Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Mobilní telefony a aplikace pro iOS

Tran The Dat

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

The Dat Tran

Informatika

Název práce

Mobilní telefony a aplikace pro iOS

Název anglicky

Mobile phones and applications for iOS

Cíle práce

Hlavním cílem práce je charakterizovat a zhodnotit prostředí a nástroje pro tvorbu aplikací na operační systém iOS, včetně vytváření a testování vlastní aplikace.

Dílčí cíle:
- klasifikace a hodnocení operačních systémů
- požadavky na software a hardware s důrazem na iOS
- programy na vývoj aplikací a tvorbu dokumentace
- závěry a doporučení

Metodika

Metodika je založena na studiu informačních zdrojů a vytvoření přehledu řešené problematiky, týkající se mobilních telefonů a tvorby aplikací na operačním systému iOS, vytváření papírového prototypu a programu (aplikace). Na závěr bude ukázka dokončené aplikace a budou formulovány závěry a doporučení.

Doporučený rozsah práce

40-50 stran

Klíčová slova

iOS, aplikace, hardware, software, telefony, testování, design

---

Doporučené zdroje informací

Co je UX a UI design. Co je UX a UI design [online]. Dostupné z: https://www.cojeuxui.cz
Jak vytvořit aplikaci pro iPhone (s obrázky) – wikiHow. [online]. Dostupné z: https://www.wikihow.cz/Jak-vytvo%C5%99it-aplikaci-pro-iPhone
LACKO, Ľuboslav. Vývoj aplikací pro iOS. Brno: Computer Press, 2018. ISBN 978-80-251-4942-3.
WEB & MOBILE DEVELOPMENT AGENCY | Rascasone. WEB & MOBILE DEVELOPMENT AGENCY | Rascasone [online]. Copyright © [cit. 04.06.2022]. Dostupné z: https://www.rascasone.com
Začínáme programovat ve Swiftu – část 1. – Letem světem Applem. Letem světem Applem – Apple magazín [online]. Copyright © Všechna práva vyhrazena [cit. 07.06.2022]. Dostupné z: https://www.letemsvetemapplem.eu/2019/01/12/zaciname-programovat-ve-swiftu-cast-1/

Předběžný termín obhajoby

2023/24 ZS – PEF

Vedoucí práce

doc. Ing. Jiří Vaněk, Ph.D.

Garantující pracoviště

Katedra informačních technologií

Elektronicky schváleno dne 7. 6. 2022

doc. Ing. Jiří Vaněk, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 27. 10. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Děkan

V Praze dne 13. 03. 2024

Čestné prohlášení

Prohlašuji, že svou bakalářskou práci "Mobilní telefony a aplikace pro iOS" jsem vypracoval samostatně pod vedením vedoucího bakalářské práce doc. Ing. Jiřího Vaňka, Ph.D. a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 15.03.2024 _____

Poděkování

Rád bych touto cestou poděkoval doc. Ing. Jiřímu Vaňkovi, Ph.D. za cenné rady, věcné připomínky a vstřícnost při konzultacích při vypracovávání této bakalářské práce.

Aplikace iOS

Abstrakt

Tato bakalářská práce se zaměřuje na charakterizaci a hodnocení prostředí pro tvorbu aplikací pro operační systém iOS, včetně vývoje a testování vlastní aplikace. Motivací pro tuto práci je dlouhodobý zájem autora o vývoj aplikací a možnost osobního rozvoje v této oblasti.

Bakalářská práce je rozdělena na dvě hlavní části. Teoretická část se věnuje základním pojmem v oblasti mobilních telefonů, operačních systémů, programovacích jazyků pro sytém iOS a nástrojů pro tvorbu aplikace. Praktická část představuje autorské zkušenosti s vývojem aplikací a zahrnuje vlastní aplikaci. Nakonec práce obsahuje vyhodnocení vývojového prostředí, kde byla aplikace vytvořena.

Tímto způsobem se bakalářská práce snaží poskytnout pohled na vývoj aplikací pro iOS a může sloužit jako užitečný zdroj informací pro další výzkum v této oblasti.

Klíčová slova: iOS, Xcode, Swift, Objective-C, Aplikace, Android, Prototyp, programování, software

Applications iOS

Abstract

This bachelor thesis focuses on characterizing and evaluating the environment for developing applications for the iOS operating system, including the development and testing of a custom application. The motivation for this work stems from the author's long-standing interest in application development and the opportunity for personal growth in this area.

The bachelor thesis is divided into two main parts. The theoretical part addresses fundamental concepts in the field of mobile phones, operating systems, programming languages for the iOS system, and tools for application development. The practical part presents the author's experiences with application development and includes a custom application. Finally, the thesis includes an evaluation of the development environment where the application was created.

In this way, the bachelor thesis aims to provide insight into application development for iOS and can serve as a useful source of information for further research in this area.

Keywords: iOS, Xcode, Swift, Objective-C, Application, Android, Prototype, programming, software

Obsah

1 Úvod **9**

2 Cíl práce a metodika **10**

2.1 Cíl práce 10

2.2 Metodika 10

3 Teoretická část **11**

3.1 Úvod do vývoje aplikací 11

3.2 Seznámení s operačními systémy 12

3.2.1 Android 13

3.2.2 iOS 14

3.3 Xcode 15

3.3.1 Vývojové prostředí 15

3.3.3 Playgrounds 20

3.4 Flutter 20

3.5 Objective-C 22

3.6 SWIFT 23

4 Praktická část **24**

4.1 Výběr prostředí 24

4.2 Požadavky na software a hardware 25

4.3 Prototyp a návrh aplikace 26

4.4 První spuštění 27

4.5 Interface builder 29

4.6 Vytváření aplikace 33

5 Zhodnocení a doporučení **38**

6 Závěr **40**

7 Seznam použitých zdrojů **41**

8 Seznam obrázků **44**

9 Přílohy **44**

1 Úvod

V dnešní digitální éře, kdy mobilní telefony tvoří nedílnou součást každodenního života, jsou operační systémy Android a iOS středem pozornosti.[1] Tyto systémy přinesly revoluci do způsobu, jakým integrujeme s technologiemi a aplikacemi, a otevřely nekonečné možnosti pro vývojáře po celém světě. Tato bakalářská práce se zabývá zkoumáním a charakterizováním prostředí vytváření aplikace pro jednoho z těchto operačních systémů, a to iOS od společnosti Apple.

iOS, s jeho intuitivním uživatelským rozhraním a bezpečnostními funkcemi, představuje jedinečnou platformu pro tvorbu mobilních aplikací, které mění způsob, jakým lidé komunikují, pracují, navíc i nabízí další formu zábavy. Pro vývoj aplikací pro iOS je nezbytný nástroj Xcode, integrované vývojové prostředí, které nabízí vývojářům širokou škálu funkcí pro návrh, vytváření a testování aplikací. Jazyk Swift, vyvinutý společností Apple, se stal klíčovým nástrojem pro vývoj aplikací pro iOS a přinesl efektivitu a flexibilitu do světa programování.[2]

Tato práce se zaměřuje na zkoumání prostředí pro vývoj aplikace do systému iOS za využití nástroje Xcode a programovacího jazyka Swift. Zabývá se analýzou prostředí, technologií a postupů, které jsou nezbytné pro úspěšný vývoj aplikací pro tuto platformu.

2 Cíl práce a metodika

2.1 Cíl práce

Hlavním cílem této práce je charakterizovat a zhodnotit prostředí a nástroje pro tvorbu aplikací na operační systém iOS, včetně vytváření a testování vlastní aplikace. Tento cíl je rozdělen do dílčích cílů, které zahrnují klasifikaci a hodnocení operačních systémů, stanovení požadavků na software a hardware s důrazem na iOS, identifikaci programů na vývoj aplikací a tvorbu dokumentace a následně formulaci závěrů a doporučení. Cílem této práce je poskytnout ucelený pohled na prostředí a nástroje nezbytné pro vývoj aplikací pro iOS a přispět k obohacení znalostí v této oblasti.

2.2 Metodika

Metodika této bakalářské práce se opírá o systematický přístup k prozkoumání problematiky spojené s mobilními telefony a tvorbou aplikací pro operační systém iOS. Prvním krokem bylo detailní studium relevantních informačních zdrojů, které poskytly ucelený přehled o této problematice, včetně aktuálních trendů, technologií a postupů v oblasti tvorby aplikací pro iOS. Na základě této analýzy byl vypracován a detailně zdokumentován teoretický rámec práce, který poskytuje teoretický základ a metodologický rámec pro další zkoumání a práci s touto problematikou.

Následně byl proveden návrh a vytvoření prototypu aplikace, což umožnilo rychlou validaci navrhovaných funkcí a uživatelského rozhraní. Tento prototyp sloužil jako výchozí bod pro další iterativní vývoj aplikace. Samotné programování aplikace probíhalo v prostředí integrovaného vývojového prostředí Xcode, které se stalo základním nástrojem pro vytváření aplikací pro iOS. Během této fáze byly aplikovány moderní principy softwarového inženýrství, jako je testování a simulace ve virtuálním zařízení.

V závěru práce je důležité shrnout, proč bylo vybráno konkrétní programovací jazyk a vývojové prostředí pro tvorbu aplikace. Hlavním cílem této práce není jen vytvoření funkční a uživatelsky přívětivou aplikaci pro iOS, ale také charakterizace a hodnocení prostředí, ve kterém probíhal vývoj.

3 Teoretická část

Tato bakalářská práce se skládá ze dvou hlavních částí: teoretické a praktické. Teoretická část se zaměřuje na důležitá témata spojená s vývojem mobilních aplikací pro operační systém iOS.

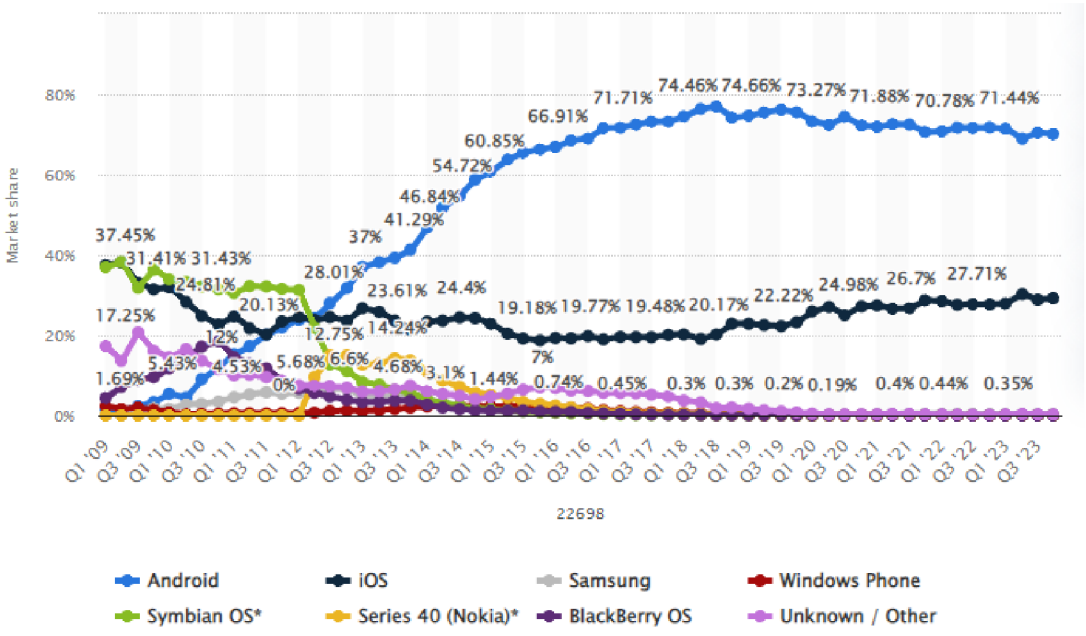
První část práce se věnuje mobilním operačním systémům obecně a detailně se zabývá iOS, který je vyvinut společností Apple a patří mezi nejpopulárnější operační systémy pro mobilní zařízení na světě. Tato část poskytuje užitečný kontext a základní informace pro pochopení celé problematiky. Dalším klíčovým bodem teoretické části jsou programovací jazyky Swift a Objective-C, vývojové prostředí Xcode a Flutter, které jsou klíčovými nástroji pro tvorbu aplikací pro iOS. Swift je známý svou efektivitou a snadnou čitelností, což ho činí atraktivní volbou pro vývojáře. Xcode je integrovaný vývojový prostředek s bohatým sadou nástrojů pro vývoj iOS aplikací, které je přímo vytvořen společností Apple.

3.1 Úvod do vývoje aplikací

Vývoj mobilních aplikací je proces navrhování, vytváření a testování software určeného pro mobilní zařízení, jako jsou chytré telefony a tablety. Klíčovými kroky tohoto procesu jsou plánování a návrh aplikace, samotný vývoj kódu, testování na různých zařízeních, publikace v obchodech s aplikacemi a udržování aplikace v aktuálním stavu. Celkově se jedná o komplexní úkol, který vyžaduje pečlivou přípravu a technickou zručnost.

3.2 Seznámení s operačními systémy

Mobilní aplikaci lze definovat jako software, který je navržen pro běh na přenosném zařízení, například mobilním telefonu či tabletu. Na trhu existuje mnoho různých mobilních platforem, což představuje výzvu při vývoji aplikací. Zákazníci, kteří si přejí vytvořit mobilní aplikaci, často požadují, aby byla dostupná na více platformách. To v praxi znamená, že pro dosažení plnohodnotné funkcionality na různých platformách by bylo třeba vyvíjet aplikaci mnohokrát, neboť každá platforma má své specifické požadavky na vývoj. Obrázek 1 představuje graf, který ukazuje zastoupení mobilních operačních systému v letech 2009 až do roku 2023.



Obrázek 1 - zastoupení mobilních operačních systémů na trhu v letech 2009-2023 [5]

V grafu je vidět zastoupení mobilních operačních systémů na trhu, kde v poslední dominují Android a iOS, které dohromady tvoří zhruba 99% podíl. Android, vyvinutý společností Google, je nejpoužívanějším systémem díky své otevřenosti a dostupnosti na široké škále zařízení.[3] Naopak iOS, vytvořený společností Apple, je exkluzivně dostupný na zařízeních této značky a vyniká uživatelským prostředím a bezpečností.[4] Tento duopol ovládající mobilní trh určuje strategie vývoje aplikací a má zásadní vliv na celý ekosystém mobilních technologií. V této části práce se zaměříme hlavně na operační systém iOS a Android.

3.2.1 Android

Operační systém Android, vyvinutý společností Google, je jedním z předních mobilních operačních systémů na světě. Jeho rozsáhlá penetrace na trhu a otevřený zdrojový kód z něj činí atraktivní platformu pro vývoj mobilních aplikací.[6]

Otevřený zdrojový kód: Android je postaven na otevřeném zdrojovém kódu, což znamená, že jeho zdrojový kód je volně dostupný pro vývojáře. Tato otevřenost umožňuje vývojářům upravovat systém dle svých potřeb, což vede k rozmanitosti verzí Androidu na trhu.[7]

Google Play: Pro distribuci aplikací na Androidu slouží Google Play Store, což je obchod s aplikacemi, kde uživatelé mohou stahovat a aktualizovat aplikace. Google Play Store je velmi rozsáhlý a obsahuje miliony aplikací, což poskytuje uživatelům širokou škálu aplikací, her, filmů a elektronických knih.[8]

Rozmanitost hardware: Android běží na různých zařízeních od různých výrobců. Tato rozmanitost zahrnuje různé velikosti obrazovek, výkonné procesory a různé designy. To dává uživatelům možnost vybrat si zařízení, které nejlépe vyhovuje jejich potřebám.[9]

Verze Androidu: Android má mnoho verzí s různými jmény (například Android 14 s kódovým označením "Upside Down Cake"). Každá nová verze přináší vylepšení a nové funkce, jako jsou například nové vzhledy aplikací, lepší zabezpečení a opravy nedostatků minulých verzí.[9]

Vývoj aplikací pro Android: Vývoj aplikací pro Android je možný pomocí programovacího jazyka Java nebo Kotlin. Android Studio, vývojové prostředí od Google, je klíčovým nástrojem pro vývojáře. Toto prostředí poskytuje bohatou sadu nástrojů a zdrojů, které usnadňují proces vytváření a testování aplikací pro Android. [10]

Android je významným hráčem na trhu mobilních operačních systémů a jeho otevřenost, rozmanitost a bohatý ekosystém aplikací ho činí atraktivní volbou pro uživatele i vývojáře.

3.2.2 iOS

Operační systém iOS, vyvinutý společností Apple, je jedním z nejvýznamnějších hráčů na trhu mobilních operačních systémů. Jeho jedinečné vlastnosti a integrovanost s ekosystémem Apple ho činí významnou platformou pro mobilní aplikace.[6]

Základní bezpečnost a ochrana soukromí: iOS je známý svou vysokou úrovní bezpečnosti a ochranou osobních dat uživatelů. Apple dbá na to, aby aplikace pro iOS prošly pečlivou kontrolou, a umožňuje uživatelům dávat aplikacím omezený přístup k citlivým údajům. [12]

Homogenita: iOS je pevně spojen s konkrétními zařízeními od Apple, jako jsou iPhone, iPad a iPod Touch. Tato homogenita v hardwaru a softwaru zjednodušuje vývoj aplikací, protože vývojáři nemusí řešit širokou škálu různých konfigurací a verzí.[11]

App Store: Aplikace pro iOS jsou dostupné pouze prostřednictvím App Store, který Apple přísně kontroluje. Tato kontrola zajišťuje vysokou kvalitu aplikací a bezpečnost pro uživatele.[11]

Integrace s ekosystémem Apple: iOS je pevně propojen s dalšími produkty od Apple, jako je macOS, iCloud a Apple Watch. Tato integrace umožňuje synchronizaci dat a fungování aplikací napříč různými zařízeními.[11]

Vývoj aplikací pro iOS: Probíhá výhradně v prostředí Xcode, což je oficiální vývojové prostředí od Apple. Programovací jazyk Swift se stal stále populárnějším pro vývoj iOS aplikací díky své jednoduchosti a efektivitě.

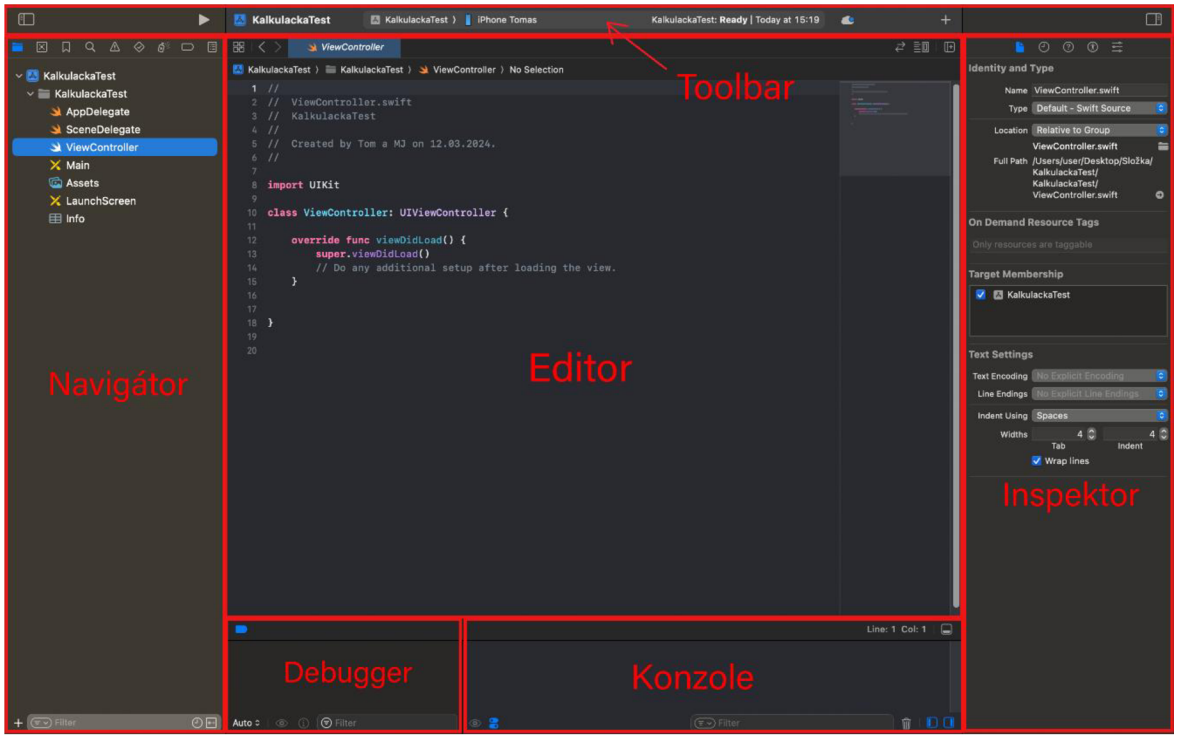
iOS významným hráčem na trhu mobilních operačních systémů a nabízí vývojářům a uživatelům unikátní kombinaci bezpečnosti, homogenity a integrace s ekosystémem Apple. Tato platforma zůstává důležitým faktorem v mobilním prostředí a stále přináší inovace a nové možnosti pro mobilní aplikace.

3.3 Xcode

Xcode, vyvinutý společností Apple, je hlavním nástrojem pro vývoj aplikací pro platformu iOS. Toto integrované vývojové prostředí nabízí velké množství nástrojů, které usnadňují proces vytváření, testování a distribuce aplikací pro iPhone, iPad a další zařízení od Apple. Jeho hlavní funkce zahrnují výkonný editor kódu, který podporuje programovací jazyky Swift a Objective-C, a vizuální nástroj Interface Builder pro návrh uživatelských rozhraní. Xcode také zahrnuje simulátor, který umožňuje vývojářům testovat aplikace na různých verzích zařízení. Důležitou součástí Xcode je také možnost ladění a profilování aplikací, což umožňuje identifikovat a opravit chyby a optimalizovat výkon. Xcode je také úzce propojen s App Store Connect, což usnadňuje publikaci aplikací v App Store a beta testování pomocí služby TestFlight. Celkově je Xcode nezbytným prostředím pro vývojáře, kteří chtějí vytvářet kvalitní aplikace pro ekosystém Apple, a poskytuje jim nástroje a zdroje pro úspěšný vývoj a nasazení aplikací pro iOS. [13]

3.3.1 Vývojové prostředí

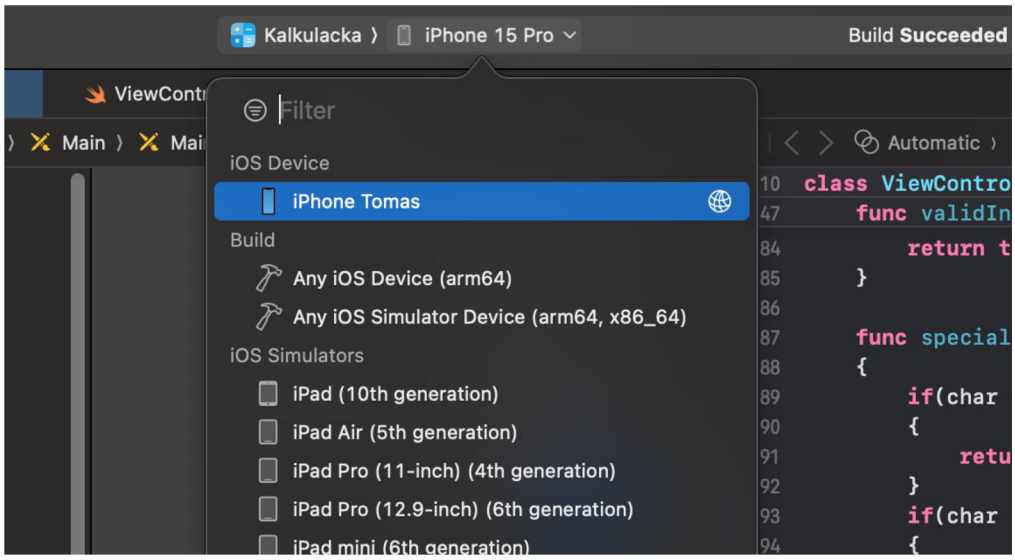
Vývojové prostředí Xcode (Obrázek 2) je složeno z pěti základních částí: Toolbar, Editor, Navigátor, Utility a Debug sekce. Debug sekce je dále rozdělena na Debugger a Konzoli. Zatímco v Utility sekci se nachází Inspektor.[13][24]



Obrázek 2 - vývojové prostředí

- **Toolbar**

Toolbar představuje klíčový prvek v ovládání celého prostředí programu. Jeho hlavní ovládací prvky pro vývoj aplikace se nacházejí na pravé straně, kde se zobrazují základní prvky pro správu pracovního prostoru. Zde se nacházejí ovládací prvky pro zobrazení/skrytí dalších oblastí prostředí. Tlačítka v levé části toolbaru umožňují spustit a zastavit program. Dále je možné zvolit, na jakém zařízení (buď připojeném extrení zařízení nebo simulátoru, viz obrázek 5) si přejeme aplikaci spustit. Střední část nás informuje o současných operacích prováděných s projektem, jako je kompilace a informace o průběhu a možných chybách. Zvláště stojí za zmínku tlačítko Assistant Editor, které je užitečné při vývoji aplikací pro iOS. Toto tlačítko umožňuje snadno spojovat soubory, které souvisejí a potřebují být upravovány současně. Když vývojář provádí změnu v jednom souboru, Assistant Editor automaticky otevře vedlejší soubor, který souvisí s aktuálně upravovaným souborem, což usnadňuje provádění souvisejících úprav v kódu. Střední část toolbaru obsahuje stavové okno, které poskytuje informace o aktuálních činnostech, které Xcode provádí. Vedle stavového okna najdeme tlačítko pro zastavení aktuálního procesu a tlačítko pro spuštění aktuálního projektu. [13]



Obrázek 3 - výběr virtuálního nebo externího zařízení

- **Editor**

V této části Xcode upoutává svou pozornost vývojář, neboť právě zde se vytvářejí klíčové funkce, které určují chování jednotlivých prvků aplikace. To znamená, že v této části se píší a opravují kódy aplikace. V horní části Editor Area najdeme nástroj zvaný Jumpbar, který slouží k rychlé navigaci mezi soubory, tvořícími projekt. Soubory lze snadno procházet použitím šipek umístěných v levé části Jumpbaru nebo kliknutím na aktuální název souboru v Jumpbaru. Když uživatel otevře rozbalovací nabídku, zobrazí se struktura souborů, což odpovídá pohledu v Navigátoru. Tímto způsobem si může uživatel jednoduše vybrat soubor, na kterém aktuálně pracuje. Editor Area může nabývat tří různých podob: Standard, Assistant, Version27. [13]

- **Navigator**

V navigační liště panelu Xcode můžete přepínat mezi několika typy navigace, zahrnujícími:

- **Project navigator**: Tento nástroj uspořádává soubory, které jsou spojeny s projektem, a umožňuje přidávání nových souborů a vytváření skupin pro organizaci projektu.
- **Symbol navigator**: Slouží k rychlé navigaci mezi symboly, jako jsou třídy, funkce a další prvky kódu.
- **Search navigator**: Pomocí tohoto nástroje můžete vyhledávat konkrétní výrazy ve všech souborech projektu a snadno přejít k výsledkům hledání.

- **Issue navigator**: Tato záložka zobrazuje seznam chyb a varování, které se vyskytly během kompilace nebo ladění aplikace. Chyby jsou označeny červeným výstražným trojúhelníkem.
- **Test navigator**: Umožňuje spravovat testovací sady a sledovat výsledky testů aplikace.
- **Debug navigator**: Slouží k monitorování průběhu ladění aplikace, což je důležitá činnost při odstraňování chyb a problémů.
- **Breakpoint navigator**: Pomocí tohoto nástroje se mohou nastavovat breakpointy v kódu, což umožní zastavit provádění aplikace v určitých bodech pro ladění.
- **Log navigator**: Zobrazuje logy a výstupy aplikace, což je užitečné při sledování chování aplikace a hledání problémů. [13]

- **Debugger**

V této oblasti Xcode se vývojář věnuje především ladění kódu, což je potřeba zejména v situacích, kdy některé části aplikace nefungují správně. Debug Area, umístěná v dolní části vývojového prostředí, slouží k této činnosti.

Debug Area poskytuje vývojáři možnost sledovat aktuální stav všech proměnných, atributů a dalších částí kódu během provádění aplikace. Před spuštěním aplikace může vývojář umístit tzv. breakpoint, což je místo v kódu, kde se provádění zastaví. Tímto způsobem může analyzovat aktuální stav všech atributů a proměnných, což je užitečné při hledání chyb.

Mezi sledovanými proměnnými patří lokální proměnné, argumenty, statické proměnné, globální proměnné, registry, instantní proměnné a výrazy. Vývojář postupuje tak, že vkládá breakpointy do kritických částí kódu a sleduje, zda se zde nevyskytují chyby. Když vývojář narazí na chybu, pokračuje v procesu debugování, aby identifikoval příčinu a nalezl možnosti opravy kódu. Debugovací prostor obsahuje řadu ovládacích prvků, které umožňují vývojáři krokovat aplikací, aktivovat nebo deaktivovat breakpointy a pracovat s vlákny programu. Apple na svých vývojářských stránkách nabízí návod, [13] jak nejlépe postupovat při debugování, což zahrnuje identifikaci chyb, lokalizaci místa v kódu, zkoumání stavu datových struktur a následnou opravu kódu.

- **Konzole**

V této části je k dispozici log programu, který poskytuje užitečné informace o běhu aplikace. Díky textovému poli „Filter“ je možné snadno vyhledávat specifické záznamy v logu, což usnadňuje diagnostiku a hledání konkrétních událostí. Pro udržení přehlednosti je možné prostřednictvím tlačítka s ikonou koše snadno odstranit již zobrazené záznamy, což pomáhá udržet log čistý a uspořádaný. [13]

- **Inspektor**

Toto je oblast v Xcodu, kde jsou zobrazeny informace o aktuálně vybraném souboru. V závislosti na typu souboru se zde mohou zobrazovat další podsekce. Například zde najdeme inspektor atributů (Attribute inspector), který umožňuje upravovat vlastnosti grafických prvků při modelování, jako je například barva pozadí, font textu atd. Další podsekcí může být "File inspector", kde jsou zobrazeny informace o právě vybraném souboru. [13]

3.3.2 Simulátor

Simulátor představuje důležitý prvek vývojového prostředí Xcode, umožňující programátorům provádět testování svých aplikací v raných stádiích vývoje. Toto testování zahrnuje kontrolu správného rozložení grafických prvků na různých zařízeních a ověření funkčnosti, kterou nelze zcela přesně simoulovat na reálném hardwaru. Díky simulátoru je možné aplikaci testovat na více zařízeních najednou, což zrychluje vývojový proces.
V Xcode lze nastavit simulaci různých zařízení, jako jsou různé modely iPhonů a iPadů, které jsou aktuálně dostupné na trhu. Programátor může také simulovat různé verze operačního systému iOS, což umožňuje testovat aplikace na různých verzích systému. Zároveň, pokud aplikace využívá GPS, je možné v simulátoru simulovat různé polohy GPS zařízení. Simulace nemusí být pouze zařízení se systémem iOS, ale také novější zařízení, jako jsou Apple Watch s WatchOS a Apple TV s tvOS, což rozšiřuje možnosti testování a zajišťuje kompatibilitu aplikace s různými platformami. Simulátor je tak neocenitelným nástrojem pro programátory iOS aplikací během vývoje a testování.[13]

3.3.3 Playgrounds

Funkce Playgrounds v programu Xcode poskytuje prostředí pro rychlé experimentování a vývoj v programovacím jazyce Swift. Původní verze této funkce byla oznámena a uvedena společností Apple Inc. 2. června 2014 během konference WWDC 2014. Playgrounds poskytují testovací prostředí, ve kterém se kód vývojáře vyhodnocuje v reálném čase. Mají schopnost vyhodnocovat a zobrazovat výsledky jednotlivých výrazů, jakmile jsou napsány (na stejné řádce nebo v boční liště), což poskytuje rychlou zpětnou vazbu programátorovi. Tento typ vývojového prostředí, známý jako read-eval-print loop (nebo REPL), je užitečný pro učení, experimentování a rychlé prototypování. Playgrounds byly používány společností Apple k publikaci průvodců a tutoriálů pro jazyk Swift, kde jsou výhody REPL zřejmé. Funkce Playgrounds byla vyvinuta oddělením Developer Tools ve společnosti Apple.[14] Podle Chrisa Lattnera, vynálezce programovacího jazyka Swift a vedoucího pracovníka a architekta v oddělení Developer Tools, byly Playgrounds "silně ovlivněny myšlenkami Brete Victora, Light Table a mnoha dalšími interaktivními systémy". Playgrounds byly oznámeny společností Apple. 2. června 2014 během konference WWDC 2014 jako součást Xcode 6 a byly uvedeny v září. V září 2016 byla aplikace Swift Playgrounds pro iPad (k dispozici také na macOS od února 2020) uvedena na trh, která tyto myšlenky začlenila do vzdělávacího nástroje. Funkce Playgrounds v Xcode nadále pokračovaly ve vývoji, přičemž nová funkcionalita postupného spouštění byla představena v Xcode 10 na WWDC 2018.[15]

3.4 Flutter

Flutter je inovativní open-source uživatelské rozhraní SDK (Software development kit, což je sada vývojových nástrojů) na vývoj softwaru vyvinutý společností Google, otevírá dveře k novým možnostem v mobilním vývoji. Tento nástroj se skládá ze dvou hlavních částí, SDK a samotného frameworku (vývojová platforma). SDK nabízí velké množství nástrojů pro vytváření, testování a optimalizaci multiplatformních mobilních aplikací. Samotný framework, napsaný v jazyce Dart, je základem pro psaní kódu aplikace a poskytuje bohatou sadu vestavěných widgetů a funkcí.[16]

Jednou z hlavních vlastností Flutteru je, že umí přímo převádět psaný kód do formátu, který může pochopit a spustit samotné zařízení, na kterém běží aplikace. To

znamená, že aplikace vytvořené pomocí Flutteru běží velmi rychle a plynule, protože se kód nepřekládá nebo interpretuje, ale spouští se přímo na zařízení. Tato schopnost také umožňuje vývojářům psát jeden univerzální kód, který lze použít na různých zařízeních, jako jsou telefony a tablety s operačním systémem Android nebo iOS. To znamená, že vývojáři nemusí psát oddělený kód pro každý typ zařízení, což ušetří čas a usnadní práci při vývoji aplikací. Flutter také nabízí flexibilitu v designu uživatelského rozhraní. Vývojáři mohou vybírat mezi jednotným designem pro všechny platformy nebo využít knihovny s nativními prvky. Základní jednotkou vývoje ve Flutteru jsou widgety, které tvoří tzv. Widget tree. Na rozdíl od jiných frameworků, kde jsou widgety spíše malými částmi grafického rozhraní, jsou ve Flutteru základní stavební bloky, ze kterých je postaveno celé uživatelské rozhraní. Vývojáři tak mají plnou kontrolu nad vzhledem a chováním svých aplikací. [17]

Uživatelské rozhraní Flutteru je k dispozici pro platformy Windows, macOS, Linux a Chrome OS. Tato multiplatformní dostupnost poskytuje široké možnosti vývoje bez ohledu na preferovaný operační systém. To napomáhá šíření a přístupnosti Flutteru a umožňuje vývojářům dosáhnout maximálního dosahu svých aplikací.

3.5 Objective-C

Objective-C je objektově orientovaný programovací jazyk, který vznikl v osmdesátých letech dvacátého století. Brad Cox a Tom Love, kteří stojí za jeho vznikem, se nechali inspirovat programovacím jazykem Smalltalk, který je založený na objektech, a rozhodli se rozšířit možnosti jazyka C o prvky objektově orientovaného programování. Tento krok umožnil vývojářům psát kód, který je strukturovaný do objektů, což usnadnilo vývoj komplexních softwarových aplikací.[18]

Objective-C získal širokou popularitu díky společnosti NeXT, která ho zvolila jako hlavní jazyk pro vývoj svého operačního systému NeXTSTEP, který byl poprvé představen v říjnu 1988. Tento systém představoval revoluční krok v počítačovém průmyslu a položil základy pro budoucí vývoj operačních systémů macOS a iOS, které si zachovaly Objective-C jako klíčový jazyk pro vývoj aplikací.[19]

Objective-C se od jiných programovacích jazyků odlišuje svou syntaxí a koncepty. Jedním z charakteristických rysů je dynamické odkazování a zpracování zpráv, což umožňuje objektům komunikovat mezi sebou za běhu programu. Tato vlastnost přispívá k flexibilitě a dynamice jazyka. I přes nástup modernějších jazyků jako Swift si Objective-C stále udržuje své místo v programátorské komunitě, zejména v oblasti vývoje aplikací pro platformy macOS a iOS. [28]

3.6 SWIFT

"Swift je nový programovací jazyk pro vývoj iOS a OS X aplikací. Swift bere to nejlepší z jazyka C a Objective-C, kam přidává nové funkce a vzory."[21]
Swift představuje revoluční programovací jazyk určený pro vývoj aplikací pro iOS a OS X. Zahrnuje ty nejlepší prvky programovacího jazyka C a Objective-C a přináší nové funkce a vzory," jehož vývoj začal v Apple v roce 2010 a byl veřejnosti představen na vývojářské konferenci v roce 2014, což bylo pro účastníky překvapením, protože existence tohoto nového programovacího jazyka byla utajena. [27]

Swift má za cíl nahradit dosud používaný Objective-C a v současné době může být používán společně s dalšími jazyky v kódu, jako jsou Objective-C, C++, a další. Jazyk Swift je plně kompatibilní s frameworky Cocoa a Cocoa Touch, což je sada knihoven, frameworků a API používaných pro vytváření aplikací pro Mac OS. Obsahuje mechanismy pro vykreslování komponent na obrazovku, práci s textem, ukládání a otevírání souborů, komunikaci s operačním systémem, nebo také komunikaci přes síť.[22] Swift byl navržen tak, aby vývojářům co nejvíce usnadnil tvorbu aplikací a čerpá ze silných stránek programovacích jazyků jako C++, C#, Ruby a dalších. Swift je flexibilní, což umožňuje vývojářům vytvářet aplikace nejen pro iOS, ale také pro MacOS, tvOS a WatchOS. [27]

Na konferenci pro vývojáře WWDC (Worldwide Developer Conference) v roce 2015 představila společnost Apple druhou verzi jazyka Swift. Kromě nových funkcí, které jazyk posílily, bylo také oznámeno, že Swift se stane open source, což byl pro Apple značný krok. Tato změna znamená, že Swift bude dostupný na různých platformách v budoucnosti. Otevření zdrojového kódu Swift proběhlo začátkem prosince 2015, kdy byly veškeré zdrojové kódy zveřejněny na portálu www.swift.org. Spolu s kódy byly uvolněny také knihovny pro Linux, což je důležité, protože Linux je dominantní platformou pro servery. Rozšířením Swift na tuto platformu budou moci vývojáři plynule přecházet mezi vývojem her a správou serverů. Mezi prvními velkými firmami, které využily otevřenost Swift, byla společnost IBM. IBM představila IBM Swift Sandbox, což je nástroj umožňující vývojářům programovat v prostředí webového prohlížeče. Tento nástroj umožňuje kódování na jedné straně a zobrazuje výstup aplikace na straně druhé.[8][23]

4 Praktická část

V praktické části této práce byly tyto teoretické znalosti uplatněny při vývoji konkrétní aplikace pro iOS. Vybralo se prostředí a jazyk pro vytváření aplikace Tato aplikace slouží jako praktický demonstrační projekt, který umožnil praktické ověření a uplatnění získaných teoretických dovedností a znalostí.

Pro vývoj aplikace v bakalářské práci byl zvolen jazyk Swift z několika důvodů. Za prvé, Swift je modernější a čistší jazyk než Objective-C, což znamená, že psaní kódu je rychlejší a snazší. Jeho způsob zpracování dat a proměnnými v kódu, přispívá k bezpečnosti a minimalizaci chyb. Dále, Swift je aktivně vyvíjen a podporován společností Apple, což znamená, že je pravděpodobnější, že bude podporován i v budoucích verzích operačních systémů a vývojových nástrojů.

4.1 Výběr prostředí

Vývojové prostředí Xcode a Flutter jsou často používané platformy pro tvorbu mobilních aplikací. Xcode je vývojové prostředí, které je vyvíjeno a udržováno společností Apple a je primárně určeno pro vývoj aplikací pro iOS a macOS. Na druhou stranu, Flutter je open-source framework vyvinutý společností Google, který umožňuje vývoj multiplatformních aplikací pro různé operační systémy včetně iOS, Androidu a dalších. Xcode je považován za lepší volbu z několika důvodů. Za prvé, Xcode je plně integrován s ekosystémem Apple a poskytuje přístup k všem nezbytným nástrojům, knihovnám a funkcím pro vývoj aplikací pro iOS a macOS. To znamená, že vývojáři mohou využívat všech výhod poskytovaných Apple, jako je například přístup k nejnovějším funkcím a technologiím. Dále, Xcode je optimalizován pro vývoj aplikací pro iOS a macOS, tudíž, poskytuje pokročilé nástroje pro ladění, testování a optimalizaci výkonu aplikací. Xcode také podporuje jazyk Swift, který je preferovaným programovacím jazykem pro vývoj aplikací pro platformy Apple. To umožňuje vývojářům psát čistý a efektivní kód s minimálními chybami a problémy.

Kromě toho, Xcode je oficiálně podporováno společností Apple, což zajišťuje stabilitu a dlouhodobou udržitelnost vývojového prostředí. To je klíčovým faktorem pro vývoj aplikací pro iOS a macOS, kde je důležité mít jistotu, že vývojové prostředí bude podporováno i v budoucích verzích operačních systémů a vývojových nástrojů.

Celkově lze tedy říci, že Xcode je lepší volbou pro vývoj aplikací pro iOS a macOS z důvodu své integrace s ekosystémem Apple, pokročilých nástrojů pro vývoj a optimalizaci výkonu aplikací a dlouhodobé podpory a udržitelnosti ze strany společnosti Apple, což přispěje k úspěšnému vývoji aplikace v rámci bakalářské práce.

4.2 Požadavky na software a hardware

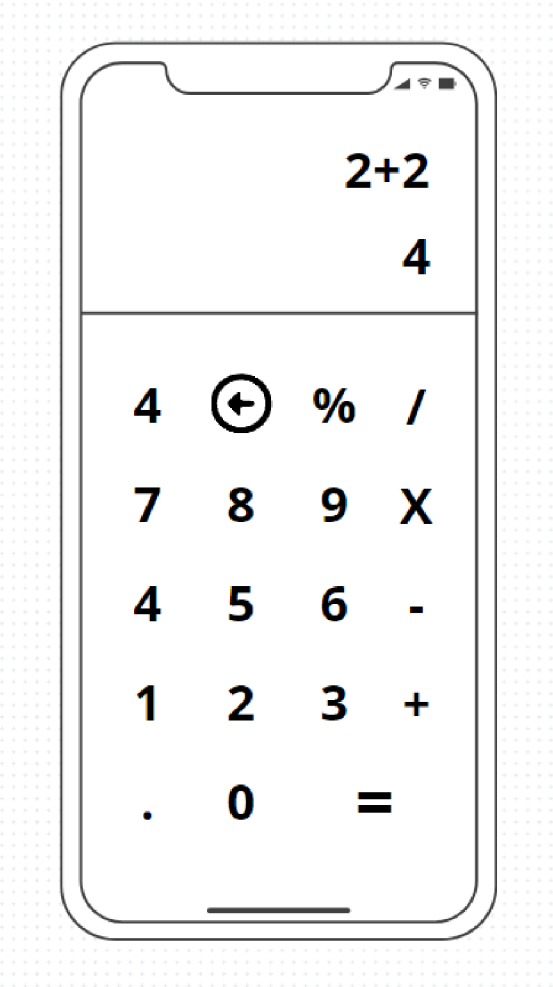
„Pokud máte počítač s macOS. Nemusíte už od začátku vývoje aplikace nic investovat“[23]

Pro vývoj aplikací a her je nezbytné mít k dispozici vhodný hardware, ale také správný software. Základem je vlastnit počítač s operačním systémem MacOS. Existuje několik možností, jaký hardware si vybrat, včetně přenosných počítačů jako Macbook Pro, stolních počítačů iMac nebo profesionálních pracovních stanic Mac Pro.[26] Osobně doporučuji začínajícím programátorům využívat iMac, který nabízí spojení výkonu s velkým displejem. Nicméně, vývoj aplikací není problém ani na přenosných zařízeních od Apple. Dokonce i zařízení s nižším hardwarovým výkonem jsou schopna zvládnout vývoj. Pokud je potřeba maximální výkon, zejména pro náročné grafické aplikace, je nejlepší volbou využití Mac Pro, i když je cena tohoto zařízení významnou překážkou pro méně zkušené vývojáře.

Pro vývoj aplikací pro platformy iOS je nezbytné využívat vývojového prostředí Xcode, které je dostupné zdarma v App Store.[26] Xcode nabízí širokou škálu nástrojů a funkcí pro tvorbu aplikací, včetně integrovaného editoru kódu, simulátorů pro testování aplikací a nástrojů pro ladění. Kromě toho je také důležité mít znalosti programovacího jazyka Swift, který je preferovaným jazykem pro vývoj aplikací pro platformu iOS. Pro tvorbu prototypu a designu aplikace je možné využít různé grafické nástroje, jako je Figma nebo Canva. Důležité je také sledovat aktuální trendy a doporučení společnosti Apple, pro zajištění kompatibility a optimálního výkonu aplikací.

4.3 Prototyp a návrh aplikace

V rámci praktické části bakalářské práce byl vytvořen prototyp aplikace (Obrázek 4) za využití programu Canva, což je nástroj pro tvorbu grafických designů a prototypů, bez nutnosti hlubších znalostí grafického designu. Tento prototyp sloužil jako základní návrh pro budoucí vývoj aplikace. Bylo rozhodnuto, že se bude pracovat na aplikaci kalkulačka. Prvním krokem bylo navržení uživatelského rozhraní pomocí Canva. Tento prototyp obsahuje základní funkce a prvky, které jsou typické pro kalkulačku, jako jsou číselné tlačítka, operátory a displej pro zobrazování výsledků výpočtů. V praxi je tato fáze prototypování klíčová pro pochopení požadavků uživatelů a pro optimalizaci uživatelského rozhraní před samotným vývojem aplikace.



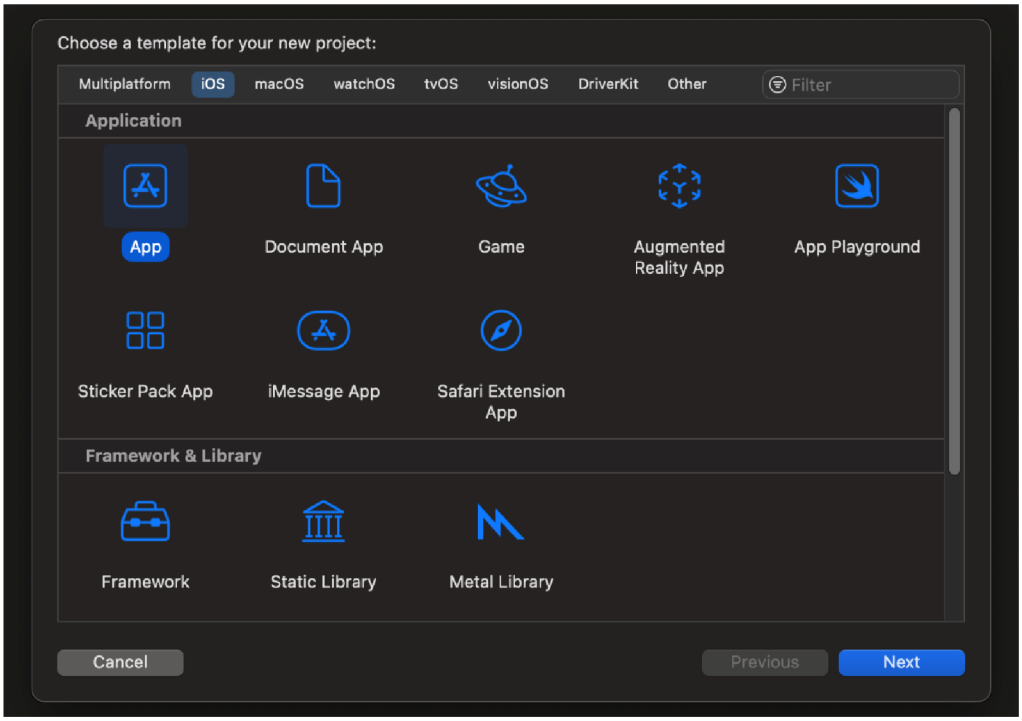
Obrázek 4 - prototyp aplikace

4.4 První spuštění

Po prvním spuštění vývojového prostředí Xcode se zobrazí okno (Obrázek 5), které nabízí možnost vytvoření nového projektu, nového playgroundu, stáhnutí existujícího projektu z repositáře nebo otevření již existujícího projektu z pravé části okna. [24]

Obrázek 5 - úvodní stránka Xcode

Po zvolení možnosti „Create new Xcode project“ se vytvoří nový projekt. V dalších oknech je třeba vybrat typ projektu (Obrázek 6). Šablony pro různé typy projektů jsou automaticky vygenerovány, což usnadní orientaci. Prozatím se vybere možnost „App“ v sekci „iOS“ a pro pokračování se klikne na tlačítko „Next“.[24]

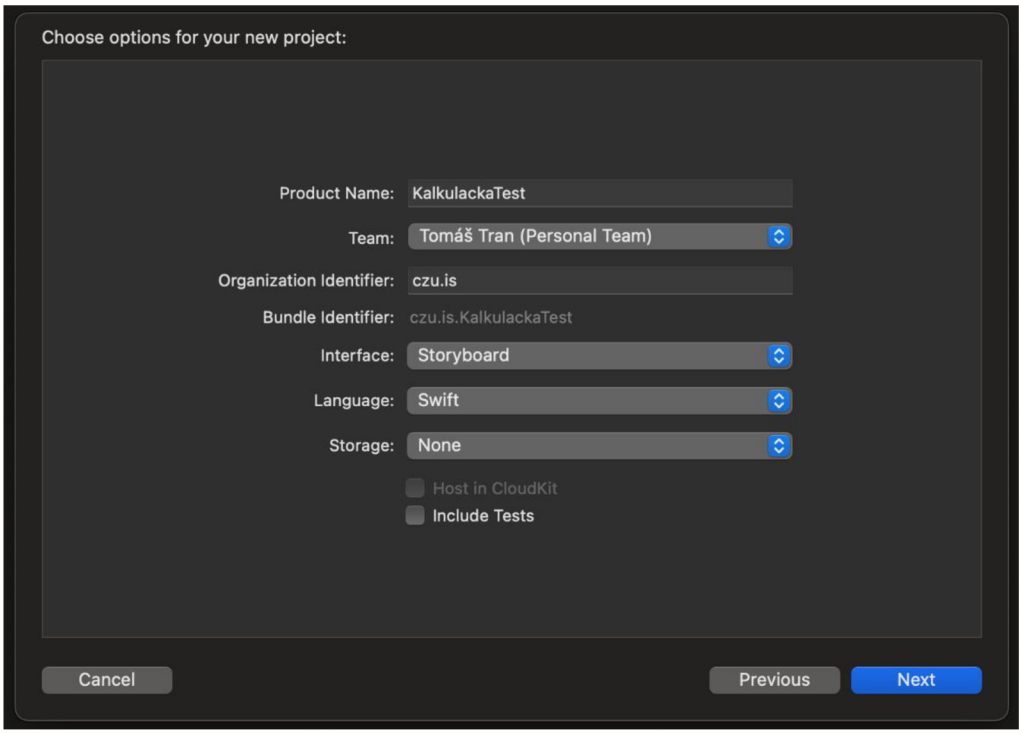


Obrázek 6 - výběr platformy

V dalším okně je potřeba vyplnit informace o projektu. „Product Name“ znamená jméno projektu, v tomto případě „Kalkulacka“. V poli „Organization Identifier“ je obvykle zapsáno reverzní doménové jméno, jako například „com.spolecnost“.[24]

Jako programovací jazyk byl vybrán Swift a v položce „Interface“ byla zvolena možnost „Storyboard“ (Obrázek 7). Storyboard je vizuální editor, který umožňuje uživatelům jednoduše navrhovat uživatelské rozhraní aplikace pomocí přetažení a upuštění prvků na obrazovku. Tímto způsobem lze snadno vytvářet různé obrazovky a interakce mezi nimi, což usnadňuje návrh a vývoj aplikace. [24][25]

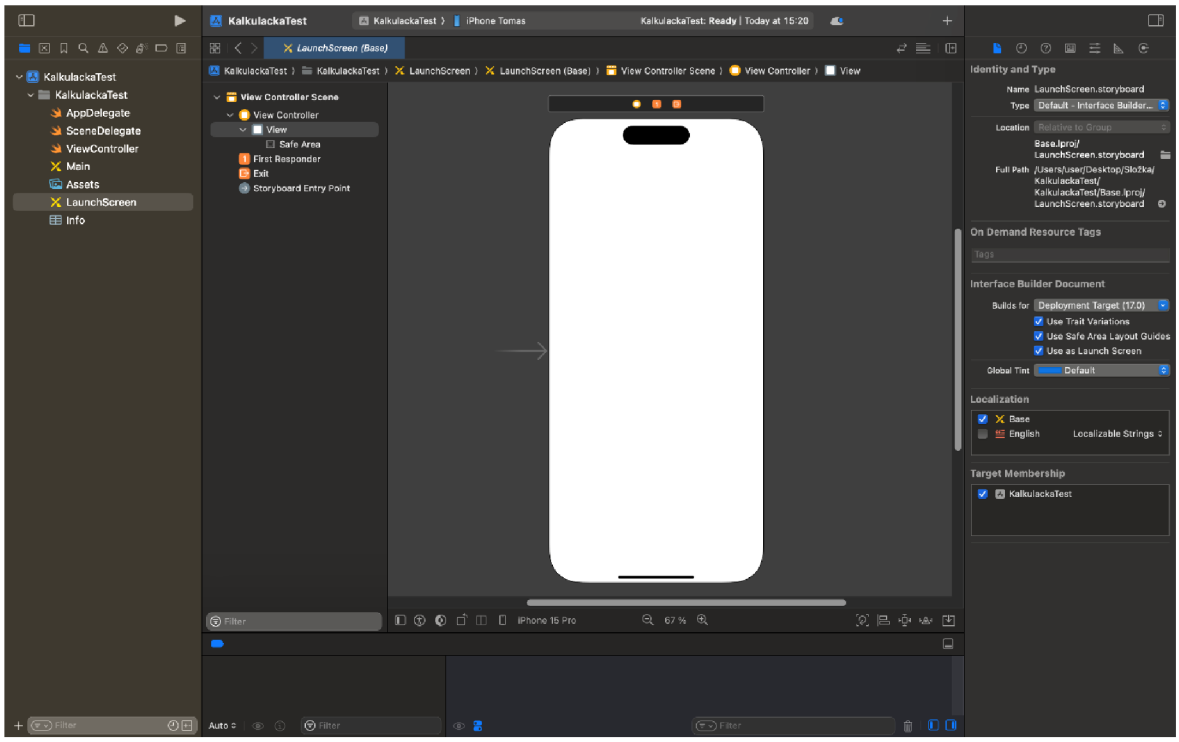
Díky použití storyboardu je možné vizuálně identifikovat a upravovat všechny obrazovky a přechody v aplikaci na jednom místě, což zlepšuje přehlednost a usnadňuje správu projektu. Tím se také zjednodušuje spolupráce mezi členy týmu, protože všichni mohou snadno vidět a upravovat uživatelské rozhraní. [24]



Obrázek 7 - nastavení vlastnosti projektu

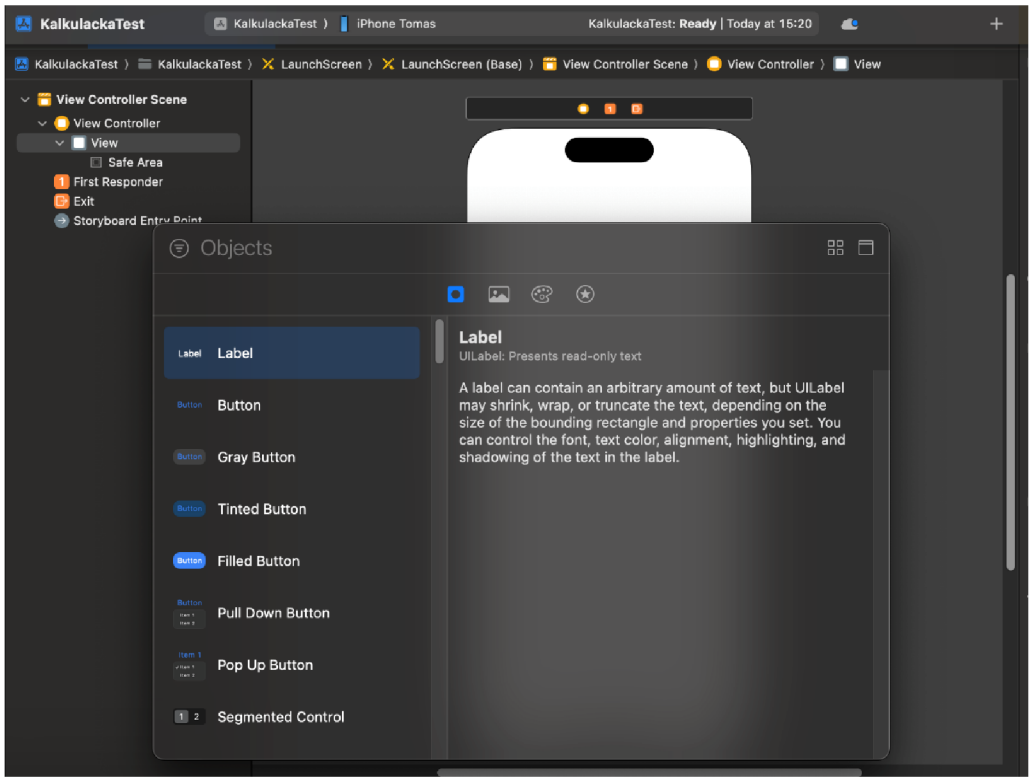
4.5 Interface builder

V Interface Builderu, integrovaném do vývojového prostředí Xcode, je možné vytvářet kompletní grafická rozhraní aplikací bez nutnosti psaní kódu. Tento nástroj je založen na pokročilém systému pozicování nazývaném Auto Layout, který umožňuje snadné navrhování rozložení prvků (Obrázek 8). Pomocí souborů s příponou .xib nebo .storyboard lze vizuálně modelovat jednotlivé obrazovky aplikace. XIB soubory jsou ve skutečnosti XML soubory, které popisují uspořádání prvků grafického rozhraní, zatímco storyboard je soubor obsahující několik XIB souborů, které určují navigaci mezi jednotlivými obrazovkami. [24]



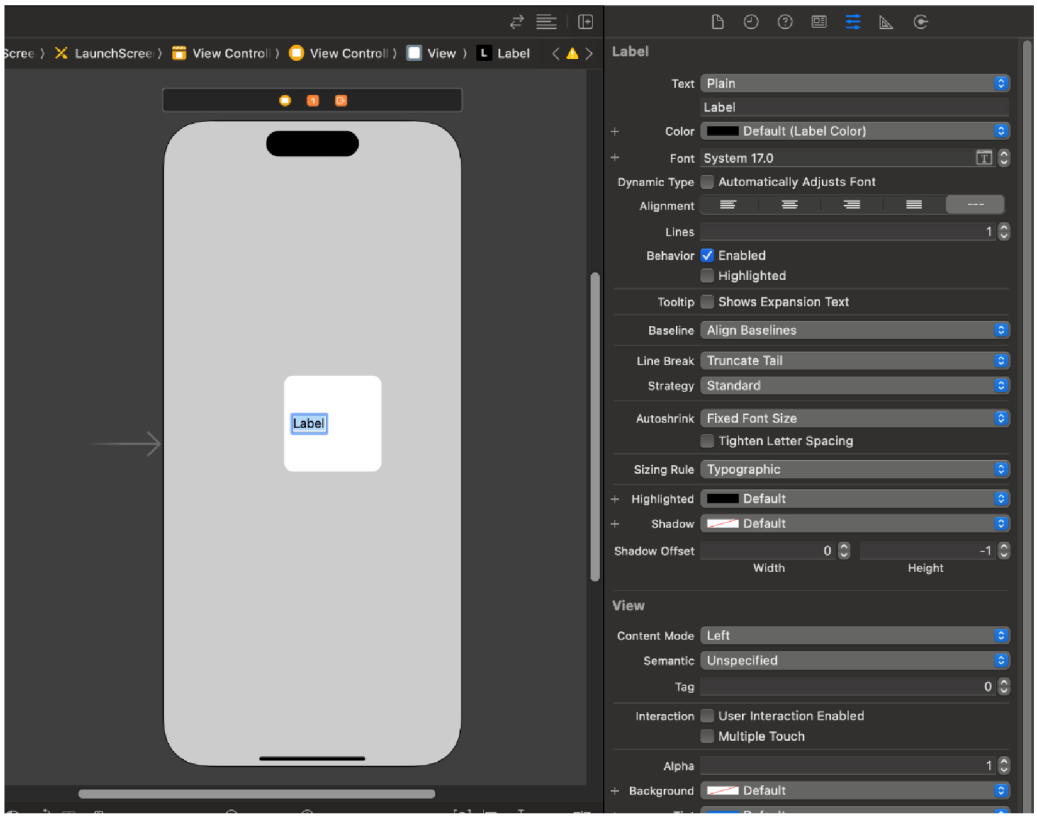
Obrázek 8 - storyboard

V Interface Builderu lze snadno manipulovat s grafickými prvky pomocí funkce přetažení z knihovny objektů, která obsahuje širokou škálu dostupných prvků, jako jsou tlačítka, textová pole nebo obrázky (Obrázek 9). Pro zajištění správného umístění a velikosti prvků je nezbytné přidat jim omezení (constraints), která určují jejich polohu a chování v závislosti na změnách velikosti obrazovky a zařízení. V levé části Interface Builderu se nachází struktura souboru, která poskytuje přehled o všech umístěných prvcích a jejich hierarchii. Zde lze také přidávat, odebírat nebo upravovat jednotlivé prvky. [24]



Obrázek 9 - knihovna prvků pro aplikaci

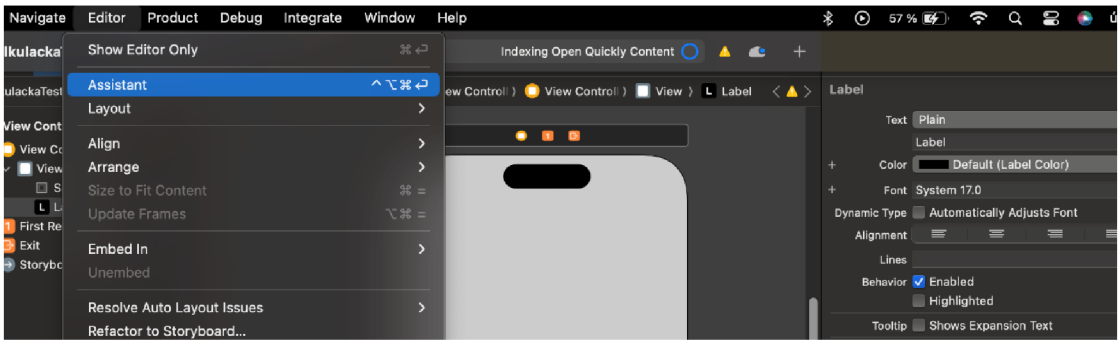
Hlavní pracovní plocha obsahuje samotné grafické prvky aplikace, kde je možné upravovat jejich vlastnosti a provádět detailní úpravy. V dolní části rozhraní se nachází přepínač zařízení, který umožňuje zobrazit náhled vybrané obrazovky na různých zařízeních a v různých orientacích. Pro detailní úpravy vlastností a chování jednotlivých prvků slouží Inspektor, který se nachází v pravé části Interface Builderu (Obrázek 10). Zde lze upravovat široké spektrum vlastností, jako jsou barva, velikost písma, zarovnání a další. Size Inspector pak poskytuje přehled o omezeních jednotlivých prvků a umožňuje jejich editaci včetně specifických nastavení pro různá zařízení. Celkově Interface Builder poskytuje uživatelsky přívětivé prostředí pro rychlé a efektivní vytváření komplexních grafických rozhraní bez nutnosti psaní kódu. [24]



Obrázek 10 - nastavení vzhledu prvků v inspektoru

Asistenční editor

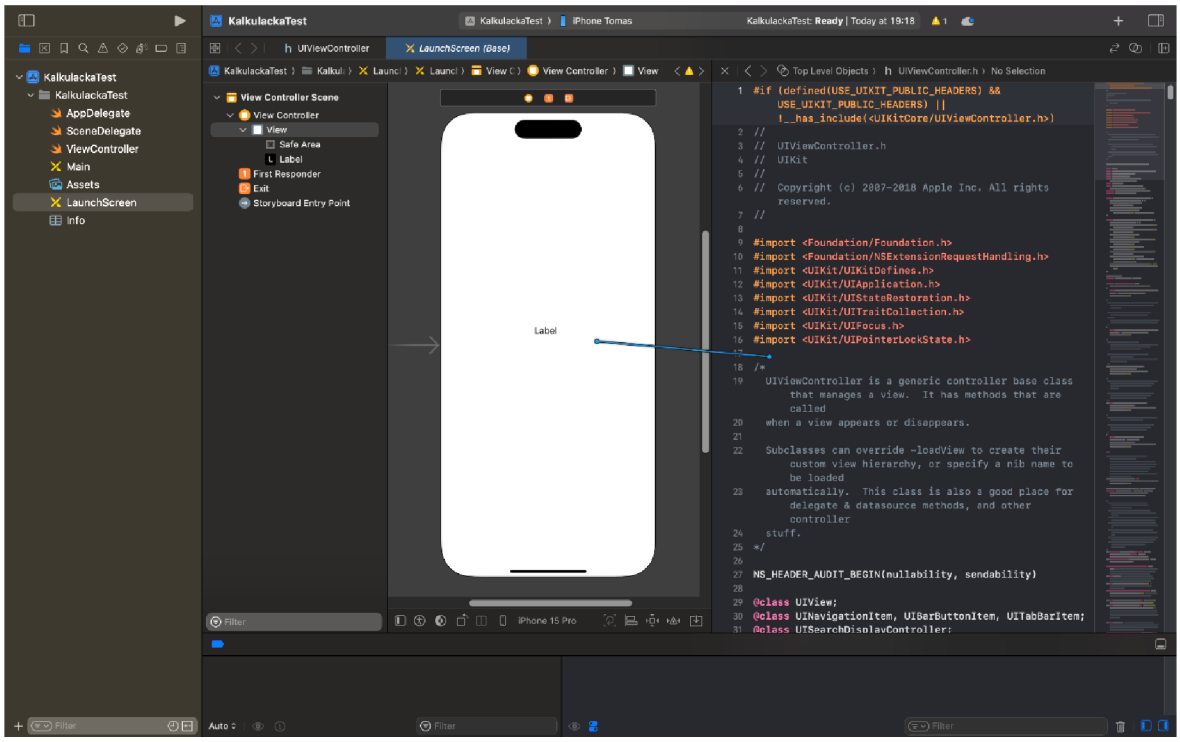
Asistenční editor lze aktivovat kliknutím na editor v horní části obrazovky a poté kliknout na tlačítko „Assistant“ (Obrázek 11).



Obrázek 11 - zapínání asistenčního editoru

Po aktivaci se hlavní editor rozdělí na dvě části: v levé části zůstane hlavní dokument, na kterém se aktuálně pracuje a v pravé části se zobrazí "inteligentní" část, kterou Xcode automaticky vybere jako nejvhodnější soubor k otevření. Tento asistenční editor se často

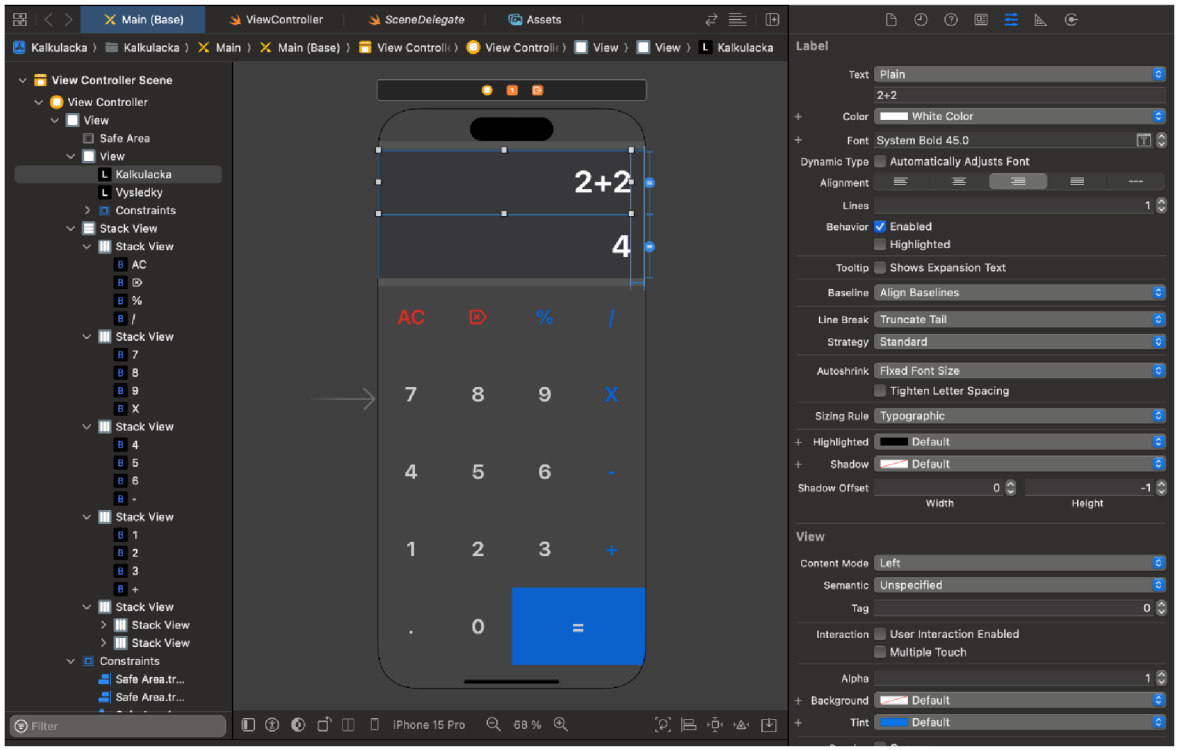
využívá při propojování prvků z Interface Builderu se souborem obsahujícím kód pro danou obrazovku (Obrázek 12). Například při potřebě přiřazení určité funkce k tlačítku se v asistenčním editoru zobrazí právě ten soubor, ve kterém můžete provést příslušné úpravy. [24]



Obrázek 12 - propojení prvků s kódy v editoru

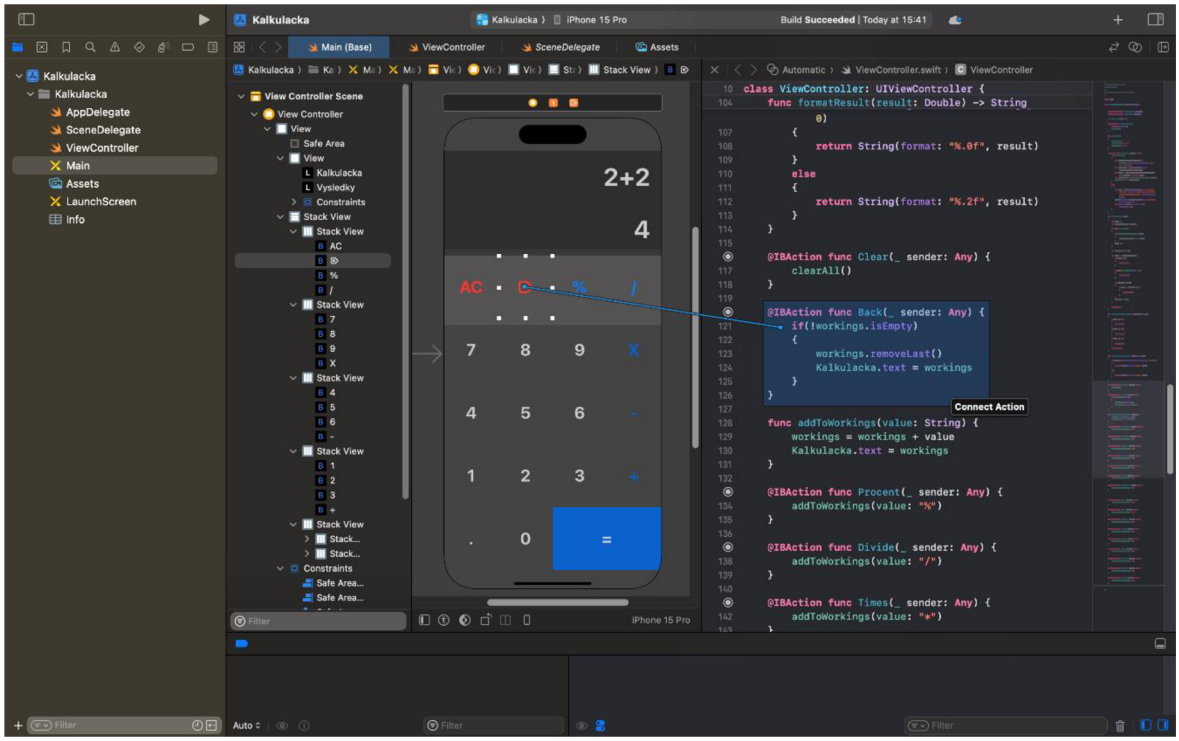
4.6 Vytváření aplikace

Prvním krokem při vytváření aplikace bylo využito nástroje Interface Builder v programu Xcode. Tento nástroj poskytoval uživatelsky přívětivé prostředí pro návrh grafického rozhraní bez nutnosti psaní kódu. S Interface Builderem bylo snadno přidáváno a uspořádáváno prvky, jako jsou tlačítka, textová pole a další, na obrazovky aplikace. Díky pokročilému systému pozicování Auto Layout bylo jednoduše definováno umístění a velikost prvků a zajištěno tak jejich správné zobrazení na různých zařízeních (Obrázek 13). Využití souborů .xib a .storyboard umožnilo vizuální modelování jednotlivých obrazovek aplikace a usnadnilo práci při návrhu uživatelského rozhraní. Nakonec byly pomocí Inspektoru detailně nastaveny vlastnosti jednotlivých prvků a přidána omezení (constraints), která určila jejich chování na různých zařízeních a při změnách velikosti obrazovky.



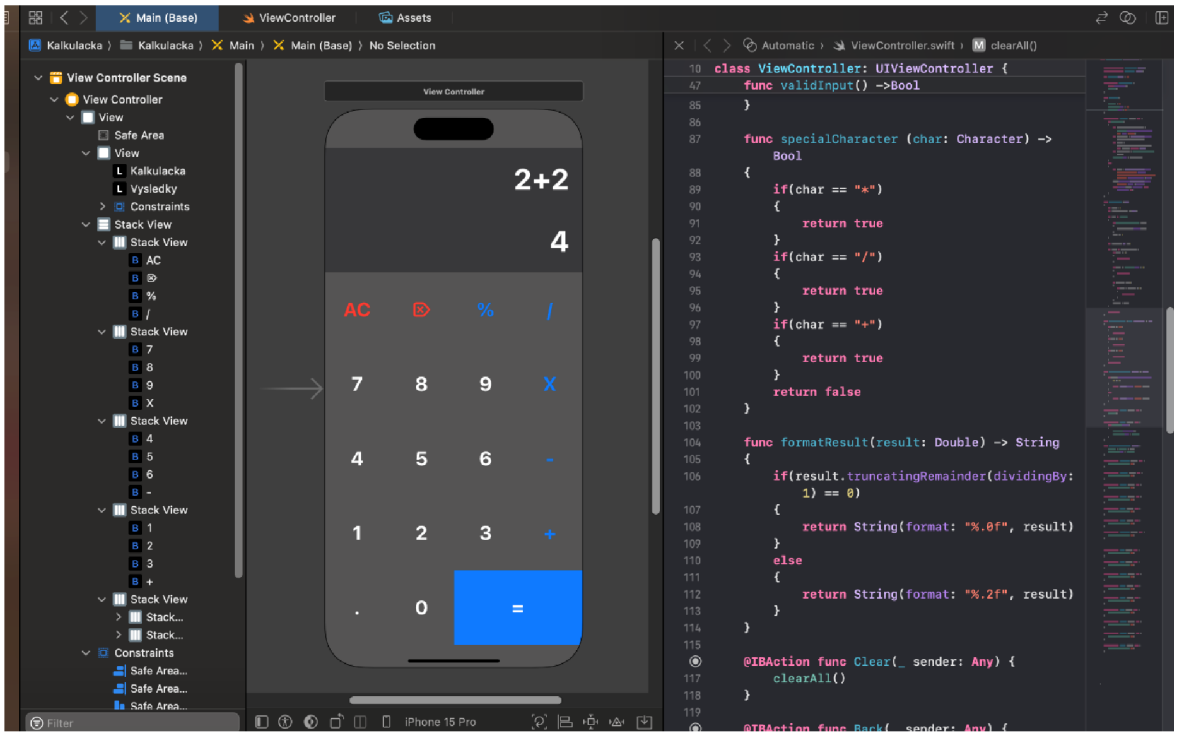
Obrázek 13 - vytváření UI (user interface) ve storyboardu

Po dokončení návrhu grafického rozhraní byl aktivován asistenční editor, který je integrován do prostředí programu Xcode. Tento krok umožnil propojení prvků vytvořených v Interface Builderu se souborem obsahujícím kód aplikace. Asistenční editor byl spuštěn jednoduchým kliknutím na tlačítko "Assistant" v horní části obrazovky. Po aktivaci se hlavní editor rozdělil na dvě části: v levé části zůstal hlavní dokument, na kterém se pracovalo v Interface Builderu, a v pravé části se zobrazil soubor, který Xcode automaticky vybral jako nejvhodnější pro otevření (Obrázek 14). Tím bylo umožněno snadné propojení grafických prvků s odpovídajícím kódem, což je obzvláště užitečné při přiřazování funkcí tlačítkům nebo manipulaci s daty z textových polí.



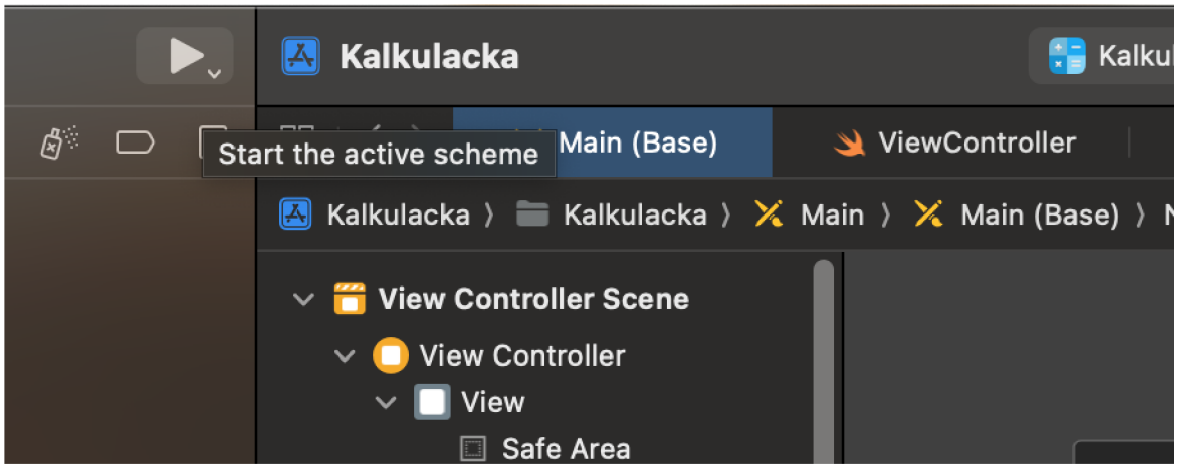
Obrázek 14 - propojení tlačítek s editorem pro správnou funkčnost kódů

Následně po propojení prvků byl kód aplikace vytvořen a otestován. Kód byl napsán v programovacím jazyce Swift a obsahoval funkce pro zpracování uživatelských vstupů, provádění požadovaných výpočtů a zobrazení výsledků (Obrázek 15). Při testování kódu byla zajištěna správná funkčnost všech tlačítek a textových polí aplikace. Kód obsahoval také validaci uživatelského vstupu, aby bylo zajištěno, že kalkulačka dokáže zpracovat pouze platné vstupy a vyhodit chybovou hlášku v případě neplatného vstupu. Dále byly implementovány funkce pro manipulaci s vstupními daty, jako je mazání posledního znaku, vymazání všech dat a přidání procentuálního symbolu. Celkově byl kód aplikace pečlivě vytvořen a otestován, aby byla zajištěna jeho správná funkčnost a uživatelská přívětivost.



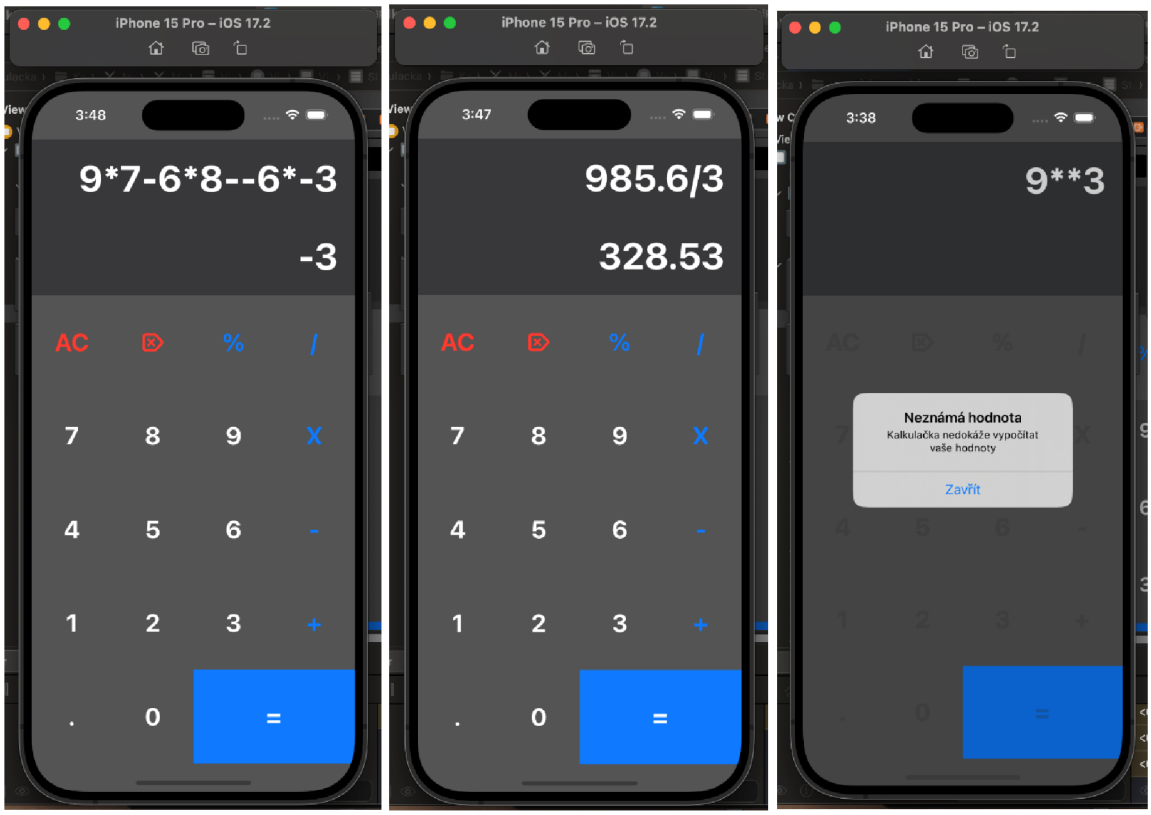
Obrázek 15 - psaní a testování kódů

Pro zapnutí simulace, tudíž testování funkčnosti vytvořené aplikaci na simulovaném virtuálním zařízení. K tomu bylo nejprve nutné spustit simulaci pomocí tlačítka "Start the active scheme" v toolbaru Xcode (Obrázek 16). Po spuštění simulace se zobrazí obrazovka zařízení, na které se rozhodlo testovat (jednálo se o zařízení iPhone 15 pro, viz Obrázek)



Obrázek 16 - zapínání simulace

Testování začalo ověřením základních aritmetických operací, jako je sčítání, odčítání, násobení a dělení, a to jak s celými čísly, tak s desetinnými. Byly provedeny různé kombinace vstupních hodnot a operací, aby byla zajištěna správná funkčnost kalkulačky i v nejkomplikovanějších situacích (Obrázek 17). Dále byla testována funkce pro manipulaci s vstupními daty, jako je mazání posledního znaku, vymazání všech dat a přidání procentuálního symbolu. Bylo ověřeno, že tyto operace fungují správně a bezchybně. Během testování byla také zkontrolována správná funkce zobrazování výsledků a chybových hlášek. Bylo zajištěno, že výsledky jsou přesné a že aplikace adekvátně reaguje na neplatné vstupy uživatele, například vyhazuje chybové hlášky v případě neplatných vstupů. Celkově bylo testování prováděno důkladně a systematicky, aby byla zajištěna správná funkčnost všech operací kalkulačky. Po úspěšném absolvování testování byla aplikace připravena k dalšímu vývoji a nasazení na reálná zařízení.



Obrázek 17 - testování aplikace

5 Zhodnocení a doporučení

Vývojové prostředí Xcode bylo vyhodnoceno jako lepší volba než Flutter z několika důvodů. Za prvé, Xcode poskytuje komplexní a přehledné prostředí pro vývoj aplikací pro iOS a macOS. Jeho integrace s ekosystémem Apple umožňuje přístup k nejnovějším funkcím a technologiím, což usnadňuje vytváření aplikací v souladu s aktuálními standardy a požadavky. Dále, Xcode je optimalizován pro vývoj aplikací pro iOS a macOS, což zahrnuje pokročilé nástroje pro ladění, testování a optimalizaci výkonu aplikací. Tento faktor přispívá k efektivnímu procesu vývoje a zvyšuje produktivitu vývojářů. Interface Builder v Xcode usnadňuje tvorbu uživatelského rozhraní aplikací pomocí intuitivního grafického editoru. To umožňuje rychlé itekrace a testování designových konceptů bez nutnosti ručního psaní kódu. Navíc, Xcode je oficiálně podporován společností Apple, což zaručuje stabilitu a dlouhodobou udržitelnost vývojového prostředí. To je důležité pro dlouhodobý vývoj aplikací pro iOS a macOS, kde je klíčové mít jistotu, že vývojové prostředí bude podporováno i v budoucích verzích operačních systémů a vývojových nástrojů. Celkově lze tedy říci, že výběr Xcode jako vývojového prostředí pro tvorbu aplikace byl založen na jeho integraci s ekosystémem Apple, pokročilých nástrojích pro vývoj a optimalizaci výkonu aplikací a dlouhodobé podpoře a udržitelnosti ze strany společnosti Apple. To přispělo k úspěšnému vývoji aplikace v praktické části a v rámci bakalářské práce. Aplikace prokázala spolehlivou funkčnost při testování základních aritmetických operací a manipulaci se vstupními daty. Zobrazování výsledků a chybových hlášek funguje přesně a aplikace adekvátně reaguje na uživatelské vstupy.

Flutter zaostává v několika ohledech oproti Xcode a vývojovému prostředí pro iOS a macOS. Za prvé, i když Flutter umožňuje vývoj multiplatformních aplikací pro různé operační systémy, včetně iOS a Androidu, není tak těsně propojen s ekosystémem Apple jako Xcode. To znamená, že vývojáři mohou mít omezený přístup k některým funkcím a technologiím specifickým pro zařízení Apple, což může ovlivnit výslednou kvalitu aplikace. Dalším omezením Flutteru je obtížnost. I když Flutter nabízí atraktivní možnosti pro vývoj aplikací pomocí jednoho kódu pro různé platformy, může to být pro začátečníky náročné. Na rozdíl od toho je Xcode navržen tak, aby usnadňoval vývoj aplikací pro iOS a macOS s jasným rozhraním a integrovanými nástroji pro tvorbu, testování a ladění.

Swift podporuje funkcionální programování a poskytuje širokou škálu nástrojů a knihoven, což usnadňuje tvorbu komplexních aplikací s čistým a efektivním kódem. Díky

svému aktivnímu vývoji a podpoře ze strany Apple je Swift pravděpodobnější, že bude podporován i v budoucích verzích operačních systémů a vývojových prostředí. Swift se také vyznačuje vysokou výkonností a efektivitou, což je důležité pro mobilní aplikace, které musí být rychlé a spolehlivé. Jeho optimalizace a moderní vlastnosti umožňují vývojářům psát čistý a efektivní kód s minimálními chybami a problémy. Další výhodou Swiftu je jeho široká kompatibilita s existujícími knihovnami a frameworky pro vývoj aplikací pro platformy Apple, což usnadňuje integraci a rozšíření funkcionalit aplikací. Celkově lze říci, že Swift je vynikajícím volbou pro vývoj mobilních aplikací pro iOS a macOS, který nabízí vysokou produktivitu, bezpečnost a efektivitu.

Swift přináší několik významných vylepšení oproti Objective-C. Jedním z hlavních rozdílů je modernější syntaxe, která je čistší a lépe čitelná. Díky tomu je psaní kódu v Swiftu rychlejší a snazší, což zvyšuje produktivitu vývojářů. Je navržen tak, aby byl rychlejší a výkonnější než Objective-C. Díky své aktivní podpoře a integraci s ekosystémem Apple je pravděpodobnější, že Swift bude podporován i v budoucích verzích operačních systémů a vývojových nástrojů.

Samostatně se doporučuje využít kombinaci vývojového prostředí Xcode a programovacího jazyka Swift pro vývoj mobilních aplikací pro platformy iOS, kvůli maximální kompatibilitě. Xcode poskytuje uživatelsky přívětivé prostředí s integrovanými nástroji pro ladění, testování a optimalizaci výkonu aplikací, což usnadňuje a urychluje celý proces vývoje. Jazyk Swift je moderní, čistý a bezpečný, což zvyšuje efektivitu psaní kódu a minimalizuje možnost chyb. Díky aktivní podpoře od společnosti Apple je Swift stabilní volbou pro dlouhodobý vývoj aplikací, s pravděpodobností budoucí podpory v dalších verzích operačních systémů. Kombinace Xcode a Swiftu tedy představuje solidní základ pro úspěšný vývoj aplikací pro platformy Apple v rámci bakalářské práce.

iOS byl vybrán jako cílová platforma pro tuto aplikaci kvůli několika faktorům. Prvním důvodem je vyšší kupní síla uživatelů iOS, což může vést k vyššímu potenciálu generování příjmů. Dále je iOS známý svou jednotnou uživatelskou zkušeností a přísnějšími bezpečnostními opatřeními, což může být důležité v některých oblastech, jako jsou finance či zdravotnictví. Vývoj v jazyce Swift a v prostředí Xcode je efektivní a umožňuje rychlý vývoj s minimálními chybami. Android je více fragmentovaný a méně konzistentní, ale nabízí širší dosah a možnost personalizace. Celkově je volba mezi iOS a Androidem závislá na konkrétních potřebách a preferencích vývojáře a cílového publika aplikace.

6 Závěr

Hlavním cílem práce bylo poskytnout ucelený přehled prostředí a nástrojů potřebných pro vývoj aplikací pro operační systém iOS. Tento cíl byl rozložen na dílčí úlohy, včetně prozkoumání a hodnocení operačních systémů, specifikace softwarových a hardwarových požadavků se zaměřením na iOS, identifikace klíčových nástrojů pro vývoj aplikací, a sestavení dokumentace a formulace závěrů a doporučení.

Metodika práce vycházela ze systematického přístupu, který zahrnoval detailní studium relevantních informačních zdrojů a vývoj teoretického rámce.

V praktické práci byla získána celá řada cenných zkušeností a dovedností v oblasti vývoje aplikací pro iOS pomocí platformy Xcode. Byla úspěšně vytvořena aplikace, která slouží jako praktický demonstrační projekt pro aplikace pro iOS. Celý proces vývoje byl pečlivě prováděn s ohledem na systematický přístup a postupnou validaci dosažených výsledků. Při tvorbě prototypu byl klíčovým krokem detailní návrh uživatelského rozhraní, který byl následně realizován v prostředí Storyboard. Tento nástroj umožnil intuitivní tvorbu a vizualizaci UI designu bez vyšších znalostí o grafickém designu, což výrazně urychlilo proces vytváření aplikace. Díky prostředí Xcode bylo možné efektivně ladit a testovat aplikaci v simulátoru, což umožnilo identifikovat a odstranit chyby ještě před nasazením do reálného prostředí. Proces vývoje v Xcode byl plynulý a přehledný, což umožnilo snadné orientování a rychlou iteraci v jednotlivých fázích vývoje. Důraz byl kladen na detailní studium a porozumění jednotlivým funkcím a možnostem prostředí Xcode, čímž byla zajištěna efektivní a kvalitní práce. Kromě toho, prostřednictvím praktické části práce jsem získal ucelený obraz o procesu vývoje aplikací pro iOS, což mi poskytlo klíčové znalosti a dovednosti pro budoucí práci v oboru mobilního vývoje. Zkušenosti získané během práce v Xcode budou nepochybně cenným přínosem pro mé další profesní úsilí a rozvoj v oblasti vývoje aplikací pro iOS.

Celkově lze tuto bakalářskou práci charakterizovat jako komplexní studii vývoje mobilních aplikací pro iOS, kde teoretická část poskytuje důležitý teoretický základ a praktická část umožňuje aplikaci těchto znalostí v reálném prostředí. Tímto způsobem přispívá k lepšímu porozumění procesu vývoje aplikací pro iOS a jeho praktickým aspektům.

7 Seznam použitých zdrojů

[1] IEEE. Big Data Analytics for User-Activity Analysis and User-Anomaly Detection in Mobile Wireless Network. IEEEXPLORE [online]. 2017 [cit. 2024-01-06]. Dostupné z: https://ieeexplore.ieee.org/abstract/document/7811244

[2] APPLE. Xcode. Developer [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://developer.apple.com/xcode/

[3] Welcome to the Android Open Source Project. Android Open Source Project [online]. Open Handset Alliance [cit. 2024-01-06]. Dostupné z: https://source.android.com/

[4] APPLE. IOS 17. IOS 17 [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://www.apple.com/cz/ios/ios-17/

[5] STATISTA. Global market share held by mobile operating systems since 2009. Statista [online]. 2009 [cit. 2024-01-06]. Dostupné z: https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/

[6] STATCOUNTER. Mobile Operating System Market Share Worldwide [online]. 2022. [cit. 2024-1-06]. Dostupné z: https://gs.statcounter.com/os-market-share/mobile/worldwide.

[7] GOOGLE. Android Open Source Project. Source [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://developer.android.com/

[8] GOOGLE. Android is made for you. Accessibility [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://www.android.com/accessibility/

[9] JavaTpoint. Android Versions [online]. 2021. [cit. 2024-01-06]. Dostupné z: https://www.javatpoint.com/android-versions.

[10] GOOGLE. The Android Show. Developer [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://developer.android.com/

[11] BRITANNICA. IOS. VOLLE, Adam. Britannica [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://www.britannica.com/topic/iOS

[12] APPLE. Apple Platform Security. Apple support [online]. 2022 [cit. 2024-01-06]. Dostupné z: https://support.apple.com/cs-cz/guide/security/welcome/web

[13] APPLE. Documentation Xcode. Developer [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://developer.apple.com/documentation/xcode

[14] UELMEN, J. A Simple Swift Tutorial: Playgrounds and Fundamentals. 2015 [online]

udacity.com. [cit. 2024-01-06]. Dostupné z: http://blog.udacity.com/2015/03/learn-swift-tutorialfundamentals.html

[15] 9TO5MAC. Apple announces Swift Playgrounds for iPad at WWDC, public release in fall. MAYO, Benjamin. 9to5mac [online]. 2016 [cit. 2024-01-06]. Dostupné z: https://9to5mac.com/2016/06/13/apple-announces-swift-playgrounds-for-ipad/

[16] GOOGLE. Flutter [online]. 2017 [cit. 2024-01-06]. Dostupné z: https://docs.flutter.dev/ui

[17] BIESSEK, Alessandro. Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2. 2. Packt Publishing, 2019. ISBN 978-1788996082.

[18] The History of Objective-C. In: Techotopia [online]. 2011 [cit. 2024-01-06]. Dostupné z: http://www.techotopia.com/index.php/The_History_of_Objective-C

[19] SINGH, Amit. A Brief History of Mac OS X. In: OSXBook [online]. 2003 [cit. 2024-01-06]. Dostupné z: http://osxbook.com/book/bonus/ancient/whatismacosx/history.html

[20] BIANCUZZI, Federico. Masterminds of Programming: Conversations with the Creators of Major Programming Languages (Theory in Practice (O'Reilly)). 1. O'Reilly Media, 2009. ISBN 978-0-596-51517-1.

[21] MALIK, Waqar. Learn Swift 2 on the Mac: For OS X and iOS. 2. Apress, 2015. ISBN 978-1484216286.

[22] APPLE. What Is Cocoa? Documentation Archive [online]. 2013 [cit. 2024-01-06]. Dostupné z: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html

[23] Vývoj aplikací pro iOS. 2. Brno: Computer Press, 2018. ISBN 978-80-251-4942-3.

[24] ARPIT, Kulsreshtha. IOS 17 App Development for Beginners. 1. Londýn: BPB Publications, 2018. ISBN 978-93-55515-858.

[25] KEUR, Christian. IOS Programming: The Big Nerd Ranch Guide. 4. Addison-Wesley Professional, 2014. ISBN 978-0321942050.

[26] APPLE. Xcode. App Store [online]. 2024 [cit. 2024-01-06]. Dostupné z: https://apps.apple.com/us/app/xcode/id497799835?mt=12

[27] The Swift Programming Language (Swift 5.7) [online]. 1. Apple, 2014 [cit. 2024-01-06]. Dostupné z:

https://www.appsdissected.com/wp-content/uploads/2022/11/SwiftProgrammingLanguage57.pdf

[28] TIMMER, John. A fast look at Swift, Apple's new programming language [online]. [cit. 2024-01-06]. A fast look at Swift, Apple’s new programming language Dostupné online z: https://arstechnica.com/gadgets/2014/06/a-fast-look-at-swift-apples-new-programming-language/

[29] GOOGLE. Install Flutter. Flutter [online]. 2017 [cit. 2024-01-06]. Dostupné z: https://docs.flutter.dev/get-started/install

8 Seznam obrázků

Obrázek 1 - zastoupení mobilních operačních systémů na trhu v letech 2009-2023 [5] 12
Obrázek 2 - vývojové prostředí 16
Obrázek 3 - výběr virtuálního nebo externího zařízení 17
Obrázek 4 - prototyp aplikace 26
Obrázek 5 - úvodní stránka Xcode 27
Obrázek 6 - výběr platformy 28
Obrázek 7 - nastavení vlastnosti projektu 29
Obrázek 8 - storyboard 30
Obrázek 9 - knihovna prvků pro aplikaci 31
Obrázek 10 - nastavení vzhledu prvků v inspektoru 32
Obrázek 11 - zapínání asistenčního editoru 32
Obrázek 12 - propojení prvků s kódy v editoru 33
Obrázek 13 - vytváření UI (user interface) ve storyboardu 34
Obrázek 14 - propojení tlačítek s editorem pro správnou funkčnost kódů 35
Obrázek 15 - psaní a testování kódů 36
Obrázek 16 - zapínání simulace 36
Obrázek 17 - testování aplikace 37

9 Přílohy

V příloze jsou soubor s vytvořeným programem (zdrojové kódy).