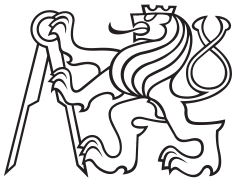
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačů

Interaktivní úkoly na mobilním telefonu

Ondřej Viskup

Vedoucí: Ing. David Sedláček, Ph.D.
Obor: Software
Studijní program: Otevřená informatika
Květen 2022

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Viskup** Jméno: **Ondřej** Osobní číslo: **483724**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Software**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Interaktivní úkoly na mobilním telefonu

Název bakalářské práce anglicky:

Interactive tasks on a mobile phone

Pokyny pro vypracování:

1) Seznamte se s aktuálním stavem projektu EduARd (rozšířená realita pro školy).
2) Ve spolupráci s vedoucím práce navrhněte minimálně pět nových interaktivních úloh pro rozšíření projektu (např. pexeso, obrázkové sudoku, skládání rozstříhaného obrázku, překreslování obrázku, seřazení obrázkových kostek - tzv. Loydova patnáctka).
3) Navrhněte a implementujte rozšíření stávajících aplikací o navržené úkoly. Stávajícími aplikacemi jsou webový editor úloh a dvě mobilní aplikace v jazycích React Native a Kotlin.
4) Při návrhu a realizaci postupujte podle metodiky User Center Design (UCD).
5) Vzájemně porovnejte implementované úkoly na obou variantách mobilních aplikací. Ověřte stabilitu aplikace na systému Android i iOS.

Seznam doporučené literatury:

1. T. Lowdermilk, User-Centered Design, O'Reilly Media, 2013
2. B. EisenmanLearning React Native: Building Native Mobile Apps with JavaScript, O'Reilly Media, 2016
3. B. Fling, Mobile Design and Development, O'Reilly Media, 2009
4. Šimon Maňour, Virtuální naučné stezky v React Native, BP FEL, 2019
5. Anastasia Surikova, Webový klient pro správu mobilních výukových úloh systému EduARd, BP FEL, 2019
6. EduARd: David Sedláček, [online], https://eduard.fel.cvut.cz/

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D. katedra počítačové grafiky a interakce FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **02.02.2022** Termín odevzdání bakalářské práce: **20.05.2022**

Platnost zadání bakalářské práce: **30.09.2023**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____
Datum převzetí zadání

_____
Podpis studenta

Poděkování

Děkuji svému vedoucímu za cenné rady a postřehy při vypracovávání zadání bakalářské práce. Dále bych chtěl poděkovat rodině, přátelům a své partnerce za podporu při studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškerou použitou literaturu.

V Praze, 20. května 2022

Abstrakt

Tato práce rozšiřuje již existující projekt EduARd o možnost přidávat interaktivní úlohy do virtuálních naučných stezek. Cílem práce je analyzovat existující aplikace systému EduARd a navrhnout možné rozšíření aplikací o interaktivní úlohy.

Rozšiřovanými aplikacemi jsou webová aplikace Editoru, mobilní aplikace v React Native a mobilní aplikace ve Flutteru. Na základě návrhu rozšíření proběhne implementace a zhodnocení celé práce včetně návrhu možných dalších rozšíření a vylepšení.

Klíčová slova: webová aplikace, design zaměřený na uživatele, React, systém pro správu obsahu, React Native, multiplatformní aplikace, virtuální naučné stezky, Expo, Flutter, mobilní aplikace, systém EduARd, interaktivní úloha

Vedoucí: Ing. David Sedláček, Ph.D. Katedra počítačové grafiky a interakce, Karlovo náměstí 13 (místnost: E-425), Praha 2

Abstract

This thesis extends the EduARd system by the possibility of creating interactive tasks for the virtual educational trails. This project aims to analyse the current state of existing applications of the EduARd system and design an extension that adds interactive tasks to them.

Applications to be extended are the web application Editor, the React Native mobile application, and the Flutter mobile application. The extension will then be implemented in all of them based on the created design. Eventually, the whole project will be summarized, and possible extensions will be presented.

Keywords: web application, user-centered design, React, content management system, React Native, multiplatform application, virtual educational trails, Expo, Flutter, mobile application, system EduARd, interactive task

Title translation: Interactive tasks on a mobile phone

Obsah

Obrázky

Kapitola 1

Úvod

S tím jak se v průběhu pandemie korona viru výuka přesunula ze školních lavic a poslucháren do on-line prostředí, hledají vzdělávací instituce způsoby, jak svým žákům i nadále poskytovat kvalitní a plnohodnotnou výuku. Nenadálost celé situace okolo pandemie jenom dokazuje, že se školy musejí poohlížet po alternativních metodách vzdělávání. Žáci se na hodinách často nudí nebo je zaměstnávají jiné zajímavější aktivity. S tím jak udělat výuku zajímavější a atraktivnější pro všechny zúčastněné snad bojují od nepaměti všichni pedagogové.

Zatraktivnit a dostat výuku ven je i jedním z cílů projektu *EduARd*[13], který je vyvíjen v rámci katedry počítačové grafiky a interakce Fakulty elektrotechnické ČVUT v Praze.

1.1 Motivace

V rámci projektu *EduARd* již existuje systém učebnic, který umožňuje uživatelům tvořit naučné obrazovky včetně obrázků, audio-vizuálního obsahu, naučných textů a otázek. Jako další typ obsahu se nabízí přidání interaktivních úloh, které by uživatelům procházejícím virtuální naučnou stezkou nabídly prostor k odreagování se a alternativní způsob poznávání dané problematiky. Správcům učebnic by se tak nabídl nový způsob, jak zprostředkovat nabývání vědomostí po vzoru *školy hrou*.

1.2 Cíle práce

Cílem této práce je navrhnout a implementovat rozšíření projektu *EduARd* o interaktivní úlohy, které budou moci koncoví uživatelé plnit na přenosných

zařízeních jako je například mobilní telefon či tablet. K dosažení tohoto cíle bude nutné nejdříve analyzovat aktuální stav aplikací, kterých se rozšíření bude týkat. Těmi jsou jmenovitě *Webový klient pro správu mobilních výukových úloh systému* EduARd vyvinutý Anastasií Surikovou[15] (v textu označovaný dále jen jako **Editor**), projekt *Virtuální naučné stezky v React Native* vyvinutý Šimonem Maňourem[17] (v textu označovaný dále jen jako **RN klient**) a *Mobilní Flutter aplikace pro řešení výukových úloh v systému* EduARd vyvinutá Lukášem Šimonem[16] (v textu označovaná dále jen jako **Flutter klient**). Následně bude nutné navrhnout a implementovat jednotlivé interaktivní úlohy s ohledem na uživatelskou přívětivost a zážitek. V neposlední řadě bude nutné připravit testovací scénáře a s pomocí uživatelů otestovat nově implementovanou funkcionalitu ve všech třech aplikacích.

1.3 Struktura práce

Tento text bude členěn do několika kapitol a podkapitol, přičemž nejdříve popíšeme aktuální stav jednotlivých aplikací. Dále představíme jednotlivé implementované úlohy a rozebereme jejich návrh. Poté popíšeme implementaci jednotlivých úloh a testování všech daných funkčních a kvalitativních požadavků. Nakonec zhodnotíme celou práci a popíšeme možná rozšíření této práce.

Kapitola 2

Aktuální stav projektu EduARd

V této části práce stručně popíšeme v jakém stavu se nachází tři rozšiřované aplikace.

2.1 Struktura dat

Každá virtuální naučná stezka je definovaná souborem ve formátu *XML*[1]. [17]

Soubor XML přibližně odpovídá struktuře popsané takto (společně i s násobnostmi)[2]:

```
<questionset> ... učebnice / virtuální naučná stezka
 <task> ... úkol (0 až N)
  <location> ... geolokační data úkolu
   <latitude/>
   <longitude/>
  </location>
  <questionslide> ... obrazovka úkolu (0 až N)
   <questions>
    <question /> ... otázka (0 až N)
    <description /> ... text (0 až N)
    <images /> ... obrázek (0 až N)
    <audio /> ... zvuková nahrávka (0 až N)
    <video /> ... video (0 až N)
```

[1] Extensible Markup Language je značkovací jazyk, který vývojářům umožňuje definovat libovolné tagy - značky, pomocí kterých lze budovat stromovou strukturu. Je oblíbeným formátem pro prohledávání, sdílení a zálohování dat. `https://developer.mozilla.org/en-US/docs/Web/XML/XML_introduction` [cit. 2022-05-12]

[2] `https://gitlab.fel.cvut.cz/eduard/web_frontend` [cit. 2022-05-12]

```
    </questions>
   </questionslide>
  </task>
 </questionset>
```

Pro tuto práci bude stěžejní pochopení struktury tvořitelného obsahu tj. značek *question, description, images, audio, video*. Struktura otázky (*question*) se jeví jako nejkomplexnější a nejvíce podobná tomu, co si představíme pod tím, jak by měla vypadat struktura interaktivní úlohy. Otázek je několik typů s tím, že každý typ má společné vlastnosti a pak některé vlastnosti, které jsou charakteristické pouze pro daný typ otázky.

Společnými vlastnostmi jsou například: *typ otázky, text otázky, správná odpověď, nutnost zamíchat možné odpovědi, nutnost vyhodnotit uživatelskou odpověď.*

Dále jednotlivé typy otázek přidávají další vlastnosti, jako například: *nutnost rozlišovat velká a malá písmena u uživatelské odpovědi, seznam opcí, rozsah intervalu, validní interval, krok intervalu.*

Oba seznamy vlastností otázek byly přejaty z původní práce autorky *Editoru* Anastasie Surikové[3] [13].

Struktura interaktivní úlohy v XML bude inspirovaná strukturou otázky popsané výše a bude popsaná detailněji v Návrhové části v sekci 3.6.1.

K tomu, abychom mohli s daty ze souboru virtuální naučné stezky pracovat v *Editoru*, nejdříve musíme jeho obsah *parsovat*[4]. K tomuto účelu autorka Editoru vytvořila služby na zpracování a převod z XML do JavaScriptového objektu.[5] Podobné služby se vyskytují taktéž v RN klientovi[6] a Flutter klientovi.[7]

2.2 Editor

Editor systému *EduARd* je napsán v jazyce JavaScript ve standardu ES6[8] s použitím knihovny **React**[4], která umožňuje jednoduchou tvorbu interaktivních uživatelských rozhraní. React pro stavbu webové stránky používá

[3] Surikova, str. 9

[4] Parsování nebo syntaktická analýza v programování slouží ke konverzi formátovaného textu do nějaké datové struktury. V případě JavaScriptu tyto struktury představují objekty. [14]

[5] Surikova, str. 29

[6] Maňour, str. 36

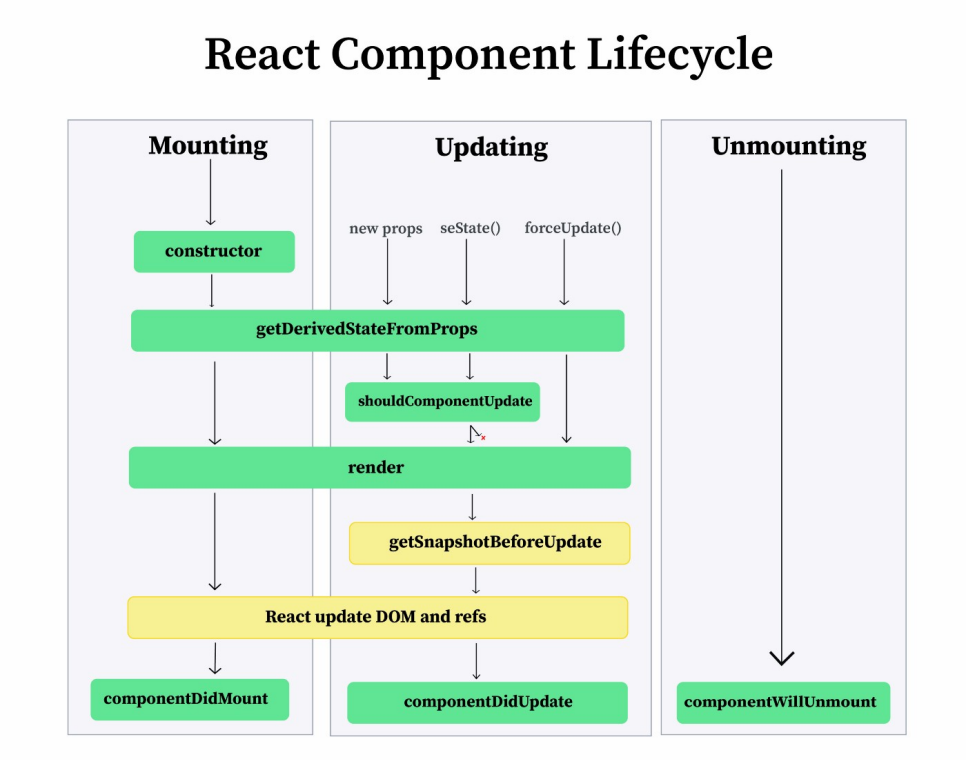
[7] Šimon, str. 27

[8] Surikova, str. 25

komponenty, které generují *DOM*[9] na základě aktuálního stavu komponenty. Tyto komponenty také reagují na veškeré změny dat a v příslušném *DOM*u upraví odpovídající větve struktury. V krátkosti popíšeme životní cyklus komponenty *Reactu*.

2.2.1 Životní cyklus (Life-cycle) komponenty ve frameworku React / React Native

Důležitým aspektem knihovny *React* jsou komponenty a jejich životní cyklus. *React* nabízí několik tzv. *lifecycle metod*, které vývojář může *overridovat*[10].



Obrázek 2.1: Lifecycle metody frameworků React / React Native, převzatý z [10]

Jak ukazuje obrázek 2.1, komponenta *Reactu* během svého *lifecyclu* probíhá celkem třemi fázemi:

[9]Document Object Model - reprezentuje webovou stránku jako stromovou strukturu a umožňuje manipulaci s její strukturou, obsahem a styly pomocí skriptů a programovacích jazyků `https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction` [cit. 2022-05-11]

[10]Overriding je princip v objektově orientovaném programování umožňující vývojáři přepsat implementaci metody na podtřídě, která byla definovaná v rodičovské třídě. Overriding se používá v případě, že chceme definovat odlišné chování pro některé třídy potomků dědících z rodičovských tříd. `https://it-slovnik.cz/pojem/overriding` [cit. 2022-05-18]

1. **Mounting** - inicializace komponenty, její první vykreslení a aktualizace *DOMu*
2. **Updating** - aktualizace komponenty (např. při změně dat), způsobí překreslení komponenty a opětovné generování *DOMu*
3. **Unmounting** - úklidová fáze, odebírání komponenty z *DOMu*

Nejčastěji používanými metodami jsou: [10]

- `render()` je metoda, která se stará o vykreslování komponenty. Je to jediná metoda, kterou musíme implementovat, pokud dědíme třídu *React.Component*
- `componentDidMount()` je metoda, která se zavolá po tom, až jsou všechny prvky komponenty vykresleny. V rámci této metody můžeme definovat volání na API[11], nastavování event listenerů, atp.
- `componentDidUpdate()` je metoda, která se zavolá, pokud se změnila data (buď *props*, která komponenta získá na vstupu při inicializaci nebo její vnitřní stav *state*) v rámci komponenty a komponenta se v reakci na tyto změny překreslila
- `componentWillUnmount()` je metoda, která se zavolá těsně před destrukcí dané komponenty. V rámci této metody bychom měli odhlásit všechny event listenery a zrušit všechna neuskutečněná volání na API.

2.2.2 Stylování komponent pomocí React Bootstrap a preprocessoru SASS

Pro správu stylů je použita knihovna **React Bootstrap**, která poskytuje nejen styly pro uživatelské komponenty, ale přidává už hotové komponenty *layoutu*[12], navigační lišty, formulářů, tabulek, *pop-upů*[13], tlačítek a mnoho dalších připravených k okamžitému použití[3]. Díky tomu může vývojář v relativně krátkém čase vytvořit webovou aplikaci s hezkým designem bez nutnosti definovat vlastní styly pomocí *CSS*[14].

Pokud bychom přesto chtěli použít vlastní styly, *Editor* používá preprocessor **SASS**. Jedná se o rozšíření *CSS*, které přidává proměnné, vnořená

[11] API - Application Programming Interface je rozhraní aplikace poskytující data a funkcionalitu jiným aplikacím `https://www.ibm.com/cloud/learn/api` [cit. 2022-05-12]

[12] rozvržení stránky - např. mřížková struktura

[13] vyskakovací okna

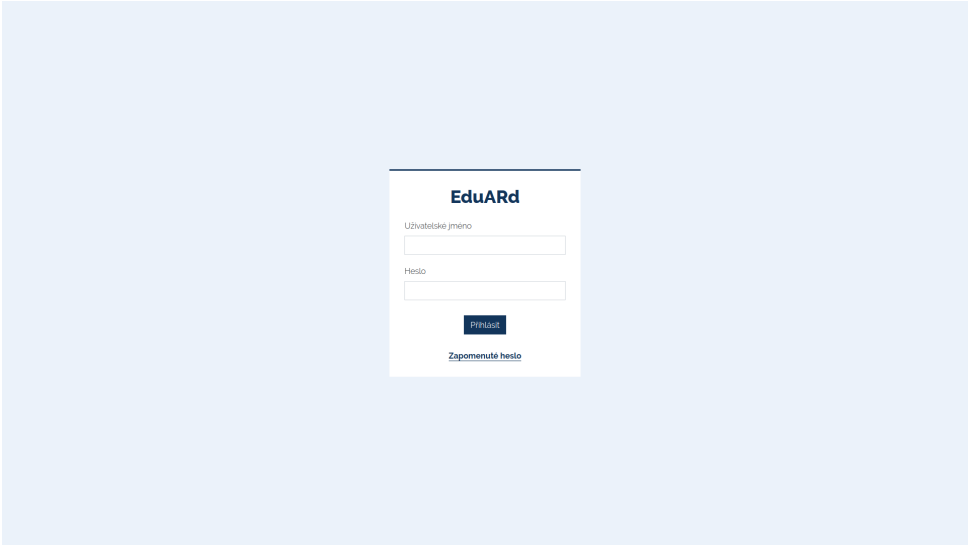
[14] Cascading Style Sheets - umožňují vývojáři definovat pravidla stylování, které se při sestavení webové aplikace aplikují na jednotlivé prvky šablony HTML `https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/What_is_CSS` [cit. 2022-05-12]

pravidla, *mixiny*[15] a funkce.[6] Klade důraz na přepoužitelnost, což umožňuje přehlednější a efektivnější zápis pravidel *CSS*.

Správa balíčků knihoven a komponent je řešena přes **Yarn**[16]. Další implementační detaily jsou k dispozici v původní práci autorky.

Vzhledem k tomu, že samotný projekt neobsahoval informaci o tom, na jaké verzi *Node.js* byl vytvořen, bylo nejdříve nutné zjistit, na jaké verzi se podaří spustit. Nakonec se ho podařilo rozběhnout ve verzi 13.14.0 a na této verzi probíhal i veškerý pozdější vývoj. Pro další vývoj a údržbu by bylo dobré projekt aktualizovat na vyšší verzi *Node.js*, nicméně rozsah této práce už tuto aktualizaci neumožnil.

2.2.3 Vizuální stránka aplikace



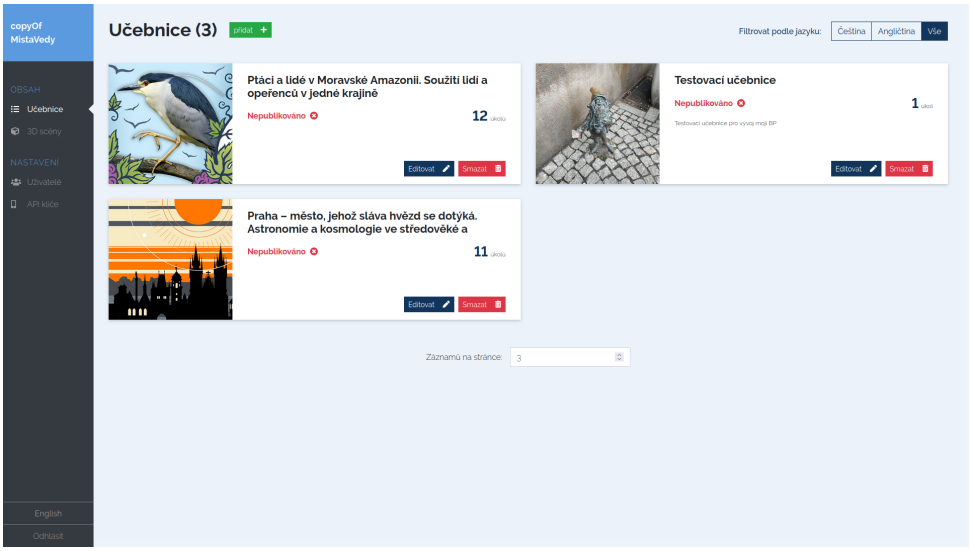
Obrázek 2.2: Přihlašovací stránka Editoru

Po startu aplikace se nám otevře přihlašovací formulář se dvěma políčky na zadání uživatelského jména a hesla. Dále obsahuje tlačítka *Přihlásit* a *Zapomenuté heslo* (viz obrázek 2.2). Po přihlášení se uživatel dostane na hlavní stránku *Editoru* (viz obrázek 2.3). *Editor* mimo jiné umožňuje editaci učebnice. Vzhledem k náplni zadání práce nás bude zajímat především přidávání a editace jejího obsahu. Pro přidávání obsahu na obrazovce úkolu slouží zelený FAB[17] s ikonou plusu (viz obrázek 2.4). Po jeho stisknutí se na obrazovce vytvoří položka obsahu, podle toho, jaký obsah jsme vybrali. Dále se detailněji

[15] *Mixin* v rámci SASSu je pravidlo, které vývojáři umožňuje definovat styly, které lze přepoužít v jiném CSS pravidle `https://sass-lang.com/documentation/at-rules/mixin` [cit. 2022-05-12]

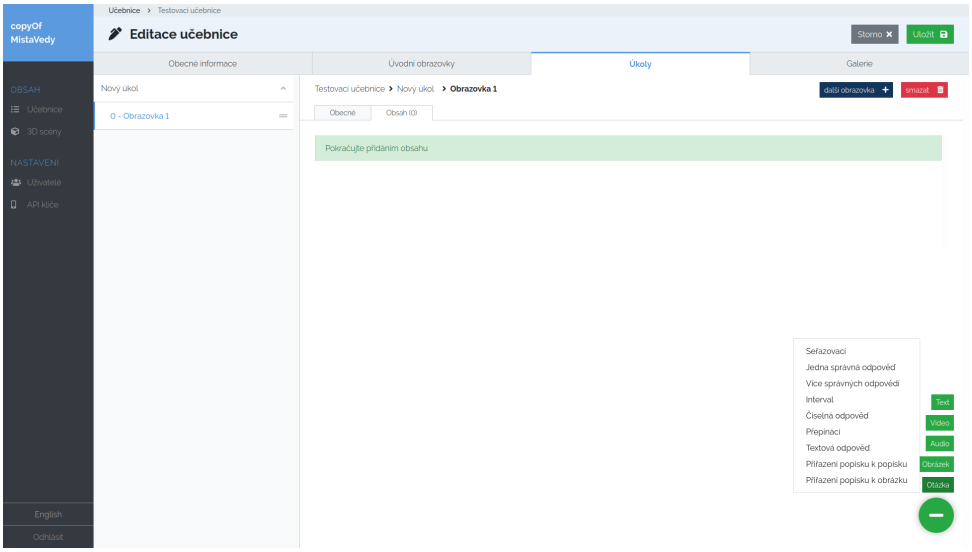
[16] Yarn umožňuje správu knihoven a komponent, jejich jednoduché přidávání, aktualizaci a odebírání `https://yarnpkg.com/getting-started` [cit. 2022-05-12]

[17] Floating Action Button je tlačítko, které po stisknutí vyvolá primární akci pro da-



Obrázek 2.3: Hlavní stránka Editoru

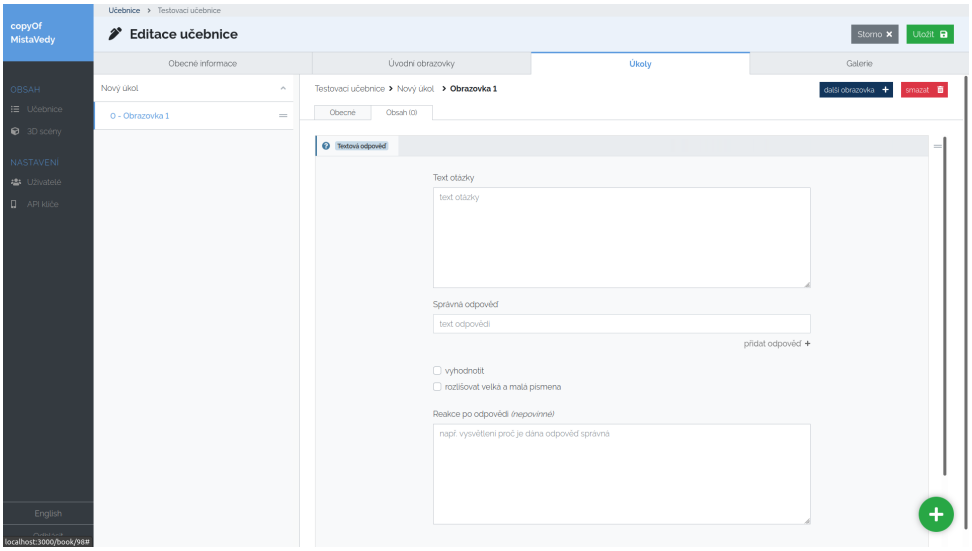
zaměříme na vytvoření nové otázky. Ta se totiž svojí strukturou nejvíce blíží struktuře interaktivní úlohy, kterou máme navrhnout a implementovat.



Obrázek 2.4: Přidání nového obsahu do učebnice v Editoru

Na obrázku 2.5 můžeme vidět formulář pro tvorbu nové otázky. Formulář je složen z pomyslného záhlaví, které je shodné pro všechny typy otázek, obsahující pole pro vyplnění textu otázky a pole pro správnou odpověď na otázku. Tělo formuláře se liší podle typu otázky. Dále formulář obsahuje zaškrtávací políčko pro vyhodnocení správné odpovědi otázky a nepovinné textové pole pro zdůvodnění správné odpovědi.

nou stránku, v tomto případě přidání nového obsahu `https://material.io/components/buttons-floating-action-button` [cit. 2022-05-13]



Obrázek 2.5: Přidání nové otázky v Editoru

Aplikace je bilingvální a podporuje jak český, tak i anglický jazyk.

O dalších funkcích Editoru se zmiňuje autorka původní práce a my je nebudeme dále rozebírat.

2.3 RN klient

RN klient je napsaný v jazyce JavaScript s použitím frameworků React Native a Expo. Podobně jako výše zmíněný *Editor* je aplikace bilingvální a podporuje jak český, tak i anglický jazyk.

2.3.1 React Native a Expo

React Native je framework pro tvorbu uživatelských rozhraní pro mobilní aplikace pro systémy *Android* a *iOS* založený na již zmíněném Reactu. Výhodou použití React Native pro vývoj mobilních aplikací je to, že pro oba systémy vyvíjíme pouze **jeden** zdrojový kód v JavaScriptu. Podobně jako v Reactu sestavujeme uživatelské rozhraní aplikace z komponent. React Native pak podle systému, na kterém aplikace běží tyto komponenty transformuje na nativní komponenty implementované v *Javě* (pro systém *Android*) a v *Objective-C* (pro systém *iOS*). Při spuštění aplikace na mobilním telefonu se potom k vykreslování uživatelského rozhraní použijí nativní komponenty místo *webview*. [9]

React Native umožňuje dva způsoby vývoje aplikace. Pomocí Expo CLI[18] z frameworku Expo nebo pomocí React Native CLI. [5] Autor původní aplikace si pro vývoj vybral framework Expo a to z důvodu lepší dokumentace, podpory velkého množství knihoven a zejména možnosti testovat aplikaci i v systému *iOS*.[19]

Expo je framework pro vývoj aplikací v React Native, který na rozdíl od React Native CLI umožňuje vývoj bez nutnosti znalosti *Xcode* a *Android Studia*. Součástí Expa je i SDK[20], které obsahuje nespočet nástrojů a knihoven pro tvorbu mobilních aplikací. Dalším prvkem frameworku je i aplikace *Expo Go*, kterou je možné nainstalovat na mobilním zařízení a pomocí které lze aplikaci testovat přímo při vývoji. [1]

Aplikace běží na *Expo SDK* ve verzi 41 a *Node.js* ve verzi 14.19.0. Zde se podobně jako v případě Editoru nabízí aktualizace na vyšší verze obou prostředí, nicméně vzhledem k rozsahu práce a časovým nárokům zadání nebylo tuto aktualizaci možné provést.

2.3.2 Krátký popis funkcionality aplikace

Pro spuštění a správné fungování aplikace na mobilním zařízení uživatele musí být správně nadefinovaný API klíč.[21] Ten je možné vygenerovat pomocí *Editoru* v záložce *API klíče*. Vygenerovaný klíč musí být součástí všech volání na back-endový server *EduARdu*, aby si mohla klientská aplikace stahovat data o virtuálních naučných stezkách.[22]

Po spuštění aplikace se otevře úvodní obrazovka se třemi tlačítky *Prohlížet stezky, Správce stezky* a *O aplikaci* (viz obrázek 2.6a). Po kliknutí na tlačítko *Správce stezek* se otevře obrazovka *Správce stezek* se dvěmi záložkami *Dostupné stezky* a *Stažené stezky*, na které je možné stahovat a mazat virtuální naučné stezky, které byly vytvořené pomocí *Editoru* (viz obrázek 2.6). Pro stažení stezky klikneme na její kartu v záložce *Dostupné stezky* a následně klikneme na tlačítko *Stáhnout data*.

Po stažení stezky je možné ji prohlížet po návratu na úvodní scénu a kliknutí na tlačítko *Prohlížet stezky*. To otevře obrazovku *Prohlížet stezky*, na které kliknutím na kartu stezky otevřeme danou stezku.

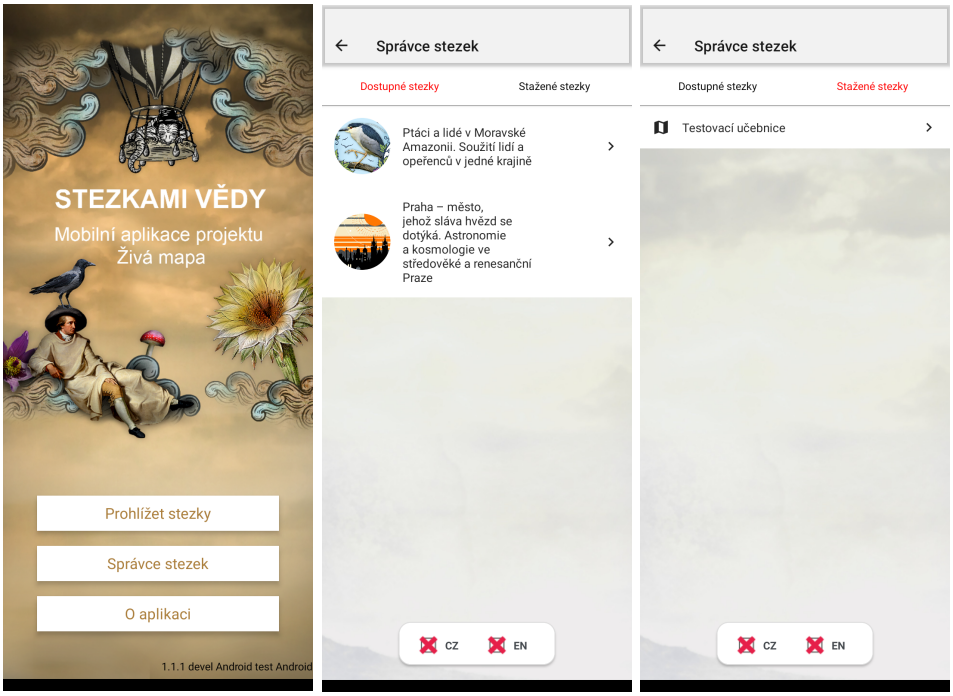
[18] Command Line Interface - program v příkazové řádce, který interpretuje příkazy zadané uživatelem a vykonává systémové funkce `https://www.w3schools.com/whatis/whatis_cli.asp` [cit. 2022-05-12]

[19] Maňour, str. 30-31

[20] Software Development Kit je balíček nástrojů pro vývoj a údržbu softwarových aplikací `https://techterms.com/definition/sdk` [cit. 2022-05-12]

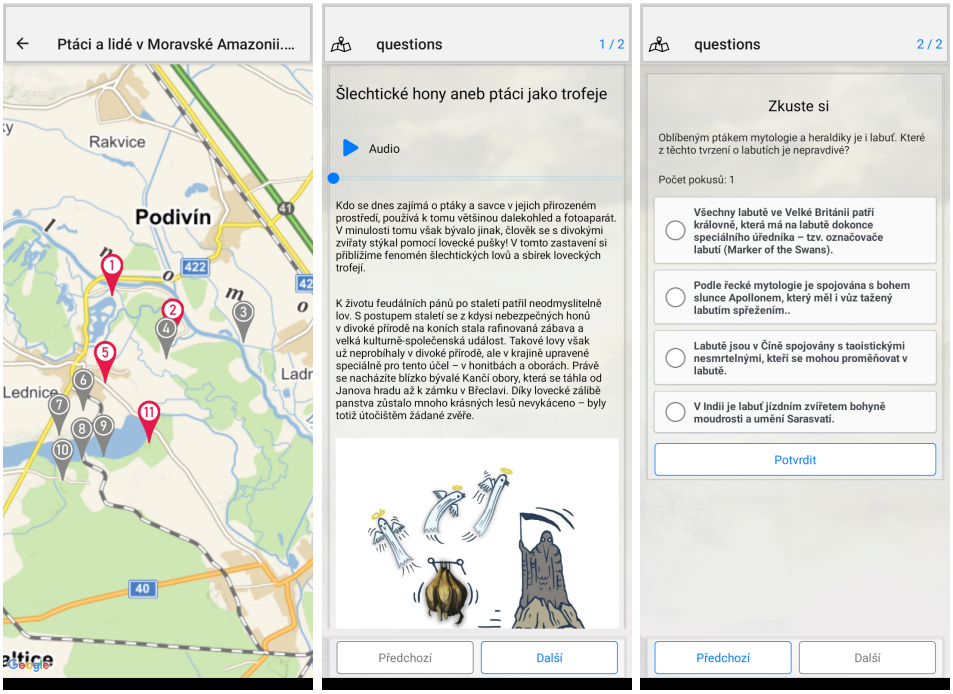
[21] API klíč slouží k identifikaci klientské aplikace při přístupu na veřejná API, v případě neplatnosti klíče může server poskytující API přístup aplikaci zablokovat [11]

[22] Maňour, str. 11



(a) : Úvodní obrazovka **(b) :** Dostupné stezky **(c) :** Stažené stezky

Obrázek 2.6: Úvodní obrazovka a Správce stezek v RN klientovi



(a) : Mapa úkolů stezky **(b) :** Obrazovka úkolu **(c) :** Obrazovka s otázkou

Obrázek 2.7: Prohlížení virtuální naučné stezky v RN klientovi

Po otevření stezky se zobrazí úvodní obrazovka stezky. Jednotlivé úkoly je možné otevírat buď pomocí mapy (viz obrázek 2.7a) nebo pomocí seznamu úkolů. Po otevření úkolu se nám otevře obrazovka úkolu, která obsahuje tvořitelný obsah zmíněný výše v subsekci 2.1. V případě obrázku 2.7b se jedná o obsah typu *audio, description* a *images*. Na obrazovce na obrázku 2.7c lze vidět jednu otázku typu *Singlechoice*.

Data o vyplnění otázek se v aplikaci uchovávají v databázi typu *klíč/hodnota* z knihovny *AsyncStorage*, kterou poskytuje samotný *React Native*. Tímto způsobem zůstanou data uchovaná i po zavření aplikace[23].

2.4 Flutter klient

Flutter klient je napsaný v jazyce *Dart* s využitím frameworku Flutter od společnosti Google. [2] Flutter je framework pro tvorbu multiplatformních nativně kompilovaných aplikací pro systémy *Android*, *iOS*, webové aplikace a desktopové aplikace. Na rozdíl od frameworku React Native se nespoléhá na nativní komponenty jednotlivých systémů, ale k vykreslování uživatelského rozhraní používá 2D renderovací engine *Skia*. Vývojář tak má větší kontrolu nad každým pixelem, který se vykresluje na obrazovku. Komponentu v rámci Flutteru nazýváme *widget*. [8]

Po spuštění aplikace se otevře úvodní obrazovka (viz obrázek 2.8a) s přihlašovacím formulářem. Přístup do systému uděluje správce uživatelských účtů v *Editoru*. Bez něj není možné aplikaci spustit.

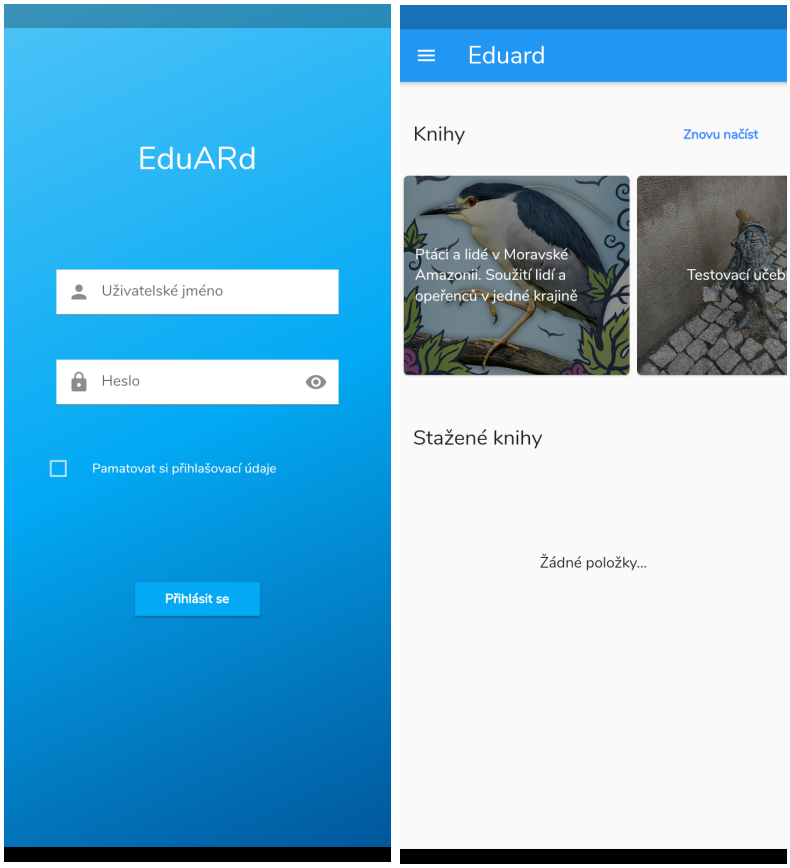
Po přihlášení se otevře obrazovka, na které lze stahovat a mazat dostupné stezky do mobilního zařízení. Pro ukládání informací o stezkách používá *Flutter klient* lokální databázi *Hive*, která funguje na principu klíč/hodnota. Hive je schopná i pomocí adaptérů zadané záznamy databáze *serializovat*[24]. Je proto možné ukládat i složitější datové objekty a struktury [25]. Hive dále uchovává informace o virtuální naučné stezce, její veškerý tvořitelný obsah a informaci o vyplněných otázkách a o správnosti všech odpovědí.

Na úvodní obrazovce stezek (viz 2.8b) nejprve ze seznamu *Knihy* vybereme stezku, kterou chceme stáhnout. Kliknutím na kartu stezky, provedeme její stažení. Po stažení se stezka objeví v seznamu *Stažené knihy*. Staženou stezku pak otevřeme kliknutím na její kartu.

[23] Maňour, str. 37

[24] Serializace slouží ke konverzi objektů do bajtových streamů, které je pak možné ukládat do paměti, databáze nebo do souboru. Hlavním cílem serializace je uchování stavu objektu pro to, abychom byli schopni jej znovu vytvořit v tom stavu, v jakém jsme jej uložili. `https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/serialization/` [cit. 2022-05-14]

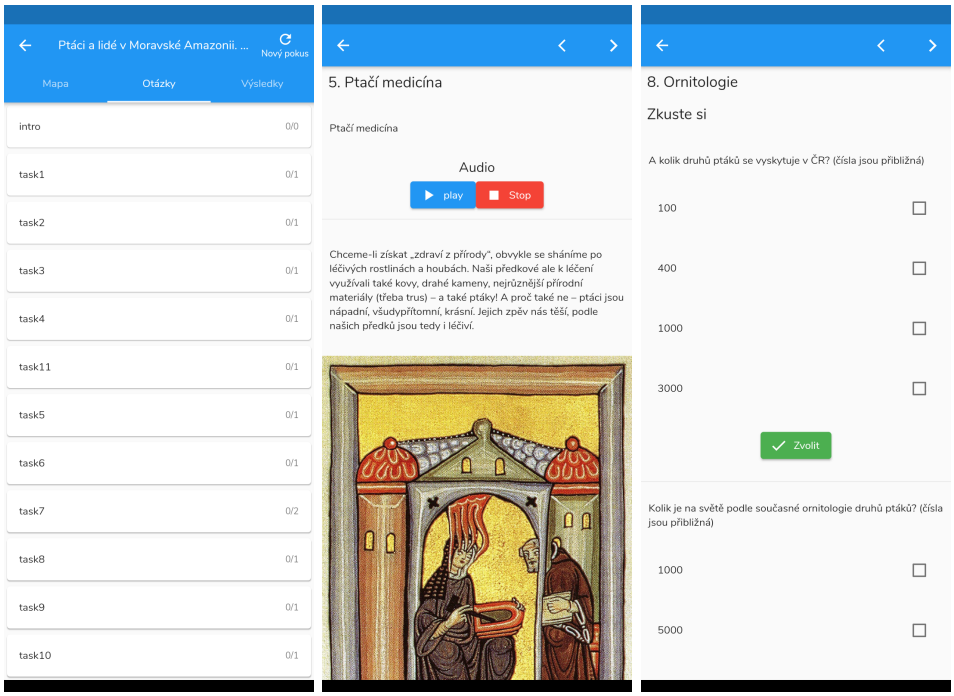
[25] Šimon, str. 27



(a) : Úvodní obrazovka **(b) :** Správce stezek

Obrázek 2.8: Úvodní obrazovka a správce stezek ve Flutter klientovi

Podobně jako u *RN klienta* i zde existují dva přístupy zobrazení úkolů virtuální naučné stezky - pomocí mapy a jak lze vidět na obrázku 2.9a také pomocí seznamu. Po kliknutí na kartu úkolu se otevře obrazovka úkolu. Ta je opět jako v případě *RN klienta* posuvná a obsahuje tvořitelný obsah. Na obrázku 2.9b lze vidět obrazovku úkolu, která obsahuje nadpis, audio nahrávku, odstavec textu a obrázek. Na posledním obrázku 2.9c můžeme vidět obrazovku se dvěma otázkami typu *Singlechoice*.



(a) : Seznam úkolů stezky **(b) :** Obrazovka úkolu **(c) :** Obrazovka s otázkami

Obrázek 2.9: Prohlížení virtuální naučné stezky ve Flutter klientovi

Kapitola 3

Návrh interaktivních úloh

V této kapitole krátce představíme interaktivní úlohy, jejich grafický návrh pro obě mobilní aplikace a návrh struktury úlohy pro soubor virtuální naučné stezky ve formátu XML.

Pro návrh grafického rozhraní úloh byl použit bezplatný nástroje pro tvorbu uživatelských rozhraní *Figma*.[1]

3.1 Rozvržení obrazovky úlohy

V aktuálním stavu obou mobilních aplikací se tvořitelný obsah jednoho úkolu zobrazuje všechen na jedné obrazovce, která je posuvná. Toto řešení není příliš šťastné vzhledem k tomu, že bychom chtěli, aby se uživatel plně soustředil na plnění interaktivní úlohy a mohlo by se jednoduše stát, že se omylem v rámci obrazovky posune nahoru nebo dolů. Pro tento případ byl navržen design, kde se na obrazovce úlohy objeví pouze *Karta interaktivní úlohy* s krátkým popisem úlohy a tlačítkem, které uživatele přesune na obrazovku *Detail interaktivní úlohy*, kde bude uživatel úlohu plnit.

Zároveň chceme uživateli měřit čas, který mu úloha zabere, tudíž požadujeme, aby měl kontrolu nad tím, kdy se čas začne odpočítávat.

3.2 Představení interaktivních úloh

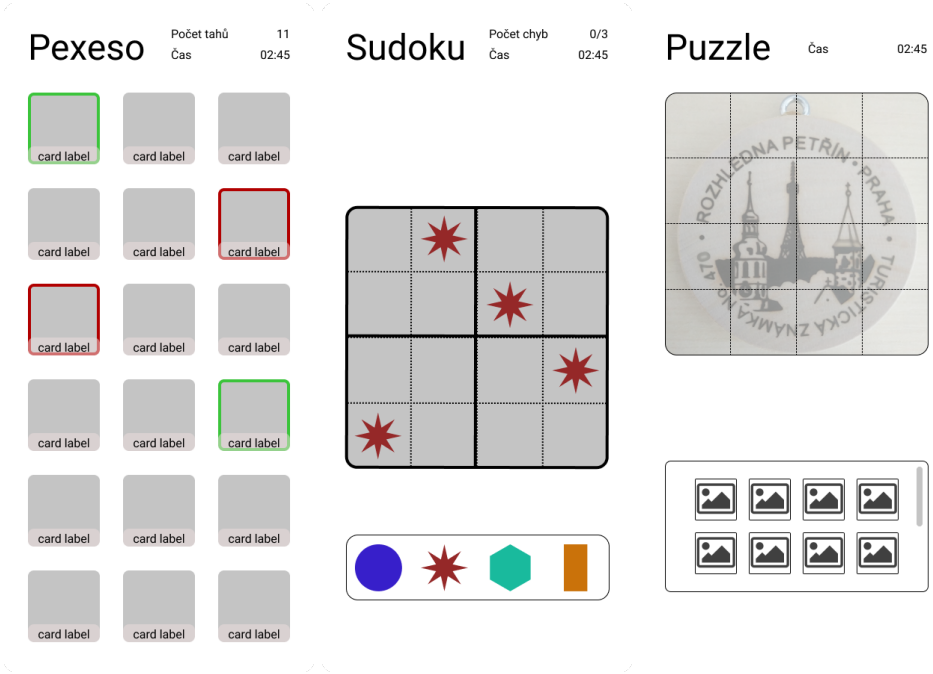
Při návrhu jednotlivých úloh bylo nutné:

[1] Figma je dostupná na `https://www.figma.com/`

1. aby nebyly příliš lehké, ale zároveň ani příliš těžké
2. aby měl uživatel motivaci a důvod je plnit
3. aby bylo možné je v reálném čase naimplementovat a otestovat jejich funkčnost

3.2.1 Interaktivní úloha Pexeso

Pexeso je hra na paměť pro jednoho a více hráčů. Hraje se s většinou obrázkovými kartičkami, které se vyskytují v párech. Na herní plochu se vyskládají hrací kartičky lícem dolů. Hráči se střídají v otáčení dvou kartiček. Pokud se obrázky nebo symboly na kartičkách shodují, hráč si kartičky bere k sobě a dostává bod. V opačném případě otočí kartičky zpět lícem dolů. Pozice kartiček se během hry nemění. Hra končí v tu chvíli, kdy na herní ploše nejsou žádné kartičky. Vyhrává ten hráč, který má na konci hry nejvíce bodů.



(a) : Pexeso **(b) :** Obrázkové sudoku **(c) :** Puzzle

Obrázek 3.1: Návrhy úloh ve Figmě, první část

V naší variantě implementace budeme počítat pouze s jedním hráčem a cílem hry je získat všechny dvojice kartiček za nejmenší počet tahů v co nejkratším čase.

Hra probíhá vždy se sudým počtem kartiček, tudíž je nutné zajistit, aby v *Editoru* nebylo možné zadat lichý počet kartiček. Dále je nutné, aby kartiček

nebylo příliš mnoho, proto aby se vešly na celou obrazovku a byly stále dobře čitelné. V rámci tvorby úlohy v *Editoru* by bylo dobré, kdyby uživatel měl možnost nastavit vzhled jednotlivých kartiček nahráním obrázku.

Samotná hra na mobilním klientovi by pak probíhala zhruba takto: uživatel postupně vybere dvě kartičky, které se otočí lícem nahoru. Systém pak vyhodnotí, zda jsou otočené kartičky shodné. V případě že ano, obarví je zeleně a nechá otočené. V opačném případě, je obarví červeně a po krátké době je opět otočí lícem dolů. Tyto kroky se opakují do té doby než nejsou lícem nahoru otočeny všechny kartičky, což značí konec hry a výhru uživatele.

Pokud uživatel nestihne v časovém limitu sesbírat tj. otočit lícem nahoru všechny kartičky, hra končí prohrou uživatele. Ten už nebude moci s hrou dále interagovat, ale bude mít možnost zkusit hru hrát znovu.

Návrh grafického rozhraní úlohy popisuje obrázek 3.1a.

3.2.2 Interaktivní úloha Obrázkové sudoku

Sudoku je hra pro jednoho hráče. Má jednoduchá pravidla: v každém řádku, v každém sloupci a v každém čtverci o čtyřech polích musejí být obrázky z nabídky, přičemž každý z nich tu musí být vždy jen jednou. Některé obrázky jsou již do pole doplněny před začátkem hry. Cílem hry je vyplnit celé pole obrázky, tak aby byla splněna všechna pravidla.

V naší variantě implementace se bude uživatel snažit vyplnit celé pole v co nejkratším čase a za nejmenší možný počet tahů.

Pro implementaci vytváření úlohy v *Editoru* budeme požadovat, aby uživatel měl možnost zvolit jaké obrázky pro hru se mají použít. Zároveň budeme chtít, aby měl možnost vybrat obtížnost sudoku. V závislosti na obtížnosti se pak na mobilním klientu některá pole zobrazí a jiná zůstanou skrytá. Uživatel mobilního klienta bude potom muset chybějící symboly či obrázky doplnit. Jelikož by se při vytváření sudoku v *Editoru* muselo pokaždé testovat, zda je zadané pole validní, nebylo by od věci, kdyby *Editor* obsahoval nějaký generátor řešení, který by uživatel vytvářející úlohu mohl použít. Tím by se komplexita tvorby validního herního pole pro sudoku přenesla na *Editor*.

Samotná hra na mobilním klientovi by probíhala zhruba takto: klient na základě definované obtížnosti zakryje některá z polí herního pole. Uživatel vybere pole, do kterého chce doplnit symbol nebo obrázek. To se následně zvýrazní. Poté uživatel vybere symbol nebo obrázek, který chce do pole doplnit. Ten se doplní do pole a systém ověří správnost přiřazení. V případě, že je přiřazení chybné, pole se obarví červeně. Hra končí správným přiřazením všech symbolů nebo obrázků do herního pole. V případě, že uživatel nestihl správně přiřadit všechny symboly nebo obrázky do herního pole hra končí

prohrou uživatele. Opět, tak jako v případě *Pexesa* a ve všech dalších úlohách, bude mít možnost zkusit hrát hru znovu.

Návrh grafického rozhraní úlohy popisuje obrázek 3.1b.

3.2.3 Interaktivní úloha Puzzle

Puzzle je hra pro jednoho a více hráčů. Cílem hry je ze zamíchaných obrázkových dílů sestavit celý obrázek. V naší variantě implementace se bude uživatel snažit poskládat celý obrázek co nejrychleji a za nejmenší počet tahů.

Uživatel tvořící úlohu v *Editoru* bude mít možnost vybrat obrázek, který se v mobilním klientovi rozseká a počet dílků, na který se má obrázek rozsekat.

Hra na mobilním klientovi by probíhala zhruba takto: klient nejdříve na základě definovaného obrázku a počtu dílků obrázek rozdělí a jednotlivé dílky zamíchá. Uživatel pak příslušné dílky přesouvá z oblasti obrázkových dílků na herní plochu, kde se skládá výsledný obrázek. Dílky bude možné pokládat pouze na předem určená políčka. Vyhodnocení správnosti poskládání obrázku proběhne po položení všech dílků. Pokud složený obrázek odpovídá původnímu obrázku, hra končí vítězstvím uživatele. V opačném případě uživatel musí obrázkové dílky mezi sebou vyměňovat do té doby, než bude složený obrázek shodný s původním obrázkem. Pokud se to uživateli nepovede v rámci časového limitu, hra končí prohrou.

Návrh grafického rozhraní úlohy popisuje obrázek 3.1c.

3.2.4 Interaktivní úloha Loydova patnáctka

Loydova patnáctka je hlavolam pro jednoho hráče. Skládá se z 15 čtverců, označených od 1 do 15. Tyto čtverce jsou položeny do krabice o velikosti 4×4. Poslední pole zůstává prázdné. Cílem hlavolamu je posuvnými pohyby z nějakého původního uspořádání přes prázdné pole přemístit čtverce tak, aby byly v vzestupném pořadí tj. od 1 do 15. Pro některé z těchto původních uspořádání neexistuje správné řešení. [12]

V naší verzi budeme místo čtverců s čísly uvažovat čtvercové dílky zadaného obrázku. Tvorba úlohy v *Editoru* by měla uživatelům umožnit volbu původního obrázku a velikosti hlavolamu. Podobně jako v případě sudoku by bylo dobré dát možnost uživateli vygenerovat původní uspořádání místo toho, aby musel toto původní uspořádání vymýšlet sám. Předešlo by se tím i případům, že by uživatel nedopatřením vytvořil neřešitelné zadání.

Na mobilním klientovi by pak úloha mohla probíhat zhruba takto: podobně jako u puzzle, klient nejdříve na základě definovaného obrázku a počtu

dílků obrázek rozdělí a jednotlivé dílky vyskládá na herní plochu podle definovaného původního uspořádání. Uživatel se pak posouváním čtvercových polí do prázdného prostoru snaží poskládat původní obrázek. Vyhodnocování shodnosti obrázků bude probíhat po každém přesunu čtvercových polí. V případě, že obrázky rovnají, hra končí a uživatel vítězí. Pokud se uživateli nepovede přesunout čtvercová pole na správná místa v časovém limitu, hra končí a uživatel prohrává.

Návrh grafického rozhraní úlohy popisuje obrázek 3.2a.

(a) : Loydova patnáctka **(b) :** Křížovka

Obrázek 3.2: Návrhy úloh ve Figmě, druhá část

3.2.5 Interaktivní úloha Křížovka

Křížovka je vědomostní úloha, ve které je cílem doplnit správně slova do vyznačených polí podle nápovědy. Výsledkem luštění křížovky je tajenka, kterou se snažíme uhodnout.

Uživatel vytvářející úlohu v *Editoru* specifikuje nejdříve tajenku, okolo které se křížovka bude sestavovat. Pro každé písmeno tajenky se mu vytvoří položka formuláře, kam bude možné zadat nápovědu a správnou odpověď.

Délku tajenky budeme muset nějakým způsobem omezit, aby se na obrazovku mobilního klienta vešlo celé křížovkové pole společně s nápovědami.

Na mobilním klientovi nejprve načteme křížovkové pole společně s nápovědami. Poté uživatel kliknutím na jednotlivé řádky křížovky doplňuje slova podle dané nápovědy. Až uživatel doplní všechna slova do křížovky, systém vyhodnotí správnost jednotlivých odpovědí. V případě, že jsou všechna slova správně vyplněná, uživatel vítězí. V opačném případě musí zkusit doplnit slova znova. Pokud nestihne doplnit všechna slova správně v časovém limitu, hra končí prohrou.

Návrh grafického rozhraní úlohy popisuje obrázek 3.2b.

3.3 Odměna po splnění úlohy

Po jedné z konzultacích s vedoucím bylo rozhodnuto, že by uživatel měl být nějakým způsobem notifikovaný o úspěšném splnění úlohy. Do *Editoru* bude přidána možnost definovat audio nahrávku, která se spustí při úspěšném splnění úlohy. Tato audio nahrávka bude nadefinovaná pro celou učebnici a uživatel bude mít možnost u každé z úloh rozhodnout, zda se má audio přehrát nebo ne.

Zároveň, abychom uživatele motivovali k plnění úloh a dali mu důvod u plnění úloh se snažit, bylo rozhodnuto, že se po úspěšném splnění úloh zobrazí vítězný obrázek. To bude platit pro úlohy sudoku a křížovka. V případě pexesa dostane uživatel za odměnu kartičky použité v úloze. V případě puzzle a Loydovy patnáctky bude uživateli odměnou složený obrázek.

3.4 Cílová skupina

Pro potřeby *user-centered designu* (designu zaměřeného na uživatele) je nutné vybrat cílovou skupinu uživatelů, pro kterou daný produkt vytváříme.

3.4.1 User-centered design

Design zaměřený na uživatele (*user-centered design*, zkráceně *UCD*) je metodika vývoje softwaru, která se zaměřuje na získání povědomí o cílové skupině uživatelů, kteří budou vyvíjený produkt využívat. Design produktu je založen na porozumění uživatele, jeho potřebám a účelu, který má daný produkt plnit. Vyhodnocení toho, zda byly uživatelské potřeby naplněny je vykonáváno přímo uživateli z cílové skupiny. Hlavními fázemi *UCD* jsou: [7]

1. Nalezení a identifikování cílové skupiny uživatelů
2. Vymezení byznysových a funkčních požadavků na design aplikace
3. Vytvoření designu na základě požadavků - tento krok může probíhat iterativně v několika kolech
4. Vyhodnocení designu - ideálně testováním přímo s uživateli

3.4.2 Původní cílová skupina

Z předchozí analýzy autorky *Editoru* vyšlo, že cílová skupina pro aplikaci *Editoru* jsou zaměstnanci školy a to zejména učitelé, kteří vytvářejí virtuální naučné stezky.[2] Přidáním funkcionality tvorby interaktivních úloh cílovou skupinu nezměníme. Budeme se tedy soustředit na to, aby grafické rozhraní pro tvorbu interaktivních úloh odpovídalo co nejvěrněji grafickému rozhraní původní aplikace.

Bohužel podobná analýza cílové skupiny neproběhla ani u *RN klienta*, ani u *Flutter klienta*. Budeme tedy předpokládat, že hlavní cílovou skupinou budou žáci školy. Opět se budeme snažit o to, abychom příliš nezasáhli do původního designu a nové komponenty a funkcionalitu přizpůsobili tak, aby respektovala původní návrh a architekturu obou mobilních klientů.

3.5 Požadavky

V této sekce shrneme požadavky na implementaci, které vycházejí z návrhu popsaného výše.

3.5.1 Funkční požadavky

Obecné požadavky pro každou úlohu

- **FR01 - Audio při splnění úlohy:** Požadujeme, aby uživatel *Editoru* mohl nahrát, změnit a odebrat audio nahrávku, která se má přehrát po splnění interaktivní úlohy
- **FR02 - Obrázek při splnění úlohy:** Požadujeme, aby uživatel *Editoru* mohl nahrát, změnit a odebrat obrázek, který se má zobrazit po splnění interaktivní úlohy

[2] Surikova, str. 6

- **FR03 - Odměna uživateli za splnění úlohy:** Chceme uživatele obou mobilních aplikací motivovat k plnění interaktivních úloh tím, že jim za splnění úlohy dáme nějakou odměnu - přehrajeme vítězné audio nebo zobrazíme vítězný obrázek
- **FR04 - Nastavení časového limitu pro splnění úlohy:** Požadujeme, aby uživatel *Editoru* mohl specifikovat, upravit a odebrat časový limit pro splnění interaktivní úlohy
- **FR05 - Měření času při plnění interaktivní úlohy:** Požadujeme, aby uživatelé obou mobilních klientů měli kontrolu nad tím, kdy se začne odpočítávat časový limit pro splnění interaktivní úlohy
- **FR06 - Splnění interaktivní úlohy:** Požadujeme, aby uživatelé obou mobilních klientů byly notifikovaní o splnění interaktivní úlohy a aby bylo zřejmé, které úlohy uživatel splnil a které ne
- **FR07 - Vypršení časového limitu:** Požadujeme, aby uživatelé obou mobilních klientů byly notifikovaní v případě, že jim vypršel čas pro splnění dané úlohy a aby s úlohou nemohli dále interagovat
- **FR08 - Možnost opakování již splněných úloh:** Požadujeme, aby uživatelé obou mobilních klientů mohli libovolně opakovat interaktivní úlohy, které byly již jednou splněny

Specifické požadavky pro jednotlivé úlohy

- **FR09 - Pexeso:** Požadujeme, aby uživatel *Editoru* mohl vytvořit, upravit a vymazat interaktivní úlohu Pexeso
- **FR10 - Pexeso, správa kartiček:** Požadujeme, aby uživatel *Editoru* mohl vytvořit, upravit a vymazat kartičky k interaktivní úloze Pexeso a zároveň přidat, změnit a odebrat vlastní obrázek ke každé z kartiček
- **FR11 - Spuštění a dokončení úlohy Pexeso:** Požadujeme, aby uživatel obou mobilních klientů byl schopen spustit úlohu Pexeso a aby ji mohl úspěšně dokončit
- **FR12 - Sudoku:** Požadujeme, aby uživatel *Editoru* mohl vytvořit, upravit a vymazat interaktivní úlohu Sudoku
- **FR13 - Sudoku, generátor řešení:** Požadujeme, aby uživatel *Editoru* mohl využít generátoru řešení Sudoku pro tvorbu interaktivní úlohy Sudoku
- **FR14 - Spuštění a dokončení úlohy Sudoku:** Požadujeme, aby uživatel obou mobilních klientů byl schopen spustit úlohu Sudoku a aby ji mohl úspěšně dokončit

- **FR15 - Puzzle:** Požadujeme, aby uživatel *Editoru* mohl vytvořit, upravit a vymazat interaktivní úlohu Puzzle
- **FR16 - Spuštění a dokončení úlohy Puzzle:** Požadujeme, aby uživatel obou mobilních klientů byl schopen spustit úlohu Puzzle a aby ji mohl úspěšně dokončit
- **FR17 - Loydova patnáctka:** Požadujeme, aby uživatel *Editoru* mohl vytvořit, upravit a vymazat interaktivní úlohu Loydova patnáctka
- **FR18 - Loydova patnáctka, generátor řešení:** Požadujeme, aby uživatel *Editoru* mohl využít generátoru řešení Loydova patnáctky pro tvorbu interaktivní úlohy Loydova patnáctka
- **FR19 - Spuštění a dokončení úlohy Loydova patnáctka:** Požadujeme, aby uživatel obou mobilních klientů byl schopen spustit úlohu Loydova patnáctka a aby ji mohl úspěšně dokončit
- **FR20 - Křížovka:** Požadujeme, aby uživatel *Editoru* mohl vytvořit, upravit a vymazat interaktivní úlohu Křížovka
- **FR21 - Spuštění a dokončení úlohy Křížovka:** Požadujeme, aby uživatel obou mobilních klientů byl schopen spustit úlohu Křížovka a aby ji mohl úspěšně dokončit

3.5.2 Kvalitativní požadavky

- **QR01 - Zachování designu všech rozšiřovaných aplikací:** Požadujeme, aby se design rozšiřovaných částí aplikací shodoval nebo co nejméně lišil od už vyvinutých částí aplikací.
- **QR02 - Intuitivní uživatelské rozhraní:** Požadujeme, aby design rozšiřovaných částí aplikace byl intuitivní a aby nepůsobil problémy s používání žádnému z uživatelů.

3.6 Rozšíření XML struktury

Rozšíření struktury pro ukládání a načítání dat o virtuální naučné stezce vychází z původní struktury dat popsané v sekci 2.1.

Nejpřímočařejší variantou rozšíření by bylo rozšířit původní značku `<question />` o nový typ, který by charakterizoval interaktivní úlohy. Výhoda tohoto přístupu by byla v tom, že ve všech projektech již existují služby, které dokáží tuto značku parsovat a přiřadit k objektům otázky. Nicméně vzhledem k tomu, že už tak značka `<question />` obsahuje mnoho typů, bylo rozhodnuto

tento seznam typů dále nerozšiřovat a raději zavést pro interaktivní úlohu novou značku `<intertask />`. Nová struktura vypadá takto:

```
<questionset> ... učebnice / virtuální naučná stezka
 <task> ... úkol (0 až N)
  <location> ... geolokační data úkolu
   <latitude/>
   <longitude/>
  </location>
  <questionslide> ... obrazovka úkolu (0 až N)
   <questions>
    <question /> ... otázka (0 až N)
    <intertask /> ... interaktivní úloha (0 až N)
    <description /> ... text (0 až N)
    <images /> ... obrázek (0 až N)
    <audio /> ... zvuková nahrávka (0 až N)
    <video /> ... video (0 až N)
   </questions>
  </questionslide>
 </task>
</questionset>
```

Nově značka `<questionset />` obsahuje atribut `rewardAudio`, který obsahuje název audio souboru, který se má přehrát po úspěšném splnění úlohy.

3.6.1 XML struktura jednotlivých úloh

Značka `<intertask />` obsahuje atributy:

- `order` značí pořadí obsahu v rámci obrazovky
- `type` je jeden z typů interaktivní úlohy: `memory` pro pexeso, `sudoku` pro sudoku, `puzzle` pro puzzle, `sliding_puzzle` pro Loydovu patnáctku a `crossword` pro křížovku
- `title` je nadpis interaktivní úlohy
- příznak `playRewardAudio` značí, zda se má při splnění úlohy přehrát audio nahrávka
- pokud je přítomen atribut `timeLimit`, pak vyjadřuje časový limit pro splnění úlohy v sekundách
- pouze pro Sudoku:

  - `difficulty` může nabývat hodnot *easy* a *hard* a značí, kolik polí se má odkrýt při spuštění úlohy na mobilním klientovi

- pouze pro Sudoku a Křížovku:
  - `rewardImage` obsahuje název souboru obrázku, který se má zobrazit po úspěšném splnění úlohy

- pouze pro Puzzle a Loydovu patnáctku:
  - `size` specifikuje, na kolik částí se má obrázek v rámci klienta před zahájením úlohy rozdělit

Dále obsahuje potomka `<description />`, který ve svém těle obsahuje CDATA[3] s popisem úlohy.

Mimo jiné každá z úloh obsahuje další potomky, které jsou charakteristické pouze pro ni.

Pexeso

Značka `<card />` slouží pro zápis dat o jednotlivých kartičkách *Pexesa*. Každá z nich obsahuje potomka `<image />`, který ve svém těle nese informaci o jménu souboru obrázku.

Obrázkové sudoku

Značka `<images />` ve svých potomcích `<image />` podobně jako u *Pexesa* nese informaci o obrázcích použitých v *Sudoku*. Dále úloha obsahuje značku `<solution />`, která nese informaci o tom, jak vypadá řešení *Sudoku* jako textový řetězec.

Puzzle

Puzzle obsahuje značku `<image />`, která obsahuje název souboru obrázku, který se má v klientské aplikaci rozsekat před spuštěním úlohy

[3] CDATA - Character data jsou části XML, které nejsou považovány za součást značkovacího jazyka a jsou interpretovány jako textový řetězec `https://www.w3.org/TR/REC-xml` [cit. 2022-05-13]

Loydova patnáctka

Obdobně jako *Puzzle* obsahuje značku `<image />`, která zde plní stejný účel. Dále obsahuje značku `<puzzle />`, která nese informaci o tom, jak vypadá původní uspořádání úlohy jako textový řetězec.

Křížovka

Křížovka obsahuje značku `<solution />`, která obsahuje tajenku ve formátu CDATA. Dále obsahuje několik značek `<caption />`, které obsahují atribut `position` označující začáteční index daného řádku v rámci mřížkového uspořádání křížovkového pole, nápovědu `<hint />` a odpověď `<answer />`.

Kapitola 4

Implementace interaktivních úloh

V této části práce popíšeme, jakým způsobem byl návrh interaktivních úloh realizován a jakým způsobem byly naplněny funkční a kvalitativní požadavky.

4.1 Editor

V krátkosti popíšeme způsob, jakým se data o virtuální naučné stezce dostávají do *Editoru*. Potom popíšeme způsob, jakým se data ukládají do formátu XML. Nakonec ukážeme, jak vypadá výsledná implementace.

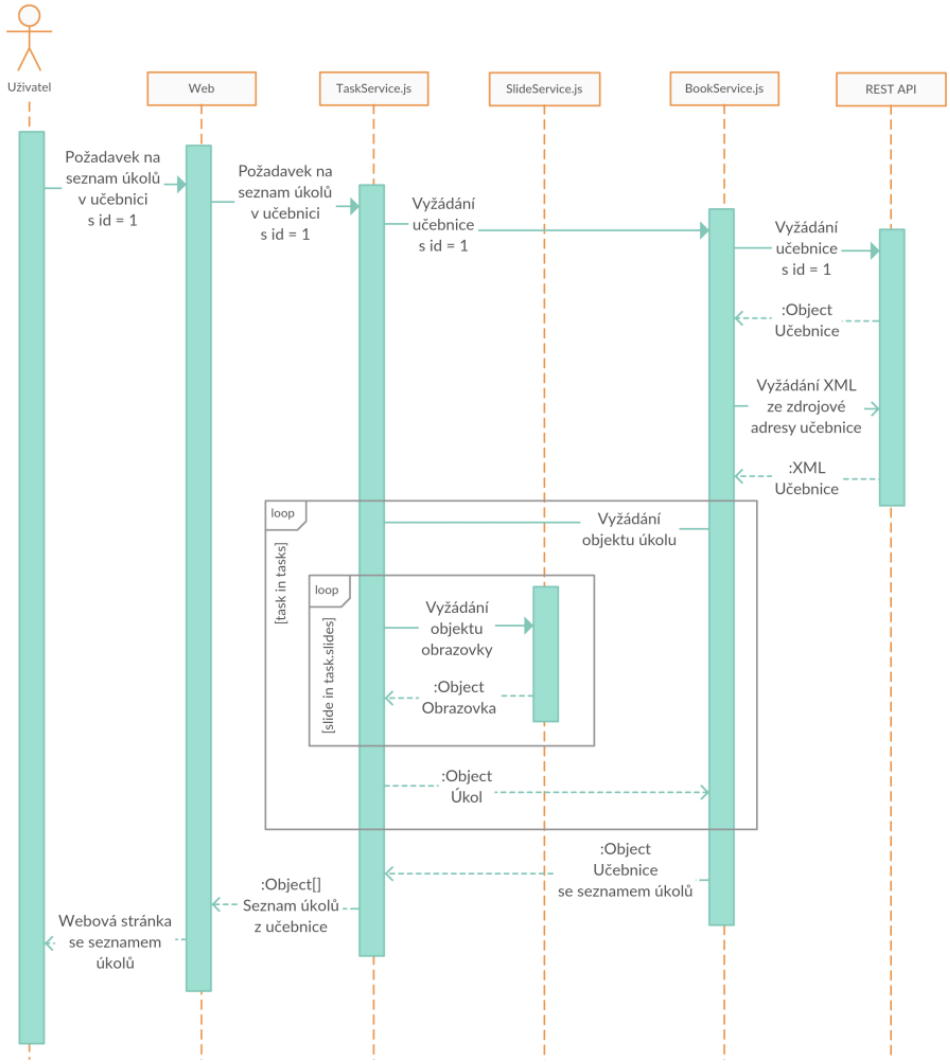
4.1.1 Načítání dat

Jak již bylo zmíněno v sekci 2.1 o struktuře dat, v rámci *Editoru* jsou přítomny služby pro konverzi XML souboru na JavaScriptové objekty. Aplikace ale nejdříve musí tento soubor získat. To je umožněno pomocí služby *BookService*, která volá API back-endu, ze kterého obdrží objekt učebnice. Ten v sobě obsahuje atribut `resourcePath`, který představuje adresu, na které lze stáhnout obsah učebnice ve formátu XML. Ve staženém souboru, tato služba najde značku `<questionset />`, která značí seznam úkolů učebnice. Pak probíhá parsování jednotlivých prvků učebnice, podle toho, na jaké obrazovce *Editoru* se zrovna nacházíme. Jak data probíhají lze vidět na obrázku 4.1, který je převzatý z původní práce autorky.[1]

Nás bude zajímat převážně služba *SlideService*, která se stará o parsování jednotlivých obrazovek úkolu společně s jeho tvořitelným obsahem z formátu XML do JavaScriptových objektů. Tato služba už v sobě obsahuje další služby pro parsování jednotlivých typů obsahu: *QuestionService* starající se o otázky,

[1] Surikova, str. 30

DescriptionService starající se o textový obsah a *ImageService*, která se stará o obrázky a audiovizuální obsah.[2] Po vzoru *QuestionService* je implementována nová služba *InteractiveTaskService*, která zajišťuje parsování interaktivních úloh ze souboru XML. Pracuje s objekty třídy *InteractiveTask*, která obsahuje atributy podle specifikace popsané v sekci 3.6.1.



Obrázek 4.1: Sekvenční diagram zobrazení seznamu úkolů učebnice, převzato z [15]

4.1.2 Ukládání dat

Pro uložení virtuální naučné stezky se v *Editoru* využívá služba *XMLService*. Ta při uživatelově požadavku na uložení učebnice vezme objekt učebnice

[2] Surikova, str. 29

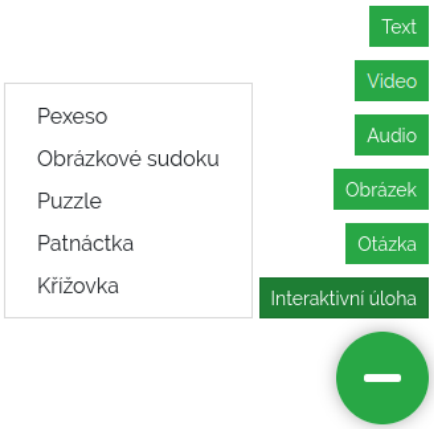
včetně všech obrazovek, úkolů a tvořitelného obsahu a vytvoří z něj XML reprezentaci učebnice, která se pomocí volání back-endu uloží na server.

Pro účely konverze objektů interaktivních úloh bylo nutné ve službě *XMLService* vytvořit novou metodu `_constructInteractiveTask()`, která vytvoří značku `intertask` se všemi atributy popsanými v sekci 3.6.1. Ta dále používá nové pomocné metody pro konverzi jednotlivých interaktivních úloh. Jmenovitě se jedná o:

- `_constructMemory()` pro *Pexeso*,
- `_constructSudoku()` pro *Obrázkové sudoku*,
- `_constructPuzzle()` pro *Puzzle* a *Loydovu patnáctku*,
- `_constructCrossword()` pro *Křížovku*

4.1.3 Uživatelské rozhraní

Jak již bylo zmíněno výše v sekci 2.2, v původní verzi *Editoru* se obsah na obrazovku přidával pomocí zeleného FAB s ikonou plusu. Po kliknutí na něj se otevře menu s výběrem obsahu, který můžeme přidávat. V rámci implementace (viz obrázek 4.2) byla přidána další položka menu *Interaktivní úloha*, která obsahuje nově implementované interaktivní úlohy.

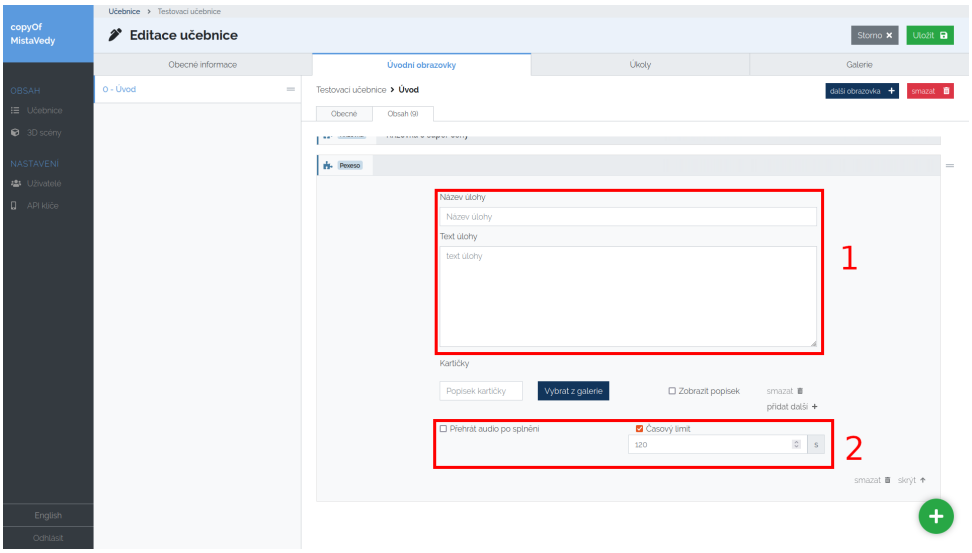


Obrázek 4.2: Nová položka menu přidávající možnost tvorby interaktivní úlohy

Karta interaktivní úlohy

Po kliknutí na jednu z položek menu se vytvoří nová instance komponenty *SlideInteractiveTask*, která představuje kartu interaktivní úlohy, která obsahuje

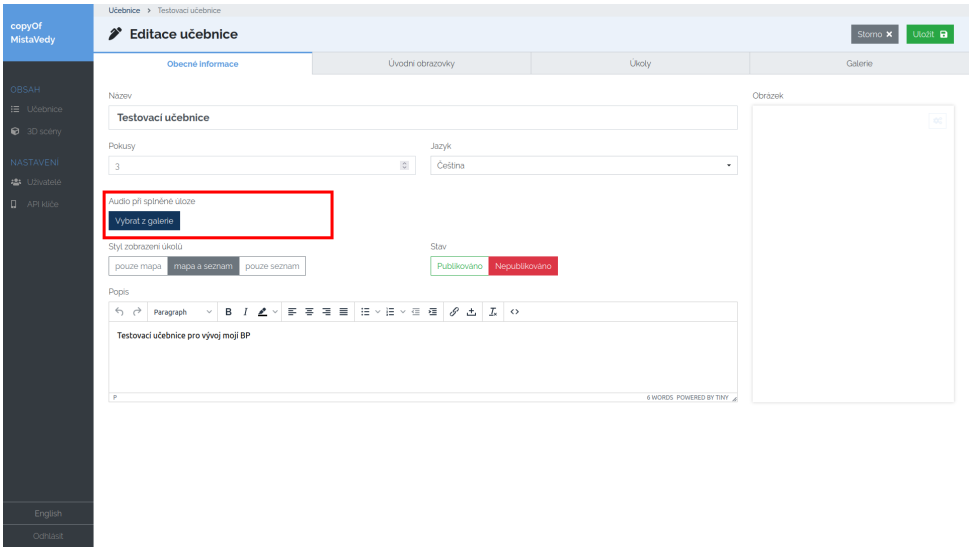
formulář, ve kterém uživatel může definovat jednotlivé prvky interaktivní úlohy. Vzhled a rozpoložení karty bylo výrazně inspirováno vzhledem komponenty pro vytvoření otázky (viz obrázek 2.5). Příklad této karty lze vidět i na obrázku 4.3. Část označená jedničkou je záhlaví formuláře, které obsahuje dvě textová pole (pro vyplnění *názvu úlohy* a *textu úlohy*) a je shodné pro všechny úlohy. Část označená dvojkou je zápatí formuláře a obsahuje dvě zaškrtávací pole. První z nich umožňuje uživateli specifikovat, zda se po splnění dané úlohy přehraje *audio nahrávka*, kterou lze pro celou virtuální naučnou stezku definovat (o tom více v dalším odstavci). Druhé zaškrtávací pole slouží pro specifikování *časového limitu*. Pokud uživatel vybere, že se má aplikovat časový limit, zpřístupní se textové pole, kam uživatel zadá časový limit v sekundách.



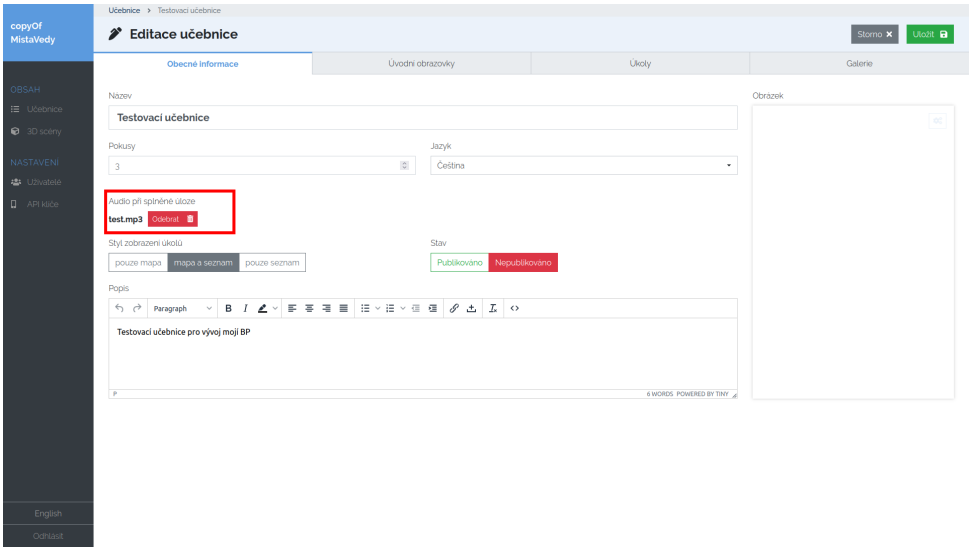
Obrázek 4.3: Ukázka karty formuláře interaktivní úlohy

Audio při splněné úloze

V rámci návrhu implementace jsme definovali požadavek na možnost volby audio nahrávky, která se má přehrát po splnění interaktivní úlohy. Nová komponenta **AudioBlock** se stará o správu této nahrávky. V sobě obsahuje již implementovanou komponentu *Galerie*, která uživateli umožňuje nahrávat do systému obrázky, audio nahrávky a videa. Nahraný obsah, pak lze přiřadit k tvořitelnému obsahu. Na obrázku 4.4 můžeme vidět, jak nově vypadá úvodní obrazovka editace učebnice s novou komponentou *AudioBlock* bez vybrané nahrávky označenou červeným rámečkem. Na obrázku 4.5 je opět v červeném rámečku vidět varianta, kdy je jako audio nahrávka vybrán soubor s názvem *test.mp3*.



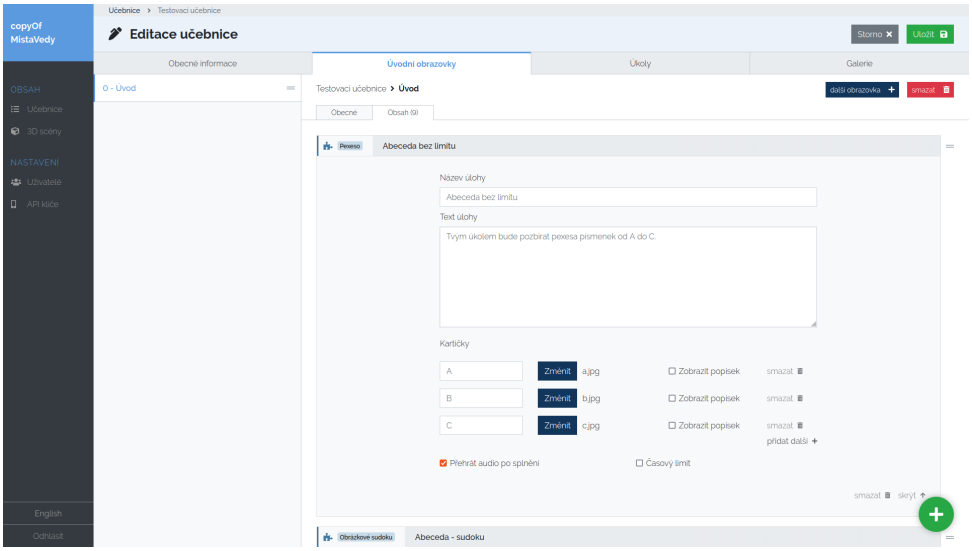
Obrázek 4.4: Kolonka pro výběr audio nahrávky na hlavní stránce editace učebnice - bez zvolené audio nahrávky



Obrázek 4.5: Kolonka pro výběr audio nahrávky na hlavní stránce editace učebnice - se zvolenou audio nahrávkou

Interaktivní úlohy

V této podsekci popíšeme, jakým způsobem jsou implementovány jednotlivé formuláře pro tvorbu interaktivních úloh. O tom, jakým způsobem se vytvořené úlohy zobrazují na obou mobilních klientech se dozvíte v následujících sekcích.



Obrázek 4.6: Vyplněný formulář pro úpravu interaktivní úlohy Pexeso

Interaktivní úloha Pexeso

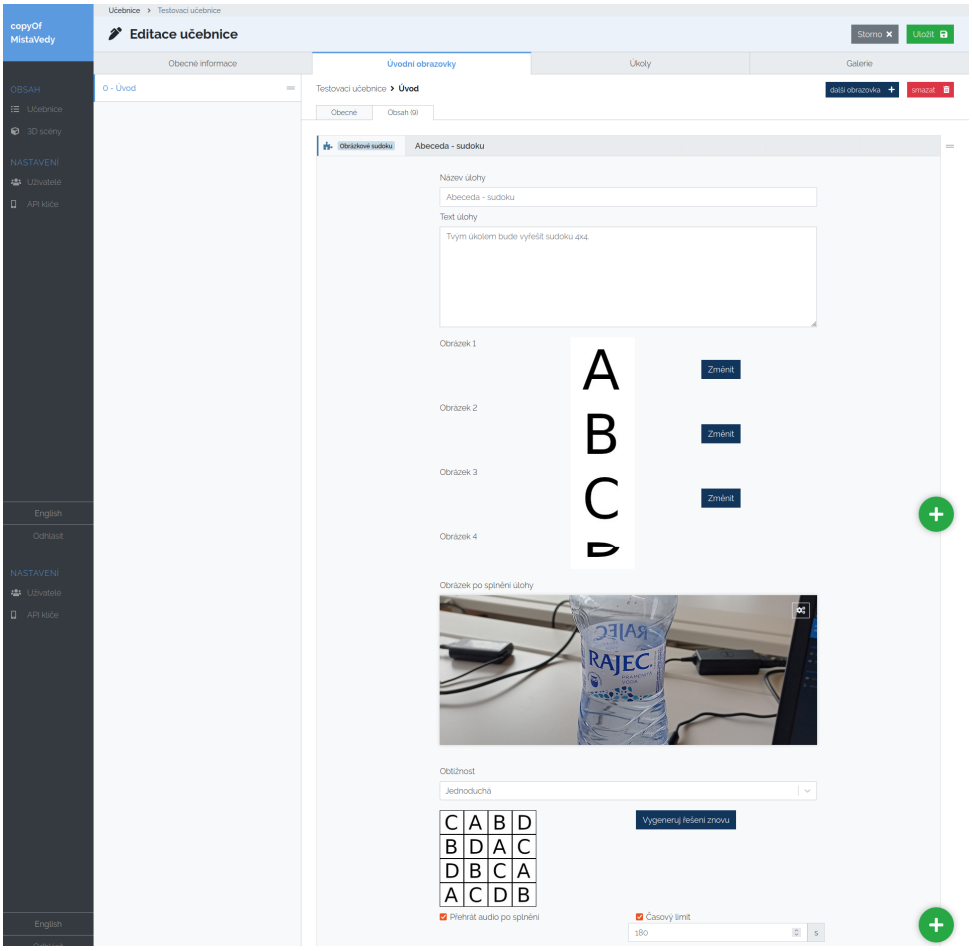
Formulář pro tvorbu interaktivní úlohy **Pexeso** se skládá ze záhlaví a zápatí, které je shodné pro každou úlohu. Toto záhlaví a zápatí jsme definovali o dva odstavce výše (v sekci 4.1.3) a při popisu dalších úloh je již zmiňovat nebudeme.

Důležité pro nás bude tělo formuláře. Jak lze vidět z obrázku 4.6, tělo formuláře obsahuje seznam kartiček, které budou použity pro hru *Pexesa*. Bylo zvoleno, že těchto kartiček bude možné použít pouze omezený počet. To aby se všechny vešly na obrazovku mobilního telefonu a byly dobře čitelné. Maximální počet kartiček je **18**, což odpovídá rozložení 3×6. *Editor* podporuje tvorbu jednotlivých kartiček, nikoliv párů a proto lze vytvořit pouze **9** *unikátních* kartiček. Pokud se uživatel pokusí vytvořit více kartiček, systém vypíše chybovou hlášku. Každá kartička obsahuje popisek, má přiřazený obrázek z *Galerie*, který lze libovolně měnit a informaci o tom, zda se má popisek na kartičce ukazovat nebo ne.

Interaktivní úloha Obrázkové sudoku

Druhou implementovanou úlohou bylo **Obrázkové sudoku**. Na obrázku 4.7 je zobrazeno tělo formuláře pro tuto úlohu. V první části lze opět pomocí *Galerie* vybrat čtyři obrázky nebo symboly, které se použijí v obrázkovém sudoku. Dále lze vybrat obrázek, který se zobrazí při úspěšném vyřešení úlohy. Požadavek na tuto funkcionalitu jsme taktéž zadefinovali v návrhové části.

Uživatel dále může vybrat jednu z obtížností - jednoduchou nebo těžkou.



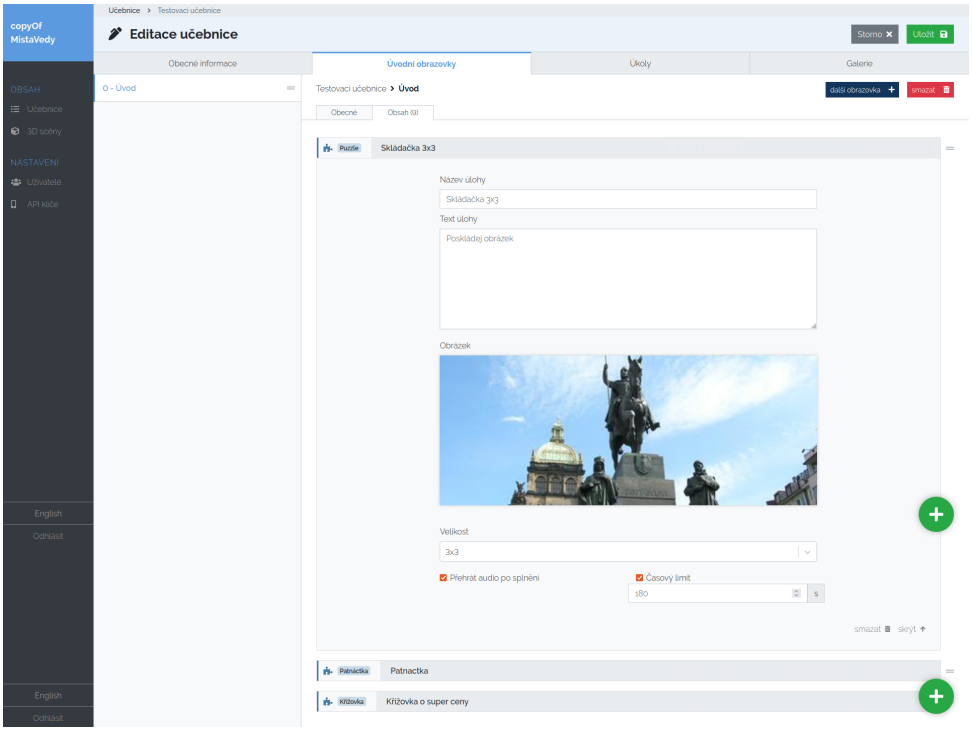
Obrázek 4.7: Vyplněný formulář pro úpravu interaktivní úlohy Obrázkové sudoku

Výběr obtížnosti se projeví při spuštění obrázkového sudoku na mobilním klientovi a to tak, že na jejím základě systém zakryje některá z polí řešení sudoku. Více o tom zmíníme v sekci věnované implementaci na mobilních klientech. Aby uživatel nemusel zadání a řešení sudoku vymýšlet sám, může použít generátor sudoku, který generuje správné řešení. Jako generátor řešení je použita knihovna **mad5dsudoku** od autora *mad5d*, která je dostupná v knihovně balíčků *npm*[3].

Interaktivní úloha Puzzle

Další úlohou kterou jsme v rámci této práce implementovali bylo **Puzzle**. Uživatel má možnost definovat obrázek a počet dílků, na kolik se má tento obrázek rozsekat. Podporované rozměry jsou *3×3* a *4×4*. Vyplněný formulář lze vidět na obrázku 4.8.

[3] `https://www.npmjs.com/package/mad5dsudoku` [cit. 2022-05-16]



Obrázek 4.8: Vyplněný formulář pro úpravu interaktivní úlohy Puzzle

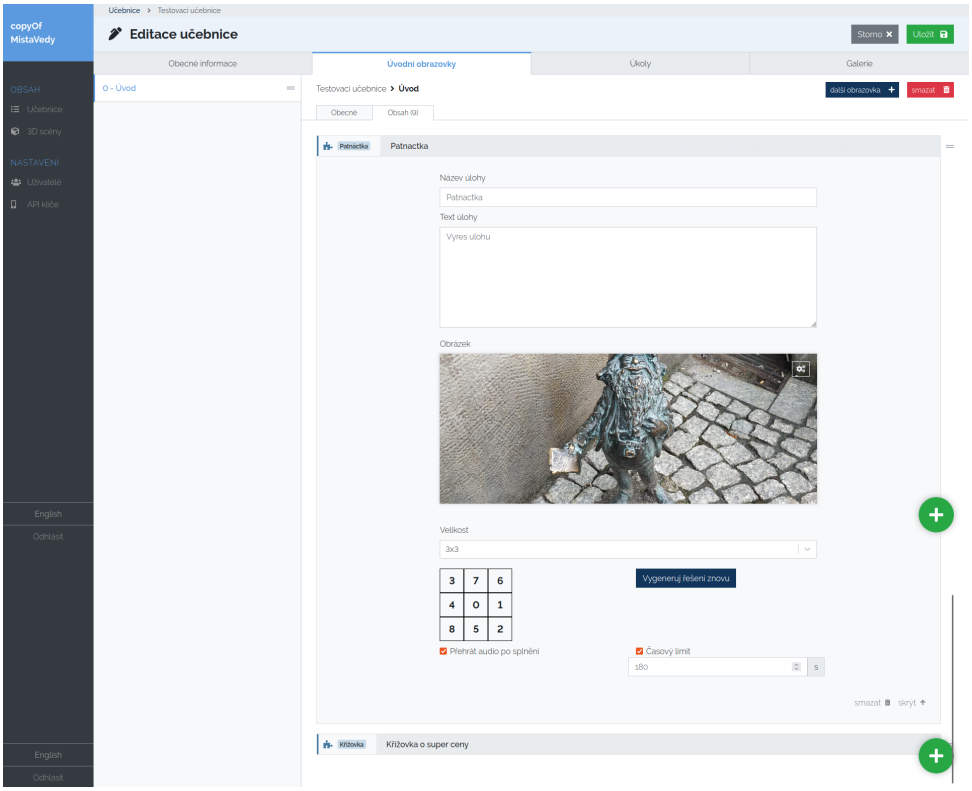
Vzhledem k tomu, že úloha na mobilních klientech funguje správně pouze s obrázky o čtvercových rozměrech, měli bychom o tom uživatele nějakým způsobem informovat. Zatím tato logika implementována není.

Interaktivní úloha Loydova patnáctka

Implementace formuláře pro tvorbu a úpravu úlohy **Loydova patnáctka** se opírá o a implementaci *Puzzle* z předchozího odstavce. Jediné, co nabízí formulář nad rámec formuláře pro Puzzle je možnost generování validního zadání. Obdobně jak tomu bylo v případě *Sudoku*, dáváme uživateli možnost řešení vygenerovat, aby předešel případným chybám při ručním zadávání a aby měl jistotu, že úlohu bude možné vyřešit. Finální vzhled formuláře lze vidět na obrázku 4.9.

O generování zadání úlohy se stará knihovna **15puzzle** od autora *HiraginoYuki*, která je dostupná v knihovně balíčků *npm*[4].

[4] `https://www.npmjs.com/package/15-puzzle` [cit. 2022-05-17]



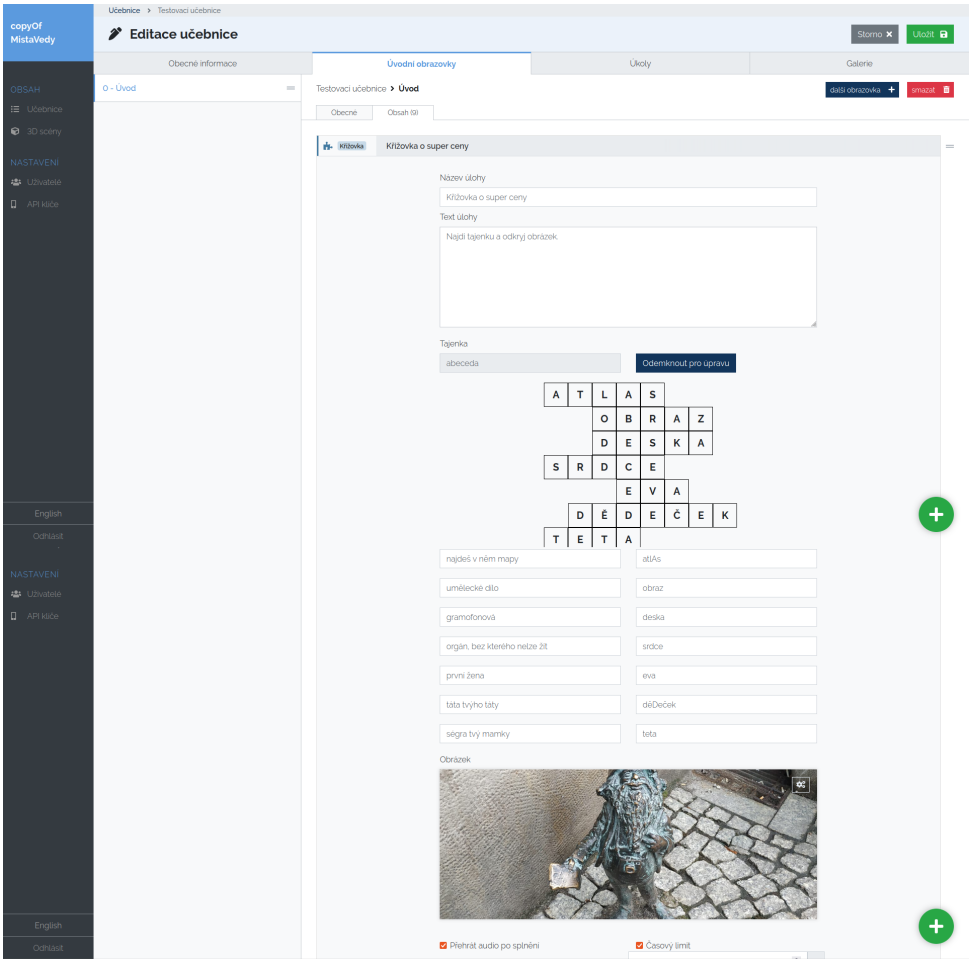
Obrázek 4.9: Vyplněný formulář pro úpravu interaktivní úlohy Loydova patnáctka

Interaktivní úloha Křížovka

Nejkomplexnější úlohou pro implementaci do *Editoru* je **Křížovka**. Jak můžeme vidět z obrázku 4.10, hlavní kostru celé úlohy tvoří *tajenka* a kolem ní je postavený celý formulář pro tvorbu a editaci úlohy.

Pro každé písmeno tajenky se vytvoří tzv. *caption*, která obsahuje *nápovědu* a *odpověď*. Pro každou *caption* se v formuláři vytvoří řádek, kam uživatel může zadat danou *nápovědu* a *odpověď*. Jedinými nároky na odpověď jsou délka odpovědi a to, že odpověď musí obsahovat dané písmeno tajenky. Aplikace na tyto nároky reaguje a obsah políček kontroluje. Jakmile uživatel opustí políčko *tajenka*, tajenka se pro úpravy uzamkne. Zámek se zde používá z toho důvodu, že změnou tajenky uživatel přijde o všechny již vyplněné *captions*. Pro odemknutí tajenky musí uživatel kliknout na tlačítko *Odemknout pro úpravu* a svoji volbu potvrdit ve vyskakovacím okně.

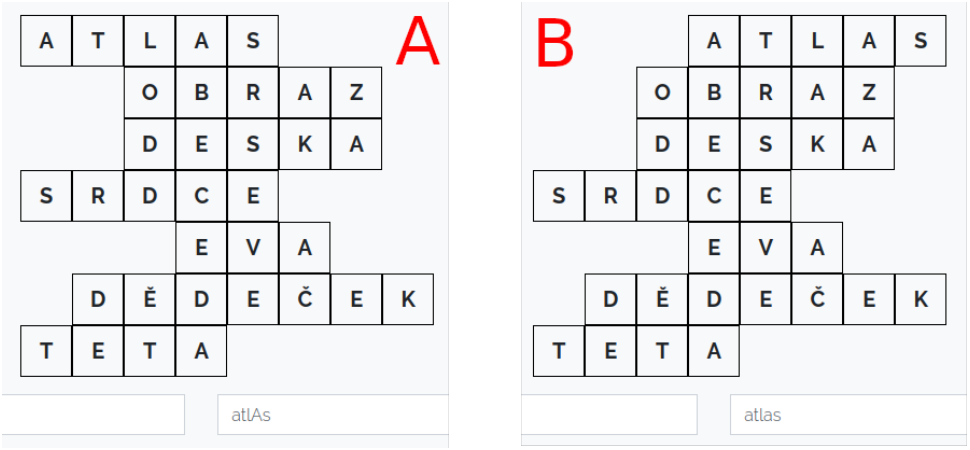
Vzhledem k tomu, že se jak v *Editoru*, tak v mobilních klientech musíme zabývat tím, jak bude křížovka vypadat vizuálně, musíme vymyslet způsob, jak ji uživateli zobrazit. Pro tyto potřeby bylo zvoleno řešení čtvercovou mřížkou *11×11*. Vizualizace křížovky se uživateli *Editoru* objeví pod políčkem pro zadání tajenky. Aktuální stav aplikace umožňuje pouze tajenku, která je uprostřed mřížky a délka všech odpovědí musí být maximálně **11** písmen.



Obrázek 4.10: Vyplněný formulář pro úpravu interaktivní úlohy Křížovka

Mimo jiné musí každá odpověď *protínat* tajenku tj. musí obsahovat alespoň jedno písmeno, které je shodné s písmenem v tajence na daném řádku a nesmí svou délkou přesáhnout hranice čtvercové mřížky. Do budoucna by bylo dobré uživateli nabídnout, aby mohl vybrat pozici (tj. číslo sloupce ve čtvercové mřížce) tajenky.

Na rozdíl od políčka odpovědi není políčko nápovědy citlivé na velká nebo malá písmena. Pokud odpověď obsahuje více než jedno stejné písmeno, které odpovídá danému písmenu tajenky, může uživatel použitím velkého písmena určit, na které pozici bude tajenka protínat daný řádek. Na obrázku 4.10 v prvním řádku křížovky můžeme vidět *caption* s odpovědí **atlas**. Když se podíváme pod vizualizaci tajenky na příslušný řádek v formuláři, vidíme že odpověď **atlas** je napsána v políčku jako `atlAs`. Když by uživatel zapsal odpověď jako `atlas`, pro protnutí tajenky by se místo písmena `A` na čtvrté pozici (varianta A) slova použilo písmeno `A` na první pozici (varianta B) slova



Obrázek 4.11: Dvě rozdílné pozice protnutí odpovědi s tajenkou

(viz obrázek 4.11).

4.2 Mobilní klienti

Dále popíšeme implementaci úloh na obou mobilních klientech a srovnáme rozdíly v jejich implementacích. Podobně jako v předchozí části o implementaci úloh do *Editoru*, nejdříve vysvětlíme jakým způsobem se data o virtuální naučné stezce dostávají do obou aplikací.

4.2.1 Načítání dat

Obdobně jako u *Editoru* ze sekce 4.1.1 i zde se pro obstarání dat používá volání na API back-endu. V **RN klientovi** se o toto stará komponenta *DownloadManager*, která představuje obrazovku *Správa stezek* zmíněnou v sekci 2.3. Ta v rámci *lifecycle* metody `componentDidMount()` volá zmíněné API, ze kterého obdrží odkaz na soubor XML popisující virtuální naučnou stezku. Uživatel pak stisknutím tlačítka *Stáhnout data* stáhne odpovídající XML soubor. Ten se pak v aplikaci překládá na JavaScriptový objekt, se kterým už je dále možné pracovat podobným způsobem jakým toho bylo u *Editoru*.

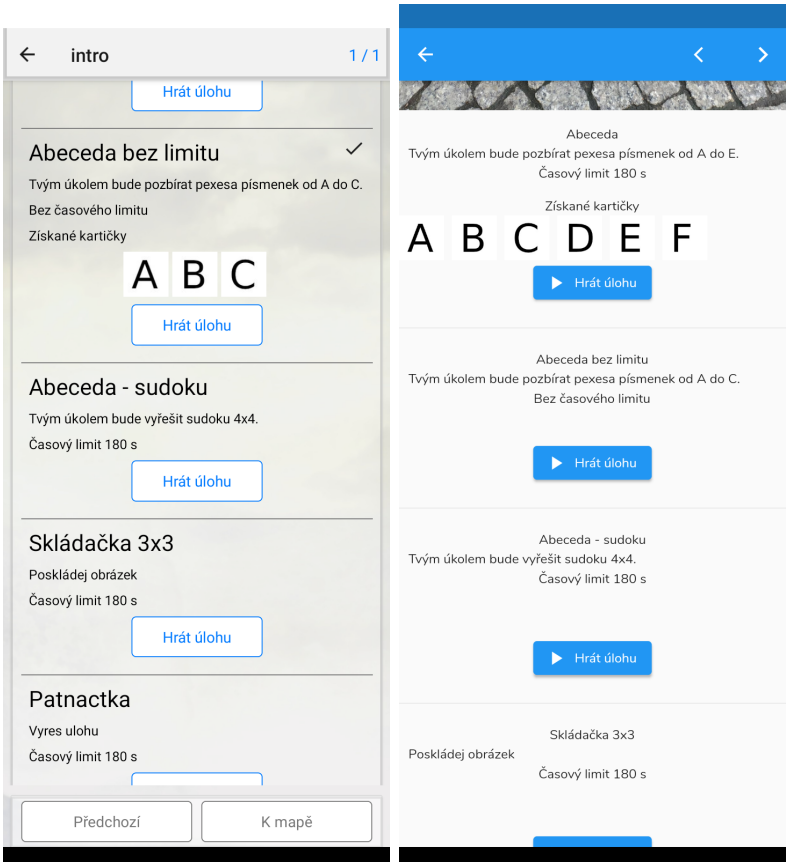
Ve **Flutter klientovi** se o načítání dat stará třída *RestClient*, která používá HTTP klienta *Dio*[5]. Stažený XML soubor se parsuje do Hive objektů, které jsou pak ukládány do lokální databáze mobilního zařízení (viz sekce 2.4).

Data o splněných úlohách se ukládají lokálně do mobilního zařízení a to

[5] `https://pub.dev/packages/dio` [cit. 2022-05-18]

způsobem, který jsme popsali výše v rámci analýzy aktuálního stavu aplikací systému *EduARd* v kapitole 2.

Implementace jednotlivých interaktivních úloh byla inspirována rozhraním otázek, proto jsou způsob a některé atributy předávané komponentám resp. *widgetům* shodné.



(a) : v RN klientovi **(b) :** ve Flutter klientovi

Obrázek 4.12: Obrazovka úkolu se splněnou úlohou Pexesa

4.2.2 Zobrazení obrazovky úkolu s tvořitelným obsahem

Jak již bylo zmíněno výše v sekci 2.3 resp. 2.4, obrazovka zobrazující úkol je posuvná, což není žádoucí pro plnění interaktivní úlohy. Jako možné řešení jsme zvolili implementaci malých komponent interaktivních úloh, které zobrazují pouze *název úlohy, popis úlohy, časový limit* a *informaci o splnění úlohy* (fajfku) společně s *odměnou za splnění* - vítězný obrázek nebo kartičky pexesa. Dále obsahují tlačítko *Hrát úlohu*, kterým lze danou úlohu spustit. Stisknutím tlačítka se uživatel přesměruje na novou obrazovku, na které se nachází samotná interaktivní úloha. Srovnání obou implementací nabízí obrázek 4.12.

4.2.3 Konstrukce úlohy

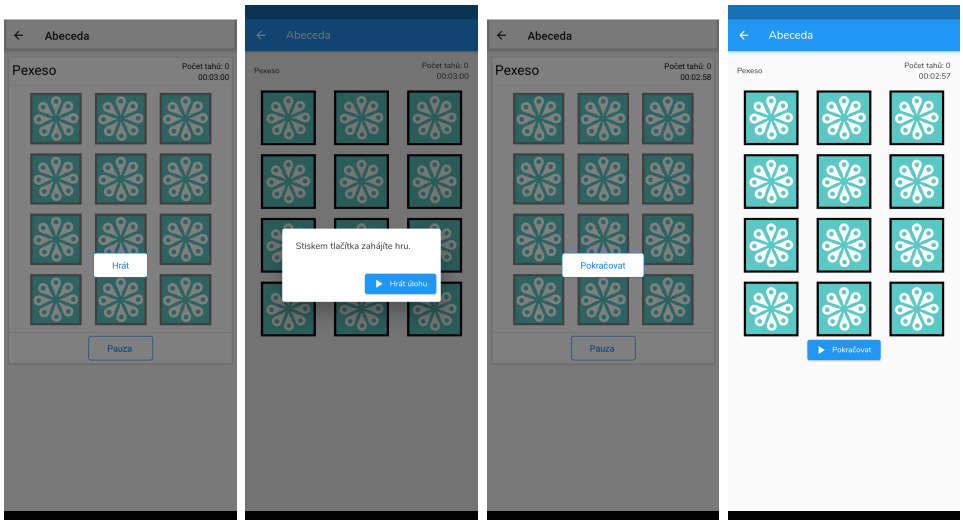
Každá z interaktivních úloh sdílí některé atributy a metody, které zajišťují její správnou funkčnost a zobrazování. Sdílenými atributy a metodami jsou:

- `state.gameState` obsahující jednu z hodnot z pole `[Loaded, Started, Paused, Won, Lost]`
- `state.audio` obsahující audio nahrávku, která se má přehrát po úspěšném splnění úlohy
- `state.timerIsPaused` obsahující informaci o stavu časovače *Timer*
- `state.turnCount` obsahující informaci o počtu tahů, které uživatel doposud vykonal
- metoda `playRewardAudio()`, která spustí audio nahrávku v případě splnění úlohy
- metoda `timeRanOutCallback()`, kterou předáváme časovači *Timer* a zavolá se v případě, že uživateli vypršel čas potřebný pro vyřešení úlohy
- metoda `notifyWin()` v *RN klientovi*, kterou dostaneme na vstupu jako parametr *props* z malé komponenty, která se nachází na obrazovce úkolu. Slouží k nastavení parametru `completed`, který značí splnění úlohy a na základě kterého se zobrazuje odměna za splnění a fajfka

Data, která předáváme do interaktivní úlohy jsou:

- `rewardAudioUri` je cesta k vítězné audio nahrávce
- příznak `playRewardAudio`, který nám říká, zda se má přehrát audio nahrávka po splnění dané úlohy
- `description` je popis úlohy
- `title` je název úlohy
- objekt `gameData` v *RN klientovi* resp. `interactiveTaskData` ve *Flutter klientovi*, který obsahuje data konkrétní úlohy (bude popsaný detailněji pro každou úlohu později)
- V *RN klientovi* předáváme `notifyWin`, což je metoda, která se zavolá při výhře uživatele, zmíněno výše
- Ve *Flutter klientovi* úloze předáváme Hive objekt modelu úlohy `taskModel`, který upravujeme v rámci obrazovky úlohy použitím metody `_setDone()` místo použití metody `notifyWin()`

Co se týče časovače *Timer*, v *RN klientovi* pro něj vznikla samostatná komponenta, která na základě stavu rodičovské komponenty (konkrétně parametru `state.timerIsPaused`) odpočítává čas, který je součástí stavu *Timeru*. Ve *Flutter klientovi* se používá třída `Timer`, která je součástí jazyka *Dart*. Tato třída umožňuje definovat na časovači funkci, která se má volat v určených periodách. Ošetřený je zde i případ pro vypršení časového limitu.



(a) : Zahájení v React Native

(b) : Zahájení úlohy Flutter

(c) : Pauza v React Native

(d) : Pauza ve Flutteru

Obrázek 4.13: Zahájení a pauza v interaktivní úloze

4.2.4 Zahájení úlohy

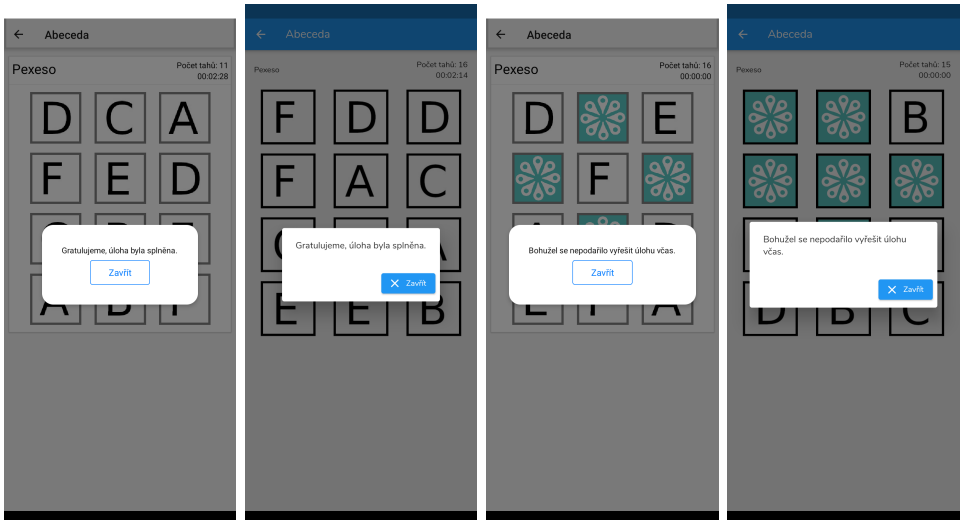
Interaktivní úlohu uživatel zahájí stiskem tlačítka *Hrát úlohu* na obrazovce úkolu. Po jeho stisknutí se otevře modální okno s tlačítkem *Hrát* v *RN klientovi* resp. *Hrát úlohu* ve *Flutter klientovi* viz obrázek 4.13. Stisknutím tlačítka se modální okno zavře a spustí se časovač úlohy.

Obdobným způsobem funguje tlačítko *Pauza*, které zastaví časovač a zároveň znemožní uživateli interagovat s úlohou.

4.2.5 Ukončení úlohy

Interaktivní úloha na mobilním klientovi může skončit třemi způsoby: *výhrou uživatele, prohrou uživatele* nebo *opuštěním obrazovky* případně *aplikace*. Jak lze vidět na obrázku 4.14 V obou případech, kde hra skončí výhrou nebo prohrou uživatele, se otevře modální okno s tlačítkem *Zavřít*. Implementace se liší tím, že v *RN klientovi* se modální okno vykresluje v závislosti na stavu komponenty, zatímco ve *Flutter klientovi* používá události. Abychom zamezili

tomu, že se modální okno ve Flutteru otevře dříve než je příslušný widget načtený musíme pro vyvolání události použít metodu `addPostFrameCallback()` na `SchedulerBindingu`. Ta nám umožní otevřít modální okno až potom, co je widget načtený[6].



(a) : Výhra v React Native **(b) :** Výhra ve Flutteru **(c) :** Prohra v React Native **(d) :** Prohra ve Flutteru

Obrázek 4.14: Zahájení a pauza v interaktivní úloze

Na obou klientech se v případě že uživatel úlohu splní v časovém limitu, přehraje vítězná audio nahrávka současně se zobrazením modálního okna. Po zavření modálního okna se uživatel může vrátit na předchozí stránku stisknutím šipky doleva, která se nachází v záhlaví obrazovky. Tuto akci může udělat také kdykoliv během plnění interaktivní úlohy.

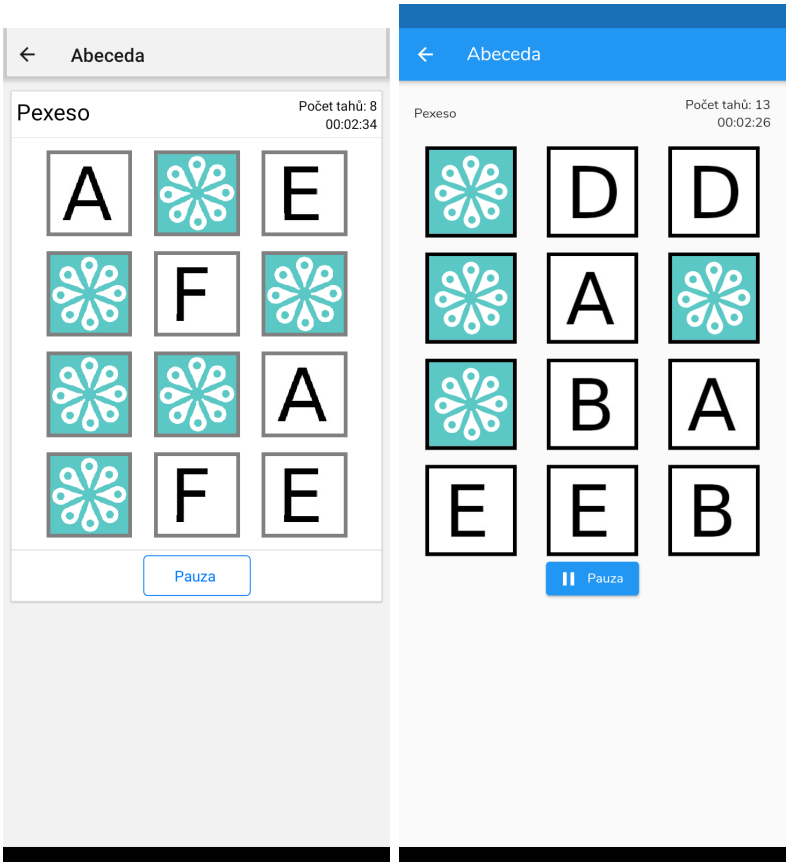
4.2.6 Interaktivní úloha Pexeso

První implementovanou úlohou do obou klientů bylo **Pexeso**. Jak bylo již zmíněno v sekci 4.2.3, každá úloha dostává parametr `gameData` resp. `interactiveTaskData`, která obsahuje data specifická pro tuto úlohu. V případě *Pexesa* je to pole `cards`, které obsahuje obrázkové kartičky. Než je úloha spuštěna, každá kartička dostane svoje `id` a je zduplikována. Potom je celé pole `cards` zamícháno a rozprostřeno do mřížky 3×N, kde N je číslo od 1 do 6 podle počtu kartiček. Maximální počet kartiček pro úlohu *Pexesa* je **18**. Vzhled celé úlohy můžeme vidět na obrázku 4.15. Zde se logika úlohy v obou klientech příliš neliší.

Hra probíhá takto:

[6] `https://api.flutter.dev/flutter/scheduler/SchedulerBinding/addPostFrameCallback.html` [cit. 2022-05-19]

1. Uživatel klikne na jednu z kartiček
2. Systém pak vybranou kartičku otočí lícem nahoru
3. Uživatel klikne na další kartičku
4. Systém vyhodnotí, zda se jedná o shodné nebo rozdílné kartičky a na základě toho krátkodobě obarví jejich okraje zeleně v případě shodných obrázků a červeně v případě rozdílných obrázků
5. Následně se vyhodnotí, zda byly otočeny všechny kartičky a pokud ano, hra končí výhrou uživatele



(a) : Pexeso v React Native **(b) :** Pexeso ve Flutteru

Obrázek 4.15: Interaktivní úloha Pexeso v obou mobilních klientech

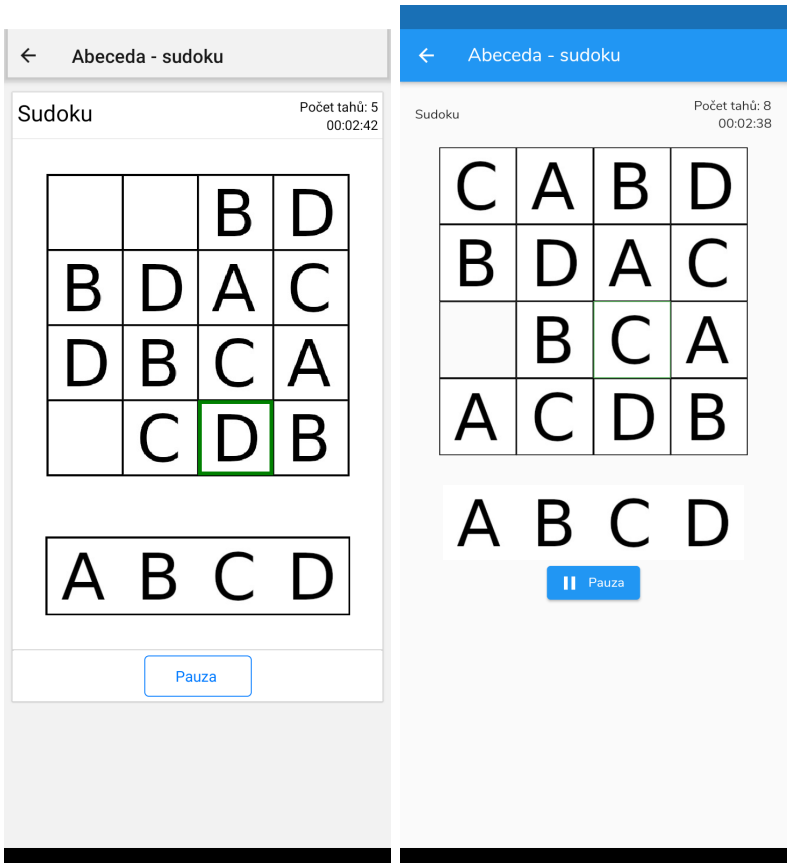
4.2.7 Interaktivní úloha Obrázkové sudoku

Další implementovanou úlohou do obou klientů bylo **Obrázkové sudoku**. Oba klienti nejdříve na základě atributu `difficulty` náhodně zakryjí některá políčka herního pole. V případě, že je zvolena lehká obtížnost, klient zakryje **8** políček z 16, pro těžkou obtížnost zakryje **10** políček z 16.

Pro parametr `gameData` resp. `interactiveTaskData` používáme:

- `rewardImageUri` resp. `imageFileName` je cesta k vítěznému obrázku
- pole `pictures` obsahující informace o symbolech nebo obrázcích, které má uživatel k dispozici k doplňování do herního pole
- dvojrozměrné pole `solution` obsahující řešení úlohy
- dvojrozměrné pole `puzzle` obsahující zadání úlohy, které vznikne zakrytím některých polí pole `solution` na základě nastavené obtížnosti úlohy

Celkový vzhled této úlohy na obou mobilních klientech můžeme vidět na obrázku 4.16.



(a) : Obrázkové sudoku v RN **(b) :** Obrázkové sudoku ve Flutteru

Obrázek 4.16: Interaktivní úloha Obrázkové sudoku v obou mobilních klientech

Hra probíhá takto:

1. Uživatel vybere políčko, do kterého chce doplnit symbol nebo obrázek

2. Systém vyznačí vybrané políčko
3. Uživatel klikne na symbol nebo obrázek, které chce do daného políčka přiřadit
4. Systém vyhodnotí, zda se jedná o správný symbol nebo obrázek a podle toho obarví jeho okraj zeleně v případě správného přiřazení a červeně v případě nesprávného přiřazení
5. Následně se vyhodnotí, zda byly všechny symboly a obrázky správně přiřazeny a pokud ano, hra končí výhrou uživatele

4.2.8 Interaktivní úloha Puzzle

Puzzle je první interaktivní úlohou, kde se implementace logiky v obou mobilních klientech liší. Implementačním rozdíl je způsob zobrazování dílků *Puzzle*. Zatímco v *Reactu* a tudíž i v *React Native* lze použít trik pro zobrazování určité části obrázku, ve *Flutteru* tuto možnost nemáme. Trik spočívá v tom, že dílek obrázku vnoříme do kontejnerového elementu, který má omezenou výšku a šířku. Pak pomocí stylů nastavíme negativní *margin* vnořenému obrázku podle toho, kam ho chceme posunout. Rozměry vnořeného obrázku jsou omezeny na rozměry kontejneru, který je zobrazený na obrázku 4.17a.

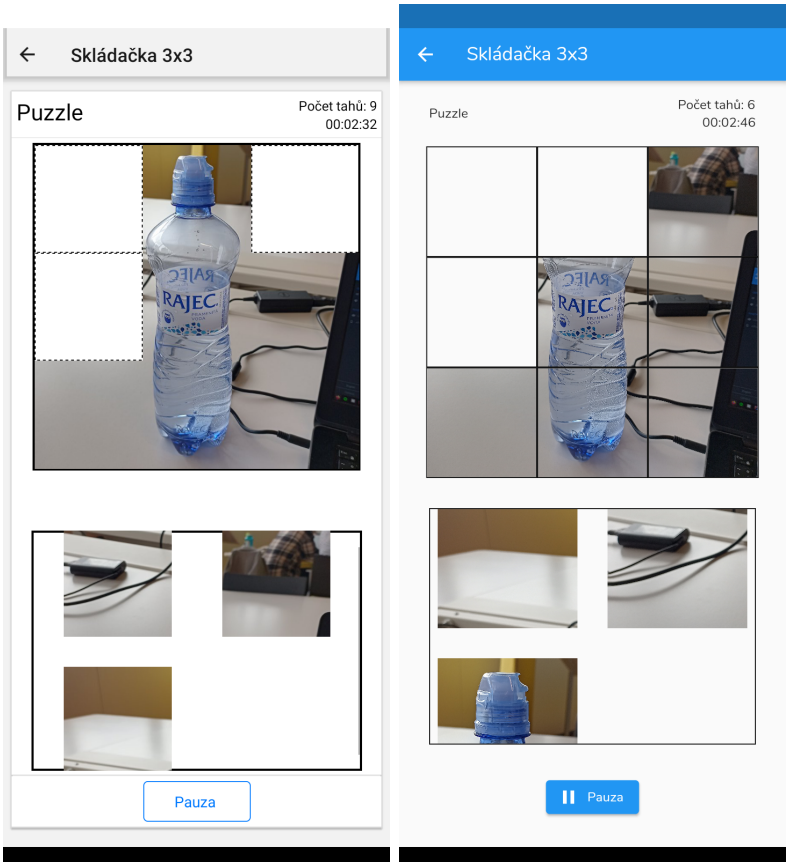
Trik je popsán následujícím pseudokódem:

```
<View style={{ overflow: 'hidden',
               width: imageCoords.partSize,
               height: imageCoords.partSize }}>
 <Image style={{ width: maxImageSize,
                 height: maxImageSize,
                 marginTop: -imageCoords.coords[id].y,
                 marginLeft: -imageCoords.coords[id].x }}/>
</View>
```

Ve *Flutteru* tento trik bohužel nefunguje. Jednou z možností je obrázek rozsekat a rozdělit do jednotlivých souborů. Tato možnost ale vytváří velké množství souborů na mobilním zařízení uživatele a toho jsme se chtěli vyvarovat. Další možností je rozsekat obrázek symbolicky na datové celky, které pak budeme moci zobrazovat v rámci našich widgetů. K tomuto účelu využijeme knihovnu **image** od Brendana Duncana[7] do jazyka *Dart*, která umožňuje datovou manipulaci s obrázky.

K tomu abychom obrázek rozsekali nejdříve musíme převést *Flutter* obrázek na obrázek z knihovny *image*. Potom pomocí funkce `copyCrop()` z knihovny

[7] `https://github.com/brendan-duncan/image` [cit. 2022-05-19]



(a) : Puzzle v React Native **(b) :** Puzzle ve Flutteru

Obrázek 4.17: Interaktivní úloha Puzzle v obou mobilních klientech

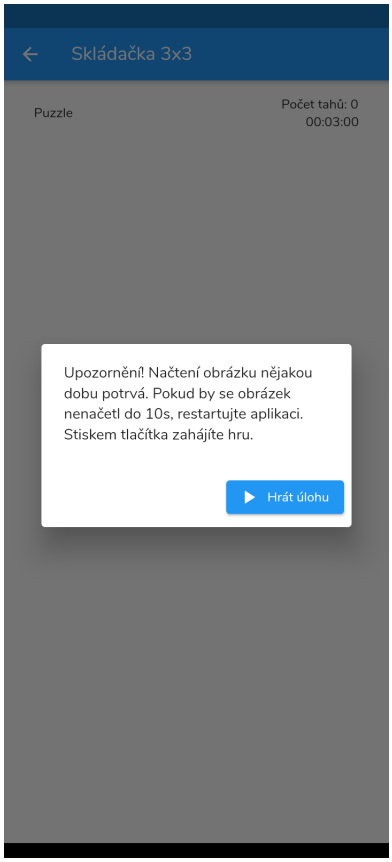
image obrázek rozsekáme podle určených rozměrů. Abychom rozsekané dílky obrázku mohli použít ve *Flutteru*, musíme je převést na widgety.

Problém s tímto řešení je takový, že konverze obrázků na widgety je výpočetně náročná a zablokuje vlákno, které vykresluje uživatelské rozhraní. Z časových důvodů nebylo možné tento problém vyřešit. Vzhledem k tomu, že tento problém výrazně snižuje uživatelský zážitek, by bylo dobré, kdyby se ho do budoucna podařilo vyřešit. Jako dočasné řešení byl do úvodního modálního okna přidán text upozorňující uživatele o této nepříjemnosti (viz obrázek 4.18).

V parametru `gameData` resp. `interactiveTaskData` úloze předáváme:

- `imageUri` resp. `imageFileName` je cesta k obrázku, který má uživatel za úkol poskládat
- `size` označuje, na kolik dílků se má daný obrázek rozsekat

V rámci úlohy pak uživatel přiřazuje dílky puzzle z posuvné nabídky v



Obrázek 4.18: Modální okno informující uživatele o zablokování UI vlákna ve Flutter klientovi pro úlohy Puzzle a Loydova patnáctka

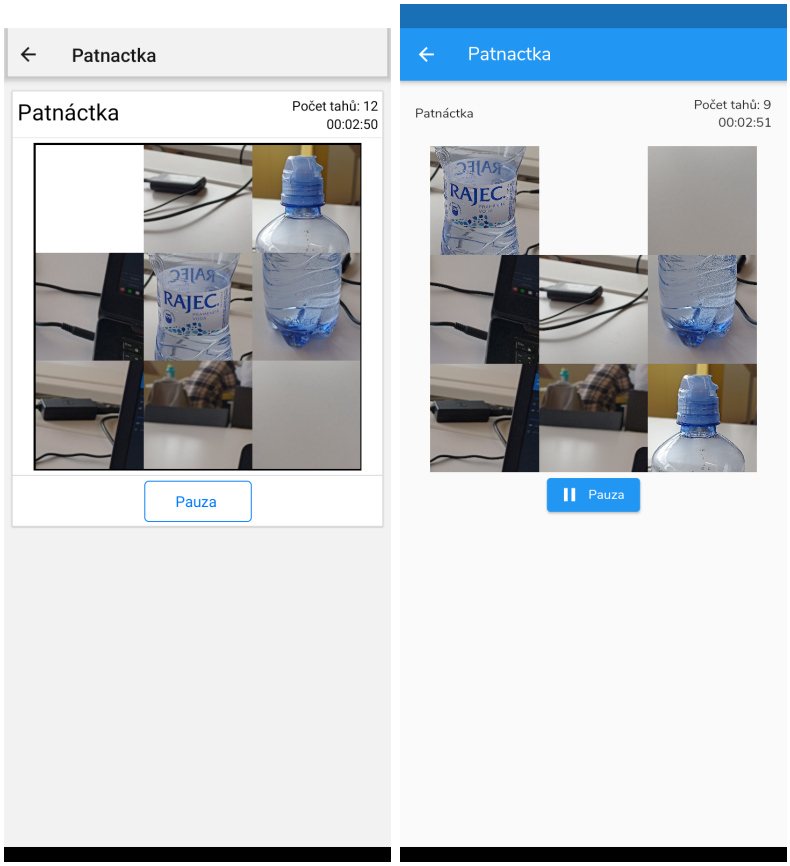
dolní části obrazovky. Při každém přiřazení se kontroluje, zda jsou všechny dílky na správném místě. V případě že ano, uživatel vyhrává.

4.2.9 Interaktivní úloha Loydova patnáctka

U **Loydovy patnáctky** je opět nutné rozsekat obrázek na několik menších dílků. Způsob, jakým to děláme je sdílený mezi *Puzzle* a touto úlohou a je popsaný v předchozí sekci. To, co se liší jsou data, které úloze předáváme, herní logika a vzhled obrazovky úlohy (viz obrázek 4.19).

V parametru `gameData` resp. `interactiveTaskData` úloze předáváme:

- `imageUri` resp. `imageFileName` a `size` obdobně jako u předchozí úlohy představují skládaný obrázek a počet dílků, na kolik se má obrázek rozsekat
- pole `puzzle`, které představuje původní uspořádání úlohy Loydovy patnáctky



(a) : Loydova patnáctka v React Native

(b) : Loydova patnáctka ve Flutteru

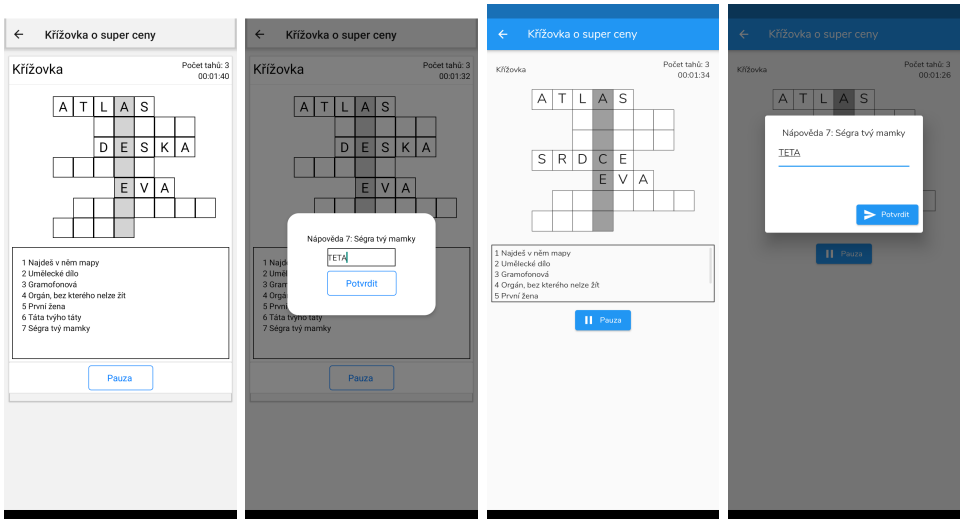
Obrázek 4.19: Interaktivní úloha Loydova patnáctka v obou mobilních klientech

Úloha pak probíhá tímto způsobem: uživatel klikne na dílek skládačky, který je vedle prázdného políčka. Vybraný dílek se do tohoto prázdného políčka posune. Při každém posunu se kontroluje, zda se uživateli povedlo úspěšně poskládat obrázek. Prázdné políčko by vždy mělo skončit v levém horním rohu. Pokud se uživateli povedlo poskládat daný obrázek, vyhrává hru a prázdné políčko se vyplní zbylým dílkem skládačky.

4.2.10 Interaktivní úloha Křížovka

Poslední implementovanou úlohou byla **Křížovka**. Svým vzhledem vychází ze své implementace v *Editoru*, kde ji tvoří čtvercová mřížka. Samotná implementace *Křížovky* se na obou mobilních klientech příliš neliší. To lze vypozorovat i z obrázku 4.20.

Jako parametr `gameData` resp. `interactiveTaskData` úloze předáváme:



(a) : Křížovka v React Native

(b) : Doplnění řádku v RN

(c) : Křížovka ve Flutteru

(d) : Doplnění řádku Flutter

Obrázek 4.20: Interaktivní úloha Křížovka v obou mobilních klientech

- `rewardImageUri` resp. `imageFileName` je cesta k vítěznému obrázku
- `solution` je textový řetězec obsahující tajenku křížovky
- pole `captions` je kolekce nápověd a odpovědí společně s pozicí indexu, na které začíná daný řádek tajenky

Úloha probíhá takto:

1. Uživatel vybere řádek tajenky, který chce vyplnit
2. Systém otevře modální okno s nápovědou a textovým polem pro vyplnění odpovědi
3. Uživatel vyplní odpověď a stiskne tlačítko *Potvrdit*
4. Systém vyhodnotí, zda má zadaná odpověď stejný počet písmen jako vybraný řádek tajenky a pokud ano, pak nastaví řádek tajenky na zadanou odpověď
5. Po každé zadané odpovědi systém kontroluje, zda je celá křížovka správně. Pokud ano, uživatel vyhrává. Pro splnění úlohy je nutné nejen najít tajenku, ale také správně vyplnit všechny řádky křížovky

Kapitola 5

Testování funkčních požadavků

V této kapitole představíme testování funkčních požadavků a vyhodnotíme, jestli se nám je podařilo splnit. Při popisu funkčních požadavků se budeme odkazovat na seznam, který jsme definovali v sekci 3.5.1.

5.1 Požadavky na Editor

Následující tabulky prezentují testy jednotlivých funkčních požadavků na *Editor* a jejich výsledky.

Název požadavku	**FR01** - Audio při splnění úlohy
Předpoklad pro spuštění	Vytvořená učebnice v *Editoru*
Očekávané chování	Uživatel je schopen nahrát, změnit a odebrat audio nahrávku pomocí nové komponenty *AudioBlock*
Výsledek testu	úspěch

Název požadavku	**FR02** - Obrázek při splnění úlohy
Předpoklad pro spuštění	Vytvořená interaktivní úloha typu *Obrázkové sudoku* nebo *Křížovka* v *Editoru*
Očekávané chování	Uživatel je schopen nahrát, změnit a odebrat obrázek pro danou interaktivní úlohu pomocí komponenty *ImageBlock*
Výsledek testu	úspěch

Název požadavku	**FR04** - Nastavení časového limitu pro splnění úlohy
Předpoklad pro spuštění	Vytvořená interaktivní úloha v *Editoru*
Očekávané chování	Uživatel je schopen na úlohu aplikovat časový limit a specifikovat jeho výši v sekundách
Výsledek testu	úspěch

Název požadavku	**FR09** - Pexeso
Předpoklad pro spuštění	Vytvořený úkol, součástí kterého je alespoň jedna obrazovka v *Editoru*
Očekávané chování	Uživatel je schopen pomocí zeleného FAB vytvořit interaktivní úlohu *Pexeso*, tu následně pomocí formuláře upravovat a smazat pomocí tlačítka *smazat*
Výsledek testu	úspěch

Název požadavku	**FR10** - Pexeso, správa kartiček
Předpoklad pro spuštění	Vytvořená interaktivní úloha typu *Pexeso*
Očekávané chování	Uživatel je schopen vytvářet, upravovat a mazat kartičky k interaktivní úloze *Pexeso* a zároveň přidat, změnit a odebrat vlastní obrázek ke každé z kartiček
Výsledek testu	úspěch

Název požadavku	**FR12** - Sudoku
Předpoklad pro spuštění	Vytvořený úkol, součástí kterého je alespoň jedna obrazovka v *Editoru*
Očekávané chování	Uživatel je schopen pomocí zeleného FAB vytvořit interaktivní úlohu *Obrázkové sudoku*, tu následně pomocí formuláře upravovat a smazat pomocí tlačítka *smazat*
Výsledek testu	úspěch

Název požadavku	**FR13** - Sudoku, generátor řešení
Předpoklad pro spuštění	Vytvořená interaktivní úloha typu *Obrázkové sudoku*
Očekávané chování	Uživatel je schopen po kliknutí na tlačítko *Vygeneruj řešení* vygenerovat validní řešení pro úlohu *Obrázkové sudoku*
Výsledek testu	úspěch

Název požadavku	**FR15** - Puzzle
Předpoklad pro spuštění	Vytvořený úkol, součástí kterého je alespoň jedna obrazovka v *Editoru*
Očekávané chování	Uživatel je schopen pomocí zeleného FAB vytvořit interaktivní úlohu *Puzzle*, tu následně pomocí formuláře upravovat a smazat pomocí tlačítka *smazat*
Výsledek testu	úspěch

Název požadavku	**FR17** - Loydova patnáctka
Předpoklad pro spuštění	Vytvořený úkol, součástí kterého je alespoň jedna obrazovka v *Editoru*
Očekávané chování	Uživatel je schopen pomocí zeleného FAB vytvořit interaktivního úlohu *Loydova patnáctka*, tu následně pomocí formuláře upravovat a smazat pomocí tlačítka *smazat*
Výsledek testu	úspěch

Název požadavku	**FR18** - Loydova patnáctka, generátor řešení
Předpoklad pro spuštění	Vytvořená interaktivní úloha typu *Loydova patnáctka*
Očekávané chování	Uživatel je schopen po kliknutí na tlačítko *Vygeneruj zadání* vygenerovat validní zadání pro úlohu *Loydova patnáctka*
Výsledek testu	úspěch

Název požadavku	**FR20** - Křížovka
Předpoklad pro spuštění	Vytvořený úkol, součástí kterého je alespoň jedna obrazovka v *Editoru*
Očekávané chování	Uživatel je schopen pomocí zeleného FAB vytvořit interaktivní úlohu *Křížovka*, tu následně pomocí formuláře upravovat a smazat pomocí tlačítka *smazat*
Výsledek testu	úspěch

5.2 Požadavky na mobilní klienty

Následující tabulky prezentují testy jednotlivých funkčních požadavků na *RN klient* a *Flutter klient* a jejich výsledky.

Název požadavku	**FR03** - Odměna uživateli za splnění úlohy
Předpoklad pro spuštění	Spuštěná jakákoliv interaktivní úloha ve stavu `Started`
Očekávané chování	Uživateli se po splnění úlohy přehraje audio nahrávka a zobrazí vítězný obrázek
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Název požadavku	**FR05** - Měření času při plnění interaktivní úlohy
Předpoklad pro spuštění	Spuštěná jakákoliv interaktivní úloha ve stavu `Loaded`
Očekávané chování	Uživatel je schopen stisknutím tlačítka *Hrát* v RN klientovi, příp. *Hrát úlohu* ve Flutter klientovi zahájit odpočítávání časového limitu a je schopen jej zastavit po stisknutí tlačítka *Pauza*
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Název požadavku	**FR06** - Splnění interaktivní úlohy
Předpoklad pro spuštění	Spuštěná jakákoliv interaktivní úloha ve stavu `Started`
Očekávané chování	Uživatel je po splnění úlohy notifikován o splnění úlohy vyskakovacím modálním oknem a po návratu na obrazovku úkolu se zobrazí u úlohy fajfka a odměna za splnění úlohy
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch, fajfka ale chybí

Název požadavku	**FR07** - Vypršení časového limitu
Předpoklad pro spuštění	Spuštěná jakákoliv interaktivní úloha ve stavu `Started`
Očekávané chování	Uživatel je po skončení časového limitu notifikován vyskakovacím modálním oknem a je mu znemožněna interakce s úlohou
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Název požadavku	**FR08** - Možnost opakování již splněných úloh
Předpoklad pro spuštění	Obrazovka úkolu obsahující alespoň jednu interaktivní úlohu, která již byla jednou splněna
Očekávané chování	Uživatel je schopen kliknutím na tlačítko *Hrát úlohu* spustit danou úlohu a znovu ji splnit
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Název požadavku	**FR11** - Spuštění a dokončení úlohy Pexeso
Předpoklad pro spuštění	Obrazovka úkolu obsahující alespoň jednu interaktivní úlohu typu *Pexeso*
Očekávané chování	Uživatel je schopný stisknutím tlačítka *Hrát úlohu* zahájit interaktivní úlohu typu *Pexeso* a úspěšně ji dokončit splněním vítězné podmínky definované v sekci 3.2
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Název požadavku	**FR14** - Spuštění a dokončení úlohy Sudoku
Předpoklad pro spuštění	Obrazovka úkolu obsahující alespoň jednu interaktivní úlohu typu *Obrázkové sudoku*
Očekávané chování	Uživatel je schopný stisknutím tlačítka *Hrát úlohu* zahájit interaktivní úlohu typu *Obrázkové sudoku* a úspěšně ji dokončit splněním vítězné podmínky definované v sekci 3.2
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Název požadavku	**FR16** - Spuštění a dokončení úlohy Puzzle
Předpoklad pro spuštění	Obrazovka úkolu obsahující alespoň jednu interaktivní úlohu typu *Puzzle*
Očekávané chování	Uživatel je schopný stisknutím tlačítka *Hrát úlohu* zahájit interaktivní úlohu typu *Puzzle* a úspěšně ji dokončit splněním vítězné podmínky definované v sekci 3.2
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch s výhradami - spuštění úlohy na chvíli zablokuje UI vlákno

Název požadavku	**FR19** - Spuštění a dokončení úlohy Loydova patnáctka
Předpoklad pro spuštění	Obrazovka úkolu obsahující alespoň jednu interaktivní úlohu typu *Loydova patnáctka*
Očekávané chování	Uživatel je schopný stisknutím tlačítka *Hrát úlohu* zahájit interaktivní úlohu typu *Loydova patnáctka* a úspěšně ji dokončit splněním vítězné podmínky definované v sekci 3.2
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch s výhradami - spuštění úlohy na chvíli zablokuje UI vlákno

Název požadavku	**FR21** - Spuštění a dokončení úlohy Křížovka
Předpoklad pro spuštění	Obrazovka úkolu obsahující alespoň jednu interaktivní úlohu typu *Křížovka*
Očekávané chování	Uživatel je schopný stisknutím tlačítka *Hrát úlohu* zahájit interaktivní úlohu typu *Křížovka* a úspěšně ji dokončit splněním vítězné podmínky definované v sekci 3.2
Výsledek testu	*RN klient*: úspěch *Flutter klient*: úspěch

Kapitola 6

Závěr

Cílem této bakalářské práce bylo rozšířit projekt EduARd o rozhraní na tvorbu a spouštění interaktivních úloh. Toto rozšíření se týkalo celkem tří aplikací systému EduARd: *Editoru*, mobilního klienta v jazyce *React Native* a mobilního klienta v jazyce Dart s použitím frameworku *Flutter*. Celkem mělo být navrženo a implementováno minimálně pět různých interaktivních úloh. Nejdříve proběhla analýza aktuálního stavu všech tří dotčených aplikací. Následně bylo navrženo rozhraní a datová reprezentace interaktivní úlohy s ohledem na metodiku *user-centered designu*. Bylo představeno celkem pět interaktivních úloh společně s jejich logickým a grafickým návrhem. Na základě návrhu byly stanoveny funkční a kvalitativní požadavky na rozšíření aplikací. Poté proběhla implementace tohoto rozšíření, která se co nejvíce snažila zachovat původní vzhled včetně architektury jednotlivých aplikací. Na závěr bylo otestováno, zda byly splněny všechny funkční požadavky.

Ačkoliv se podařilo hlavní cíle práce naplnit, je zde další prostor pro zlepšení. Bylo by potřeba aktualizovat knihovny, na kterých aplikace běží, zjednodušit zdrojový kód a případně sjednotit společné rozhraní pro oba mobilní klienty. Možnými dalšími rozšířeními této práce by mohlo být např. zjednodušení a unifikace rozhraní pro každou úlohu (definice vstupních dat, funkce popisující jeden tah v úloze, podmínka splnění úlohy, atp.), přidání validací na políčka formuláře pro tvorbu interaktivních úloh, vytvoření dalších typů úloh včetně komplexnější úloh, které by vyžadovaly uživatelovu zručnost a chytrý systém, který by úlohu vyhodnotil (např. překreslování zadaného obrázku, kde by pak systém s umělou inteligencí vyhodnocoval, jak moc se liší od původního obrázku) nebo interaktivní kvízy pro venkovní vycházku učitele s žáky, která by hodnotila rychlost a správnost zadaných odpovědí mezi všemi žáky najednou.

Příloha A

Literatura

[1] Expo Documentation. https://docs.expo.dev/. [cit. 2022-05-12].

[2] Flutter Documentation. https://docs.flutter.dev/resources/faq. [cit. 2022-05-13].

[3] React-bootstrap documentation. https://react-bootstrap.github.io/. [cit. 2022-05-12].

[4] React documentation. https://reactjs.org/. [cit. 2022-05-11].

[5] React Native documentation: Setting up the development environment. https://reactnative.dev/docs/environment-setup. [cit. 2022-05-12].

[6] Sass: Documentation. https://sass-lang.com/documentation. [cit. 2022-05-12].

[7] User-Centered Design Basics. https://www.usability.gov/what-and-why/user-centered-design.html. [cit. 2022-05-14].

[8] Andrea Bizzotto. Flutter vs. React Native – Which is the Best Choice for Your next App? - Blog Udemy. https://blog.udemy.com/flutter-vs-react-native/. [cit. 2022-05-13].

[9] Bonnie Eisenman. Chapter 1: What Is React Native? https://www.oreilly.com/library/view/learning-react-native/9781491929049/ch01.html, 2015. [cit. 2022-05-12].

[10] Iqra Fatima. React Lifecycle Methods — React.js. https://iqrafatimame.medium.com/react-lifecycle-methods-react-js-6efe6f1d7c1f. [cit. 2022-05-18].

[11] Fabian Hinsenkamp. What are API Keys and Why are they so important? https://medium.com/codex/

what-are-api-keys-and-why-are-they-so-important-7fb4307575e, 2022. [cit. 2022-05-12].

[12] Wolfram MathWorld. 15 Puzzle. https://mathworld.wolfram.com/15Puzzle.html. [cit. 2022-05-13].

[13] Katedra počítačové grafiky a interakce. Leták EduARd. https://fel.cvut.cz/cz/vz/produkty/letak-eduard-1.pdf. [cit. 2022-05-11].

[14] Puneet Sapra. What is Parsing? https://medium.com/the-mighty-programmer/what-is-parsing-4012f997d265, 2020. [cit. 2022-05-12].

[15] Anastasia Surikova. Webový klient pro správu mobilních výukových úloh systému EduARd. České vysoké učení technické v Praze, 2019.

[16] Lukáš Šimon. Mobilní Flutter aplikace pro řešení výukových úloh v systému EduARd. České vysoké učení technické v Praze, 2021.

[17] Šimon Maňour. Virtuální naučné stezky v React Native. České vysoké učení technické v Praze, 2019.

Příloha B

Návod na spuštění přiložených souborů

Pro účely testování doporučuji použít učebnici *Testovací učebnice*, která v sobě obsahuje všechny interaktivní úlohy.

Editor

Přílohy přiložené k práci obsahují archiv `editor-src.zip`. Pro spuštění projektu nejdříve musíte nainstalovat *Node.js*. Je otestované, že projekt funguje na verzi **13.14.0**. Pro správu verzí *Node.js* lze použít balíček **n**[1] pro *Linux* a *macOS* nebo **nvm-windows**[2] pro *Windows*.

Po instalaci *Node.js* přejděte do kořenového adresáře projektu:

```
web_frontend-feat-viskuond-interactive-tasks
```

a nainstalujte všechny závislosti pomocí příkazu `yarn`.

Aplikaci lokálně spustíte pomocí příkazu `yarn start`. Ta je pak dostupná na adrese `http://localhost:3000/`. Přístupové údaje do aplikace naleznete v souboru `editor-pass.txt`.

RN klient

Přílohy přiložené k práci obsahují archiv `rn-klient.zip`, který je rozdělen na dvě části. Z nich je potřeba extrahovat soubor `rn-klient.apk`. Ten lze potom nainstalovat na mobilní telefon se systémem *Android*.

[1] `https://www.npmjs.com/package/n` [cit. 2022-05-20]
[2] `https://github.com/coreybutler/nvm-windows` [cit. 2022-05-20]

Případně lze využít zdrojové kódy projektu. Ty jsou obsaženy v archivu `rn-klient-src.zip`. Pro spuštění nejdříve musíte nainstalovat *Node.js*. Je otestované, že projekt funguje na verzi **14.19.0**. Pak je potřeba nainstalovat *Expo CLI*. Lze použít příkaz `npm install -g expo-cli`. Případně lze postupovat podle návodu, který je dostupný na `https://docs.expo.dev/workflow/expo-cli/`. Projekt je otestovaný na verzi CLI **5.4.3**.

Po instalaci *Node.js* a *Expo CLI* přejděte do kořenového adresáře projektu:

```
react_client-feat-viskuond-interactive-tasks
```

a nainstalujte všechny závislosti pomocí příkazu `npm install`. Po stažení potřebných závislostí, projekt spustíte příkazem `expo start`. Na adrese `http://localhost:19002/` se spustí vývojový server Expa, který lze použít pro debugování a ladění aplikace. Jelikož projekt používá *Expo SDK* verze **41**, která přestala být v nové verzi *Expo Go* podporována, je nutné pro spuštění projektu na mobilním telefonu pomocí *Expo Go* stáhnout nižší verzi. Pro spuštění projektu je potřeba použít *Expo Go* ve verzi **2.19.7** pro *Android*, případně verzi **2.19.6** pro *iOS*. Návod na downgrade *Expo Go* je dostupný zde: `https://dev.to/sasurau4/the-way-to-downgrade-expo-client-on-android-2mca`.

Otevřete downgradovanou aplikaci *Expo Go* a načtěte QR kód, který je na obrazovce na adrese `http://localhost:19002/`. Pokud by se aplikace nespustila, zkuste použít mód Tunnel.

Flutter klient

Přílohy přiložené k práci obsahují archiv `flutter-klient.zip`, který je rozdělen na dvě části. Z nich je potřeba extrahovat soubor `flutter-klient.apk`. Ten lze potom nainstalovat na mobilní telefon se systémem *Android*. Přístupové údaje do aplikace naleznete v souboru `flutter-pass.txt`.

Případně lze využít zdrojové kódy projektu. Ty jsou obsaženy v archivu `flutter-klient-src.zip`. Pro spuštění nejdříve musíte nainstalovat *Flutter SDK*. Je otestované, že projekt funguje na verzi **2.0.0**. Jak nainstalovat Flutter je k dispozici v oficiální dokumentaci na `https://docs.flutter.dev/get-started/install`. Pokud se vám nedaří nainstalovat Flutter ve verzi **2.0.0** lze použít tento trik: přejděte do složky, kde je nainstalováno SDK a tam spusťte příkaz `git checkout 2.0.0`[3].

Po úspěšné instalaci *Flutter SDK* přejděte do složky

```
flutter_client_2-feat-viskuond-interactive-tasks/eduard
```

[3]Podle návodu z `https://fluttercorner.com/how-to-downgrade-flutter-sdk/` [cit. 2022-05-20]

a pomocí příkazu `flutter packages pub get` stáhněte potřebné závislosti. Pak ještě musíte provést build aplikace pomocí příkazu `flutter packages pub run build_runner build` s příznakem *–delete-conflicting-outputs*, aby se správně vygenerovaly soubory potřebné pro fungování lokální databáze Hive. Pak lze lokálně aplikaci spustit připojením mobilního telefonu k počítači. Pomocí příkazu `flutter devices` zjistěte ID vašeho telefonu. Pak zavoláním příkazu `flutter run -d` *ID* spustíte aplikaci na daném zařízení. Přístupové údaje do aplikace naleznete v souboru `flutter-pass.txt`.