GYMNÁZIUM, PRAHA 6, ARABSKÁ 14

Programování

MATURITNÍ PRÁCE

2021

Timotej Adamec

GYMNÁZIUM, PRAHA 6, ARABSKÁ 14

Arabská 14, Praha 6, 160 00

MATURITNÍ PRÁCE

Předmět: Programování
Téma: Doprovodná webová aplikace pro android aplikaci na poznávačky

Autor: Timotej Adamec
Třída: 4.E
Školní rok: 2020/2021
Vedoucí práce: Mgr. Jan Lána
Třídní učitel: Mgr. Barbora Novosadová

Prohlašuji, že jsem jediným autorem tohoto projektu, všechny citace jsou řádně označené a všechna použitá literatura a další zdroje jsou v práci uvedené. Tímto dle zákona 121/2000 Sb. (tzv. Autorský zákon) ve znění pozdějších předpisů uděluji bezúplatně škole Gymnázium, Praha 6, Arabská 14 oprávnění k výkonu práva na rozmnožování díla (§ 13) a práva na sdělování díla veřejnosti (§ 18) na dobu časově neomezenou a bez omezení územního rozsahu.

V Praze dne 1. 1. 2021

Timotej Adamec

Anotace

Tato práce se zabývá vývojem responsivní webové aplikace doplňující android aplikaci na poznávačky, která je dostupná na Google Play Store pod názvem „Memimg”. Aplikace kromě stránky obsahující legální informace o ochraně soukromí zprostředkovává i testovací přostředí, jinak doposud dostupné jen pro vlastníky aplikace Memimg, tedy jen pro platformu Android.

Práce obsahuje popis principů, na kterých webová aplikace funguje, seznam použitých technologií a podrobnou dokumentaci rozebírající fungování a vzájemnou provázanost jednotlivých součástí aplikace. Text je doplňen o obrázky a ukázky kódu.

Annotation

This work deals with the development of a responsive web application accompanying the Android application for image recognition exams, which is available on the Google Play Store under the name "Memimg". The web application in addition to the page containing legal information on privacy protection, the application also provides an exam environment, otherwise so far only available for owners of the Memimg app, ie only for the Android platform.

The work contains a description of the principles on which the web application works, a list of used technologies, and detailed documentation describing the operation and interdependence of each application component. The text is supplemented with illustrations and code samples.

Zadání práce

Popis práce:

Práce by se zabývala vytvářením webu, který by sloužil jako doplněk (rozšíření produktu) pro aplikaci Memimg (původně Poznávačka). Webová aplikace by řešila problém testování žáků, kteří nevlastní produkt s operačním systémem Android. Hlavní funkcí webu by byla možnost vyplnění testu, který vytvořil jiný uživatel v androidové aplikaci. To si lze představit jako formu Kahootu. Webová aplikace by tak umožnila plnou funkčnost testování uživatelů s aplikací Memimg. Testovaný uživatel si pak test může spustit na jakémkoliv zařízení s jakýmkoliv rozlišením, jelikož web bude adaptivní vůči typu obrazovky. Flutter dále umožňuje rozšířit kódovou základnu a vytvořit androidové nebo ios aplikaci (možný budoucí vývoj). Webová aplikace by byla propojena s databází původní androidové aplikace Memimg a běžela by na web hostingu.

Platforma:

- Web
- Flutter
- Dart

Obsah

Úvod

Motivací pro vytvoření této webové aplikace bylo ucelit projekt z minulého roku a vytvořit tak kompletní a plně funkční produkt. Tento projekt řeší problém testování žáků, kteří nemají k dispozici aplikaci Memimg, tedy těch, kterých nevlastní zařízení s operačním systémem Android. Hlavní funkcí webu by byla možnost vyplnění testu, který vytvořil jiný uživatel v androidové aplikaci. Další motivací bylo porozumět a osvojit si celý proces testování, jelikož já osobně jsem na testování v původní aplikaci nepracoval. Aplikace by měla být vizuálně co možná nejpřívětivější a jednoduchá, aby každý uživatel přesně věděl, co má v daný okamžik dělat. Struktura je tudíž jednoduchá, na úvodní stránce uživatel zadá kód, který mu umožní přístup k testu. Dále uživatel zadá svoje jméno a pokračuje do testu. Test vyplní a u posledního zástupce může test ukončit. Celá aplikace by měla být plně responzivní a přizpůsobovat se typu obrazovky jako je tablet, mobilní telefon nebo monitor. Jako technologii pro realizaci této myšlenky jsem zvolil Flutter.

1 Použité technologie

1.1 Flutter

Rozhodl jsem se pro Flutter, protože se mi líbí idea Flutteru – nativně kompilované aplikace pro mobilní zařízení, web a stolní počítače z jediné kódové základny. Flutter je open-source SDK, vytvořené společností Google.

1.1.1 Dart

Flutter aplikace jsou psány v jazyce Dart. Dart je OOP (object-oriented programming) a je založen na syntaxi jazyka Java, takže pro mne nebylo složité přejít na jazyk Dart, jelikož Javu se na naší škole učíme už od prvního ročníku.

1.2 Firebase Firestore

Firestore je databáze od Firebase a Google Cloud Platform. Databáze je NoSQL, dokumentově-orientovaná databáze [1]. To znamená, že zde nejsou žádné řádky nebo sloupce. Místo toho existují tzv. dokumenty a každý dokument obsahuje kolekce. V každém dokumentu je seznam klíčů a hodnot. Cloudová řešení jsou v dnešní době populární díky nejen díky škálovací schopnosti databáze, ale i díky pohodlí spojené s řešením hardwarové stránky věci.

1.3 Git & GitHub

Git je nejčastěji používaný systém pro správu verzí. Git sleduje změny provedené v souborech, takže máte záznam o tom, co bylo provedeno, a můžete se vrátit ke konkrétním verzím, pokud to někdy budete potřebovat. Git také usnadňuje spolupráci a umožňuje sloučit změny více lidí (nebo jednoho člověka na více zařízeních) do jednoho zdroje.

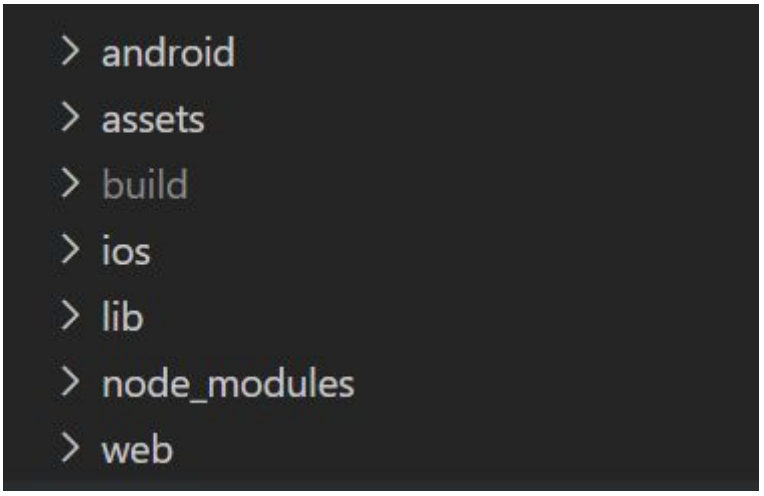
GitHub je cloudová hostingová služba, která vám umožní spravovat Git úložiště. Pokud máte projekty, které používají Git, pak je GitHub navržen tak, aby vám pomohl lépe je spravovat.

1.4 Visual Studio Code

Visual Studio Code je IDE (Integrated Development Environment), neboli vývojové prostředí. Ve vývojovém prostředí píše vývojář většinu svého kódu. Vývojová prostředí obsahují řadu užitečných funkcí urychlujících průběh vývoje.

2 Struktura programu

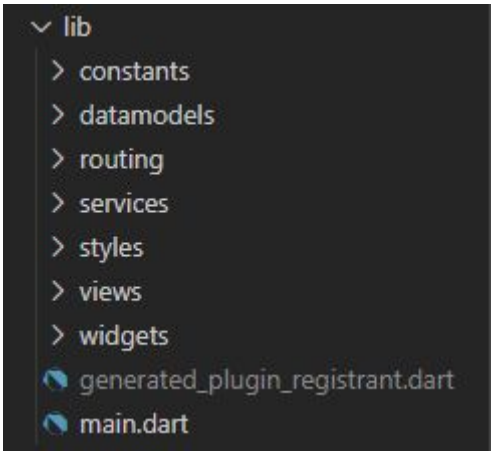
Z pohledu programátora vypadá nejvyšší level struktury programu následujícně:



Obr. 1: Složková struktura programu

Složka *android*, *ios* a *web* každá obsahuje dodatečné informace, jak má Flutter aplikaci na konkrétní platformu vytvořit. V mém případě mi stačilo upravit jediný soubor ve složce *web*, kde jsem inicializoval Firebase. Složka, kde se nachází zdrojový kód je složka *lib* a ve složce *assets* jsou zdroje jako obrázky nebo styly písma. Následující sekce budou nést názvy složek, ve kterých se nacházejí soubory plnící podobnou funkci.

2.1 *lib*



Obr. 2: Kořenová struktura aplikace

lib zahrnuje veškerý kořenový kód, přičemž kořenem aplikace je soubor main.dart, který není nijak zvlášť zajímavý (obsahuje inicializaci navigační služby, motivu atd.). Adresáře nyní můžeme rozdělit na obsahově bohaté a na obsahově chudé. Obsahově chudé by byly ty adresáře, které se už dále nevětví a čerpají z nich obsahově bohatší adresáře a můžeme je tedy nazývat „doplňující". Tyto doplňující složky jsou téměř všechny až na poslední dvě a to *views* a *widgets*. Ve zkratce plní doplňující adresáře následující funkce:

- *constants* => obsahuje projektové konstanty (jako například barvy)
- *datamodels* => pojímá datové modely (zástupce v testu nebo výsledek testu)
- *routing* => zahrnuje navigaci aplikace mezi destinacemi (i URL)
- *services* => služby tvoří prostředníka mezi dvěma částmi aplikace (v našem případě aplikace při vstupu do testu volá navigační funkci, která dál posílá příkaz do *routing*, aby přesměrovala uživatele na požadovanou destinaci)
- *styles* => obsahuje textové styly, určené pro různé typy obrazovek a různá umístění

2.1.1 *views*

Flutter funguje na bázi tzv. widgetů (stavební prvky uživatelského prostředí) [2]. Teoreticky by tedy *views* mohly být pod adresářem *widgets*, ale to už je na osobní preferenci. *views* pak není nic jiného, než obrazovky (základní widgety), které uživatel může vidět. Ty jsou v našem případě pouze 3 a to domovská obrazovka, obrazovka o soukromí a testovací obrazovka. *views* dále může ještě obsahovat rozšiřující rozložení podle typu obrazovky nebo popřípadě vnořené rozložení (více obrazovek pod navigační lištou).

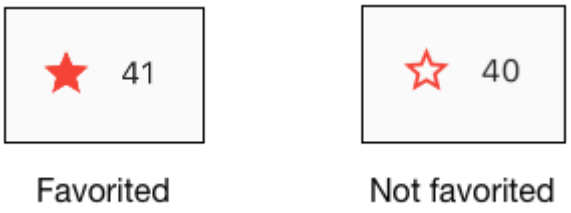
2.1.2 *widgets*

widgets už jsou konkrétní stavební bloky vnořené ve *views*, které se pak mohou dále skládat ještě z menších prvků. Hlavní rozdíl mezi *views* a *widgets* je, že každý prvek z *views* se na obrazovce nevyskytuje vícekrát, zatímco každý widget z *widgets* se může zobrazit vícekrát.

Soubory ve *widgets*, stejně jako ve *views*, jsou všechny tzv. widgety. Ve flutteru je widget způsob, jak deklarovat a vytvořit uživatelské rozhraní. Widget může něco zobrazit, může pomoci definovat design, může pomoci s rozvržením, zvládne interakci uživatele atd. Například Padding je widget, Margin je widget, Center je widget, řádky rozložení a sloupce jsou také widgety [2].

Widget je buď stateful, nebo stateless. Pokud se widget může změnit - například při interakci uživatele - je to stateful widget. Stateless widget se nikdy nemění. Stav widgetu je uložen v objektu State a odděluje stav widgetu od jeho vzhledu [2].

Příklad stateful widgetu.

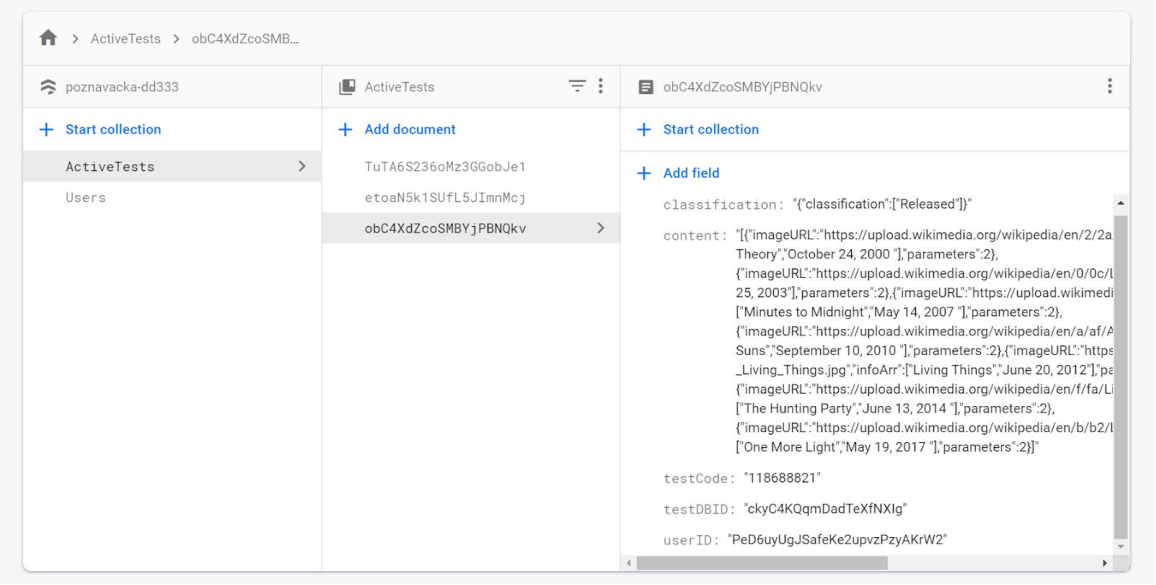


Obr. 3: Příklad stateful widgetu

Tento widget má dva stavy - Favorited a Not favorited. Musí reagovat na uživatelovu interakci s dotykem na hvězdičku a následně musí překreslit uživatelské rozhraní.

3 Struktura databáze

Kořen databáze obsahuje dvě kolekce - `ActiveTests` a `Users`. `ActiveTests` obsahuje pouze aktivní testy, tedy ty testy, které jsou přístupné uživatelům s přístupovým kódem.



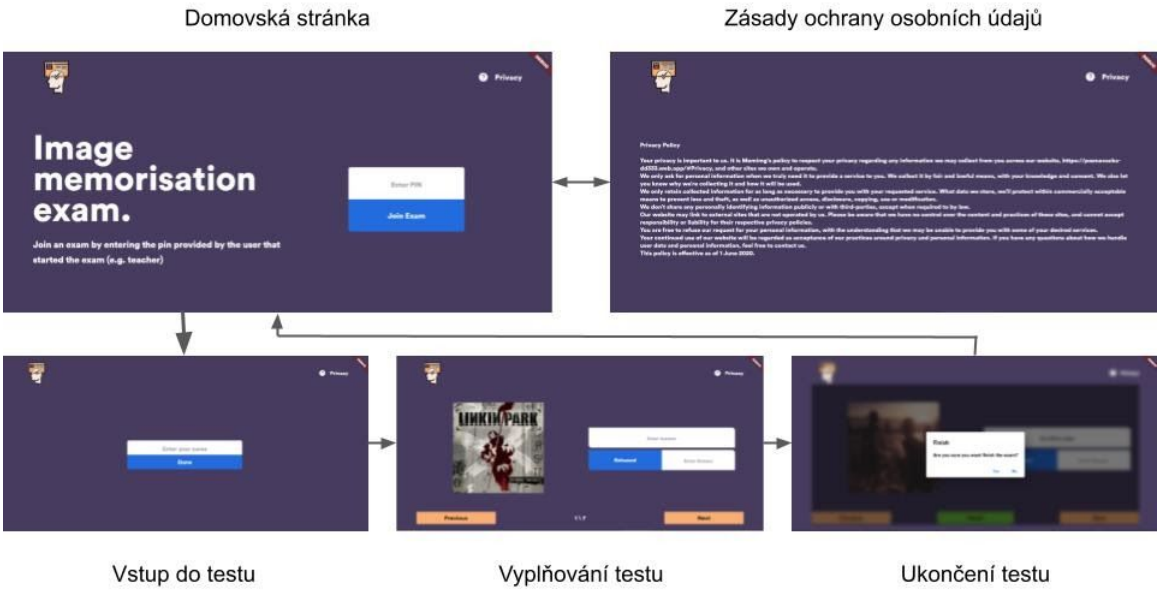
Obr. 4: Struktura databáze

Každý dokument pak obsahuje informace o testu, které potřebuje program pro běh a odeslání výsledků.

V kolekci `Users` jsou pak uživatelé, kteří v původní androidové aplikaci Memimg vytvořili poznávačky, které sdíleli nebo ty, které zpřístupnili jako testy. V podkolekci `Exams` pak najdeme další podkolekci `Results`, kam se ukládají výsledky testovaných, které si testující může zobrazit v androidové aplikaci.

4 Fungování programu

Nyní se podíváme na praktické fungování a flow programu.



Obr. 5: Diagram fungování programu

Prostředí je designováno s hlavním cílem, kterým je jednoduchost, jelikož v praktickém využití není program používán opakovaně jedním uživatelem, ale pouze jednou mnoha uživateli. V úvodní části je uživateli prezentována hlavní funkce programu a možnost pro vstup do testu. Z právních důvodů je v aplikaci i sekce o zásadách ochrany osobních údajů.

Poté, co uživatel napíše do kolonky pro pin číslo, které mu dá učitel, popřípadě kterákoliv osoba, která test organizuje, program odešle na server požadavek o kontrolu existence testu pod určeným pinem a pokud existuje, přenese uživatele do předtestové části, kde zadá své jméno a pokračuje k vyplnění testu. Test nemá daný časový limit. To má plně ve svých rukou člověk, který test spustil.

Program neindikuje uživateli, zda je jeho vstup správnou odpovědí, aby se předešlo tipování a zkoušení všech možných variant, dokud jedna není ta správná. Odpověď je správná, pokud uživatelův vstup bez prázdných znaků před a po vstupu a se všemi znaky převedenými do malých písmen je identický se správnou odpovědí upravenou stejnými procesy. Vyhodnocení testu tedy v kódu vypadá následovně

```
for (int i = 0; i < ExamView.answers.length; i++) {
  for (int k = 0; k < ExamView.answers[i].length; k++) {
    if (ExamView.answers[i][k].toLowerCase().trim() ==
        representatives[i].infoArr[k].toLowerCase().trim()) {
      correctlyAnswered++;
    }
  }
}
```

Obr. 6: Ukázka kódu - kontrola odpovědí

, kde `ExamView.answers` jsou uživatelovy vstupy a `representatives.infoArr` jsou správné odpovědi.

Aplikace si pamatuje všechny vstupy, které uživatel zadal, a tak se uživatel nemusí strachovat o ztracení informací a může se k jednotlivým zástupcům vracet a upravovat svoje odpovědi.

Uživatel může test ukončit, když dojde v testu k poslední položce. Program se vyplňujícího zeptá, zda opravdu chce test odevzdat. Nato program provede součet správných odpovědí a odešle je na server a uživatele přenese na domovskou stránku.

Závěr

Vytvořená aplikace je plně funkční a reálně použitelná. Grafické rozhraní dle mého plní nevyrušující účel a uživatele vede. Vzhledem k tomu, že Flutter mi na začátku procesu byl poměrně cizí, šel vývoj pomaleji než jsem očekával (k tomu přispíval i fakt, že Flutter pro web je v testovací fázi). I přesto jsem stanovené cíle splnil. Aplikace by mohla využít vyhodnocení testu ihned po ukončení testu, aby uživatel rovnou věděl, jak si vedl. Projekt se dá dále rozvíjet i jako aplikace na další platformy jako Android, iOS a desktop. Zdárným vytvořením aplikace jsem demonstroval mojí schopnost učit se nové věci.

Bibliografie

[1] Cloud Firestore. Cloud Firestore | Firebase [online]. [cit. 2021-03-08]. Dostupné z: https://firebase.google.com/docs/firestore

[2] Introduction to widgets. Introduction to widgets - Flutter [online]. [cit. 2021-03-08]. Dostupné z: https://flutter.dev/docs/development/ui/widgets-intro

Seznam obrázků