Mendelova univerzita v Brně
Provozně ekonomická fakulta

Aplikace pro online testování žáků na základních a středních školách

Diplomová práce

Vedoucí práce:
prof. Ing. Radim Farana, CSc., FEng.

Bc. Lukáš Prokop

Brno 2021

Poděkování

Děkuji vedoucímu mé diplomové práce prof. Ing. Radimu Faranovi, CSc., FEng. za odborné vedení, podnětné rady, ochotu a čas, který mi věnoval. Děkuji také Mgr. et Mgr. Michalu Šimáněmu, Ph.D. za doporučení literatury i rady v oblasti vzdělávání. Mé poděkování patří také mé rodině a všem respondentům, jejichž připomínky mi pomohly práci dokončit.

Čestné prohlášení

Prohlašuji, že jsem práci **Aplikace pro online testování žáků na základních a středních školách** vypracoval samostatně a veškeré použité prameny a informace uvádím v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a v souladu s platnou Směrnicí o zveřejňování závěrečných prací.

Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 7. května 2021 .....

podpis

Abstract

Prokop, L. Application for Online Testing of Students in Primary and Secondary Schools. Diploma thesis. Brno, 2021.

This diploma thesis deals with design and implementation of web application for online testing of students in primary and secondary schools using modern approaches and technologies for development. The web application consists of an interface for teachers and an interface for students. The application allows teachers to create new tests, modify existing tests, display submitted tests and evaluate them. Students can run tests, complete them, submit them and display their results after the evaluation by the teacher. At the beginning of the thesis, the issue of education in the didactic system and the issue of a modern approach to the development of web applications are described. Furthermore, an analysis of application requirements and evaluation of available solutions were performed. Based on the obtained information, the web application was designed and subsequently implemented, the functionality was verified and the achieved results were assessed. In conclusion, the possibilities of further continuation are discussed.

Key words: modern development approaches, web application, modern education, online testing, single-page application

Abstrakt

Prokop, L. Aplikace pro online testování žáků na základních a středních školách. Diplomová práce. Brno, 2021.

Tato diplomová práce se zabývá návrhem a implementací webové aplikace pro online testování žáků na základních a středních školách s využitím moderních přístupů a technologií k vývoji. Webová aplikace se skládá z rozhraní určeného pro vyučující a z rozhraní určeného pro žáky. Vyučujícím aplikace umožňuje vytvářet nové testy, upravovat stávající testy, zobrazovat odevzdané testy a vyhodnocovat je. Žáci mohou testy spustit, vyplnit, odevzdat a po vyhodnocení vyučujícím zobrazit svůj výsledek. Na začátku práce je popsána problematika vzdělávání v didaktickém systému a problematika moderního přístupu k vývoji webových aplikací. Dále byla provedena analýza požadavků na aplikaci a zhodnocení dostupných řešení. Na základě získaných informací byla navržena a následně implementována webová aplikace, ověřena funkcionalita a zhodnoceny dosažené výsledky. V závěru práce jsou diskutovány možnosti dalšího pokračování.

Klíčová slova: moderní přístupy k vývoji, webová aplikace, moderní vzdělávání, online testování, jednostránková aplikace

Obsah

1 Úvod **8**
1.1 Cíl diplomové práce . . . 8

2 Vzdělání v didaktickém systému **9**
2.1 Didaktické prostředky v procesu vyučování . . . 9
2.1.1 Moderní učební pomůcky . . . 9
2.2 Hodnocení žáků a prověřování jejich znalostí . . . 10
2.2.1 Metody prověřování znalostí žáků . . . 11
2.3 Využívání digitálních zařízení ve výuce . . . 12
2.3.1 Pozitiva využívání digitálních zařízení . . . 12
2.3.2 Negativa využívání digitálních zařízení . . . 12
2.3.3 Aktuální situace . . . 13
2.4 Koncept bezpapírové školy . . . 14
2.5 Koronavirus jako revoluce ve školství . . . 15

3 Aplikační software **17**
3.1 Charakteristika . . . 17
3.2 Rozdělení aplikací . . . 17
3.2.1 Desktopové aplikace . . . 17
3.2.2 Webové aplikace . . . 19
3.2.3 Mobilní aplikace . . . 20
3.2.4 Cloudové aplikace . . . 21
3.3 Jak si správně vybrat? . . . 22

4 Moderní přístupy k vývoji webových aplikací **23**
4.1 Počátky a okolnosti vzniku webových aplikací . . . 23
4.2 Současné trendy ve vývoji webových aplikací . . . 23
4.2.1 Progresivní webové aplikace . . . 24
4.2.2 Boti a umělá inteligence . . . 25
4.2.3 AMP . . . 26
4.2.4 Motion design v uživatelském rozhraní . . . 27
4.2.5 Tmavé téma . . . 27
4.2.6 Push notifikace . . . 28
4.2.7 A11Y . . . 29
4.2.8 Další současné trendy . . . 30
4.3 Single-page aplikace . . . 30
4.3.1 Tradiční multi-page vs. moderní single-page aplikace . . . 31
4.3.2 Výhody single-page aplikací oproti tradičním aplikacím . . . 32
4.4 Architektura a návrhové vzory . . . 33
4.4.1 MVC . . . 33
4.4.2 MVP . . . 34
4.4.3 MVVM . . . 35

4.4.4 Moderní přístup k návrhovým vzorům v souvislosti s webovými aplikacemi . . . 36
4.5 Technologie využívané pro frontend . . . 37
4.5.1 React . . . 39
4.5.2 Angular . . . 40
4.5.3 Vue.js . . . 41
4.6 Technologie využívané pro backend . . . 42
4.6.1 Express . . . 42
4.6.2 Django . . . 42
4.6.3 Laravel . . . 43
4.6.4 Rails . . . 43
4.6.5 Spring . . . 43
4.7 Databázové systémy . . . 43
4.7.1 Základní typy databází . . . 44
4.7.2 Rozhodující faktory při výběru databáze . . . 45
4.7.3 Přehled konkrétních databází . . . 46

5 Analýza dostupných řešení 47
5.1 Kahoot . . . 47
5.1.1 Připomínky vyučujících k aplikaci Kahoot . . . 48
5.2 Socrative . . . 48
5.2.1 Připomínky vyučujících k aplikaci Socrative . . . 49
5.3 Microsoft Forms . . . 49
5.3.1 Připomínky vyučujících k aplikaci Microsoft Forms . . . 50
5.4 Google Forms . . . 50
5.4.1 Připomínky vyučujících k aplikaci Google Forms . . . 51

6 Analýza a specifikace požadavků 52
6.1 Use case diagram . . . 52
6.2 Use case specifikace . . . 53
6.2.1 Zobrazit přehled vytvořených testů . . . 53
6.2.2 Vytvořit nový test . . . 53
6.2.3 Upravit vytvořený test . . . 53
6.2.4 Smazat vytvořený test . . . 54
6.2.5 Zobrazit vyplněné testy od žáků . . . 55
6.2.6 Opravit vyplněný test . . . 55
6.2.7 Přidat novou otázku . . . 56
6.2.8 Upravit otázku . . . 56
6.2.9 Smazat otázku . . . 57
6.2.10 Nastavit parametry testu . . . 57
6.2.11 Zobrazit přehled vyplněných testů . . . 58
6.2.12 Spustit test . . . 58
6.2.13 Odevzdat test . . . 59

6.2.14 Zobrazit vyhodnocený test 60
6.2.15 Zaregistrovat se 60
6.2.16 Přihlásit se 61
6.2.17 Obnovit zapomenuté heslo 61
6.3 Funkční požadavky a stěžejní body pro návrh aplikace 62

7 Návrh aplikace a její implementace 63
7.1 Návrh struktury databáze 63
7.2 Volba technologií 65
7.3 Princip komunikace mezi frontendem, backendem a databází 66
7.3.1 Princip komunikace v rámci vyvíjené aplikace 66
7.4 Implementační detaily 67
7.4.1 Autentizace 67
7.4.2 Autorizace 70
7.4.3 Frontend 71
7.4.4 Optimalizace datových přenosů 72

8 Dosažené výsledky 73
8.1 Vytváření a úprava testu 73
8.2 Spuštění a vyplňování testu 78
8.3 Vyhodnocení testu 79
8.4 Přehled testů 82

9 Ověření funkcionality 83
9.1 Ověřování během vývoje 83
9.2 Uživatelské ověření výsledného produktu 84

10 Závěr 86

11 Literatura 88

1 Úvod

Žijeme v dynamické době, jejíž součástí je digitální revoluce, čtvrtá průmyslová revoluce a rychlý rozmach technologií. Nejen od lidí jako samostatných celků, ale i od firem je očekáváno, že na tuto novou éru budou reagovat. Kamenné obchody přesouvají sortiment na e-shopy, lidé si své finance řídí prostřednictvím internetového bankovnictví, strávníci vyměňují návštěvy restaurací za online objednávky prostřednictvím dovozových služeb, nabídky a poptávky se z novin a nástěnek přesouvají na inzertní portály, dopisní korespondence je nahrazována e-mailem, chatem a videohovory, a to není ani zdaleka vše. Ačkoliv se jedná o zcela rozdílné oblasti, jedno mají společné – ve všech je potřeba nějaký software, tedy počítačový program provádějící určitou činnost. A právě software je jedna z věcí, které patří ke stěžejním částem procesu tzv. digitální transformace – přechodu na digitální technologie.

Zatímco mnoho oblastí s procesem digitalizace nemá problém a těší se v zásadě velkým úspěchům, samostatnou kapitolou jsou základní a střední školy. K rozvoji z pohledu digitalizace v nich sice dochází – zavádí se elektronické žákovské knížky, některé školy využívají informační systémy, někteří vyučující poskytují žákům své digitální materiály k výuce, avšak jakmile přijde řada na zkoušení, rozdíly mezi způsobem ověřování znalostí u současné generace a generací předchozích nejsou příliš velké. Každý z nás za svůj život mnohokrát slyšel před začátkem zkoušky stále se opakující větu vyučujícího vyzývající ke schování všech věcí a ponechání pouze pera a papíru na lavici. Co kdyby ale v této větě slova *pero* a *papír* nahradil například mobilní telefon, počítač, notebook či tablet? Co kdyby se změnil způsob zkoušení tak, aby žákům dovolil pracovat s moderními technologiemi a vyučujícím ušetřil mnoho času a sil při přípravě testů i jejich následném opravování?

1.1 Cíl diplomové práce

Cílem mé diplomové práce je navrhnout a následně implementovat aplikaci pro online testování žáků na základních a středních školách s využitím moderních přístupů a technologií k vývoji. Mojí snahou je, aby aplikace přinesla žákům možnost využít moderní technologie při výuce, poskytla jim zajímavější a interaktivnější zkoušky a umožnila jim mít dosažené výsledky stále na dosah ruky. Pro vyučující bude aplikace znamenat rychlejší přípravu testů a přístup k opravování i vytváření testů kdykoliv a odkudkoliv bez nutnosti jejich fyzického přenášení.

Teoretická část práce se bude věnovat problematice vzdělávání v didaktickém systému a využívání digitálních technologií ve vzdělávání, aplikačnímu softwaru a především moderním přístupům k vývoji webových aplikací. V rámci praktické části budou analyzovány požadavky na danou aplikaci, vytvořen návrh řešení a provedena implementace řešení. Závěrem budou zhodnoceny dosažené výsledky a diskutovány možnosti dalšího rozšíření projektu.

2 Vzdělání v didaktickém systému

Vzdělání je proces, ve kterém jedinec rozvíjí svoji individualitu, své zkušenosti a logické myšlení. Zároveň proniká do mezilidských vztahů a nabývá schopnosti orientovat se v množství uvažovaných pohledů. Hlavním cílem vzdělávání je úspěšné začlenění se do kulturního a společenského života, ale i do pracovního procesu. (Skalková, 2007)

Sekce v této kapitole se věnují didaktickým prostředkům používaným v procesu vyučování, hodnocení žáků, metodám prověřování jejich znalostí, využívání digitálních zařízení ve výuce a konceptu bezpapírové školy, který má velmi blízkou souvislost s tématem této diplomové práce.

2.1 Didaktické prostředky v procesu vyučování

Pod pojmem didaktické prostředky se skrývají všechny předměty, které úzce souvisí s konkrétní vyučovací metodou. Jejich cílem je zkvalitňovat průběh vyučovacího procesu, pomáhat k osvojení dovedností i vědomostí a napomáhat k snazšímu procesu učení žáků. Patří mezi ně například vybavení tříd a škol, přičemž hlavní součástí jsou učební pomůcky, které jsou závislé na rozvoji techniky a kultury a významně napomáhají ve využívání principu názornosti. Konkrétními příklady učebních pomůcek jsou mimo jiné literární pomůcky v podobě učebnic či atlasů, zvukové pomůcky v podobě zvukových nahrávek či hudebních nástrojů, skutečné předměty, modely nebo projekce filmů a videí prostřednictvím dataprojektorů. (Skalková, 2007)

2.1.1 Moderní učební pomůcky

Moderní didaktické prostředky a učební pomůcky významně přispívají k inovaci ve vzdělávání, která se nejen pro zaujetí žáka zdá být nepostradatelnou. Stále více pomůcek je spojeno s digitálními technologiemi, které mají velmi široké využití taktéž v oblasti školství.

Zatímco v minulém století si pod pojmem tabule představil každý zelenou tabuli s bílou křídou, v tom současném je začaly nahrazovat bílé magnetické tabule s fixou a od roku 2006 ve velké míře také **interaktivní tabule**. Jedná se o interaktivní dotykovou plochu velkých rozměrů, na kterou je pomocí dataprojektoru vysílán obraz, který je možný pomocí této plochy ovládat. (Suchomelová, 2017)

Interaktivní tabule mají mnoho výhod. Výuka je pro žáky zajímavým zážitkem, mohou se do ní snadno a poutavě zapojovat, rozvíjejí si počítačovou gramotnost a text psaný na tabuli je možné uložit a se žáky sdílet. Nevýhody mohou souviset s nedostatečnými dovednostmi vyučujících pro provozování tabule, časově náročnější přípravou materiálů, špatnou čitelností při nevhodném osvětlení učebny či poničením žáky během přestávek a volných hodin.

Další moderní pomůckou se staly **interaktivní učebnice**, které pracují s myšlenkou propojení tradiční papírové učebnice s multimédii v podobě videa, audia

či interaktivního cvičení. V roce 2007 s takovým konceptem přišlo jako první Nakladatelství Fraus, které v současnosti nabízí více než 5500 interaktivních knih. S interaktivními učebnicemi souvisí i **obohacení tištěných knih o interaktivní modely**, které pracují na principu naskenování ikony chytrým telefonem, na kterém se zobrazí 3D model. (Suchomelová, 2017)

2.2 Hodnocení žáků a prověřování jejich znalostí

Hodnocení v kontextu vzdělávání slouží ke klasifikaci výkonů žáků a měří hloubku a šíři jejich znalostí a dovedností. Ačkoliv se bez něj učitelé nedokáží obejít, často bývá kritizováno z důvodu nespolehlivosti a nepřesnosti. Správné hodnocení by nemělo pouze klasifikovat výkon, ale poskytovat zpětnou vazbu, motivovat a inspirovat jedince. (Petty, 2008)

V širším slova smyslu uplatňuje učitel hodnocení během celého procesu vyučování. Řadíme sem například souhlasné či nesouhlasné pokynutí hlavou, pochvaly a napomenutí, různé tóny hlasu, různé pohledy či projev zájmu o osobnost žáka. (Skalková, 2007)

Mezi základní typy hodnocení patří hodnocení sociálně normované a hodnocení individuálně normované. **Hodnocení sociálně normované** poskytuje učiteli možnost použití stejného měřítka pro všechny žáky. Toho je dosaženo pomocí shodných úloh, které řeší žáci v jediném časovém okamžiku. Takový typ hodnocení staví žáka do pozice soutěžícího, který se umístí na určitou pozici žebříčku na základě porovnání s výkony ostatních žáků. **Individuálně normované hodnocení** spočívá v tom, že učitel sleduje výkon žáka a porovnává jej s předchozími výkony žáka. Zatímco v případě sociálně normovaného hodnocení může slabší žák zažívat psychickou zátěž z toho, že se vždy řadí k těm horším, u individuálně normovaného hodnocení může ve srovnání se svými předchozími výkony dosáhnout pochvaly a zažít pocit úspěchu. Ideální variantou je kombinace sociálně normovaného i individuálně normovaného hodnocení, které se vzájemně velmi dobře doplňují a v praxi se může jednat o velmi vhodný způsob hodnocení výkonu. (Kolář, Šikulová, 2009)

Dalšími typy hodnocení jsou formální a neformální hodnocení. **Formální hodnocení** přichází po předchozím upozornění žáků o jeho provádění, žáci musí mít možnost se na hodnocení připravit a požadované učivo si zopakovat. **Neformální hodnocení** vychází z pozorování výkonů žáků při běžných činnostech. V neposlední řadě se rozlišuje také průběžné a závěrečné hodnocení. **Průběžné hodnocení** je dílčí hodnocení žáka. Učitel obvykle během delšího časového období (například pololetí) získá větší množství průběžných hodnocení. **Závěrečné hodnocení** je konečné zhodnocení žáka, provádí se na konci výukového období předmětu a určuje stupeň prospěchu žáka. (Kolář, Šikulová, 2009)

2.2.1 Metody prověřování znalostí žáků

Mezi základní metody sloužící k prověřování znalostí žáků patří písemné zkoušení, ústní zkoušení a didaktické testy.

Písemné zkoušení umožňuje v relativně krátké době vyzkoušet větší počet žáků, díky čemuž je ve srovnání s ústním zkoušením považováno za časově úspornější. Forma písemného prověřování znalostí je náročná pro učitele z hlediska přípravy zadání, ale i pro žáky, neboť se nemohou opírat o případnou pomoc učitele a musí se spoléhat pouze na vlastní vědomosti. (Skalková, 2007) Písemné zkoušky mohou mít popisný nebo výpočtový charakter podle toho, zda žáci něco vysvětlují nebo zda něco matematicky či graficky řeší. Z hlediska času mohou být krátkodobé (přibližně do 10 minut) nebo dlouhodobé (i celou hodinu). (Loveček, Čadílek, 2005)

Ústní zkoušení je běžnou a velmi často používanou metodou. Spočívá v rozhovoru mezi žákem a učitelem, přičemž učitel klade otázky a žák na ně odpovídá. Je důležité, aby učitel navodil příznivou atmosféru a nesnažil se žáka uvést do stresové situace. Ústní zkoušky nevypovídají pouze o tom, jak je daný žák schopný se učit, ale poukazují také na jeho schopnosti správného vyjadřování, plynulosti řeči a soustředění. Učitel vnímá gestikulaci i mimiku žáka. U ústního zkoušení je velmi důležité, aby učitel zvládl techniku kladení otázek. Otázky musí být jednoznačné a stylisticky správné. (Skalková, 2007) Ústní zkoušení může mít několik podob. Zatímco *individuální zkoušení* spočívá ve vyvolání jednoho žáka, kterému je položena otázka a očekává se od něj odpověď, při *frontálním zkoušení* jsou ověřovány znalosti u většího počtu žáků. Učitel může během krátkého času vyzkoušet větší množství žáků a zopakovat například učivo, na které se bude navazovat. Poslední podobou je *kombinované zkoušení*, při kterém je současně zkoušeno více žáků – například jeden odpovídá na otázku a druhý počítá příklad na tabuli. Kombinované zkoušení je pro učitele nejnáročnější. (Loveček, Čadílek, 2005)

Didaktické testy jsou specifickým typem, jak prověřit znalosti žáků písemnou formou. Oproti běžnému písemnému zkoušení se vyznačují stručnou odpovědí, která má nejčastěji podobu výběrové odpovědi, kterou žák označí zatržením. Je důležité si uvědomit, že didaktické testy poskytují pro učitele významnou orientační funkci, avšak není vhodné, aby byla klasifikace žáka založena pouze na nich. (Loveček, Čadílek, 2005) Žák v nich odpovídá velmi stručně a není veden k systematickému ústnímu projevu (Skalková, 2007).

Ve které z následujících vět je pravopisná chyba?

A) Z temného lesa se ozívaly hlasité povely myslivců.
B) Podzimní dny se ohlásily nepříjemnými plískanicemi.
C) O víkendu stovky nadšených turistů obdivovaly krásy města.
D) Všechny děti se spravedlivě rozdělily o otcovy celoživotní úspory.

Obrázek 1: Ukázka otázky z didaktického testu z českého jazyka
Zdroj: https://www.statnimaturita-cestina.cz/didakticky-test/didakticky-test-2020

2.3 Využívání digitálních zařízení ve výuce

Otázka využívání digitálních zařízení – mobilních telefonů, tabletů, notebooků a počítačů – při výuce ze strany žáků se v posledních letech stala velmi populární a intenzivně se řeší nejen u nás, ale i v zahraničí. Vznikají dva odlišné tábory vyučujících, přičemž jedni se k této otázce staví velmi pozitivně a druzí velmi negativně. Nyní bych rád shrnul pozitiva a negativa, která se k tomuto tématu váží. Základní myšlenky vycházejí z článku[1] vzdělávací instituce Oxford Learning a jsou doplněny o mé vlastní poznatky.

2.3.1 Pozitiva využívání digitálních zařízení

Mezi výhody využívání digitálních zařízení žáky při výuce patří zejména:

- **Hlasovací a komunikační platformy při výuce**. Mnoho žáků se stydí vyjádřit svůj názor veřejně před ostatními. Online hlasovací a komunikační platformy nabízejí zajímavý způsob, jak získat od žáků zpětnou vazbu nebo názor, aniž by museli čelit posměchu svých spolužáků.
- **Využívání vzdělávacích a organizačních aplikací**. Studenti mohou ve výuce využívat nejrůznější vzdělávací, ale i organizační aplikace, díky kterým mohou získat lepší studijní návyky. Vteřiny hledání v elektronickém slovníku místo dlouhých minut hledání v papírovém. Také zapisování úkolů do elektronického kalendáře, který může automaticky předem upozornit na nutnost odevzdání, je pro žáka efektivnější než zapisování do běžného deníčku. Případně může úkoly zapisovat do kalendáře přímo vyučující a jen je se svými žáky sdílet.
- **Snadné dohledávání informací**. Zařízení mohou žákům pomoci dostat se snadno k dalším informacím, které je o probíraném tématu zajímají. Ne všechny informace, obzvlášť ty aktuální, jsou obsaženy v učebnicích.
- **Efektivní zapisování poznámek**. V matematice se sešit bude těžce nahrazovat, ale v teoretických předmětech může být pro mnoho žáků pohodlnější a rychlejší si poznámky zapisovat elektronicky do tabletu nebo notebooku. Ať už proto, že píší na klávesnici rychleji než rukou, nebo proto, že nemusí mít obavy z toho, že sešit ztratí a přijdou o své poznámky. Automatické sdílení poznámek na cloudovém úložišti může zabránit i tomu, že by přišli o své poznámky při ztrátě či poškození svého zařízení.

2.3.2 Negativa využívání digitálních zařízení

Mezi nevýhody využívání digitálních zařízení žáky při výuce patří zejména:

- **Vyrušování a rozptýlení**. Neustálé kontrolování sociálních sítí a textová komunikace se spolužáky vede k rozptýlení a nedostatečnému vnímání probíra-

[1] https://www.oxfordlearning.com/should-cell-phones-be-allowed-classrooms/

ného učiva. Vyučující musí žáky upozorňovat, aby odložili nebo vypnuli svá zařízení, což narušuje chod výuky.

- **Zanevření na reálný svět**. Žáci mezi sebou přestávají komunikovat, což vede ke zhoršenému utváření vztahů mezi nimi, neschopnosti spolupráce či vzájemnému naslouchání.
- **Kyberšikana**. Focení nebo natáčení svých spolužáků i přes to, že si to nepřejí a často o tom ani neví. To může snadno vést ke kyberšikaně, která je pro vyučující mnohem hůře odhalitelná než běžná forma šikany.
- **Podvádění**. Především mobilní telefony mohou vést k přístupu žáků k nepovoleným informacím během testu. Kvůli tomu také ztrácejí motivaci se učit, když vědí, že jim tento způsob prochází.

2.3.3 Aktuální situace

Mobilní telefony, ale i jiná digitální zařízení se stala běžnou součástí života většiny lidí a rok od roku se dostávají k dětem stále nižšího a nižšího věku. Po dlouhých neshodách o tom, zda je možné žákům zakázat nošení mobilního telefonu do prostředí školy se k celé situaci vyjádřila Česká školní inspekce v oficiálním stanovisku[2] v lednu 2019.

ČŠI ve svém stanovisku zmiňuje, že je možné užívání telefonů na půdě školy regulovat, avšak musí to být obsaženo ve školním řádu a v souladu s platným právním řádem. Není možné zakázat nošení mobilních telefonů a podobných zařízení do školy, protože se jedná o věci osobní potřeby a bylo by to v rozporu s právním řádem. Ředitel školy ale může specifikovat pravidla omezující používání mobilních telefonů ve škole například během vyučování, aby se předešlo riziku vyrušování a podvádění. Bez relevantních důvodů ale není možné omezit používání těchto zařízení o přestávkách či volných hodinách, protože tato doba není označována jako vzdělávací proces. Taktéž není možné žákům odebírat telefony bezdůvodně (preventivně), ale až v okamžiku, kdy žák poruší školní řád.

Na druhou stranu ČŠI v závěru svého stanoviska upozorňuje na to, aby školy braly v potaz možnosti didaktického využití mobilních telefonů ve vzdělávání.

Výrazná změna nastala v červenci 2020, kdy proběhla novelizace školského zákona[3] (zákon č. 561/2004 Sb.) a byla do něj přidána následující formulace: *„Školní řád nebo vnitřní řád může omezit nebo zakázat používání mobilních telefonů nebo jiných elektronických zařízení dětmi, žáky nebo studenty, s výjimkou jejich používání v nezbytném rozsahu ze zdravotních důvodů.“*, která umožňuje zakázat mobilní telefony nejen při vyučování, ale i během přestávek.

[2] https://skav.cz/odborna-stanoviska-skav/

[3] https://www.msmt.cz/dokumenty-3/skolsky-zakon-ve-zneni-ucinnem-od-25-8-2020

2.4 Koncept bezpapírové školy

Jak se zdá, namísto snahy o digitalizaci a rozvoj školství se situace otáčí ve snahu zabránit žákům ve využívání moderních technologií při výuce. A to v době, kdy se ve světě pojem tzv. *paperless school* stává velmi populárním.

Podle amerického výzkumu[4] se odhaduje, že průměrná škola spotřebuje denně okolo 2000 listů papíru. To by v českých podmínkách, kdy školní rok trvá 10 měsíců, vycházelo na více než půl milionu papírů za jeden školní rok na jedné škole. Pracovní listy, testy, domácí úkoly, informační letáčky, výpisy z vysvědčení, žádosti, formuláře a mnoho dalšího, to jsou konkrétní příklady, které stojí za vysokou spotřebou papíru a mají vliv nejen na životní prostředí, ale také na náklady spojené s pořizováním papíru. Je třeba vzít v potaz také náklady spojené s tonery do tiskáren a kopírek či jejich pravidelnou údržbou. (Hakes, 2020)

Podívejme se na konkrétní výhody, které přicházejí s přechodem na koncept bezpapírové školy:

- **Značná úspora financí i času**. Škola ušetří finance za papír, tonery a údržbu, zaměstnanci čas strávený kopírováním, rozdáváním a seřazováním.
- **Zvýšená efektivita vyučování**. Domácí úkoly bude škola zadávat online a žáci je budou online odevzdávat, díky čemuž budou mít všichni zúčastnění přehled o aktuálním stavu.
- **Zapálenější žáci**. Dnešní generace žáků je zvyklá pravidelně používat mobilní telefony, tablety, notebooky i počítače. Zapojení digitálních technologií do procesu vzdělávání může zvýšit zapálení žáků do výuky, obzvlášť když je využíváno prvků gamifikace.
- **Zpřístupnění multimediálních zdrojů studentům**. Videa, podcasty i kvízy pro ověření znalostí.
- **Vylepšení komunikace mezi učiteli a rodiči**. Konec poznámek v papírových žákovských knížkách, informačních letáčků a žádostí pro rodiče, které často žák ztratí. Díky digitální komunikaci mohou učitelé komunikovat přímo s rodiči bez nutnosti žáka jako prostředníka.
- **Lepší organizace aneb žádné ztracené papíry**. Předejde se nečitelnému písmu ze strany vyučujícího i ze strany studenta. Vyučující nebudou muset přenášet hromady papírů, schovávat je a dohledávat v případě zájmu studenta.

A co je hlavní, bezpapírové školy mají **mnohem snazší přesun do systému virtuálního vzdělávání** než běžné školy. (Hakes, 2020) Taková situace nastala v roce 2020 při pandemii koronaviru, kdy z prezenční na distanční (online) formu vzdělávání muselo v České republice ze dne na den přejít 1,5 milionu žáků a 170 tisíc vyučujících. A ve většině případů tento přechod značně vázl. (Perknerová, 2020)

[4] https://www.recordnations.com/2020/01/schools-spend-paper-per-year/

2.5 Koronavirus jako revoluce ve školství

Zanedlouho poté, co se 1. března 2020 na území České republiky objevily první případy osob s pozitivním nálezem COVID-19, vyhlásila vláda nouzový stav a začala zavádět mimořádná opatření. Tato opatření se dotkla mnoha oblastí, přičemž školství je jednou z těch hlavních, kde došlo k výrazným změnám. Osobní přítomnost žáků na výuce byla na základních, středních i vysokých školách zakázána a nahrazena distanční formou vzdělávání.

Přesun do virtuálního vzdělávání byl pro mnohé školy nepředstavitelný. Zatímco například ve škole v Ludgeřovicích nedaleko Ostravy, kde fungují digitálně již šestým rokem, to nebyl žádný zásadní problém, pro tři čtvrtiny žáků po celé republice se stal hlavním komunikačním prostředkem e-mail. S některými žáky dokonce nikdo ze školy nekomunikoval. (Perknerová, 2020) Postupně se podle zjištění České školní inspekce stala nejčastější komunikační platformou používanou na základních školách aplikace WhatsApp a na středních školách informační systém Bakaláři. (Doseděl, 2020)

Protože žáci ani učitelé z většiny škol nebyli na digitální výuku připraveni, rodiče se často museli ujmout role pedagogů a hlídačů. Na povrch také začaly vyplouvat otázky, zda je za této situace vhodné soustředit se na všechny předměty, když právě přechod do krizového stavu, mnohá rozvolnění a další úlevy mohly vést k významné revizi vzdělávacích plánů. Jen asi v třetině základních škol a 16 % středních škol se rozhodli v distančním vzdělávání prioritizovat určité předměty a tématické celky, ve zbytku škol postupovali přesně podle vzdělávacího plánu a předměty jako výtvarná výchova či tělesná výchova se rozhodli vyučovat distančně. (Doseděl, 2020)

Samostatnou kapitolou distančního vzdělávání je technika. Ačkoliv je 21. století a technických možností kolem nás je víc než dost, právě technika se stala hlavním problémem online vzdělávání, ať už z pohledu nedostatečných znalostí na straně učitelů, nebo z pohledu zastaralého a nedostatečného technického vybavení škol. Nalézt alternativu pro tabuli s křídou bylo pro mnoho vyučujících nepředstavitelné. (Doseděl, 2020)

V září, tedy na začátku nového školního roku, vydalo Ministerstvo školství, mládeže a tělovýchovy Metodické doporučení pro vzdělávání distančním způsobem[5], ve kterém mimo jiné zmiňuje vhodnost používaní jednotné komunikační platformy jako například MS Teams. Ačkoliv se výuka a komunikace výrazně zlepšila, jeden zásadní problém přetrvává – a to ověřování znalostí u žáků. V první vlně pandemie někteří vyučující vyčkávali, vyhýbali se alternativním možnostem a na žáky nahrnuli testy z učiva za několik měsíců během června, kdy pandemie slábla a prezenční varianta výuky byla obnovena. Podobná situace se opakovala i v novém školním roce v prosinci, kdy došlo k opětovnému uvolnění opatření, ačkoliv se jedna z kapitol Metodického doporučení věnuje právě způsobům a pravidlům hodnocení. Ministerstvo zmiňuje nepostradatelnost kvalitní zpětné vazby a doporučuje nahradit běžné sumativní hodnocení, jehož formou je známkování, formativním hodnocením, kte-

[5]https://www.edu.cz/methodology/metodika-pro-vzdelavani-distancnim-zpusobem/

ré se neorientuje primárně na výsledek. V doporučení jsou zmíněny také konkrétní názvy aplikací, které by měli učitelé využívat – ty jsou ale označeny jako nástroje podporující autoevaluaci, nikoliv jako adekvátní náhrada za běžné písemné zkoušky. Jedná se o aplikace *Kahoot*, *Socrative*, *Microsoft Forms* a *Google Forms*. Často se ale v prostředí základních i středních škol vyučující uchylují k tomu, že zadání zkoušky raději zveřejní žákům ve Wordu či Excelu a stanoví termín, do kterého mají žáci vyplněnou zkoušku odevzdat – ať už emailem, nebo například do odevzdávárny na MS Teams. Dokáží tedy tyto aplikace nahradit písemné zkoušky? A jaké jsou jejich nedostatky a omezení, se kterými se učitelé potýkají? Tomu se budu detailně věnovat v kapitole 5 týkající se dostupných řešení.

3 Aplikační software

Tato kapitola se věnuje charakteristice aplikačního softwaru a zejména jednotlivým typům aplikací včetně jejich základních vlastností a odlišností. Závěr kapitoly je věnován radám, jak správně postupovat při rozhodování, který z typů je vhodný pro konkrétní projekt.

3.1 Charakteristika

Aplikační software, označovaný též jako *aplikace*, je program vykonávající specifickou činnost pro koncového uživatele, případně pro jinou aplikaci. Aplikační software může zahrnovat jak jedinou aplikaci, tak celý balík různých aplikací. (Rouse, 2018)

Pro komunikaci aplikace s uživatelem je využíváno grafické uživatelské rozhraní, textové uživatelské rozhraní nebo příkazový řádek. Grafické rozhraní je nejoblíbenější pro běžné uživatele, protože obsahuje interaktivní ovládací prvky a práce s ním je velmi intuitivní. Naopak příkazový řádek vyžaduje od uživatele přesnou znalost příkazů, pomocí kterých se aplikace ovládá, a neumožňuje použití myši.

Příkladem aplikací mohou být kancelářské balíky, výukové programy, programy pro elektronickou poštu, webové prohlížeče, hry, grafické editory, vývojové nástroje, informační systémy a další.

3.2 Rozdělení aplikací

Každá aplikace je závislá na určitém prostředí či operačním systému podle toho, jakým způsobem a s jakým cílem byla vyvíjena. Z tohoto pohledu lze rozlišit aplikace desktopové, webové, mobilní či cloudové. Každý typ aplikace je vhodný pro jiné využití a má své výhody i nevýhody.

3.2.1 Desktopové aplikace

Počátkem roku 2000 byla většina softwaru pro stolní počítače zabalená do krabic, které se prodávaly ve větších obchodech. Z toho také vzniklo označení *krabicový software*. Na krabici byly uvedeny systémové požadavky, podle kterých bylo možné se rozhodnout, zda daný software bude fungovat na požadovaném operačním systému s určitou hardwarovou konfigurací. Po nalezení a zakoupení vyhovujícího softwaru stačilo už jen vytáhnout z krabice CD, které se vložilo do CD-ROM mechaniky a software se nainstaloval do počítače. (Jensen, 2017)

Obrázek 2: Ukázka krabicového softwaru
Zdroj: http://www.buildorbuy.org/themes/800/win98box.jpg

Postupem času se s rozvojem webových prohlížečů a rozšířením přístupu k internetu, jehož rychlost je stále navyšována, změnil způsob vývoje a především distribuce softwaru. Namísto prodeje v krabicích se jeho prodej přesunul na internetové obchody, ve kterých si zákazníci zakoupí virtuální produkt, respektive licenci k virtuálnímu produktu, který si následně prostřednictvím webového prohlížeče stáhnou do svého počítače. Díky tomuto kroku v distribuci se software dostal mnohem blíže k zákazníkům a jeho pořízení začalo být mnohem snazší. (Jensen, 2017)

Desktopové aplikace jsou tedy typ softwaru, který se instaluje přímo na počítač a který je možné spouštět nezávisle na jiných aplikacích. Kurilo (2020) mezi jejich specifické vlastnosti řadí:

- **Nevyžadováno připojení k internetu.** Desktopové aplikace jsou většinou schopny pracovat nezávisle na připojení k internetu. Nedochází tak k výpadkům či zpomalení při zhoršené kvalitě internetového připojení.
- **Optimální využití zdrojů a vyšší výkonnost.** Operace jsou vykonávány velmi rychle za předpokladu kvalitního hardwaru.
- **Bez vzdáleného přístupu.** Desktopové aplikace jsou vázány na jeden systém a není možné k nim přistupovat například z jiného počítače.
- **Nutnost instalace.** Aplikace tohoto typu využívají prostor pevného disku zařízení, na kterém jsou nainstalovány. Taktéž data v nich vytvořená jsou obvykle ukládána přímo do zařízení. Množství aplikací, které lze využívat, je závislé na velikosti pevného disku.
- **Ruční aktualizace.** Zatímco například u webových aplikací jsou aktualizace automaticky prováděny vývojáři a aplikace je vždy po spuštění aktuální, u desktopových aplikací se zobrazí pouze informaci o dostupné aktualizaci. Dokud ji uživatel nenainstaluje, novou verzi nemá.

S desktopovými aplikacemi se setkáváme při každodenních činnostech. Na počátku tohoto tisíciletí byl dominantním systémem Microsoft Windows a právě pro

tento operační systém vznikala většina aplikací. Postupně se ale přidávaly další operační systémy – macOS od Applu, Google Chrome OS od Googlu či svobodný operační systém Linux. S tím přišlo uvědomění, že již není možné vyvíjet aplikace pouze pro Microsoft Windows, ale zároveň není efektivní vytvářet aplikace pro každý z těchto operačních systémů zvlášť. Vznikl tak pojem *multiplatformní desktopové aplikace*, což je označení pro aplikace, které mohou běžet na více počítačových platformách. (Jensen, 2017)

V začátcích multiplatformních desktopových aplikací vznikly frameworky jako *Mono* či *Qt*, které umožňovaly vyvíjet desktopové aplikace fungující pod třemi hlavními operačními systémy. Pro schopnost práce s těmito frameworky bylo potřeba znát programovací jazyky jako C, C++ a C#. Porozumění nejen samotnému jazyku, ale i jeho uplatnění v rámci frameworku vyžadovalo velké množství času, zatímco porozumění a naučení se technologií využívaných při webovém vývoji bylo z časového hlediska mnohem méně náročné. Na základě této myšlenky vznikly nové frameworky *NW.js* a *Electron*, které jsou skvělou příležitostí pro vytváření desktopových aplikací za pomocí velmi podobného kódu, jako se používá při vytváření webových aplikací. Především znovupoužitelnost kódu a využití již získaných dovedností vedlo k velkému rozmachu desktopových aplikací realizovaných tímto způsobem. (Jensen, 2017)

3.2.2 Webové aplikace

Ačkoliv by se mohly zdát multiplatformní desktopové aplikace jako ideální řešení, stále mají některá zásadní negativa, která uživatele do jisté míry omezují. Jakmile začalo rozšiřování internetu do firem i domácností a jeho rychlost se stávala dostatečnou pro běžnou práci, začaly se objevovat webové aplikace, které poskytovaly podobnou funkčnost jako ty desktopové.

Webové aplikace jsou takové aplikace, jejichž zdrojové soubory jsou umístěny na vzdáleném serveru a k nimž je přistupováno prostřednictvím webového prohlížeče. Stejně jako v desktopových aplikacích lze vytvářet a upravovat obsah a manipulovat s daty. Kurilo (2020) mezi specifické vlastnosti webových aplikací řadí:

- **Žádná instalace.** Spuštění aplikace probíhá zadáním URL adresy do prohlížeče. Aplikace sama o sobě nevyužívá žádné místo na disku zařízení, ze kterého je k ní přistupováno.
- **Přístup odkudkoliv a z jakéhokoliv zařízení.** Jednou z hlavních výhod webových aplikací je jejich nezávislost na zařízení. Zatímco k desktopové aplikaci je nutné přistupovat ze zařízení, na kterém byla nainstalována, k webové aplikaci lze přistupovat zcela odkudkoliv. A to nejen z počítače, ale také z tabletu či mobilního telefonu, respektive z jakéhokoliv zařízení, které disponuje webovým prohlížečem.
- **Automatické aktualizace.** Při každém spuštění webové aplikace se automaticky spustí nejnovější verze. Není potřeba stahovat aktualizaci nebo novou

verzi aplikace, instalovat ji a znovu spouštět. I když se může zdát, že to není nic náročného, problém nastává i z toho pohledu, že uživatelé aktualizace odkládají a pracují s verzemi aplikací, které například nemají opravené odhalené bezpečnostní chyby nebo postrádají novou důležitou funkcionalitu.

- **Závislost na připojení k internetu.** Ačkoliv je v dnešní době již přístup k internetu možný téměř odkudkoliv a kdykoliv, je potřeba tuto vlastnost zmínit. K webovým aplikacím není možné přistupovat bez připojení k internetu, jelikož je nutné, aby se prohlížeč spojil se vzdáleným serverem a získal potřebná data.
- **Bezpečnostní hrozby.** Jakmile jsou data umístěna na vzdáleném serveru, je třeba brát v potaz otázku bezpečnosti. Webové servery jsou častým cílem útočníků a veškerá citlivá data je potřeba v maximální míře chránit.

3.2.3 Mobilní aplikace

Dalším typem aplikací jsou mobilní aplikace. Jak již název vypovídá, jedná se o aplikace, které jsou určeny pro mobilní zařízení. V posledních letech je o vývoj aplikací pro chytrá zařízení velký zájem. Podle statistik[6] ze září roku 2020 je v internetovém obchodu App Store, který nabízí aplikace pro mobilní zařízení s operačním systémem iOS, celkem 1,85 milionů aplikací. V obchodu Google Play určeném pro uživatele se systémem Android se nachází 3,04 miliony aplikací. K výraznému nárůstu počtu aplikací, ale i množství jejich stažení došlo v roce 2020 kvůli pandemii koronaviru, během které se do online prostředí přesunulo mnoho lidí všech věkových kategorií mimo jiné s cílem udržet si vzdálený kontakt se svou rodinou, přáteli i kolegy. V roce 2020 využívá chytré mobilní telefony po celém světě téměř 3,5 miliardy uživatelů, což při počtu 7,7 miliardy lidí na planetě Zemi je téměř polovina lidí. Zatímco v některých státech ještě nejsou chytré telefony tak oblíbené, v Jižní Koreji vlastní chytrý mobilní telefon 95 % dospělých.

Co nám ale tyto statistiky mohou prozradit? Vzhledem k tomu, že téměř polovina všech lidí na světě vlastní mobilní telefon a většina z nich jej pravděpodobně aktivně využívá, z mobilních aplikací se stává mocný nástroj. I když mají mobilní zařízení prohlížeč, ve kterém je možné zobrazit běžné webové aplikace, mobilní aplikace mají své specifické vlastnosti, kterými se od těch webových odlišují a které jim dávají přidanou hodnotu. Podle Xanthopoulose a Xinogalose (2013) mezi hlavní vlastnosti mobilních aplikací, kterými se od webových aplikací odlišují patří:

- **Přístup k hardwaru a datům.** Mobilní aplikace mají zásadní výhodu v tom, že mohou přistupovat k hardwaru daného zařízení a k datům, která jsou na něm uložena. Mohou komunikovat například s fotoaparátem, galerií, GPS, mikrofonem či seznamem kontaktů.

[6] https://www.statista.com/

- **Mohou pracovat offline.** I když dnes již mnoho mobilních aplikací ke své plné funkčnosti využívá připojení k internetu, ve své podstatě mohou mobilní aplikace pracovat nezávisle na dostupnosti internetu.
- **Vyšší výkonnost.** Stejně jako desktopové aplikace jsou výkonnější než webové aplikace, o mobilních aplikacích je možné říci to stejné s ohledem na to, že jsou umístěny přímo v zařízení.

Při vývoji mobilních aplikací musíme rozlišovat mezi *nativními* a *hybridními* mobilními aplikacemi. **Nativní mobilní aplikace** je vyvíjena pomocí programovacích jazyků, které jsou podporovány přímo daným operačním systémem. Tyto programovací jazyky jsou odlišné pro Android, iOS, Windows Phone i další operační systémy. Výhodou nativních mobilních aplikací je především vyšší výkon, nevýhodami je vyšší čas i cena vývoje, nutnost dodržovat přesně předepsané pokyny daného ekosystému a nutnost naučit se specifický programovací jazyk. **Hybridní mobilní aplikace** je vyvíjena pomocí univerzálních technologií zabalených následně do balíků, které fungují na mnoha mobilních zařízeních s různými operačními systémy. Obvykle se k jejich vývoji využívají jednodušší technologie jako HTML, CSS, JavaScript a některý z frameworků, který umožní přistupovat k hardwaru stejně tak, jako je to možné u nativních aplikací. Hlavními výhodami jsou nezávislost na platformě, snadný vývoj a časová i finanční nenáročnost. (Panhale, 2016)

3.2.4 Cloudové aplikace

Posledním a zároveň nejnovějším typem aplikací, o kterém se zmíním, jsou cloudové aplikace. Jsou to takové aplikace, ve kterých je část logiky zpracována přímo v cloudu. Taktéž data jsou v cloudu uložena. Uživatel k takové aplikaci přistupuje přes webový prohlížeč a aplikace se tváří stejně jako běžná webová stránka či webová aplikace. (Barton, 2018)

V čem je tedy zásadní rozdíl mezi webovými a cloudovými aplikacemi? V architektuře. Webové aplikace potřebují stálé připojení k internetu, aby správně fungovaly, oproti tomu cloudové aplikace mohou vykonávat část procesů na lokálním zařízení a s cloudem komunikovat především z důvodu stahování a odesílání dat. Podle Bartona (2018) existují tři základní způsoby návrhu cloudových aplikací:

- Na lokálním zařízení se spustí webový prohlížeč a aplikace vypadá jako běžné webové řešení. O logiku, data i formát výstupu se v tomto případě stará cloud. Příkladem takové aplikace je *DropBox*, ve kterém mohou uživatelé nahrávat a editovat soubory, avšak všechna data jsou uložena na vzdáleném cloudu.
- Na lokálním zařízení se spustí webový prohlížeč a při startu aplikace se stáhne aplikace vypadající jako webová aplikace v prohlížeči, která ale běží lokálně na zařízení. Tato lokální aplikace se stará o část procesů a komunikuje prostřednictvím API s cloudem. Příkladem takové aplikace je *Google Docs*, kde i po

ztrátě internetového připojení je možné dokumenty editovat. Změny se uloží lokálně a při dalším obnovení připojení se promítnou na server.

- Na lokálním zařízení se spustí nativní aplikace, která se stará o většinu logiky a v případě výpadku připojení ukládá data lokálně. Po obnovení se data synchronizují se serverem.

S pokrokem technologií se hranice mezi cloudovými a webovými aplikacemi začínají stále více stírat. V obou případech jsou data uložena na vzdáleném úložišti a ke zpracování velké části procesů se využívá výkon serveru. Často se tak jedná jen o obchodní nálepku, která má za cíl působit na zákazníka jako nějaké nové jedinečné řešení. (Barton, 2018)

3.3 Jak si správně vybrat?

Výběr typu aplikace, který bude v rámci projektu realizován, je zásadní krok, který je vhodné důkladně promyslet. Pokud by zvolený typ nevyhovoval a byla by později potřeba změna, s velkou pravděpodobností by bylo nutné veškerou odvedenou práci zahodit a začít od začátku. Jaké otázky je vhodné si předem položit?

V první řadě je to **účel využití aplikace**. Pokud se bude jednat o aplikaci, která bude využívána ve firmě vybavené stolními počítači či notebooky, nebylo by rozumné se rozhodnout pro vývoj mobilní aplikace. S tím také souvisí **náročnost aplikace na velikost pracovní plochy**. V případě grafického editoru či aplikace pro střih videa bude potřebná velká obrazovka, naopak v chatovací aplikaci si lze vystačit s obrazovkou mobilního zařízení.

Dalším parametrem je **potřebná výkonnost**. Aplikace zobrazující aktuální týdenní menu vybraných restaurací bude z pohledu výkonnosti téměř nenáročná, oproti tomu aplikace pro práci s 3D grafikou již bude potřebovat výkonné zařízení. Také **možnost vzdáleného přístupu** je třeba dobře zvážit. Budou cíloví uživatelé vždy používat aplikace na stejném zařízení a na stejném místě, nebo se budou chtít mít možnost do aplikace připojit i z domu a mít v ní svá aktuální data?

Ani **aktualizace** není dobré podcenit. Uživatelé nemusí být technicky zdatní a pokud po nich bude vyžadováno, aby několikrát do měsíce sami prováděli aktualizaci, většinu z nich to jistě nepotěší. Pokud bude aktualizace provádět technický pracovník a ve firmě na něj bude čekat několik desítek nebo stovek zařízení s danou aplikací, připraví jej to o mnoho času.

Uvedu celou situaci na příkladu. Cílem je připravit novou aplikaci pro vedení účetnictví. Nejdříve lze vyloučit mobilní zařízení, protože aplikace bude rozsáhlá a bude potřeba velká interakce s klávesnicí. Z pohledu výkonnosti nebude aplikace nijak extrémně náročná, proto lze zvažovat jak desktopovou, tak webovou aplikaci. Teď už záleží na konkrétní specifikaci. Pokud v aplikaci bude pracovat více účetních a budou chtít mít možnost sdílet spolu účetnictví, není možné využít aplikaci s lokálním úložištěm. V takovém případě bude webová aplikace nejlepší možnou volbou.

4 Moderní přístupy k vývoji webových aplikací

Tato kapitola pojednává o počátcích vzniku webových aplikací a současných trendech v jejich vývoji. Zaměřuje se také na single-page aplikace, tradiční i moderní návrhové vzory, technologie využívané pro frontend i backend a databázové systémy.

4.1 Počátky a okolnosti vzniku webových aplikací

Na počátku 90. let 20. století byl představen koncept *World Wide Web*, jehož cílem bylo umožnit vědcům z různých oblastí přistupovat jednoduchým způsobem k jakýmkoliv informacím a dokumentům a mít také možnost sdílet i své dokumenty s dalšími vědci. World Wide Web představoval rozsáhlé úložiště informací, ke kterému bude mít přístup velký počet uživatelů. Ačkoliv si dnes pod tímto pojmem každý představí výkonnou platformu poskytující rozsáhlé množství nástrojů a aplikací, které umožňují spolupracovat a vzájemně komunikovat, tehdy byl pohled pouze statický. Až postupem času zjistili, že adresa statické stránky by se mohla proměnit na adresu odkazující na program, který lze spustit na serveru a který může uživateli vrátit výsledek. (Jazayeri, 2007)

V roce 1999 stanovil profesor Piero Fraternali požadavky, které musí webová aplikace splňovat, aby mohla být klasifikována jako webová aplikace. Musí:

- být schopna pracovat s nestrukturovanými daty (např. databázové záznamy) i strukturovanými daty (např. multimediální soubory),
- mít grafické rozhraní na vysoké kvalitě,
- obsahovat navigační prvek pro přístup k datům,
- se zabývat dynamikou, strukturou obsahu a způsobem prezentace
- a podporovat proaktivní chování (tj. filtrování obsahu).

Počátkem 21. století se vývoj webových aplikací a celého Webu velmi zrychlil. Z konceptu World Wide Web se stal koncept *Web 2.0*, který charakterizuje webové aplikace jako pružné dynamické aplikace interagující s uživatelem. O popularizaci tohoto konceptu se postaral Tim O'Reilly. Facebook, Wikipedie, Twitter, Skype, YouTube, WordPress, Google a tisíce dalších, to jsou typičtí představitelé konceptu Web 2.0. Jedná se tak převážně o sociální sítě, virtuální encyklopedie, video platformy, blogovací systémy, vyhledávače či komunikační aplikace. (Jazayeri, 2007)

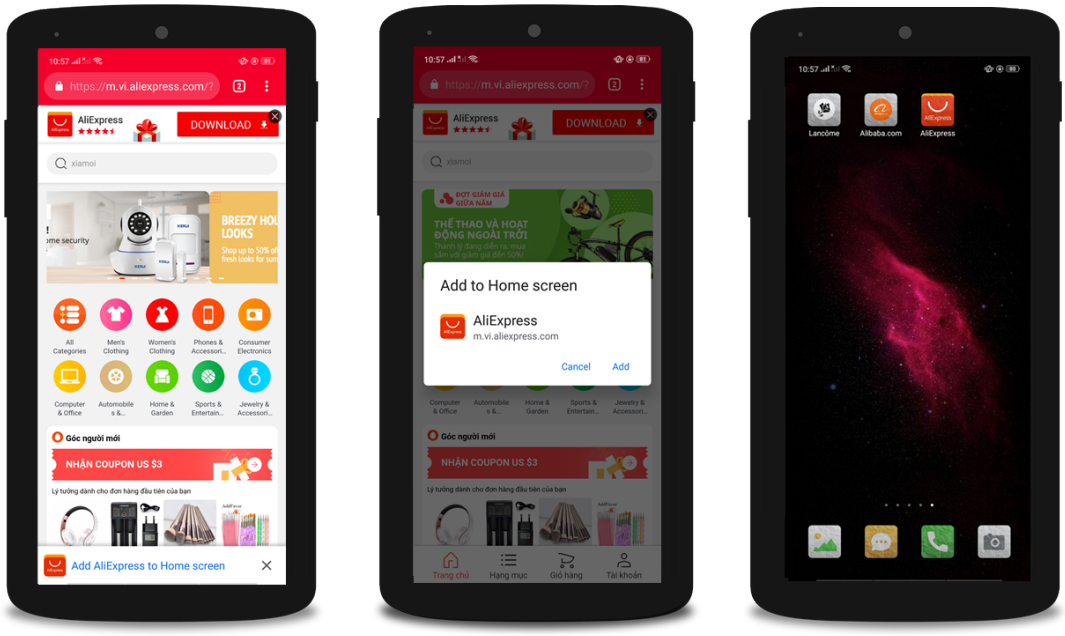
4.2 Současné trendy ve vývoji webových aplikací

Pro úspěšné vytvoření moderní webové aplikace je třeba následovat současné trendy, které udávají, jakými směry se vývoj v této oblasti ubírá a na které aspekty by se měl vývojář zaměřit. V následujících podsekcích jsou vybrány a představeny ty nejdůležitější a nejzajímavější z nich.

4.2.1 Progresivní webové aplikace

Progresivní webové aplikace (označované zkratkou PWA) kombinují prvky klasických webových aplikací s prvky nativních aplikací. S neformální definicí progresivních webových aplikací přišel v roce 2015 Alex Russell, softwarový inženýr z Googlu, s designerem Francesem Berrimanem. Alex Russell na svém blogu píše, že progresivní webové aplikace musí splňovat několik vlastností – musí být **responzivní**, **nezávislé na připojení**, tedy schopné pracovat v offline režimu, mít schopnost **interakce jako aplikace**, vždy **aktuální**, dostatečně **zabezpečené**, **dohledatelné** díky W3C manifestu poskytujícímu základní údaje o aplikaci, **instalovatelné** na základě pokynů přímo v prohlížeči bez nutnosti návštěvy obchodu s aplikacemi jako Google Play nebo App Store, a **dostupné** prostřednictvím URL adresy. (Russell, 2015)

Jak progresivní webovou aplikaci uživatel pozná? Pomocí běžného prohlížeče na danou webovou stránku přistoupí stejně jako na jakoukoliv jinou webovou stránku. Jakmile stránka odhalí, že s ní uživatel dostatečně pracuje, zobrazí se výzva o ponechání. Pokud uživatel výzvu přijme, na domovskou obrazovku se přidá ikona dané stránky, aniž by byl chod nějak narušen. Když uživatel na stránku příště přistoupí kliknutím na ikonu, stránka se zobrazí na celou obrazovku ve svém prostředí (tedy ne v běžném prohlížeči), a to i bez připojení k internetu. (Russell, 2015)



Obrázek 3: Ukázka chování progresivní webové aplikace čínského online tržiště AliExpress
Zdroj: https://www.tigren.com/examples-progressive-web-apps-pwa/

Rok 2018 byl zlomový pro úspěch progresivních webových aplikací. Počínaje tímto rokem začaly firmy upravovat své webové aplikace tak, aby splňovaly parametry progresivních aplikací. Mnohým z nich to přineslo znatelné úspěchy v podnikání, o nichž informuje web PWA Stats[7]. Mezinárodnímu výrobci kávy a řetězci kaváren Starbucks se díky PWA podařilo navýšit denní počet unikátních návštěvníků přicházejících na jejich web z mobilu na dvojnásobek, čímž dosáhli téměř stejného počtu objednávek od desktopových uživatelů a mobilních uživatelů. Službě pro přepravu cestujících Uber se po vytvoření PWA podařilo dosáhnout toho, že se jejich aplikace načte za méně než 3 vteřiny i na velmi pomalé a zastaralé telefonní technologii 2G. Čínskému online tržišti AliExpress přineslo PWA o 104 % více uživatelů, kteří navštěvují 2 krát více stránek než dříve a stráví zde o 74 % více času.

Zatímco ze zahraničí přicházejí zprávy o úspěších díky PWA stále častěji a nyní se těší obrovské popularitě, na českém poli je tento trend teprve v počátcích. Mezi první patří úspěchy společností Kiwi, českého internetového vyhledávače letenek, a Dáme jídlo, služby pro rozvoz jídla z restaurací k jednotlivcům.

4.2.2 Boti a umělá inteligence

Jedním ze současných trendů je používání chatbotů založených na umělé inteligenci. Chatbot je aplikace sloužící k automatizované komunikaci s lidmi. V čem spočívá kouzlo chatbotů? Lidé jsou nyní zvyklí na to, že když potřebují v online prostředí něco najít, zadají požadovaná slova do vyhledávacího pole, získají desítky, stovky, tisíce a často i statisíce výsledků, které prochází a hledají ten nejrelevantnější pro jejich potřeby. To se ale s příchodem chatbotů začíná výrazně měnit – představují totiž komunikaci mezi zákazníkem (uživatelem) a webovou aplikací. Uvedu příklad na situaci obchodu se sportovním vybavením. Pokud půjde zákazník vybírat nové lyže do kamenného obchodu a natrefí na schopného prodejce, bude mu klást otázky ohledně jeho představ a požadavků, aby mu pomohl s výběrem. V případě internetového obchodu bude mít různé kategorie zboží a případně filtry, na základě kterých se okruh zboží zúží. Avšak takový způsob může být pro spoustu zákazníků nepohodlný a mohou se cítit bezradní. Chatbot pro takový internetový obchod může přinést nový rozměr nakupování. Zákazník položí chatbotovi dotaz na zboží a ten začne reagovat a pokládat upřesňující otázky stejně jako reálný prodavač.

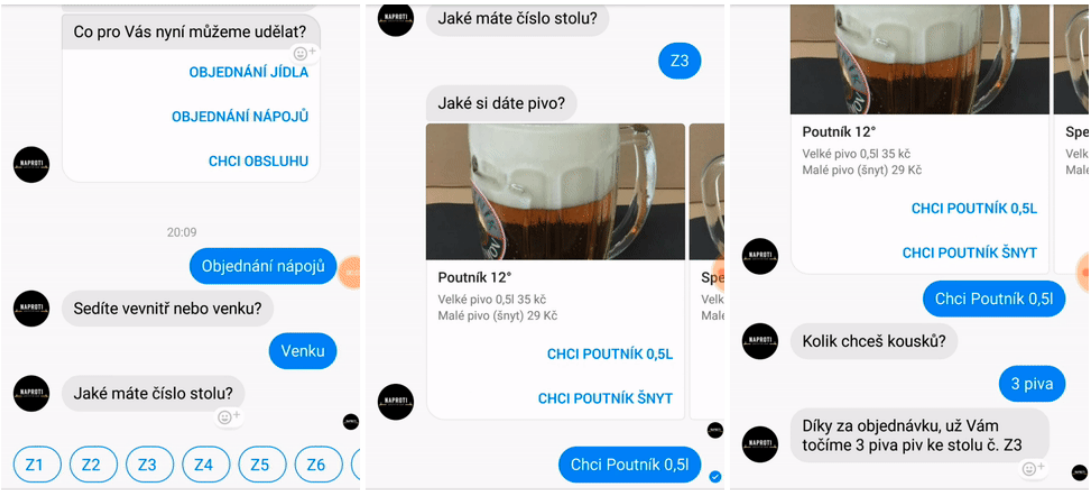
Sféra internetových obchodů není ani zdaleka jediným místem, kde se mohou chatboti uplatnit. Například moderní internetové bankovnictví může se zákazníkem komunikovat stejně jako bankéř na pobočce či technickou podporu (tzv. help desk) v mnoha institucích může do určité míry zastoupit schopný chatbot. To může přinášet spoustu výhod nejen v nižších nákladech na zaměstnance, ale také v pohodlnosti v podobě neomezené časové dostupnosti pro zákazníky.

Umělá inteligence je nyní považována za nejvýznamnější trend v digitálním světě. Známé společnosti jako Facebook, Google, Microsoft či Amazon vytvořily řadu open-source vývojových platforem chatbotů. Weboví vývojáři nemusejí studovat

[7] https://www.pwastats.com/

programovací jazyky pro umělou inteligenci, ale vystačí si s běžnými jazyky jako Python, Javascript, C++ či .NET. Zmíním zde nejznámější frameworky a platformy pro vytváření chatbotů.

Dialogflow je nástroj Googlu sloužící k vytváření chatbotů a aplikací ovládaných hlasem. V současnosti podporuje přes 20 jazyků a do aplikací může být integrován prostřednictvím REST API. Dílem Microsoftu je *Microsoft Bot Framework*. Posledním, který zmíním, je *Amazon Lex* – ten nabízí vývojářům technologie hlubokého učení, které sám Amazon používá ve svém virtuálním asistentovi Amazon Alexa.



Obrázek 4: Ukázka chatbota OrderNow pro restaurace a hospody
Zdroj: https://blog.talfa.cz/zajimave-ceske-messenger-chatboti/

4.2.3 AMP

Zkratka AMP je označení pro *Accelerated Mobile Pages* – v doslovném překladu *zrychlené mobilní stránky*. Jedná se o open-source projekt, za kterým stojí společnost Google. Podle oficiálních stránek projektu[8] jde o projekt, jehož cílem je zlepšit celý ekosystém mobilního obsahu. Na rychlost načítání webových stránek je dnes kladen velký důraz. Delší doba načítání stránky může zásadně ovlivnit to, zda se uživatel rozhodne na stránce zůstat nebo se přesune na jinou stránku s podobným obsahem, která pro něj bude uživatelsky přívětivější. Projekt AMP se snaží o to, aby se stránky načítaly extrémně rychle a aby bylo možné se po nich snadno pohybovat v mobilním prostředí. AMP může velmi ovlivnit dobu, kterou uživatel na webu stráví, a v případě online prodejů také výrazně zvýšit tržby. Projekt je vhodný pro všechny typy projektů – od blogů přes zpravodajské servery až po videoplatformy.

Tyto stránky jsou stejné jako jiné běžné HTML stránky, avšak mají omezenou sadu funkcionalit vycházejících ze specifikace AMP. Všechny tyto funkcionality jsou dostupné prostřednictvím knihovny webových komponent, která se stále rozšiřuje. U všech komponent je prioritně kladen důraz na rychlost. Vytvořené AMP soubory je

[8] https://amp.dev/support/faq/overview/

možné ukládat do cache paměti, což ještě zrychluje dobu potřebnou k získání obsahu. Pokud nechce konkrétní projekt používat vlastní cache pamět, Google k projektu poskytuje také svoji Google AMP Cache, která je zcela zdarma pro použití kýmkoliv bez jakýchkoliv limitů.

4.2.4 Motion design v uživatelském rozhraní

Smysl motion designu v uživatelském rozhraní spočívá v rozpohybování a animování jednotlivých elementů webu či webové aplikace. Hlavní výhodou motion designu je, že přitahuje pozornost uživatele. Díky lidskému periferním vidění je s pomocí motion designu možné upozornit uživatele na konkrétní místo, a to i takové, které zrovna v danou chvíli nesleduje přímým pohledem. Všechny tyto animace by ale měly být plynulé a minimalistické tak, aby přilákaly pozornost, ale zároveň nebyly pro uživatele rušivé. Hlavním cílem animací by mělo být podpoření způsobu použitelnosti daného webu či aplikace. (Laubheimer, 2020)

Příkladem uplatnění motion designu je fixní horizontální hlavička, která je responzivní a mění svoji výšku v závislosti na šířce obrazovky, respektive šířce okna. V takovém případě je vhodné, aby při každé změně byl přechod mezi velikostmi plynulý a nikoliv skokový.

Další oblastí jsou grafy, u kterých je v současnosti kladen velký důraz na interaktivitu. Nechť existuje graf úmrtí na covid-19, kde jednotlivé sloupce budou měsíce a výšky těchto sloupců budou počty zemřelých v daném měsíci, přičemž konkrétní zemi bude možné vybrat ze seznamu. Příkladem využití motion designu v takovém případě je, že při změně země se jednotlivé sloupce buď plynule zvětší, nebo plynule zmenší, aniž by se změnil celý graf sám o sobě. Taková animace pro uživatele způsobí, že si mnohem lépe uvědomí, jaké rozdíly mezi danými zeměmi jsou, než kdyby se jen celý graf změnil během okamžiku.

Vyskakovací okna, rozbalovací nabídky nebo rolování, to jsou další z míst, kde je přínosné uplatnit motion design. (Laubheimer, 2020)

4.2.5 Tmavé téma

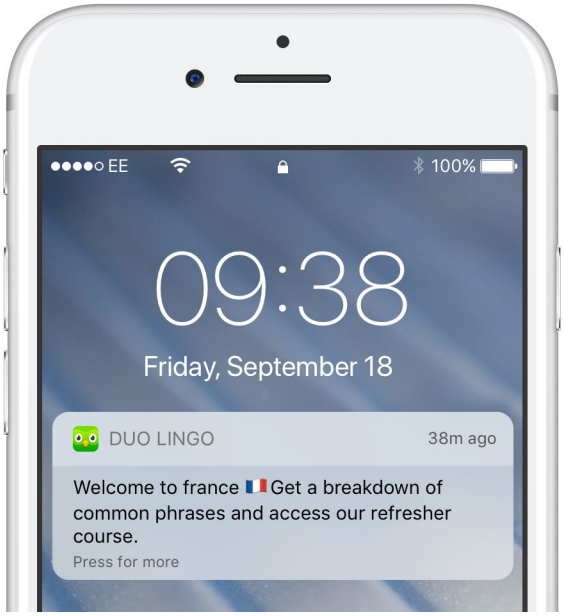
Temný režim u webů i webových aplikací si získal v posledních letech velkou oblibu uživatelů po celém světě a těší se stále větší popularitě. Facebook, Twitter, YouTube či WhatsApp, to jsou jen některá z populárních jmen, která nabízejí svým uživatelům tmavé téma. Zatímco u běžných uživatelů se jedná o novinku, mezi vývojáři je tmavé téma populární již mnoho let. Co stojí za oblíbeností tohoto trendu?

Popularita temného režimu úzce souvisí s biologií člověka. Tento režim může výrazně snížit namáhání očí ve špatných světelných podmínkách. Je ale třeba brát ohled na kontrast textu a pozadí, protože v případě stoprocentního kontrastu (tj. bílého textu a černého pozadí) bude naopak čtení mnohem náročnější a oči budou více namáhány. Obecně je doporučováno využívat světlé téma během dne a tmavé téma během noci. S tím souvisí, že nejlepší možností pro uživatele je poskytnutí volby mezi světlým a tmavým tématem. (Jerelyn, 2020)

Za tímto trendem ale nestojí pouze vliv na oči. Zajímavostí je, že Google provedl experiment vlivu světlého a tmavého tématu na spotřebu baterie. V rámci experimentu došli k závěru, že tmavé téma může snížit při maximálním jasu spotřebu baterie až o 63 % na AMOLED displejích, protože bílé pixely jsou výrazně náročnější na energii než tmavé pixely. S nižší spotřebou baterie souvisí vyšší životnost baterie, přičemž na oba tyto parametry je v posledních letech kladen velký důraz zejména u mobilních zařízení. (Jerelyn, 2020)

4.2.6 Push notifikace

Push notifikace jsou zprávy objevující se na mobilním zařízení. Jsou podobné textovým SMS zprávám, avšak mohou chodit pouze těm, kteří využívají aplikaci či stránku, ze které pocházejí, a kteří je povolili. (Mali, 2018)



Obrázek 5: Ukázka push notifikace
Zdroj: https://appinstitute.com/what-is-a-push-notification/

Využití push notifikací je velmi rozsáhlé. Podívejme se na některé z forem notifikací a příklady jejich konkrétního použití:

- zpětná vazba – požádání zákazníka o hodnocení aplikace
- transakční zprávy – potvrzení objednávky, sledování objednávky, aktualizace stavu objednávky
- události – mimořádné slevové akce, důležité aktuality
- personalizované nabídky – produkty podle dřívějšího vyhledávání

- geograficky cílené nabídky – upozornění zákazníka, že se nachází poblíž své oblíbené restaurace či kavárny

Push notifikace jsou mocným prostředkem pro komunikaci se zákazníkem, avšak musí být správně provedeny. Notifikacím je mimo jiné potřeba zvolit výstižný titulek i popis, cílit je na vhodné uživatele, správně je načasovat a využívat výstižná slova motivující k akci. (Mali, 2018)

4.2.7 A11Y

Projekt A11Y[9] se zabývá přístupností webových stránek a webových aplikací. Zkratka A11Y vychází z anglického slova *accessibility*. Hlavní myšlenkou tohoto projektu je, aby vývojáři mysleli na to, že na světě existuje velké množství lidí, kteří mají nějaké zdravotní postižení či omezení. Webové stránky i aplikace přitom lze přizpůsobit tak, aby jim tito lidé byli schopni plně porozumět díky čtečce obrazovky, braillskému hmatovému displeji či zvětšovací lupě. Mnoho z nich je schopno se díky těmto zařízením velmi přiblížit k běžnému životu a objednávat si jídlo, hledat si práci či komunikovat se svými přáteli. Při návrhu se doporučuje dbát na to, aby výsledku porozumělo co nejširší publikum – lidé trpící barvoslepostí, epilepsií, dyslexií, dyskalkulií a dalšími poruchami.

Pod tímto projektem se skrývá celá řada technik, které pokud vývojář dodrží, vyprodukuje přístupný web. Přístupnost je v mnoha zemích právo, které je chráněno zákonem. Mezi konkrétní techniky patří například:

- používání prostého jazyka bez idiomů a metafor
- používání atributu `lang` v `html` elementu pro učení jazyka stránky
- viditelný odlišný styl pro element, který je aktivní (tj. je v něm kurzor), aby uživatel věděl, kde se na stránce právě nachází
- používání atributu `alt` u všech `img` elementů, který slouží k popisku obrázku pro uživatele, kteří jsou nevidomí
- dodržování úrovní nadpisů a používání pouze jednoho nadpisu první úrovně `h1`
- zakázání automatického spouštění multimediálního obsahu, aby nedošlo k zneklidnění lidí trpících ADHD a podobnými postiženími

Kompletní seznam technik se nachází na oficiálních stránkách projektu jako interaktivní kontrolní seznam[10], který si mohou vývojáři během vývoje odškrtávat a tím docílit přístupného webu.

[9] https://www.a11yproject.com/
[10] https://www.a11yproject.com/checklist/

4.2.8 Další současné trendy

Mezi další současné trendy v oblasti vývoje webových stránek a webových aplikací patří mimo jiné:

- **Hlasové vyhledávání**. K masivnímu rozšíření hlasového vyhledávání došlo se vznikem hlasových asistentů Google Home, Alexa či Amazon Echo Dot. Nastává éra, ve které by měli vývojáři dbát na to, aby byl web optimalizovaný pro hlasové vyhledávání a tato zařízení byla schopna z webu načítat informace.
- **Single-page aplikace**. Vícestránkové aplikace se nahrazují jednostránkovými aplikacemi, které vytváří lepší uživatelský zážitek.
- **Javascriptové frameworky**. JavaScript pomáhá definovat dynamické chování a je hojně využíván na webových stránkách i v aplikacích. Javasriptové rámce výrazně zjednodušují a urychlují vývoj.
- **Blockchain**. Blockchain je velmi aktuálním tématem v oblasti bezpečnosti informací a transakcí. Kryptoměny se již nepoužívají jen jako platidlo na deep webu a dark webu, ale i některé větší internetové obchody je začínají vnímat a nabízet jako jednu z platebních možností.
- **Bezserverové aplikace a architektury**. Bezserverová architektura umožňuje vyvíjet aplikace bez nutnosti správy architektury. O tu se stará poskytovatel cloudových služeb, který zajišťuje bezpečnost a spolehlivost.

4.3 Single-page aplikace

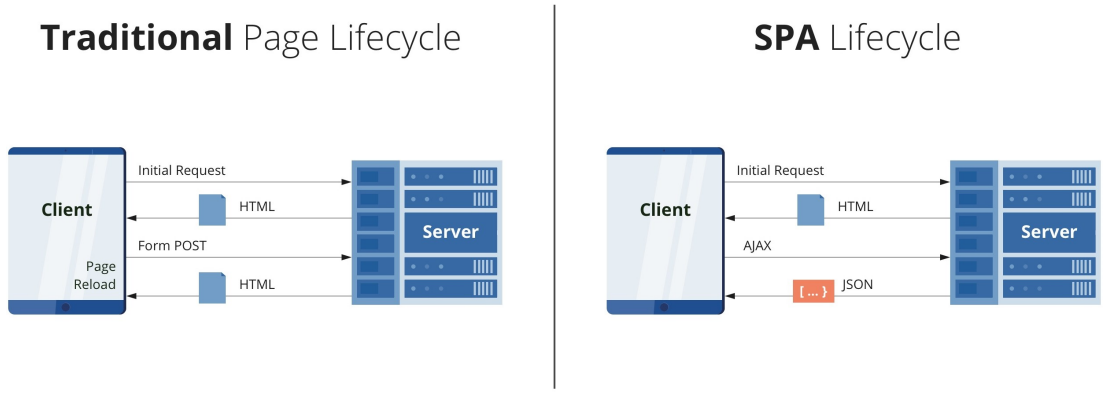
Single-page aplikace, často označované zkratkou SPA, patří mezi nejmodernější přístupy k vývoji softwaru. Jedná se o jednostránkové aplikace, které oproti klasickým aplikacím nabízejí výrazně uživatelsky přívětivější prostředí a značně rychlejší načítání stránek (Scott, 2016). Kde se ale tento koncept vzal a co stojí za jeho vznikem?

Vývojáři se mnoho let snažili (a stále snaží), aby měli uživatelé z rozhraní a práce s webovými aplikacemi stejný pocit, jako kdyby pracovali s aplikacemi desktopovými. Především výkon byl často limitujícím faktorem, a proto vývojáři přicházeli s nativními prostředími jako Java applet, Microsoft Silverlight či Adobe Flash. Všechna tato nativní prostředí se chovají jako pluginy vytvářející nástavbu k prohlížeči. A tak vznikla myšlenka vytvořit koncept single-page aplikace, která bude poskytovat tytéž cíle, ale zároveň nebude nutné mít v prohlížeči nainstalovány speciální pluginy. (Scott, 2016)

Počátky této myšlenky se datují do období, kdy se na povrch začal dostávat AJAX – technologie vývoje webových aplikací, které mění obsah svých stránek bez nutnosti kompletního opětovného načtení. AJAX v kombinaci se schopností JavaScriptu dynamicky aktualizovat Document Object Model (DOM) a schopností CSS měnit styl stránky se dostal do popředí moderního webového vývoje. Koncept single-page aplikace to celé posouvá na ještě vyšší úroveň – manipulace s obsahem

neprobíhá pouze na úrovni konkrétní stránky, ale na úrovní celé aplikace. (Scott, 2016)

4.3.1 Tradiční multi-page vs. moderní single-page aplikace



Obrázek 6: Porovnání životního cyklu multi-page a single-page aplikace
Zdroj: https://dzone.com/articles/the-comparison-of-single-page-and-multi-page-appli

Pro porovnání toho, jak se liší tradiční multi-page aplikace od moderní single-page aplikace, si srovnejme jejich životní cyklus, který je možné vidět na obrázku 6.

V případě tradiční multi-page aplikace probíhá komunikace mezi klientem a serverem tak, že klient odešle inicializační požadavek serveru a server vrátí klientovi HTML dokument. Následné další požadavky ze strany klienta k serveru probíhají prostřednictvím HTTP metod (např. GET a POST), které server zpracuje a opět vrátí klientovi HTML dokument. Při každém takovém navrácení HTML dokumentu musí proběhnout opětovné načtení celé webové stránky. (Scott, 2016)

U single-page aplikace probíhá komunikace mezi klientem a serverem takovým způsobem, že klient odešle inicializační požadavek serveru a server vrátí klientovi HTML dokument. Do tohoto okamžiku je postup shodný jako u multi-page aplikace, avšak další požadavky ze strany klienta jsou k serveru směřovány prostřednictvím technologie AJAX. Odpověď serveru se také liší – namísto celé webové stránky vrátí server pouze JSON, což je datový formát pro přenos dat. Klient data získaná ve formátu JSON zpracuje a aktualizuje příslušné elementy na webové stránce, aniž by musela být celá webová stránka načtená znovu. (Scott, 2016)

Žádné kompletní obnovování stránky, přesun prezentační logiky ke klientovi a odpovědi serveru formou řetězce obsahujícího data, to jsou zásadní změny, které SPA přináší. V následujících odstavcích je rozeberu podrobněji.

V single-page aplikacích se pracuje s tzv. pohledy neboli *views*, což nejsou kompletní HTML stránky, ale pouze určité části z modelu DOM. Při počátečním načtení stránky jsou inicializovány nástroje potřebné k vytváření a správě pohledů. Pokud je potřeba nový pohled, je generován lokálně v prohlížeči a dynamicky vsazen do

DOM pomocí JavaScriptu. Všechny tyto kroky se provádějí bez nutnosti obnovení stránky v prohlížeči, což výrazně zrychluje práci s webovou aplikací. (Scott, 2016)

Prezentační logika se v single-page aplikacích přesouvá převážně na stranu klienta, který má za úkol kombinovat HTML kód reprezentující strukturu webu s daty získanými ze serveru. O tento úkol se obvykle stará JavaScript. (Scott, 2016)

V SPA lze k vykreslování dat využívat několik přístupů, přičemž hlavní jsou dva – první spočívá v tom, že fragmenty HTML kódu jsou s daty kombinovány přímo na serveru a server v odpovědi vrací kompletní kód; druhý spočívá v tom, že proces vykreslování se provádí kompletně na straně klienta a při transakcích mezi klientem a serverem se odesílají a přijímají pouze samotná data. Pro výměnu dat v druhém způsobu, který je zároveň v současnosti oblíbenější, se obvykle používá datový formát JSON. (Scott, 2016)

4.3.2 Výhody single-page aplikací oproti tradičním aplikacím

Hlavní výhody webových aplikací jsem již zmínil v kapitole Aplikační software. Nyní shrnu výhody, které je možné podle Scotta (2016) získat při volbě single-page aplikace:

- **Čekání je redukováno na minimum.** Rychlost načítání webové stránky je jeden z nejdůležitějších aspektů, který rozhodne o tom, zda uživatel na stránce zůstane nebo navštíví jinou. U single-page aplikací je jak prvotní načtení, tak následné vykreslování pohledů velmi rychlé, protože se načítá pouze malé množství dat.
- **Rychlá komunikace mezi klientem a serverem.** Protože jsou přenášena pouze samotná data, transakce jsou z pohledu množství dat menší a tím pádem i rychlejší než u tradičních aplikací, kde je celá stránka znovu vykreslena a je přenášen kód celé HTML stránky.
- **Vykreslování bez nutnosti opětovného načtení.** SPA aplikace stahují strukturu webové stránky předem a následně pouze obsah dynamicky mění pomocí pohledů.
- **Lepší schopnost cachování.** Single-page aplikace mohou ukládat do mezipaměti jakákoliv lokální data. Aplikace pouze pošle požadavek na server a následně všechna přijatá data uloží a využívá. Díky tomu je možné zavést offline funkčnost dané aplikace s tím, že synchronizace dat proběhne v okamžiku obnovení připojení.
- **Pohodlný debugging.** Debugging neboli ladění je v těchto aplikacích snazší, protože frameworky jako Angular či React mají své vlastní nástroje, které se instalují přímo do prohlížeče. Ty umožňují zkoumat prvky stránky a data s nimi spojená a také monitorovat síťové operace.

4.4 Architektura a návrhové vzory

Návrhové vzory jsou pojmenovaná a popsaná řešení typického problému spočívajícího v oddělení dat, logiky a výsledné prezentace výstupu, který je vhodné také u webových aplikací řešit. Návrhové vzory vznikly s myšlenkou, že pokud je kód segmentován na základě toho, za co odpovídá, je snazší aplikaci nejen navrhovat a vyvíjet, ale také následně udržovat. (Kouraklis, 2016) V následujících podsekcích jsou představeny jednotlivé základní návrhové vzory. Poslední podsekce se věnuje moderním přístupům k těmto tradičním návrhovým vzorům, což hraje velkou roli při návrhu moderních webových aplikací.

4.4.1 MVC

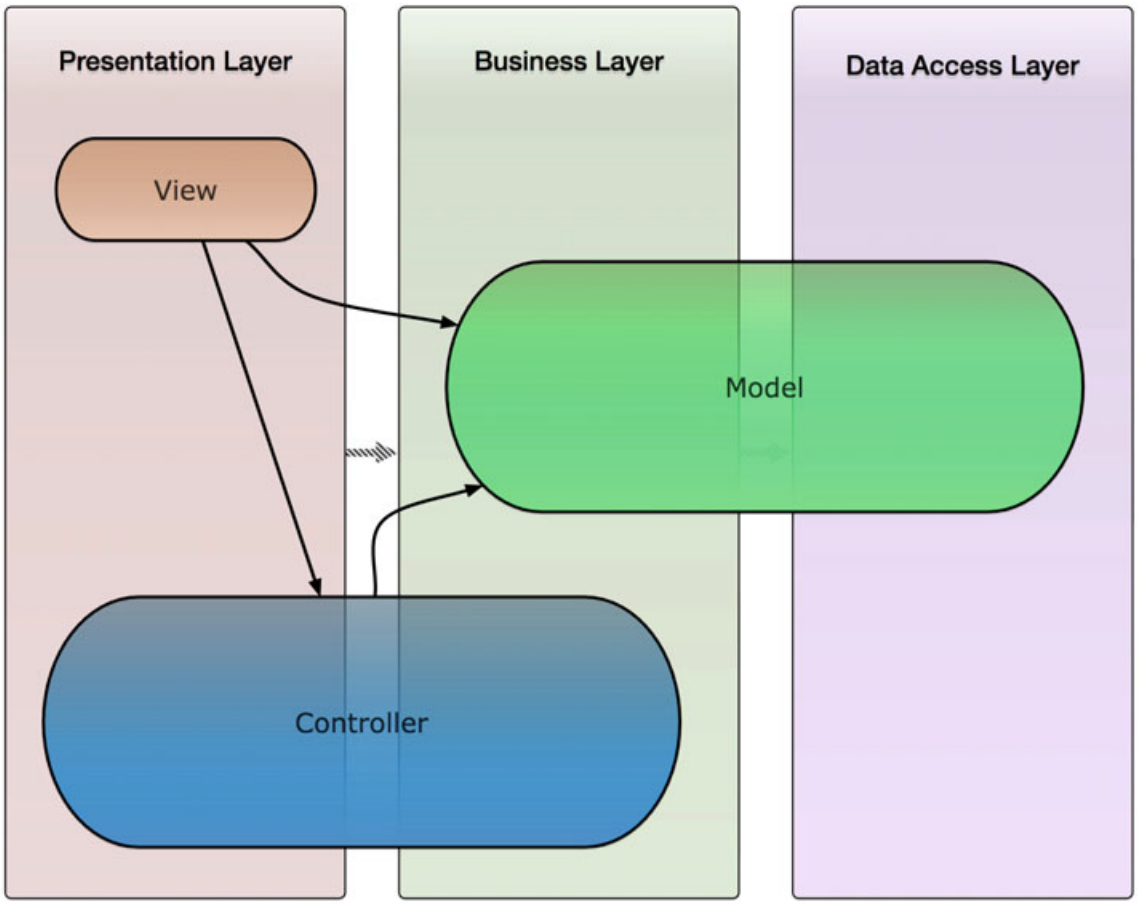
MVC neboli *Model-View-Controller* patří mezi nejstarší známé návrhové vzory, jejichž snahou je oddělit od sebe následující tři vrstvy, a získat tak třívrstvou architekturu:

- **Prezentační vrstva**: uživatelské rozhraní, které vyobrazuje uživateli data, reprezentuje stavy či odlišné formy dat a může se též podílet na kontrole vstupních údajů.
- **Aplikační (business logic) vrstva**: tato část zahrnuje veškerou logiku a funkce aplikace, věnuje se také výpočtům, zpracování dat a jejich validaci.
- **Datová vrstva**: mechanismus spojující aplikaci s databází, ve které jsou uchována data, a zajišťující zpřístupnění a konzistenci dat.

Jak již název napovídá, *M* označuje *Model*, *V* označuje *View* a *C* označuje *Controller*. To jsou tři vrstvy, do kterých tento návrhový vzor rozděluje aplikaci. Význam jednotlivých vrstev je následující:

- **Model** – reprezentuje stav celé aplikace i dat, zahrnuje data, aplikační logiku i validační logiku. Také udržuje veškerou komunikaci s databází. Nikdy se nestará o to, jakým způsobem jsou data prezentována.
- **View** – definuje, co přesně uživatel v aplikaci vidí. Zabývá se také interakcí uživatele s aplikací a vizuální reprezentací dat z modelu.
- **Controller** – důležitý bod, který přijímá signály z prvků uživatelského rozhraní, na základě předdefinované logiky je zpracovává a modelu posílá příkazy k aktualizaci stavu vycházející z obdrženého vstupu.

Při porovnání třívrstvé architektury a komponent návrhového vzoru MVC, lze si na obrázku 7 všimnout toho, že vzájemné mapování není jednoduché, jelikož dochází k výraznému překryvu funkcí a úkolů. (Kouraklis, 2016)

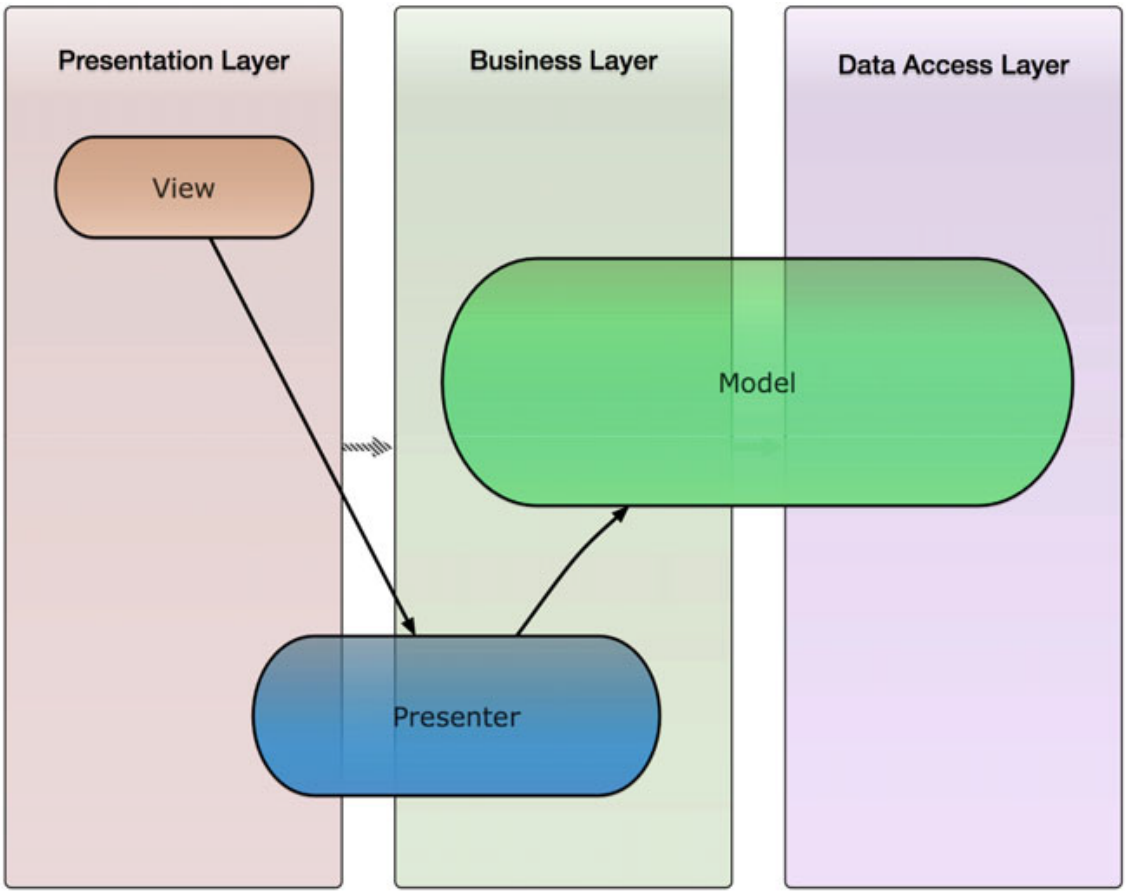


Obrázek 7: Vztah mezi třívrstvou architekturou a návrhovým vzorem MVC
Zdroj: (Kouraklis, 2016)

4.4.2 MVP

MVP neboli *Model-View-Presenter* je návrhový vzor, který vychází z návrhového vzoru MVC a řeší jeho zásadní nedostatky. Komponenta Controller je nahrazena komponentou *Presenter*. Tato nová komponenta přebírá prezentační logiku z komponenty View a zároveň získává přímý přístup ke komponentě Model. View nebere v potaz existenci komponenty Model, což platí i naopak. (Kouraklis, 2016)

Při implementaci aplikace využívající návrhový vzor MVP se obvykle vytváří Presenter jako první, protože přebírá naprostou většinu odpovědnosti. Toho si lze všimnout i na obrázku 8, kde se Presenter v porovnání s Controllerem přesunul hlouběji do aplikační vrstvy. Také uspořádání nabízí především možnost kvalitnějšího testování, protože Model i View lze nahradit při testování falešnými komponentami. (Kouraklis, 2016)



Obrázek 8: Vztah mezi třívrstvou architekturou a návrhovým vzorem MVP
Zdroj: (Kouraklis, 2016)

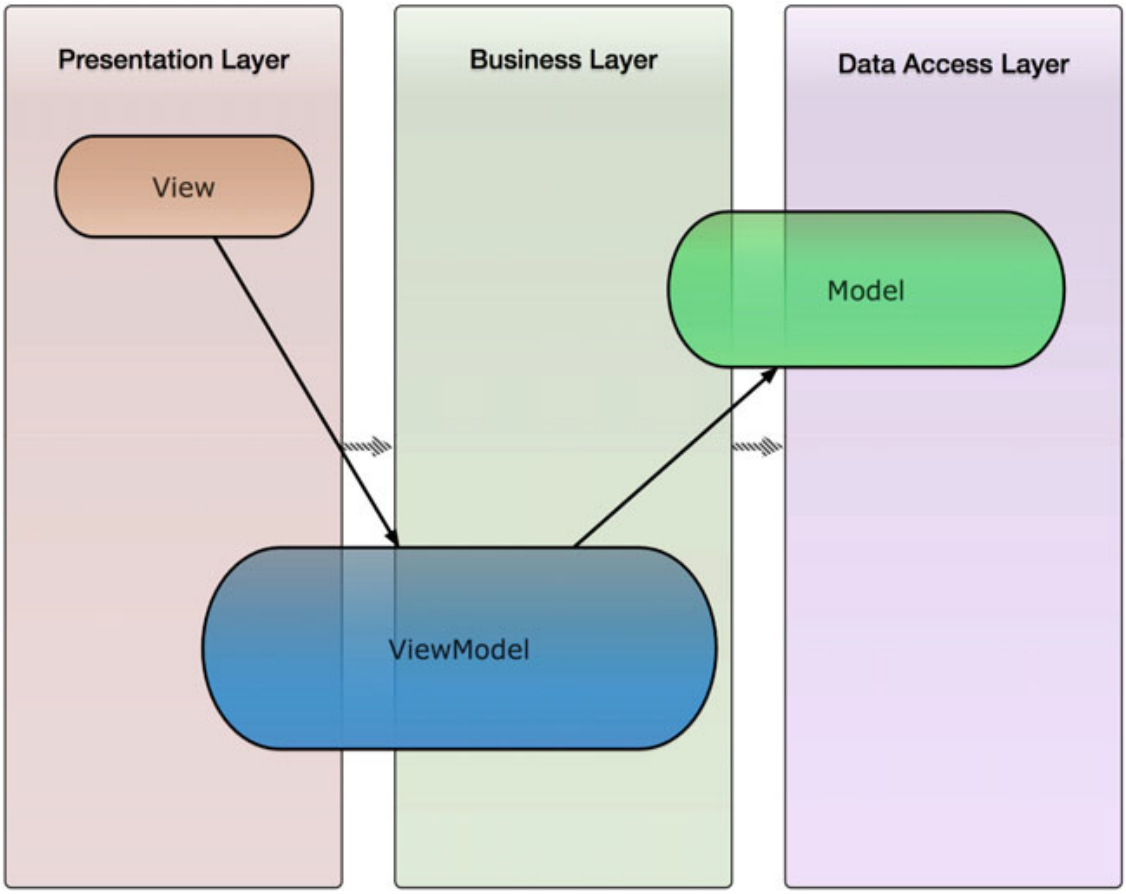
4.4.3 MVVM

MVVM neboli *Model-View-ViewModel* je třetím a posledním vzorem ze skupiny základních návrhových vzorů. Tento návrhový vzor vychází z myšlenek návrhových vzorů MVC i MVP a přináší novou komponentu *ViewModel*, která nahrazuje Presenter i Controller. Tak jako Presenter je i ViewModel prostředníkem mezi komponentami Model i View. Stará se o vlastnosti dat i prezentační logiku, díky čemuž může kompletně definovat a spravovat komponentu View. Všechna data, která mají být libovolným způsobem prezentována v komponentě View, musí být mapována na odpovídající vlastnost v komponentě ViewModel, která má přehled o změnách komponent View i Model a zajišťuje jejich vzájemnou synchronizaci. (Kouraklis, 2016)

Jednou ze silných stránek MVVM je fakt, že umožňuje nejen snadno pochopit to, co dělá která část aplikace, ale i vytvářet spojení mezi zpracovávanými daty a jejich způsobem prezentace uživatelům. (Kouraklis, 2016)

Při porovnání tohoto návrhového vzoru s třívrstvou architekturu, jak je vidět na obrázku 9, se komponenta Model posouvá hlouběji do datové vrstvy, protože se už

z větší části zabývá pouze daty. Částečně se ještě nachází v aplikační vrstvě, protože ta zahrnuje transformaci dat. Nová komponenta ViewModel zasahuje hlouběji do aplikační vrstvy, protože zahrnuje širší škálu úkolů. Část zasahující do prezentační vrstvy zahrnuje logiku a stav zobrazení. (Kouraklis, 2016)



Obrázek 9: Vztah mezi třívrstvou architekturou a návrhovým vzorem MVVM
Zdroj: (Kouraklis, 2016)

4.4.4 Moderní přístup k návrhovým vzorům v souvislosti s webovými aplikacemi

Při vývoji jakékoliv aplikace je vhodné se rozhodnout pro využití některého návrhového vzoru, který může být výrazně nápomocný v přehlednosti kódu i porozumění principu fungování celé aplikace. Výjimkou nejsou ani webové aplikace, ať už multi-page nebo single-page, ovšem zde není výběr tak jednoznačný. Zmíněné tři tradiční návrhové vzory MVC, MVP a MVVM jsou základní vzory, ze kterých je možné při návrhu moderní webové aplikace vycházet, nikoliv je používat v jejich tradiční formě. Jak již bylo zmíněno, webové aplikace jsou aplikace, které ke svému běhu využívají webový prohlížeč, který lze považovat za velmi specifické prostředí využívající tři základní jazyky – HTML, CSS a JavaScript. Aplikovat zmíněné návrhové vzory

v jejich podobě na webové aplikace by nebylo rozumné řešení, proto se v posledních letech začal objevovat pojem *MV** neboli *Model-View-**. (Scott, 2016)

Každý webový framework, který se pro vývoj webových aplikací objevuje, přichází s vlastním řešením architektury, která obvykle vychází z tradičních návrhových vzorů, avšak nedrží se jich striktně. Přestože komponenty Model a View jsou snadno identifikovatelné, s třetí komponentou je to mnohem složitější – mohou využívat Controller, Presenter i ViewModel, avšak také mohou využívat libovolnou hybridní variantu či nemusejí obsahovat třetí komponentu vůbec. Kromě pojmu *MV** se také začal využívat pojem *MVW* neboli *Model-View-Whatever*, který symbolizuje totožnou věc a říká, že třetí komponentou může být naprosto cokoliv. (Scott, 2016)

S tím vším se pojí jedna zásadní myšlenka uplatňovaná vývojáři při vývoji moderních webových aplikací – není důležité umět správně pojmenovat a zdůvodnit návrhový vzor využívaný ve vyvíjené aplikace, mnohem důležitější je umět vhodně navrhnout a následně rozdělit aplikaci takovým způsobem, který je pro danou konkrétní aplikaci co možná nejvhodnější. (Scott, 2016)

Pokud se zaměříme konkrétně na single-page aplikace, koncept často sestává z následujících čtyř částí:

- **Models** – Modely – reprezentují data, se kterými aplikace pracuje, vlastnosti dat, logiku pro přístup k datům, k jejich správě i k ověření jejich validity. Často modely zahrnují také aplikační (business) logiku.
- **Views** – Pohledy – definují, co uživatel v aplikaci vidí, jak s ní interaguje a jak jsou data z modelu vizuálně reprezentována.
- **Templates** – Šablony – jsou znovupoužitelné stavební bloky obsahující zástupné symboly reprezentující data, které se využívají pro zobrazování dynamického obsahu. Mohou také obsahovat upřesňující pokyny pro správné vykreslení v rámci pohledu, který se skládá z jedné či více šablon.
- **Bindings** – Vazby – slouží k přidružení dat z modelu k prvkům v šabloně.

Díky vzájemné komunikaci těchto částí mohou vznikat dynamické pohledy reprezentované šablonami, které jsou závislé na datech prostřednictvím vazeb. (Scott, 2016)

4.5 Technologie využívané pro frontend

Při vývoji frontendu se nabízí velké množství jazyků a knihoven, které je možné využít. Webový prohlížeč ale rozumí pouze několika základním jazykům. Mezi hlavní trojici jazyků určených pro frontend patří HTML, CSS a JavaScript, které společně vykreslují webové stránky do podoby, se kterou se každý den v prostředí internetu lze setkat. Všechny ostatní knihovny a technologie jsou postaveny na těchto třech jazycích a pro vývojáře jsou efektivnějšími nástroji.

HTML neboli *Hypertext Markup Language* je značkovací jazyk, který se používá pro tvorbu webových stránek. Pomocí tohoto jazyka se tvoří základní elementy, ze kterých se web skládá. Sám o sobě není jazyk nikterak inteligentní a pouze umožňuje vystavět základní strukturu webu. (Chheng, 2013)

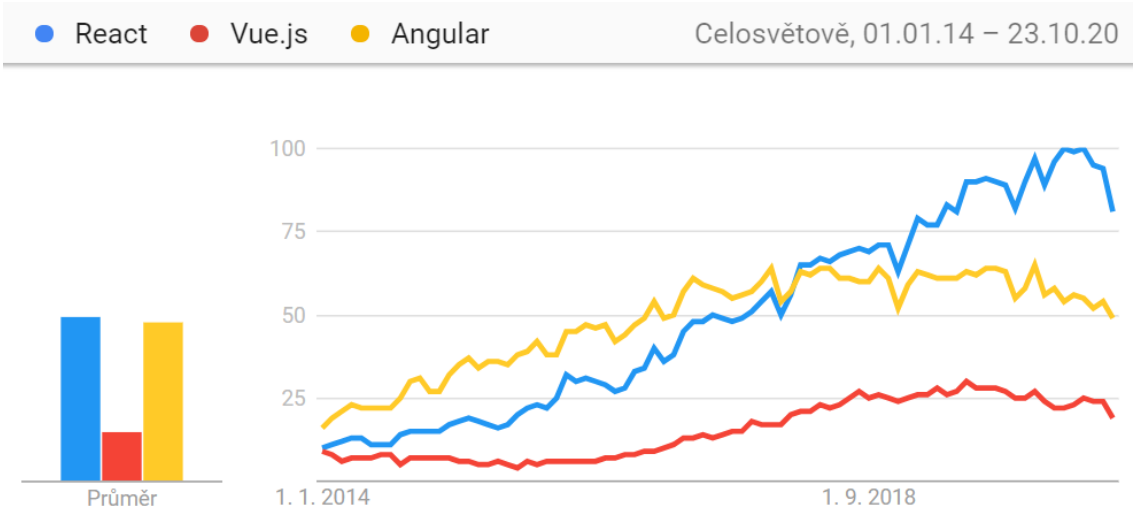
CSS neboli *Cascading Style Sheets* je jazyk, pomocí něhož je možné definovat vizuální podobu webových stránek. Kaskádové styly umožňují upravit výchozí vzhled zahrnující bílé pozadí, černé texty a modré hypertextové odkazy. (Chheng, 2013)

Trendem v oblasti kaskádových stylů jsou tzv. CSS preprocesory. Mezi nejznámější z nich patří **LESS** a **SASS**. Tyto preprocesory jsou skriptovací jazyky, které kompilují zdrojové kódy v nich napsané do běžného CSS, kterému webový prohlížeč rozumí. V posledních letech jsou velmi oblíbené především proto, že urychlují proces vývoje a poskytují vývojářům možnost využít základní programovací logiku přímo v kaskádových stylech.

JavaScript je skriptovací jazyk, který umožňuje webu přidat dynamické chování a interaktivní prvky. Na rozdíl od programovacích jazyků běží na straně klienta, tedy přímo v prohlížeči. Pomocí objektového modelu dokumentu *DOM* přistupuje k jednotlivým prvkům HTML dokumentu, se kterými může téměř libovolně nakládat. Populární je knihovna *jQuery*, která dokáže s využitím podstatně kratšího kódu realizovat totožné činnosti jako čistý JavaScript. (McFarland, 2014)

S ohledem na potřeby vytvářet stále komplexnější webové aplikace se dostaly do popředí javascriptové frameworky, jejichž cílem je usnadnění vývoje webových aplikací. Frameworky mají mnoho integrovaných funkcí i knihoven třetích stran, které mohou vývojáři využívat. Výsledkem toho celého je, že vývojáři nemusí řešit mnoho problémů, které za ně již vyřešil někdo jiný. Vývojový proces se zrychlí a kvalita kódu vzroste.

Mezi nejpopulárnější frontendové frameworky bezpochyby patří *React*, *Angular* a *Vue.js*. Je velmi těžké odhadovat, který z nich je v současnosti nejoblíbenější, avšak něco může napovědět výstup ze služby Google Trends, která analyzuje popularitu vyhledávacích dotazů. Jak je možné vidět na grafu, Angular až do počátku roku 2018 byl mezi uživateli nejoblíbenější, avšak v lednu roku 2018 nastal obrat a React jej v oblíbenosti významně překonal a i na konci roku 2020 si stále drží první příčku.



Obrázek 10: Google Trends

4.5.1 React

React je v současnosti považován za nejoblíbenější open-source javascriptový framework, který se využívá k vytváření uživatelských rozhraní. Za vývojem tohoto frameworku stojí firma Facebook a rozsáhlá komunita samostatných vývojářů i celých společností. React se obvykle používá při vytváření single-page aplikací či mobilních aplikací. Právě díky možnosti vytváření znovupoužitelných komponent se jedná o robustní frontendový framework, který nachází využití při vytváření atraktivních a interaktivních rozhraní za krátkou dobu. (Uzayr, Cloud, Ambler, 2019)

Nejprve bych rád zmínil výhody, které React nabízí a díky kterým se vyplatí přemýšlet o jeho zvolení pro vlastní projekt:

- **Uživatelský komfort díky VDOM.** VDOM vychází z DOM neboli *Document Object Model*, což je objektově orientovaná reprezentace HTML nebo XML dokumentu. V podstatě se jedná o API umožňující přístup ke struktuře dokumentu nebo jeho částí včetně modifikace. VDOM neboli *Virtual DOM* je programovací koncept, ve kterém je virtuální reprezentace uživatelského rozhraní uchovávána v paměti a synchronizována se skutečným DOM pomocí speciální knihovny ReactDOM. To umožňuje vývojáři, aby „oznámil“ Reactu, v jakém stavu má být uživatelské rozhraní, a knihovna zajistí, aby se DOM s tímto stavem shodoval. Výrazně se tak zjednoduší a abstrahuje manipulace s atributy, událostmi a ručními úpravami DOM.

- **Opětovné používání komponent.** Koncepčně jsou komponenty podobné funkcím v JavaScriptu – přijímají libovolné vstupy a vracejí elementy popisující to, co se má objevit na obrazovce. Komponenty umožňují rozdělit uživatelské rozhraní na nezávislé a opakovaně použitelné části, přičemž o každé této části je možné přemýšlet jako o izolované jednotce. Právě možnost opětovného použití

komponent a jednoduchost jejich upgradu výrazně šetří vývojářům čas.

- **Snadno naučitelný framework.** React je poměrně snadné se naučit a začít jej používat. Tvorba webových aplikací je s ním mnohem jednodušší i díky tomu, že nabízí vývojáři obrovskou flexibilitu při výběru toho, co skutečně ke své práci potřebuje.
- **Poskytuje velké množství užitečných nástrojů.** Pod názvem React se neskrývá pouze samotný framework, ale mnoho doplňujících nástrojů, které tomuto frameworku dávají přidanou hodnotu. Mezi nejznámější nástroje patří *Create React App* poskytující kompletní sestavení nového projektu a nastavení vývojářského prostředí s cílem optimalizovat produkci aplikací a ušetřit vývojářům čas strávený nad konfigurací, *Jest* jakožto nástroj vytvořený za účelem testování komponent vytvořených v Reactu, *React Toolbox* představující kolekci několika desítek často používaných komponent navržených pomocí konceptu Material Design, *React Bootstrap* nabízející balíček všech komponent z nejznámější frontendové sady nástrojů Bootstrap přepracovaný pro React či *React Studio* pro vytváření komponent v Reactu na základě grafických návrhů bez nutnosti pokročilé znalosti programování.

Důležité je zmínit i některé nevýhody, které React má. Ačkoliv je relativně snadné naučit se jeho základy, kvůli jeho robustnosti je **velmi zdlouhavé naučit se jej důkladně** a porozumět všem funkcím, které nabízí. Druhou nevýhodou je **nepropracovaná dokumentace** způsobená především vysokým tempem vývoje. (Uzayr, Cloud, Ambler, 2019)

Pro lepší představivost, jakých výsledků je možné s Reactem dosáhnout, uvádím webové aplikace, které jsou na něm postavené. Mezi ty nejznámější patří sociální sítě Facebook a Instagram, streamovací platforma Netflix, cloudové úložiště Dropbox či služba zprostředkovávající pronájem ubytování Airbnb.

4.5.2 Angular

Angular je frontendový framework, který se využívá pro vývoj webových aplikací. Za jeho vývojem stojí firma Google. Právě kvůli vývojovému týmu z Googlu se tento framework stal velmi rychle populární a až do roku 2018 neměl v oblíbenosti konkurenci. Pro úplnost informací je třeba si uvést, že existují dva frameworky – Angular a AngularJS. Jedná se o dva odlišné frameworky postavené na dvou odlišných, ačkoliv poměrně úzce souvisejících platformách. Zatímco AngularJS využívá návrhový vzor *Model-View-Controller* a neumožňuje vytvářet mobilní aplikace, Angular využívá komponenty a je vhodný k vývoji mobilních aplikací. (Uzayr, Cloud, Ambler, 2019)

I když dnes již není Angular nejpopulárnější, stále je na něm postaveno velké množství projektů. Mezi hlavní výhody Angularu patří:

- **Základním jazykem je TypeScript.** TypeScript je nástavba nad jazykem JavaScript, která jej rozšiřuje o statické typování a další atributy známé z objektově orientovaného programování. TypeScipt je následně kompilován do JavaScriptu.
- **Architektura založená na komponentách.** Tak jako React je i Angular založený na práci s komponentami, které přijímají vstupy a vracejí elementy v odpovídající podobě. Díky tomu je možné rozdělit rozhraní na jednotlivé části, které lze libovolně používat.
- **Jednoduchý a rychlý data binding.** Pojmem *data binding* se označuje závislost logiky či stavu proměnné v modelu na vykreslení aplikace. Jakmile se změní stav nějaké proměnné, DOM se musí aktualizovat. Angular nabízí jednosměrný i obousměrný data binding.

Mezi nevýhody frameworku Angular patří bezpochyby **obtížnost porozumění a naučení**, což často odrazuje vývojáře od tohoto frameworku. I často jednoduché úkony se v Angularu provádějí velmi složitě a kvůli tomu produktivita značně klesá. Se zmíněnou složitou strukturou souvisí také to, že Angular **není vhodný pro vytváření rozsáhlých projektů**. (Uzayr, Cloud, Ambler, 2019)

Mezi známé projekty, které využívají Angular, patří například e-mailová služba Gmail, magazín Forbes či internetový platební systém PayPal.

4.5.3 Vue.js

Vue.js byl vytvořen v roce 2014 jako open-source javascriptový framework, který slouží k vytváření uživatelských rozhraní a single-page aplikací. Z trojice React, Angular a Vue.js se jedná o framework, který má nejvíce adaptivní architekturu. Za jeho vytvořením stojí softwarový vývojář Evan You a nyní se na dalším vývoji podílejí týmy z různých společností. Mezi hlavní výhody tohoto frameworku mimo jiné patří: (Uzayr, Cloud, Ambler, 2019)

- **Zcela podporuje TypeScript.** Vývojáři mohou pro pohodlnější vývoj využít místo JavaScriptu jeho nástavbu TypeScript.
- **Velmi jednoduchý k naučení.** Oproti výše zmíněným frameworkům je Vue.js bezpochyby nejsnazší k naučení, a to právě díky jeho jednoduché syntaxi, které lze velmi rychle porozumět.
- **Podrobná dokumentace.** Rozsáhlou dokumentací se může Vue.js pyšnit, oproti předchozím frameworkům nabízí detailní dokumentaci včetně mnoha vzorových příkladů, které pomohou k rychlejšímu naučení.

Vzhledem k tomu, že je Vue.js poměrně mladým frameworkem, zásadní nevýhodou je **velmi malá komunita** uživatelů. Není možné se také spoléhat na rozsáhlé množství komponent, protože Vue.js **nabízí pouze minimum pluginů a komponent**, u kterých často ani není zajištěna stabilita. (Uzayr, Cloud, Ambler, 2019)

4.6 Technologie využívané pro backend

Zatímco srdcem frontendu je kombinace tří jazyků, na backendu mají vývojáři volnější ruce při výběru. Šestice nejpopulárnějších jazyků určených pro vývoj backendu webových aplikací zahrnuje jazyky Java, PHP, JavaScript, Ruby, Python a .NET.

V rámci backendu vývojář řeší to, co není vidět, respektive to, co se děje v zákulisí – API, webové služby, implementace a komunikace s databázovým systémem a mnoho dalšího. Tak jako se při vývoji frontendu webových aplikací stále častěji využívají frameworky, jinak tomu není ani u backendu. S velkým množstvím jazyků vhodných pro backend souvisí také velké množství frameworků, které pro tyto jazyky vznikají. V následujících podsekcích jsou zmíněny ty z nich, které jsou v současnosti nejoblíbenější a nejpoužívanější.

4.6.1 Express

Express je rychlý a minimalistický framework postavený na prostředí Node.js, které slouží pro vykonávání kódu napsaného v jazyce JavaScript mimo prostředí webového prohlížeče. Tento framework využívá silný výkon Node.js, díky čemuž mezi jeho hlavní přednosti patří především rychlost. Dalšími důvody, proč zvolit pro backend právě Express je jeho flexibilita a plná podpora pro vytvoření REST API. Jeho základní zprovoznění a následné pochopení je pro vývojáře poměrně snadné, ačkoliv z oficiální dokumentace je často složité zjistit, které postupy jsou správné. Express také nabízí sadu již předpřipravených modulů, které zjednodušují ty nejčastější činnosti potřebné na backendu, jako je například parsování HTTP požadavků či autentizace prostřednictvím účtů na sociálních sítích a populárních službách. Ve svých projektech tento framework využívají společnosti jako IBM či Uber. (David, 2020) (Goel, 2020)

4.6.2 Django

Django je framework postavený na jazyce Python, který je možné použít nejen na backend, ale i na frontend. Nabízí velké množství funkcí a nástrojů, přičemž jeho hlavním benefitem je Django ORM neboli *Object-Relation Mapper* – nástroj určený pro interakci s databází, umožňuje vytváření SQL dotazů, manipulaci s databází i získávání výsledků. Nástroje a balíky tohoto frameworku pokrývají mnoho oblastí jako například autentizaci, správu session, generování sitemapy či vytváření e-shopu. Rozsáhlá komunita, vysoká přizpůsobivost a snadné testování kódu patří k dalším pozitivům tohoto frameworku. Vývojáře může ale odradit, že je Django určen spíše pro větší projekty a může zahrnovat příliš funkcí, které ani nebudou v projektu využity. Také vzniklé chyby nejsou ideálně interpretované, a proto jejich odhalení může stát mnoho času. Tento framework si zakládá na tom, že existuje jeden určitý způsob, jak nějakou věc provést, a tím by se měl vývojář řídit. Je tak vhodný především pro vývojáře, pro které je Python jejich nejoblíbenější programovací jazyk.

Django se ve svých projektech rozhodly využít společnosti jako Google, YouTube či Instagram. (David, 2020) (Goel, 2020)

4.6.3 Laravel

Dalším z oblíbených backendových frameworků je Laravel, který používá jazyk PHP, jeden z nejpopulárnějších jazyků ve webovém prostředí, a z pohledu architektury využívá návrhový vzor MVC. Laravel se pyšní především snadnou tvorbou API a dostatečným množstvím předpřipravených balíků usnadňujících práci, avšak naproti tomu stojí zásadní nevýhoda – z pohledu výkonnosti se nemůže Laravel ani zdaleka vyrovnat Djangu nebo Expressu. A právě tento faktor může být velmi limitující především ve větších projektech. (Goel, 2020)

4.6.4 Rails

Rails nebo také Ruby on Rails je backendový framework postavený na jazyce Ruby. Taktéž využívá z pohledu architektury návrhový vzor MVC. Rails je považován za framework vhodný pro začátečníky, kteří se chtějí naučit princip tvorby backendu, avšak při snaze o hlubší porozumění roste jeho obtížnost i náročnost celého vývoje neúměrně rychle. Ve svých projektech Rails využívá například GitHub nebo Airbnb. (Goel, 2020)

4.6.5 Spring

Spring je dalším z backendových frameworků, které v současnosti patří mezi oblíbené. Z pohledu architektury využívá návrhový vzor MVC a je postavený na jazyce Java. Výkonnost a škálovatelnost patří mezi hlavní vlastnosti, které Spring přináší, a to i díky tomu, že je postavený právě na programovacím jazyce Java, který je silně typovaný. Na druhou stranu se Java řadí k složitějším programovacím jazykům a tento framework je použitelný především pro profesionální práci vývojářů, kteří mají s Javou bohaté zkušenosti. Společnosti jako Wix nebo TickerMaster se rozhodly tento framework využít ve svých projektech. (Goel, 2020)

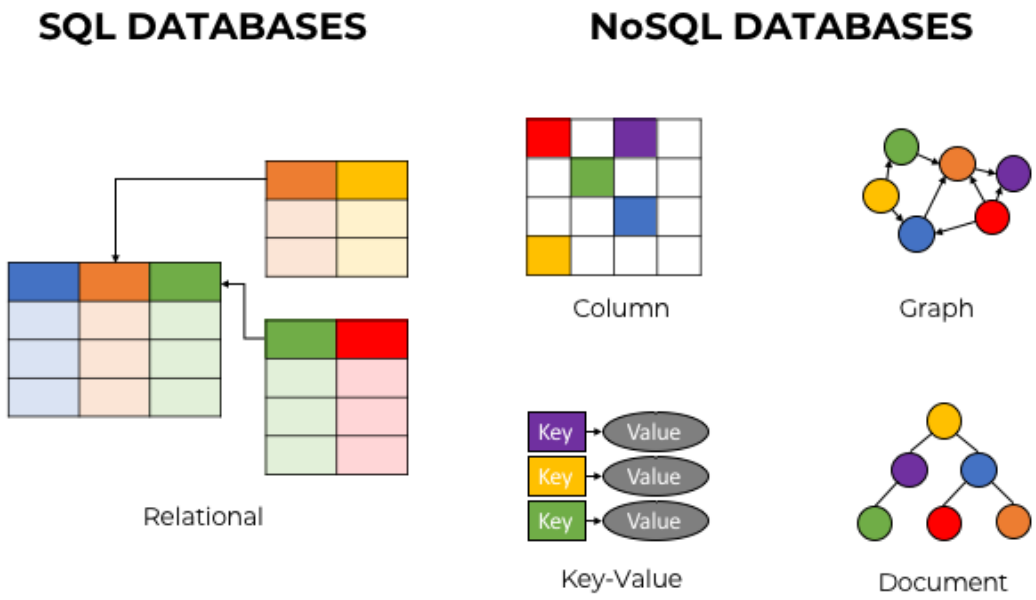
4.7 Databázové systémy

Databázový systém je systém pro uchovávání záznamů, jehož hlavním úkolem je udržování a zpřístupňování informací, a to kdykoliv, kdy je o to požádáno. Záznamy mají podobu organizovaných souborů obsahujících strukturované informace a nazýváme je *databáze*. Databáze jsou obvykle řízeny *systémem pro správu databáze* – jedná se o sadu programů, které umožňují správu databáze zahrnující definici dat (vytváření relací a závislostí, omezování integrity), manipulaci s daty či kontrolu zabezpečení a integrity dat. (Foster, Godbole, 2014)

A jaké výhody databázový systém přináší? Napomáhá vyhnout se redundanci a nekonzistentním datům, umožňuje aplikovat bezpečnostní omezení, poskytuje

vysoký výkon díky rychlosti zpracování požadavků, není závislý na zvoleném programovacím jazyce, nabízí jednoduchý způsob získávání dat a dovoluje měnit strukturu záznamů bez ztráty dat. (Foster, Godbole, 2014)

4.7.1 Základní typy databází



Obrázek 11: Přehled možností struktur relačních a nerelačních databází
Zdroj: https://openclassrooms.com/en/courses/5671741-design-the-logical-model-of--your-relational-database/6255746-compare-relational-and-nosql-databases

Foster a Godbole (2014) a Tobin (2020) rozlišují podle způsobu ukládání dat a vazeb mezi nimi několik základních typů databází:

- **Relační databáze**. Relační neboli SQL databáze využívají jazyk SQL, který nabízí vysoce organizovaný a strukturovaný přístup k informacím. Strukturu relačních databází je možné si představit jako adresář kontaktů v tabulce, kde každá osoba bude na samostatném řádku a sloupce budou obsahovat jednotlivé vlastnosti, které o daných osobách lze evidovat – např. jméno, příjmení, telefonní číslo či e-mailovou adresu.
- **Nerelační databáze**. Nerelační neboli NoSQL databáze umožňují uspořádat informace mnohem volnějším způsobem, proto je do nich možné ukládat libovolná nestrukturovaná data. Tento typ databází pracuje s formáty jako JSON, který je nezbytný pro moderní webové aplikace. V nerelačních databázích neexistuje žádné jasně definované schéma, které by bylo nutné dodržet. Strukturu nerelačních databází je možné si představit jako složky, do kterých lze ukládat různé soubory.

- **Objektové databáze**. V objektových databázích jsou informace uspořádané formou objektů stejně jako v objektově orientovaných programovacích jazycích. Objektové databáze se začaly objevovat s rostoucím zájmem o objektově orientované programování. Objektově orientovaný návrh přináší výhody jako zapouzdření, dědičnost, polymorfismus či reference mezi objekty. V závislosti na konkrétní situaci může, ale i nemusí objektově orientovaný návrh databáze přinést výrazné výhody.

4.7.2 Rozhodující faktory při výběru databáze

U webových aplikací se nejčastěji rozhoduje mezi SQL a NoSQL databází. Při výběru je důležité se zamyslet nad tím, jak budou vypadat data, která budou v aplikaci využívána, jakým způsobem se k nim bude přistupovat, v jaké formě se je bude chtít získávat a jaké jsou požadavky na škálovatelnost. Jednotlivé faktory podrobněji rozeberu v následujících odstavcích.

Zásadním faktorem při rozhodování je **struktura dat**, tedy to, jak budou data vypadat. Pokud jsou data v zásadě strukturovaná, pak bude vhodnou volbou SQL databáze. V SQL databázi se využívají tabulky, které mají definovanou strukturu. Jednotlivé řádky vložené do tabulek se nazývají záznamy. Tyto záznamy musí striktně splňovat strukturu tabulky. Pokud jsou ale data nestrukturovaná, je vhodnější využít NoSQL databázi. V NoSQL databázi se používají kolekce, do kterých se ukládají jednotlivé dokumenty. Dokumenty v rámci kolekce nemají (ale mohou mít) pevně definovanou strukturu, čímž poskytují potřebnou flexibilitu. Vlastnosti databázových transakcí jsou definovány v případě SQL pod zkratkou ACID a v případě NoSQL pod zkratkou BASE. Zkratka ACID skrývá následující vlastnosti – atomicita, konzistence, izolace a trvalost. Pod zkratkou BASE se skrývají vlastnosti jako „v podstatě dostupný, změkčení stavu, případně konzistentní“. Zjednodušeně řečeno, BASE je opakem ACID a poskytuje maximální míru flexibility. (Sharma, Dave, 2012)

Dalším faktorem je **dotazování se na data**. Je třeba se zamyslet nad tím, jak často se bude na data v databázi dotazovat a jak rychle je potřebné je získat. Vzhledem k tomu, že SQL pracuje se strukturovanými daty, jejich získání je rychlejší než v případě NoSQL. V případě dotazů na NoSQL databázi je často potřeba provést další zpracování dat v aplikační vrstvě.

Škálovatelnost je faktor, nad kterým je vhodné se také zamyslet. SQL databáze jsou vertikálně škálovatelné, čímž se rozumí potřeba hardwarového vylepšení serveru, na kterém databáze běží (tj. CPU, RAM či SSD). Jsou navržené tak, aby fungovaly na jednom serveru, a to proto, aby byla zachována integrita dat. Jejich škálování proto není snadné. NoSQL databáze přinášejí výhodu v podobě horizontální škálovatelnosti, což znamená, že je možné přidat více serverů, na kterých bude databáze umístěna. Vertikální škálování může být za určitých situací již velmi finančně nákladné, a proto se v praxi často přistupuje k horizontálnímu škálování. (Sharma, Dave, 2012)

4.7.3 Přehled konkrétních databází

Jak oblast SQL, tak oblast NoSQL zahrnuje široké množství konkrétních databází, ze kterých je možné si vybírat.

V kategorii SQL databází je nejpopulárnější open-source relační databáze *MySQL* a její vylepšená verze *MariaDB*, dále například *Microsoft SQL server* od společnosti Microsoft, podniková databáze *PostgreSQL* či nákladnější řešení od společnosti *Oracle*.

Jako zástupce v kategorii NoSQL databází lze zmínit *MongoDB*, v současnosti nejpopulárnější NoSQL databázi, dále open-source databázi *Redis* či databázi *Cassandra*, za jejímž vývojem stojí společnost Facebook.

Cloudové služby AWS, Microsoft Azure a Google Cloud Platform nabízejí taktéž široký výběr databází. Z kategorie SQL jsou to například *RDS*, *Aurora*, *Azure SQL Database* či *Cloud SQL*. Z kategorie NoSQL je možné zmínit například *DynamoDB*, *DocumentDB*, *CosmosDB* či populární *Cloud Firestore* od společnosti Google.

5 Analýza dostupných řešení

Tato kapitola se zabývá konkrétními dostupnými řešeními pro tvorbu online testů, které mimo jiné doporučuje i Ministerstvo školství, mládeže a tělovýchovy ve svém Metodickém doporučení pro vzdělávání distančním způsobem. Ke každému řešení jsou uvedeny nejen základní informace o dané aplikaci a mé osobní názory, ke kterým jsem během testování jednotlivých aplikací dospěl, ale také nejdůležitější připomínky vyučujících, kteří s aplikacemi v praxi pracují.

5.1 Kahoot

Aplikace Kahoot[11] sama o sobě hovoří jako o aplikaci pro vytváření, hraní a sdílení výukových her (tzv. kahootů) s přáteli a rodinou. Nabízí několik placených variant v rozmezí 100 Kč až 300 Kč za měsíc, ale i variantu zcela zdarma. Ta má ale mnoho omezení jako například maximálně 10 hráčů a velmi omezená nabídka typů otázek. Aplikace nabízí databázi vytvořených kahootů, které je možné okamžitě hrát.

Prvním viditelným omezením pro potřeby českého školství je **absence českého jazyka**. Celá aplikace je dostupná pouze v angličtině, ať už se jedná o rozhraní pro učitele nebo žáky. Používání aplikace může být kvůli tomu limitující nejen pro vyučující, kteří neovládají anglický jazyk, ale také především pro mladší žáky.

Rozhraní pro vytváření nového kahootu se tváří jako aplikace PowerPoint, kde každý slajd je jedna otázka. U každé otázky je možné vložit její název, nastavit časový limit, zvolit počet bodů za správnou odpověď, vybrat typ otázky, přidat obrázek a nastavit možnosti. Celé rozhraní **působí dětským dojmem**. Po vytvoření a uložení kahootu se vygeneruje PIN kód, který může hráč vložit do své herní aplikace. Po zadání přezdívky se čeká na spuštění kvízu učitelem. Následně učitel ve svém rozhraní posouvá otázky a vidí odpovědi žáků. Žáci vidí v daném okamžiku otázku a mají časový limit na její zodpovězení. Poměrně nepříjemná je **podkresová hudba**, která během celého běhu aplikace hraje a není možné ji vypnout. Do toho aplikace vydává spoustu dalších zvuků téměř při každé akci, která se na obrazovce uskuteční.

Odpovědi žáků je možné zobrazit zpětně po dokončení testu, avšak ve verzi zdarma je možné zobrazit pouze 5 nejlepších výkonů, ty zbylé jsou skryté. Pokud si učitel již během testu neznačí úspěšnost svých žáků někde mimo aplikaci, k **souhrnným výsledkům se nedostane**. To je velmi omezující a vytváří to dojem nepoužitelnosti verze zdarma pro jiné účely než testování a hraní si, při kterém učitele nezajímá, jak hráči (resp. žáci) dopadnou.

Stručně je možné říci, že aplikace je nástrojem pro tvorbu zábavných kvízů, která najde zalíbení především u menších dětí. Je vhodná spíše k průběžnému procvičování nebo ozvláštnění vyučovací hodiny, avšak adekvátní náhradou za písemné zkoušky být nedokáže. Verze zdarma je velmi limitující a pro větší počet žáků nepoužitelná. Využití najde především v malých skupinách.

[11] https://kahoot.com/

5.1.1 Připomínky vyučujících k aplikaci Kahoot

K aplikaci Kahoot se nejčastěji objevují následující připomínky:

- při použití existujícího kahootu je pro žáky velmi snadné nalezení správných odpovědí,
- složité nastavení toho, aby žáci hráli sami bez asistence vyučujícího,
- počet znaků v otázce je omezený,
- nutné kvalitní připojení, jinak se aplikace nenačte správně a není možné v určitém okamžiku během testu pokračovat.

5.2 Socrative

Aplikace Socrative[12] je definovaná jako nástroj pro formativní hodnocení, jehož cílem je poskytovat zábavné a účinné nástroje pro testování znalostí studentů v reálném čase. V současné době jej po celém světě používají více než 3 miliony uživatelů a je k dispozici ve 14 jazycích.

Bohužel ani Socrative nepřekvapil a **absence češtiny** je prvním omezením, kterého si uživatel po příchodu na webové stránky všimne.

Z pohledu žáka probíhá testování tak, že žák vyplní název místnosti, který mu poskytne vyučující. Následně zadá své jméno a připojí se do místnosti. Jako výhodné se může na první pohled zdát, že žák neabsolvuje žádnou registraci ani nemá žádný účet, avšak ve skutečnosti hrozí, že si **žáci mezi sebou vymění jména** a budou vyplňovat test za někoho jiného.

Po připojení do místnosti se spustí test. Mezi otázkami je možné libovolně přepínat, je možné je vyplňovat v libovolném pořadí a každá otázka se vždy nachází na samostatné stránce. Na konci se pomocí tlačítka test ukončí a odešle vyučujícímu.

Z pohledu vyučujícího je k dispozici varianta účtu zdarma, která má hlavní omezení v podobě maximálně 50 žáků v jedné místnosti, nebo placená varianta účtu, která nabízí vyšší počet žáků a další vylepšené funkce. Placená varianta se pohybuje od 100 Kč do 175 Kč za měsíc. Po zaregistrování se vyučujícímu vytvoří místnost s jedinečným identifikátorem, který je složen z jeho příjmení a číslic. Tento identifikátor zadávají žáci při připojování se k testu a vyučující jej může libovolně změnit. Problém je, že **identifikátor místnosti musí být jedinečný** v rámci celé aplikace, proto může být velmi složité najít vhodný identifikátor, který ještě není zabraný a zároveň nebude příliš dlouhý a pro žáka pochopitelný.

Po kliknutí na tlačítko Create Quiz zadá vyučující název kvízu a následně pomocí tlačítek přidává jednotlivé druhy otázek. Na výběr je z otázek otevřených, otázek pravda/nepravda a otázek na výběr možností. U uzavřených i otevřených otázek je možné určit správnou odpověď, která se poté automaticky vyhodnotí. Vyhodnocení správnosti otevřené otázky probíhá tak, že vyučující může zapsat jednu

[12] https://www.socrative.com/

či více správných variant. Pokud ale student například zapíše odpověď na začátku s velkým písmenem, zapomene tečku, zadá více mezer a podobně, odpověď bude vyhodnocena jako nesprávná. Aplikace zcela **postrádá možnost revize automatického opravení** ze strany vyučujícího. Pokud aplikace vyhodnotí odpověď jako nesprávnou a vyučující by ji chtěl uznat, nemá žádnou možnost, jak to učinit.

Aplikace Socrative se myšlence vytváření online testů pro žáky velmi přibližuje. Má jednoduché a poměrně přehledné rozhraní. Verze zdarma nabízí dostatečné množství funkcí pro možnost reálného využití aplikace. Přesto se aplikace hodí více k tvorbě testů s uzavřenými odpověďmi.

5.2.1 Připomínky vyučujících k aplikaci Socrative

K aplikaci Socrative se nejčastěji objevují následující připomínky:

- neumožňuje používat jiné matematické symboly než horní a dolní index,
- neposkytuje možnost revize automatického opravení a možnost úpravy bodů,
- není možnost náhodného vygenerování menšího počtu otázek z vytvořené větší sady otázek (aby nebyly mezi žáky otázky pouze promíchané, ale některé zcela odlišné),
- souběžně může běžet pouze jedna aktivita (pro každou aktivitu je potřeba vytvořit samostatný profil).

5.3 Microsoft Forms

Microsoft Forms[13] je aplikace z balíčku Office 365, která se skládá ze dvou základních částí – formulářů a kvízů. Prostředí pro vytváření formulářů a kvízů jsou víceméně shodná, část týkající se kvízů nabízí některé funkcionality navíc. Hlavní výhodou této aplikace je fakt, že je **zcela zdarma bez jakýchkoliv limitů na používání** a je kompletně v české lokalizaci. Existuje také její varianta Forms Pro, která pouze rozšiřuje produkt o ankety, pokročilé analýzy, pokročilé sdílení a personalizaci.

Vytváření nového testu začíná kliknutím na tlačítko Nový kvíz. Otevře se okno, ve kterém lze zadat název a popisek kvízu a začít vkládat otázky. Na výběr jsou uzavřené i otevřené otázky několika různých typů. U každé otázky je možné určit, zda je její vyplnění povinné a kolik bodů žák dostane za její správné zodpovězení. U otázky je možné nastavit náhodné zamíchání odpovědí. Aplikace také nabízí speciální modul pro matematiku, pomocí kterého je možné do otázek vkládat matematické symboly.

Uživatelské rozhraní například v porovnání s aplikací Socrative **působí méně přehledným dojmem**. Obzvlášť matoucí je, že nastavení celého kvízu je ukryté pod třemi tečkami v horizontálním panelu, pod nimiž se teprve nachází rozbalovací menu, ve kterém je vidět položka Nastavení. V nastavení jsou přitom důležité funkce

[13] https://forms.office.com/

jako automatické zobrazení výsledků po odeslání kvízu, omezení uživatelů, kteří mohou kvíz vyplnit, nastavení počátečního a koncového data či aktivace náhodného pořadí otázek.

Vzhledem k tomu, že se jedná o aplikaci společnosti Microsoft, pod kterou patří i v současnosti hojně používaná aplikace Teams, je možné přímo do konkrétního týmu přidat již vytvořený formulář namísto posílání odkazu k vyplnění.

Ačkoliv je možné použít vyhodnocení automatické, vyučující může procházet vyplněné kvízy jednotlivých žáků, prohlédnout si jejich odpovědi, upravit bodové hodnocení a připsat slovní hodnocení. Po publikování skóre si může žák zpětně prohlédnout své výsledky.

5.3.1 Připomínky vyučujících k aplikaci Microsoft Forms

K aplikaci Microsoft Forms se nejčastěji objevují následující připomínky:

- nelze nastavit časový limit, který má žák na vyplnění testu, ale pouze počáteční a koncové datum,
- při sdílení přes odkaz se nezaznamená jméno žáka a je nutné pro něj vytvořit kolonku,
- není možnost náhodného vygenerování menšího počtu otázek z vytvořené větší sady otázek (aby nebyly mezi žáky otázky pouze promíchané, ale některé zcela odlišné),
- při sdílení přes odkaz nelze zamezit tomu, aby žáci vyplnili test vícekrát,
- ověření správné odpovědi u otevřené otázky nerozlišuje malá a velká písmena (může být potřebné v některých předmětech),
- neumožňuje propojení s hodnotícími kritérii a automatické generování známky (vrací pouze počet bodů).

5.4 Google Forms

Google Forms[14] je aplikace pro vytváření formulářů a kvízů. Je součástí bezplatné webové sady Google Docs, plně **v české lokalizaci** a její funkcionality nejsou nijak limitovány.

Aplikace Google Forms a Microsoft Forms jsou velmi podobné. V obou případech má tvůrce kvízu na výběr z několika typů otázek, kvíz je možné sdílet přímým odkazem či zasláním emailu (v případě Microsoft Forms ještě přes QR kód). Google Forms mají **lepší podporu nahrávání médií** do kvízů a **pohodlnější způsob organizace otázek**. Z hlediska nabízených funkcionalit se jedná o téměř totožné produkty.

[14] https://www.google.com/forms/

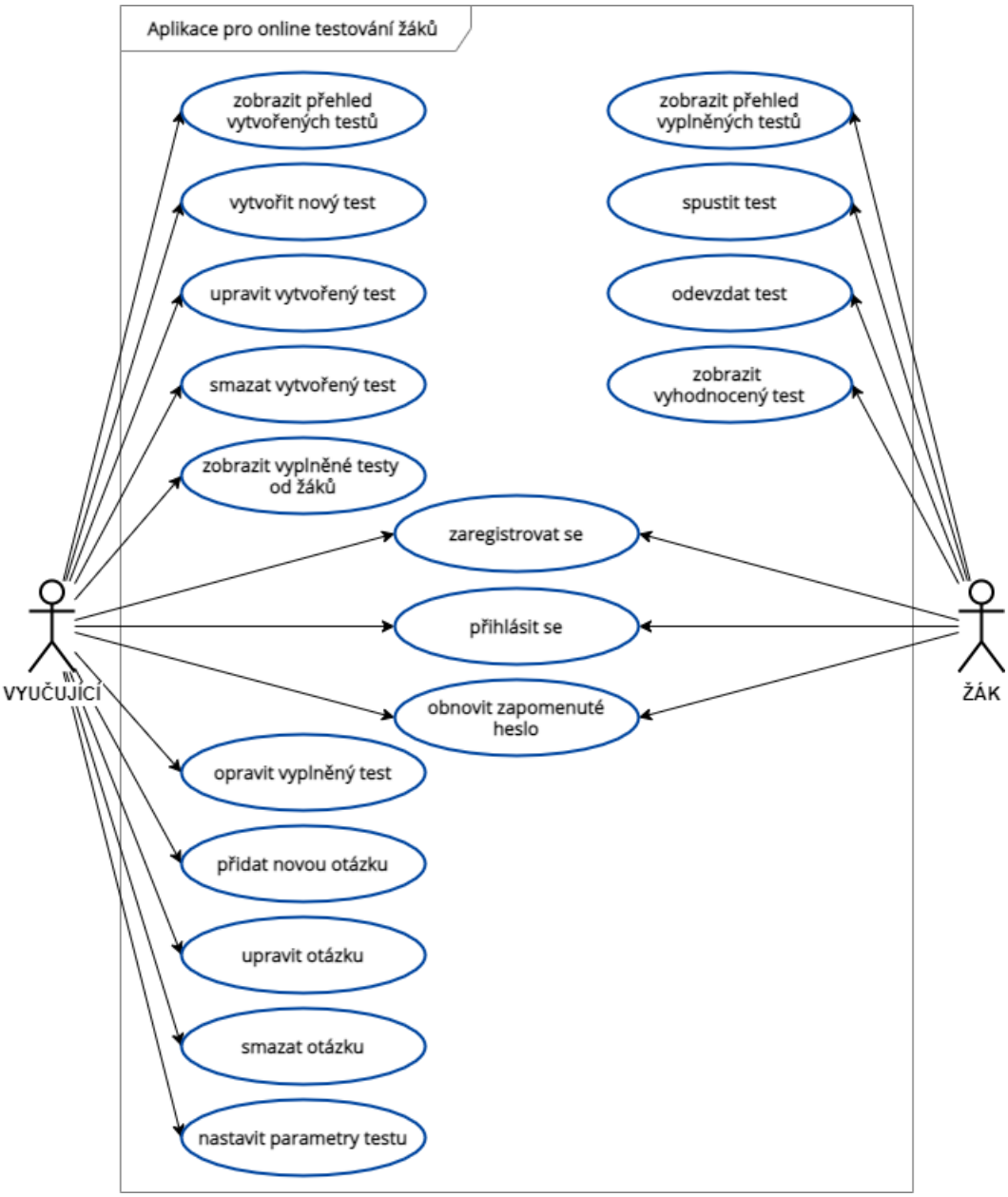
5.4.1 Připomínky vyučujících k aplikaci Google Forms

Vzhledem k tomu, že Google Forms jsou téměř identické jako Microsoft Forms, taktéž připomínky jsou shodné. Jako nejčastější je zmiňována absence časového limitu k vyplnění testu.

6 Analýza a specifikace požadavků

V této kapitole je vyobrazen use case diagram neboli diagram případů užití k mnou vytvářené aplikaci pro online testování. V druhé části kapitoly je k jednotlivým případům užití uvedena taktéž specifikace.

6.1 Use case diagram



Obrázek 12: Use case diagram

6.2 Use case specifikace

Zatímco use case diagram ukazuje, jaké funkcionality aplikace obsahuje, use case specifikace definuje doplňující informace o jednotlivých funkcionalitách.

6.2.1 Zobrazit přehled vytvořených testů

Krátký popis:
Use case umožňuje zobrazit přehled všech testů, které vyučující v aplikaci vytvořil.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém vygeneruje seznam testů, jejichž autorem je přihlášený vyučující.
2. Vyučující může seznamem procházet a vytvořené testy zobrazit, upravit či smazat.

6.2.2 Vytvořit nový test

Krátký popis:
Use case umožňuje vyučujícímu vytvořit nový test.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém vygeneruje okno pro vytváření nového testu.
2. Vyučující nastavuje parametry testu, přidává, upravuje a maže otázky.
3. Systém průběžně ukládá vytvářený test do databáze.

6.2.3 Upravit vytvořený test

Krátký popis:
Use case umožňuje vyučujícímu upravit již existující test, který vyučující dříve vytvořil.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém vygeneruje okno pro úpravu existujícího testu a načte data z databáze.
2. Vyučující upravuje parametry testu či mění otázky.
3. Systém průběžně ukládá změny v upravovaném testu do databáze.

6.2.4 Smazat vytvořený test

Krátký popis:
Use case umožňuje vyučujícímu smazat již existující test, který vyučující dříve vytvořil.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém se zeptá vyučujícího, zda chce skutečně smazat daný test.
2. Vyučující potvrdí smazání testu.
3. Systém ověří, že daný test ještě žádný žák nevyplnil a je možné jej smazat.
4. Systém odstraní data spjatá s daným testem z databáze a odstraní jej z rozhraní vyučujícího.

Alternativní tok 1:

3.1 Test již některý ze žáků vyplnil a systém informuje vyučujícího, že tento test již není možné smazat a zamítne možnost smazání.

Alternativní tok 2:

2.1 Vyučující zamítne smazání testu – například z důvodu chybného výběru testu ke smazání.

3.1 Systém ponechá veškerá data spjatá s testem a operace se neuskuteční.

6.2.5 Zobrazit vyplněné testy od žáků

Krátký popis:
Use case umožňuje vyučujícímu zobrazit jeden či více vyplněných testů od žáků.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Vyučující si vybere test v přehledu vyplněných testů.
2. Systém vygeneruje okno pro zobrazení testu včetně odpovědí konkrétního žáka.
3. Vyučující vidí jednotlivé odpovědi daného žáka na testové otázky.

6.2.6 Opravit vyplněný test

Krátký popis:
Use case umožňuje vyučujícímu opravit jeden či více vyplněných testů od žáků.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém vygeneruje okno pro zobrazení testu včetně odpovědí konkrétního žáka.
2. Vyučující může u jednotlivých odpovědí na otázky nastavit počet získaných bodů.
3. Systém průběžně ukládá změny prováděné vyučujícím v testu žáka.
4. Vyučující zpřístupní opravený test žákovi.
5. Systém přenastaví práva u testu tak, aby žák mohl své výsledky včetně zpětné vazby vyučujícího zobrazit.

Alternativní tok 1:

4.1 Vyučující se rozhodne zatím opravený test žákovi nezpřístupnit.

5.1 Systém veškeré změny uloží, ale ponechá práva u testu tak, aby žák prozatím nemohl zobrazit výsledky testu včetně zpětné vazby vyučujícího.

6.2.7 Přidat novou otázku

Krátký popis:
Use case umožňuje vyučujícímu přidat novou otázku do již existujícího či právě vytvářeného testu.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém vygeneruje nabídku možných typů otázek.
2. Vyučující zvolí požadovaný typ otázky.
3. Systém šablonu otázky vloží do vytvářeného či existujícího testu.
4. Vyučující do šablony otázky vkládá data.
5. Systém průběžně vkládaná data ukládá do databáze.

6.2.8 Upravit otázku

Krátký popis:
Use case umožňuje vyučujícímu upravit vloženou otázku z existujícího či právě vytvářeného testu.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Vyučující upraví data v šabloně otázky.
2. Systém nahradí v databázi původní data nově vloženými.

6.2.9 Smazat otázku

Krátký popis:
Use case umožňuje vyučujícímu smazat vloženou otázku z existujícího či právě vytvářeného testu.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém se zeptá vyučujícího, zda chce skutečně smazat danou otázku.
2. Vyučující potvrdí smazání otázky.
3. Systém odstraní data spjatá s danou otázkou z databáze a odstraní ji z rozhraní pro přípravu testu.

Alternativní tok 1:

2.1 Vyučující zamítne smazání otázky – například z důvodu chybného výběru otázky ke smazání.

3.1 Systém ponechá veškerá data spjatá s otázkou a operace smazání se neprovede.

6.2.10 Nastavit parametry testu

Krátký popis:
Use case umožňuje vyučujícímu nastavit všeobecné parametry týkající se konkrétního vytvářeného nebo již vytvořeného testu.

Aktéři:
Vyučující, Systém

Podmínky pro spuštění:
Vyučující musí být přihlášen v aplikaci.

Základní tok:

1. Systém zobrazí vyučujícímu přehled parametrů, které může u daného testu nastavit.
2. Vyučující parametry nastaví.
3. Systém nastavené parametry průběžně ukládá do databáze.

6.2.11 Zobrazit přehled vyplněných testů

Krátký popis:
Use case umožňuje žákovi zobrazit všechny testy, které vyplnil.

Aktéři:
Žák, Systém

Podmínky pro spuštění:
Žák musí být přihlášen v aplikaci.

Základní tok:

1. Systém vygeneruje seznam testů, které žák vyplnil.
2. Žák může seznamem procházet, vstoupit do rozpracovaného testu či zobrazit hodnocení včetně zpětné vazby vyučujícího, pokud již bylo vyučujícím zadáno a zveřejněno.

6.2.12 Spustit test

Krátký popis:
Use case umožňuje žákovi spustit konkrétní test na základě kódu poskytnutého vyučujícím.

Aktéři:
Žák, Systém

Podmínky pro spuštění:
Žák musí být přihlášen v aplikaci.
Žák musí znát kód testu, který má spustit.

Základní tok:

1. Systém požádá o vložení kódu testu.
2. Žák vloží kód testu.
3. Systém vygeneruje okno s instrukcemi k testu a tlačítkem pro zahájení.
4. Žák zahájí test.
5. Systém vygeneruje test k vyplnění.
6. Žák test vyplní.
7. Systém průběžně ukládá odpovědi žáka do databáze.

Alternativní tok 1:

2.1 Žák vloží chybný (neexistující) kód testu.

3.1 Systém informuje o nesprávném (neexistujícím) kódu testu a nabídne možnost opakování akce v podobě vložení kódu nového.

Alternativní tok 2:

4.1 Žák test nemůže zahájit, protože ještě nenastal čas otevření testu k vyplnění.

6.2.13 Odevzdat test

Krátký popis:
Use case umožňuje žákovi odevzdat spuštěný test.

Aktéři:
Žák, Systém

Podmínky pro spuštění:
Žák musí být přihlášen v aplikaci.
Žák musí mít spuštěn test.

Základní tok:

1. Žák test příslušným tlačítkem odevzdá.
2. Systém vyzve žáka k potvrzení odevzdání.
3. Žák odevzdání potvrdí.
4. Systém test ukončí a odevzdá.

Alternativní tok 1:

1.1 Vyprší časový limit.

2.1 Systém ukončí a odevzdá test automaticky.

6.2.14 Zobrazit vyhodnocený test

Krátký popis:
Use case umožňuje žákovi zobrazit vyučujícím vyhodnocený test.

Aktéři:
Žák, Systém

Podmínky pro spuštění:
Žák musí být přihlášen v aplikaci.
Žákův vyhodnocený test musí být vyučujícím nastaven jako zveřejněný.

Základní tok:

1. Žák si z opravených testů vybere konkrétní test, který chce zobrazit.
2. Systém zobrazí šablonu testu včetně odpovědí žáka, správných odpovědí, přidělených bodů a získaného hodnocení.

6.2.15 Zaregistrovat se

Krátký popis:
Use case umožňuje žákovi či vyučujícímu vytvořit uživatelský účet do aplikace.

Aktéři:
Žák, Vyučující, Systém

Základní tok:

1. Žák či vyučující zadá do formuláře své celé jméno, e-mail a vybere si uživatelskou roli.
2. Systém zkontroluje zadané údaje a ověří jejich unikátnost.
3. Systém vytvoří uživatelský účet a odešle na email přihlašovací heslo.

Alternativní tok 1:

2.1 Zadaný e-mail již existuje.

3.1 Systém nevytvoří uživatelský účet a vyzve k použití jiného e-mailu.

6.2.16 Přihlásit se

Krátký popis:
Use case umožňuje žákovi či vyučujícímu přihlásit se do aplikace.

Aktéři:
Žák, Vyučující, Systém

Základní tok:

1. Žák či vyučující zadá do formuláře svůj e-mail a heslo.
2. Systém zkontroluje zadané údaje.
3. Systém přihlásí žáka či vyučujícího do aplikace.

Alternativní tok 1:

2.1 Jsou zadány nesprávné údaje.

3.1 Systém žáka či vyučujícího nepřihlásí do aplikace a vyzve k jejich opravě.

6.2.17 Obnovit zapomenuté heslo

Krátký popis:
Use case umožňuje žákovi či vyučujícímu obnovit zapomenuté heslo do aplikace.

Aktéři:
Žák, Vyučující, Systém

Základní tok:

1. Žák či vyučující zadá do formuláře svůj e-mail.
2. Systém zkontroluje zadané údaje, vygeneruje potvrzení žádosti o nové heslo a zašle jej na zadaný e-mail.
3. Žák či vyučující potvrdí, že má o změnu hesla opravdu zájem.
4. Systém vygeneruje nové heslo, přiřadí jej k uživatelskému účtu patřícímu k zadanému e-mailu a odešle na e-mail.

Alternativní tok 1:

3.1 Žák či vyučující žádost o změnu hesla nepotvrdí.

3.1 Systém ponechá stávající heslo.

6.3 Funkční požadavky a stěžejní body pro návrh aplikace

Nejen volba moderních technologií a přístupů k vývoji, ale také specifická, v oblasti základních a středních škol dosud chybějící a zároveň potřebná funkcionalita – to jsou hlavní složky mé diplomové práce, kterými vyniká nad dosud dostupnými a používanými řešeními. Po konzultaci s vyučujícími na téma potřebných funkcionalit, které by v takové aplikaci ocenili, a analýze názorů v diskuzích a na sociálních sítích, kde se především v době pandemie koronaviru rozvíjely přínosné diskuze, jsem dospěl k následujícímu seznamu stěžejních bodů:

- **Jednoduché a svižné rozhraní v českém jazyce.** Aplikace by měla mít jednoduché uživatelské rozhraní se snadnou orientací, všechny funkcionality by měly být dostupné kompletně v českém jazyce. Přechody mezi částmi aplikace by měly být plynulé, bez dlouhého načítání.
- **Optimalizace pro pomalé připojení k internetu.** Aplikace by neměla obsahovat zbytečně náročnou grafiku, která výrazně negativně ovlivňuje dobu načítání. Žáci i vyučující mohou být v podmínkách s pomalým připojením k internetu, ve kterém by měla aplikace fungovat také spolehlivě.
- **Pokročilé automatizované opravení.** Opravení by si mělo umět poradit s uzavřenými i otevřenými otázkami. Mělo by umožňovat rozlišovat malá a velká písmena, mít možnost neomezeného počtu variant správné odpovědi, ignorovat přebytečné mezery a další nechtěné znaky.
- **Pokročilá revize odpovědí.** Takový způsob revize, při které bude žák na první pohled vidět, ve kterých otázkách chyboval. Vyučující by měl mít možnost zpětné vazby k celému testu a úpravy získaných bodů.
- **Náhodné generování otázek z vytvořené sady.** Vyučující velmi postrádají možnost generování určitého počtu otázek z připravené sady. Díky tomu se může výrazně snížit opisování mezi žáky.
- **Nastavení časového limitu na test.** Nastavit počáteční a koncové datum testu je pro mnoho vyučujících nedostatečné a neefektivní řešení. Mít možnost časového limitu, který začne běžet po spuštění testu žákem, by znamenalo zvýšení spravedlivosti podmínek pro všechny žáky.
- **Automatické propojení s hodnotícími kritérii.** Aplikace by měla umět sama vypočítat procentuální úspěšnost a přidělit známku na základě předem definovaných hodnotících kritérií.
- **Propracované otázkové moduly.** Šablony reprezentující typy otázek by měly vycházet z písemných zkoušek typických pro české školství, především v oblasti základních a středních škol. S jejichž pomocí by mělo být možné přepracovat písemné zkoušky do online podoby v co největší míře bez zásadních změn.

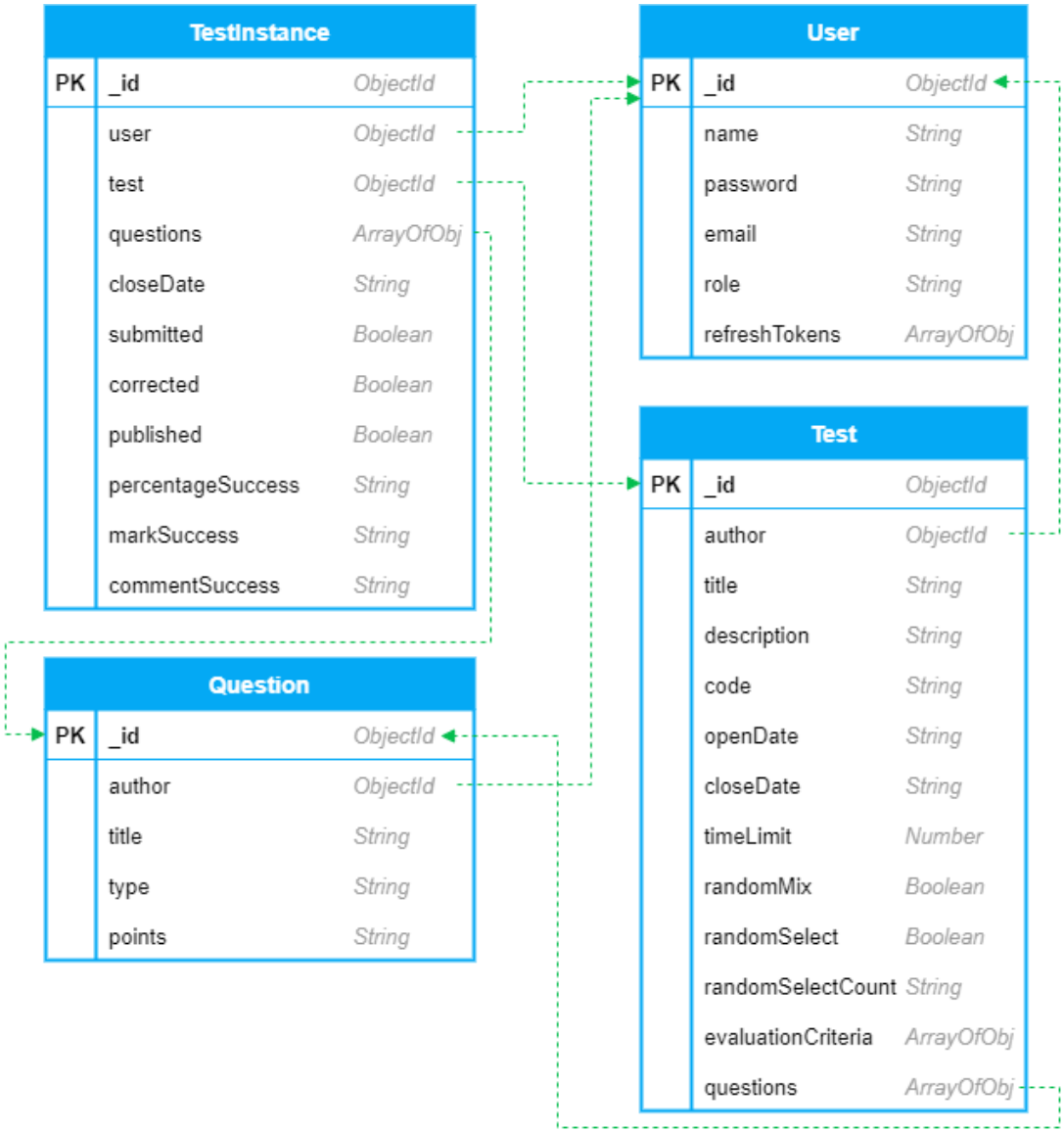
7 Návrh aplikace a její implementace

Tato kapitola se zabývá návrhem struktury databáze, technologiemi zvolenými pro implementaci, principem komunikace mezi frontendem, backendem a databází, a důležitými implementačními detaily zmiňujícími klíčové chování aplikace.

7.1 Návrh struktury databáze

Zásadním krokem před implementací je návrh struktury databáze, který bude klíčový pro způsob ukládání dat, získávání dat a práce s nimi. Na základě rozhodujících faktorů zmíněných v podsekci 4.7.2 jsem se rozhodl pro využití nerelační neboli NoSQL databáze. Mezi hlavní důvody tohoto rozhodnutí patří nejen možnost efektivní práce s JSON formátem, který je základem moderních webových aplikací, ale především fakt, že některá data nebudou mít pevnou strukturu. Základní pevnou strukturu kolekcí lze vidět na NoSQL modelu na obrázku 13. Databáze se bude skládat ze čtyř kolekcí:

- **User** - kolekce sloužící pro evidenci uživatelů aplikace. Bude evidováno celé jméno uživatele, heslo, email, uživatelská role (zda se jedná o vyučujícího či žáka) a pole refresh tokenů. Tato kolekce bude stěžejní pro autentizaci uživatelů.
- **Test** - kolekce sloužící pro evidenci jednotlivých vytvořených testů. U každého testu bude evidován jeho autor, nadpis, popisek, unikátní přístupový kód, datum a čas otevření a uzavření testu k vyplňování, časový limit na vyplnění testu, boolean proměnná evidující náhodné míchání otázek, boolean proměnná evidující náhodný výběr určitého počtu otázek z celkového počtu, počet otázek pro náhodný výběr, hodnotící kritéria a pole otázek.
- **Question** - kolekce sloužící pro evidenci jednotlivých otázek. U každé otázky bude přinejmenším evidováno zadání, autor, typ otázky a počet bodů za správné vyplnění.
- **TestInstance** - kolekce sloužící pro evidenci jednotlivých instancí testů. U každé instance bude evidován žák, který test vyplňuje, test, ze kterého instance vychází, pole otázek, datum a čas posledního možného odevzdání (závislý na časovém limitu a času zahájení testu), boolean proměnné evidující, zda je instance úspěšně odevzdána, opravena vyučujícím a zveřejněna, a na závěr atributy evidující procentuální úspěšnost žáka, získanou známku a komentář k dosaženému výsledku.



Obrázek 13: NoSQL model databáze

Zmíněná kolekce *Question* je příkladem toho, proč je vhodné využít NoSQL databázi. Ačkoliv má základní pevnou strukturu, rozšířená struktura této kolekce se bude lišit na základě typu otázky, protože u každé otázky budou potřeba jiné atributy. V SQL by to bylo možné řešit pomocí generalizace/specializace, a to tak, že k základní tabulce by musely existovat další tabulky pro každý typ otázky. Každá tabulka navíc znamená nárůst složitosti struktury databáze. Nelze ale také zapomínat na to, že při vytvoření nového typu otázky by bylo nutné upravit schéma databáze a přidat odpovídající tabulku.

Dalším problémem s SQL databází může být také fakt, že některé SQL databáze (jako například MySQL) neposkytují datový typ pole, což by například ztížilo

způsob ukládání otázek náležících k testu v kolekci *Test* či otázek náležících k instanci v kolekci *TestInstance*. Místo pole by musela vzniknout další pomocná tabulka obsahující dvojice hodnot id testu – id otázky, případně id instance – id otázky.

To vše zjednoduší použití NoSQL databáze, jejíž struktura nemusí být striktně definovaná, a jednotlivé dokumenty tak v rámci kolekce mohou mít odlišnou strukturu.

7.2 Volba technologií

Při vývoji moderních webových aplikací se často používají ověřené *stacky* neboli skupiny programovacích jazyků a technologií pro frontend, backend i databázi, jejichž vzájemné fungování dává smysl. Mezi nejpopulárnější stacky současnosti patří:

- **MERN** = MongoDB + Express.js + React + Node.js,
- **MEAN** = MongoDB + Express.js + Angular + Node.js,
- **MEVN** = MongoDB + Express.js + Vue.js + Node.js,
- **LAMP** = Linux + Apache + MySQL + PHP.

Přestože první tři zmíněné představují stacky vhodné pro vývoj moderních aplikací, poslední zmíněný je stále nejčastěji zastoupený. Jedná se o ověřenou klasiku fungující již mnoho let, která se využívá především pro webové stránky a webové aplikace využívající systém pro správu obsahu (CMS) jako například WordPress či Drupal.

Jak je možné vidět, první tři zmíněné stacky se liší ve frameworku využitém pro frontend, jinak jsou jejich technologie shodné. Srovnání technologií pro frontend jsem se věnoval v sekci 4.5 a na základě zmíněných poznatků jsem se rozhodl využít pro implementaci stack označovaný jako **MERN** – **MongoDB** jako databáze, **Express.js** jako framework pro backend, **React** jako framework pro frontend a **Node.js** jako prostředí pro spouštění kódu v JavaScriptu mimo webový prohlížeč, tedy na straně serveru.

Proč právě tato kombinace technologií? Nespornou výhodou této kombinace je, že React i Express.js jsou frameworky založené na JavaScriptu, což umožňuje vyvíjet frontend i backend aplikace ve stejném programovacím jazyku. To velmi zefektivní vývoj, protože není nutné v hlavě přepínat mezi dvěma jazyky. Zároveň je kód lépe čitelný a snadněji se v něm orientuje.

7.3 Princip komunikace mezi frontendem, backendem a databází

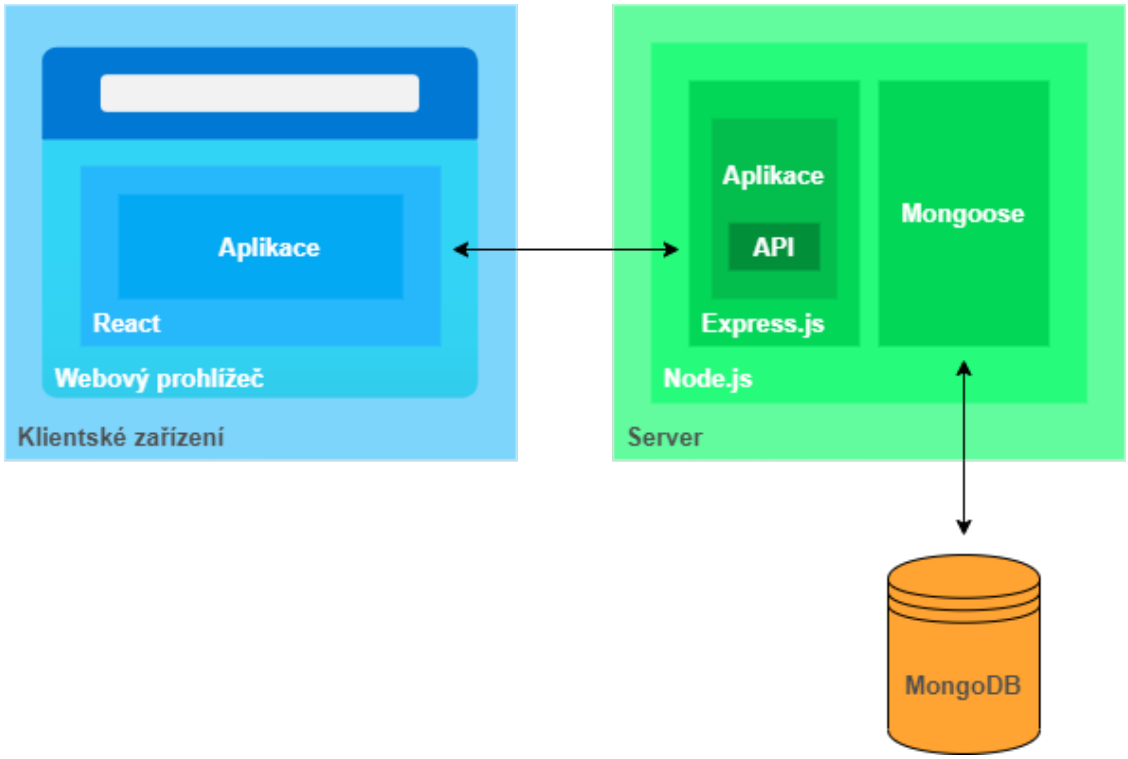
S využitím MERN stacku lze snadno zkonstruovat třívrstvou architekturu, která je neznámějším případem vícevrstvé architektury a na jejímž principu běží velké množství webových aplikací. Třívrstvá architektura se skládá z prezenční vrstvy, která zobrazuje uživateli informace a může kontrolovat zadávané vstupy, aplikační vrstvy, která zahrnuje jádro aplikace sestávající z funkcí, výpočtů a zpracování dat, a datové vrstvy, kterou tvoří nejčastěji databáze.

7.3.1 Princip komunikace v rámci vyvíjené aplikace

Schéma principu komunikace lze vidět na obrázku 14. Pro běh webové aplikace na klientském zařízení je potřeba webový prohlížeč, s jehož pomocí dojde ke spuštění webového rozhraní aplikace. Na frontendu běží deklarativní javascriptový framework React, který slouží k vytváření komplexních rozhraní pomocí komponent. Tyto komponenty lze do sebe jednoduše vnořovat a opakovaně používat, což při správném návrhu a používání zabraňuje opakování stejného kódu. Tyto komponenty obvykle mají podobu šablony, do které se doplní data a následně proběhne renderování do podoby HTML kódu. Pro získání dat komunikují komponenty s backendem, respektive se serverem, na kterém běží softwarový systém Node.js. Tato komunikace probíhá prostřednictvím HTTP požadavků, které jsou směřovány na REST API, a to tak, že po získání dat je možné vykreslit komponentu znovu bez nutnosti opětovného načtení stránky, což výrazně zrychluje a zpříjemňuje práci s webovou aplikací.

Na backendu běží systém Node.js s frameworkem Express.js, který je velmi rychlý a minimalistický. Express.js se stará především o efektivní směrování URL (tj. zachycení a zpracování URL adres přicházejících na server v podobě HTTP požadavků). Jednotlivé URL adresy mají přidělené funkce, které se obvykle starají o komunikaci s databází, v tomto případě s MongoDB. Funkcím jsou předána vstupní data a parametry, které přicházejí od klienta, následně je vyslán požadavek do databáze, která po zpracování požadavku vrátí odpověď. Odpověď je závislá na metodě v požadavku – může se jednat o samotná data, ale také například pouze o odpověď oznamující, že vložení či úprava dat proběhli v pořádku. Veškerý přenos dat probíhá ve formátu JSON, ke kterému umějí přistupovat všechny používané technologie, a není proto potřebné formu dat jakýmkoliv způsobem upravovat.

Samotná databáze spravuje veškerá data, která je v rámci aplikace potřebné ukládat. Struktura databáze i ukládaná data vycházejí z mého návrhu struktury databáze, který se nachází v sekci 7.1.



Obrázek 14: Schéma principu komunikace

7.4 Implementační detaily

V následujících podsekcích zmiňuji části implementace, které považuji za důležité pro správné porozumění chování a bezpečnosti aplikace. Některé zajímavé implementační detaily týkající se konkrétních částí aplikace jsou pro přehlednost zmíněny až kapitole 8 pojednávající o dosažených výsledcích.

7.4.1 Autentizace

Autentizace je proces sloužící k ověření identity a je nedílnou součástí této aplikace, do níž je přistupováno na základě uživatelského účtu.

Každý uživatelský účet obsahuje mimo jiné email a heslo, na jejichž základě je prováděna autentizace. Při vytváření účtu se odešle požadavek na server, kde proběhnou následující operace:

- zkontroluje se, zda již neexistuje uživatel se zadaným emailem, a pokud ano, vrátí se chyba
- vygeneruje se pseudonáhodné heslo o délce 12 znaků
- vytvoří se instance modelu uživatelského účtu

- proběhne hašování hesla algoritmem bcrypt s generováním kryptografické soli v 12 kolech
- instance modelu se uloží do databáze do příslušné kolekce
- odešle se email s přihlašovacími údaji
- server pošle odpověď, že vytvoření účtu proběhlo úspěšně

Bcrypt je hašovací funkce, která v sobě zahrnuje kryptografickou sůl, čímž je chráněna proti útokům pomocí duhové tabulky (tabulka s předem vypočtenými hodnotami hashe pro usnadnění a urychlení prolomení hašovací funkce útočníkem). Taktéž se jedná o funkci, u níž je možné zvolit libovolný počet iterací, respektive kol, čímž se ale výpočet zpomaluje.

Přihlašování do aplikace probíhá na základě emailu a obdrženého vygenerovaného hesla, které se v požadavku odešlou na server, a následně proběhnou tyto kroky:

- zkontroluje se, zda existuje uživatel se zadaným emailem, a pokud ne, vrátí se chyba
- proběhne porovnání hesla, které uživatel zadal, s hashem uloženým v databází, který reprezentuje původní heslo, a pokud nenastane shoda, vrátí se chyba
- vytvoří se přístupový token (access token)
- vytvoří se obnovující token (refresh token)
- tokeny se přiřadí k uživatelskému účtu
- server vrátí v odpovědi JSON objekt obsahující přístupový token, obnovující token a základní informace o uživateli (id, jméno, email a uživatelská role)

Tokeny poskytují způsob autorizace každého požadavku od klienta k serveru. Jsou generovány na straně serveru, a to na základě tajného klíče, který si vývojář zvolí a který je na serveru uložen. Po přihlášení do aplikace je token uložen v úložišti v prohlížeči klienta označovaném jako localStorage.

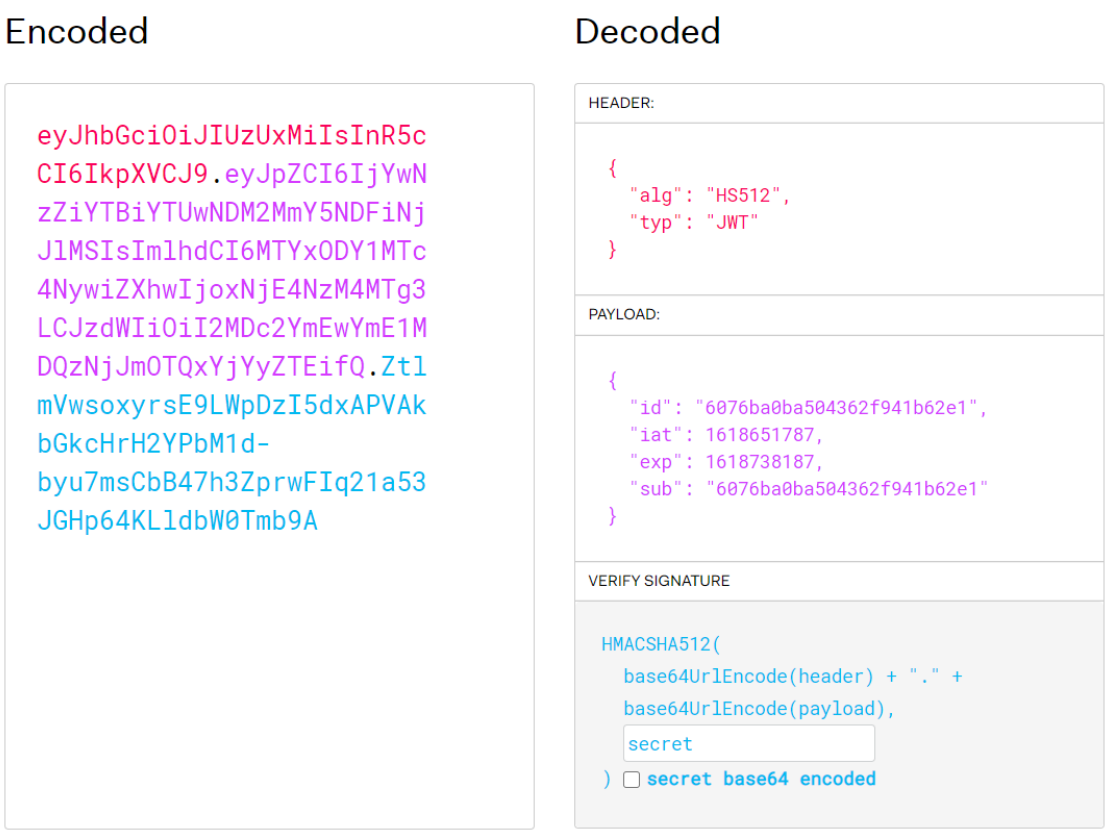
Každý token má z bezpečnostních důvodů omezenou životnost. Omezení životnosti přístupového tokenu omezuje dobu, po kterou by mohl případný útočník ukradený token zneužít. Když vyprší platnost přístupového tokenu a uživatel stále potřebuje přístup k aplikaci, bez užití obnovujícího tokenu by aplikace čelila problému získání nového přístupového tokenu bez nutnosti zásahu uživatele. K vyřešení tohoto problému je využit obnovující token, který umožňuje aplikaci získat nový přístupový token bez zásahu uživatele. Přístupový i obnovující token mají svoji omezenou životnost, přičemž pro každý z těchto typu tokenů může být odlišná.

Jako konkrétní způsob řešení práce s tokeny jsem zvolil JSON Web Token, který je známý pod zkratkou JWT a který představuje způsob bezpečné výměny informací mezi dvěma stranami, v tomto případě mezi klientem a serverem. Hlavním

cílem JWT je možnost ověření autenticity dat, tedy vlastnosti, že data nebyla během přenosu mezi zmíněnými dvěma stranami nijak pozměněna.

Podepsaný serializovaný JWT sestává ze třech částí, které jsou od sebe odděleny tečkou. Jsou to:

- hlavička (header), která se skládá ze dvou částí – hašovacího algoritmu a typu tokenu,
- data (payload), která jsou hlavní částí tokenu nesoucí samotnou konkrétní informaci,
- a podpis, který je založen na hlavičce, datech a tajném klíči uloženém na serveru.



Obrázek 15: Ukázka podepsaného zašifrovaného a dešifrovného tokenu z aplikace

Na obrázku 15 je možné vidět konkrétní příklad JWT z aplikace. Na levé straně je zašifrovaná podepsaná serializovaná varianta tokenu a na pravé straně je dešifrovaný token. Jak lze vyčíst z hlavičky tokenu, v aplikaci jsem zvolil algoritmus HS512, což je zkratka pro HMAC + SHA512. Tento algoritmus využívá kryptografickou hashovací funkci SHA-512. Podle zdroje Next Generation Cryptography[15] od

[15] https://tools.cisco.com/security/center/resources/next_generation_cryptography

společnosti Cisco z října roku 2020 je považován SHA-512 za algoritmus splňující parametry postkvantové bezpečnosti (tj. odolný vůči případným útokům kvantových počítačů) a v kombinaci s HMAC taktéž adekvátní způsob pro ověřování integrity dat. V datové části je uchováván jedinečný identifikátor uživatele a datum a čas expirace tokenu.

Pokud by se hacker pokusil nahradit data uvedená v hlavičce či v datové části tokenu, token se stane neplatným, protože podpis nebude odpovídat původním hodnotám. Hacker nemá ani možnost vygenerovat si nový podpis, protože tajný klíč, který se pro šifrování používá, se nachází na straně serveru.

7.4.2 Autorizace

Autorizace je proces povolení přístupu nebo získání souhlasu s provedením nějaké operace či akce. Stejně jako v případě autentizace se jedná o nepostradatelnou součást této aplikace.

V aplikaci je autorizace prováděna na dvou úrovních – na frontendu u klienta a na backendu na serveru, přičemž na každé úrovni má jiný cíl a řeší jinou problematiku.

Autorizace na frontendu se primárně stará o přizpůsobení uživatelského rozhraní aplikace na základě uživatelské role. Uživatelské rozhraní pro žáky je odlišné od uživatelského rozhraní pro vyučující, každá role má specifický cíl a specifické potřeby při využívání aplikace. Uživatelská role přihlášeného uživatele je získávána z dat, která se společně s tokenem ukládají do úložiště v prohlížeči při autentizaci. Hlavním cílem je uživateli poskytnout odpovídající rozhraní přizpůsobené potřebám a na bezpečnost zde není kladen přílišný důraz.

V případě autorizace na backendu je ale situace zcela odlišná a bezpečnost je v tomto případě naprostou prioritou. Hlavním cílem je, aby se neautorizovaný uživatel v žádném případě nedostal k datům. Vzhledem k tomu, že na backendu je vytvořené REST API, ke kterému se z frontendu přistupuje pomocí požadavků, autorizace je zde řešena na úrovni jednotlivých endpointů. Autorizační proces popíši na příkladu jedné z částí aplikace, kde se vyučujícímu zobrazí tabulka se všemi testy, které vytvořil. Celé to probíhá v následujících krocích:

- z frontendu je vyslán požadavek na server na získání všech testů přihlášeného uživatele
- backendový směrovač vyhodnotí cestu požadavku a přidělí k ní odpovídající funkci, která se provede
- před provedením se spustí autorizační middleware, který:
  - z hlavičky požadavku vyextrahuje token, a pokud se v ní token nenachází, vrátí chybu a ukončí celý proces
  - ověří platnost přístupového tokenu, a pokud je nevalidní či již expiroval, vrátí chybu a ukončí celý proces

  - získá id uživatele a doplní jej do požadavku
  - povolí zahájení provedení odpovídající funkce
- provede se odpovídající funkce a vrátí se odpověď

Konkrétní podoba funkce je již závislá na konkrétním požadavku a jeho typu. Součástí požadavku mohou být parametry (jako například id testu či id uživatele) a tělo požadavku obsahující konkrétní data. V některých případech je potřeba zkontrolovat nejen to, zda uživatel existuje a má platný token, ale také to, zda má daný uživatel právo žádat o konkrétní data s konkrétními parametry. Příkladem je například situace, kdy by se snažil uživatel s určitým identifikátorem získat přístup k testům jiného uživatele s jiným identifikátorem. Takový požadavek je zablokován a server vrací chybovou odpověď.

Za zmínku z pohledu implementace ještě stojí, že pro komunikaci s databází je využito asynchronní programování. Moderní JavaScript nabízí dvojici klíčových slov `async/await`, pomocí kterých je možné prováděnou funkci libovolně pozastavovat. Tím je zajištěno, že funkce bude pokračovat až v okamžiku, kdy data získaná z databáze budou kompletní.

7.4.3 Frontend

Jak jsem již zmínil v sekci týkající se výběru technologií pro implementaci aplikace, celý frontend je postavený na webovém frameworku React. Z pohledu implementace bych rád zmínil, že aplikace využívá deklarativní směrování, což je přístup, kterým lze vykreslovat jednotlivé komponenty v závislosti na URL adrese v single-page aplikaci. Díky tomu při přechodech mezi jednotlivými částmi aplikace (a tedy jednotlivými URL adresami) nedochází k opětovnému načítání celé stránky, ale pouze k zobrazování a skrývání jednotlivých komponent, což značně zrychluje práci s celou aplikací a podporuje kladný vizuální dojem na uživatele.

Velkou výhodou využití JavaScriptu jako hlavního jazyku pro frontend (a v případě této aplikace i pro backend) je možnost využití správce javascriptových balíčků označovaného jako NPM. Jedná se o největší bezplatný registr na celém světě obsahující více než milion balíčků, které lze velmi jednoduše spárovat se svým projektem. Jeho hlavním cílem je podpořit myšlenku moderního vývoje webových aplikací a poukázat na možnost navýšení produktivity, elegantnosti i bezpečnosti vývoje za využití již existujících a otestovaných částí kódu, které by bylo velmi neefektivní psát znovu. Příkladem takového balíčku využitém v aplikaci je *Material-UI*, což je v současnosti světově nejpopulárnější sada UI komponent. Využití takového balíčku přináší výhodu nejen v tom, že aplikace obsahuje UI prvky, jejichž styl se drží aktuálního trendu v oblasti webového designu, ale také je zajištěno a důkladně otestováno, že se budou prvky chovat stejně či velmi podobně na různých typech zařízení, ze kterých se bude k aplikaci přistupovat.

7.4.4 Optimalizace datových přenosů

Důležitou součástí implementace je v neposlední řadě i optimalizace datových přenosů. Cílem moderní aplikace je mimo jiné i to, aby se změny v ní prováděné ukládaly automaticky bez nutnosti k tomu určených tlačítek, což je příjemné nejen z uživatelského hlediska, ale slouží to také i jako určitá ochrana před ztrátou dat v případě výpadku připojení k internetu. V mnoha aplikacích se stává, že tlačítko pro uložení slouží k uložení většího počtu prvků stránky, na nichž může uživatel pracovat i delší dobu. Poté když dojde k výpadku připojení k internetu, stránka zmizí a po obnovení připojení je práce uživatele ztracena.

V případě této aplikace jsou využívány tzv. hooky, což jsou primitiva, díky kterým mohou funkcionální komponenty využívat stav či životní cyklus a zároveň nemusí být definovány jako třída. Každá změna (a to i například každá změna písmene) je ukládána do hooku určeného pro práci se stavem. V takovém hooku je možné definovat výchozí hodnoty, což zabrání vznikům nedefinovaných hodnot. Pokud by ale při každé takové změně byla změna odesílána i do databáze, významně by to zpomalovalo aplikaci a navyšovalo datový tok. Proto jsem zvolil variantu v podobě tzv. onBlur události, což je událost, která nastane vždy, když aktuální objekt ztratí fokus. V praxi to znamená, že aktualizace v databázi se provede vždy, když aktuální prvek přestane být aktivní (tj. uživatel klikne mimo něj či do jiného prvku).

Další optimalizace spočívá v tom, že pro celou stránku není používán jeden stavový hook, ale každá komponenta má svůj hook, případně i několik různých hooků. Na straně backendu také dochází u některých endpointů k analýze zaslaných dat a podle toho přizpůsobené aktualizaci. To znamená, že není potřeba z frontendu zasílat kompletní data, ale pouze ta, která se mění. Tím se výrazně snižuje velikost dat, která se při komunikaci mezi frontendem a backendem zasílají, zvyšuje se rychlost komunikace a snižuje se zatížení serveru.

8 Dosažené výsledky

V této kapitole bych rád představil výslednou podobu aplikace, která v rámci mé diplomové práce vznikla. Jednotlivé sekce pokrývají části aplikace, do kterých ji je možné rozdělit. Autentizaci a autorizaci jsem se již důkladně věnoval v předchozí kapitole, proto zde bude zmíněna pouze funkcionalita a podoba aplikace po přihlášení.

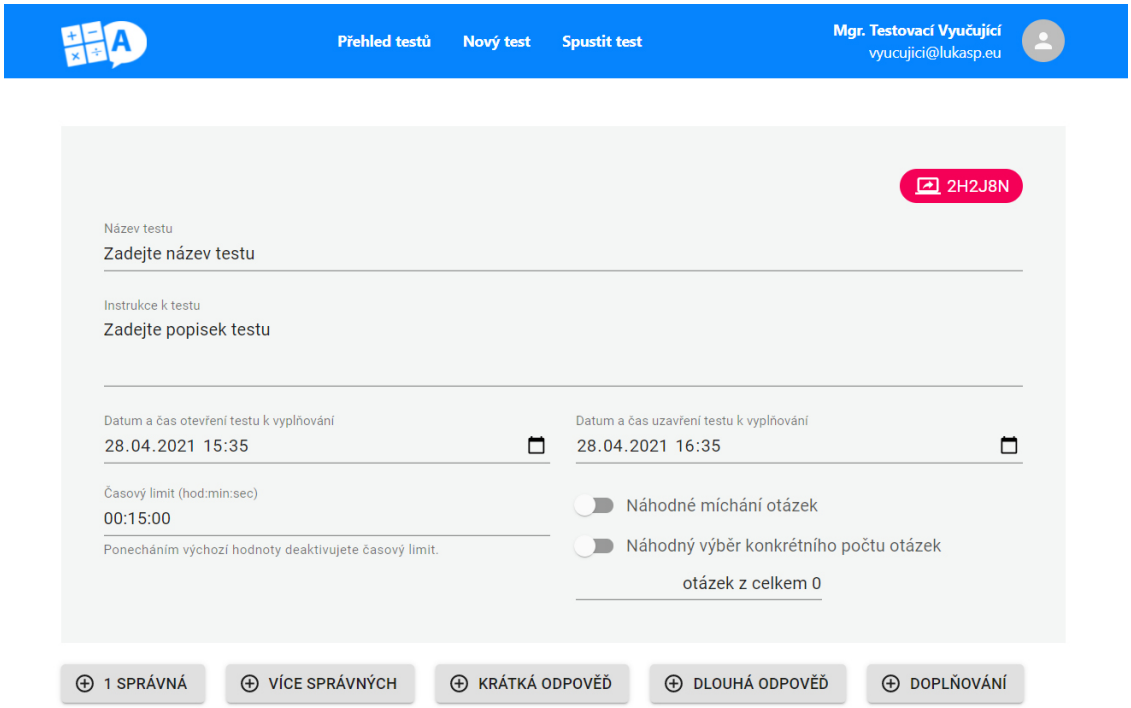
8.1 Vytváření a úprava testu

Začít vytvářet test lze stisknutím položky Nový test v menu aplikace, čímž se vytvoří nový test, který získá v rámci databáze unikátní identifikátor a zároveň je pro něj na straně backendu vygenerován unikátní kód složený z číslic a velkých písmen, který slouží pro spuštění testu. Vyučující má u každého testu možnost následujících nastavení:

- název testu
- instrukce k testu – zobrazí se žákovi před zahájením testu
- datum a čas otevření testu k vyplňování
- datum a čas uzavření testu k vyplňování
- časový limit – ponecháním výchozí hodnoty 00:00:00 se deaktivuje
- aktivace náhodného míchání otázek
- aktivace náhodného výběru konkrétního počtu otázek z celkového počtu

Následně může vyučující začít přidávat do testu jednotlivé otázky, přičemž má na výběr z pěti modulů, které pokrývají většinu možných typů otázek:

- 1 správná – libovolné množství možností na výběr a jedna z nich je správná
- více správných – libovolné množství možností na výběr a jedna či více je správných
- krátká odpověď – krátká otevřená odpověď na otázku
- dlouhá odpověď – dlouhá otevřená odpověď na otázku
- doplňování – doplňování písmen či slov do textu



Obrázek 16: Rozhraní pro vytváření a úpravu testu

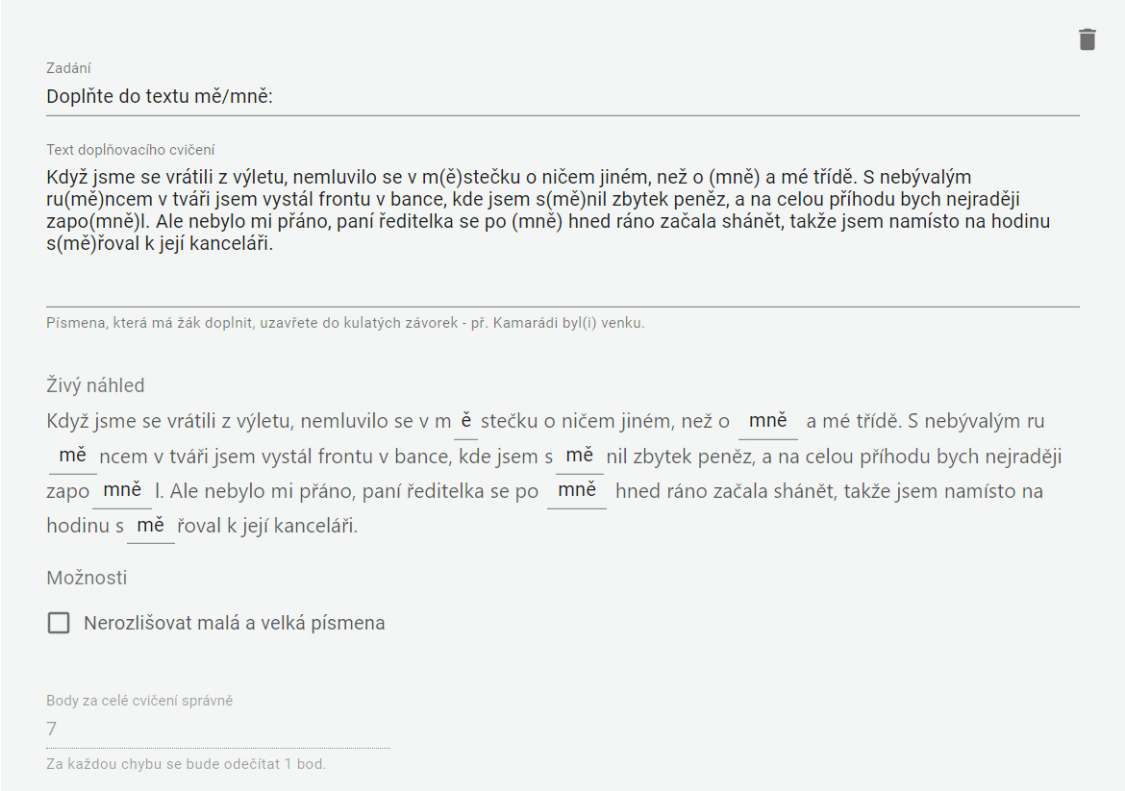
Poslední ze zmíněných možností byla velmi poptávaná a chybějící v existujících řešeních. Byla navržena a naimplementována tak, aby byla využitelná pro široké spektrum předmětů. Doplňovací cvičení lze využít například:

- v českém jazyce pro tvorbu pravopisných cvičení, kde žáci mohou doplňovat do textu písmena jako například i/y nebo s/z, spojky, předložky a další pravopisné jevy,
- v literatuře pro doplňování jmen autorů k jejich dílům,
- v matematice pro doplňování správných výsledků do příkladů, pokud vyučující nechce mít každý příklad jako samostatnou otázku,
- v cizích jazycích pro správné rozmístění slovíček z nabídky do textu
- či v dějepise pro doplňování roků k textu popisujícímu určitou historickou událost.

Jak lze vidět na zmíněných příkladech, tento otázkový modul má velmi široké využití. Vytvoření takového cvičení je zcela jednoduché. Vyučující pouze píše text a místa, která chce vynechat obalí do kulatých závorek. Vyučující má také možnost převzít již existující text například z nějakého článku, vložit jej do modulu a následně obalit do závorek jevy, které chce před žákem skrýt. Tento modul obsahuje také živý náhled, ve kterém vyučující při vytváření cvičení průběžně vidí, jak cvičení se

správnými odpověďmi vypadá.

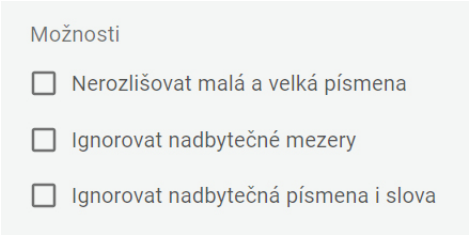
Z pohledu implementace je to řešeno tak, že se průběžně s každou změnou kontroluje textové pole, v něm se identifikují sekvence začínající a končící kulatou závorkou a ty jsou ve výstupu nahrazeny textovým polem, kterému je jako hodnota vložen obsah sekvence. Díky tomu nemusí vyučující následně zadávat správné odpovědi pro automatizované vyhodnocení, ale oba tyto kroky se provádějí současně. Jednotlivé jevy se také automaticky počítají a v závislosti na jejich počtu je nastaven počet bodů za cvičení. Když žák test spustí, uvidí tento typ otázky podobně jako vyučující v živém náhledu, avšak bez správných odpovědí. Při implementaci byl kladen důraz i na jeden podstatný detail, který je ale často opomíjen. Kdyby mělo textové pole velikost správné odpovědi, žák by mohl snadno odhalit, jaká odpověď do daného pole patří. Kdyby byla všechna textová pole stejně velká, doplňovací cvičení by bylo velmi nepřehledné. Proto jsem pro implementaci zvolil elegantní řešení, které ve výchozím stavu nastaví všechna pole na velikost jednoho znaku. Jakmile ale do pole žák zadá znak druhý, pole se okamžitě roztáhne na rozměr dvou znaků. Při přidávání a odebírání dalších znaků pole dynamicky přizpůsobuje svoji velikost, díky čemuž není žák schopen odhalit, kolik znaků má správná odpověď.



Obrázek 17: Modul doplňovacího cvičení

Tak jako je v případě doplňovacího cvičení možné nastavit, aby nebyla rozlišována malá a velká písmena, i u ostatních typů otázek se nacházejí možnosti, které

mohou zjednodušit automatizované vyhodnocení. Kromě nerozlišování malých a velkých písmen a ignorování nadbytečných mezer považuji za velký přínos právě třetí možnost, která přináší ignorování nadbytečných písmen i slov. Při diskuzi s žáky základních i středních škol jsem se často setkal s jejich rozhořčením z toho, že do online testu napsali správnou odpověď, ale protože byla lehce jinak formulovaná, nebyla jim uznána. Uvedu tuto situaci na příkladu. Vyučující v testu napsal otázku „Uveďte jméno posledního biskupa Jednoty bratrské, který také napsal slavné dílo Labyrint světa a ráj srdce.“, přičemž správná odpověď byla „Jan Amos Komenský“. Žákyně do odpovědi napsala „Je to Jan Amos Komenský.“ a tato odpověď jí nebyla uznána, ač byla zcela správná. A právě tento problém řeší třetí nabízená možnost, která si dokáže najít v odpovědi žáka shodu se správnou odpovědí.

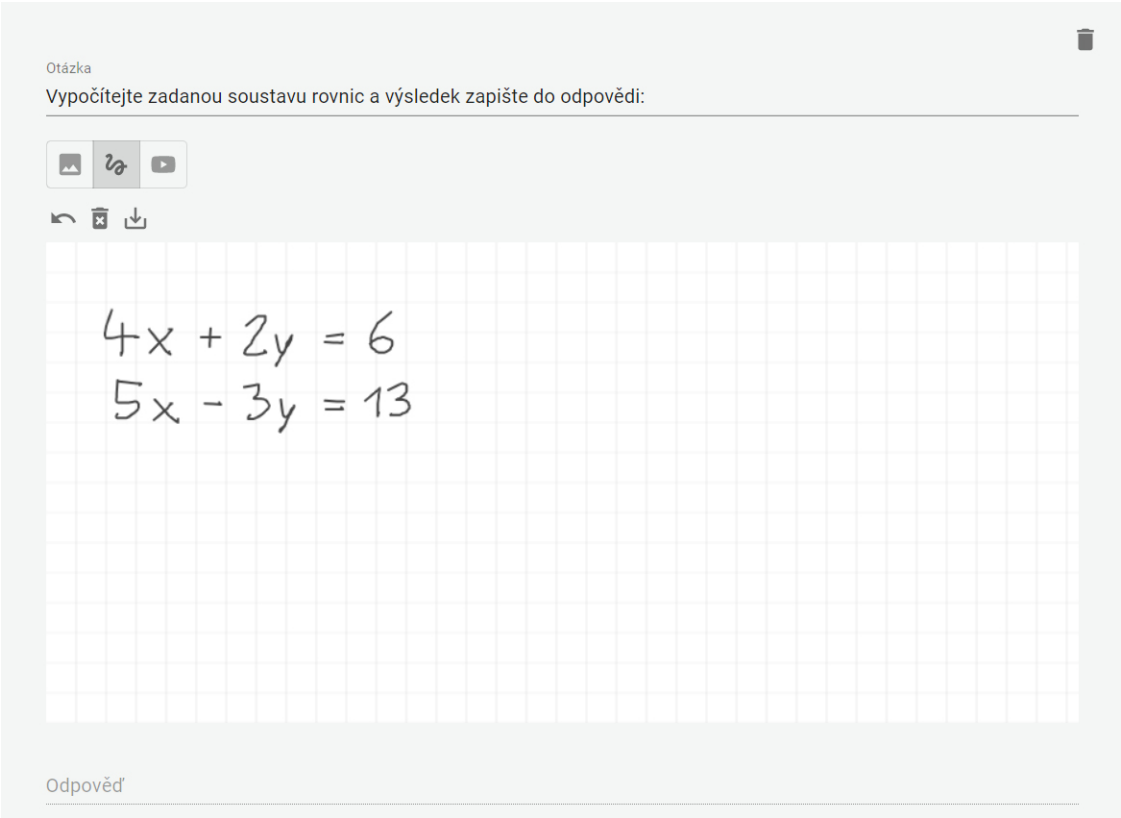


Obrázek 18: Možnosti nastavení otázky

Za významný přínos této aplikace považuji také možnost rychlého vložení multimediálního obsahu k jednotlivým otázkám. Ke každé otázce je možné vložit:

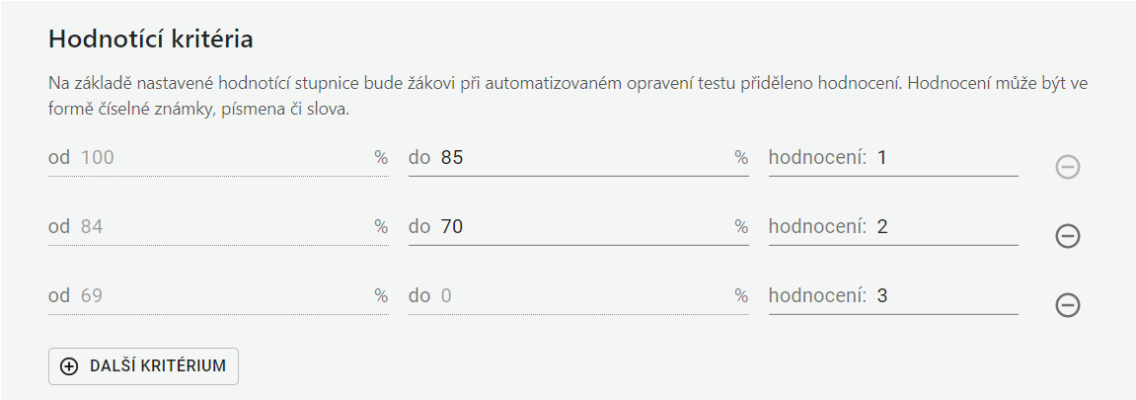
- obrázek z libovolné URL adresy bez nutnosti zdlouhavého nahrávání,
- náčrtek vytvořený přímo v aplikaci,
- a libovolné video z platformy YouTube.

Multimediální obsah má výrazný vliv na rozšíření možností jednotlivých testů. Díky němu mohou být testy pro žáky mnohem zajímavější oproti běžným písemným zkouškám na papíře. Jaké může být jeho využití v praxi? Obrázek může mít široké využití například v biologii, kde mohou žáci poznávat stromy, květiny, houby či kameny. Náčrtek může využít vyučující matematiky, který bude chtít zapsat soustavu rovnic nebo nakreslit trojúhelník, u kterého má žák určit velikost jedné strany. Video své využití může nacházet v předmětech, kde se vyučují cizí jazyky. Vyučující může například vložit video a chtít po žákovi, aby odpověděl na otázku týkající se toho, co v rozhovoru zazní. Právě poslech je v cizích jazycích velmi důležitý a často mu ve školách není věnován dostatečný prostor. Žáci při něm také nemusí být závislí na nekvalitním zvukovém výstupu ve třídě a mohou si jej přehrát sami pro sebe do sluchátek.



Obrázek 19: Ukázka multimediálního obsahu v podobě náčrtku

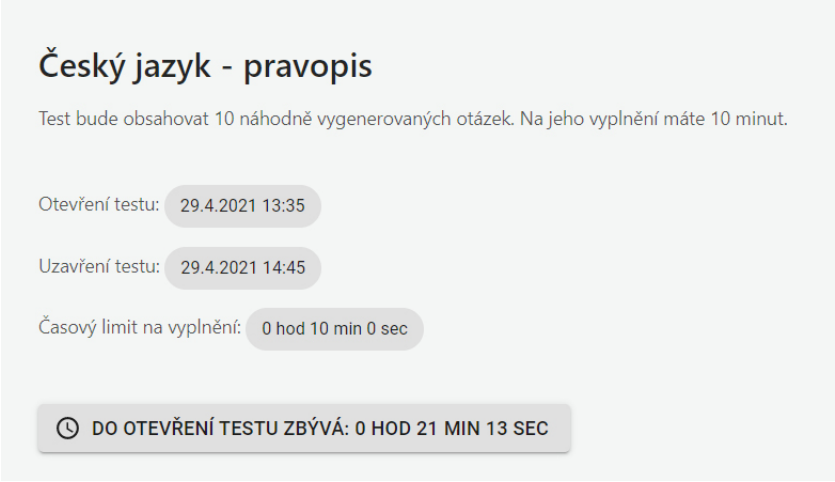
Na konci rozhraní pro vytváření testu je možné definovat hodnotící kritéria, která budou využita při automatizovaném vyhodnocení testu pro udělení konkrétního hodnocení na základě procentuální úspěšnosti. Lze definovat libovolné množství kritérií, přičemž je automaticky hlídána procentuální návaznost jednotlivých kritérií, aby byl pokrytý celý procentuální rozsah od 0 % do 100 %.



Obrázek 20: Nastavení hodnotících kritérií

8.2 Spuštění a vyplňování testu

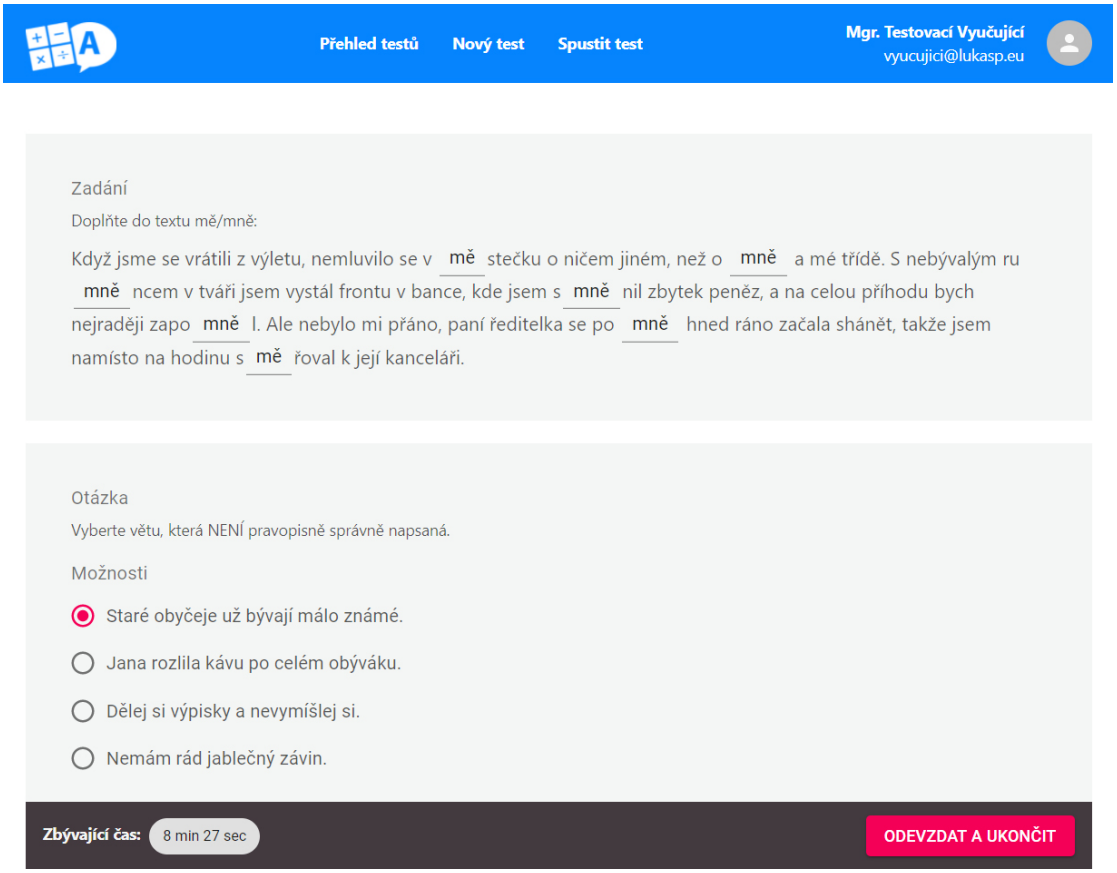
Spustit test lze stisknutím položky Spustit test v menu aplikace, čímž se zobrazí okno pro zadání kódu testu, který chce uživatel spustit. Po zadání platného kódu se test načte a zobrazí se základní informace a instrukce k jeho vyplnění. Zahájit test je možné stisknutím příslušného tlačítka. Pokud test ještě nebyl otevřen k vyplnění, v tlačítku se zobrazí odpočet, který se automaticky aktualizuje. Jakmile nastane čas spuštění testu, tlačítko se změní ve stisknutelné a je možné pomocí něj test zahájit.



Obrázek 21: Načtení testu

Na základě toho, zda bylo při vytváření zvoleno náhodné míchání či náhodný výběr určitého počtu otázek je na backendu vytvořena instance testu, která je přidělena přihlášenému uživateli. V rámci instance je definován také specifický čas odevzdání pro danou instanci. Po vykreslení testu začne běžet odpočet času zbývajícího do odevzdání, přičemž uživatel může otázky zodpovídat v libovolném pořadí. Z pohledu implementace je zde zajištěno, že jeho odpovědi jsou průběžně ukládány k dané instanci pro případ, že by na zařízení, na kterém test vyplňuje, došlo k nějakému problému či výpadku. Pokud by si žák například omylem zavřel prohlížeč, k testu se může kdykoliv vrátit a navázat, avšak jeho časový limit ubíhá i po dobu jeho nepřítomnosti.

Po vyplnění testu může žák test ukončit a odevzdat stisknutím k tomu určeného tlačítka. Vzhledem k tomu, že by ke stisknutí mohlo dojít omylem, zobrazí se dialog v modálním okně, který žádá o potvrzení odevzdání a zároveň upozorňuje na fakt, že po odevzdání již nebude možné své odpovědi upravit. Pokud by vypršel časový limit a žák by sám test neodevzdal, odevzdá se test automaticky bez žádosti o potvrzení. Pokud by žák z testu či celé aplikace odešel během vyplňování, již se k němu nevrátil a časový limit by vypršel, vyučující je informován o tom, že test nebyl řádně odevzdán. Mojí snahou zde bylo maximálně simulovat reálné školní prostředí, ve kterém by mohl žák úmyslně test také neodevzdat.



Obrázek 22: Vyplňování testu

V případě, že u testu není definován časový limit, zobrazí se pouze možnost „Odevzdat a ukončit“. Odpočítávání zbývajícího času i tlačítko pro odevzdání jsou umístěny do fixního pruhu, který je stále viditelný i při rolování stránky, což umožňuje mít neustálý přehled o zbývajícím času i možnost okamžitě test odevzdat.

Po odevzdání testu se žákovi zobrazí informační hláška o úspěšném odevzdání testu. Dokud nebude test vyhodnocený a zveřejněný vyučujícím, žák svůj test nemůže zobrazit.

8.3 Vyhodnocení testu

Třetí hlavní částí aplikace je vyhodnocení testu, které je určeno vyučujícímu. Veškeré instance testů se zobrazují v přehledu daného testu od fáze jejich vytvoření. Díky tomu má vyučující přehled o tom, kdo na testu ještě pracuje, kdo jej již odevzdal či kdo neodevzdal. Jakmile žák test odevzdá, vyučujícímu se odkryje možnost test opravit a následně jej zveřejnit.

ZVEŘEJNIT VŠECHNY OPRAVENÉ TESTY

Žák	Stav testu	Stav vyhodnocení	Výsledek	Akce
František Novák	odevzdán	Opraveno Zveřejnit		
Eliška Vyskočilová	odevzdán	Opraveno Zveřejnit	60 % 3	
Pavel Lesný	žák vyplňuje			

Obrázek 23: Přehled instancí testu

Žák: Eliška Vyskočilová

Zadání

Doplňte do textu mě/mně:

Když jsme se vrátili z výletu, nemluvilo se v mně stečku o ničem jiném, než o mě a mé třídě. S nebývalým ru mě ncem v tváři jsem vystál frontu v bance, kde jsem s mě nil zbytek peněz, a na celou příhodu bych nejraději zapo mně l. Ale nebylo mi přáno, paní ředitelka se po mě hned ráno začala shánět, takže jsem namísto na hodinu s mně řoval k její kanceláři.

_____ / 7

Otázka

Vyberte větu, která NENÍ pravopisně správně napsaná.

Možnosti

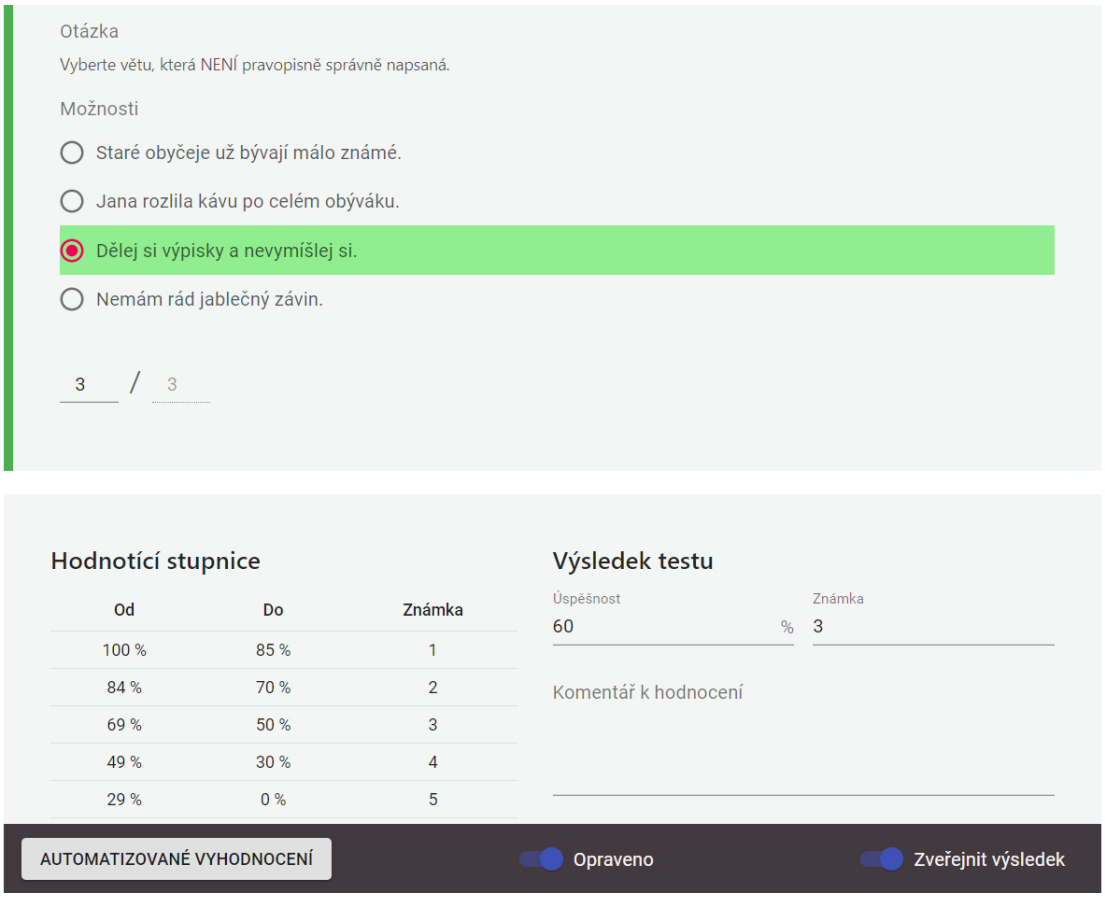
- Staré obyčeje už bývají málo známé.
- Jana rozlila kávu po celém obýváku.
- Dělej si výpisky a nevymíšlej si.
- Nemám rád jablečný závin.

AUTOMATIZOVANÉ VYHODNOCENÍ Opraveno Zveřejnit výsledek

Obrázek 24: Vyhodnocení testu

Po zvolení konkrétního testu k opravení se zobrazí test žáka tak, jak jej odevzdal, avšak doplněný o správné odpovědi. V případě prvních čtyř typů otázek se správná odpověď zobrazuje v zeleném obdélníku, v případě doplňovacího cvičení jsou zeleným obdélníkem označeny správné odpovědi a červeným obdélníkem chybné odpovědi. Vyučující tak na první pohled vidí, které odpovědi jsou správné a které chybné. Nyní má dvě možnosti:

- test se rozhodne opravit manuálně, což znamená, že bude procházet jednotlivé otázky a přidělovat body dle vlastního uvážení,
- nebo se rozhodne opravit test automatizovaně, což znamená, že se spustí vyhodnocovací skript, který projde všechny odpovědi, automaticky přidělí počet bodů a test oznámkuje.



Obrázek 25: Automatizované vyhodnocení testu

Z pohledu implementace funguje vyhodnocovací skript tak, že prochází v rámci cyklu postupně všechny otázky z dané instance testu. V každé iteraci cyklu proběhne rozhodnutí, do které kategorie vyhodnocení otázka patří, a to na základě jejího typu. Pro každý typ je definován samostatný způsob vyhodnocení, který je mimo

jiné založen na způsobu ukládání dat dané otázky v databázi. Všechny kategorie vyhodnocení mají společné to, že nejprve jsou vyhodnoceny možnosti nastavení, na základě kterých je nastavena míra přísnosti vyhodnocení otázky. Celé vyhodnocení probíhá na frontendu a je založené na datech získaných při načtení testu, díky čemuž není potřeba opětovně komunikovat s backendem a zbytečně zatěžovat server.

Důležitým vizuálním prvkem, který se zobrazuje, jakmile jsou k otázce přiděleny body, je barevný pruh na levé straně každé otázky. Ten signalizuje úspěšnost žáka v dané otázce. Pruh může být:

- červený – pokud žák dostane 0 bodů a jeho odpověď je zcela nesprávná,
- zelený – pokud žák dostane plný počet bodů a jeho odpověď je zcela správná,
- oranžový – pokud žák dostane více než minimum a méně než maximum.

Důležité je také zmínit, že po spuštění automatizovaného vyhodnocení je možné cokoliv manuálně opravit. Vyučující není na výsledku automatizovaného vyhodnocení nijak závislý a může jej použít jak pro kompletní opravu, tak pouze pro prvotní vyhodnocení s tím, že vyhodnocení následně upraví podle sebe.

Po dokončení vyhodnocení může zkontrolovat dosaženou úspěšnost a známku a doplnit k hodnocení libovolný komentář. Následně pomocí přepínačů může test označit jako opravený a případně zveřejnit výsledek. Jakmile označí test jako opravený, žák se dozví, že test se opravuje, ale nemůže si ještě zobrazit výsledek. Ten až po zveřejnění výsledků vyučujícím. Oba tyto úkony může vyučující provést buď u konkrétního testu, nebo v přehledu všech instancí daného testu. Taktéž je pro jednodušší zveřejnění naimplementována funkce hromadného zveřejnění všech již opravených testů. Tuto akci je možné provést stisknutím příslušného tlačítka v přehledu instancí testu.

8.4 Přehled testů

Přehled testů má k dispozici jak vyučující, tak žák, přičemž každému z nich se zobrazuje v odlišné formě. Vyučující má v přehledu testů testy, které vytvořil. U každého z nich má možnost zobrazit kód testu v podobě řetězce i v podobě QR kódu v modálním okně, což slouží k tomu, aby mohl kód například promítnout žákům pomocí projektoru na plátno a ti si jej buď opsali do aplikace, nebo test načetli vyfocením QR kódu. Dále má možnost test upravit, smazat či zobrazit všechny vyplněné testy k danému testu. Žák vidí v přehledu testů všechny testy, které spustil a do každého z nich může vstoupit. To, zda po vstupu do testu se zobrazí test otevřený, odevzdaný či vyhodnocený závisí na stavu, který je u každé instance testu evidován.

9 Ověření funkcionality

Ověření funkcionality patří mezi nezbytnou součást vývoje jakéhokoliv softwaru. Jeho cílem je nejen odhalit chyby, které mohou mít vliv na funkčnost i bezpečnost softwaru, ale také získat názor uživatelů na vytvořený produkt.

9.1 Ověřování během vývoje

První fází ověřování bylo ověřování během vývoje. Pro debugování frontendu jsem využil nástroj *React Developer Tools*, který je doporučovaný jako nejlepší nástroj pro odladění webové aplikace implementované v Reactu. Nástroj dokáže přímo ve webovém prohlížeči zobrazit jednotlivé použité komponenty, jejich stromové uspořádání, hooky s nimi spjaté a mnoho dalšího, to vše v aktuálním stavu. Kromě toho upozorní na veškeré chyby jako například čtení dat z nedefinované proměnné či čtení dosud nezískaných dat, detailně je specifikuje a poskytne informace o tom, v jaké komponentě a na jakém řádku zdrojového kódu se tento problém nachází. Díky tomuto nástroji byly odhaleny problematické části na frontendu a ošetřeny tak, aby k problémům v produkční verzi aplikace nedocházelo. Nástroj jsem používal po celou dobu psaní kódu, díky čemuž byly problémy odhaleny včas a nedocházelo k jejich kumulování.

Pro testování backendu jsem zvolil nástroj *Postman*, který slouží mimo jiné jako API klient. S jeho pomocí je možné velmi jednoduše a rychle vytvářet HTTP požadavky s nastavením různých způsobů autorizace, hlavičky i samotného těla v různých formátech. Po odeslání požadavku se zobrazí stavový kód, data získaná v odpovědi, velikost dat a čas spotřebovaný k vyřízení požadavku. Díky těmto údajům bylo možné jednotlivé požadavky optimalizovat a zajišťovat u nich dostatečnou bezpečnost. Tento nástroj také umožňuje automatizaci manuálního testování pomocí vytváření kolekcí požadavků, což výrazně urychluje čas strávený testováním, protože jednotlivé požadavky v rámci kolekce se provádějí automaticky. U každého požadavku je zároveň možné napsat javascriptový test, který se po získání odpovědi ze serveru provede. S využitím tohoto nástroje bylo testování backendu velmi efektivní a pohodlné.

Kromě ověřování aplikace pomocí zmíněných nástrojů jsem taktéž manuálně ověřoval správnou logiku a chování aplikace, které by bylo velmi složité zkoumat pomocí testů. Vždy po dokončení určitého funkčního celku aplikace jsem vyzkoušel všechny možné případy chování, které u dané komponenty či daného celku mohou nastat. Zaměřoval jsem se především na správné aktualizování stavů, sdílení dat mezi komponentami a vizuální chování jednotlivých prvků.

9.2 Uživatelské ověření výsledného produktu

Po dokončení ověřování během vývoje jsem realizoval ověření výsledného produktu uživateli. Každý respondent dostal možnost vyzkoušet si jak rozhraní pro vyučující, tak rozhraní pro žáky, přičemž byl požádán o vytvoření zkušebního testu v rozhraní pro vyučujícího, jeho vyplnění v rozhraní pro žáka, jeho vyhodnocení v rozhraní pro vyučujícího a jeho zobrazení v rozhraní pro žáka. Všem respondentům byl před začátkem zkoušení aplikace poskytnut seznam hlavních vlastností, kterými aplikace disponuje, přičemž jim nebylo nijak upřesněno, jak se tyto vlastnosti chovají a kde je najdou, aby bylo možné ověřit, jak si respondenti poradí s orientací v aplikaci. Respondentům byl poskytnut neomezený čas pro vyzkoušení aplikace. Následně byl s každým respondentem veden polostrukturovaný rozhovor, což je metoda patřící do skupiny výzkumných rozhovorů, která se vyznačuje tím, že tazatel má připravenou základní kostru témat a otázek, na které se chce zpovídaného ptát, avšak zároveň má možnost se od nich odklonit v reakci na odpovědi zpovídaného. Výhodou této metody je především kombinace přehlednosti a flexibility.

Jednotlivá témata a otázky byly pokládány respondentům v takovém pořadí, v jakém si měli aplikaci vyzkoušet. Nejprve byly kladeny otázky týkající se vytváření testu. Všichni respondenti se shodli na tom, že vytváření testu je snadné. Cenili si především toho, že nastavení parametrů testu je na první pohled viditelné a není nutné jej složitě hledat. Jeden z respondentů, který je vyučující, vyzdvihl možnost otázky typu doplňování, kterou v dosud vyzkoušených existujících řešeních postrádá nejen on, ale i jeho kolegové napříč různými předměty. Taktéž pozitivně ohodnotil, že jsou nabízeny v aplikaci pouze typy otázek, které jsou opravdu ve školství využívány, a že nabídka není tak jako v mnohých existujících řešeních doplněna o typy otázek využitelných pouze v obecných formulářích, nikoliv ve školství. Další otázky směřovaly na možnost multimediálního obsahu ve formuláři. Všichni respondenti si této možnosti všimli, vyzkoušeli si ji a byli překvapeni možností vložení obrázků i videí bez nutnosti nahrávání. Velmi příznivě ocenili také nástroj pro vytvoření náčrtku přímo v aplikaci. Na závěr této části dostali respondenti otázku týkající se způsobu zadávání hodnotících kritérií. Někteří respondenti neporozuměli principu hodnotících kritérií, což s velkou pravděpodobností mohlo být způsobeno nejen jejich neznalostí pohledu vyučujícího, ale i nedostatečným označením a popisem v aplikaci, což je ve finální verzi upraveno. Po vysvětlení byli s principem fungování hodnotících kritérií spokojeni.

Další část rozhovoru se týkala spuštění a vyplnění testu z pohledu žáka. Možnost spuštění testu pomocí unikátního kódu či QR kódu bez nutnosti sdílení odkazu se těšila mezi respondenty velkému úspěchu. K samotnému průběhu testu neměli žádné připomínky. V této části rozhovoru jsem se mimo jiné zaměřil na časový limit, který byl velmi žádanou funkcí. Většina respondentů se shodla na tom, že umístění odpočítávání času je nerušivé a vhodně umístěné, protože je kdykoliv vidět a není potřeba se posouvat po obrazovce. Zajímal jsem se také o jejich emoční reakce související s časovým limitem, který by mohl způsobovat zbytečný stres. Zde

převažoval názor, že na jednu stranu je časový limit stresující, ale na druhou stranu poskytuje odpočítávání možnost okamžitě vidět, kolik času zbývá. Jeden z respondentů uvedl příklad z praxe, kdy při písemné zkoušce ve třídě vyučující řekl, že na test bude 20 minut a poté oznámil 2 minuty před koncem testu, že do odevzdání zbývají 2 minuty, což neumožňovalo zavčas zareagovat a vhodně si rozvrhnout čas na vyplnění testu. Taktéž některé třídy podle slov respondenta nemají hodiny či na ně není ze zadních řad třídy vidět, což výrazně ztěžuje orientaci v čase.

Ve třetí části rozhovoru jsem respondentům pokládal otázky, které se týkaly vyhodnocení testu z pohledu vyučujícího. Pro lepší přiblížení praktické podoby jsem všem ukázal, jak by vypadal přehled vyplněných testů při větším počtu žáků. Rozhraní pro zobrazení vyplněných testů i pro vyhodnocení konkrétního testu respondentům připadalo přehledné a jednoduché, líbilo se jim použití moderních prvků v podobě přepínačů místo běžných tlačítek. Taktéž pro většinu respondentů byla vizualizace správnosti v podobě barevných pruhů u otázek vnímána jako originální nápad, který přijali velmi dobře. Automatizované hodnocení považují všichni respondenti za velký přínos aplikace. Tuto funkci jsem rozebral důkladně s jedním z respondentů, který je vyučující. Ten vyzdvihl především to, že je možné automatizovaně opravit veškeré nabízené typy otázek, což v praxi výrazně zrychluje vyhodnocení testů. U otevřených otázek pochválil možnost pokročilého nastavení hodnocení snadnou formou s tím, že některé z již existujících aplikací nabízejí pokročilé nastavení pomocí regulárního výrazu, který je ale pro většinu vyučujících něco, čemu nerozumí, a proto pak pokročilé nastavení nevyužívají. Také ocenil propojení automatizace s předdefinovanými hodnotícími kritérii, díky čemuž celý systém automatizovaného hodnocení získává podobu plně využitelnou v praxi.

Čtvrtá část rozhovoru se týkala zobrazení opraveného testu z pohledu žáka. Tak jako v případě vyhodnocení testu, respondenti zde ocenili zejména barevné pruhy značící úspěšnost, resp. neúspěšnost u každé otázky. Také vyzdvihli přehledné zobrazení všech absolvovaných testů. Několik respondentů se shodlo v tom, že jim zcela vyhovuje možnost mít všechny testy přehledně na jednom místě s tím, že si je mohou kdykoliv zpětně dohledat, otevřít a projít si své odpovědi.

Poslední část rozhovoru byla věnovaná celkovým pocitům z práce s aplikací a případným postřehům, připomínkám či dotazům. Všichni respondenti uvedli, že se jim s aplikací pracovalo příjemně a plynule a že by si dokázali představit její využití v praxi. Několik respondentů vyzkoušelo práci s aplikací na dotykovém displeji a byli s ní taktéž spokojeni. Jeden z respondentů vyjádřil přání v podobě instalovatelné aplikace pro mobilní zařízení, což je zároveň jedno z plánovaných možných rozšíření tohoto projektu.

10 Závěr

Cílem mé diplomové práce bylo navrhnout a implementovat aplikaci pro online testování žáků na základních a středních školách s využitím moderních přístupů a technologií k vývoji. V teoretické části jsem se věnoval problematice vzdělávání v didaktickém systému a využívání digitálních technologií ve vzdělávání, aplikačnímu softwaru a především moderním přístupům k vývoji webových aplikací. V rámci praktické části jsem analyzoval požadavky na tuto aplikaci, vytvořil návrh řešení a provedl implementaci řešení. Po dobu celé implementace probíhalo ověřování funkcionality, přičemž završeno bylo ověřením dokončené aplikace samotnými uživateli.

Výsledkem této diplomové práce je moderní webová aplikace umožňující online testování žáků, která je určena primárně pro základní a střední školy. Aplikace sestává z rozhraní pro vyučující a z rozhraní pro žáky. Vyučující mohou vytvářet nové testy a upravovat již vytvořené testy, zobrazovat žáky odevzdané testy a vyhodnocovat je, a to buď manuálně, anebo s využitím automatizovaného vyhodnocení. Žáci mohou testy na základě unikátního kódu či QR kódu spouštět, vyplňovat, následně odevzdat vyučujícímu k vyhodnocení a posléze si opravené testy zobrazit. V celé aplikaci je brán zřetel na nenadálé situace v podobě výpadku připojení k internetu či selhání zařízení, na kterém je test vyplňován, a to pomocí pravidelného automatického ukládání a možnosti návratu ke spuštěnému testu. Během návrhu i implementace byl kladen důraz především na to, aby byla aplikace rychlá, přehledná, poskytovala všechny potřebné a žádané funkce a zároveň si udržela jednoduchost používání. Na základě reakcí respondentů, kteří se na ověření dosažených výsledků podíleli, se zdá, že tohoto cíle bylo dosaženo. Výsledná aplikace je dostupná na webové adrese https://app.examingo.cz, kde je možné si ji vyzkoušet.

Vzhledem k tomu, že se téma online testování stalo za poslední rok velmi populární a dostupných kvalitních nástrojů není rozsáhlé množství, aplikace může mít velký potenciál z hlediska budoucího rozvoje. V následujících odstavcích zmíním některá důležitá rozšíření, která by mohla být pro aplikaci přidanou hodnotou a podílet se na zvýšení komfortu jejího použití, ale i na vylepšení nabízených funkcí.

Prvním z možných rozšíření je vytvoření nativní mobilní aplikace, kterou by mohli využít jak žáci pro vyplňování testů, tak vyučující pro jejich vytváření. Ačkoliv je současná webová aplikace responzivní a optimalizovaná pro mobilní zařízení a jejímu použití na těchto zařízeních nic nebrání, nativní aplikace by dokázala přinést výhodu v podobě nainstalované aplikace, kterou bude možné snadno spustit bez použití prohlížeče a která bude moci využívat funkce telefonu. Mohla by například obsahovat zabudovanou čtečku QR kódu využívající fotoaparát mobilního telefonu. Zvolený framework React pro implementaci frontendu přináší zásadní výhodu pro vytvoření takové aplikace, jelikož existuje framework React Native, který z Reactu vychází. Pro vytvoření nativní aplikace na základě již existující webové aplikace v Reactu stačí porozumět některým specifikům a za relativně krátký čas je možné vytvořit nativní aplikaci.

Se vznikem nativní mobilní aplikace souvisí i druhé možné rozšíření, kterým jsou notifikace. I když je možné notifikace využívat i ve webových aplikacích, jejich oblíbenost lze pozorovat především u mobilních aplikací. Jako vhodný způsob notifikací bych zvolil tzv. push notifikace, pomocí kterých je možné zasílat uživateli krátká oznámení. Notifikace by své uplatnění v aplikaci našly zejména v situacích, kdy vyučující zveřejní výsledky vyhodnoceného testu. Žák by díky tomu měl okamžitou informaci, že byl jeho test vyhodnocen. Do těchto zpráv by bylo možné zahrnout i získané výsledné hodnocení, což by umožnilo žákovi, aby si vybral, zda mu stačí vědět získaný výsledek, či zda chce vidět vyhodnocený test a kliknutím na notifikaci se přesune do mobilní aplikace.

Třetím možným rozšířením této aplikace je zavedení podpory exportu výsledků testů žáků do formátu, který podporují elektronické žákovské knížky pro import dat. Takové rozšíření by mělo přínos především pro vyučující, kteří by si vybrali, pro který typ elektronické žákovské knížky chtějí data exportovat, a následně po stažení dat z aplikace by je mohli importovat do elektronické žákovské knížky bez nutnosti manuálního zadávání.

Závěrem bych rád zmínil, že původní myšlenkou této diplomové práce bylo vytvořit aplikaci, která se bude používat v rámci prezenční výuky jako náhrada písemných testů. Tato myšlenka vznikla v době, kdy nikdo z nás nepředpokládal, jakým směrem se kvůli pandemii koronaviru školství ubere. Je tedy předpokládáno, že žáci s aplikací budou pracovat v učebně pod dohledem vyučujícího a není jejím cílem zajistit, že nebudou při jejím používání podvádět. Otázkou je, zda by to vůbec bylo možné podchytit. Bylo by možné vynutit režim celé obrazovky či zakázat některé klávesy, bylo by možné zajistit sdílení okna žáka, ve kterém test vyplňuje, bylo by možné zasílat vyučujícímu informace o podezřelém chování žáka, nic z toho by ale nebylo možné dokázat a ani by nebylo možné zajistit, že žák nebude používat více monitorů či další zařízení. Tak jako není možné nechat žáky ve třídě při vypracovávání písemné zkoušky samotné bez dozoru, tak ani není možné nechat žáky vyplňovat online test bez dozoru. Za zmínku stojí také fakt, že online testy budou mít vždy určité limity, které nelze překonat. Přestože ve většině předmětů je jejich použití možné, například pro rýsování v geometrii je využít prakticky nelze. Testy tohoto typu mají také omezení v tom, že s jejich pomocí vyučující dokáže ověřit pouze určitou oblast látky do určité hloubky, nepozná však, zda má žák mezi probíranými tématy souvislosti a jak dané látce komplexně rozumí.

11 Literatura

BARTON, Zach. *What is a Cloud Application?* Cloudbakers.com [online]. 2018 [cit. 2020-10-16]. Dostupné z: https://www.cloudbakers.com/blog/what-is-a-cloud-application.

DAVID, Matthew. *5 back-end frameworks for better cross-platform development*. Techbeacon.com [online]. 2020 [cit. 2020-11-20]. Dostupné z: https://techbeacon.com/app-dev-testing/5-back-end-frameworks-better-cross--platform-development.

DOSEDĚL, Tomáš. *Ani koronavirus revoluci do školství nepřinesl. Dočkáme se někdy?*. Svetchytre.cz [online]. 13.5.2020 [cit. 2021-02-07]. Dostupné z: https://www.svetchytre.cz/a/pcNq2/ani-koronavirus-revoluci-do-skolstvi--neprinesl-dockame-se-nekdy.

FOSTER, Elvis C. a Shripad GODBOLE. *Database Systems: A Pragmatic Approach*. New York: Apress, 2014. ISBN 978-1-4842-0878-6.

GOEL, Aman. *10 Best Web Development Frameworks*. Hackr.io [online]. 2020 [cit. 2020-11-20]. Dostupné z: https://hackr.io/blog/top-10-web-development-frameworks-in-2020.

HAKES, Tyler. *The Paperless School: 9 Ways to Reduce Waste and Increase Efficiency*. Frevvo.com [online]. 2020 [cit. 2021-02-02]. Dostupné z: https://www.frevvo.com/blog/paperless-school/.

CHHENG, Tommy. *The Non-Technical Guide to Web Technologies*. United States: CreateSpace, 2013. ISBN 1492791539.

JAZAYERI, Mehdi. *Some Trends in Web Application Development*. In: Future of Software Engineering (FOSE '07) [online]. IEEE, 2007, 2007, s. 199-213 [cit. 2020-10-19]. ISBN 0-7695-2829-5. Dostupné z: doi:10.1109/FOSE.2007.26.

JENSEN, Paul B. *Cross-platform desktop applications: using Electron and NW.js*. Shelter Island, NY: Manning Publcations Co., [2017]. ISBN 9781617292842.

JERELYN, Amanda. *Why dark mode web designs are gaining popularity* Searchenginewatch.com [online]. 2020 [cit. 2021-02-01]. Dostupné z: https://www.searchenginewatch.com/2020/09/30/why-dark-mode-web--designs-are-gaining-popularity/.

KOLÁŘ, Zdeněk a Renata ŠIKULOVÁ. *Hodnocení žáků*. 2. dopl. vyd. Praha: Grada, 2009. Pedagogika (Grada). ISBN 978-80-247-2834-6.

KOURAKLIS, John. *MVVM in Delphi: Architecting and Building Model View ViewModel Applications*. New York: Apress, 2016. ISBN 978-1-4842-2214-0.

KURILO, Andrew. *Desktop App vs Web App: Comparative Analysis*. Digital Skynet [online]. 2020 [cit. 2020-10-02]. Dostupné z: https://digitalskynet.com/blog/Desktop-App-vs-Web-App-Comparative--Analysis.

LAUBHEIMER, Page. *The Role of Animation and Motion in UX* Nngroup.com [online]. 2020 [cit. 2021-02-01]. Dostupné z: https://www.nngroup.com/articles/animation-purpose-ux/.

LOVEČEK, Aleš a Miroslav ČADÍLEK. *Didaktika odborných předmětů* [online]. Brno, 2005 [cit. 2021-02-05]. Dostupné z: http://boss.ped.muni.cz/vyuka/material/puvodni/skripta/dop/didodbpr.pdf. Skripta. Masarykova univerzita, Pedagogická fakulta, Katedra didaktických technologií.

MALI, Nabeena. *What Is A Push Notification? The Ultimate Guide to Push Notifications* Appinstitute.com [online]. 2018 [cit. 2021-02-01]. Dostupné z: https://appinstitute.com/what-is-a-push-notification/.

MCFARLAND, David Sawyer. *JavaScript & jQuery: The Missing Manual*. 3rd ed. United States: O'Reilly Media, 2014. ISBN 9781491947074.

PANHALE, Mahesh. *Beginning Hybrid Mobile Application Development*. New York: Apress Media, 2016. ISBN 978-1-4842-1315-50.

PERKNEROVÁ, Kateřina. *Učení online: Ukázalo hned několik Achillových pat českého školství*. Denik.cz [online]. 7.12.2020 [cit. 2021-02-02]. Dostupné z: https://www.denik.cz/co-covid-vzal-dal-skoly/uceni-online-skoly-covid--20201207.html.

PETTY, Geoffrey. *Moderní vyučování*. 5. vyd. Praha: Portál, 2008. ISBN 978-80-7367-427-4.

ROUSE, Margaret. *What is application?*. WhatIs.com [online]. 2018 [cit. 2020-04-14]. Dostupné z: https://searchsoftwarequality.techtarget.com/definition/application.

RUSSELL, Alex. *Progressive Web Apps: Escaping Tabs Without Losing Our Soul* Infrequently.org [online]. 2015 [cit. 2021-01-29]. Dostupné z: https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without--losing-our-soul/.

SCOTT, Emmit. *SPA Design and Architecture: Understanding single-page web applications*. Shelter Island: Manning, 2016. ISBN 978-161-7292-439.

SHARMA, Vatika a Meenu DAVE. *SQL and NoSQL Databases*. International Journal of Advanced Research in Computer Science and Software Engineering [online]. 2012, volume 2, issue 8, s. 20-27 [cit. 2021-03-24]. ISSN 2277 128X. Dostupné z: https://www.academia.edu/25479961/SQL_and_NoSQL_Databases.

SKALKOVÁ, Jarmila. *Obecná didaktika: vyučovací proces, učivo a jeho výběr, metody, organizační formy vyučování*. 2., rozšířené a aktualizované vydání. Praha: Grada, 2007. Pedagogika (Grada). ISBN 978-80-247-1821-7.

SUCHOMELOVÁ, Eva. *Moderní technologie a metody ve vzdělávání - výukové pomůcky*. In: Hospodářské noviny [online]. 14.9.2017 [cit. 2021-02-02]. Dostupné z: https://ihned.cz/pro-absolventy/c1-65881300-moderni--technologie-a-metody-ve-vzdelavani-vyukove-pomucky.

TOBIN, Donal. *Which Modern Database Is Right for Your Use Case?* Xplenty.com [online]. 2020 [cit. 2020-12-04]. Dostupné z: https://www.xplenty.com/blog/which-database/.

UZAYR, Sufyan Bin, Nicholas CLOUD a Tim AMBLER. *JavaScript Frameworks for Modern Web Development: The Essential Frameworks, Libraries, and Tools to Learn Right Now*. Second Edition. New York: Apress Media, 2019. ISBN 978-1-4842-4994-9.

XANTHOPOULOS, Spyros a Stelios XINOGALOS. *A comparative analysis of cross-platform development approaches for mobile applications*. In: Proceedings of the 6th Balkan Conference in Informatics on - BCI '13 [online]. New York, New York, USA: ACM Press, 2013, 2013, s. 213-[cit. 2020-10-09]. ISBN 9781450318518. Dostupné z: doi:10.1145/2490257.2490292.

Seznam obrázků

Obrázek 1: Ukázka otázky z didaktického testu z českého jazyka
Zdroj: https://www.statnimaturita-cestina.cz/didakticky-test/didakticky--test-2020 . . . 11
Obrázek 2: Ukázka krabicového softwaru
Zdroj: http://www.buildorbuy.org/themes/800/win98box.jpg . . . 18
Obrázek 3: Ukázka chování progresivní webové aplikace čínského online tržiště AliExpress
Zdroj: https://www.tigren.com/examples-progressive-web-apps-pwa/ . . . 24
Obrázek 4: Ukázka chatbota OrderNow pro restaurace a hospody
Zdroj: https://blog.talfa.cz/zajimave-ceske-messenger-chatboti/ . . . 26
Obrázek 5: Ukázka push notifikace
Zdroj: https://appinstitute.com/what-is-a-push-notification/ . . . 28
Obrázek 6: Porovnání životního cyklu multi-page a single-page aplikace
Zdroj: https://dzone.com/articles/the-comparison-of-single-page-and-multi--page-appli . . . 31
Obrázek 7: Vztah mezi třívrstvou architekturou a návrhovým vzorem MVC
Zdroj: (Kouraklis, 2016) . . . 34
Obrázek 8: Vztah mezi třívrstvou architekturou a návrhovým vzorem MVP
Zdroj: (Kouraklis, 2016) . . . 35
Obrázek 9: Vztah mezi třívrstvou architekturou a návrhovým vzorem MVVM
Zdroj: (Kouraklis, 2016) . . . 36
Obrázek 10: Google Trends . . . 39
Obrázek 11: Přehled možností struktur relačních a nerelačních databází
Zdroj: https://openclassrooms.com/en/courses/5671741-design-the-logical--model-of-your-relational-database/6255746-compare-relational-and-nosql--databases . . . 44
Obrázek 12: Use case diagram . . . 52
Obrázek 13: NoSQL model databáze . . . 64
Obrázek 14: Schéma principu komunikace . . . 67
Obrázek 15: Ukázka podepsaného zašifrovaného a dešifrovného tokenu z aplikace . . . 69
Obrázek 16: Rozhraní pro vytváření a úpravu testu . . . 74
Obrázek 17: Modul doplňovacího cvičení . . . 75
Obrázek 18: Možnosti nastavení otázky . . . 76
Obrázek 19: Ukázka multimediálního obsahu v podobě náčrtku . . . 77
Obrázek 20: Nastavení hodnotících kritérií . . . 77
Obrázek 21: Načtení testu . . . 78

Obrázek 22: Vyplňování testu 79
Obrázek 23: Přehled instancí testu 80
Obrázek 24: Vyhodnocení testu 80
Obrázek 25: Automatizované vyhodnocení testu 81