Aplikácia pre tvorbu testov

Technická zpráva k projektu do předmětu ITU
FIT VUT v Brně, 2020

Název týmu
Itučkári

Autoři
Eva Moresová, xmores01
Jakub Richtarik, xricht29
Jiří Žák, xzakji02

Obsah

1. Zadání a organizace týmu

1.1 Cíl

Cieľom projektu je vytvoriť aplikáciu pre tvorbu testov. Aplikácia bude umožňovať intuitívne vytváranie testových otázok rôzneho typu, ich kategorizovanie a úpravu po uložení, s možnosťou pridania ukážky (obrázok, alebo textová ukážka). Vytvorený test bude možné uložiť a exportovať do PDF pre prípadnú tlač. Taktiež bude možné vygenerovať pre vytvorený test viacero testovacích skupín, vytvoriť a vytlačiť šablónu so správnymi odpoveďami. Užívateľ si bude môcť zobraziť predošlé vytvorené testy a otázky. Aplikáciu bude možné v budúcnosti napojiť na väčší systém pre online testovanie študentov s automatickou opravou testov.
Aplikácia je primárne určená pre učiteľov základných a stredných škôl. Bude slúžiť na urýchlenie a uľahčenie procesu tvorby testov a tiež ich prehľadnú správu.

1.2 Tým

Eva Moresová	- dizajn a tvorba desktopovej aplikácie, práca s api
Jakub Richtarik	- návrh a tvorba backendu (MVC), bočný panel a responzivita webovky
Jiří Žák	- dizajn a tvorba webovej aplikácie, práca s api

1.3 Roadmapa

1. Rozdelenie zodpovedností v rámci tímu a výber technológií, ktoré budú použité. (Kapitoly: 1.2, 3.4, Termín: 6.10.)
2. Prieskum užívateľských potrieb (na starosti Jakub Richtarik) a spoločný návrh dátového modelu aplikácie (štruktúry zobrazovaných a ukladaných dát). (Kapitoly: 2.2, 3.3, Termín: 10.10.)
3. Každý člen si pripraví návrh vzhľadu (štruktúry) GUI (zahŕňa vytvorenie makety a uskutočnenie pilotných testov), prezentuje návrh ostatným členom tímu, spoločne sa vyberie a prípadne ešte upraví ten najlepší. (Kapitoly: 4.*, Termín: 14.10.)
4. Návrh architektúry aplikácie, prepojenie jednotlivých častí (backend s webovým a desktopovým GUI) a návrh backendového API. (Kapitoly: 3.1, 3.4, Termín: 16.10.)
5. Implementácia jednotlivých častí, počas ktorej sa členovia tímu synchronizujú minimálne v týždňových intervaloch. (Kapitoly: 5.1, 5.2, Termín: 6.12.)
6. Finálne testovanie vytvorených aplikácií. (Kapitoly: 5.3, 5.4, Termín: 13.12.)

1.4 Rizika a opatření

Medzi najväčšie riziká, ktoré môžu nastať pri realizácii projektu patria nižšie uvedené. Všetky z nich sme sa snažili čo najvia eliminovať.

1. neznalosť vlastného tímu - je lepšie pracovať s ľuďmi, ktorých poznáme, aby sme vedeli, čo od nich môžeme očakávať a či sa na nich môžeme spoľahnúť
2. výpadok člena tímu - riešením je snažiť sa vyhotoviť produkt v predstihu (pri výpadku je čas na dorábku zvyšnými členmi) a rozvrhnúť prácu rovnomerne, aby nestál projekt na 1 osobe
3. nesúhlas v tíme - rozhodovanie hlasovaním (demokracia v praxi), robenie kompromisov, stratégia win-win
4. problémy v komunikácii - z dôvodu COVID-u nemožnosť osobných mítingov, preto pravidelné videohovory s ich nahrávaním, aby sa členovia tímu mohli neskôr vrátiť k tomu, čo zaznelo
5. neznalosť technológií - dá sa vyriešiť len naštudovaním

2. Průzkum a zkušenosti

2.1 Existující řešení

Survio (Jiří Žák)

Rozsáhlejší webová aplikace na tvorbu testů či dotazníků. Aplikace nabízí opravdu hodně věcí, které už nemají s tvorbou testů moc společného. Ovládání je překvapivě jednoduché a intuitivní. Bohužel pokud si uživatel neplatí premium účet, tak je omezen pouze na 100 odpovědí na měsíc.

Testedit (Jiří Žák)

Webová stránka nabízející implementaci přímo do vlastních stránek. Bohužel za tuto funkci se platí. Demo aplikace, kterou zde mají k vyzkoušení, pak pouze ukáže jak vypadají hotové testy. Je to opět velice jednoduché a přehledné, takže uživatel by neměl mít problém se v tom vyznat a případně si i vyzkoušet předpřipravený test.

Daypo (Eva Moresová)

Služba je dostupná ako webová stránka aj desktopová aplikácia. Poskytuje veľa druhov otázok, nielen otázky s možnosťami alebo s otvorenou otázkou a možnosť rôzneho bodovania otázok. Prostredie je jednoduché a poskytuje iba potrebné funkcie. Avšak kvôli zastaralému vzhľadu je aplikácia neprehľadná. Niektoré tlačidlá neobsahujú text, iba ikonu, z ktorej novému užívateľovi nie je hneď jasná ich funkcia.

E-kurz (Eva Moresová)

Aplikáciu je možné využívať cez webovú stránku alebo stiahnuť do mobilu. Vytváranie testov je jednoduché a prehľadné, testy je možné triediť do priečinkov. Dizajn stránky je moderný a

prívetivý. Dajú sa ale vytvárať iba základné typy otázok, čo nemusí byť pre niektorých užívateľov postačujúce. Ďalšou nevýhodou je, že aplikácia je dostupná len po anglicky.

WebTesty.sk (Jakub Richtarik)

Ide o webovú aplikáciu s dobre prepracovaným dizajnom. Ďalšou výhodou je, že neobsahuje príliš veľa tlačidiel, kvôli ktorým by došlo k informačnému smogu pre používateľa.
Nevýhodou je určite mätúce ovládanie. Občas je totiž z veľkého tlačidla klikateľnou časťou len text. Ďalším príkladom je tlačidlo dole pod otázkou, pri kt. by človek očakával, že prejde k vytvoreniu ďalšej otázky, miesto toho slúži len na presuny medzi už vytvorenými otázkami. Navyše je užívateľ schopný vytvárať ďalšie otázky v teste, bez toho aby predchádzajúce uložil (príznak nikde nesignalizuje), a tak môže dôjsť k tomu, že keď je celý test hotový, používateľ neskôr zistí, že má vlastne 20 prázdnych otázok a k vyplneným dátam sa už nedostane.

ClassMarker (Jakub Richtarik)

Webstránka kladie dôraz na funkčnosť pred dizajnom. Obrovskou výhodou je, že dokáže vytvoriť v podstate akýkoľvek test. Nevýhodou je ale angličtina pre český a slovenský trh (nie každý učiteľ ZŠ a SŠ vie po anglicky). A ešte väčším negatívom je zahltenosť obsahom - jednoduchí učitelia, kt. nie sú až taký zdatní s počítačmi, by sa strácali vo veľkom počte podstránok a nastavení. Chýba preto nejaká možnosť voľby medzi jednoduchým a rozšírenejším rozhraním.

2.2 Uživatelské potřeby

Primárnymi užívateľmi budú učitelia základných a stredných škôl, no neskôr by sa systém mohol rozširovať do viacerých sfér (testy pre autoškoly, psychologické testy, atď...).
Z rozhovorov vyplynulo, že učitelia by boli vďační najmä za prehľadnú správu testov. Keďže niektorí si toľko neveria s aplikáciami (online testovanie), ocenia ich možnosť tlače, s odpoveďovým hárkom pre kontrolu. Aplikácia by nemala byť veľmi náročná s veľkým množstvom tlačidiel v ktorých sa stratia.
Asi najväčším prínosom by pre nich bola práve možnosť vkladania už použitých úloh do nových testov, kt. vo väčšine existujúcich riešení chýba.

Súhrn vlastností systému (čo by mal systém vedieť)

- vytvorenie kategórií, podkategórií, ich premenovanie a mazanie
- vyššie spomenuté pri testoch
- vytváranie otázok s písaním odpovedí, ale aj s výberom možností
- možnosť pridania ukážky textovej/obrázkovej, no v budúcnosti kvôli angličtinárom aj zvukovej, prípadne nejaký nástroj pre matematikárov na vkladanie vzorcov
- vkladanie už existujúcich úloh, prípadne informácia, v kt. testoch sa už nachádzajú
- generovanie skupín, odpoveďového hárku
- test by sa mal ukladať priebežne počas jeho tvorby, aby sa nestalo, že pri 20. úlohe stratíme internetové spojenie

2.3 Shrnutí

Existujúce riešenia väčšinou poskytujú buď všetky funkcie potrebné pri tvorbe testov a sú zložité na ovládanie alebo sú dostatočne jednoduché aby ich vedeli ovládať aj učitelia, ktorí sú “počítačovými laikmi” avšak nemajú potrebnú funkcionalitu. Tento nedostatok by sme sa pokúsili v našej aplikácii odstrániť. Dáme dôraz na vytvorenie aplikácie, ktorá bude jednoduchá a funkčná. Tak ako pri väčšine riešení, dizajn našej aplikácie by mal byť moderný a užívatelsky prívetivý. Existujúce aplikácie sú vo veľkej miere v angličtine, keďže však tvoríme aplikáciu pre slovenský/český trh, zvolíme jeden z týchto jazykov ako jazyk aplikácie.

3. Architektura řešení

3.1 Architektura systému

Systém bude mať nasledujúce časti:

1. databáza - (lokálne phpMyAdmin, na serveri PostgreSQL) uložené dáta, komunikácia cez kontroléry v Laravel[1] backende (api)
2. Laravel kontroléry - ukladanie dát do databázy, ich načítavanie a vytváranie rôznych pohľadov, komunikácia s aplikáciami v JSON formáte
3. prostredie pre tvorbu testov - ako webová, tak aj desktopová aplikácia budú nahrávať a upravovať informácie v databáze prostredníctvom api

3.2 Architektura aplikace/í

Desktopová aplikácia

Vzhľadom na zvolené technológie (WPF) sa aplikácia riadi návrhovým vzorom Model, View, ViewModel (MVVM), ktorý je pre WPF typický.

1. Model - Popisuje dáta, s ktorými aplikácia pracuje. O stave ovládacích prvkov nesmie nič vedieť.
2. View - Reprezentuje užívateľské rozhranie v jazyku XAML. Môže ísť o okno, stránku, alebo ovládací prvok.
3. ViewModel - Podobná funkcia ako Controller v MVC. Spojuje View s Modelom a drží stav aplikácie. S View je prepojený pomocou bindingu dát. Svoje dáta poskytuje v dátových štruktúrach, ktoré vyvolávajú pri ich zmene udalosti. Vďaka tomu sa nové dáta môžu zobraziť v užívateľskom rozhraní hneď, ako sa vo ViewModeli zmenia. Na základe stavu aplikácie filtruje dáta.

Webová aplikace

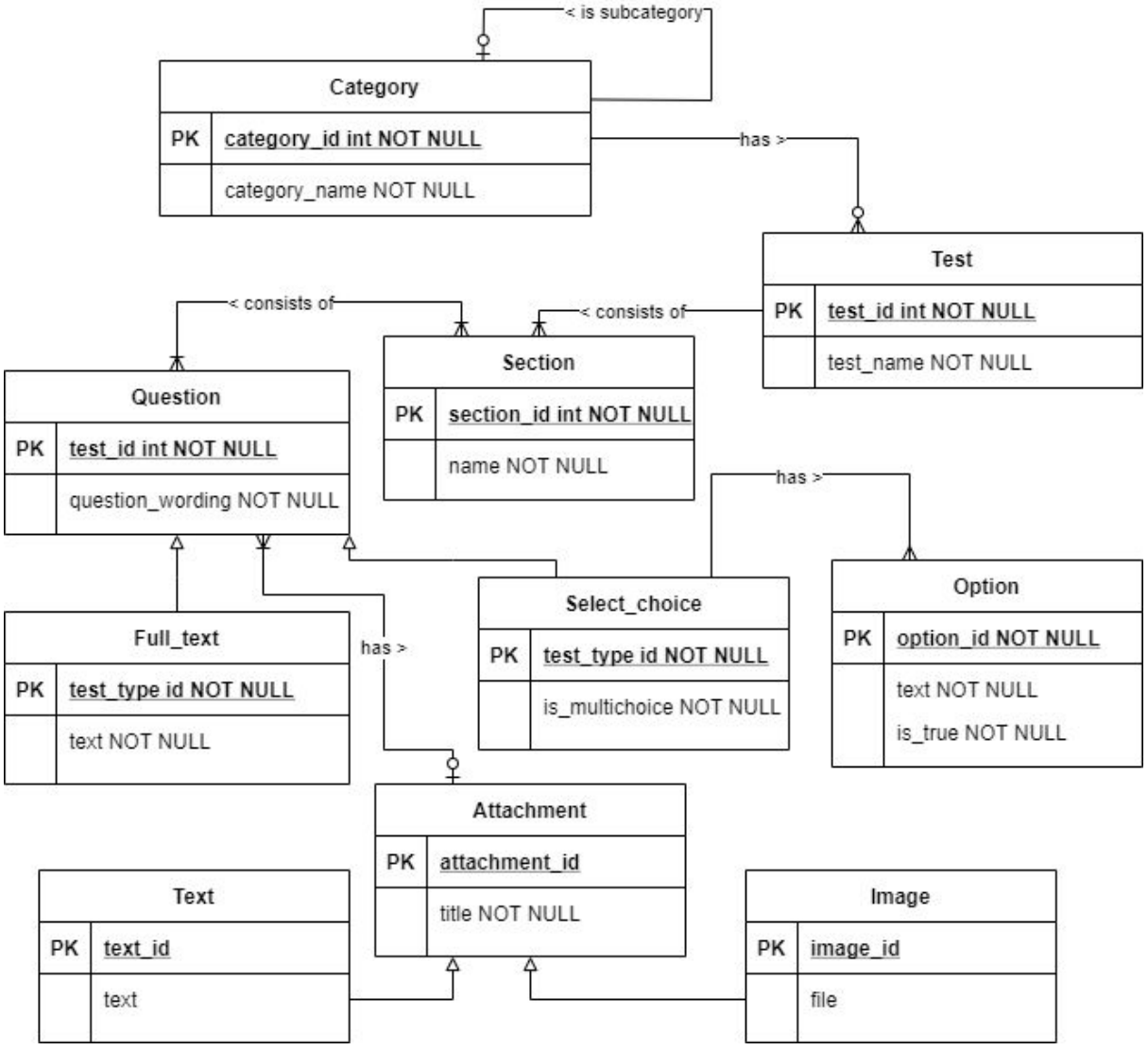
Při implementaci webové aplikace byl použitý návrhový vzor Model, View, Controller.

1. Model - popisuje data se kterými bude aplikace pracovat. O model se stará aplikační rozhraní.
2. View - zobrazuje aktuální stav aplikace a jeho data. Bude implementováno pomocí HTML, JSX, JSON a SASS.

[1] https://laravel.com/docs/8.x/

3. Controller - práci s datovým modelem a co zobrazí View zajistí Controller, který bude implementován pomocí moderního Javascriptu. Aplikace se bude měnit podle toho co udělá uživatel a pošle tento stav na View, který nové změny ihned zobrazí.

3.3 Datový model



Testy budú uložené v hierarchii kategórií a podkategórií, s nejakým názvom. Test pozostáva z 1, alebo viacerých sekcií (teória, dopĺňanie… alebo podľa celkov učiva). Pri každej budú uložené otázky (ich znenie) s plnou odpoveďou, alebo s jednotlivými možnosťami (1 správna, alebo viacero). Ku každej otázke bude možné pridať textovú prílohu (ukážku), alebo obrázok.

3.4 Vybrané technologie a implementace

Backend je realizovaný pomocou php frameworku Laravel, ktorý jednoducho umožňuje vytvárať modely, kontrolery, taktiež migráciu tabuliek (ich vytvorenie) do databázy. Ďalšou z

funkcionalít je naplnenie tabuliek vzorovými dátami cez seeder. Predpokladá sa implementácia autentizácie pomocou mena a hesla a autorizácia pomocou cookies.
Aplikácie komunikujú s modelom pomocou api, využívajúc čistú url ("Clean URLs", "RESTful URLs"), čiže url, kt. je user-friendly.

Method	URI
POST	attachments
GET|HEAD	attachments
DELETE	attachments/{attachment}
PUT|PATCH	attachments/{attachment}
GET|HEAD	attachments/{attachment}
POST	categories
GET|HEAD	categories
DELETE	categories/{category}
GET|HEAD	categories/{category}
PUT|PATCH	categories/{category}
POST	options
GET|HEAD	options
DELETE	options/{option}
PUT|PATCH	options/{option}
GET|HEAD	options/{option}
GET|HEAD	questions
DELETE	questions/{question}
GET|HEAD	questions/{question}
GET|HEAD	tests
POST	tests
DELETE	tests/{t}
PUT|PATCH	tests/{t}
GET|HEAD	tests/{t}
GET|HEAD	tests/{t}/sections
POST	tests/{t}/sections
PUT	tests/{t}/sections/{s}
DELETE	tests/{t}/sections/{s}
POST	tests/{t}/sections/{s}/questions
GET|HEAD	tests/{t}/sections/{s}/questions
PUT	tests/{t}/sections/{s}/questions/{question}
DELETE	tests/{t}/sections/{s}/questions/{question}
GET|HEAD	tests/{t}/sections/{s}/questions/{question}/attach

Desktopová aplikácia bude realizovaná pomocou Windows Presentation Foundation. Vývojová platforma WPF podporuje veľkú škálu funkcií pre vývoj aplikácií (aplikačný model, ovládacie prvku, dátové väzby, grafika a iné). Využíva jazyky XAML a C#.

Webová aplikace bude vytvořena především moderním Javascriptem, který se bude starat o hladký chod aplikace. K snadnější práci bude využívat JSX a JSON. Moderním Javascriptem dále zajistíme i přístup k api a práci s ním. Na stylování aplikace jsme zvolili SASS preprocessor.

4. Návrh GUI - webová aplikace [xzakji02]

4.1 Požadavky na GUI

Webová aplikace zobrazí menu s kategoriemi, podkategoriemi a testy. Uživatel bude moct vytvořit kategorii, podkategorii či test. Poté co klikne na vybraný test se mu daný test zobrazí a bude možné ho libovolně upravovat. Po uložení se vše přepošle do databáze a uživatel si ho může kdykoliv znovu zobrazit. Dizajn aplikace by měl být intuitivní a jednoduchý pro rychlé pochopení funkčnosti a snadné ovladatelnosti. Základem by mělo být použití jen pár barev. Aplikace by měla být single page.

4.2 Makety

4.3 Pilotní test

Tento návrh testovala studentka pedagogické fakulty ze které bude učitelka prvního stupně. Dostala za úkol vytvořit krátký test a jednu podkategorii. Bude sledováno jak rychle se uživatelka zorientuje v aplikaci a jakou dobu ji zabere zadané věci splnit. Dále vyhodnotíme, jestli je spokojená s návrhem a barevným dizajnem aplikace.

4.4 Vyhodnocení testu a revize návrhu

Uživatelka se překvapivě rychle zorientovala a vytvořit pár podkategorii ji zabralo jen několik minutek. Hlavní a stěžejní úkol, vytvořit krátký test, zabral více času, ale i tak to byl splněn rychleji než se původně očekávalo, takže to hodnotím kladně. Zato s dizajnem moc spokojená nebyla. Nelíbilo se jí rozvržení stránky a barevný návrh aplikace. Na tyto věci ohledně rozvržení a barevné škály bych se chtěl nejvíce zaměřit a co nejvíce je vyladit. Dále by chtěla mít možnost si test vyexportovat do pdf, což mě zaujalo a chtěl bych tuto funkci implementovat. Nejvíce byla uživatelka překvapená z nápadu single page aplikace, což ji velmi zaujalo a zhodnotila tento nápad kladně za což jsem velmi rád.

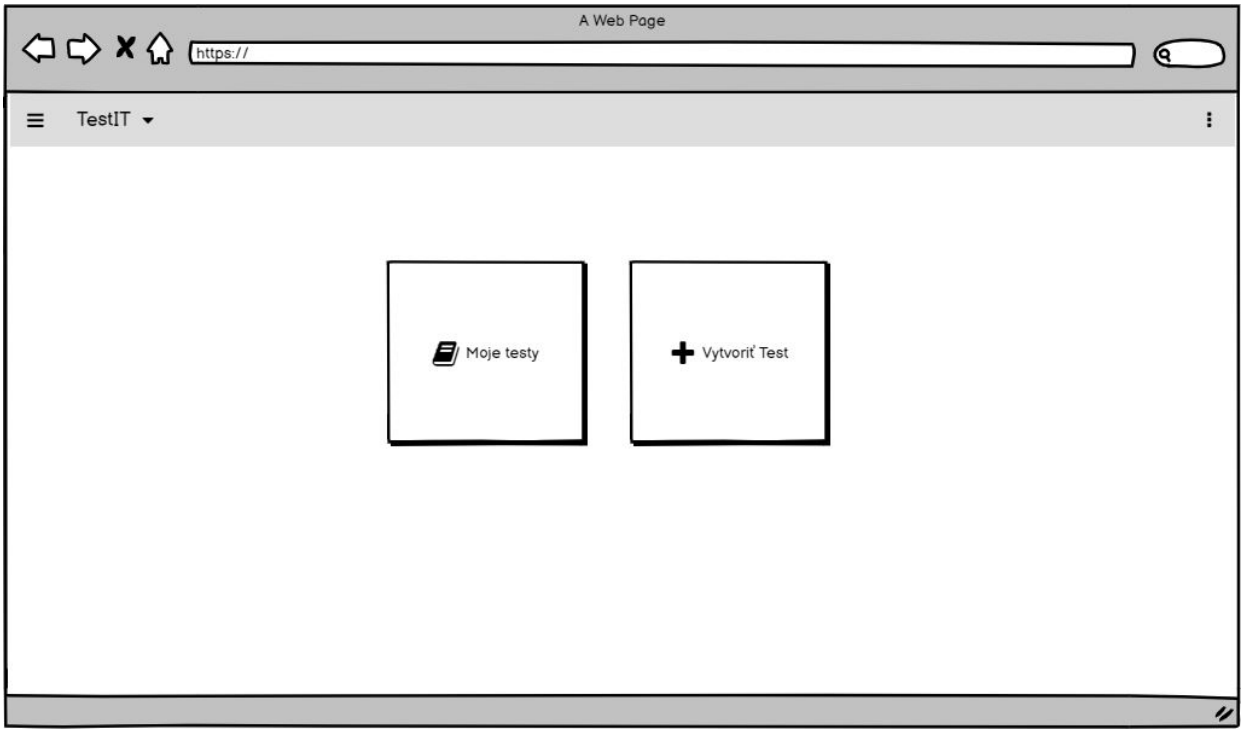
5. Návrh GUI - desktopová aplikace [xmores01]

5.1 Požadavky na GUI

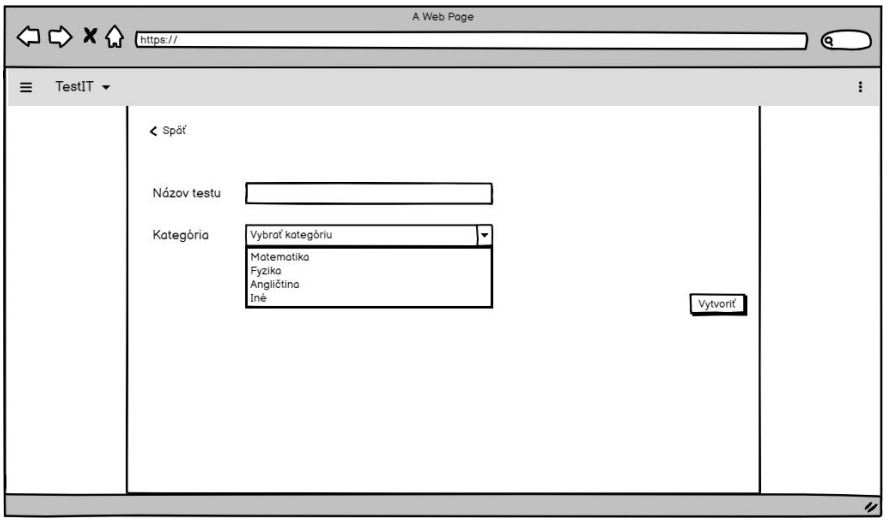
Užívateľ musí byť schopný jednoducho, rýchlo a bez školenia vytvoriť test. Dôležité je zobrazenie kategórií, prípadne podkategórií, do ktorých sú testy roztriedené - malo by byť prehľadné, jednotlivé kategórie jasne oddelené. Dizajn GUI by mal byť jednoduchý, prívetivý a moderný, farby uspôsobené tak, aby aj pri dlhšom používaní neunavovali oči užívateľa. Keďže ide o desktopovú aplikáciu, mala by mať jednoduchú inštaláciu a responzívny obsah.

5.2 Makety

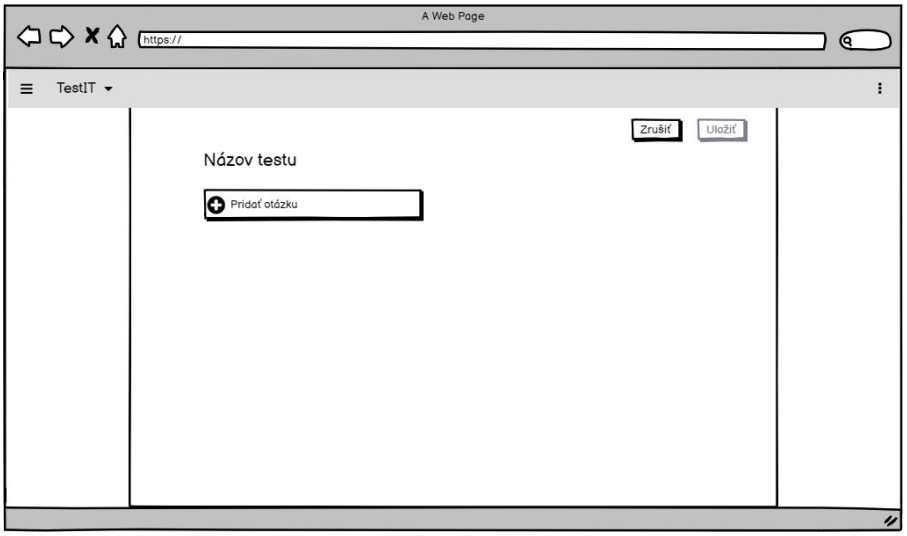
Úvodná obrazovka:

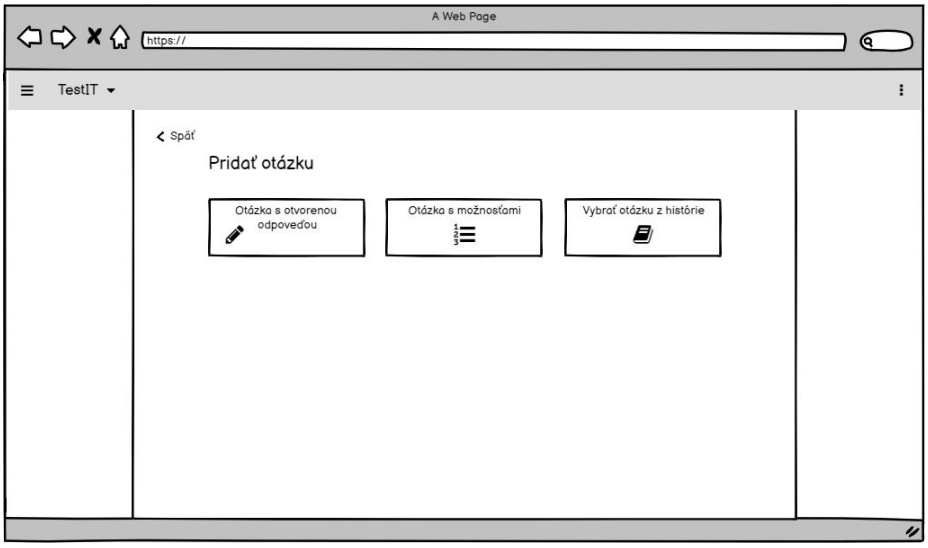


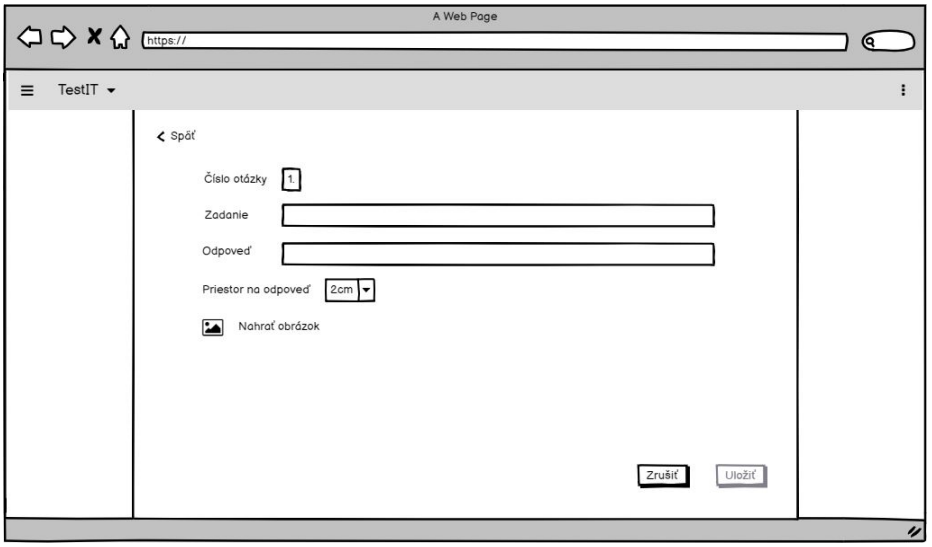
Prvá obrazovka pri vytváraní testu:



Pridanie otázky:







5.3 Pilotní test

Pilotný test bol realizovaný na stredoškolskej učiteľke angličtiny. Vytvorené makety mali definované akcie po kliknutí na určité prvky, pomocou čoho sa simulovala interakcia s aplikáciou. Úlohou bolo vytvoriť jednoduchý test s tromi otázkami, pričom užívateľ videl aplikáciu prvýkrát až počas testu.
Sledovaná bola schopnosť užívateľa zorientovať sa v aplikácií bez predošlého vysvetlenia, prípadne čokoľvek, čo bolo preňho mätúce. Ďalej čas a počet krokov (kliknutí), ktorý bol potrebný na základné úkony, ako napríklad vytvorenie otázky. Tiež sa pozorovalo či užívateľ postupuje očakávaným spôsobom alebo od aplikácie očakáva iný workflow.

5.4 Vyhodnocení testu a revize návrhu

Test odhalil, že hlavnou vadou v návrhu je moc veľká komplikovanosť. Vytvorenie testu bolo síce priamočiare, ale časovo náročné - užívateľ sa musel preklikať veľkým počtom obrazoviek, kým sa mu podarilo dosiahnuť to, čo chcel. Taktiež oddelenie časti aplikácie určenej pre vytváranie testov a časti na prehľad uložených testov nie je praktický. Návrh bol teda zmenený tak, aby kategórie a vytvorené testy boli viditeľné hneď po otvorení aplikácie a vytváranie testu a jeho otázok bolo na jednej stránke.

6. Návrh GUI - webová aplikace (panel) [xricht29]

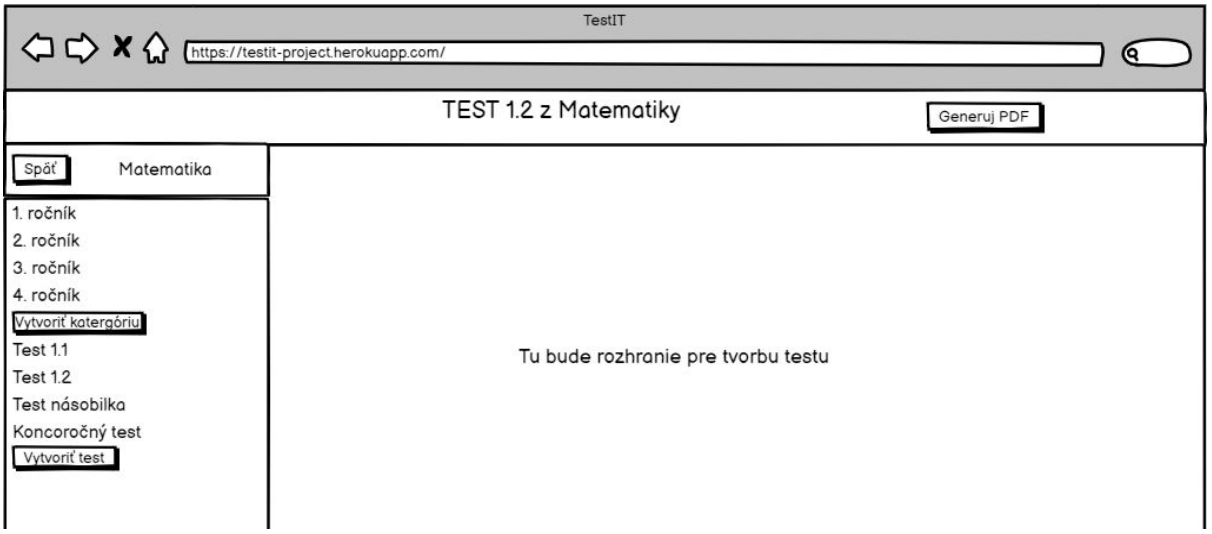
6.1 Požadavky na GUI

V aplikácii je potrebné vytvoriť akéhosi správcu testov. Realizovať by sa mohol pomocou fixného bočného panelu, kde by sa dalo jednoducho prechádzať jednotlivými kategóriami a podkategóriami, alebo aj pomocou tlačidla, ktoré by po kliknutí vykreslilo pop-up okno.

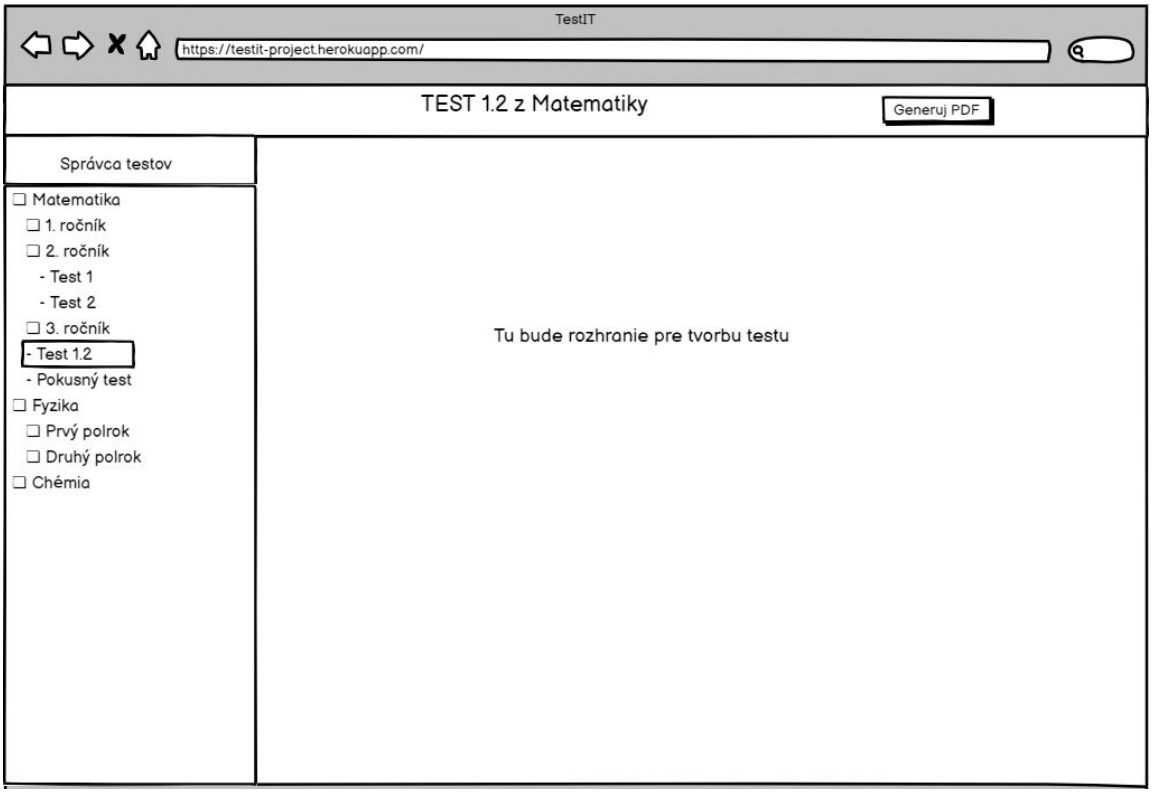
6.2 Makety

Prvotnou ideou bolo vytvoriť bočný panel, v ktorom by bol zobrazený len obsah aktuálne rozkliknutej kategórie/podkategórie. Mohli by sa vytvárať nové kategórie, alebo nové testy. Po kliknutí na daný test, by sa v hlavnej časti obrazovky zobrazil test, ktorý by sme mohli vytvárať a upravovať. Na základe toho bolo vypracovaná Varianta 1 (2. až po testoch).

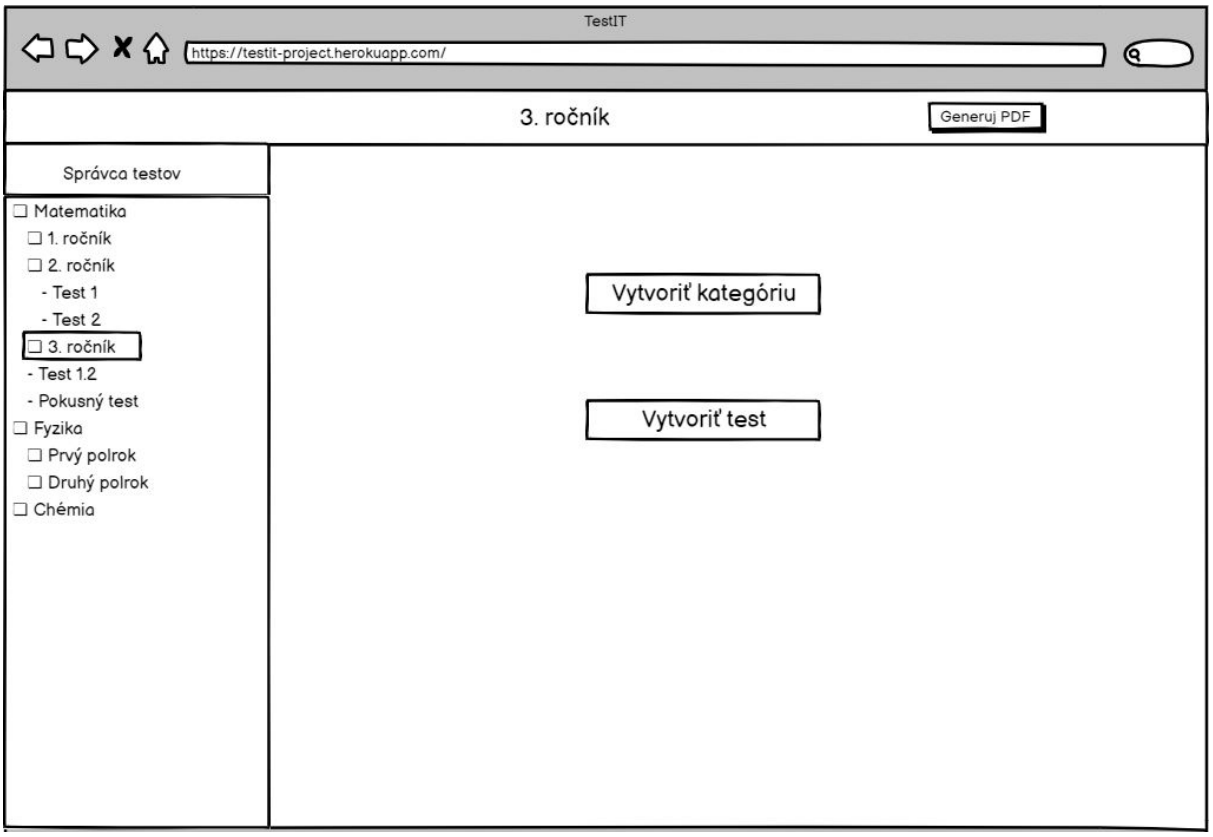
Varianta 1 (zobrazená len aktuálna kategória):



Varianta 2 (v paneli je zobrazená celá štruktúra kategórií a testov):



zobrazenie testu



vytváranie kategórie/testu po rozkliknutí kategórie

6.3 Pilotní test

Návrh a testovanie bolo realizované programom Balsamiq[2] na Variante 1 (Varianta 2 ešte neexistovala). Zúčastnila sa ho učiteľka slovenčiny cirkevného gymnázia.
Úlohou bolo vytvárať nové kategórie, podkategórie, testy (bez ich realizácie, čiže len pridávanie do panelu). Cieľom bolo zistiť, či je prostredie user-friendly, či je zjavné, ktoré tlačidlo ako funguje bez nápovedy, a vo všeobecnosti, či sa v prostredí užívateľ ľahko zorientuje.

6.4 Vyhodnocení testu a revize návrhu

Z testu vyplynulo, že užívateľ sa v prvej variante strácal, ak bol zanorený napríklad v 4. úrovni kategórii. Problém totiž nastal, ak mal pre predmety Slovenčina a Literatúra kategórie a podkategórie s rovnakými názvami a nebol si istý, kde sa vlastne nachádza. Ak by sme chceli použiť túto variantu, užívateľ navrhol, aby bola niekde v hornej časti vypísaná celá aktuálna cesta (ako napr. v programe Total Commander).

Následne bol návrh prerobený na Varianta 2, kedy je v paneli zobrazená celá štruktúra kategórií a testov. S výsledkom bol užívateľ spokojnejší. Išlo o princíp File Explorer na Windows, ktorý aj väčšina učiteľov základných a stredných škôl používa a sú naň zvyknutý. Komplikáciou by bola možno situácia, kedy by zanáranie do kategórií, resp. dlhé názvy, prinútili rozšíriť bočný panel, alebo prirobiť horizontálny scrollbar.

[2] https://balsamiq.cloud/

7. Implementace GUI - webová aplikace (tvorba testů) [xzakji02]

Hlavní část webové aplikace se skládá z vytváření sekcí a otázek. Je samozřejmostí, že se tyto dvě věci dají upravovat a mazat. Dále jsme zde vytvořili věci navíc jako je generování variant testů, možnost stáhnout test v podobě pdf nebo přidávání příloh.

7.1 Implementace

Projekt je rozdělen do několika částí(složek) pro větší přehlednost. Ve složce src se nachází všechny soubory spojené s tímto projektem vytvořené námi. Složka se dělí na komponenty(složka v src), sass složku a zbytek souborů.

Základ celé implementace bylo aplikaci napojit na API s backhandem. Toto spojení bylo navázáno pomocí http requestů. Při posílání správných requestů na správné url adresy server přidává, maže nebo upravuje různé věc v aplikacii a aplikace tak může hned reagovat a změnit data přímo před uživatelem. Tímto je také zajištěno, že obě aplikace pracují se stejnými daty. Takže pokud je něco upraveno v jedné aplikaci, tak se to ve druhé hned projeví. Takto je zajištěn Controller v MVC architektuře.

Příklad:
Pokud uživatel klikne na nějaký test v bočním menu, tak se posílá http get dotaz na server. Ten vzápětí posílá odpověď s příslušnými daty a tyto data uložíme do připravených proměnných ze, kterých se vykresluje. Test se poté objeví v pravé části aplikace, kde ho může uživatel libovolně modifikovat. Tímto je zajištěn View a připravené proměnné(struktury) jsou Model v MVC architektuře.

Tvorba sekcí:
Test je rozdělen do sekcí. V hlavičce každé sekce lze nastavit počet bodů za danou sekci a počet generovaných otázek v sekci. Podle počtu generovaných otázek se pak generují různé varianty testu. To je také hlavní důvod těchto sekcí. Tyto vygenerované varianty lze i vytisknout do PDF. Tlačítko na generování variant a PDF se nachází vpravo nahoře v menu v každém testu. Po stisknutí tohoto tlačítka se objeví popup okno, kde si uživatel může vytvořit varianty, smazat je a vytisknout je. Samotné sekce lze samozřejmě také upravovat i mazat.

Tvorba otázek:
Každá sekce se skládá z otázek. Při jejich tvorbě je nutnost nejdříve vyplnit její znění a poté stisknout tlačítko vytvořit. Tímto se otázka přidá do databáze a objeví se v dané sekci. Poté již lze otázku libovolně upravovat. Při samotném upravování se data neposílají do databáze, ale upravují se pouze lokálně. Data se odešlou do databáze až když uživatel stiskne na

tlačítko uložit. Je to vytvořeno tímto způsobem, aby byla zajištěna větší rychlost a plynulost aplikace. Toto platí i při vytváření optionu u otázky typu choice. Ty se také pouze vytváří lokálně a po uložení celé otázky se uloží do databáze společně s otázkou. Pokud uživatel stiskne na tlačítko smazat, tak je poslán delete http request, otázka se hned smaže a zmizí z View. Každá otázka může mít i příloh v podobě obrázku. Příloha se musí nejdříve přidat do otázky a poté je ještě nutné uložit otázku jako celek. Po uložení otázky se sama automaticky přepne do view modu, kde nelze otázku upravovat. Pokud by ji uživatel stejně chtěl upravit, tak musí nejdříve stisknout tlačítko edit a poté platí vše co je popsáno výše.

7.2 Použité nástroje a knihovny

Aplikace je vytvořena pomocí javascriptové knihovny React. V Reactu lze snadno vytvářet nové komponenty, což se zde náramně hodilo při přidávání nových otázek a případně zase při jejich mazání či jen úpravě. Na stylování webové aplikace je použit Sass a Tailwind. Ze začátku jsme využívali jen Sass, ale pro větší přehlednost jsme přidali ještě Tailwind. Horní menu je vygenerováno pomocí Laravelu.

7.3 Finální testování

Testování bylo rozděleno na dvě části. V první části se testovala funkčnost celého webu. Toto jsme prováděli přímo my sami. Poté co už byla celá funkčnost hotová, tak jsem aplikaci dal na testování studentce vysoké pedagogické školy, pro kterou tato aplikace byla primárně vytvořena. Ona sama si mohla celou aplikaci vyzkoušet a popsat nám případné problémy. Jediné s čím nebyla úplně spokojená bylo barevné rozpoložení aplikace, což jsme s její pomocí vyladili. Díky její pomoci by aplikace měla být přehlednější a tím jednodušší na ovládání.

7.4 Vyhodnocení testu

Díky finálnímu testování jsme dokázali odhalit poslední velké chyby v aplikaci. Vzhled a dizajn byly taktéž vyladěný se zaměřením na větší přehlednost. Díky tomuto zaměření by proto neměl být problém se v aplikaci snadno orientovat a začít ji hned, po prvním spuštění, používat. Uživatelé byli obecně s aplikací spokojeni a sdělili nám co by se jim tam v budoucnu líbilo. Aplikace je proto připravená i na další případné rozšíření.

8. Implementace GUI - desktopová aplikace [xmores01]

Keďže je aplikácia implementovaná vo frameworku WPF, bol použitý vyššie uvedený návrhový vzor MVVM.

8.1 Implementace

Implementácia projektu je usporiadaná vo viacerých priečinkov. Najdôležitejšie sú:

- Views - triedy, ktoré definujú zobrazenie aplikácie,
- ViewModels - triedy controllerov,

- Services - obsahuje triedy, ktoré slúžia na komunikáciu s Rest API servera,
- Commands - obsahuje definíciu základnej triedy commandov.

Modely sú definované v druhej časti projektu TestITData.

ViewModely vždy obsahujú inštanciu niektorého z Modelov, ktorého obsah (dáta) sú načítané zo serveru pomocou asynchronných metód niektorej triedy zo Services. Ďalej obsahuje aj metódy slúžiace na manipulovanie s dátami modelu a špeciálne metódy commands, ktoré slúžia na komunikovanie s view.
Každá View má ako dátový kontext priradenú inštanciu ViewModelu. Pomocou bindovania dát sa naviaže na atribúty ViewModelu (čo sú väčšinou dáta z Modelu), ako napríklad text, názov a tieto potom zobrazuje. Pri zmene hodnôt týchto dát, či už zo strany View pomocou interakcie užívateľa alebo zo strany ViewModelu, sú obe strany informované pomocou vyvolania PropertyChanged udalosti. Ďalší spôsob, ako sú View a ViewModel prepojené je pomocou už spomínaných commandov. Pri udalosti View, ako je napríklad kliknutie na tlačidlo sa zavolá špecifikovaný command, ktorý je implementovaný vo ViewModeli. Tieto commandy slúžia na načítanie, updatovanie alebo mazanie dát a volajú sa z nich asynchronné metódy zo Services, ktoré komunikujú zo serverom.

Grafické rozhranie sa skladá z dvoch hlavných častí, bočného menu a zobrazenia/vytvorenia testu.

8.1.1 Bočné menu

Bočné menu je implementované pomocou TreeView elementu, ktorý vzhľadom pripomína štruktúru priečinkov. Koreňová kategória je naviazaná na CategoryViewModel, podkategória na SubcategoryViewModel a test na TestViewModel. Všetky tieto triedy dedia základnú spoločnú funkcionalitu z triedy MenuItemViewModel a hlavný TreeView element má tiež vlastnú triedu ViewModelu MenuViewModel. Tieto ViewModely uchovávajú refencie na rodičovský element ako aj potomkov a umožňujú rôznu funkčnosť pre každú zo skupín elementov. Pridávanie kategórií a testov, ako aj ich mazanie je umožnené pomocou ContextMenu elementu, ktorý sa zobrazí po pravom kliknutí. Možnosti v tomto menu su nabindované na commandy v prislúchajúcom ViewModeli a spôsobia poslanie želaného dotazu na server.

8.1.2 Zobrazenie testu

Pri spustení aplikácie sa vždy vytvorí jedna inštancia TestView, ktorá zabezpečuje zobrazenie a vytváranie alebo upravovanie testu. Avšak pokým užívateľ neklikne na niektorý z testov v bočnom menu, zostáva view skrytá. Po kliknutí sa tejto view priradí ako dátový kontext TestViewModel vybraného testu a teda bindingy vo view zobrazujú dáta a volajú commandy práve pre daný test a následne sa nastaví ako viditeľná.

8.2 Použité nástroje a knihovny

Desktopová verzia aplikácie pre tvorbu testov bola implementovaná vo frameworku WPF (Windows presentation forms).
Pri komunikácii s Rest API serveru bola k práci s Json objektami použitá knižnica Newtonsoft.Json. Pri komunikácii medzi View a ViewModelom, konkrétne ak bolo treba pri nejakej udalosti zabezpečiť vyvolanie metódy alebo commandu z ViewModelu, bola použitá knižnica Microsoft Behaviors.

8.3 Finální testování

Testovanie funkčnosti prebehlo už počas implementácie po dokončení väčšieho celku (napríklad bočné menu).
Finálne testovanie prebehlo s rovnakým užívateľom, ako pilotné testy, teda učiteľkou angličtiny na strednej škole. Mala za úlohu vytvoriť reálny test a potom zhodnotiť, ako sa jej aplikácia využívala.

8.3.1 Výsledky testovania

Užívateľ ocenil možnosť vytvárania rôznych testových variant pri generovaní pdf, prehľadnosť bočného menu, ako aj to, že pridávanie a mazanie kategórií je veľmi podobné prieskumníku priečinkov, s ktorým sa už na systéme Windows stretol. Užívateľ taktiež navrhoval zlepšiť vizuálne oddelenie jednotlivých otázok v teste, čo bolo zapracované. Celkovo bol užívateľ s aplikáciou spokojný.

8.4 Vyhodnocení testu

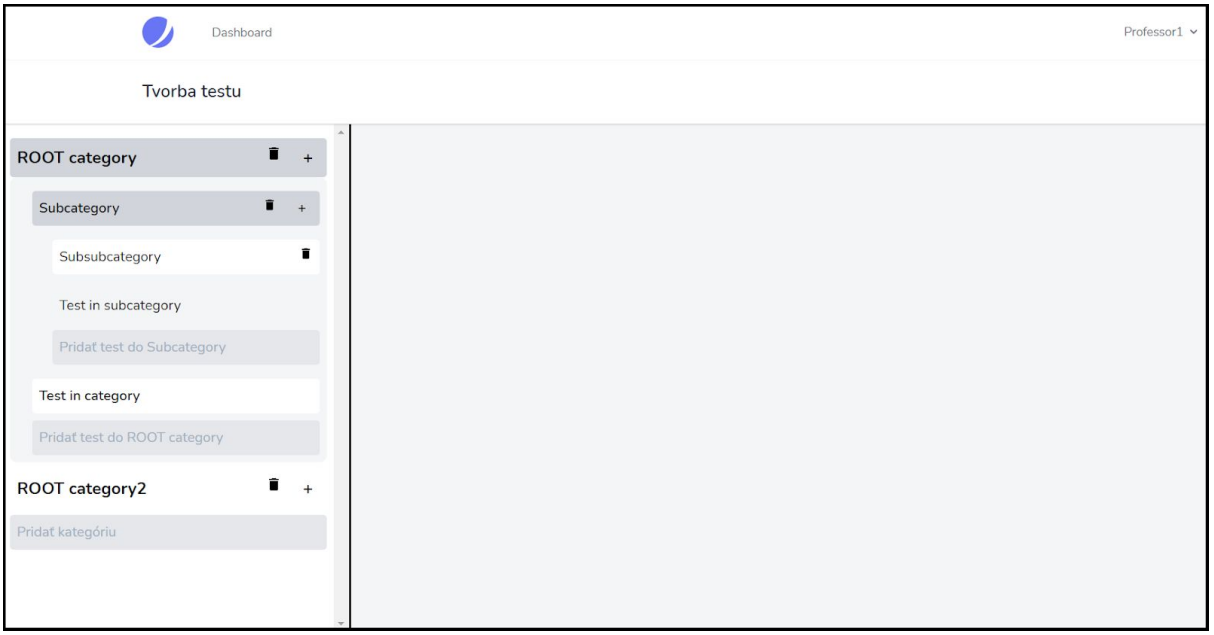
Stejně jako u vyhodnocení Pilotního testu. Výsledky diskutovat, není nutná revize návrhu, stačí popis, co a jak je nutné do budoucna udělat jinak.

9. Implementácia GUI - webová aplikace (bočný panel a responzivnost) + desktopová aplikace (Pdf pop-up) [xricht29]

V rámci webovej aplikácie bol implementovaný bočný panel, ktorý slúži pre správu testov a kategórií a v rámci desktopovej aplikácie bol implementovaný pop-up pre správu testových variant (skupín), ich generovanie, tlač a tiež tlač jednotlivých odpoveďových hárkov.

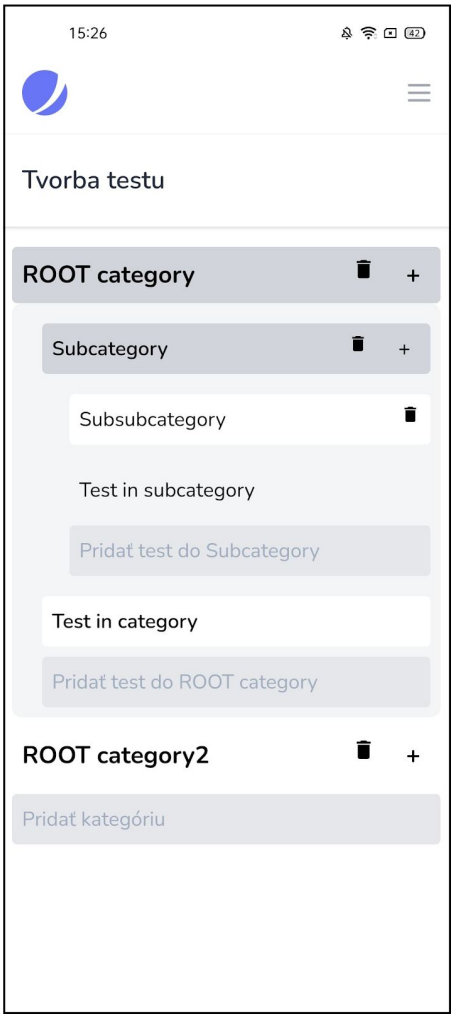
9.1 Implementace

Bočný panel - Implementovaný bol tak, ako aj tvorba testov, v Reacte. Reprezentovaný je javascriptovým komponentom TheNavigation, do ktorého sú následne načítané komponenty SubCategories a SubSubCategories s k nim príslušnými *scss* štýlami. V tejto webovej verzii je podporované maximálne 3-stupňové zanorenie kategórií a testov a to z dôvodu šírky tohto panelu. Jednotlivé API dotazy (či už pre prácu s kategóriami, alebo vytvorenie zatiaľ prázdneho testu), sú realizované pomocou funkcie *fetch()*, ktorá najprv poslúži pre získanie csrf-tokenu a ten následne vloží do hlavičky konkrétnej požiadavky. Podľa toho, o akú operáciu ide, používame metódy *GET*, *POST*, *PUT*, alebo *DELETE*. Http požiadavky sa posielajú vo formáte *JSON*.

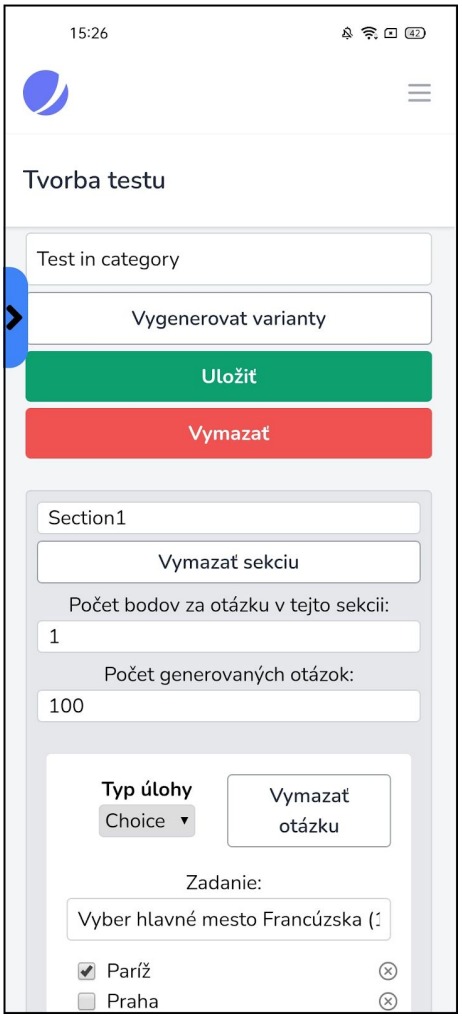


Bočný panel - PC verzia

Responzívnosť - Responzívnosť bola do istej miery riešená pomocou štýlov v knižnici Tailwind CSS, avšak, keďže v Reacte nie je implementovaná plná podpora (napríklad trieda *flex-wrap*), musel byť tento fakt vykompenzovaný ručnou úpravou štýlov, pomocou pravidla *@media (max-width: Npx) {...štýly….}*
Štýly boli menené pre jednotlivé komponenty, no najväčšia zmena nastala pre bočný panel (v index.scss), kedy je pre mobilnú verziu zasúvateľný do strany, pomocou zmeny jeho pozície a so spomaleným efektom (pravidlo *transition: all 0.5s;*)



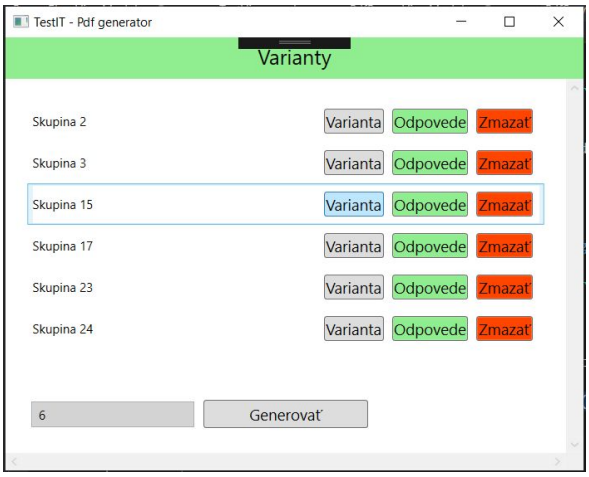
Vysunutý bočný panel - mobil



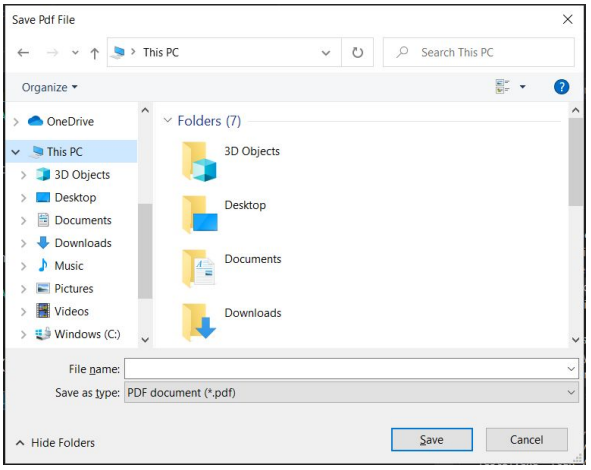
Zasunutý bočný panel - mobil

Pdf popup - Pre desktopovú aplikáciu implementovanú v jazyku C# bolo vytvorené nové modálne okno (framework WPF). Reprezentované je 2 view-ami PdfPopupView.xaml a PopupElemView.xaml (jednotlivé varianty testu v zozname) , spolu s ich view-modelami PdfPopupViewModel.cs a PopupElemViewModel.cs. Implementovaná je možnosť generovať nové varianty v určitom počte a následná práca s týmito variantami. Konkrétne možnosť vymazaania danej varianty, vygenerovania jej testovej PDF-verzie a PDF dokumentu so správnymi možnosťami, slúžiaciemu pre opravu testu. Pri týchto operáciach sa využívajú funkcie zo súboru PdfService.cs, ktorý zabezpečuje komunikáciu prostredníctvom REST-API (napojenie na dátový model) a generovanie PDF (pre testovanie / pre opravu). Platí, že

elementy z view sú namapované na atribúty príslušných view-model tried, prostredníctvom väzby *Binding*.



Pop-up pre prácu s variantami



Dialógové okno z *System.Windows.Forms*

9.2 Použité nástroje a knihovny

Celá tvorba testu, a teda aj bočný panel boli implementované v JavaScriptovej knižnici, respektíve webovom frameworku *React*. Pre lepšiu prácu so štýlami webstrány, bol použitý *Tailwind CSS*. Horná lišta, ako aj spôsob prihlasovania a uvítacia nástenka sú realizované v *Laraveli (8.9.0)*, pomocou komponent z knižnice *Livewire*.

Čo sa týka pop-upu, tak ten, ako aj celá C# aplikácia, boli vytvorené pomocou frameworku *WPF* a na generovanie PDF bola použitá knižnica *itext7 (v.7.1.13)*. Následný správca súborov, ktorý slúži na zvolenie miesta uloženia výsledného PDF, bol realizovaný pomocou triedy *SaveFileDialog* z knižnice *System.Windows.Forms*.

9.3 Finální testování

Testovanie panelu prebiehalo plnením zadaných úloh (tvorba testov, kategórií, podkategórií, ich úprava a mazanie). Testovanie responzívnosti bolo realizované prácou s prostredím pre tvorbu testu na školskom tablete a tiež testovaním na mobile. Testu sa účastnili 2 učitelia gymnázia, ktorí použili tablety zo svojho gymnázia. Pri pop-upe v C# aplikácii bolo úlohou vygenerovať varianty, zmazať ich a stiahnuť si PDF súbory s testovou skupinou a odpoveďovým hárkom.

9.4 Vyhodnocení testu

Testovaným účastníkom sa bočný panel páčil po dizajnovej stránke, najmä prevedenie na mobilnej verzii. Práca s ním im nerobila žiaden problém. Niektorým chýbala možnosť viacstupňového zanárania, čiže aby to nebolo obmedzené. Tento fakt by sa mohol vyriešiť tak, že by panel obsahoval scrollbar aj do strany, avšak efektívnejšie by asi bolo názvy orezať podľa pravého okraja, ako na mobilnej verzii fakultného mailu (štýly *text-overflow: ellipsis;*). Stále sa však naskytuje otázka, ako inak, efektívnejšie, odlíšiť jednotlivé úrovne, ako odtabovaním doprava.

Čo sa týka C# popupu, tak do budúcna by napríklad mohlo dialógové okno pre ukladanie navrhnúť aj názov PDF súboru, prípadne otvoriť naposledy otvorené umiestnenie. Takisto by vygenerované pdf mohlo obsahovať obrázkové prílohy, keďže API pre prácu s nimi je plne funkčné a odskúšané na lokálnej verzii. Na server, by však bolo potrebné pripojiť nejaké úložisko na dáta (väčšinou platené, alebo trial Amazon S3).

Pri knižnici na generovanie PDF dokumentov sa ukázalo, že sa s ňou veľmi jednoducho a efektívne pracuje… nevýhodou však bolo, že občas nezobrazovala písmená s mäkčeňmi. Boli pokusy o odstránenie tohto problému, kedy by sa nastavilo UTF-8 kódovanie, avšak bez úspechu. Možno by pomohol iný font písma, alebo stojí za zváženie náhrada inou knižnicou, ktorá by napríklad podporovala aj rôzne matematické funkcie a vzorce.

Od učiteľov tiež zaznela požiadavka, či by nebolo možné testy hromadne nahrať do systému. Väčšina z nich si totiž počas rokov vyučovania vytvorila veľké množstvo testov a tie by radi nejako nahrali do systému, aby v tom mali poriadok a mali možnosť generovať rôzne varianty. Možno by preto bolo vhodné v budúcnosti vytvoriť akýsi nástroj, ktorý by vedel prejsť nahraté testy, v určenom formáte, rozlíšiť otázky od odpovedí / možností a tie pridať do databázy systému.

10. Závěr

V tomto projekte sa nám podarilo navrhnúť a implementovať funkčný systém pre tvorbu testov, pozostávajúci z 2 aplikácií (c# + web), pracujúcich prostredníctvom REST-API s Laravel backendom, napojeným na databázu. Tento systém sa nám (okrem toho, že dokáže vytvárať testy, generovať PDF varianty a odpoveďové hárky) podarilo rozšíriť aj o plnohodnotné online testovanie, v rámci predmetu IIS.

Medzi najväčšie nedostatky, ktoré boli zistené pri testovaní, a ktoré by sa dali v budúcnosti do istej miery odstrániť, patrí:

- viac úrovní zanorenia v hierarchii kategórií a testov, pre webovú aplikáciu
- úložisko na serveri, pre sfunkčnenie otázkových príloh v teste
- zmeniť font, prípadne knižnicu pre generovanie PDF, kvôli kódovaniu (mäkčene)
- doimplementovať výber už existujúcich otázok do aplikácií (na úrovni servera funkčné, avšak potrebné obmedziť pre jednotlivé kategórie, napr. v prípade, že užívateľ vyučuje viaceré predmety)

Užívatelia ocenili najmä moderný dizajn, prehľadnosť aplikácie (oproti na Slovensku využívanému Edupage), intuitívne ovládanie, responzívnosť pri mobilnej verzii, či nezahltenosť informáciami.

Pri vypracovaní tohto projektu sme získali významné skúsenosti s prácou v tíme a naučili sme sa, resp. zdokonalili v programovaní v jazykoch PHP, Javascript, C#, s využitím rôznych knižníc a frameworkoch, ako Laravel a LiveWire, Tailwind CSS, React, a tiež WPF a iText.
Hlavným prínosom práce v tíme bola možnosť konzultovať riešenie s ostatnými členmi, a tak dosiahnuť lepšieho konečného výsledku, než individuálne a taktiež možnosť spoločne riešiť problémy, niektoré problémy a chyby, ktoré sa pri implementácii vyskytli. Jediná vec, ktorá

skomplikovala našu prácu, ktorú by sme v prípade priaznivejšej pandemickej situácie spravili inak, je organizovanie osobných stretnutí namiesto online hovorov.

Reference

Terry G. Lee, Začíname s WPF, 2018 [online]. [cit. 20.11.2020].
Dostupné z:
https://docs.microsoft.com/cs-cz/visualstudio/designers/getting-started-with-wpf?view=vs-2019
Václav Dajbych, MVVM: model-view-viewmodel, 2009 [online]. [cit. 20.11.2020].
Dostupné z:
https://www.dotnetportal.cz/clanek/4994/MVVM-Model-View-ViewModel

Použitá dokumentácia:
https://laravel.com/docs/8.x/
https://laravel-livewire.com/docs/2.x
https://tailwindcss.com/docs
https://reactjs.org/

Nástroj pre vytváranie MockUpov: https://balsamiq.cloud/