**Vyšší odborná škola a Střední průmyslová škola, Jičín, Pod Koželuhy 100**

**18-20-M/01 Informační technologie**

**Mobilní aplikace pro výběr počítače**

**Individuální praktická maturitní práce**

**Jičín 2023 Lukáš Matuška**

Prohlašuji, že jsem individuální́ praktickou maturitní práci zpracoval(a) samostatně̌ a uvedl(a) v ní všechny prameny, literaturu a ostatní́ zdroje, které jsem použil(a).

V Jičíně 15. 3. 2023

**Anotace**

Tato práce se zabývá vývojem mobilní aplikace pro platformu iOS s využitím nejnovějších technologií a postupů. Jedná se o internetový obchod pro prodej notebooků s přidanou hodnotou průvodce pro výběr vhodného notebooku. Hlavní cíl je vytvořit rychlou, jednoduchou, moderní a přehlednou aplikaci pro veškeré věkové skupiny.

**Klíčová slova**

Vývoj mobilní aplikace, internetový obchod s notebooky, doporučování notebooku, platforma iOS, Xcode, Swift, SwiftUI

**Annotation**

This thesis deals with the development of a mobile application for the iOS platform using the latest technologies and practices. It is an online store for selling laptops with a value added guide for choosing a suitable laptop. The main goal is to create a fast, simple, modern and easy to use app for all age groups.

**Keywords**

Mobile application developement, internet store with notebooks, notebook recommendation, iOS platform, Xcode, Swift, SwiftUI

**Obsah**

[1 Úvod 6](#_Toc129807406)

[2 Teoretická část 7](#_Toc129807407)

[2.1 Podobné služby 7](#_Toc129807408)

[2.1.1 Microsoft 7](#_Toc129807409)

[2.1.2 NZXT 7](#_Toc129807410)

[2.2 Naše služba 7](#_Toc129807411)

[2.2.1 Cílová skupina 8](#_Toc129807412)

[2.2.2 Doporučení notebooku 8](#_Toc129807413)

[2.3 Tvorba aplikace pro iOS 9](#_Toc129807414)

[2.4 Použité technologie 10](#_Toc129807415)

[2.4.1 Vývojové prostředí 10](#_Toc129807416)

[2.4.2 Programovací jazyky 10](#_Toc129807417)

[2.4.3 Uživatelské rozhraní 11](#_Toc129807418)

[2.4.4 Backend 11](#_Toc129807419)

[2.4.5 Google Firebase 11](#_Toc129807420)

[2.4.6 Firebase Cloud Firestore 12](#_Toc129807421)

[2.4.7 Firebase Authentification 12](#_Toc129807422)

[2.4.8 Firebase Storage 12](#_Toc129807423)

[2.4.9 Firebase Cloud Functions 13](#_Toc129807424)

[2.4.10 Stripe Payments 13](#_Toc129807425)

[3 Praktická část 14](#_Toc129807426)

[3.1 Návrh designu 14](#_Toc129807427)

[3.2 Návrh datové struktury a databáze 15](#_Toc129807428)

[3.3 Založení projektu aplikace 16](#_Toc129807429)

[3.4 Napojení na backend Firebase 16](#_Toc129807430)

[3.5 Konfigurátor produktu 17](#_Toc129807431)

[3.6 Průvodce výběrem 18](#_Toc129807432)

[3.6.1 Způsob vyhodnocení 18](#_Toc129807433)

[3.7 Zákaznické účty 19](#_Toc129807434)

[3.7.1 Pomocná třída 19](#_Toc129807435)

[3.7.2 Uživatelské rozhraní 20](#_Toc129807436)

[3.8 Přijímání plateb 21](#_Toc129807437)

[3.8.1 Nastavení Stripe 21](#_Toc129807438)

[3.8.2 Nastavení aplikace 22](#_Toc129807439)

[3.8.3 Zpracování platby 22](#_Toc129807440)

[3.9 Pokladna 23](#_Toc129807441)

[3.9.1 Zadaní platební karty 24](#_Toc129807442)

[3.9.2 Zpracování objednávky 24](#_Toc129807443)

[3.10 Testování 26](#_Toc129807444)

[4 Závěr 27](#_Toc129807445)

[5 Zdroje 28](#_Toc129807446)

[6 Seznam obrázků 29](#_Toc129807447)

[7 Přílohy 30](#_Toc129807448)

[7.1 Tabulka odpovědí pro dotazník 30](#_Toc129807449)

[7.2 Datová struktura aplikace 34](#_Toc129807450)

# Úvod

Pokud si dnes na internetu chceme zakoupit nový notebook, máme nepřeberně různých obchodů, značek a modelů. Když si chceme vybrat ten správný, tak musíme alespoň základně rozumět technickým specifikacím, nebo si načíst různé články a recenze.

Přední české internetové obchody sice nabízí funkci průvodce nebo konfigurátoru, ale vždy vyžadují nějaké znalosti uživatele. Dokážou poradit ohledně kompatibility jednotlivých počítačových komponentů, ale nedoporučí konkrétní konfiguraci.

Toto je motivace pro vznik internetové služby, která uživateli automatizovaně pomůže s výběrem vhodného notebooku během okamžiku. Služba na uživatele neklade žádné nároky ohledně technických znalostí a může jí tedy využít naprosto kdokoliv.

Nejrozšířenější zařízení v dnešní době je mobilní telefon, a proto je cíl této maturitní práce vytvořit intuitivní mobilní aplikaci pro operační systém iOS, která tuto službu zprostředkuje všem uživatelům mobilních telefonů značky Apple. Aplikace dále umožní uživateli doporučený notebook rychle a jednoduše zakoupit.

Pro zbytek uživatelů zpracovává webovou verzi této služby Matěj Sedláček, který se zároveň podílí na tvorbě společné databáze a backendu služby.

V budoucnu plánujeme službu veřejně provozovat ve formě internetového obchodu na webu a jako aplikaci dostupnou ke stažení na Apple App Store pod názvem Compler.

# Teoretická část

## Podobné služby

### Microsoft

Podobnou službu nabízí Microsoft na adrese <https://www.microsoft.com/en-us/windows/help-me-choose>. Jedná se o krátký kvíz, jehož výstupem je seznam počítačů. Je velmi omezený a není vůbec specifický. Máte možnost zvolit, zda chcete počítač pro základní účely, učení, práci, hraní her, nebo výtvarné činnosti. Poté už máte možnost zvolit si pouze velikost obrazovky a značku.

Hlavní nevýhodou je právě absence konkrétnosti, tudíž systém uživateli zobrazí velký seznam počítačů a s výběrem mu moc nepomůže. Další nevýhodou je absence počítačů s jiným operačním systémem než Windows.

### NZXT

Další službou s podobným záměrem, ale úplně jiným zpracováním je konfigurátor od společnosti NZXT dostupný na <https://nzxt.com/build/pc>. Jedná se o konfigurátor stolních počítačů, kde je možné navolit si vlastní komponenty. Systém uživatele vede a pomáhá mu zvolit správné komponenty zobrazením aktuálního výkonu počítače a dodržením daného rozpočtu. Tento systém je zaměřen pouze na hráče počítačových her a uživatel musí mít alespoň základní znalosti o komponentech počítače.

## Naše služba

Každá konkurenční služba je nějak omezená. Buďto způsobem využití (například pouze pro hraní her) nebo operačním systém (pouze Windows). Na rozdíl od konkurenčních služeb chceme, aby na naše služba uměla doporučit konkrétní notebook pro konkrétního zákazníka bez výše zmíněných omezeních.

Pro obchodní strategii spoléháme na jednoduchost a unikátnost. Veškeré konkurenční služby jsou totiž v angličtině a žádná nefunguje v češtině. Pro českého zákazníka, který se nevyzná v počítačích bude náš systém jediná volba svého druhu. Dále chceme poskytovat vynikající zákaznický servis a doplňkové služby ke koupi.

### Cílová skupina

Tato mobilní aplikace je určena pro uživatele mobilních telefonů značky Apple, především je cílena na nezkušené uživatele jakéhokoliv věku, kteří nemají představu, jak počítač funguje a neví podle čeho nový počítač vybírat. Proto musí být aplikace jednoduchá na používání a velmi srozumitelná i pro nejméně znalého uživatele.

Nejdůležitější je pro takového uživatele navodit pocit důvěryhodnosti a seriózního jednání. Je totiž důležité, aby uživatel věřil, že doporučení, které mu aplikace nabízí je správné. Také je pro tohoto uživatele důležité, aby měl jistotu, že pokud mu počítač nebude fungovat, tak se může spolehnout na někoho, kdo mu pomůže a problém vyřeší.

Pokud budou veškeré výše zmíněné podmínky splněné, je největší šance na konverzi (tj. že si uživatel počítač zakoupí).

### Doporučení notebooku

Abychom mohli uživateli doporučit notebook, musíme se nejdříve seznámit s jeho požadavky. K tomu nám pomůže dotazník, který se bude skládat z několika otázek a uzavřených odpovědí. Po zodpovězení všech otázek uživatelem systém musí vyhodnotit konkrétní požadavky na specifikace notebooku.

Nejvíce nás zajímají požadavky na velikost obrazovky, výdrž baterie, kapacitu uložiště a operační paměti, a hlavně požadavky na výpočetní a grafický výkon. Požadavky na vzhled a ostatní funkce notebooku musíme zanedbat, jelikož to nelze pomocí jednoduchého dotazníku správně doporučit.

Výpočetní a grafický výkon budeme stanovovat u každého notebooku, respektive konfiguraci, v databázi pomocí skóre z benchmarku Geekbench 5 CPU a Geekbench 5 Compute OpenCL.

## Tvorba aplikace pro iOS

Mobilní aplikaci pro platformu iOS můžeme vytvořit dvěma způsoby. Buďto můžeme vytvořit nativní aplikaci nebo aplikaci tzv. Cross-platform.

Nativní aplikace je aplikace vytvořená přímo pro danou platformu pomocí specifických nástrojů a jazyků dané platformy. Jelikož používáme specializovaný jazyk a nástroje, tak nativní aplikace funguje mnohem rychleji a je více optimalizovaná. Také poskytuje uživatelské rozhraní, které se objevuje napříč systémem a je tak více přívětivá pro uživatele. Také jako vývojář můžeme využívat specializované systémové knihovny a API a lépe přistupovat k hardwarovým prostředkům. [11]

Cross-platform aplikace je aplikace vytvořená v univerzálním prostředí, které umožní výslednou aplikaci spustit na více platformách. Příkladem může být React Native nebo Flutter. Tyto nástroje umožňují napsat jednu aplikaci a tu potom kompilovat například pro web, Windows, macOS, iOS a Android. Pokud chceme mít aplikaci dostupnou na všech platformách, tak se jedná o velkou časovou úsporu. Nevýhodou Cross-platform aplikace je pomalejší chod a limitovaná funkcionalita, jelikož můžeme použít pouze funkce, které podporují všechny platformy. Další nevýhodou je méně intuitivní uživatelské rozhraní, jelikož nesdílí systémové prvky a může být pro uživatele složitá na zorientování. [11]

Jelikož neplánujeme podporu pro veškeré platformy a chceme využít systémové knihovny, tak pro naši aplikaci zvolíme první způsob, a to nativní aplikaci. Jako bonus bude pro uživatele známé systémové uživatelské rozhraní.

## Použité technologie

Obrázek Diagram komunikace aplikace

### Vývojové prostředí

Jediný způsob, jak můžeme vytvořit nativní aplikaci pro Apple platformy je užitím Xcode pro macOS. Jedná se o vývojové prostředí od společnosti Apple, které obsahuje vše potřebné k vývoji a publikaci aplikace pro veškeré platformy Applu. Umožňuje vývoj, debug, kompilaci kódu pro zařízení a zprostředkovává podepisování aplikací. Nabízí veškeré dostupné systémové knihovny a simulátor zařízení pro testování.

### Programovací jazyky

Pro vytvoření nativní aplikace pro operační systém iOS máme pouze 2 možnosti: Objective-C nebo Swift.

Objective-C je objektově orientovaný programovací jazyk od společnosti Apple, který pracuje na nízké úrovni a přímo vychází z jazyku C. Byl vydán již v roce 1984 a jedná se o primární způsob psaní kódu pro iOS. V Objective-C jsou napsané veškeré systémové API. [5]

Swift je moderní vysokoúrovňový programovací jazyk vytvořený společností Apple v roce 2014. Navazuje na Objective-C, ale poskytuje velmi jednoduchý a čitelný kód (podobný například C#). Je velmi úzce propojený s Objective-C, tudíž lze oba tyto jazyky používat současně. [6]

Pro psaní kódu budeme využívat moderní jazyk Swift, jelikož je to skvělá volba pro jednoduché aplikace jako je tato.

### Uživatelské rozhraní

Uživatelské rozhraní budeme vytvářet pomocí frameworku SwiftUI, který vychází z jazyka Swift. SwiftUI je nový framework pro tvorbu uživatelského rozhraní pro iOS 13 a novější. Nabízí intuitivní nástroje pro design, včetně živého náhledu. S rozsáhlou knihovnou systémových prvků se jedná o nejrychlejší a nejefektivnější způsob tvorby UI pro iOS. [7]

### Backend

Pro tuto aplikaci budeme potřebovat backend, který bude zajišťovat ověření uživatelů, správu objednávek a produktů a zprostředkovávat celý systém na doporučení notebooku.

Mohli bychom si vytvořit vlastní server s veřejnou IP adresou, který by obsahoval databázi a běžel by na něm náš backend kód. To by ale bylo velmi nákladné, a proto využijeme již hotového řešení v podobě cloud-backendu. Jedním z těchto řešení je Google Firebase, který obsahuje vše, co potřebujeme.

### Google Firebase

Google Firebase je platforma, která zprostředkuje služby pro vývoj mobilních a webových aplikací. Obsahuje služby pro sledování chování uživatelů, hostuje databáze a soubory, zajišťuje autorizaci uživatele a umožňuje běh kódu na serveru.

Pro základní funkce a limitovaný počet uživatelů Firebase nabízí zdarma předplatné. V případě potřeby využití pokročilejších funkcí, nebo růstu počtu uživatelů je potřeba platit měsíční poplatek. [1]

Firebase obsahuje API pro komunikaci s backendem. Toto API je zahrnuto v různých SDK (Software Developement Kit) pro různé platformy. My budeme využívat Firebase iOS SDK pro implementaci služeb Firebase do aplikace na platformě iOS. [1]

### Firebase Cloud Firestore

Cloud Firestore (dále pouze Firestore) je jednou ze služeb Firebase. Jedná se o NoSQL databázi, která se skládá z dokumentů a kolekcí. Pro práci se využívá SDK nebo REST. [2]

Firestore je flexibilní, škálovatelná databáze pro klientské i serverové aplikace. Umožňuje efektivně získávat data pomocí dotazů, zajišťuje synchronizaci a umí fungovat i v offline módu, pokud uživatel zrovna nemá dostupné internetové připojení. [2]

Firestore budeme využívat pro úschovu uživatelských dat, správu objednávek a jako databázi produktů.

### Firebase Authentification

Firebase Authentification (dále pouze Firebase Auth) poskytuje služby pro identifikaci a autentifikaci uživatele. Společně s Firestore také dokáže spravovat data o uživateli. Pomocí SDK můžeme vytvářet uživatelské účty a přihlašovat uživatele. Firebase Auth umožňuje přihlášení pomocí emailové adresy, telefonního čísla a pomocí sociálního přihlášení (Apple, Google, Facebook, Twitter…). [3]

Firebase Auth budeme využívat pro správu zákaznických účtů v aplikaci. To nám umožní uložit uživateli údaje v pokladně na příště a zobrazit historii objednávek.

### Firebase Storage

Firebase Storage je cloudové úložiště, které poskytuje Firebase. Umožňuje ukládat a spravovat soubory v cloudu, jako jsou obrázky, videa, zvukové soubory a jiné soubory. [12]

Firebase Storage budeme využívat pro uložení produktových obrázků a jejich následné zobrazení v aplikaci pomocí URL odkazů.

### Firebase Cloud Functions

Firebase Cloud Functions umožňuje psát kód v JavaScriptu, TypeScriptu a jiných podporovaných jazycích a nahrávat tento kód do cloudu, kde bude spouštěn v reakci na událost v aplikaci (změna v databázi) nebo pomocí REST API. [13]

Firebase Cloud Functions poskytuje mnoho výhod, včetně možnosti spouštět kód na serveru bez nutnosti konfigurace nebo správy serveru, což snižuje náklady a zjednodušuje vývoj aplikací. Firebase Cloud Functions také nabízí integraci s dalšími službami Firebase, což umožňuje vývojářům rychle vytvářet a spravovat komplexní aplikace. [13]

Jedná se o nejdůležitější část aplikace, jelikož bude zajišťovat zpracovávání online plateb a systém doporučení notebooku.

### Stripe Payments

Stripe Payments (dále pouze Stripe) nabízí hotové řešení systému pokladny a příjímání online plateb. Stripe obsahuje rozsáhlá API a SDK pro implementaci do iOS aplikace. Stripe nabízí hotové prostředí pokladny, které stačí implementovat v aplikaci, nebo si můžeme vytvořit vlastní rozhraní pokladny a využít Stripe API pro zprostředkování bezpečné online platby platební kartou, nebo pomocí expresní pokladny Apple Pay. [4]

# Praktická část

## Návrh designu

Než můžeme začít vytvářet aplikaci, potřebujeme alespoň základní návrh, jak by aplikace mohla vypadat. Za tímto účelem využijeme grafický nástroj Figma, je zdarma a je určený k tvorbě designu uživatelského rozhraní.

Snažíme se o přehledný minimalistický design v souladu se systémem iOS. Využíváme pouze systémové prvky, aby vše uživateli připadalo přirozené i při prvním spuštění aplikace.

Jako barevnou paletu jsme zvolili kombinaci velmi tmavé šedé a zářivé oranžové, doplněno ještě o další světlejší odstíny šedé a o bílou barvu.

Tmavé pozadí je dnes populární u technologických společností, a navíc dle magazínu Bootcamp je tmavé pozadí elegantní, více estetické a dodává pocit luxusu. [9]

Obsah obrázku text, monitor, černá, snímek obrazovky

Popis byl vytvořen automatickyOranžová se velmi hodí na doplnění tmavě šedého pozadí a dle magazínu Wishdesk se jedná o barvu optimismu, radosti a energie. [10]

Obrázek Návrh designu pomocí aplikace Figma

## Návrh datové struktury a databáze

Při tvorbě podobné aplikace je nejprve potřeba vytvořit návrh datové struktury a databáze. Ve Firestore máme k dispozici databázi, která se skládá z kolekcí a dokumentů. Dokument má své ID a obsahuje definovaná data a může obsahovat i další kolekce. Kolekce jsou sbírky dokumentů. Tímto způsobem můžeme větvit a organizovat naši databázi. Jelikož se nejedná o klasickou databázi, tak zde nemusíme řešit vazby a kardinalitu. [2]

První vytvoříme kolekci Products, která představuje všechny produkty dostupné v obchodě. Každý dokument v kolekci Products představuje jeden produkt. Dokument obsahuje obecné informace o produktu, obrázky, hlavní vlastnosti, popis, zaměření, dostupné barvy a všechny dostupné konfigurace. Každá konfigurace obsahuje technické specifikace, cenu a scóre z Geekbench 5 (CPU single core, CPU multi core, GPU OpenCL).

Dále vytvoříme kolekci ShippingMethods, která obsahuje všechny dostupné způsoby přepravy a bude využívána při tvorbě objednávky. Každý dokument představuje jeden způsob přepravy. Dokument obsahuje název přepravy, popis a cenu.

Také vytvoříme kolekci Orders, kde budou uložené veškeré objednávky. Každý dokument představuje jednu objednávku. Dokument obsahuje objednaný produkt, datum vytvoření, údaje o zákazníkovi a stav objednávky a platby.

Celou datovou strukturu aplikace a databáze si můžeme vizualizovat pomocí diagramu viz příloha číslo 2.

## Založení projektu aplikace

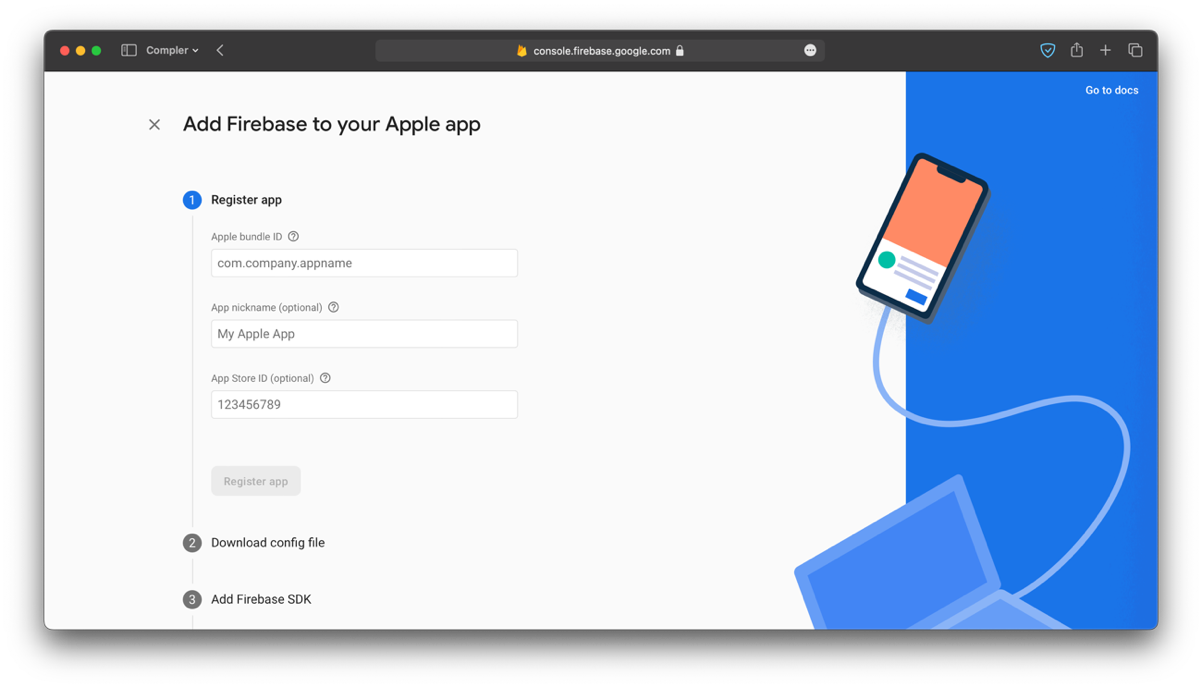
Jelikož vytváříme aplikaci pro platformu iOS, projekt si vytvoříme v prostředí Xcode.  
Vytvoříme nový projekt typu „App“ pro platformu iOS a zvolíme Swift jako programovací jazyk a SwiftUI pro tvorbu uživatelského rozhraní.

Po založení projektu v nastavení upravíme minimální vývojovou verzi iOS na iOS 16.0, abychom při vývoji mohli využívat nových užitečných prvků ze SwiftUI 4.0, které nám vývoj uživatelského rozhraní usnadní.

## Napojení na backend Firebase

Na stránkách Firebase si vytvoříme zdarma účet. Poté si můžeme ve Firebase vytvořit nový projekt s vhodným názvem.

Dále je potřeba do projektu přidat naši iOS aplikaci, v administraci klikneme na přidání nové aplikace a zvolíme platformu Apple. Nyní musíme zadat Apple bundle ID, které nalezneme v Xcode u našeho projektu.

Poté budete vyzvání na stažení konfiguračního souboru, který musíme přidat do projektu v Xcodu. Bez tohoto souboru a správného bundle ID komunikace nebude fungovat.

Obrázek Formulář pro přidání aplikace do Firebase

V Xcode projektu ještě musíme importovat Firebase SDK, to učiníme pomocí Swift Package Manageru. Vyhledáme Firebase iOS SDK a zvolíme potřebné knihovny a to: FirebaseAuth, FirebaseFirestore, FirebaseFirestoreSwift, FirebaseFunctions a FirebaseStorage.

Obsah obrázku text, snímek obrazovky, monitor

Popis byl vytvořen automatickyPoslední krok k úspěšnému spojení je konfigurace Firebase při spuštění aplikace. Při použití SwiftUI můžeme přidat inicializátor k hlavnímu View aplikace a konfiguraci provést tam.

Obrázek Konfigurace Firebase v iOS aplikaci

## Konfigurátor produktu

Konfigurátor se nachází v detailu produktu a umožňuje uživateli navolení požadované konfigurace produktu. V konfigurátoru je možné zvolit barvu, procesor, grafickou kartu, velikost operační paměti a velikost uložiště.

Nejprve se pro vybraný produkt získají z databáze veškeré dostupné konfigurace. Z tohoto pole se následně vytvoří sety pro každou sekci konfigurátoru (například set všech dostupných barev nebo procesorů) a předvolí se nejlevnější nebo doporučená konfigurace. V každé příslušné sekci se nachází všechny možnosti reprezentované tlačítky.

Důležité je neustále kontrolovat jaké možnosti je možné zvolit, jelikož produkt nemusí obsahovat konfigurace pro veškeré možné kombinace. Nedostupné možnosti zašednou, aby uživatel nemohl nechtěně navolit neexistující konfiguraci.

Obsah obrázku text

Popis byl vytvořen automatickyKaždá dostupná možnost v konfigurátoru zobrazuje také relativní cenu vzhledem k ceně aktuálně zvolené konfigurace (nedostupné a zvolené možnosti cenu neukazují). Tato cena se u každé možnosti při každé interakci přepočítá.

Obrázek Uživatelské rozhraní konfigurátoru

## Průvodce výběrem

Průvodce výběrem funguje jako dotazník o 8 otázkách. Otázky byly zvoleny ve spolupráci s Matějem Sedláčkem tak, aby dotazník byl jednoduchý, rychlý a zároveň pokryl co nejvíce možných způsobů využití notebooku a potřeb uživatele. Samozřejmě nelze pokrýt všechny možnosti, jelikož by byl dotazník moc dlouhý a složitý, proto jsme se zaměřili na to nejdůležitější pro neznalého uživatele.

### Způsob vyhodnocení

Vyhodnocování je zpracováváno pomocí Firebase Cloud Functions. Vytvoříme obslužnou funkci, která pomocí dodaných odpovědí z dotazníku doporučí nejvhodnější notebook z nabídky a 2 alternativy.

Každá odpověď dotazníku má přiřazený ideální parametr, z těchto ideálních parametrů složíme ideální konfiguraci počítače a v naší databázi notebooků hledáme nejpodobnější konfiguraci.

Vytvoříme pole konfigurací, kde každá konfigurace má vypočtený číselný rozdíl od ideální konfigurace. Toto pole seřadíme podle rozdílů od nejmenšího po největší a vrátíme první 3 položky pole.

## Zákaznické účty

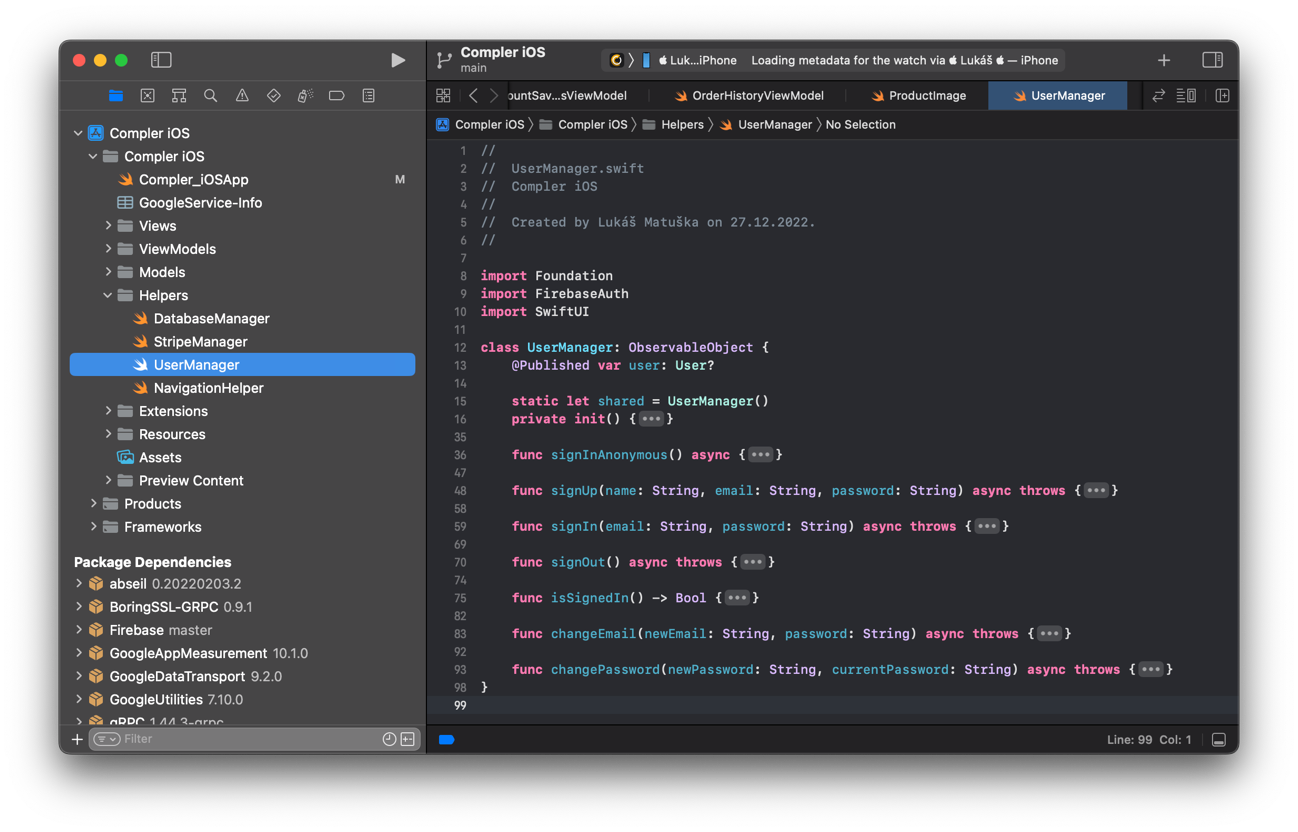
Zákaznické účty budou spravovány pomocí Firebase Auth. Budeme umožňovat vytvoření účtu a následné přihlášení pomocí emailové adresy a hesla.

Zákaznický účet bude v aplikaci dobrovolný a umožní přihlášenému uživateli uložit údaje v pokladně pro další nákup a sledovat vytvořenou objednávku.

### Pomocná třída

Pro obsluhu SDK si vytvoříme pomocnou třídu UserManager, která uchovává aktuální informace o uživateli a poskytuje veškeré obslužné metody. Chceme, aby to byla tzv. „singleton“ instance, to znamená, že chceme pouze jednu instanci dostupnou pro celou aplikaci. To zařídíme nastavením inicializátoru na private a vytvořením statické konstanty shared. Nyní k této sdílené instanci můžeme přistupovat z celé aplikaci pomocí UserManager.shared.

Abychom co nejvíce redukovali prodlevy využíváme asynchronní kód při zasílání požadavků na server (metody označené jako async). Jedná se o použití tzv. „Swift Concurrency“, která byla vydána v iOS 15.0.

UserManager implementuje protokol ObservableObject, to zajistí, že se provede aktualizaci grafického rozhraní v případě změně dat v proměnné označené jako @Published. V tomto případě to znamená, že grafické rozhraní bude vždy aktuálně reprezentovat stav proměnné user.

Obrázek Pomocná třída UserManager

### Uživatelské rozhraní

V přihlašovacím View je podstatné využít správné prvky a modifikátory, aby mohl systém správně uživateli pomáhat při vyplňování. Při správné integraci systém uživateli předvyplní jeho jméno, emailovou adresu a automaticky navrhne nové silné heslo, a to následně uloží do klíčenky.

Pro jméno a email využijeme klasický prvek TextField, ale pro heslo využijeme speciální variantu a to SecureField, která místo charakterů zobrazuje pouze tečky.

Obsah obrázku text

Popis byl vytvořen automatickyAutomatické doplňování umožníme využitím modifikátorů „textContentType“, kde určíme, jaké data dané textové pole obsahuje.

Obrázek Uživatelské rozhraní přihlašovací obrazovky

## Přijímání plateb

### Nastavení Stripe

Nejprve si musíme vytvořit účet na Stripe, poté v administraci povolíme platby kartou a získáme publishable key a secret key, které budeme potřebovat pro implementaci.

Když chceme přijmout platbu v aplikaci musíme na serveru vytvořit Stripe payment intent, který reprezentuje danou transakci. Za tímto účelem využijeme Firebase Functions ve spojení se Stripe API.

Ve Functions si definujeme dříve získané klíče (publishable a secret) a vytvoříme funkce, které tyto klíče zpřístupní aplikaci. Dále si vytvoříme funkci, která vytváří payment intent a vrací jeho id a tzv. client secret.

Dále si ve Functions vytvoříme funkci, která bude fungovat jako koncový bod pro Stripe webhooks. Url adresu této naší funkce zadáme v administraci Stripe. Tato funkce je volána pro každý Stripe event (jako je vytvoření payment intentu nebo přijetí platby).

Obsah obrázku text

Popis byl vytvořen automatickyPrimárně samotná aplikace při dokončení platby označí objednávku jako zaplacenou. V případě, že nastane problém a aplikace objednávku neoznačí budeme využívat „payment\_intent.succeeded“ webhook, který indikuje, že platba proběhla úspěšně jako záložní systém pro označení objednávky.

Obrázek Firebase Functions pro obsluhu Stripe API

### Nastavení aplikace

Jediné, co je potřeba udělat v projektu iOS aplikace je přidat Stripe SDK pomocí Swift Package Manageru a při spuštění aplikace nastavit publishable key pro Stripe API, což řeší níže uvedená pomocná třída. Stejně jako konfiguraci Firebase toto řešíme v inicializátoru hlavního View aplikace.

### Zpracování platby

Pro dokončení platby je potřeba po získání payment intentu pomocí StripeManageru (pomocná třída) tento intent potvrdit. Potvrzením dojde k dokončení platby a ke stržení peněz z platební karty.

Obsah obrázku text, snímek obrazovky, monitor, černá

Popis byl vytvořen automatickyK potvrzení využijeme poskytnutý SwiftUI modifikátor „paymentConfirmationSheet“ ze Stripe SDK. Tento modifikátor funguje tak, že pokud platba vyžaduje nějaké ověření ze strany banky, tak se zobrazí dialogové okno neboli sheet s výzvou. Tento modifikátor použijeme ve View pokladny.

Obrázek Uživatelské rozhraní pokladny

## Pokladna

Poté co si uživatel vybere produkt a klikne na tlačítko „Koupit“ zobrazí se mu View pokladny. Pokladna obsahuje jeden produkt a aplikace neobsahuje košík, nelze tedy objednat více produktů v jedné objednávce. Tento způsob jsme zvolili, jelikož náš cílový uživatel ve většině případů chce počítač pro sebe a košík by byl zbytečný krok navíc.

Pokladna obsahuje textová pole pro zadaní kontaktních, doručovacích a fakturačních údajů. Z databáze se získají dostupné způsoby doručení a zobrazí se formou tlačítek. Pokladna také umožňuje přihlášení stávajícího uživatele nebo vytvoření nového uživatelského účtu za účelem uložení údajů a následnému automatickému doplnění údajů do pokladny.

### Zadaní platební karty

Pro zadaní údajů platební karty používáme speciální prvek STPPaymentCardTextField ze Stripe SDK, který uživateli umožní zadat číslo karty, datum expirace a CVV kód. Tento prvek také obsahuje živou kontrolu zadaných údajů.

Jelikož se ale jedná o prvek pro framework UIKit a naše aplikace využívá novější SwiftUI je nutné prvek obalit do UIViewRepresentable, pomocí kterého můžeme využívat starší prvky z UIKitu ve SwiftUI.

Vytvoříme si UIViewRepresentable a uvnitř něho instanci STPPaymentCardTextField, kterému nastavíme požadovaný vzhled, placeholder pro pole expirace karty a zakážeme zadávání PSČ, jelikož to nepotřebujeme a jedná se o zbytečný krok pro uživatele.

Obsah obrázku text

Popis byl vytvořen automatickyTento náš nový prvek CardField využijeme ve View pokladny.

Obrázek CardField prvek

### Zpracování objednávky

Po vyplnění všech požadovaných polí v pokladně se uživateli zpřístupní tlačítko pro objednání.

Při kliknutí na toto tlačítko se vytvoří nová objednávka, zavolá se serverová funkce pro získání čísla objednávky a objednávka se uloží do databáze. Následně se vytvoří nový Stripe payment intent pro tuto objednávku a platbu potvrdíme. Pokud bude pro platbu potřeba například zadat autorizační SMS kód, zobrazí se uživateli příslušné UI.

Po skončení zpracovávání platby se zavolá funkce onPaymentComplete. V případě, že platba proběhla úspěšně, označíme objednávku jako zaplacenou a uživateli se zobrazí View potvrzení objednávky. V případě neúspěchu se uživateli v pokladně zobrazí chyba.Obsah obrázku text, monitor, snímek obrazovky, černá

Popis byl vytvořen automaticky

Obrázek Kód pro vytvoření a zaplacení objednávky

## Testování

Po dokončení aplikace jsme ji nechali vyzkoušet uživateli z cílové skupiny.

Pomocí jejich zpětné vazby jsme optimalizovali algoritmus pro doporučení notebooku, aby vracel vhodnější možnosti. Zjistili jsme totiž, že algoritmus doporučuje nepřiměřeně výkonné a drahé notebooky, proto jsme lehce ubrali nároky na výkon. Po úpravě jsme aplikaci znovu nechali otestovat a nyní již oslovení uživatelé byli mnohem spokojenější s doporučeným notebookem.

Také jsme zjistili, že je nutné přidat více informací o společnosti, aby bylo na první pohled patrné, co přesně nabízí. Ukázalo se, že jsme tento aspekt při návrh design zanedbali. Nápravu jsme učinili přidáním další informační karty na úvodní obrazovku s názvem společnosti, logem a výstižným popisem.

Při testování jsme zaznamenali, že někteří uživatelé mají problém s konfigurátorem produktu a trvalo jim příliš dlouho zjistit, jak přesně funguje. Jelikož je jednoduchost jedním z hlavních cílů aplikace, přidali jsme vysvětlující dialogy a zároveň jsme přidali popisy k jednotlivým oddílům konfigurátoru (co je to procesor atd.).

# Závěr

Podařilo se nám sestavit kompletní aplikaci internetového obchodu s přidanou hodnotou průvodce pro výběr notebooku. Testováním jsme ověřili a zajistili, že aplikace funguje správně a je pro cílovou skupinu obchodu srozumitelná a přehledná.

Při programování jsme využívali moderní postupy a struktury. Jelikož jsme využívali nejnovější verzi SwiftUI, narazili jsme na několik systémových chyb a ty řádně nahlásili v programu Apple Developer.

Před zveřejněním aplikace by bylo vhodné přidat možnost registrace pomocí Apple ID (Sign in with Apple) a možnost platby pomocí Apple Pay. Jedná se o prostředky pro maximalizaci uživatelského komfortu. Tyto funkce nejsou součástí této práce, jelikož jejich aktivace vyžaduje placený vývojářský účet Apple Developer.

Také je potřeba vytvořit systém pro rozesílání informačních emailů ohledně objednávek a vytváření faktur, jelikož v případě nárůstu počtu zákazníků by bylo časově náročně toto zajišťovat ručně.

Do budoucna by aplikace šla jednoduše rozšířit o další sortiment, a to stolní počítače, tablety. Také by se vyplatilo ještě investovat do průvodce výběrem a udělat ho ještě více konkrétní, jelikož nyní je směrován na běžné uživatele bez specifických nároků.

# Zdroje

1. Firebase. Firebase [online]. [cit. 09.02.2023]. Dostupné z: <https://firebase.google.com>
2. Firestore | Firebase. Firebase [online]. [cit. 09.02.2023]. Dostupné z: <https://firebase.google.com/docs/firestore>
3. Firebase Authentication. Firebase [online]. [cit. 09.02.2023]. Dostupné z: <https://firebase.google.com/docs/auth>
4. Accept a payment | Stripe [online]. [cit. 09.02.2023]. Dostupné z: <https://stripe.com/docs/payments/accept-a-payment?platform=ios&ui=custom>
5. Apple Developer [online]. Copyright © 2023 [cit. 09.02.2023]. Dostupné z: <https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>
6. Apple Developer [online]. Copyright © 2023 [cit. 09.02.2023]. Dostupné z: <https://developer.apple.com/swift/>
7. Apple Developer [online]. Copyright © 2023 [cit. 09.02.2023]. Dostupné z: <https://developer.apple.com/xcode/swiftui/>
8. CGDirector [online]. [cit. 09.02.2023]. Dostupné z: <https://www.cgdirector.com/what-kind-of-computer-do-i-need/>
9. Bootcamp [online]. [cit. 09.02.2023]. Dostupné z: <https://bootcamp.uxdesign.cc/e-commerce-dark-mode-case-study-ff3d67efae81>
10. Wishdesk [online]. [cit. 09.02.2023]. Dostupné z: <https://wishdesk.com/blog/color-schemes-for-ecommerce-site>
11. Native vs Cross-Platform Development: Pros & Cons Revealed. *Mobile and web app development company — Uptech* [online]. Copyright © 2016 [cit. 03.03.2023]. Dostupné z: <https://www.uptech.team/blog/native-vs-cross-platform-app-development>
12. Cloud Storage for Firebase. *Firebase* [online]. [cit. 09.02.2023]. Dostupné z: <https://firebase.google.com/docs/storage>
13. Cloud Functions for Firebase. *Firebase* [online]. [cit. 09.02.2023]. Dostupné z: <https://firebase.google.com/docs/functions>

# Seznam obrázků

[Obrázek 1 Diagram komunikace aplikace 10](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807391)

[Obrázek 2 Návrh designu pomocí aplikace Figma 14](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807392)

[Obrázek 3 Formulář pro přidání aplikace do Firebase 16](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807393)

[Obrázek 4 Konfigurace Firebase v iOS aplikaci 17](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807394)

[Obrázek 5 Uživatelské rozhraní konfigurátoru 18](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807395)

[Obrázek 6 Pomocná třída UserManager 20](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807396)

[Obrázek 7 Uživatelské rozhraní přihlašovací obrazovky 21](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807397)

[Obrázek 8 Firebase Functions pro obsluhu Stripe API 22](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807398)

[Obrázek 9 Uživatelské rozhraní pokladny 23](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807399)

[Obrázek 10 CardField prvek 24](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807400)

[Obrázek 11 Kód pro vytvoření a zaplacení objednávky 25](file:////Users/lukasmatuska/Library/Mobile%20Documents/com~apple~CloudDocs/SPŠ%20STUDIUM/IMP/Individuální%20maturitní%20práce.docx#_Toc129807401)

# Přílohy

## Tabulka odpovědí pro dotazník

Otázky a odpovědi pro dotazník byly tvořeny ve spolupráci se studentem Matějem Sedláčkem.

| Odpověď | Rozlišení kamery | Velikost obrazovky | Výrobce | Výdrž baterie | Kapacita uložiště | Kapacita OP | CPU SC skóre | CPU MC skóre | OpenCL skóre |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Chci kvalitní kameru na časté videohovory | >= FullHD | – | | | | | | | |
| Vystačím si se základní kamerou | >= HD | – | | | | | | | |
| Preferuji operační systém Windows | – | | není Apple | – | | | | | |
| Preferuji operační systém macOS | – | | je Apple | – | | | | | |
| Na operačním systému mi nezáleží | – | | | | | | | | |
| Většinu dne jsem bez možnosti nabíjení a potřebuji se na notebook spolehnout | – | | | >= 10h | – | | | | |
| Potřebuji notebook využívat alespoň několik hodin bez nabíječky | – | | | >= 5h | – | | | | |
| Notebook mám neustále připojený k napájení | – | | | | | | | | |
| Chci mít hodně úložného místa | – | | | | 512 GB | – | | | |
| Potřebuji mít dostatek uložiště na klasické používání | – | | | | 256 GB | – | | | |
| Nepotřebuji mnoho uložného prostoru | – | | | | | | | | |
| Potřebuji kompaktní notebook na cestování | – | < 15“ | – | | | | | | |
| Preferuji velkou obrazovku | – | > 14“ | – | | | | | | |
| Na velikosti notebooku mi nezáleží | – | | | | | | | | |
| Chci si užít plynulé hraní nejnovějších her | – | | | | | 16 GB | 1000 | 4000 | 120000 |
| Plánuji hrát náročnější hry | – | | | | | 16 GB | 1000 | 3000 | 60000 |
| Vystačím si s hraním méně náročných her | – | | | | | 8 GB | 700 | 2000 | 30000 |
| Počítačové hry nehraji vůbec | – | | | | | | | | |
| Pracuji s 3D grafikou | – | | | | | 32 GB | 1500 | 5000 | 40000 |
| Zabývám se střihem videa | – | | | | | 16 GB | 1200 | 4000 | 30000 |
| Budu pouze upravovat fotografie | – | | | | | 8 GB | 700 | 1500 | 10000 |
| Nezabývám se kreativní grafickou tvorbou | – | | | | | | | | |
| Zabývám se vývojem ve 3D (například Unity) | – | | | | | 16 GB | 1500 | 5000 | 40000 |
| Vyvíjím aplikace a programy | – | | | | | 16 GB | 1200 | 4000 |  |
| Vývojem softwaru se nezabývám | – | | | | | | | | |

## Datová struktura aplikace