Jihočeská univerzita v Českých Budějovicích

Ekonomická fakulta

Katedra aplikované matematiky a informatiky

Teze bakalářské práce

Vývoj Multiplatformní Cashback Aplikace ve Flutteru

Vypracoval: Tomáš Čeloud

Vedoucí práce: Mgr. Radim Remeš, Ph.D.

České Budějovice

2023

Klíčová slova: Flutter, Dart, Cashback, Cross-platform

Anotace: Bakalářská práce popisuje problematiku vývoje mobilních aplikací. Jejím cílem je popsat proces vývoje mobilní aplikace pro podporu nakupování se službou cashback ve frameworku Flutter, která je dostupná na mobilních zařízeních se systémy Android a iOS. V rešeršní části se práce zabývá typy mobilních aplikací a jejich rozdělením, popisem alternativních vývojových možností, vývojovými prostředími a následným seznámením s vybraným frameworkem Flutter a programovacím jazykem Dart. Praktická část představuje popis vývoje samotné aplikace. Jejím cílem je vyvinout uživatelsky příjemnou aplikaci umožňující zobrazit hlavní informace o účtu formou nástěnky, zobrazovat poslední transakce, registraci partnerského účtu a obchodů, ale také možnost platby pomocí QR kódu. Představuje využité knihovny a jejich případné alternativy a popisuje registraci prostřednictvím služby BankId zabezpečenou standardem OpenId.

Úvod

Za téma bakalářské práce jsem zvolil vývoj multiplatformní aplikace ve frameworku Flutter. Tento nástroj na vývoj multiplatformních aplikací mě zaujal, a proto jsem chtěl prohloubit své znalosti jeho využitím při vývoji mobilní aplikace. Flutter patří mezi novější vývojové nástroje, ale jeho popularita v IT sektoru se prudce zvyšuje, a to hlavně díky jeho jednoduchosti a rychlosti.

Práce je rozdělena na teoretickou a praktickou část. Teoretická část se věnuje obecné problematice vývoje a návrhu mobilních aplikací a dostupným alternativám v tomto odvětví. Pozornost bude také věnována samotnému Flutteru a programovacímu jazyku Dart, ve kterém je celá aplikace programována.

Praktická část je poté zaměřena na samotný vývoj aplikace a zdokumentování jeho postupu, včetně všech nově nabytých poznatků ohledně vývoje samotného a věnuje se také struktuře aplikace a využitým knihovnám z oficiálního úložiště balíčků pub.dev.

Přehled literatury

Mezi hlavní literární zdroje patřila dokumentace Flutter, Dart, Android a Apple dostupné z:

- Flutter documentation | Flutter
- Dart documentation | Dart
- Documentation | Android Developers
- Featured | Apple Developer Documentation

Cíl práce

Hlavním cílem bakalářské práce je vytvoření frontendu mobilní cashback aplikace Pointly, která by měla po ověření uživatele prostřednictvím služby BankId umožnit uživateli platit v partnerských obchodech s procentuálním cashbackem. Pointly by měla umožňovat registraci partnerských podúčtů po uvedení kontaktních údajů a identifikačního čísla osoby, registraci obchodů na základě vyplnění adresy a procentuální odměny pro zákazníka, platbu pomocí QR kódu, zobrazení posledních účtenek a v neposlední řadě biometrické ověření uživatele pomocí otisku prstu, pokud je na zařízení dostupné. Aplikace umožní přepnutí režimu mezi světlým a tmavým a výběr jazyka mezi českým a anglickým, a to na základě zařízení nebo dle preferencí uživatele. Pokud by jazyk zařízení nebyl v aplikaci dostupný, výchozím jazykem aplikace bude angličtina.

Dart

Programovací jazyk Dart je vyvíjen společností Google. Jedná se o klientsky optimalizovaný jazyk pro vývoj aplikací. Cílem Dartu je nabídnout co nejprodukti vnější programovací jazyk pro cross-platform, umožňující flexibilní spuštění programů bez ohledu na platformu a framework. Dart poskytuje jazyk a prostředí, které pohání všechny Flutter aplikace, ale zároveň podporuje mnoho základních vývojářských úloh, jako je formátování, testování a analýza. (The Language, 2021)

Dart nabízí bezpečnost datových typů, která zajišťuje, že hodnota proměnné vždy odpovídá jejímu statickému typu. Dále nabízí vlastnost tzv. null safety, která vynucuje inicializaci všech proměnných nějakou konkrétní hotnotou.

Poskytuje širokou škálu knihoven, které poskytují pevnou základnu pro mnoho programovacích operací jako jsou například JSON a UTF-8 kodéry a dekodéry. Knihovna pro práci s matematickými konstantami, pseudonáhodnými čísly atd.

Kromě základních knihoven je dostupné také mnoho API rozhraní. Mezi tato rozhraní patří například škálovatelná knihovna **http**, která slouží k vytváření HTTP požadavků. Za zmínku stojí dále **intl**, který umožňuje jednoduše pracovat s formáty datumů, čísel, ale také třeba s výchozím globálním prostředím atd.

K dispozici jsou dva druhy kompilace, ve Flutteru se Dart používá jak při vývoji, tak i při finální kompilaci, ale typ kompilace je odlišný. V režimu vývoje a ladění se jedná o kompilátor nesoucí název just-in-time, zkráceně JIT. Kompilace kódu tímto způsobem se hodí v průběhu vývoje, neboť dochází k průběžné kompilaci, což umožňuje například funkci hot-reaload.

Druhým typem kompilace, kterou Dart nabízí, je kompilátor ahead-of-time nebo zkráceně AOT. Tento kompilátor funguje tak, že celý kód je zkompilován před samotným spuštěním aplikace, což umožňuje vyšší rychlost provedení a také umožňuje lepší předvídatelnost výkonu oproti jazykům využívajícím just-in-time kompilátor pro distribuční verzi aplikace. (Sullivan, 2019)

Flutter

Flutter je sada nástrojů vytvořená společností Google, pro tvorbu nativně kompilovaných aplikací pro mobilní zařízení, web a desktop na základě jednoho sdíleného zdrojového kódu.

Framework Flutter je navržen pro podporu mobilních aplikací pro systémy Android a iOS, ale zároveň pro interaktivní aplikace, které mohou fungovat na webových stránkách nebo počítači.

Flutter poskytuje pluginy pro Android Studio, IntelliJ IDEA a VS Code. V dokumentaci na internetu jsou dostupné veškeré informace potřebné ke konfiguraci editoru a tipy na využití pluginu. (Introduction, 2017)

Flutter klade důraz na widgety jako základní kompoziční jednotku. Widgety jsou základním stavebním kamenem uživatelského rozhraní. Každý widget je neměnnou deklarací části uživatelského rozhraní. Tyto widgety tvoří hierarchii založenou na kompozici. Widgety se do sebe vnořují, takže každý widget má svého rodiče a může přijímat jeho kontext. Tato struktura se nese až ke kořenovému widgetu. (Flutter Architectural Overview, 2017)

Tvorba projektu

Předpokladem je, aby byl Flutter nainstalovaný na zařízení, kde se chystá vývoj aplikace. Pro samostatné vytvoření projektu je potřeba využít konzolového rozhraní, tzv. CLI.

Konzolové rozhraní umožňuje vývojáři nebo vývojovému prostředí komunikovat s Flutterem. Pro umožnění komunikace například z Windows shell je nezbytné přidat Flutter mezi proměnné prostředí ve vlastnostech systému a nastavit k němu cestu.

Pokud máme Flutter přidanýa windows shell nám umožňuje využívat jeho příkazů, můžeme se přesunout do složky,kde chceme náš projekt založit a zadáme příkaz **flutter create <název_projektu>.**

State management

Uživatelské rozhraní v aplikaci se musí na základě různých akcí měnit. Nezáleží úplně na tom, zda se jedná o změnu vyvolanou vstupem od uživatele, nebo o změnu vyvolanou například ze serveru. V moment, kdy se tato změna provede, aplikace musí správně zareagovat a upravit komponenty, které jsou touto změnou ovlivněny.

Komponent jako takový nic nedělá, kromě toho, že popisuje součást uživatelského rozhraní aplikace. Pokud uživatel očekává změnu komponenty na základě nějaké akce, změna musí nastat. K tomu, aby se změna vizuálně projevila slouží tzv. stav.

Knihovna, kterou jsem vybral pro state management je Riverpod. Ten nám umožňuje definovat si stav widgetu jako proměnou a poté využívat jeho hodnoty kdekoli v aplikaci, stačí si proměnou pouze zavolat. K tomu účelu se definuje tzv. provider. Ovšem definicí providera to nekončí, dalším krokem je úprava samotného widgetu.

Riverpod přináší upravené widgety, jedná se o spotřebitele neboli consumer. Toto rozšíření znamená, že již nemusíme využívat Statefull a Stateless widgety, ale právě upravené *ConsumerWidget* a *ConsumerStatefullWidget*. V těchto widgetech poté můžeme využívat právě našich definovaných providerů, kteří obsahují stav aplikace. Kód se díky tomu stává o hodně udržitelnějším, neboť můžeme logiku widgetu držet odděleně od jeho vizuálu.

Routing

Schopnost navigace mezi různými stránkami aplikace je označována za tzv. routing. Tato funkcionalita je nezbytná pro každou vícestránkovou aplikaci. Routing umožňuje uživateli pohybovat se mezi různými sekcemi aplikace. V ideálním případě by veškeré stránky, na které se může uživatel prokliknout, měly být předem určeny a označeny. (Příloha 1 – Navigační diagram)

Z hlediska vývoje aplikace si musíme vysvětlit, co je to tzv. navigation stack. Při navrhování cest routingu musíme mít vždy přehled o tom, na jaké stránce se uživatel právě nachází a na jakých stránkách se nacházel předtím. Povědomí o těchto stavech nám dává dohromady tzv. navigation stack neboli navigační zásobník.

Pokud bychom jednoduše chtěli vysvětlit funkci tohoto zásobníku, jedná se o řadu obrazovek, které uživatel uvnitř aplikace navštívil. V moment, kdy je do zásobníku vložena nová obrazovka, jedná se o právě zobrazenou stránku, kterou může uživatel obsluhovat. Jestliže uživatel iniciuje ukončení stránky například tlačítkem zpět, tak by měla být obrazovka ze zásobníku vyjmuta a uživatel by měl být vrácen na předchozí, anebo námi vybranou stránku.

Local storage

Ukládání uživatelských dat mimo vrstvu databáze a schopnost aplikace k nim přistupovat bez nutnosti připojení k internetu řeší lokální úložiště neboli local storage. Uživatelská data uložena v lokálním úložišti by neměla být kritická, neexistuje zde žádná záruka toho, že zapsaná data zůstanou uložena napořád. Pro vymazání těchto dat stačí uživateli např. vymazat data o aplikaci nebo občas i samotnou aplikaci pouze aktualizovat.

Řešení lokálního úložiště nabízí knihovna s názvem **shared_preferences**. Knihovna umožňuje aplikaci ukládat malé množství dat na zařízení, kde je momentálně spuštěna. Data jsou na zařízení uložena ve formě klíč-hodnota. Datové typy, ve kterých můžeme pomocí knihovny **shared_preferences** ukládat, jsou například čísla, řetězce nebo booleanovské hodnoty čili pravda/nepravda.

Lokální úložiště v cashback aplikaci používám pro uchovávání uživatelských dat týkajících se nastavení aplikace.

Pro zpřístupnění lokálního úložiště v aplikaci musíme nejprve při jejím spuštění načíst veškerá data, která jsou pro danou aplikaci uložena na disku. Načtení dat dosáhneme tím, že ve funkci *main()* před buildem celé aplikace zavoláme metodu *initializeSharedPreferences()*. Tato asynchronní metoda nám zajistí, že do proměnné prefs budou vložena data z úložiště. To nám zajistí metoda *getInstance()*, kterou obsahuje třída *SharedPreferences*.

Motiv aplikace

Navržení motivu usnadňuje vývojářům přizpůsobit vzhled a celkový dojem aplikace. Motiv samotný zahrnuje barevné schéma, typografii a celkový vizuální design. Dobře navržený motiv pomáhá zlepšení celkové vizuální estetiky, díky tomu aplikace pro uživatele působí přitažlivěji a snadněji se používá.

Používání motivů v aplikacích usnadňuje vývojářům provádění změn týkajících se vizuálního designu. Pokud z nějakého důvodu dojde k rozhodnutí změny barevné paletky, kterou aplikace využívá, dá se tato změna jednoduše vyřešit. To může ušetřit mnoho času a usnadnit udržení konzistentního designu aplikace.

Implementace motivů ve Flutteru se globálně řeší uvnitř *MaterialApp* widgetu obdobně jako routing. Uvnitř tohoto widgetu se nachází parametry *themeMode* a *theme*. Parametr *themeMode* určuje, který z motivů bude použit, pokud jsou k dispozici jak světlý, tak tmavý motiv aplikace.

Uživatelský dojem z používaní aplikace je možné zpříjemnit tím, že výchozí motiv bude nastaven dle preferencí systému. Pokud má uživatel mobilní zařízení nastavené na tmavý režim, výchozím motivem aplikace bude také tmavý a naopak.

Lokalizace a jazyk

Přizpůsobení aplikace pro uživatele žijící v jiné zemi nebo mluvících jiným jazykem řeší lokalizace, známá také jako l10n. V kontextu aplikace je tato funkcionalita nezbytná pro oslovení širšího publika, tím, že bude aplikace k dispozici ve více jazycích.

Implementace lokalizace probíhá obdobně jako implementace motivů, protože se opět jedná o součást ovlivňující celou aplikaci. Řeší se tedy uvnitř *MaterialApp* widgetu. Tento widget obsahuje dva hlavní parametry týkající se lokalizace. Prvním je *locale* a druhým *supportedLocales*.

Parametr *locale* nastavuje výchozí jazyk pro celou aplikaci. Pokud je hodnota tohoto parametru *null*, výchozí jazyk je zvolen dle nastavení systému. Jestliže jazyk systému není obsažen v podporovaných jazycích aplikace, které jsou předány parametru *supportedLocales,* tak se jako výchozí jazyk vybere první, který se v listu podporovaných jazyků nachází.

Jestliže při inicializaci lokálního úložiště nebyla nastavena žádná hodnota klíče *prefferedLanguage*, tak v metodě *getDefaultLanguage()* dochází k výběru výchozího jazyka aplikace.

BankId

Přihlášení a registrace do cashback aplikace probíhá prostřednictvím bankovní identity s použitím technického standardu OpenId. Tento standard se používá pro ověřování totožnosti uživatelů na internetu. Proces přihlášení s použitím BankId zajišťuje správnost získaných údajů jednotlivých uživatelů. Rozsah neboli scope získávaných informací od uživatele se liší a je definován na backendu při tvorbě OpenId identifikátoru, který má tvar běžného URL. Rozsah v identifikátoru musí odpovídat rozsahu nastavenému na stránkách BankId, kde je nutné aplikaci zaregistrovat.

Notifikace

Upozornění v mobilních aplikacích se nazývá notifikace. Jedná se o formu upozornění zobrazovanou ve stavovém řádku zařízení. Tato upozornění mohou být generována přímo v aplikaci nebo je lze zasílat prostřednictvím serveru přes cloudové služby.

Notifikace nemusejí přenášet pouze titulek a krátký popis, lze s jejích pomocí zobrazit uživateli také obrázek nebo třeba přehrát zvuk. Mohou být nastaveny tak, aby se uživateli zobrazovaly opakovaně v ten samý čas nebo na základě časovače.

Ve Flutteru lze využít knihovnu **futter_local_notification**, která umožňuje zobrazovat uživateli notifikace generované přímo uvnitř aplikace. Toto řešení neumožňuje zasílat uživateli tzv. push notifikace ze serveru. Lze ho využít pouze pokud je aplikace aktivní, ačkoli vizuál notifikací nelze rozeznat.

Pokud uživateli potřebujeme vyvolat notifikaci i v moment, kdy aplikace není aktivně spuštěna, musíme využít knihoven **firebase_core** a **firebase_messaging**. Tyto knihovny nám

umožňují přijímat notifikace ze serveru přes cloudové služby společnosti Google. Vizuálně jsou tyto push notifikace shodné s těmi lokálními, ovšem jejich vyvolání je odlišné.

Jestliže je aplikace úspěšně zaregistrována a propojena na Firebase, uvnitř aplikace bychom měli být schopni získat token zařízení. Na backendu zasíláme notifikaci na konkrétní zařízení pomocí metody *sendNotification*. Vstupními parametry jsou právě *deviceToken, title* a *body*. Samotné zaslání notifikace poté funguje jako klasický požadavek HTTP POST. Požadavek je odeslán na servery provozované společností Google, které ho zpracují a následně vyvolají notifikaci prostřednictvím Firebase Cloud Messaging na konkrétním zařízení, které je odlišené unikátním *deviceTokenem*.

Závěr

Hlavním cílem této bakalářské práce bylo vyvinout funkční multiplatformní proof of concept frontend mobilní cashback aplikaci ve frameworku Flutter. Mezi hlavní cíle aplikace patřilo umožnit uživateli platbu pomocí QR čtečky, registraci partnerských účtů na základě identifikačního čísla a adresy, registraci obchodů, přihlášení do aplikace prostřednictvím služby BankId, ověření uživatele pomocí biometrických údajů, změnu motivu a jazyka, a zobrazení zůstatku a transakcí formou dashboardu.

Veškeré zásadní funkce aplikace byly implementovány a otestovány pro platformy Android a iOS. Po konzultaci s vedoucím práce jsme upustili od tvorby aplikace pro webové rozhraní z důvodu ztráty některých funkcionalit, pro které byla aplikace navržena.

Využití Flutter frameworku se ukázalo jako správná volba, a to hlavně díky jeho široké komunitě a komponentům, které ve velké míře usnadnily celkový vývoj aplikace.

Mobilní aplikaci Pointly dostalo k otestování celkem sedm uživatelů, a to z důvodu získání zpětné vazby. Vizuální prostředí aplikace uživatelé hodnotili jako přívětivé, ale samotné testování se neobešlo bez chyb. V průběhu bylo nalezeno pár grafických nedostatků, a to primárně ve světlém módů aplikace. Na odhalené nedostatky jsem byl uživateli upozorněn a opravil je. Funkční stránka aplikace se v této fázi testování obešla bez významných chyb. Aplikace uživatelům připadala přehledná a intuitivní.

Vzhledem k výše uvedenému lze považovat cíle práce za splněné.

Seznam literatury použité v tezích

1. Flutter architectural overview. (2017). Flutter. 2022, dostupné z https://docs.flutter.dev/resources/architectural-overvie
2. Introduction. (2017). Flutter. 2022, dostupné z https://docs.flutter.dev/resources/faq
3. Sullivan, M. S. (2019, 4. leden). Flutter: Don't Fear the Garbage Collecto. Medium. 2022, dostupné z https://medium.com/flutter/flutter-dont-fear-the-garbagecollector-d69b3ff1ca30
4. The language. (2017). Dart. 2022, dostupné z https://dart.dev/overview

Plný seznam použité literatury je uveden v textu bakalářské práce.

Další bibliografické údaje

Type of thesis: bachelor

Author: Tomáš Čeloud

Academic: 2022/2023

Department: Department of Applied Mathematics and Computer Science, Faculty of Economics, University of South Bohemia in České Budějovice, Czech Republic

Thesis supervisor: Mgr. Radim Remeš, Ph.D.

Number of pages: 67

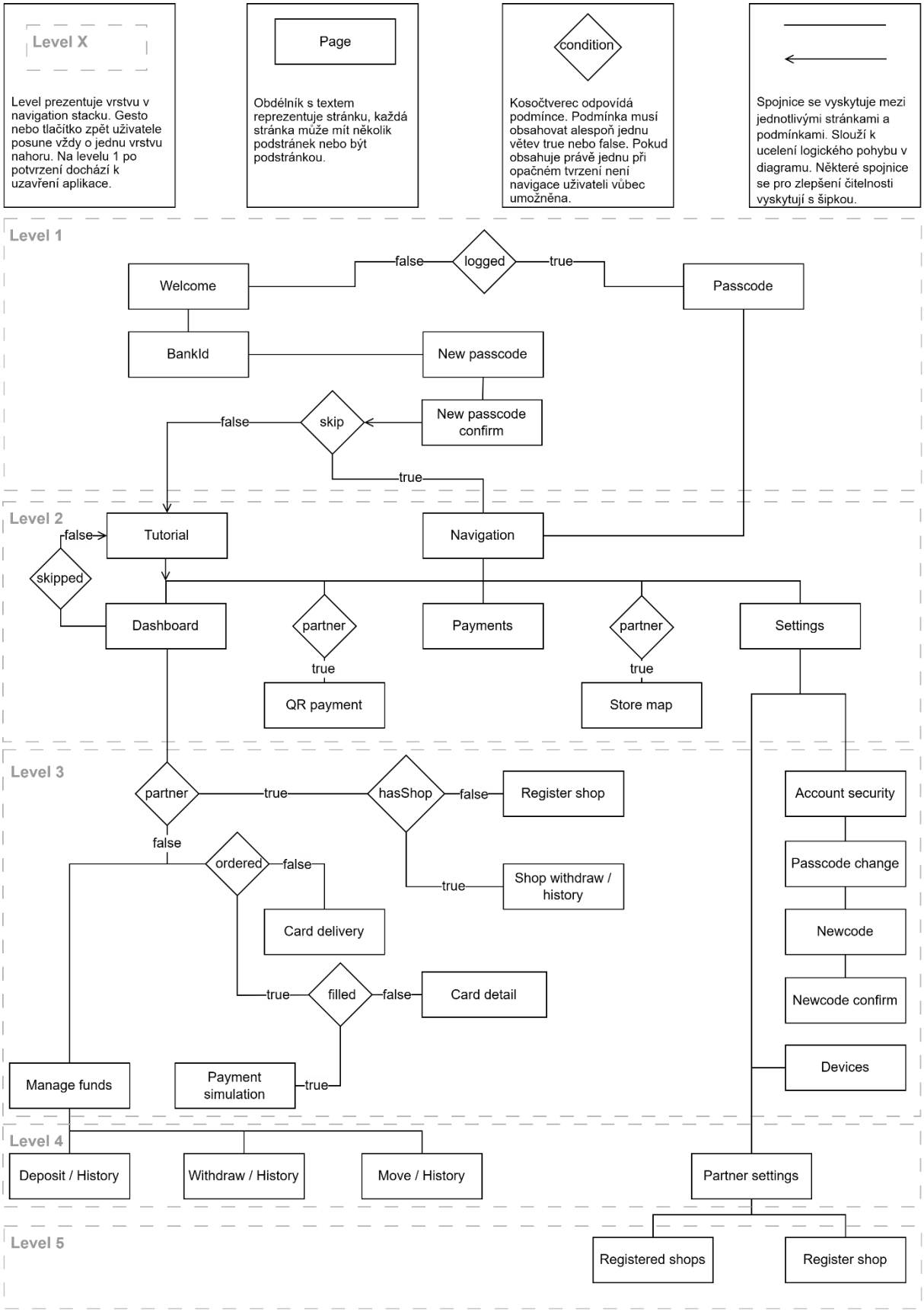
Language od thesis: CZ

Title of Thesis: Cross-platform Cashback Application Development in Flutter

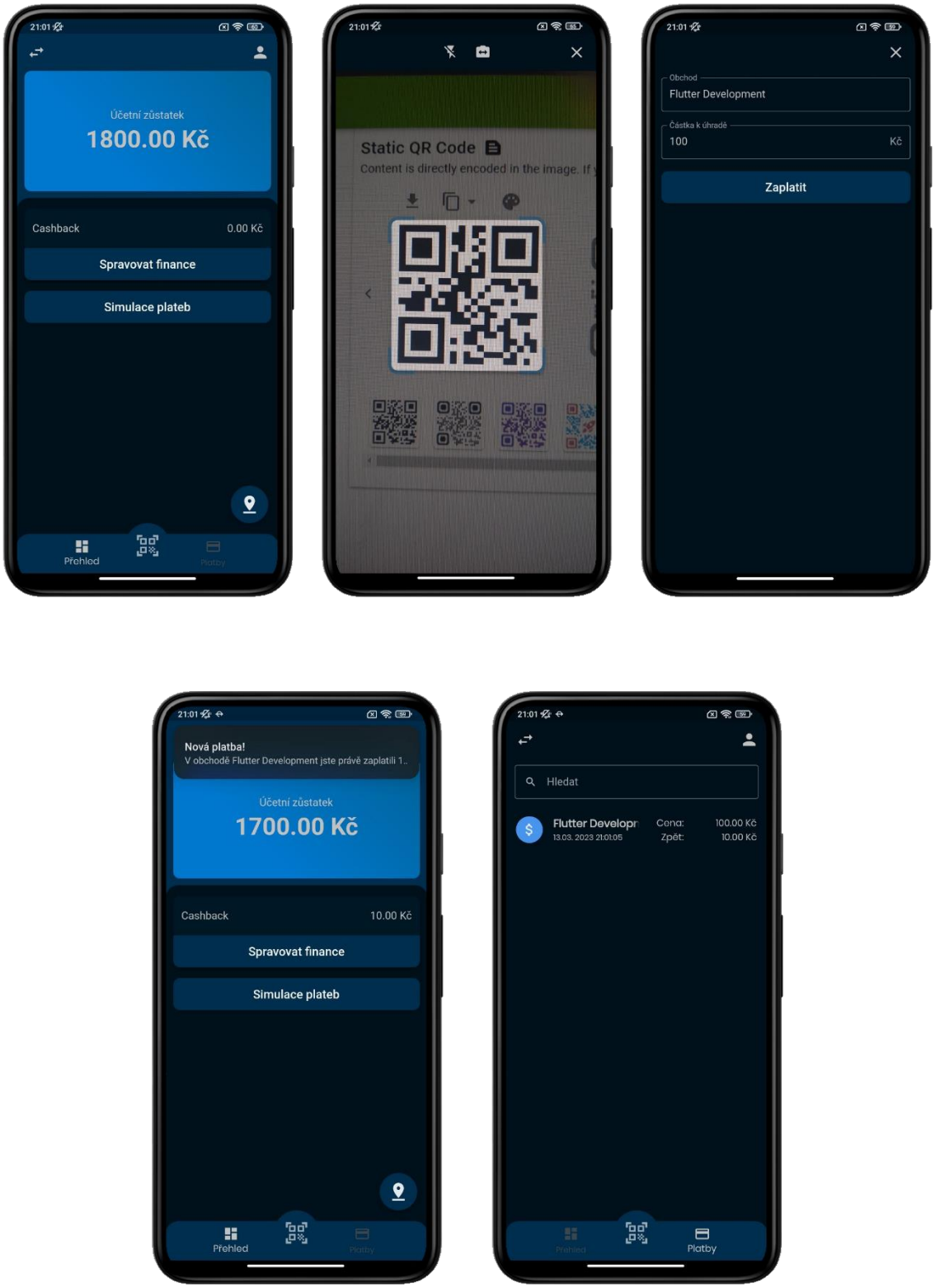
Annotation: The bachelor thesis describes the issue of mobile application development. Its aim is to describe the process of developing a mobile application to support shopping with cashback service in Flutter framework, which is available on Android and iOS mobile devices. The literary research part of the thesis discusses the types of mobile applications and how they are divided, a description of alternative development options, development environments and then an introduction to the selected Flutter framework and the Dart programming language. This work also points to a description of the development of the application itself. Its goal is to develop a user-friendly application that allows to display main account information in the form of a dashboard, display recent transactions, partner account and store registration, as well as the possibility to pay using a QR code. It presents the libraries used as well as their possible alternatives and describes registration via the BankId service secured by the OpenId standard.

Key words: Flutter, Dart, Cashback, Cross-platform

Přílohy vázané v práci:



Příloha 1 - Navigační diagram (vlastní)



Příloha 2 - Proces platby (vlastní)