky odpovídal a zda odpověděl správně, případně jakou odpověď zvolil. Tyto záznamy umožní předat uživateli

[8] https://demo.twill.io/work/works
[9] https://twillcms.com/

zpětnou vazbu např. o jeho úspěšnosti či množství zodpovězených otázek v daném tématu.

Možností je všechna tato data umístit lokálně na uživatelovo zařízení a operace s těmito daty provádět mobilní aplikací. Tím by značně narostla velikost aplikace – pokud by na otázku připadlo zadání o 10 slovech, 4 jednoslovné odpovědi a vysvětlující informace o 50 slovech a 2 obrázcích po 1 MB, velikost otázky by mohla odpovídat zhruba 2.5 MB. Při 30 otázkách na téma je to 75 MB a 750 MB pro 10 témat. Velikost úložiště dnešních smartphonů se průměrně pohybuje od 64 GB výše a při rychlosti slabšího připojení 25 Mbps by stahování aplikace trvalo kolem 3 minut. K obsahu je nutností doplnit samotnou aplikaci s knihovnami, což navýší celkovou velikost o desítky či nižší stovky megabytů. Výsledkem je už poměrně velká aplikace, která by mohla uživatele od stažení odradit. Kromě velikosti je další nevýhodou omezená manipulace s obsahem (nový obsah a jeho úpravy jsou stahovány nárazově) a složitější sbírání informací o tom, jak si uživatelé vedou (bylo by nutné zavést do aplikace průběžné odesílání informací o výsledcích v dané instanci aplikace do centrální databáze).

Odlišnou a preferovanou možností je uchovávat obsah v centrální databázi a do aplikace odesílat jen aktuálně potřebná data. Tím lze obsah upravovat průběžně, aniž by se musela aplikace aktualizovat, čímž se i zajistí stejný obsah pro všechny uživatele napříč různými verzemi aplikace. Požadavky může zpracovávat serverová aplikace, která se zároveň bude dotazovat na obsah z databáze. Pro snížení potřeby přijímat data opakovaně je možné data ukládat (tzv. cacheování) do zařízení. Jako databáze bude použit relační databázový řídící systém, který je více popsán v sekci 5.2.

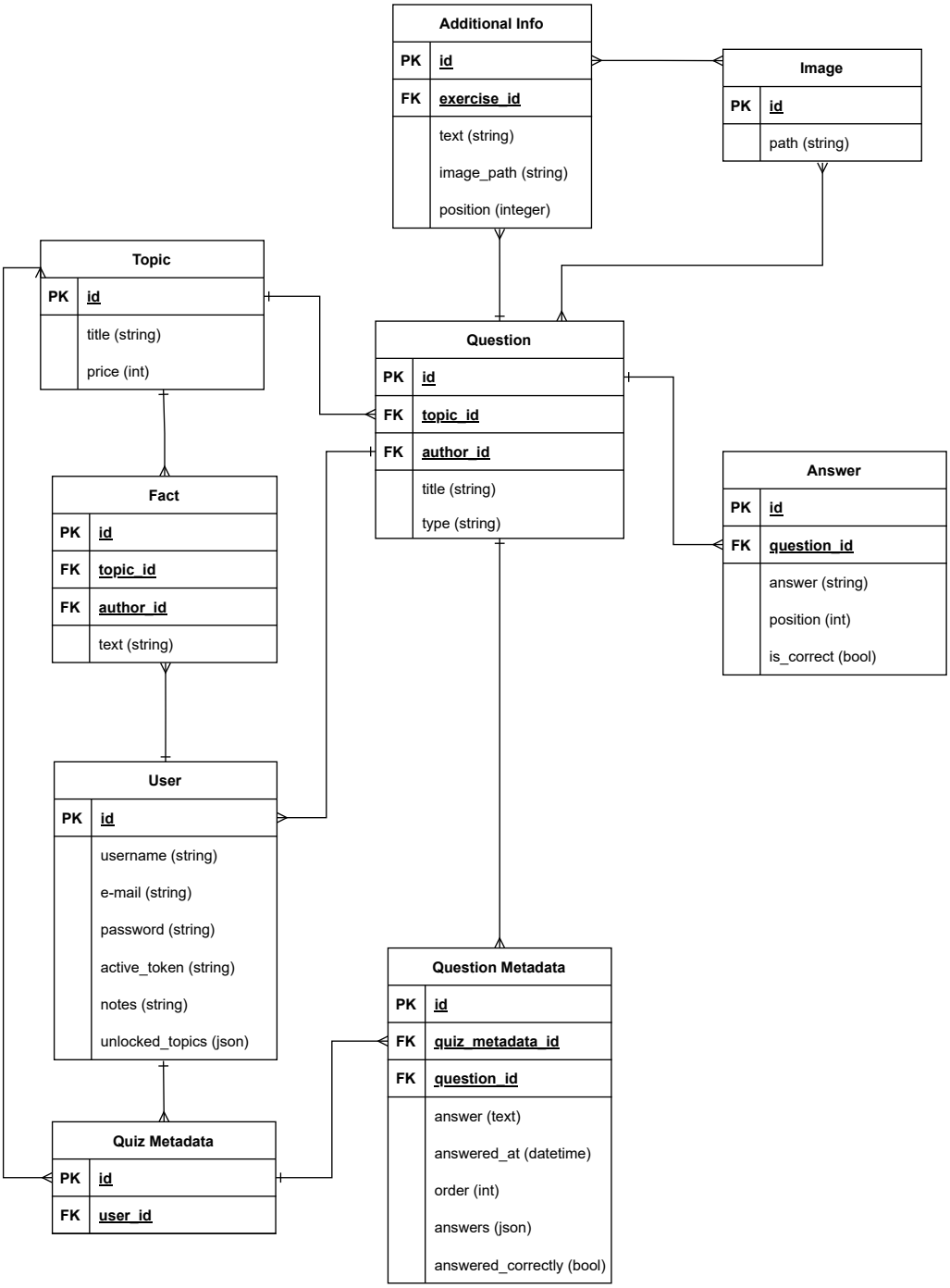
Datový model

Datový model, zobrazen na obrázku 5.2, vznikl s ohledem na požadavky na funkci aplikace v podkapitole 3.2 a zároveň bylo při návrhu dbáno na jednoduchost, prevenci redundanci dat a použití správných vazeb.

Diagram na obrázku 5.2 byl od implementačního stavu pro přehlednost zjednodušen. Nejvíce rozdílů spočívá v integraci *Twillu*, který již obsahuje řešení na některé případy použití. Hlavní rozdíly jsou následující:

- K entitě *Question* jsou obrázky přidávány za pomocí *Twillu*, který používá entity *Media* a polymorfické vazby `mediable` k provázání médií s entitami.
- Místo entity *AdditionalInfo* je použit systém bloků (entita *Block*) od *Twillu*. Pro tyto bloky lze v CMS využít formulářové rozhraní `block_editor`, ve kterém jsou bloky snadno vyplňovány a řazeny za sebe.
- Entity *Question* a *Answers* jsou napojeny na systém revizí, kde takto *Twill* umožňuje sledovat změny. Použití revizí bude užitečné při správě obsahu více uživateli pro sledování a kontrolu provedených změn.
- Tokeny uživatele jsou samostatná entita *PersonalAccessToken*, kterou v základu vytváří *Laravel*. Tato entita obsahuje atributy jako čas expirace (`expires_at`), poslední čas použití (`last_used_at`) a specifické schopnosti tokenu (`abilities`).
- Kromě zmíněných rozdílů má většina entit ještě datum vytvoření (`created_at`), poslední úpravy (`updated_at`) a smazání (`deleted_at`). Tyto atributy se používají např.

pro implicitní řazení při dotazech na databázi a `deleted_at` umožňuje označit entitu jako odstraněnou, aniž by byla úplně smazána z databáze.



Obrázek 5.2: ER diagram používaný v řešení. Pro přehlednost byl zjednodušen na vazby a entity vytvořené výhradně pro řešení – byly vypuštěny entity pro rozšířené funkce CMS a nerelevantní atributy.

Databáze a SQL

Databáze je organizovaná kolekce strukturovaných informací či dat, typicky uložená elektronicky na počítačovém systému. Databáze je obvykle řízená tzv. systémem řízení báze dat (z angl. DBMS - Database Management System). Data a DBMS jsou společně nazývány zkráceně jako databáze.

Dnes jsou nejběžnější databáze relační – data jsou strukturovány do řádků a sloupců v řadě tabulek pro efektivní zpracovávání a dotazování na data. Takto strukturovaná data lze poté jednoduše číst, upravovat, modifikovat, kontrolovat a organizovat. Nejvíce databází používá pro zápis a dotazování na data jazyk SQL (Structured Query Language).

SQL je standardizovaný jazyk pro definici a manipulaci s daty v relačních databázích. Data jsou načtena specifikací výsledné tabulky, která může být odvozena z jedné či více základních tabulek.

Mezi nejznámější open-source relační DBMS patří *MySQL*, který je vyvíjen společností Oracle Corporation. Mimo hlavní výhody, kterými je především škálovatelnost, rychlost a spolehlivost, disponuje tento systém také jednoduchostí instalace. Jeho funkčnost je ověřena v náročných produkčních prostředích technologickými společnostmi jako je YouTube, Facebook, Twitter nebo Uber[11].

5.3 REST API

Serverová aplikace bude realizovat REST API a zpracovávat HTTP požadavky odeslané mobilními aplikacemi. K ověření, zda je požadavek odeslán opravdu uživatelem aplikace, bude použita autentifikace pomocí tokenů. API (z ang. Application Programming Interface) je sada pravidel, která umožňuje rozdílným aplikacím komunikovat spolu navzájem. API se chová jako prostřední vrstva, která zprostředkovává přesun dat mezi systémy. Definice a protokoly v API poskytuje vývojářům rozhraní pro komunikaci mezi aplikacemi, což zjednodušuje integraci aplikací.

Vzdálená API jsou navržena pro interakci skrze komunikační síť. Vzdálená znamená, že API manipulují s daty mimo zařízení, které požadavky vytváří. Nejrozsáhlejší síť je internet, a proto je většina API navrhována na základě webových standardů.

Webová API typicky používají HTTP protokol na odesílání požadavků a poskytnutí dodatečných informací pro specifikování struktury odpovědi. Tyto odpovědi jsou běžně preferovány v XML (Extensible Markup Language) nebo aktuálně již JSON (JavaScript Object Notation) formátu, protože reprezentují data způsobem, který je jednoduchý pro zpracování aplikacemi.

Jedním konkrétním typem vzdáleného API je REST API. Webová API, která dodržují tyto koncepty REST architektury:[6]

1. **Architektura typu klient-server** – architektura je sestavena z klientů, serveru, zdrojů (z angl. resources) a komunikace pomocí HTTP.

2. **Bezstavovost** – na serveru nejsou uloženy žádné informace o klientovi mezi jednotlivými požadavky.

3. **Cacheability** – mělo by být možné ukládat zdroje do cache (česky volně přeloženo jako mezipaměť) na severu nebo klientovi. Cílem je zrychlit výkon na straně klienta a zvýšit škálovatelnost serveru.

4. **Vrstvený systém** – interakce mezi klientem a serverem mohou být zprostředkované dodatečnými vrstvami. Tyto vrstvy mohou nabízet výhody jako vyrovnávání zátěže, sdílené cache nebo bezpečnost.

5. **Jednotné rozhraní** – všechny API požadavky na stejný zdroj by měly vypadat stejně, bez ohledu na to odkud požadavek přišel. Odpověď má zahrnovat zdroj, který nese dostatek informací, aby s ním klient mohl pracovat, ale zároveň je co nejmenší. Pokud je to nutné, musí zpráva odeslaná klientovi obsahovat instrukce jak zdroj zpracovat.

6. **Kód na vyžádání** (nepovinný; z angl. code on demand) – server může rozšířit funkci klienta zasláním kódu, který má spustit. Tento kód by měl být spuštěn pouze na vyžádání klienta.

Návrh API

API bude fungovat s využitím nabízených funkcí frameworku *Laravel*. Jedna z důležitých částí bude tzv. routování – zachycení a rozlišení požadavku dle URI a následné zpracování kontroléry. Kontroléry jsou třídy, které slouží pro přehlednější organizaci spouštěných funkcí. V kontrolérech lze seskupit logiku pro zpracování souvisejících požadavků. V tomto případě například požadavky týkající se kvízu budou začínat cestou v URI `/api/quiz` a bude je obsluhovat `QuizController`. Příklady některých navržených API endpointů[10]:

- `/api/quiz/init` – inicializuje kvíz pro uživatele a odešle mu první otázku společně s dalšími potřebnými informacemi o kvízu
- `/api/quiz/check-result` – zkontroluje výsledek uživatelovy odpovědi, aktualizuje jeho postup v kvízu a odešle zpět výsledek a doplňující informace k otázce
- `/api/topics` – odešle všechny témata, které si uživatel může vybrat před spuštěním kvízu
- `/api/topics/unlock` – na tomto endpointu uživatel žádá o odemknutí tématu
- `/api/login` – uživatel zde žádá o získání autorizačního tokenu (nebo o ověření existujícího) využívaného v průběhu celé komunikace se serverovou aplikací

Na správu závislost pro PHP (a stažení samotného *Laravelu*) bude použit nástroj Composer. Composer umožňuje deklarovat na jakých knihovnách je projekt závislý, nalezne které verze balíčků budou nainstalovány a ty nainstaluje je do složky vendor. Po instalaci balíčků je lze dále přidávat, odebírat či měnit verze.

5.4 Webové rozhraní pro správu obsahu

Pro správu obsahu uživatelem je potřeba vytvořit uživatelské rozhraní, které k tomu uživateli poskytne potřebné nástroje a zároveň se zde bude uživatel schopný orientovat.

Jako univerzální a přizpůsobitelný nástroj pro správu obsahu na webových aplikacích bude využit *Twill*. Ten přichází již s předdefinovanými GUI prvky pro prostředí administrace i pro prvky na vstup od uživatele.

[10] endpoint – bod, kde API obdrží požadavek (v tomto případě ve formě URI) k vykonání nějaké operace

Ve *Twillu* budou vytvořeny formuláře s prvky sloužícími pro vstup dat od uživatele, bude se jednat o moduly *Otázky*, *Témata* a *Fakta*. Všem modulům bude navíc přizpůsobena přehledová stránka, aby šly na první pohled zjistit nejpodstatnější informace o daném záznamu.

Zasazení do serverové aplikace

CMS může být součástí serverové aplikace. Kromě REST API tak bude aplikace poskytovat i webové rozhraní pro správu obsahu. REST API i CMS tak budou pracovat v některých situacích se stejnými zdrojovými soubory a potřebné úpravy lze provádět na jednom místě. Jak již bylo zmíněno v kapitole 5.1, *Twill* lze zakomponovat i do již existujícího projektu.

Pokud je entita součástí modulu, obsahuje navíc mnoho atributů, které navyšují její velikost při odesílání přes HTTP a ne vždy jsou tyto informace navíc žádoucí. Tento problém bude vyřešen za použití *API Resources* od *Laravelu*, které transformují entitu před odesláním a lze takto odstranit atributy a vztahy, které nejsou výukovou aplikací vyžadovány.

Ve struktuře projektu vždy *Twill* vytváří oddělené adresáře, kde jsou definovány zdrojové soubory pro CMS. Například pro soubory typu blade k definování formulářů modulu *Otázky* bude cesta `resources/views/twill/questions`. Dále jsou odděleny kontroléry v `app/Http/Controllers/Twill`, jejichž odpovědnostní je správa a obsluha modulů v CMS a nesouvisí s kontroléry, které zpracovávají požadavky na API.

5.5 Implementace

S implementací bylo postupováno průběžně s mobilní aplikací aby splňovala aktuální požadavky a mohly být testovány funkce v celém svém pojetí. K tomu byla i postupně uzpůsobována databáze.

Serverová aplikace byla implementována v jazyce Php 8.1 za použití frameworku Laravel verze 9. Jako vývojové prostředí bylo použito PhpStorm IDE, který nabízí podporu známých frameworků, podporu frontendových technologií (JavaScript, HTML, CSS, Emmet...) nástroje na code-completion a refaktor kódu, vestavěný systém pro správu verzí kódu a další užitečné nástroje.

Struktura projektu

Po instalaci *Laravel* projektu pomocí Composeru je vytvořena výchozí struktura projektu, která je rozdělená do logických celků.

Základní adresáře a jejich obsah jsou:

- app – jádro aplikace, obsahuje téměř všechny třídy aplikace
- bootstrap – důležitý je app.php soubor, který spouští framework
- config – všechny konfigurační soubory aplikace
- database – databázové migrace, továrničky na vytváření instancí entit a seedové soubory nejčastěji používané pro počáteční vložení dat do databáze
- public – index.php soubor, který je vstupním bodem pro všechny příchozí požadavky na aplikaci.
- resources – šablony a nezkompilované assety jako CSS nebo JavaScript

- routes – soubory obsahující definice pro routování
- storage – cache, kompilované Blade šablony, logy atd.
- tests – automatizované testy, v základu dělené na testy jednotkové a funkční
- vendor – všechny závislosti Composeru

V kořenovém adresáři projektu se nachází další soubory, jako například .gitignore pro práci s verzovacím systémem Git, soubory pro definování požadovaných závislostí a balíčků (package.json nebo composer.json) aj. Dále se zde nachází .env obsahující mnoho běžných proměnných prostředí. Tento konkrétní soubor by se neměl sdílet, protože může obsahovat citlivé informace (např. heslo k databázi, přístupy k e-mailovému klientovi atd.).

Zpracování požadavků

Zpracování požadavků přicházející do aplikace probíhá tak, že požadavku je dle URI nalezena příslušná cesta (z angl. route) v souboru `routes/api.php` a je k němu spuštěna příslušná akce daného kontroléru. Příklad nadefinované cesty lze vidět na obr. 5.3

```
Route::get( uri: 'quiz/init', [\App\Http\Controllers\QuizController::class, 'init']);
```

Obrázek 5.3: Definovaná cesta pro inicializaci kvízu. Při čteni zleva doprava lze vidět, že se jedná o cestu pro požadavek typu *GET* s URI `quiz/init` a bude jej obsluhovat kontrolér `QuizController` a jeho funkce *init*.

Tato cesta slouží pro vytvoření nového kvízu pro uživatele, který požadavek zasílá. Po shodě URI se spustí funkce init ve třídě QuizController. Této funkci je *Laravelem* předán parametr ve formě objektu typu Request, který v sobě, krom jiného, obsahuje GET parametry nebo tělo POST požadavku. V tomto případě se jedná o GET parametr reprezentující identifikátory témat, které si uživatel vybral.

Po spuštění příslušné funkce jsou prováděny další operace. V rámci rozdělení závislostí jsou použity pomocné třídy končící označením *Service* (například *QuizService*) – v těchto třídách probíhá složitější logika na pozadí, zatímco kontroléry se starají o zpracování požadavku a odeslání HTTP odpovědi s kódem a tělem odpovědi v závislosti na výsledku operací.

Autentizace uživatele

Registrace, přihlášení a odhlášení uživatele probíhá v kontroléru s názvem *AuthController*. Každá tato akce má svou routu (`/register`, `/login` a `/logout`), která náleží k příslušné akci. Akce Login očekává token, který je uživateli vygenerován po registraci a odeslán v odpovědi, aby jej mohl používat při komunikaci se serverem v HTTP hlavičce *Authorization*. Odhlášením se zneplatní aktuálně používaný token uživatele.

Autentifikace se spouští jen pro cesty, které mají zaregistrovaný tzv. middleware – mechanismus pro filtrování a inspekci HTTP požadavků přicházející do aplikace. V tomto případě se jedná o middleware s názvem `auth:sanctum`. *Sanctum* je lehký autentizační systém pro Laravel. Po autentifikaci (tedy projití middlewarem) je požadavek puštěn dále a spouští se kontrolér s příslušnou akcí.

Migrace a databáze

Schéma databáze v projektu je ovládáno s použitím migrací. Ty si lze představit jako verzovací systém pro databázi. Pokud je třeba upravit schéma databáze, jsou vytvořeny nové migrace, které v sobě definují pouze konkrétní změnu a zpravidla také jak tuto změnu vzít zpět pokud dojde k situaci, kdy je nutno se vrátit o jednu či více verzí. Není tedy potřeba manuálně přidávat a odebírat sloupce či jinak modifikovat schéma databáze, stačí spustit migrace. Toto zjednodušuje práci v týmu i mimo něj, protože často se vývojové prostředí liší od produkčního a tam je nutno změny promítnout do databáze také. Záznamy o již spuštěných migracích jsou uloženy v databází.

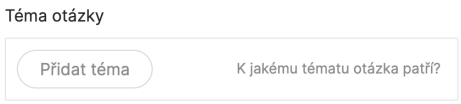
Tento projekt zahrnuje převážně migrace inicializující celé tabulky (*questions*, *topics*, *users*, *quiz_metadatas*...) a pár takových, které dodatečně tabulky upravují.

Administrace pro plnění obsahu

Pro plnění obsahu byla vytvořena administrace pomocí *Twillu*. Byly vytvořeny moduly na otázky, témata a fakta. Přehledová stránka každého modulu je přizpůsobena za použití přidělených kontrolérů, jejichž akce spouští routovací systém *Twillu*. CMS je přístupné na URI `/admin`.

Šablony definující formuláře se nachází v podadresářích v `resources/views/twill` dle jména modulu. Ve formulářích jsou použity speciální komponenty – příkladem nechť je komponenta *browser* (obrázek 5.4, která po přiděluje jednu či více entit (podle toho jak je definována kardinalita vztahu entit) k upravované entitě. Po kliknutí na „Přidat téma“ se otevře okno se seznamem entit, které mohou být přiděleny (v tomto případě se jedná o entitu *Topic* a uživatel tímto svazuje otázku s tématem). Zde vhodno doplnit, že *Twill* automaticky zpracovává většinu formulářových prvků (*checkbox*, *text*, *select*, *radio* aj.) sám a atributy přiděluje podle názvů (name parametr v komponentě), ale např. prvky vytvářející vztahy mezi entitami je třeba dospecifikovat v dané *repository* třídě, v tomto případě `QuestionRepository`.

```
<x-twill::browser
    module-name="topics"
    name="topic"
    label="Téma otázky"
    :max="1"
    itemLabel="téma"
    note="K jakému tématu otázka patří?"
/>
```



Obrázek 5.4: Vlevo je použitá komponenta tybu *browser*. Kód říká, že komponenta otevírá prohlížeč pro modul *topics*, označení je „Téma otázky“, lze přidělit maximálně 1 záznam a popisek pro uživatele je „K jakému tématu otázka patří?“.

Kapitola 6

Testování

Po návrhu a implementaci všech částí aplikace nastává další podstatná fáze, kterou je testování. Díky tomu lze ověřit, zda jednotlivé části fungují jak mají a zda řešení i funguje jako celek, jak po stránce technické, tak po stránce splnění cílů a požadavků.

Aby bylo možné zjistit, do jaké míry je aplikace použitelná, intuitivní, spolehlivá a výkonná, jsou prováděny experimenty a testování. Kromě běžného průběžného testování při vývoji je vhodné oslovit nezaujaté lidi, uživatele, aby aplikaci vyzkoušeli a například v ní provedli nějaké úkoly. Různé experimenty získávají různé výsledky – mezi některé patří UX, efektivita řešení atd.

Součástí této práce bylo provedení testů se 2 absolventy uměleckých oborů s maturitou a 5 studenty aktuálně se připravující na maturitní zkoušku z DVK. Testy byly provedeny formou focus-group a 1 na 1 (online i naživo).

6.1 Zpětná vazba od studentů

Zpětná vazba od uživatelů je cenným zdrojem informací a inspirace. Je potřeba myslet na to, že uživatelé nemohou posuzovat všechno, protože mnoho vjemů vnímají podvědomě. Zároveň ne každý nápad a kritika od uživatele je účelná pro cíle aplikace. Testování se zaměří na UX a specificky na tyto otázky:

- Jak často uživatel neví, co se stane?
- Jak často se uživatel ztrácí?
- Které aktivity ho bavily?
- Jaké interakce mu přišly otravné?

Vzhled

Studentům se v aplikaci orientovalo dobře, líbil se jim nápad s delfínem, ale obecně na ně působilo GUI mírně nezajímavě a neinteraktivně. Před dalším vývojem bude tedy vhodné promyslet a upravit barevné schéma, font, přidat interaktivní prvky, a tím aplikaci oživit do modernějšího a hravějšího stylu.

Dále by ocenili, kdyby obrazovka s dodatečnými informacemi měla formátovaný text pro zvýraznění klíčových pojmů a obecné zpřehlednění informací.

Interakce

V rámci interakce s uživatelem se vyskytlo několik připomínek, kdy uživatele mátly některé prvky v aplikaci.

- Zamčené možnosti jsou ve výchozím stavu bledé a nevýrazné, což někteří uživatelé nejprve nepochopili, předpokládali, že je to pouze alternativní (a jinak zbarvená) možnost.
- Závažnější a důležitější připomínka je, že aplikace působí staticky a trochu "bez života". Jelikož jsou studenti středních škol mladší lidé, používají aplikace, které více interagují s uživatelem například za pomocí různých efektů, častých odměn apod. Edukační aplikace by měla disponovat komfortním a zábavným prostředím, aby se uživatel v aplikaci udržel.
- Uživatelé se dále pokoušeli klikat na prvky, u kterých na první pohled není zřejmé, co dělají. Jednalo se např. o ikonku profilu uživatele nebo o ukazatel zápisků.
- Při načítání další otázky se zobrazuje náhodný a jednoduchý fakt, který uživatelé někdy nestíhali přečíst, když se další otázka načetla příliš rychle.
- Uživatel nevidí, v jaké části kvízu se nachází a neví, kolik ho tedy ještě čeká otázek, což může být frustrující, pokud má uživatel omezené množství času na používání aplikace.
- V návaznosti na předchozí bod ještě zmínili, že nelze odejít z kvízu v průběhu, kdy uživatel opět může být mírně frustrován že neví, co se stane když aplikaci celou ukončí.

Obtížnost

V rámci testování proběhla i konzultace s absolventy, kteří měli maturitu z předmětu DVK. Jejich hlavní připomínka byla, že různé střední školy mají rozdílně nastavou hranici obtížnosti maturitní zkoušky a jsou tedy kladeny jiné nároky na hloubku znalostí témat v předmětu. Pokud by aplikace měla přizpůsobitelnou obtížnost, otevřela by se také více lidem z jiných vzdělávacích institucí (vysoké či vyšší odborné školy) a i mimo ně (lidé ze široké veřejnosti zajímající se o téma)

Řešením by mohlo být, že každá otázka bude označena nějakou obtížností při jejím zadávání do systému (např. 5 stupňů obtížnosti) a uživatel by si ve svém profilu zvolil preferenci obtížnosti. Dle této preference by se mu mohly zobrazovat otázky, které odpovídají preferenci, ale zároveň by se mu sem tam zobrazila i těžší či lehčí otázka, aby uživatel cítil, že při těžší otázce se má pořád kam posouvat a učit se. Lehčí otázka by naopak mohla uživateli navodit dobrý pocit, že již něco umí. Nevýhoda toho řešení je množství otázek, které by muselo vzniknout, aby každý stupeň obtížnosti měl dostatečné množství obsahu.

Jiným, lepším řešením je implementování nějaké metody strojového učení, která by analyzovala postup a znalosti uživatele a dle toho mu předkládala otázky dle obtížnosti. Stále nemizí problém s potřebou mnoho obsahu, nicméně uživateli by se adaptivně zvyšovala obtížnost v závislosti na vědomostech, což jej může motivovat a udržet v běhu. Nevýhodou je složitější realizace řešení a měření úspěšnosti, zda algoritmus nabízí studentům otázky s obtížností odpovídající jejich znalostem v kombinaci a občasně vkládá otázky z jiného stupně obtížnosti.

6.2 Provedené úpravy

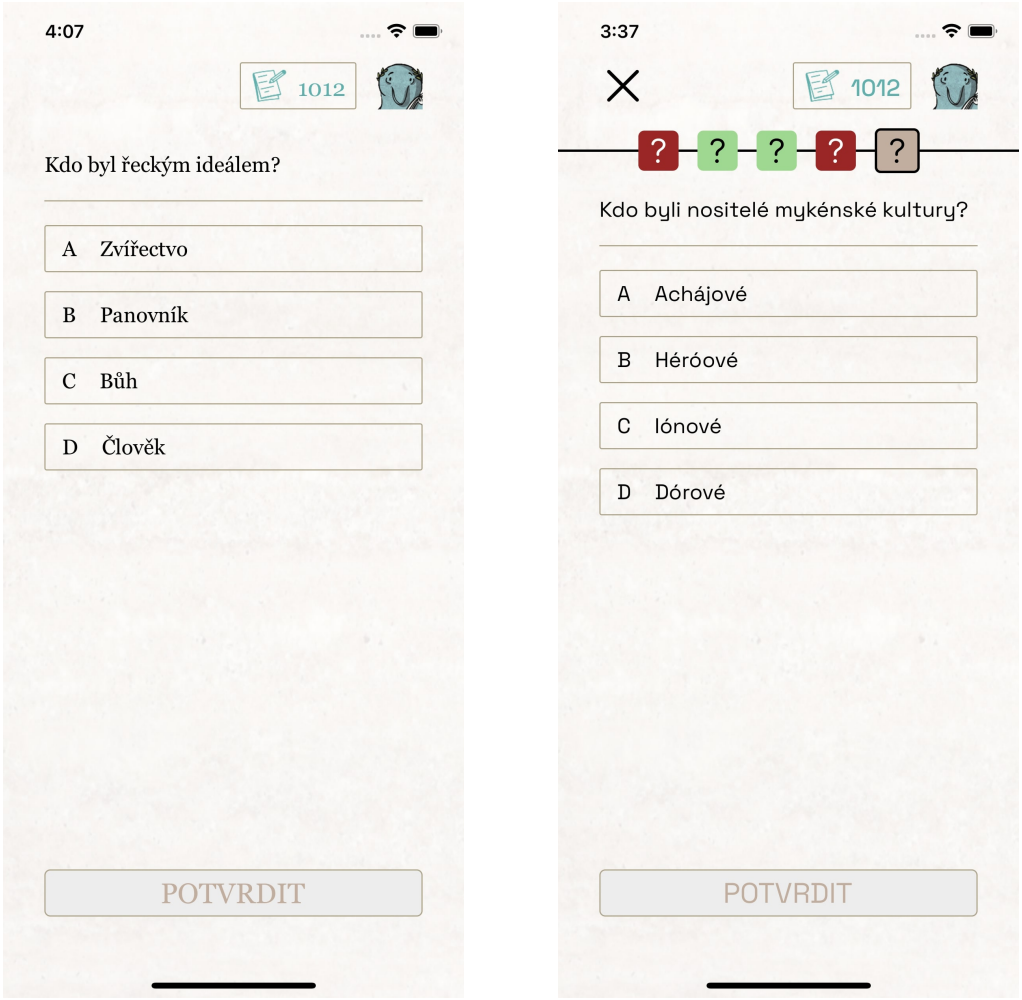
V návaznosti na zmíněné připomínky proběhly úpravy aplikace a vyřešily se některé problémy. Aktuálně provedená řešení na problémy:

- Zlepšena optimalizace GUI prvků tak, aby pro různá rozlišení mobilních zařízení nepřesahovaly své hranice, případně nebyly příliš malé.
- Font Georgia (pro iOS a výchozí patkový font pro Android) byl nahrazen modernějším, mírně hravým fontem Space Grotesk.
- Pro rozeznání ve které části kvízu se nachází byl přidán ukazatel postupu do vrchní části obrazovky. Lze nyní vidět na kolikáté je aktuálně otázce a výsledek odpovědí na předchozí otázky (obr. 6.1).
- Pro odejití z kvízu v průběhu bylo přidáno tlačítko s ikonkou křížku v levém horním rohu (obr. 6.1). Po odejití se uživateli v systému připíší zodpovězené otázky k jeho profilu, nezodpovězené jsou patřičně označeny. Odměnu dostane pouze za zodpovězené otázky.
- Zamčené možnosti v aplikaci (zatím nedostupné funkce - témata s nedostatkem obsahu, nastavení profilu atd.) byly navíc označeny ikonkou zámečku, aby uživatelé snáze pochopili, že na prvek nelze kliknout (obr. 6.2).
- Na prvky, na které uživatelé klikali a nevěděli proč nic nedělají, byl přidán popisek, který se po kliknutí na prvek ukáže (tooltip) (obr. 6.2).
- Po načtení další otázky se v načítací obrazovce zobrazí animovaný blikající text a aplikace čeká na kliknutí uživatele pro pokračování. Nabízelo se i řešení přednastaveného časové limitu, po který bude obrazovka zobrazena, nicméně při různých délkách faktů se toto řešení ukázalo nevhodné a navíc uživatel mohl již fakta znát a chtěl co nejdříve pokračovat dále.

6.3 Známé či přetrvávající problémy

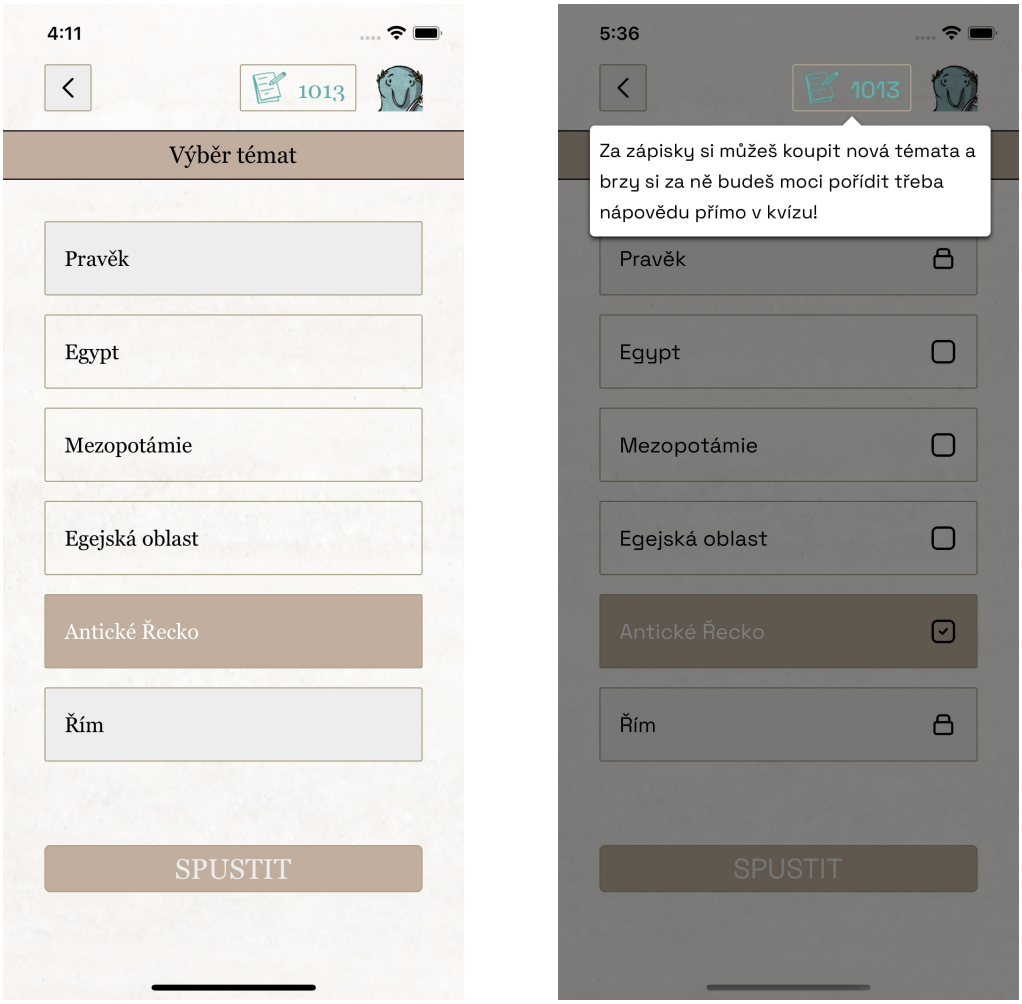
V průběhu práce byly odhaleny různé problémy, které se promítají do výsledku práce. Tyto problémy pro rozsah práce nebyly vyřešeny a lze je brát jako výchozí body pro pokračování ve vývoji v návaznosti na tuto práci.

- Primární problém je nedostatek obsahu vloženého do aplikace. Základní stavební kámen edukační aplikace je obsah, který bude uživatelům předáván a pokud je ho nedostatek, aplikace ztrácí na účinnosti . V rámci spolupráce se střední školou se zatím nepodařilo dohodnout na podmínkách spolupráce s plněním obsahu a aplikace je tedy ve fázi, kdy funguje a lze jí používat, nicméně chybí jí náplň.
- Jeden z cílů aplikace byl, aby si uživatelé spustili aplikaci sami a sem tam si spustili kvíz. Tento cíl nebyl testován. Přestože byl test naplánován a uživatelé byli ochotni spolupracovat, kvůli zmíněnému nedostatku obsahu by uživatele brzy opakování informací omrzelo a aplikace zatím nenabízí jiné funkce, které by uživatele přiměly v ní trávit čas.



Obrázek 6.1: Vlevo průběh kvízu před úpravami, vpravo po provedení úprav. Hlavním rozdílem je přítomnost ukazatele postupu a křížku v pravém horním rohu pro opuštění procvičování. Dále si lze všimnout změny fontu na hravější Space Grotesk.

- Přizpůsobující se obtížnost také nebyla vyřešena pro nedostatek různě obtížného obsahu.
- Aplikaci zůstal původní vzhled přestože jej uživatelé vnímají jako konzervativní a málo zajímavý. Důvodem je využití času k zaměření se na ladění již aktuálního řešení GUI.



Obrázek 6.2: Vlevo výběr témat před zpětnou vazbou studentů, vpravo se změnami. Hlavními rozdíly jsou přidané ikonky označující zamčená (zámeček), odemčená (prázdné políčko) a vybraná témata (zaškrtnuté políčko). Dále lze na obrázku vpravo vidět dialog pro vysvětlení prvku po kliknutí na něj.

Kapitola 7

Závěr

Cílem práce bylo navržení a vytvoření nástroje na procvičování znalostí k maturitní zkoušce. Po počátečním průzkumu mezi studenty a následném průzkumu trhu bylo řešení rozděleno na výukovou aplikaci pro uživatele, serverovou aplikaci na zpracování HTTP požadavků mobilních klientů a webové rozhraní pro tvůrce obsahu.

V průběhu práce bylo dbáno na principy UX a zvoleny moderní nástroje na tvorbu webových a mobilních aplikací. Na začátku byl proveden průzkum studentů i trhu, bylo navrženo GUI a implementován systém na tvorbu obsahu. Následně byly vytvořeny aplikace, otestovány a vyhodnoceny.

Před návrhy a realizací každé části aplikace byla důkladně nastudována relevantní teorie, na jejíž zdroje je v textu odkazováno. Tato teorie sloužila jako základ pro návrhy a další práce na tomto řešení. Výsledkem realizací serverové a výukové aplikace jsou funkční koncepty navrhovaných aplikací. Hlavním důvodem je podceněná náročnost realizace mobilní aplikace, primárně ve vztahu s laděním optimalizace GUI pro zařízení různých velikostí a porozumění vykreslování komponent při změně stavu nebo vstupních parametrů. Tím byly vynechány ty zajímavější a více motivující funkce pro uživatele (nápovědy, denní mise, statistiky).

Nicméně aplikace je ve funkčním stavu, lze si spustit kvíz s vybranými tématy, odpovídat na otázky více typů a získat zpětnou vazbu po zodpovězení. Při načítání další otázky se zobrazují fakta, čímž uživatel může nabýt ještě o něco více vědomostí. Na pozadí jsou odesílány HTTP požadavky na serverovou aplikaci, která běží na veřejném serveru. Mobilní aplikace funguje na operačních systémech Android i iOS a je dostupná skrze Expo GO.

Studenti při testování přišli s kritikou, ale i mnoha nápady na vylepšení a ocenili jak nápad aplikace, tak i stav aktuálního zpracování a těší se na další vývoj a testování. Zároveň se studenti v aplikaci pohybovali bez obtíží, neztráceli se a celkově pro ně byla interakce s aplikací pohodlným uživatelským zážitkem.

Samostatná kapitola je obsah, který se v rámci spolupráce se střední školou nepovedlo zatím dostat do stavu, aby aplikace mohla být využívána s plným potenciálem. Dle pedagoga je snadné vymyslet otázku, ale těžší vymyslet špatné odpověď tak, aby nebyla odpověď příliš jasná nebo naopak příliš zavádějící a obtížná.

Po zvážení proměnných jako aktuální stav aplikace, zájem studentů a aktuální nedostatek obsahu, jako autor práce usuzuji, že má smysl na aplikaci pokračovat a zdokonalovat. Zároveň je třeba se více zaměřit na obsah, komunikaci s potenciálními tvůrci, zjednodušení rozhraní pro správu obsahu a alternativní možnosti jeho získávání, kdy se nabízí přemýšlet nad začleněním umělé inteligence.

Literatura

[1] *Platform Architecture*. [cit. 2023-04-25]. Dostupné z: https://developer.android.com/guide/platform.

[2] B. Pajak, C. F. a A. Kittredge a H. Wilson a. *The Duolingo Method for App-based Teaching and Learning*. Pittsburgh, Pennsylvania, United States: Duolingo, Inc., January 2023 [cit. 2023-03-05]. 19 s. Dostupné z: https://duolingo-papers.s3.amazonaws.com/reports/duolingo-method-whitepaper.pdf.

[3] Churchville, F. *What is User Interface (UI)? definition from searchapparchitecture*. TechTarget, Sep 2021 [cit. 2023-05-02]. Dostupné z: https://www.techtarget.com/searchapparchitecture/definition/user-interface-UI.

[4] Duolingo. May 2022 [cit. 2023-04-25]. Dostupné z: https://investors.duolingo.com/news-releases/news-release-details/duolingo-announces-record-bookings-first-quarter-2022-and-raises.

[5] Griffith, C. *What is hybrid mobile app development: Hybrid vs native vs web*. Ionic, Sep 2022. Dostupné z: https://ionic.io/resources/articles/what-is-hybrid-app-development#h-what-is-a-hybrid-mobile-app.

[6] Hat, R. May 2020 [cit. 2023-04-18]. Dostupné z: https://www.redhat.com/en/topics/api/what-is-a-rest-api.

[7] Kinsta. *The Laravel PHP Framework – Web App Construction for Everyone*. Feb 2023 [cit. 2023-04-20]. Dostupné z: https://kinsta.com/knowledgebase/what-is-laravel/.

[8] Marsh, J. *UX for Beginners*. O'Reilly Media, Inc, 2015. ISBN 978-1-491-91268-3.

[9] Miksa, M. *Pouze 1 % obyvatel ČR stále nemá vlastní mobil. Smartphony už si oblíbili i senioři* [online]. [cit. 2023-04-04]. Dostupné z: https://mobilmania.zive.cz/clanky/pouze-1--obyvatel-cr-stale-nema-vlastni-mobil-smartphony-uz-si-oblibili-i-seniori/sc-3-a-1353422/default.aspx.

[10] MŠMT. *Rámcový vzdělávací program pro obor vzdělání 82 – 41 – M/05 Grafický design*. Praha: MŠMT [cit. 2023-05-04]. 50 s. Dostupné z: https://www.edu.cz/wp-content/uploads/2020/08/82-41-M05_Graficky_design_2020_zari.pdf.

[11] Oracle. *What Is a Database?* [cit. 2023-04-14]. Dostupné z: https://www.oracle.com/database/what-is-database/.

[12] Team, R. S. *IOS architecture*. Feb 2023. Dostupné z: https://redfoxsec.com/blog/ios-architecture/.

[13] User, S. *Maturitní Zkouška*. Dostupné z: https://maturita.cermat.cz/menu/maturitni-zkouska.

Příloha A

Obsah přiloženého paměťového média

- `mobile_app_src` – složka obsahující zdrojové kódy se závislostmi pro mobilní aplikaci
- `server_app_src` – složka obsahující zdrojové kódy se závislostmi pro serverovou aplikaci
- `text` – složka obshající text práce ve formátu PDF a adresář se zdrojovými soubory pro LaTeX
- `readme.md` – návod k instalaci řešení
- `video.mp4` – video prezentující realizovanou mobilní aplikaci

Příloha B

Plakát

Obrázek B.1: Plakát aplikace Maturita Quiz.