MASARYKOVA UNIVERZITA
FAKULTA INFORMATIKY

# VÝVOJ MOBILNÍ APLIKACE THE BMX BUDDY

DIPLOMOVÁ PRÁCE

Bc. Tomáš Carda

Brno
podzim 2015

## Prohlášení

Prohlašuji, že tato práce je mým původním autorským dílem, které jsem vypracoval samostatně. Všechny zdroje, prameny a literaturu, které jsem při vypracování používal nebo z nich čerpal, v práci řádně cituji s uvedením úplného odkazu na příslušný zdroj.

Vedoucí práce: MgA. Kateřina Spáčilová

## Poděkování

Děkuji paní MgA. Kateřině Spáčilové za ochotné odborné vedení práce, svým rodičům, kteří ve mne vložili důvěru a své přítelkyni, která mi byla oporou při tvorbě této práce.

## Shrnutí

Práce seznamuje s procesem vývoje mobilní aplikace The BMX Buddy pro zařízení s operačním systémem iOS. Ukazuje pracovní postupy od prvotní myšlenky, přes prototypování, vývoj samotné aplikace v prostředí Xcode, zveřejnění aplikace na digitální distribuční platformě Appstore a dále ukazuje postup při tvorbě propagační kampaně pro tuto aplikaci.

## Klíčová slova

Mobilní aplikace, The BMX Buddy, iOS, Xcode, prototypování, vývoj, audiovizuální dílo, After Effects, Cinema 4D, digitální postprodukce, compositing.

# Obsah


**1. Úvod** 9

**2. Technologie iOS** 11

2.1 Architektura iOS 11

2.1.1 Vrstva Cocoa Touch 12

2.1.2 Vrstva Media 12

2.1.3 Vrstva Core Services 13

2.1.4 Vrstva Core OS 13

**3. Od myšlenky k softwaru** 14

3.1 Dokumentace 14

3.1.1 The BMX Buddy - základní popis 15

3.1.2 The BMX Buddy - seznam požadavků 16

3.1.3 The BMX Buddy - dokumentace architektury 17

3.2 Sketchování 18

3.2.1 Sketchovací nástroje 18

3.2.2 Postup při sketchování 19

3.2.3 The BMX Buddy - sketch 20

3.3 Wireframing 21

3.3.1 Nástroje pro přípravu wireframů 21

3.3.2 The BMX Buddy - wireframe 22

3.4 Prototypování 24

3.4.1 Prototypovací nástroje 24

3.4.2 Prototypování pomocí aplikace POP 25

3.5 Mockupy 26

3.5.1 Nástroje pro přípravu mockupů 26

3.5.2 Příprava mockupu za pomocí Photoshop a Scala Preview 27

**4. Vývoj aplikace** 28
4.1 iOS vývojář 28
4.2 Xcode 29
4.2.1 Pracovní prostředí Xcode 29
4.2.2 Navigace v projektu 31
4.2.3 Práce s kódem 31
4.2.4 Kompilace kódu 32
4.2.5 iOS Simulator 32
4.3 Swift 33
4.3.1 Konstanty a proměnné 33
4.3.2 Množinové typy 34
4.3.3 Kontrola nad během programu 34
4.3.3.1 For cykly 34
4.3.3.2 While cykly 35
4.3.3.3 Podmíněné výrazy 35
4.3.4 Funkce 36
4.4 Model-View-Controller Design 37
4.5 The BMX Buddy - View 38
4.5.1 Login View 39
4.5.2 Map View 39
4.5.3 Spot Detail View 40
4.5.4 New Spot View 40
4.5.5 Settings View 41
4.6 The BMX Buddy - Model 42
4.6.1 Parse 42
4.6.2 Datový model The BMX Buddy 44

4.7 The BMX Buddy - Controller 45
4.7.1 StartupViewController.swift 45
4.7.2 MapViewController.swift 47
4.7.3 SpotDetailController.swift 49
4.7.4 NewSpotController.swift 49
4.7.5 SettingsViewController.swift 51
4.8 Umístění aplikce na App Store 52
4.8.1 Certifikáty 52
4.8.2 ID aplikace a Provisioning Profile 53
4.8.3 Založení aplikace v iTunes Connect 53
4.8.4 Nahrání aplikace ke schválení 54
4.8.5 Validace 55
**5. Tvorba propagačního videa** 56
5.1. Pre-produkce 56
5.1.1 Námět 56
5.1.2 Výběr postav a lokací 57
5.2 Produkce 57
5.3 Post-produkce 58
5.3.1 Střih 58
5.3.2 Vizuální efekty 58
5.3.2.1 Informační grafika 60
5.3.2.2 3D model 61
5.3.3 Finální úpravy 62
**6. Závěr** 63

# 1. Úvod

V roce 2007 společnost Apple představila pod vedením Steva Jobse nový produkt na trh s mobilními telefony pod názvem iPhone. [1] Po jeho oficiálním uvedením na trh řada skeptiků v tomto oboru striktně zavrhla možnost, že by toto nové zařízení mohlo získat jakkoli významný podíl na trhu s mobilními telefony. Postupem času se tato tvrzení ukázala jako mylná, iPhone popularizoval kategorii smartphonů se stal vzorem chytrého telefonu pro ostatní společnosti na trhu s těmito zařízeními. V současné době tvoří podíl mobilních zařízení s operačním systémem iOS 38,58 % [2] a tuto pozici si společnost Apple vybudovala především díky promyšlenému designu jejich softwarového prostředí iOS.

S příchodem prvních zařízení na trh byl zároveň představen i nový operační systém iPhone OS, který byl ve své 4. generaci přejmenován na iOS. S první verzí systému byl navíc zpřístupněn pro vývojáře iPhone SDK[1] a umožnil tak vývoj aplikací třetích stran pro tuto platformu. S příchodem druhé verze systému byla navíc představena distribuční platforma App Store[2], kde mohli vývojáři nabídnout uživatelům své aplikace a nová verze operačního systému byla navíc integrována do hudebních zařízení iPod. S představením multimediálního tabletu iPad určeného především pro konzumaci obsahu se v roce 2010 rodina zařízení využívajících systém iOS rozrostla o další kus a v současné době je na App Store dostupných více než 1 500 000 aplikací pro tato zařízení. [3]

Tato práce seznamuje se základními technikami a postupy, které jsou využívány při vývoji aplikací pro zařízení s operačním systémem iOS, od prvotního nápadu, přes realizaci, až po umístění aplikace na platfromu App Store. Dále pak ukazuje následnou tvorbu propagačního videa, které napomáhá při prezentaci vyvinuté aplikace The BMX Buddy mezi cílovou komunitou.

Práce je koncipována do tří základních částí, které jsou dále děleny do jednotlivých kapitol. První část se věnuje vývoji samotné aplikace a ukazuje kroky, které jsou nedílnou součástí při vývoji moderní mobilní aplikace. První kapitola představuje operační systém iOS, jeho funkcionalitu a strukturu, další kapitoly se věnují plánováním a přípravě materiálů potřebných k vývoji aplikace, tvorbě sketchů[3], které umožňují designérovi utříbit svoje myšlenky

1 SDK je zkratkou pro Software Developement Kit - sada nástrojů, která umožňuje tvorbu softwaru pro danou platformu.

2 Služba integrovaná do zařízení s operačním systémem iOS umožňující uživatelům stahování a instalaci aplikací třetích stran přímo do jejich zařízení.

3 Hrubý náčrtek obvykle v papírové formě.

a nápady, prototypů[1] s jejichž pomocí je možné simulovat interakci s aplikací, wireframů[2] které umožňují přesně umístit obsah na jednotlivé obrazovky a definují, jak má být aplikace organizována a mockupů[3], které jsou finální grafickou podobou výsledné aplikace.

Další část se věnuje samotnému vývoji aplikace The BMX Buddy. První kapitola věnuje seznámení s prostředím Xcode, které slouží k vývoji aplikací pro platformy společnosti Apple. Další kapitoly ukazují postup vývoje aplikace od prototypu až k finálnímu produktu a prezentaci aplikace na App Store. V této části jsou představeny techniky tvorby uživatelského rozhraní v prostředí Xcode, jeho napojení na kód, který nese základní funkcionalitu aplikace a provázání aplikace se systémem Parse, který zajišťuje potřebné backend služby[4] a SDK, který umožňuje nezávislý vývoj aplikace bez nutné závislosti na kódu na straně serveru[5]. Závěr této části je věnován postupu, jakým je finální aplikace umístěna na App Store.

Třetí část se zabývá tvorbou propagačního videa a zaměřuje se na tvorbu námětu, který by vhodně prezentoval aplikaci The BMX Buddy, dále na realizaci videa, kde jsou rozvedeny natáčecí techniky a postupy a na závěr na post-produkci[6] díla, kde jsou ukázány střihové techniky, grafické techniky a kompoziční techniky v prostředích programů Adobe Premiere, Adobe After Effects a Cinema 4D.

Tato práce nepopisuje a nerozebírá do hloubky práci s jednotlivými programy, které byly použity při tvorbě aplikace či následného propagačního videa. Popisuje postupy které vedly k realizaci a nabízí přehled témat týkajících se této problematiky, které mohou být cílem dalšího studia, ať už se jedná o fázi prototypování, vývoje či audiovizuální produkce.

---

1 Funkční model výsledného produktu, který je součástí vývojového procesu a slouží k demonstraci a testování funkcionality.

2 Návrh definující vzhled a obsahovou podobu produktu.

3 Model produktu ve finální podobě, sloužící k demonstraci designu.

4 Model poskytující programátorům způsob napojení jejich aplikací na služby poskytované serverem.

5 Počítač který posktyuje uživatelům určité služby nebo počítačový program, který tyto služby realizuje.

6 Část procesu tvorby audiovizuálního díla, při které dochází ke střihu, tvorbě grafiky a vizuálních efektů, úpravě zvuku a dalším postupům, které vedou k finální podobě audiovizuálního díla.

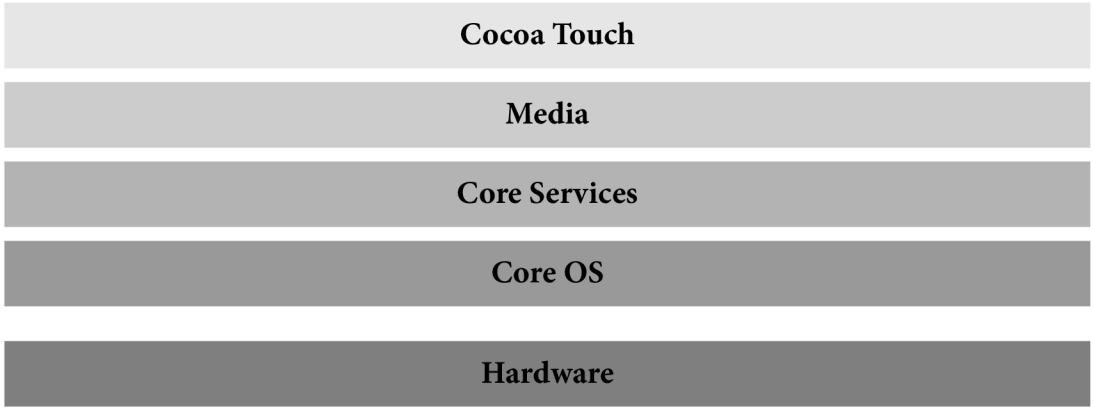
# 2. Technologie iOS

iOS je operačním systémem pro zařízení typu iPhone, iPad a iPod touch poskytujícím funkcionalitu pro práci s hardwarovými[1] zdroji těchto zařízení. Systém navíc obsahuje několik předinstalovaných aplikací jako například webový prohlížeč Safari, Telefon, Email a mnoho dalších. [4]

Společně s operačním systémem je dostupný pro vývojáře také iOS Software Developement Kit (SDK), který umožňuje vytvářet, testovat, instalovat a spouštět nativní[2] aplikace pro tato zařízení. Nativní aplikace jsou psány za použití jazyka Objective-C[3] či Swift a s použitím systémových knihoven. Pochopení technologií použitých při stavbě systému a nástrojů SDK výrazně ulehčuje designování a vývoj nových aplikací.

## 2.1 Architektura iOS

Operační systém iOS je navržen jako prostředník mezi aplikacemi a hardwarem. Tento přístup umožňuje psát aplikace, které fungují vždy stejně bez ohledu na hardware daného zařízení. Tato architektura je reprezentována čtyřmi vrstvami [Obr. 1]. Vrstva nejblíže k hardwaru s ním zajišťuje základní práci a komunikaci a vrstvy nad ní postupně poskytují čím dál více sofistikovanější služby.




Obr. 1 - Vizualizace rozložení vrstev iOS

1 Technické vybavení daného zařízení.
2 Aplikace, která je fyzicky nainstalovaná na daném zařízení a je vždy dostupná uživateli, i když je přístroj odpojen od internetové sítě. [4]
3 Objektově orientovaný programovací jazyk využívaný společností Apple.

### 2.1.1 Vrstva Cocoa Touch

Cocoa touch je nejvyšší vrstvou iOS a poskytuje většinu nejpoužívanějších knihoven pro tvorbu iOS aplikací. Tyto knihovy významným způsobem definují vzhled aplikací za pomoci služeb jako Storyboard, která slouží pro vizuální tvorbu uživatelského rozhraní, Auto Layout, která umožňuje vytvářet dynamická uživatelská rozhranní s minimem kódu a umožňují práci s technologiemi jako je multitasking[1], práce s dotykovou obrazovkou a gesty, sdílení dat AirDrop, službou Hangout a mnoha dalšími.

Mezi nejpoužívanější knihovny této vrstvy patří Address Book UI Framework, který poskytuje standardní systémové rozhraní pro práci s kontakty, Message UI Framework, který poskytuje rozhraní pro odesílání emailů a SMS, EventKit UI Framework pro práci s kalendářem, MapKit Framework, který přináší možnost integrace map do aplikací nebo například iAd Framework pomocí něhož lze do aplikací integrovat reklamní bannery.

### 2.1.2 Vrstva Media

Tato vrstva ležící v hierarchii pod vrstvou Cocoa Touch, zprostředkovává služby pro práci s médii, jako například vykreslování grafiky nebo přehrávání audia a videa.

Mezi nejčastěji využívané knihovny pro vykreslování grafiky patří UIKit Framework, který slouží k vykreslování obrázků a Beziérových křivek[2], Core Graphics Framework pro dynamické renderování[3] grafiky nebo také Core Animation Framework pro práci s animacemi.

Často používané knihovny pro práci s audiem jsou například Media Player Framework, který je používán pro přehrávání audio obsahu z uživatelovy knihovny iTunes[4] nebo AV Foundation Framework, který je využíván pro nahrávání a přehrávání audia či videa.

Stejně jako u práce s audiem je jednou z často používaných knihoven AV Foundation Framework, která slouží pro nahrávání a přehrávání videa či jednodušší AVKit Framework, který umožňuje snadné přehrávání videa jak ve standardním režimu tak v režimu plné obrazovky.

---

1 Provádění několika procesů současně.
2 Nejpoužívanější typ parametrické křivky v počítačové grafice.
3 Vykreslování obrazu na základě elektronických dat.
4 Aplikace pro organizaci a přehrávání multimédií.

### 2.1.3 Vrstva Core Services

Vrstva Core Services poskytuje high-level[1] funkcionalitu základních systémových služeb iOS. Nejdůležitějšími knihovnami této vrstvy jsou Core Foundation Framework a Foundation Framework, které definují základní typy, které využívají všechny aplikace jako například String[2] či různé typy Collections[3] a podobné. Další využívanou knihovnu této vrstvy je Core Location Framework, která poskytuje přístup k polohovým službám daného zařízení jako jsou GPS, Cell[4] či Wi-Fi a umožňují určit polohu uživatele.

### 2.1.4 Vrstva Core OS

Core OS je nejníže položenou vrstvou ve struktuře iOS a poskytuje ostatním vrstvám low-level[5] funkcionalitu, na které jsou pak založeny ostatní technologie systému. Knihovny této vrstvy jsou používány pouze ve specifických situacích, kdy je nutné pracovat s bezpečnostními službami, komunikovat s externím hardwarem a podobně.

1 Typicky více abstraktní funkce, které se zaměřují na daný systém jako celek.
2 Datový typ pro ukládní textových symbolů.
3 Datové struktury pro ukládání množin objektů.
4 Typ rádiové sítě, která se využívá v telekomunikačních službách.
5 Typicky vice detailní funkce, která se zaměřuje na jednotlivé komponenty.

# 3. Od myšlenky k softwaru

Jelikož vývoj softwaru je komplikovaným procesem, je potřebné věnovat dostatečný důraz na přípravu materiálů a dostatečně systematicky zorganizovat svoje myšlenky do takové podoby, která poté ulehčí následující vývoj. Udržet celý projekt po čas příprav je totiž nereálné a může vést k chybám a opomenutí důležitých částí softwaru a jeho funkcionality. Namísto toho existuje celá řada postupů a nástrojů, které napomáhají k sesumírování detailů o projektu do organizovaného celku a umožňují soustředit se v daném okamžiku pouze na jeden dílčí problém.

## 3.1 Dokumentace

Každý vývoj softwaru by měl začít tvorbou dokumentace požadavků na funkcionalitu a design dané aplikace. Tímto přístupem je omezen výskyt případných problémů s opomenutím funkcionality na minimum a případných chyb v designu uživatelského rozhraní. Tvorba takovéto dokumentace musí být rychlá a snadno editovatelná, aby v průběhu jejího psaní mohla být často upravována, tak jak vznikají nové nápady na další funkcionalitu. Právě pro tento účel existuje mnoho nástrojů, které splňují přesně tyto požadavky.

*Klasická papírová dokumentace* - slouží k rychlému osobnímu utřídění myšlenek a to i v průběhu meetingů. Může být velmi dobře kombinována se sketchováním a následně archivována ve formě naskenovaných dokumentů.

*Plain text dokumenty* - rychlá a jednoduchá metoda pro tvorbu seznamů požadavků na aplikaci. Oproti papírové dokumentaci neumožňuje sketchování, avšak díky elektronické podobě může být snadno sdílena a editována.

*Seznamy úkolů v to-do aplikacích* - tento způsob dokumentace napomáhá k udržení jasného plánování pracovního postupu a přehledu, co už je hotovo a co má být uděláno. Neslouží však k detailní specifikaci požadavků na software.

*Plány v project-management softwaru* - formální způsob zaznamenání informací o pracovním procesu. Tyto plány jsou vhodné pro projektové manažery, kterým slouží jako celkový přehled o vývoji. Nejsou však vhodné pro rozhodování o složitých dílčích problémech.

*Designová specifikace* - formální, definitivní a kompletní popis požadavků na software a soupis jednotlivých termínů vývoje. [5]

Tyto metody dokumentace mohou být v průběhu jejího vytváření různě kombinovány a důležité je, aby na konci této fáze vznikl jasný přehled o všech důležitých vlastnostech, která má aplikace splňovat, problémech, které je nutné vyřešit, potencionálních uživatelích a vlastnostech, které naopak mít nemá. Ve výsledku může mít tato dokumentace v závislosti na komplexnosti aplikace pouze několik stránek či se může jednat o rozsáhlý dokument se stovkami stran.

### 3.1.1 The BMX Buddy - základní popis

The BMX Buddy je aplikací, která má BMX[1] komunitě v České republice a jiných částech světa poskytnout rychlý přehled spotů[2] v okolí v podobě mapy s umístěnými body zájmu a zároveň umožnit uživatelům snadno a rychle přidávat nové spoty v podobě fotky s dalšími dodatečnými informacemi. Základním požadavkem na aplikaci je tedy hlavně rychlost při používání a jednoduchost, jelikož jakákoli nadbytečná funkcionalita by vedla ke komplikováné obsluze a následné ztrátě uživatelů. Aplikace je určena výhradně pro BMX komunitu, která rozumí dané problematice a není určena pro nikoho jiného. Aplikace také nemá sloužit jako sociální síť typu Instagram[3], ačkoliv sdílení fotek by tomu nasvědčovalo.

Základním požadavkem na aplikaci je ukázat uživateli mapu s jeho aktuální polohou a na mapě zobrazit jasně rozlišitelné body zájmu, tak aby měl uživatel ihned jasnou představu, jaký typ spotů se v jeho okolí nachází. Tyto body zájmu by mělo být možné filtrovat podle jejich typu a zobrazovat tak pouze určité typy spotů.

Jednotliví uživatelé musejí být v aplikaci registrováni, aby bylo možné monitorovat jejich využívání aplikace a jednotlivým uživatelům umožněno případně filtrovat pouze jejich vlastní body zájmu.

Po vybrání daného bodu zájmu by měl být zobrazen detail spotu s jeho fotografií a dalšími informacemi. Aplikace by dále měla umožnit navigovat uživatele z jeho akutální polohy k vybranému bodu zájmu.

Aplikace musí umožnit rychlé přidání nového spotu. Uživatelé nechtějí strávit přidáváním nového spotu více než pár sekund. Přidání nového spotu musí být provedeno bez nutnosti vyplňovat textová pole a být jednoduše obslouženo pouze jednou rukou.

---

1 Adrenalinová odnož cyklistiky, při které jsou využívány 20“ kola, na kterých jsou prováděny trikové prvky.

2 Místo, kde lze jezdit na BMX kole.

3 Sociální síť pro mobilní platformy sloužící k pořizování a sdílení fotografií.

### 3.1.2 The BMX Buddy - seznam požadavků

Následující seznam požadavků shrnuje všechnu funkcionalitu, kterou by měla výsledná aplikace obsahovat a zároveň ukazuje přehled vlastností, které by dané funkce měly mít.

- **Vytvoření nového uživatele**
    - uživatelské jméno
    - heslo
- **Přihlášení uživatele**
    - pouze jednou
- **Odhlášení uživatele**
    - resetování filtrů a zobrazení úvodní stránky s přihlášením
- **Zobrazení mapy**
    - ihned po startu
    - aktuální poloha uživatele
    - načtení bodů zájmu v okolí
        - jasně rozlišitelné body zájmu
- **Filtrování bodů zájmu**
    - podle typu
        - Park
        - Rail
        - Ledge
        - Wallride
        - Gap
        - Trails
        - Bank
        - Quarter
        - Stairs
    - přidané uživatelem
- **Zobrazení detailu spotu**
    - zobrazení fotografie
    - zobrazení typu spotu
    - zobrazení adresy, připadně názvu spotu
    - zobrazení informace o časové využitelnosti spotu
    - navigace ke spotu z aktuální polohy
- **Přidání nového spotu**
    - rychlé a jednou rukou
    - pořízení čtvercové fotografie jednou rukou
    - automatické načtení aktuální pozice
    - automatické vyplnění názvu aktuální adresou
    - rychlý výběr typu spotu
        - carousel
    - rychlý výběr časové využitelnosti spotu
    - možnost zrušení vytvoření nového spotu
    - uložení

### 3.1.3 The BMX Buddy - dokumentace architektury

Po vytvoření uceleného seznamu všech požadavků na funkcionalitu, kterou by měla aplikace obsahovat, je vhodné zamyslet se nad rozložením této funkcionality mezi jednotlivými zobrazeními dané aplikace. Dokumentace architektury pomůže při následující tvorbě sketchů a wireframů jako základní vodítko a seznam funkcionality, které mají dané obrazovky poskytovat.

```
- Login screen
        - Command: log in
        - Command: sign in
        - Nickname
        - password (hidden)
        - Alert pop-up (invalid nickname or password when loging in)
        - Alert pop-up (username already used when signing in)
- Map screen
        - Command: zoom in/out
        - Command: move around
        - Command: to current location
        - Command: to spot detail
        - Command: create new spot
        - Command: to settings
        - User location
        - Spots
- Settings screen
        - Spot types filters
        - User's spots filter
        - Command: switch on/off filters
        - Command: log out
- Spot detail screen
        - Photo
        - Spot type
        - Spot name
        - Spot safe to ride time
        - Command: navigate to spot from current location
- New spot screen
        - Command: take rectangular photo
        - Command: save new spot
        - Command: select spot type
        - Command: select safe to ride time
        - Photo
        - Spot name (editable)
        - Spot type (carousel)
        - Safe to ride time (tap bar)
```

## 3.2 Sketchování

Sketchování je základním stavebním kamenem designu. Každý nápad by měl být přenesen do vizuální podoby, kterou mohou vidět i ostatní a jednoduše si tak udělat představu o budoucím designu. Při sketchování mohou v průběhu vzniknout další otázky, které je nutné vyřešit a proto je první fáze sketchování často propojena s úpravami prvotní dokumentace.

Ke sketchování softwarového designu není potřeba výtvarného cítění. Sketchování slouží k přenesení myšlenek do vizuální podoby, sdílění těchto nápadů a jejich dalšího rozvíjení. V této fázi není důležitá přesná vizuální podoba jednotlivých prvků rozhraní ani obsah, který mají sdělovat, ale důraz je kladen na rozložení prvků po obrazovce a jejich velikost a vzájemnou kompaktnost jako celku.

Sketchování má největší potenciál při prezentacích, kde rozvíjí diskuzi o designu softwaru. Pouhá dokumentace většinou vede k nejasnostem a subjektivních představách o výsledné podobě a právě sketchování tyto nejasnosti eliminuje a vzniká konstruktivní debata na dané téma. V průběhu prezentace je tedy dobré zasahovat do původní sketche, upravovat ji, přidávat či odebírat prvky a udržovat nákres v aktuální podobě, tak jak se vyvíjí diskuze.

V průběhu sketchování často vznikají takzvané mrtvé konce[5], tedy místa, kde daný design přestává fungovat a je třeba přijít s novým nápadem. To je naprosto v pořádku a čím dříve jsou nápady přeneseny do formy nákresů, tím dříve jsou tyto mrtvé konce odhaleny a eliminovány. V některých případech je dokonce vhodné sketchovat tyto mrtvé konce záměrně, jelikož vizualizace problému může vést k novým nápadům na jeho řešení. Hlavní síla sketchování je v její rychlosti a nenáročnosti, design může být kdykoli zahozen a nahrazen novým.

### 3.2.1 Sketchovací nástroje

K vytváření rychlých nákresů potřebujeme nástoje, které nám toto umožní. Nákresy nesmějí zabrat příliš mnoho úsilí, měly by být nepřesné a s minimálním důrazem na detail. Z těchto důvodů je vhodné se vyvarovat přesných softwarových nástrojů, které vytvářejí přesné tvary a dokonalé texty, jelikož tato podoba nákresů by vedla spíše k diskuzím o grafických podobách jednotlivých prvků než jejich rozložení.

Ideálními sketchovacími nástroji je vše, co umožňuje rychlý a nepřesný nákres, který je možné upravovat a zasahovat do něj. Z tohoto důvodu je nejvhodnějším nástrojem papír či tabule, které umožňují rychlé ruční náčrty. Existuje ale řada softwarových nástrojů, které jsou také vhodné. S příchodem tabletů je možné přistupovat ke sketchování stejně za využití sketchovacích aplikací a stylusu či prstu.

### 3.2.2 Postup při sketchování

Intuitivním postupem při sktechování je přechod od dokumentace architektury k její vizuální podobě. V některých případech jednoduchých aplikací může být fáze přípravy dokumentace přeskočena a funkcionalita je rovnou zanesena do prvních nákresů.

Nákres začíná přibližným tvarem displeje zařízení, na kterém bude software realizován. V případě papírových nákresů je vhodné se velikostí nákresu co nejvíce přiblížit skutečné velikosti displeje, jelikož tento nákres poté jasně definuje prostor se kterým je možné pracovat.

Poté jsou do nákresu umisťovány jednotlivé grafické prvky jako tlačítka, menu, různé druhy polí pro prezentaci obsahu a podobně. V tomto momentě je doporučeno mít znalost HIG1[1] dané platformy, pro které je software navrhován. Design aplikace by měl z HIG vycházet a využívat grafické prvky dané platformy. Přílišná originalita je mnohdy kontraproduktivní a je vhodné využít prvků se kterými má uživatel již předchozí zkušenost.

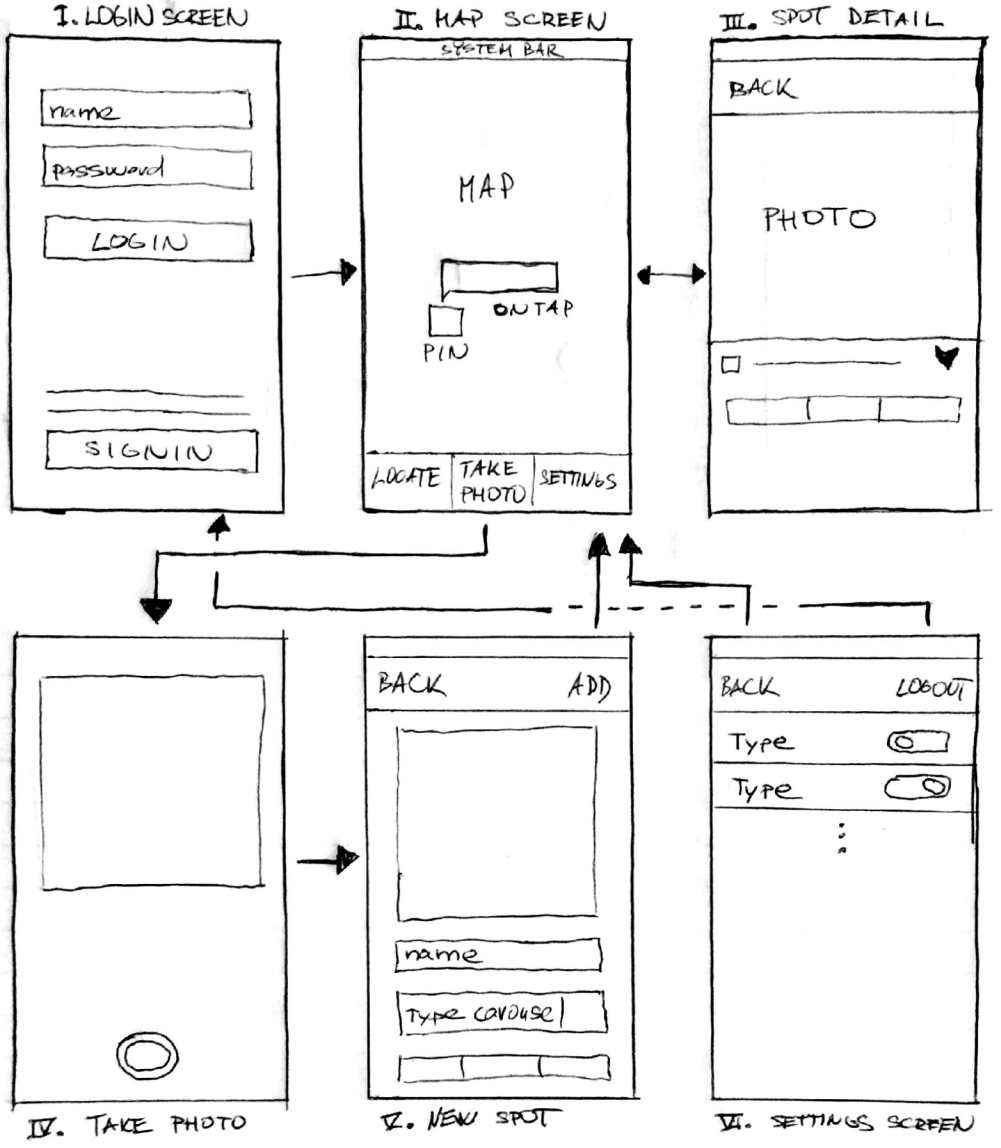
V určitých případech vzniká potřeba znázornit interakci s rozhraním. Proto může být nákres rozšířen o grafické znázornění uživatelovo interakce a následné reakce rozhraní na tento podnět. Tyto interakce jsou reprezentovány určitým počtem nákresů, které znázorňují jednotlivé kroky pro vykonání daného úkonu.

Podobně je možné znázornit práci s celým systémem a přechody mezi jednotlivými obrazovkami aplikace. Tímto způsobem vzniká jasný vizuální přehled o navigaci v aplikaci a lze diskutovat o přechodech, které lze v aplikaci vykonávat a které je nutné přidat, odebrat, sjednotit či zjednodušit a urychlit. Tato navigace je jednoduše znázorněna pomocí orientovaných cest vedoucích od jedné obrazovky k druhé, případně směřujících přímo od prvků, které danou navigaci vykonávají.

1 Human Interface Guidelines - sada grafických prvků, pravidel a doporučení pro návrh grafických rozhraní pro danou platformu.

### 3.2.3 The BMX Buddy - sketch

Na následujícím obrázku [Obr. 2] je uveden prvotní nákres uživatelského rozhraní aplikace The BMX Buddy. Vzhledem k jednoduchosti aplikace jsou načrtnuty všechny potřebné obrazovky, navigace mezi nimi a u určitých prvků i jejich interakce. Nákres již zohledňuje iOS Human Interface Guidlines a možné použití prvků z tohoto dokumentu.




Obr. 2 - Papírový sketch aplikace The BMX Buddy

## 3.3 Wireframing

Jakmile je proces sketchování dokončen a existuje jasná vizuální představa o podobě výsledné aplikace, začíná fáze přípravy wireframů. Účelem wireframů je doplnit design o informace, které byly v nákresech opomíjány. Wireframe však není nutně definitivní vizuální podobou aplikace, jako je tomu u mockupu. U jednoduchých aplikací však wireframe může být zároveň i mockupem. Hranice mezi sketchem, wireframem a mockupem jsou velmi tenké. Existuje několik hlavních rozdílů, které odlišují wireframe oproti ostatním fázím vývoje [5].

- Zakomponování všech prvků, které se mají objevit ve finálním designu. Sketch může některé vizuální prvky ignorovat.

- Definování typu, přesné pozice a velikosti. U sketche tyto vlastnosti jednotlivých prvků nejsou podstatné.

- Wireframe neudává přesnou vizuální podobu prvku narozdíl od mockupu. Používají se prvky s jednoduchým flat designem[1].

- Wireframe dokáže udávat vizuální důležitost některých prvků, například odlišnou barvou či volbou jiného řezu písma.

### 3.3.1 Nástroje pro přípravu wireframů

Narozdíl od nákresů kde preciznost nehrála roli, u wireframů je tato vlastnost na prvním místě. Proto se příprava wireframů provádí vždy v dedikovaných softwarech, které poskytují nástroje pro ulehčení návrhu. Existuje celá řada programů, které jsou pro tento účel vhodné. Hlavními funkcemi, které musejí tyto programy poskytovat, je přesné měření vzdáleností, jednoduchá editace velikosti a tvaru jednotlivých elementů a jejich rozmístění po obrazovce. Za zmínku stojí programy od společnosti Adobe jako například Fireworks, Illustrator či Photoshop, či od jiných společností jako například Balsamiq nebo OmniGraffle.

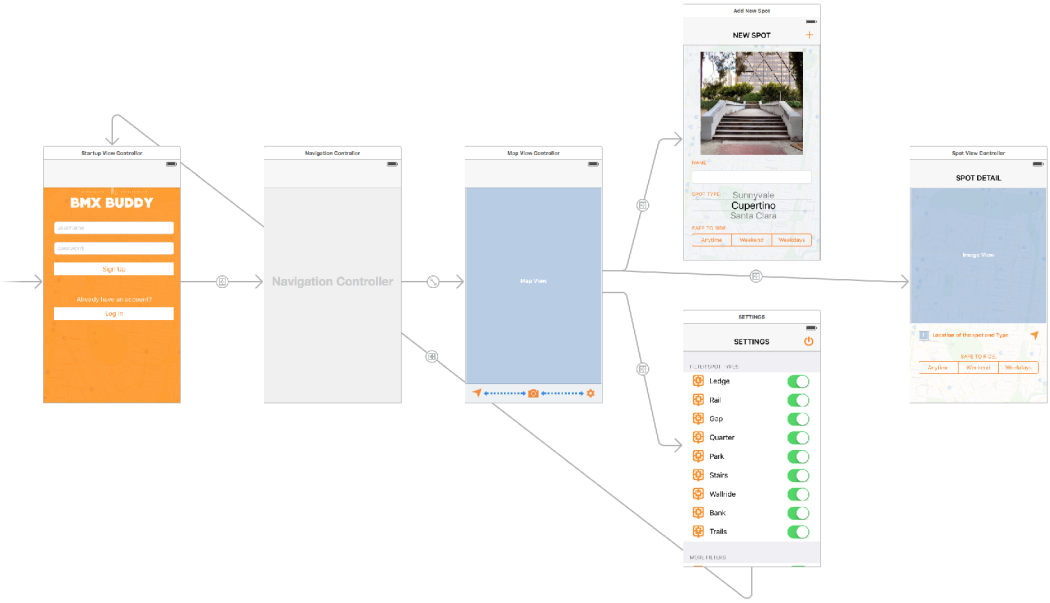
V některých případech lze pro návrh wireframů použít přímo nástroje vývojářského prostředí, jako je tomu u Xcode pro platformy Apple. Toto prostředí má vlastní nástroj pro tvorbu uživatelských rozhraní, se kterým lze velmi rychle připravit přesný wireframe a dokonce i funkční prototyp. Nevýhodou je omezení designu pouze na standardní prvky rozhraní a nemožnost rychlého návrhu vlastních prvků. Naopak výhodou je rychlý přesun od wireframu k vlastnímu vývoji, jelikož tento návrh je možné ihned napojit na logickou část aplikace.

---

1 Minimalistický návrh bez jakýchkoli grafických prvků, jako například stínování, které dávají návrhu dojem trojrozměrnosti.

### 3.3.2 The BMX Buddy - wireframe

Jelikož je The BMX Buddy poměrně jednoduchou aplikací wireframe je realizován ve vývojovém prostředí Xcode. Vývoj v tomto prostředí ve většině případů začíná návrhem uživatelského rozhraní, které se v tomto případě nazývá Storyboard. Do Storyboardu jsou přidávány nová okna (View), která reprezentují jednotlivé obrazovky aplikace. Tyto okna mají implicitně neurčitou velikost, jelikož návrh rozhraní probíhá nezávisle na zařízení, na kterém bude aplikace běžet. Pro lepší představu o rozložení prvků byla pro účely návrhu aplikace The BMX Buddy vybrána velikost okna odpovídající rozměrům 4“ displeje, kterými disponují zařízení modelové řady iPhone 5.




Obr. 3 - Storyboard pro aplikaci The BMX Buddy vytvořený ve vývojovém prostředí Xcode

Do těchto oken je pak možné vložit jednotlivé základní elementy rozhraní a nadefinovat jim určité parametry jejich vzhledu. Jelikož již při sketchování bylo hleděno na dodržování iOS Human Interface Guidelines, většina použitých prvků je právě ze základní sady. Tím je dosažena maximální možná znalost práce s rozhraním aplikace ihned po prvním spuštění. Některé elementy jako například body zájmu, které mají vlastní vzhled, jsou definovány programově a nejsou ve wireframu obsaženy, to však je rozumným kompromisem mezi rychlostí návrhu a jeho funkcí.

Rozložení jednotlivých prvků v oknech je definováno pomocí funkce Auto Layout, kterou disponuje prostředí Xcode. Prvkům jsou definována omezení (Constraints), které udávají jejich relativní či absolutní vzdálenost k jinému prvku či k samotnému oknu a stejným způsobem je pak definována i velikost těchto prvků. Touto metodou je zaručena přenositelnost designu na různé typy zařízení s různými velikostmi displeje bez nutnosti návrhu rozhraní pro každé zařízení zvlášť.

Ve Storyboardu je navíc možné vytvářet i vazby (Segue) mezi jednotlivými okny, které definují možnost přechodu z jednoho okna do druhého. Tyto vazby mohou být umístěny mezi jednotlivými okny a poté programově volány za pomocí přiřazeného identifikátoru či mohou být vyvolány některým z grafických elementů rozhraní po provedení určité akce. Tímto způsobem lze velmi snadno vytvořit funkční prototyp s možností přechodů mezi jednotlivými obrazovkami.

Storyboard může být v průběhu jeho vytváření testován za pomocí náhledových oken (preview), které umožňují sledovat rozložení prvků na jednotlivých typech zařízení nebo spuštěn pomocí iOS Simulátoru[1], kde může být otestována i základní funkcionalita. Jakmile je Storyboard v prostředí Xcode hotov může být napojen na aplikační logiku, která zajišťuje potřebnou funkcionalitu. Jednotlivým prvkům rozhraní, které vyžadují funkcionalitu lze navíc jednoduše přiřadit části kódu, což velmi zrychluje samotný proces vývoje aplikace.

1 Nástroj, který vytváří virtuální zařízení pro prototypování a testování aplikace v průběhu jejího vývoje.

## 3.4 Prototypování

Prototypy přicházejí na řadu v momentě, kdy pouhé statické snímky jednotlivých obrazovek nestačí. V některých případech je nutné vyzkoušet interakci s rozhraním a otestovat jeho chování. Prototyp je také vhodný pro testování ostatními členy týmu či náhodnou skupinou vybraných jedinců, kteří nejsou zapojeni do vývojového procesu. Tímto způsobem lze odhalit zásadní chyby v navigaci již v raných počátcích vývoje.

Prototyp tedy slouží k simulaci funkcionality dané aplikace. Simulovány mohou být pouze některé klíčové části softwaru či mohou být vytvářeny sofistikované prototypy velmi podobné finálnímu produktu. Stejně tak se mohou prototypy lišit jejich provedením. Prototyp může mít papírovou podobu až po perfektní funkční simulaci aplikace. Prototypování je jedním z nejzásadnějších fází před samotným začátkem vývoje a mělo by být opomínáno pouze ve specifických případech.

Zásadní vlastností prototypu by mělo být přiblížení se svou velikostí zařízením, na kterých bude finální produkt spouštěn. V nejlepším případě je vhodné vytvořit takový prototyp, který je možné otestovat přímo na daném zařízení. Tímto způsobem mohou být ideálním způsobem otestovány vlastnosti jako velikost jednotlivých prvků a jejich dosah.

### 3.4.1 Prototypovací nástroje

Existuje celá řada nástrojů a přístupů jak k návrhu prototypů přistupovat. Společnou vlastností je ale zacílení na znázornění nějakého druhu chování. Na následujících řádcích je uvedeno několik nejzásadnějcích přístupů.

*Papírový prototyp* - kousky nastříhaného papíru, které simulují prvky rozhraní. Pomocí pohybu těchto kusů papíru lze rychle simulovat interaktivitu rozhraní, jako například práci se slidery[1] či pop-up okny[2].

*Prototyp čaroděje ze země Oz* [5] - většinou elektronický prototyp s minimální funkcionalitou. Práce s tímto prototypem vyžaduje dohled moderátora, který pokládá otázky a případně odhaluje výsledou interakci.

*Animace a demo videa* - prototyp v této formě neumožňuje žádnou informaci, ale ukazuje chování softwaru.

---

1 Prvek rozhraní který slouží k výběru hodnot v určitém rozsahu.

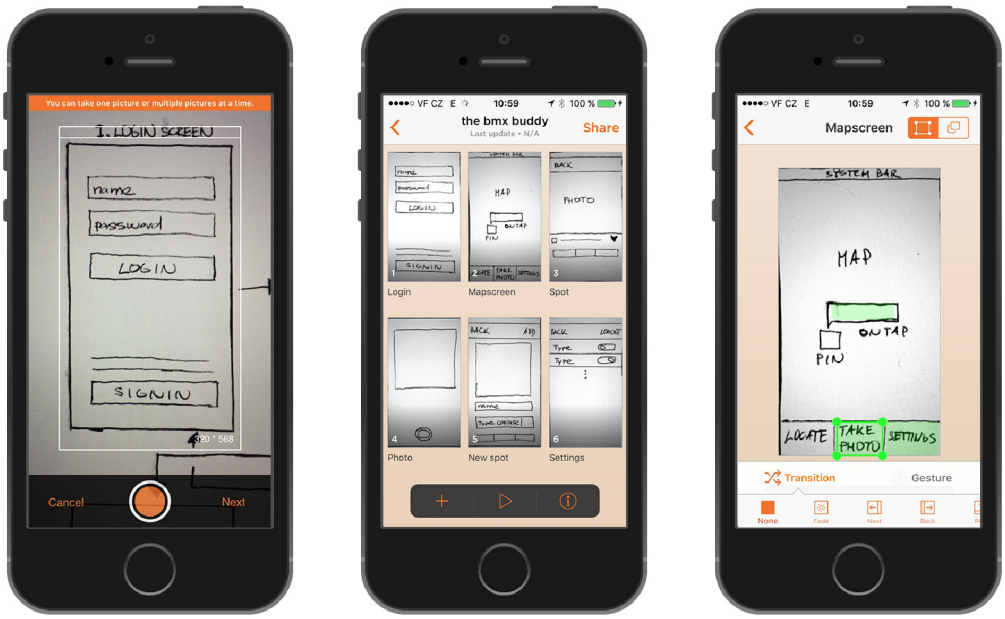
2 Okno s krátkou zprávou, které oznamuje určitou změnu stavu v aplikaci a vybízející k provedení akce.

*Interaktivní prototyp* - Existuje celá řada nástrojů pro tvorbu interaktních prototypů, které dovolují přímou interakci. Mezi tyto nástroje patří například OmniGraffle, Balsamiq, PowerPoint, POP či Storyboard vytvořený v Xcode.

*Proof of concept*[1] *software* - nejsofistikovanější podoba prototypu, Jedná se o víceméně funkční aplikaci, na které je testováno její budoucí chování. Pokud je tato metoda dostatečně dlouho opakována může dojít až k finální podobě softwaru.

### 3.4.2 Prototypování pomocí aplikace POP

Pro účely prototypování aplikací pro mobilní zařízení je vhodné použít aplikaci POP2[2], která využívá papírový sketch a přidává mu potřebnou interaktivitu. V prvním kroku jsou pořízené snímky jednotlivých obrazovek a poté pomocí intuitivních nástrojů je možné vytvořit přechody od jednotlivých prvků k jiným obrazovkám [Obr. 4]. Aplikace navíc podporuje i simulaci různých gest a je možné zvolit z nejpoužívanějších systémových přechodů mezi obrazovkami. Velmi rychle je tak vytvořen plně interaktivní prototyp.




Obr. 4 - Práce s prototypovací aplikací POP

1 Důkazová metoda sloužící k ověření teoretických vlastností.
2 Aplikace volně ke stažení na webových stránkách popapp.in

## 3.5 Mockupy

Jakmile jsou přípravné fáze hotovy, je na čase začít s přípravou finální podoby aplikace. Mockup představuje přesnou podobu obrazovky a v idálním případě je k nerozeznání od finálního produktu často označované jako pixel-perfect zobrazení. Mockup je tedy primárním výstupem designu aplikace.

Tvorba mockupu je většinou záležitostí grafika, kteří připraví finální podklady potřebné pro tvorbu aplikace. Jsou vytvořeny jednotlivé grafické elementy ve finálních podobách, které jsou poté použity při implementaci rozhraní. Ačkoli je tato fáze primárním výstupem příprav nejedná se však o tu nejpodstatnější část, jelikož podle tvrzení Steva Jobse: *„Design není jen o tom jak prostředí vypadá, ale především o pocitu, jaký má uživatel z jeho používání a celkové funkci prostředí."* Proto je důležité věnovat velký důraz na sketchování a prototypování a finální grafickou podobu řešit až v konečné fázi.

Přesto je ale nutné této fázi věnovat dostatek pozornosti, jelikož dobře provedený design může významně ovlivnit výsledný pocit z užívání softwaru. Grafický design podporuje intuitivitu, jelikož pokud rozhraní působí, že má ukotvení ve fyzickém světě, tím více je vytvořena iluze, že může být s prvky manipulováno a interagováno. [5] Design je navíc jakýmsi obalem daného softwaru a čím více působí aplikace dotažená po grafické stránce, tím více je uživatel přesvědčen, že do celkového vývoje bylo vloženo maximální úsilí. V poslední řadě dobrý design zpříjemňuje práci s daným produktem. Precizně připravené tlačítko sice neulehčí práci s tímto prvkem, ale má znatelný vliv na celkový pocit z práce s aplikací.

### 3.5.1 Nástroje pro přípravu mockupů

U přípravy mockupů existuje celá řada softwarových nástrojů, které jsou pro tento účel využívány. Většina grafiků nevyužívá k přípravě pouze jeden software, ale vybudují si své vlastní workflow1[1] pro přípravu grafických podkladů. Zde jsou dva nejčastěji využívané programy pro tvorbu grafiky.

*Photoshop* - jedná se o nejrozšířenější a nejvíce využívaný nástroj pro přípravu grafických podkladů. Počet jeho nástrojů a celková komplexnost je však mnohonásobně vyšší než je pro design grafických rozhraní potřeba a od toho se odvíjí i jeho cena. Jeho schopnost práce ve vrstvách, s vektorovými[2] tvary a orientace na bitmapovou[3] grafiku jsou však hlavním přínosem.

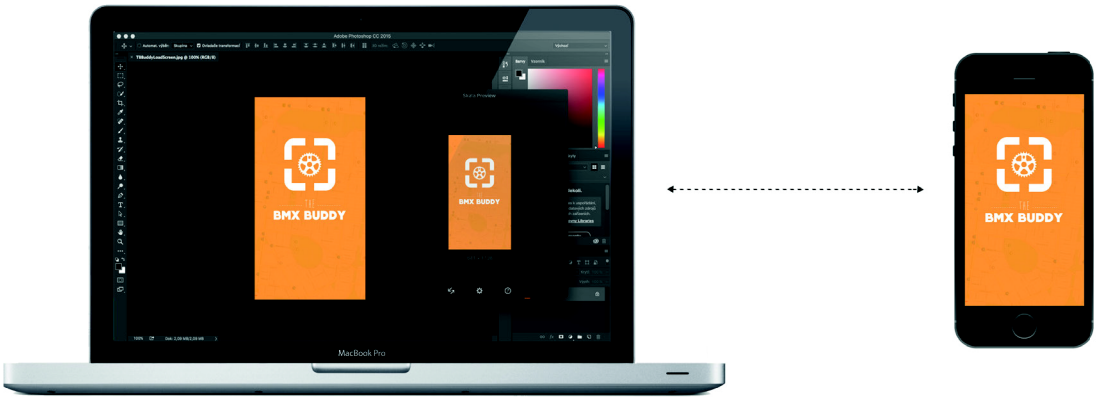
---

1 Pracovní postup, kterým je dosažené efektivní dosažení cíle.
2 Objekty definované pomocí přesných matematických funkcí.
3 Definice grafických prvků pomocí pixelové mřížky.

*Illustrator* - poskytuje stejné nástroje jako Photoshop, ale je orientovaný na přípravu grafiky pro tisk. Příprava pixel-perfect grafických podkladů je v tomto prostředí velmi snadná avšak jednotlivé míry v pixelech, které jsou důležité pro následující použití jsou definovány až při exportu.

### 3.5.2 Příprava mockupu za pomocí Photoshop a Scala Preview

Existuje celá řada nástrojů, které přípravu mockupů ještě více usnadňují. Nástroj Scala Preview [Obr. 5] je idálním nástrojem pro přenos návrhů přímo do cílového zařízení. Scala Preview[1] je nainstalován do zdrjového zařízení a pomocí Remote Connection[2] připojen k programu Photoshop. Dále je aplikace Scala Preview nainstalována na cílové mobilní zařízení. Po spárování těchto zařízení je možné v reálném čase přenášet pixel-perfect náhledy a mít tak jasnou představu o výsledné podobě.




Obr. 5 - Komunikace mezi Photoshop a Scala Preview
na zařízení iPhone

1 Aplikace ke stažení na bjango.com/mac/scalapreview/
2 Vzdálené propojení hostitele a hosta.

# 4. Vývoj aplikace

V momentě kdy jsou přípravné fáze dokončeny, je na čase, pustit se do vývoje samotné aplikace. Jelikož je aplikace The BMX Buddy určena pro zařízení iPhone, proces jejího vývoje je zaměřen čistě na použití metod pro tuto platformu. K vývoji aplikace je využíváno prostředí Xcode a samotný zdrojový kód je psán v programovacím jazyku Swift, který byl společností Apple navržen speciálně pro potřeby vývoje aplikací pro tato zařízení.

## 4.1 iOS vývojář

Narozdíl od jiných platforem, vývoj aplikací pro zařízení Apple vyžaduje více počátečního úsilí než jen začít psát řádky samotného kódu. K vývoji je zapotřebí stolní nebo přenosný počítač se systémem iOS Snow Leopard a novějším a s minimálně 6GB volného místa na pevném disku. Od verze operačního systému Lion je poté možné jednoduše přes rozhraní App Store stáhnout aktuální verzi vývojového prostředí Xcode a nainstalovat jej na dané zařízení.

Dalším krokem je vstoupení do Apple Developer Programu, které poté umožňuje vývojářům psát, spouštět a testovat své vlastní aplikace pomocí iOS simulátoru. Vstoupení do tohoto programu je zdarma, má však jistá omezení. S příchodem Xcode 7 bylo odstraněno i největší omezení, kterým byla možnost instalovat své aplikace do vlastních zařízení a jediným zpoplatněným bodem tedy nyní zůstává prezentace aplikací přes rozhraní App Store. Tímto krokem společnost Apple kontroluje aplikace, které mají potenciál být na App Store umístěny a zajišťuje tím určitou úroveň kvality aplikací, které zde lze nalézt.

Registrace jako vývojář probíhá přes Apple iOS Dev Center[1]. Zde je možné využít stávajícího Apple ID[2] či vytvořit nové speciálně pro tyto účely. Dále stačí následovat registrační proces, který končí aktivací vývojářského účtu. Vývojáři, kteří jsou rozhodnuti prezentovat svoji aplikaci na App Store či využívat další prémiové funkce, mohou poté přistoupit k registraci do placeného programu.

Posledním krokem je vytvoření Provisioning Profilu, který opravňuje vývojáře spouštět aplikaci na reálném zařízení. Tento profil nese všechny informace o vývojáři, případných dalších členech týmu, dané aplikaci a zařízení na kterém bude spuštěna. V novějších verzích Xcode je tento krok zejdnodušen a po připojení zařízení proveden pomocí stisku jediného tlačítka.

1 Lze nalézt na internetových stránkách developer.apple.com/ios

2 Unikátní účet společnosti Apple, který uživatele opravňuje využívat některé ze speciálních funkcí, ke kterým je zapotřebí registrace.

## 4.2 Xcode

Největší objem času při vývoji aplikací pro zařízení Apple je stráveno ve dvou aplikacích, kterými jsou vývojové prostředí Xcode a s ním spojený iOS simulátor. Tyto aplikace poskytují všechny nástroje, které jsou potřebné k designování, programování a testování aplikací pro tato zařízení. Xcode je takzvaným IDE[1], který spravuje všechny zdrojové soubory aplikace, umožňuje editaci kódu a uživatelského rozhraní a udržuje všechny potřebné části pohromadě.

### 4.2.1 Pracovní prostředí Xcode

Většina vývojářské práce v tomto prostředí začíná založením nového Xcode projektu, který shormažďuje všechny soubory spojené s aplikací a potřebná nastavení pro úspěšnou kompilaci[2] spustitelné aplikace. K založení nového projektu můžeme využít několika vzorových projektů, které mohou v některých případech výrazně urychlit start vývoje nebo můžeme založit prázdný projekt a mít naprostou kontrolu nad počátečním nastavením.

Po zvolení typu nového projektu je nutné nastavit několik základních parametrů jako je název aplikace, identifikátor společnosti, kterým je převrácený tvar webových stránek společnosti. Tyto dva údaje společně tvoří takzavaný Bundle Identifier, který jednoznačně identifikuje aplikaci mezi ostatními na App Store [6]. Dále je nutné vybrat jazyk, který bude použit pro vývoj aplikace. Na výběr jsou jazyky Objective-C, který je standardem pro vývoj aplikací pro zařízení Apple a nový jazyk Swift, který díky zjednodušené syntaxi má postupně Objective-C nahradit. Nakonec může být upřesněn typ zařízení pro které je aplikace určena a omezen tak vývoj například pouze na zařízení iPhone. Pokud má aplikace obsahovat vlastní databázový systém či automatizované testy, jsou zde pro tyto pokročilé vlastnosti volby Use Core Data a Include Unit Tests.

Po založení nového projektu je otevřeno pracovní prostředí Xcode [Obr. 6]. Toto prostředí je rozděleno do pěti základních oken, které mají odlišnou funkci v průběhu vývoje. Jelikož ve většině případů není potřeba využívat všechna okna najednou, je možné je jednoduše skrývat pomocí navigace v pravém horním rohu a tím zjednodušovat pracovní prostor a poskytovat více místa ostatním oknům.

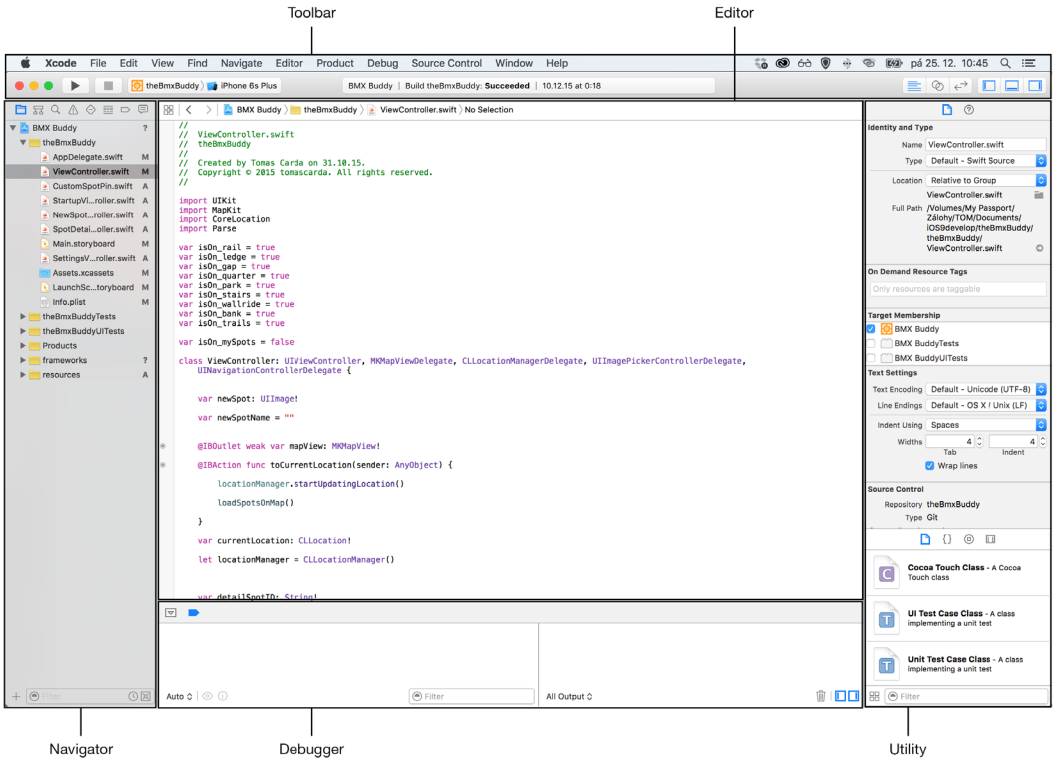
1 Integrated Developement Environment - prostředí pro komplexní vývoj softwaru.
2 Převod zdrojového kódu na spustitelný program.

*Toolbar* - okno programu, poskytující přístup k nějčastěji používaným funkcím, obsluze rozvržení prostředí a spouštění aplikace.

*Navigator* - toto okno slouží k navigaci mezi jednotlivými soubory a informacemi vázajícími se k danému projektu.

*Editor* - základní pracovní okno, které umožňuje práci s kódem, rozhraním a ostatním editovatelným obsahem.

*Utility* - toto okno poskytuje rychlý přístup k vlastnostem jednotlivých prvků a jejich editaci.




Obr. 6 - Rozložení pracovního prostředí Xcode

1 Rozhraní umožňující zobrazovat pouze textové údaje, které do něj mohou být vypisovány za běhu programu.

### 4.2.2 Navigace v projektu

Okno Navigator slouží k navigaci nejen mezi soubory, které jsou součástí projektu, ale také k navigaci strukturou kódu či hledáním chybových hlášení. Většinu času je ale využíváno pro navigaci mezi soubory. Soubory jsou třídeny pomocí hierarchické struktury, kde nejvyšším stupněm je samotný projekt. Tato struktura může být rozbalována a zabalována pro přehlednou orientaci. Tato struktura obsahuje několik základních složek, jejichž pochopení významně ulehčuje navigaci.

*Kód projektu* - tato složka nese název aplikace a obsahuje kódy jednotlivých tříd, Storyboard nesoucí informace o uživatelském rozhraní a další soubory těsně vázané s danou aplikací.

*Frameworks* - knihovny, které přinášejí aplikaci určitý stupeň funkcionality. Po založení projektu tato složka obsahuje pouze nezbytně nutné knihovny pro běh aplikace, po přidání speciální funkcionality jsou tyto knihovny přidávány právě do této složky.

*Products* - složka se soubory, které vygeneroval Xcode v průběhu vývoje. Většinou se jedná spustitelné soubory.

V závislosti na složitosti projektu, může tato hierarchie obsahovat další složky například pro testování. Je také doporučeno organizovat všechny ostatní přidané soubory do složek pro přehlednou navigaci v rámci projektu.

V průběhu vývoje je většinou nutné přidávat další soubory do struktury projektu. Pokud se jedná o zdrojová data, jako například obrázky, hudbu a podbné, tyto soubory stačí jednoduše přetáhnout do okna Navigator. Xcode nabídne vytvoření kopií těchto souborů do místa projektu. Tento krok je doporučené provést pro zajištění dostupnosti souborů.

V komplikovanějších projektech je také nutné vytvořit nové třídy pro organizaci kódu. Pro vytvoření nové třídy je nejprve vybráno cílové umístění v hierarchi projektu a poté z příslušného menu vytvořena nová třída.

### 4.2.3 Práce s kódem

Po vybrání určité třídy v navigačním okně je jejích obsah otevřen v okně Editoru. Práce s kódem funguje stejně jako v každém textovém editoru s přidáním funkcionality, která práci s kódem usnadňuje. K jednoduché navigaci k určitému blogu kódu poslouží Symbol Navigator, který nalezeneme jako druhou záložku v okně Navigator. Za pomocí tohoto okna je možné dostat se rychle k dané třídě, metodě či parametru.

Nejzásadnější vlastností editoru je schopnost automatického dokončování psaného kódu. V průběhu psaní Xcode automaticky kontroluje možné varianty pro dokončení a nabízí je uživateli. Pouhým stiskem klávesy Tab může být vložen dokončený kód. Pokud existuje více možných variant pro dokončení, lze mezi nimi přecházet pomocí šipek a vybrat požadovanou variantu. Xcode se snaží automaticky dokončovat nejen knihovní funkce, ale také nově definované proměnné, metody a další bloky kódu.

Dalším nástrojem pro ulehčení navigace v kódu je použití značky `#pragma mark`. Tato značka slouží v bloku kódu pro rozdělení logických částí a usnadnění navigace. Výskyt této značky se projeví i v Symbol Navigatoru, kde jsou následující metody a parametry jasně vizuálně odděleny a případně i pojmenovány.

Posledním velmi důležitým nástrojem je možnost přepnutí okna Editoru do Assistent Editor módu. Takto jsou zobrazeny dvě okna vedle sebe a je možné pracovat s dvěmi různými částmi projektu zároveň. Tato vlastnost je využívána zejména v místech, kdy je nutné vzájemně provázat jednu část kódu na jiné místo v projektu.

### 4.2.4 Kompilace kódu

Po dokončení psaní zdrojového kódu je na čase spustit výslednou aplikaci. Tento proces provede kompilaci instrukcí a navázání kódu na použité knihovny, které potřebuje aplikace pro svůj běh. V průběhu této fáze může dojít k nalezení chyb v instrukcích a Xcode tyto chyby zobrazí. Pokud překlad proběhne bez problému, je aplikace spuštěna na vybraném zařízení. Aplikace může být spuštěna na připojeném zařízení či na simulátoru.

Aplikace může být také pouze zkompilována ale nespuštěna, tím je ověřen kód na případný výskyt chyb. Další možností je provést analýzu aplikace, která odhalí chyby při kompilaci a identifikuje případné problémy, které nezabraňují kompilaci, ale mohly by způsobit pád aplikace za jejího běhu.

### 4.2.5 iOS Simulator

iOS Simulator vytváří virtuální zařízení na kterém je možné testovat běh aplikací. Toto zařízení se v mnoha ohledech podobá reálnému zařízení avšak není perfektním duplikátem. Některé vlastnosti zařízení jako například přístup k fotoaparátu či gyroskopu[1] nebo akcelerometru[2] nejsou podporovány a pro testování těchto funkcí je nutné připojit realné zařízení.

1 Součástka sloužící ke zjištění orientace zařízení.
2 Součástka měřící zrychlení.

## 4.3 Swift

Programovací jazyk Swift je relativně mladým jazykem, který byl uveden společností Apple teprve v červnu roku 2014 na světové konferenci World Wide Developer Conference. Jeho uvedení bylo velkým krokem, jelikož po mnoho let byl Objective-C hlavním jazykem pro tyto platformy. To že je vývojářská společnost připravena na změnu, ale dokázala hned první čísla, když pouhých 24 hodin po uvedení bylo staženo 300 000 kopií Apple Swift iBook.

Tento nový jazyk se od svého předchůdce neliší pouze syntaxí, ale také celkovým přístupem k programování. Swift není pouze objektově orientovaným jazykem, ale kombinuje také další přístupy známé z jazyků jako C#, Haskell, Ruby či Python. Swift je tak jazykem s moderní a bezpečnější syntaxí a kombinuje různá programovací paradigmata jako objektové[1], funkcionální[2] a imperativní[3] programování. Swift se stále vyvýjí a nyní se nachází ve verzi 2.1. Stále umožňuje práci s knihovnami Cocoa a Cocoa Touch, které jsou využívány v moderních OS X[4] a iOS aplikacích. [1] Velmi důležitou vlastností tohoto jazyka je také důraz na jasné definování typů - type-safe jazyk. Tímto způsobem je zaručeno, že kus kódu bude pracovat vždy pouze s určitým typem a tím je možné předcházet mnoha chybám, které by mohly vznikout za běhu programu.

### 4.3.1 Konstanty a proměnné

Konstatnty a proměnné přiřazují určitému jménu nějakou hodnotu s příslušným typem. Hodnota konstanty je po celý čas neměnná, zatímco hodnota proměnné může být počas běhu měněna. Konstatny a proměnné musejí být deklarovány [Kód 1] před jejich použítím a to za použítí klíčových slov `let` pro konstanty a `var` pro proměnnné. V případě že ihned nepřiřadíme konstatě či proměnné hodnotu, musíme definovat její typ, kterým je jasně dán druh informace, která v ní může být uložena.

```
let maximumAttempts = 10
var currentAttempts = 0
var username: String
```

Kód 1 - Příklad deklarování konstat a proměnných

1 Modelování problému pomocí objektů a komunikace mezi těmito objekty.
2 Programování vyhodnocováním matematických funkcí.
3 Programování pomocí posloupností příkazů, které definují přesný postup.
4 Operační systém pro počítače Apple Mac.

### 4.3.2 Množinové typy

Swift poskytuje tři základní typy polí `Array`, `Set` a `Dictionary` pro práci s množinami. `Array` slouží k ukládání seřazených množin, `Set` k ukládání neseřezených množin s unikátními hodnotami nad kterými poté lze provádět množinové operace a `Dictionary` slouží pro práci s množinami, kde je jednotlivým prvkům přiřazen unikátní klíč. Tyto druhy polí mají vždy definovaný typ, takže je vždy jasně dáno, jakého typu jsou jeho hodnoty.

```
var shoppingList: [String] = ["Eggs", "Milk"]
var favoriteGenres: Set<String> = [„Rock", „Hip hop"]
var airports: [String: String] = [„DUB": „Dublin", „PRG„: „Prague„]
```

Kód 2 - Příklad deklarování množin

### 4.3.3 Kontrola nad během programu

Swift poskytuje všechny dobře známé funkce pro kontrolu běhu jako jsou `for` a `while` cykly, které zajišťují mnohonásobné provedení prováděného úkonu, příkazy `if` a `switch`, které umožňují větvení programu na základě splnění různých podmínek nebo příkazy `break` a `continue`, které umožňují přesměrování běhu do jiných částí kódu. K těmto standradním příkazům navíc Swift přidává příkaz `for-in`, který umožňuje iterování přes prvky množin a jiných sekvencí.

#### 4.3.3.1 For cykly

Swift poskytuje dva typy `for` cyklů. Klasický `for` cyklus, který vykonává určitý blok kódu do té doby, dokud není dosažena určitá specifická podmínka (typicky inkrementováním určité hodnoty) a `for-in` cyklus, který vykoná blok kódu pro všechny prvky dané sekvence.

```
let names = [„Anna", „Alex", „Brian", „Jack"]
for name in names {
    print(„Hello, \(name)!")
}
```

Kód 3 - Příklad for-in cyklu

#### 4.3.3.2 While cykly

`While` cykly provádějí blok kódu dokud je splněna podmínka pro vstup do cyklu. Klasický `while` cyklus otestuje podmínku ihned a pokud je výsledkem `false`, není tělo cyklu provedeno ani jednou. Alternativou k tomuto cyklu je `repeat-while` cyklus, který otestuje podmínku až na konci. V tomto případě je zajištěn minimálně jeden průchod cyklem.

```
repeat {
        tělo cyklu
} while podmínka
```

```
while podmínka {
    tělo cyklu
}
```

Kód 4 - Rozdíl mezi while a repeat-while cyklem

#### 4.3.3.3 Podmíněné výrazy

Existují dva základní často využívané výrazy, kterými jsou příkazy `if` a `switch`. Příkaz `if` ve své základní formě ověří podmínku a na základě jejího vyhodnocení provede či neprovede své tělo. Tento příkaz může být rozšířen o větev `else`, která tvoří alternativní tělo výrazu, které je provedeno pokud podmínka není splněna.

Obdobně funguje příkaz `switch`, který vezme vstupní hodnotu a porovná ji s potenciálními možnostmi, tím je docíleno širší větvení výrazu. Jelikož je nutné obsáhnout všechny možné případy vstupu, musí příkaz `switch` obsahovat i základní větev `default`, která je provedena v případě, pokud není vybrána ani jedna z definovaných podmínek. V případě příkazu `switch` se často uplatňují i příkazy `continue`, který umožňuje vykonání další větve a `break`, který slouží například k vystoupení z větve `default`.

```
switch nějaká hodnota {
case hodnota 1:
    tělo výrazu
case hodnota 2, hodnota 3:
    tělo výrazu
default:
    break
}
```

Kód 5 - Obecná podoba podmíněného výrazu switch s voláním příkazu break

### 4.3.4 Funkce

Funkcí rozumíme samostatně stojící blok kódu, který vykonává nějaký specifický úkon. Funkce má definovné jméno, pomocí kterého je možné funkci zavolat v momentě, kdy má být vykonána. Funkce v jazyce Swift je natolik sofistikovaná, že dokáže definovat jednoduché bezparametrické funkce, až po složité objektově orientované funkce s mnoha interními i externími parametry.

Funkce je definovaná pomocí příkazu `func`, názvu funkce, vstupních parametrů, které mají definovaný název a typ a typem výstupu [Kód 6]. Tělo funkce pak může definovat nové parametry, volat parametry ležící mimo tělo funkce, či volat jiné funkce včetně sama sebe. Ve speciálních případech nemusí mít funkce žádné parametry nebo může jít o funkci bez návratovéhodnoty, která já definovaná pomocí návratového typu `Void`.

```
func sayHello(personName: String) -> String {
    let greeting = „Hello, „ + personName + „!"
    return greeting
}

print(sayHello(„Tomas„))
```

Kód 6 - Definice funkce a její následné volání

Funkce je speciálním typem bloku kódu, který je nazýván Closure. Zjednodušeně řečeno, tento blok kódu uzavírá stav ve kterém se program akutálně nachází a vykonává určitou funkci. Closure je tedy anonymní funkcí a naopak funkce je Closure, které bylo přiřazeno jméno [7].

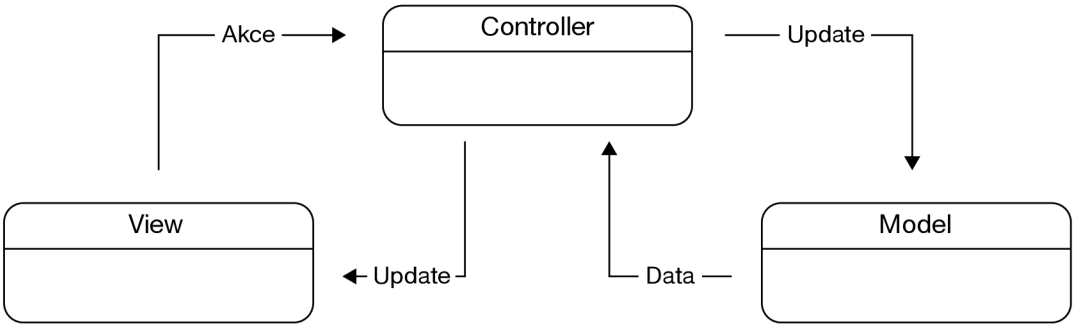
## 4.4 Model-View-Controller Design

Při vývoji aplikací v prostředí Xcode je často využíván přístup nazývaný Model-View-Controller (MVC). Při vývoji aplikace je nutné vyřešit tři základní problémy. Prvním problémem je nějakým způsobem prezentovat uživateli obsah a dovolit mu s ním manipulovat. Druhým problémem je zajistit vhodnou reakci na uživatelovo chování a zpracovat potřebná data. Třetím problémem je tato data vhodným způsobem uchovávat. Jedním způsobem je řešení všech těchto problémů na jednom místě, což vede k nepřehlednému kódu, který je složité udržovat. Proto je vhodné systematicky rozdělit vývoj na dílčí podproblémy, které jsou vyřešeny samostatně a tyto části spolu nechat komunikovat. Tato modularita vede ke snadnější údržbě, jednodušší práci v týmu a potencionální recyklaci kódu. [6]

*Model* - tento modul poskytuje informace o datech, se kterými aplikace pracuje. Nedefinuje ale vzhled aplikace ani její chování. Model může mít podobu databáze či jiného datového modelu.

*View* - modul vytvářející celkové uživatelské rozhraní dané aplikace. V prostředí Xcode lze View definovat programově či pomocí Interface Builder, kde je možné vytvářet Storyboard definující uživatelské rozhraní.

*Controller* - modul typicky napojený na View i Model obsluhující vzájemnou komunikaci a zajišťující funkcionalitu aplikace. Na základě informací z View může Controller upravit Model a naopak. V Xcode je typicky ke každému View ve Storyboadu vytvořen View Controller, který dané View obsluhuje. Projekt ale může obsahovat další Controllery, které zajišťují další funkcionalitu celé aplikace.



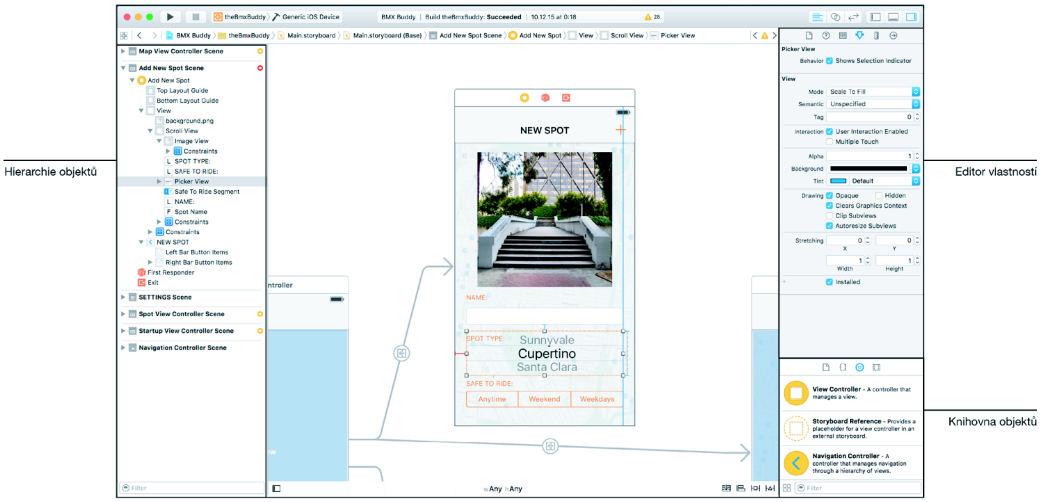

Obr. 7 - Grafické znázornění MVC designu

## 4.5 The BMX Buddy - View

Ke tvorbě uživatelského rozhraní pro aplikaci The BMX Buddy je využito nástroje Interface Builder (IB), který je součástí Xcode [Obr. 8]. IB však není pouze kreslícím nástrojem pro tvorbu GUI[1], ale pomocí propojování objektů dokáže budovat základní funkcionalitu aplikace. Tím je zajištěn výskyt méně chyb v průběhu vývoje, zkrácení času vývoje a jednodušší údržba.

IB sofistikovaným způsobem podporuje design MVC. Pomocí IB je možné vytvářet jednotlivé Views a pomocí takzvaných Connections je propojit s jejich Controllery. Pokud tedy v budoucnosti vznikne potřeba na změnu některých vlastností rozhraní, stačí zanést tyto úpravy pouze do příslušného View bez nutnosti zasahování do zdrojového kódu. Obdobně lze upravit i jednotlivé Connections. Tímto způsobem je opět předcházeno možnému vzniku chyb a nekonzistentního kódu.

IB vytváří XML[2] soubor nazývaný Storyboard, ve kterém jsou uchovávány všechny informace o jednotlivých obrazovkách, jejich objektech a vzájemných propojeních, které jsou nazývány segue. Tento soubor navíc obsluhuje alokaci[3] a inicializaci[4] jednotlivých objektů.




Obr. 8 - Rozhraní Interface Builder pro návrh GUI

1 Graphical User Interface - grafické uživatelské rozhraní prostřednictvím kterého uživatel komunikuje s aplikací.
2 Extensible Markup Language - značkovací jazyk pro popis struktur.
3 Vyhrazení místa v paměti pro daný objekt.
4 Vytvoření instance objektu v paměti s danými vlastnostmi.

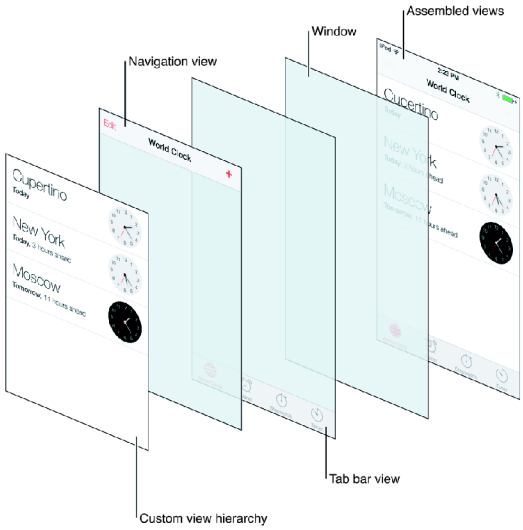
### 4.5.1 Login View

Toto View je zároveň iniciálním oknem aplikace. Tato vlastnost `Is Initial View Controller` lze jednoduše označit v Utility okně. Účel tohoto View je zajistit od uživatele při prvním spuštění aplikace jeho přihlašovací údaje a vytvořit jeho profil, případně umožnit mu připojit se ke stávajícímu profilu.

Toto View obsahuje dvě textová pole (`Text Field`), které slouží k vyplnění požadovaných informací, dvě tlačítka (`Button`), která obsluhují přihlášení či registraci, štítek (`Label`) s informacemi o procesu a dvě `Image View`, které slouží k umístění grafických prvků pozadí a loga aplikace. Funkcionalita tlačítek a informace z textových polí je zajištěna propojením na třídu `StartupViewController`, která poskytuje tomuto oknu potřebnou funkcionalitu.

### 4.5.2 Map View

Map View je základní obrazovkou aplikace. Po úspěšném přihlášení se toto okno stává iniciálním a je tedy zobrazeno po startu aplikace. Map View slouží především k procházení mapy a zobrazování jednotlivých spotů na této mapě. Dále pak slouží k navigaci skrze aplikaci a z tohoto okna lze přejít do okna s nastavením aplikace či vyvolat proces přidání nového spotu.




Obr. 9 - Hierarchie oken při použití Navigation View [11]

Standradizovanou podobou navigace na zařízeních s operačním systémem iOS je obsluha navigační lišty (`Navigation Bar`) v horní části obrazovky. Tato navigační lišta tvoří grafické rozhraní patřící ke třídě `NavigationViewController`, která poskytuje právě tuto funkcionalitu a stará se o správu navigace mezi okny a zobrazování těchto oken současně s navigační lištou [Obr. 9].

Navigace mezi okny funguje na principu zásobníku, kdy na vrchu zásobníku je aktuálně zobrazené okno a pod ním okno, které bylo zobrazeno jako předchozí. Navigační lišta je u Map View, stejně jako u Login View skryta, jelikož tato funkcionalita není u těchto oken potřebná.

Hlavním objektem Map View je `Map Kit View`, který poskytuje všechnu potřebnou funkcionalitu pro práci s mapou. Ve spodní části okna je pak lišta s navigací (`Toolbar`), která nese tři tlačítka (`Bar Button Item`), kterými je aplikace obsluhována. Okno je napojené na třídu `MapViewController`, která zajišťuje příslušnou funkcionalitu.

### 4.5.3 Spot Detail View

Účelem této obrazovky je poskytnout uživateli detailní informace o aktuálně vybraném spotu. Mezi tyto informace patří především fotografie, dále pak typ a název místa, časové omezení pro využití tohoto spotu a možnost nechat se na toto místo navigovat.

Pro tyto účely Spot Detail View obsahuje objekty jako trojici `Image View`, které prezentují fotografii, ikonu typu spotu a pozadí daného okna, tlačítko pro navigaci, štítek pro zobrazení názvu spotu a neaktivní `Segmented Control` pro prezentaci časového omezení. Třídou napojenou na toto View je třída `SpotViewController`.

### 4.5.4 New Spot View

Tato obrazovka má obdobné rozvržení jako detail příslušného spotu. Pomocí této obrazovky má možnost uživatel vytvořit nový spot a umístit jej na mapu. Na této obrazovce je zobrazen náhled fotografie pořízené fotoaparátem, navrhovaný název spotu, který je vygenerován z adresy aktuální polohy, dále je zde možnost vybrat typ spotu a jeho časové omezení a nakonec umístit spot na mapu.

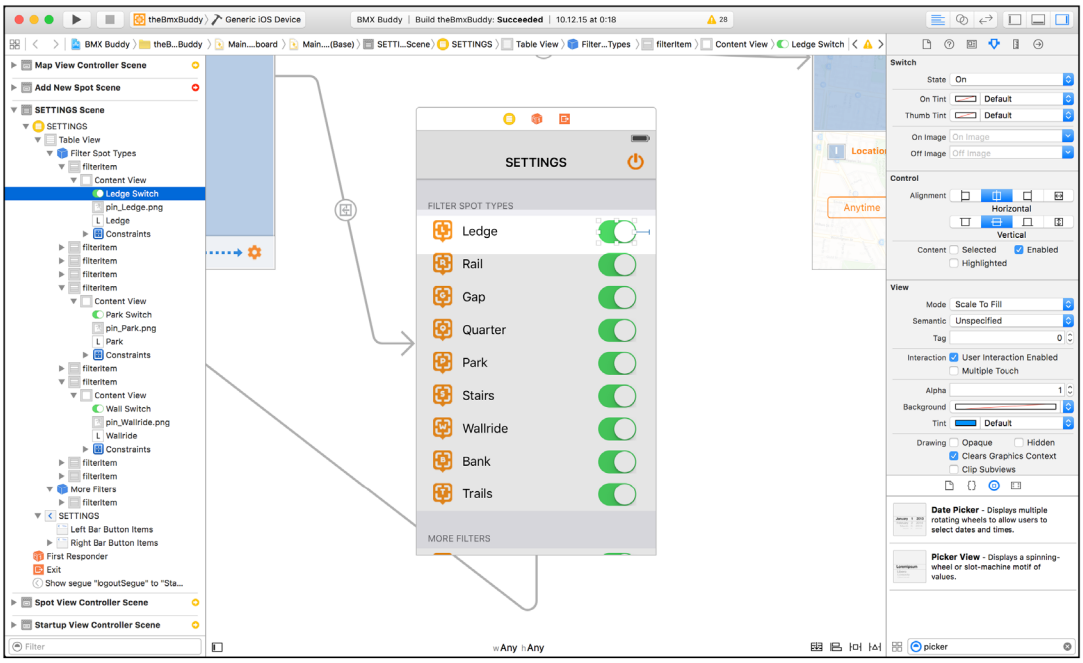
Objekty tvořící toto okno jsou dvojice `Image View` pro zobrazení fotografie a pozadí, štítky pro popisné informace, textové pole s předvyplněným názvem, carousel (`Picker View`) pro výběr typu spotu a `Segmented Control` pro výběr časového omezení. Objekty jsou navíc umístěny do rolovacího okna (`Scroll View`) které zajišťuje zobrazení všech potřebných prvků

i na zařízeních s menší obrazovkou. Tlačítko na přídání nového spotu (`Right Bar Button Item`) je umístěno v navigační liště. Funkcionalita je zajištěna pomocí třídy `NewSpotController`, která je na New Spot View napojena.

### 4.5.5 Settings View

Settings View slouží především k filtrování spotů, které jsou poté zobrazeny na mapě. Spoty můžeme filtrovat podle jejich typu nebo zobrazit pouze spoty přidané uživatelem. Z této obrazovky je také možné provést odhlášení akutálního uživatele, které uvede aplikaci do počátečního stavu.

Settings View tvoří tabulka (`Table View`), která je rozdělena na dvě statické sekce s definovaným počtem řádků obsahující jednotlivé položky ve formě buněk tabulky (`Table View Cell`). `Table View` navíc poskytuje funkcionalitu ve forme rolování obsahem, pokud na obrazovce není dostatek místa. Jednotlivé řádky nastavení jsou tvořeny `Image View` pro zobrazení ikon, štítkem pro zobrazení popisku a přepínačem (`Switch`), který obsluhuje aktivaci a deaktivaci příslušných položek. Nakonec tlačítko pro odhlášení je umístěno do pravé části navigační lišty.




Obr. 10 - Práce s IB na návrhu Settings View

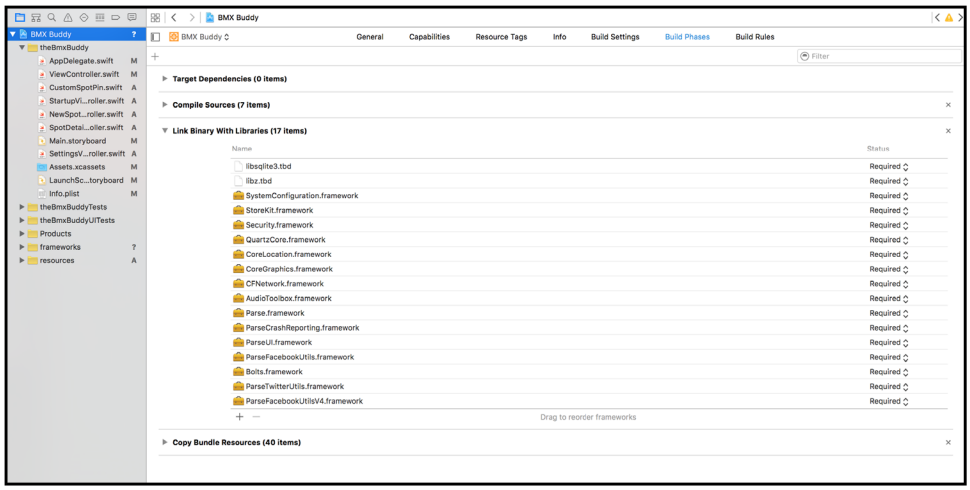
## 4.6 The BMX Buddy - Model

Datový model této aplikace obsahuje informace o jednotlivých uživatelích a spotech. Jelikož tyto informace musejí být v průběhu využívání aplikace dostupně všem uživatelům a ne jen lokálně, vzniká potřeba pro ukládání těchto dat na vzdáleném serveru, který se postará o ukládání a poskytování těchto informací. Právě pro tento účel se výborně hodí služba Parse, která zajišťuje potřebné databázové služby.

### 4.6.1 Parse

Parse je službou, která poskytuje kvalitní a stabilní backend pro návrh aplikací pro většinu běžně využívaných platforem od drobných mobilních zařízení až po desktopy. Parse zároveň poskytuje SDK, pomocí kterého lze vyvýjet aplikace bez závislosti na kódu na straně serveru, což významně uhlečuje a zrychluje proces vývoje. Parse ale poskytuje také mnohem více než jen práci s databázemi. Poskytuje také nástroje pro integrování médií sociálních sítí, push notifikace a další služby s plnou možností modifikace. [8] Tato služba je navíc za určitých podmínek naprosto zdarma.

Po vytvoření uživatelského účtu na webových stánkách služby Parse[1] stačí ve webové administraci vytvořit novou aplikaci s příslušným jménem. Aplikace dostane přidělené Application ID, které jí společně s Clinet Key jasně identifikuje. Tato kombinace klíčů slouží poté k následné komunikaci mezi aplikací a službou Parse.




Obr. 11 - Import knihoven nutných pro běh Parse

1 Parse lze nálezt na stránkách www.parse.com

Dalším krokem je intergování knihoven Parse do vytvořeného projektu v Xcode [Obr. 11]. Po založení projektu stačí přetáhnout složku `Parse.framework` do složky Frameworks v navigačním okně. Dále je nutné připojit několik knihoven, které Parse využívá pro svoje účely. Ze záložky Build Phases v okně se základnimi informace o dané aplikaci stačí do menu Link Binary with Libraries naimportovat následující knihovny:

```
CoreLocation.framework
CoreGraphics.framework
libz.1.1.3.dylib
MobileCoreServices.framework
QuartzCore.framework
Security.framework
StoreKit.framework
SystemCon guration.framework
AudioToolbox.framework
CFNetwork.framework
```

Nakonec stačí otevřít třídu `AppDelegate.swift`, importovat knihovnu Parse a upravit funkci `didFinishLaunchingWithOptions`, tak aby umožnila identifikaci aplikace ve službě Parse pomocí unikátních klíčů.

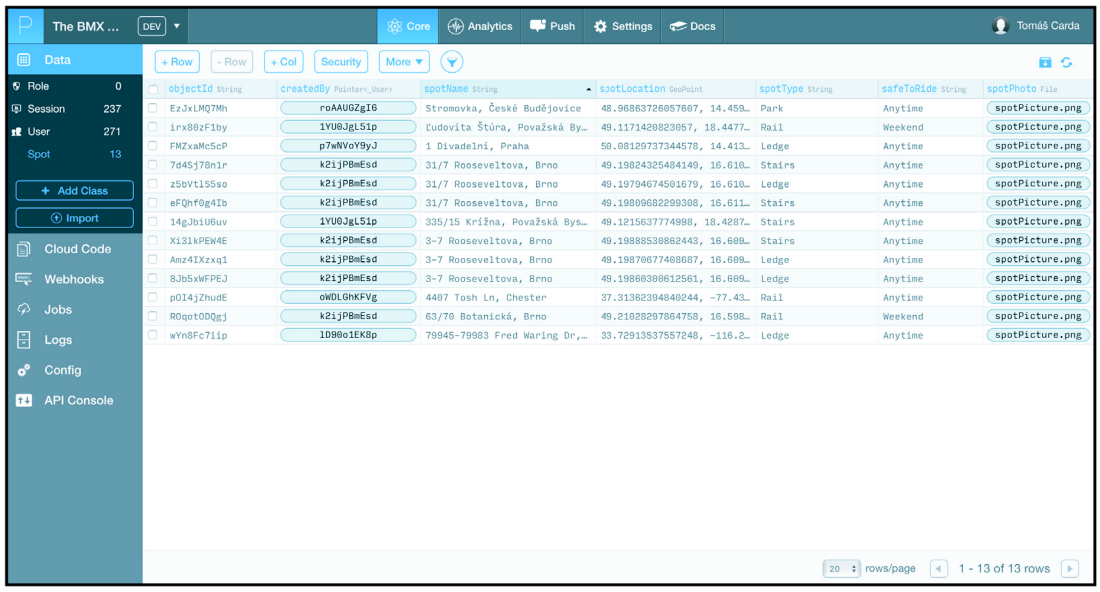
```
func application(application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [NSObject: AnyObject]?) -> Bool {

        // [Optional] Power your app with Local Datastore.
        Parse.enableLocalDatastore()

        // Initialize Parse.
        Parse.setApplicationId(„epEqAQWVRNYkOaDqe6QBr1VT8dgXMPqJPmkJdPRi“,
            clientKey: „HyTiDjCKQXoEJcMvJ5bc7MuTdhlKZHxETPHxFrKy“)

        // [Optional] Track statistics around application opens.
        PFAnalytics.trackAppOpenedWithLaunchOptions(launchOptions)

        return true
}
```

Kód 7 - Podoba funcke didFinishLaunchingWithOptions s inicializací Parse

### 4.6.2 Datový model The BMX Buddy

Databázová struktura pro aplikaci The BMX Buddy je tvořena pouze dvěma tabulkami. První tabulka jménem User slouží pro evidenci uživatelů a podoba této tabulky je dána službou Parse. Pro aplikaci The BMX Buddy jsou nejpodstatnějšími sloupci této tabulky položky `objectId`, pro jednoznačnou identifikaci, `username` pro nastavení jedinečného uživatelského jména a `password` pro bezpečné šifrované ukládání uživatelského hesla.

Druhá tabulka Spot slouží pro evidenci spotů [Obr. 12]. Základní informace jako název spotu, jeho typ a časové omezení jsou v této tabulce uchovávány ve formě textových řetězců (`String`), fotografie spotu je do tabulky uložena jako soubor (`File`) a přesná poloha spotu je uložena ve formě `GeoPoint`, což je datová struktura definovaná službou Parse pro uchovávání geografických informací o zeměpisné šířce a délce. Ostatní sloupce této tabulky jsou vytvořeny službou Parse. Za zmínku stojí jednoznačný identikátor `objectId` a údaj o tvůrci spotu `createdBy`, který má podobu ukazatele na řádek (`Pointer`) v tabulce User.




Obr. 12 - Administrace služby Parse

## 4.7 The BMX Buddy - Controller

Funkcionalita celé aplikace je zajištěna pomocí několika Controllerů, které společně s několika dalšími třídami tvoří most mezi datovým modelem a grafickým rozhraním. Každá obrazovka aplikace má svůj příslušný Controller, se kterým je propojena pomocí outletů (`IBOutlet`) a akcí (`IBAction`), a který obstarává komunikaci mezi databází Parse a také s ostatními Controllery aplikace.

`IBOutlet` je typ proměnné, která slouží ke zprostredkování informace na grafický výstup. Pokud je tedy proměnné definované jako `IBOutlet` přiřazena určitá hodnota, tato hodnota je pak načtena do grafického výstupu objektu, který je na tuto proměnnou navázán.

```
@IBOutlet weak var usernameField: UITextField!
```

Kód 8 - definice proměnné usernameField jako IBOutlet

`IBAction` je funkcí, která slouží pro provedení určité akce, která je vyvolána uživatelovo interakcí s prvkem grafického rozhraní, který je na tuto funkci navázán. Akce vyvolaná prvkem rozhraní je obvykle zpracována `IBAction` funkcí a výsledek je obyvkle uložen do `IBOutlet` proměné a zobrazen zpět na obrazovce.

```
@IBAction func logIn(sender: AnyObject) {
      vnitřní kód funkce
}
```

Kód 9 - definice funkce logIn jako IBAction

### 4.7.1 StartupViewController.swift

Tato třída poskytuje funkcionalitu při přihlášení do aplikace. Pro všechny objekty z příslušného View je v této tříde vytvořen příslušný `IBOutlet`, jelikož každý z těchto objektů může být v průběhu změněn a může prezentovat jiný obsah. Dvě tlačítka na obrazovce mají příslušné `IBAction`, které zajišťují jejich funkcionalitu. Dále tato třída definuje funkci pro práci s klávesnicí a funkci pro zobrazení pop-up okna s chybovým hlášením v případě, že se vyskytne problém při pokusu o přihlášení do systému.

Funkce `signUp` obsluhuje komunikaci se službou Parse. V prvním kroku je ověřeno, zda uživatel vyplnil tato pole, pokud ne je zavolána funkce `displayAlert`, která zobrazí chybové hlášení. Po ověření uživatelovo vstupu je znemožněná uživateli další práce s rozhraním a spuštěn pokus o komunikaci se službou Parse. I přes svůj název tato funkce obstarává jak pokus o registraci, tak o přihlášení do systému. Podle příznaku `signupActive` je zvolena větev programu, která vykonává registraci či přihlášení.

Při pokusu o registraci nového uživatele je vytvořen nový uživatel a pomocí tohoto uživatele je proveden pokus o registraci. Při úspěšném provedení je uživatel uložen v databázi a otevřeno nové okno Map View. Při neúspěchu je vráceno chybové hlášení informující o důvodu vzniku.

```
let user = PFUser()
user.username = usernameField.text
user.password = passwordField.text

user.signUpInBackgroundWithBlock({ (success, error) -> Void in
      blok kódu pro obsluhu success či error
})
```

Kód 10 - zjednodušená část kódu komunikující s Parse

Při pokusu o přihlášení je obsah textových polí ověřen s databází a v závislosti na správnosti vstupu je uživateli umožněn přístup do systému a otevřeno nové okno Map View, případně je zobrazeno chybové hlášení informující o chybě vzniklé při tomto pokusu.

Funkce `logIn` je navázána na tlačítko ve spodní části obrazovky, které mění stav obrazovky z pokusu o registraci na pokus o přihlášení a naopak. Ve skutečnosti tato funkce pouze zajišťuje změnu vizuální podoby rozhraní v závislosti na operaci, kterou uživatel chce provést a veškerá logika je přenechána funkci `signUp`.

Za zmínku stojí přetížená[1] funkce `viewDidAppear`, která v závislosti na tom, zda již aplikace byla spuštěna a na zařízení je aktivní uživatel umožní okamžité přesměrování na obrazovku Map View.

1 Metoda při které je dovolené dědičné třídě přepsat určitou metodu a dodat jí specifickou funkcionalitu.

### 4.7.2 MapViewController.swift

Tato třída poskytuje funkcionalitu základnímu oknu aplikace Map View a obsluhuje polohové služby, které jsou potřebné pro vykonávání většiny úkonů v aplikaci. Ve třídě lze nalézt `IBOutlet` pro Map Kit View, který zajišťuje prezentaci mapy a pinů na ní, `IBAction` pro přesun na akutální pozici uživatele a přidání nového spotu a další funkce zajišťující potřebnou logiku. Blok kódu je vizuálně rozdělen do několika tématických částí, pro zpřehlednění orientace.

Funkce `locationManager` se stará o zjištění akutální polohy uživatele, vycentrování mapy na tuto pozici, a překlad těchto polohových dat na textový údaj s adresou akutuální polohy. Tato funkce je zavolána vždy, když je vytvořena žádost o nové načtení polohy objektem `locationManager` a voláním jeho funkce `startUpdatingLocation()`.

```
func locationManager(manager: CLLocationManager, didUpdateLocations
locations: [CLLocation]) {

        currentLocation = locations.last!

        CLGeocoder().reverseGeocodeLocation(currentLocation,
        completionHandler: {(placemarks, error) -> Void in
                blok kódu obsluhující překlad adresy
        })

        centerMapOnLocation(currentLocation)
        locationManager.stopUpdatingLocation()
}
```

Kód 11 - zjednodušená podoba funkce locationManager

Dalším blokem kódu jsou funkce zajišťující práci s fotografiemi. Základ tvoří `IBAction addNewSpot`, která ověří přítomnost hardwarové kamery na zařízení a v případě existence inicializuje Controller pro práci s kamerou, kterému předá řízení aplikace. Další funkce v tomto bloku se postarají o uložení fotografie do proměnné, zmenšení a ořez fotografie na čtvercový rozměr o velikosti 640x640 pixelů a následný přesun fotografie na obrazovku Add New Spot View.

```
func imagePickerController(picker: UIImagePickerController,
didFinishPickingImage image: UIImage, editingInfo: [String : AnyObject]?) {

        self.dismissViewControllerAnimated(true, completion: nil)
        let selectedImage = image
        newSpot = resizeImage(selectedImage)
        performSegueWithIdentifier(„toAddNewSpot", sender: self)
}
```

Kód 12 - obsloužení práce s fotografií po jejím pořízení

Funkce `centerMapOnLocation` vycentruje mapu na aktuální polohu a zároveň ji přiblíží na požadovanou vzdálenost. Dále následují funkce starající se o prezentaci jednotlivých pinů na mapě. Funkce `loadSpotOnMap()` zjistí, které typy spotů se mají načíst a pro ně poté vytvoří dotaz v databázi Parse. Databáze na pozadí zpracuje dotaz a vrátí pole objektů, kterým je postupně procházeno a pro každý objekt vytvořen příslušný pin na mapě. O podobu tohoto pinu se poté postarají další funkce, které upraví jeho vzhled.

```
let query = PFQuery(className:"Spot")
query.whereKey(„spotType", containedIn: spotTypesSwitchedOn)
if isOn_mySpots {
        query.whereKey(„createdBy", equalTo: PFUser.currentUser()!)
}
```

Kód 13 - příklad vytvoření dotazu pro databázi Parse

Přetížená funkce `prepareForSegue` připravuje data pro přesun na další obrazovky. Na základě hodnoty identifikátoru připraví data pro vytvoření nového spotu či pro zobrazení detailu spotu. Pomocí této funkce je zajištěna komunikace mezi jednotlivými Controllery, které si tak mohou navzájem zasílat data. Je vytvořena instance cílového Controlleru a poté je možné přistoupit k jeho proměnným a přiřadit jim hodnoty.

Třída také obsahuje přetížené funkce `viewWillAppear`, `viewWillDisapperar` a `viewDidLoad`, které se starají o provedení určitých úkonů při prvním spuštění či opětovném zobrazení tohoto okna.

### 4.7.3 SpotDetailController.swift

Tato třída poskytuje funkcionalitu pro zobrazení detailních informací o vybraném spotu. Z toho důvodu má většina prvků GUI přiřazený svůj `IBOutlet` pomocí kterého jsou informace zobrazeny na obrazovce. Jelikož jediným účelem této třídy je prezentace obsahu obsahuje pouze jedinou `IBAction` funkci, která rozšiřuje interaktivitu rozhraní. Touto funkcí je `toNavigation`, která umožňuje uživateli zapnout navigování na pozici daného spotu. Tato funkce vytvoří požadované místo určení, otevře aplikaci Maps a předá jí toto místo společně s údaji, jakým způsobem má k danému místu navigovat.

Zbylou funkcionalitu zajišťuje přetížená funkce `viewDidLoad`, která zajistí vytvoření dotazu pro Parse pro načtení dodatečných informací o spotu. Na základě unikátního klíče `spotID` je načtena fotografie spotu, jeho typ, časové omezení a přesná poloha.

```
@IBAction func toNavigation(sender: AnyObject) {
        let coordinate = CLLocationCoordinate2D(
        latitude: spotLocation.latitude, longitude: spotLocation.longitude)

        let navigationLocation = MKPlacemark(coordinate: coordinate,
        addressDictionary: nil)

        let mapItem = MKMapItem(placemark: navigationLocation)

        let launchOptions = [MKLaunchOptionsDirectionsModeKey:
        MKLaunchOptionsDirectionsModeWalking]

        mapItem.openInMapsWithLaunchOptions(launchOptions)
}
```

Kód 14 - funkce toNavigation zajišťující navigaci ke spotu

### 4.7.4 NewSpotController.swift

Tato třída zajišťuje přidání nového spotu do databáze. Fotografie, poloha spotu a vygenerovaná adresa jsou přeneseny z třídy `MapViewController` pomocí metody `prepareForSegue` a uloženy do proměnných v této třídě. Jednotlivé objekty rozhraní mají přiřazen svůj `IBOutlet` pro prezentaci obsahu, dvě funkce `IBAction` obstarávají interakci s uživatelem, stejně jako další metody obstarávající práci s klávesnicí či carouselem.

Funkce `segmentChanged` reaguje na interakci s objektem Segment Control, který umožňuje výběr časového omezení. Po vybrání jednoho ze segmentů je výběr uložen v proměnné `safeToRide` jako textový údaj.

Čtveřice funkcí `pickerView` se stará o obsluhu carouselu pro výběr typu spotu. Do carouselu jsou načteny jednotlivé typy a je zajištěno uložení akutálně vybraného typu do proměnné `spotType`.

Trojice funkcí `textField` obsluhuje práci s klávesnicí. Jelikož při zobrazení klávesnice by bylo překryto textové pole, tyto funkce zajišťují posun GUI, aby bylo textové pole viditelné. Po skrytí klávesnice je obrazovka uvedena do původního stavu.

Hlavní funkcionalitu zajišťuje funkce `addSpot`, která převede data do formátů vhodných pro databázi Parse, deaktivuje obrazovku pro znemožnění uživatelovo interakce v průběhu ukládání a vyvolá požadavek na přídání řádku do tabulky Spot ve službě Parse. V případě úspěšného uložení je znovu aktivována obrazovka a otevřeno okno Map View. V případě neúspěchu je aktivována obrazovka a otevřeno pop-up okno s informací o vzniklé chybě.

```
let spotObject = PFObject(className: „Spot")

spotObject[„spotName"] = seguedName
spotObject[„spotType"] = spotType
spotObject[„createdBy"] = PFUser.currentUser()
spotObject[„spotLocation"] = newSpotPoint
spotObject[„spotPhoto"] = imageFile
spotObject[„safeToRide"] = safeToRide

spotObject.saveInBackgroundWithBlock {
(success: Bool, error: NSError?) -> Void in
	if success {
		blok kódu obsluhující úspěšné uložení
	} else {
		blok kódu obsluhující neúspěšné uložení
	}
}
```

Kód 15 - zjednodušená podoba kódu zajišťující vytvoření nového řádku v tabulce Spot v databázi Parse

### 4.7.5 SettingsViewController.swift

V této třídě je zajištěna aktivace a deaktivace typů spotů, které mají být zobrazeny a také umožnení odhlášení uživatele. Pro každý řádek tabulky ze Settings View a objekt Switch v tomto řádku je vytvořen `IBOutlet` a `IBAction`, které obstarávají zobrazení stavu a změnu globální proměnné, která tuto informaci uchovává. Funkce `logOut` zajišťuje odhlášení uživatele od služby Parse, uvedení aplikace do výchozího stavu a přesun na úvodní obrazovku pro autorizaci uživatele.

```
@IBAction func logOut(sender: AnyObject) {

        PFUser.logOut()

        isOn_rail = true
        isOn_ledge = true
        isOn_gap = true
        isOn_quarter = true
        isOn_park = true
        isOn_stairs = true
        isOn_wallride = true
        isOn_bank = true
        isOn_trails = true

        isOn_mySpots = false

        performSegueWithIdentifier(„logoutSegue", sender: self)
}

@IBOutlet weak var personalSwitch: UISwitch!

@IBAction func personalSwitchChanged(sender: AnyObject) {

        isOn_mySpots = !isOn_mySpots
}
```

Kód 16 - fuknce logOut a příklad práce s jedním z objektů Switch

## 4.8 Umístění aplikce na App Store

V momentě kdy je aplikace hotová a otestovaná, je na čase umístit ji na prezentační platformu App Store. Na App Store je v současné době přibližně 1.5 milionu aplikací a počet stažení sahá přes hranici 100 miliard. Apple přesto klade poměrně vysoké nároky na aplikace, které mohou být na App Store umístěny. Aplikace jsou kontrolovány a pravidla pro úspěšné nasazení aplikace jsou velmi striktní. Aplikace musí být především plně funkční, neměla by se v podobných formách na App Store již vyskytovat a měla by mít určitý přínos pro uživatele. Aplikace může být zamítnuta z těchto důvodů: [9]

- padá při provozu
- vykazuje chyby
- není v souladu se specifikací vývojáře
- zahrnuje nedokumentované funkce, které nesouvisejí s popisem aplikace
- používá neveřejné API
- čte nebo zapisuje data mimo svůj kontejner
- jakkoli stahuje kód
- instaluje či spouští kód
- jedná se o beta, demo nebo zkušební verzi
- je kopií aplikací v App Store již umístěných
- není velmi užitečná
- je primárně zaměřená na reklamu
- jde o trik či obsahuje falešné funkce
- podporuje nadměrnou konzumaci alkoholu a jiných návykových látek
- poskytuje nesprávné údaje
- obsahuje jen jednu skladbu či film, ta má být umístěna na iTunes Store
- obsahuje jednu knihu, ta má být umístěna na iBookstore
- omezuje uživatele, které ji mohou používat
- neřídí se pokyny iOS pro ukládání dat

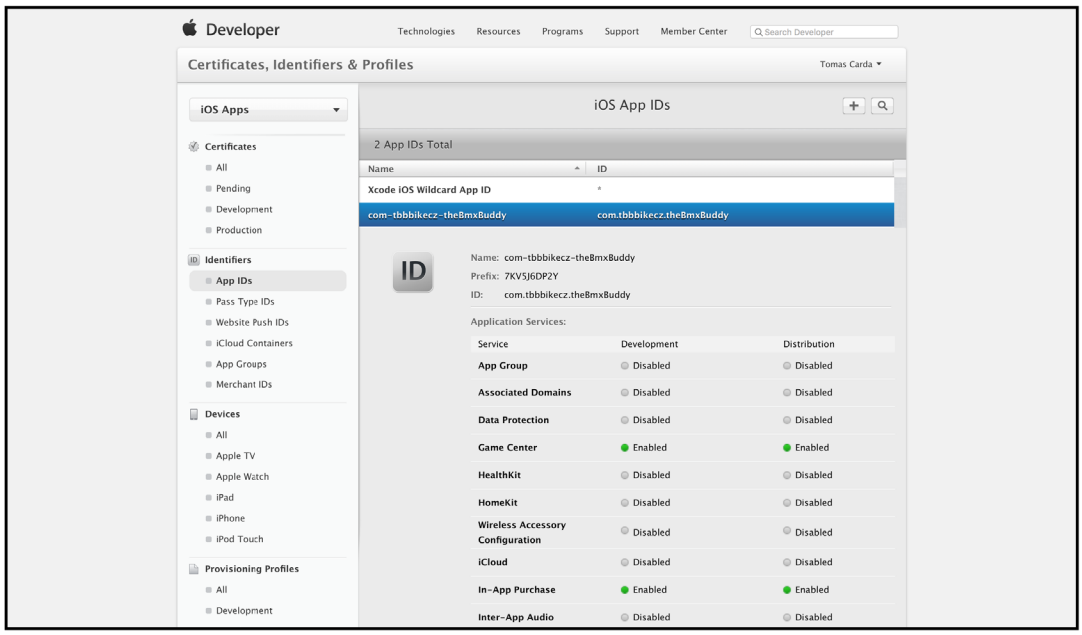
### 4.8.1 Certifikáty

Pokud je uživatel přihlášený do placeného Apple Developer Programu může ve webovém portálu developer.apple.com požádat o vytvoření digitálně podepsaného distribučního certifikátu, který je nutný pro umístění aplikace na App Store. Pro vygenerování tohoto certifikátu je potřeba využít systémovou aplikaci Klíčenka, která se stará o správu hesel a certifikátu na zařízení Mac. Pomocí Klíčenky je vygenerován soubor CertificateSigningRequest, kte-

rý je poté nutné nahrát do webového portálu. Pomocí tohoto požadavku je poté vygenerován podepsaný certifikát ios_distribution.cer, který po stažení a otevření umístí podepsaný certifikát do Klíčenky.

### 4.8.2 ID aplikace a Provisioning Profile

Ve webovém portálu je nutné zaregistrovat ID aplikace pomocí položek name a Bundle Identifier (App ID Sufix), které nalezneme v projektu aplikce v Xcode. Dalším krokem je vytvoření distribučního provisioning certifikátu, který umožňuje distribuci aplikace přes App Store či Ad-hoc[1]. Po vytvoření je potřeba tento certifikát opět stáhnout a nainstalovat do Xcode.



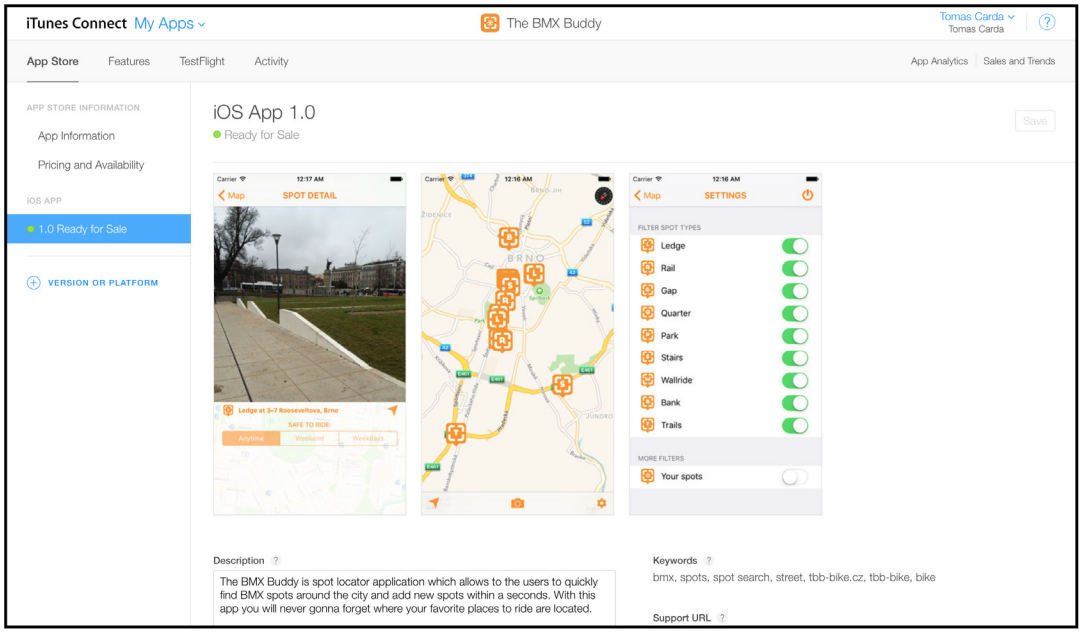

Obr. 13 - Webový portál pro Apple developery

### 4.8.3 Založení aplikace v iTunes Connect

Posledním krokem je založení profilu aplikace v prostředí iTunes Connect. Tento profil slouží jako distribuční kontejner pro nahrání aplikace na portál iTunes, ze kterého se aplikace stahují. Po vyplnění formulářů s údaji o

1 Způsob přímé distribuce přes soukromý server či přímo přenesením instalačních dat aplikace. Aplikace není umístěna na App Store.

aplikaci je založen její profil ve kterém je nutné poté vyplnit všechny důležité informace o aplikaci. Tento proces je monitorován a v případě neúplnosti údajů je vývojář informován o chybjejících datech. Status profilu také informuje o akutuálním stavu a je třeba docílit změnu statusu z Prepare for Upload na Waiting for Upload.




Obr. 14 - Prostřední iTunes Connect

### 4.8.4 Nahrání aplikace ke schválení

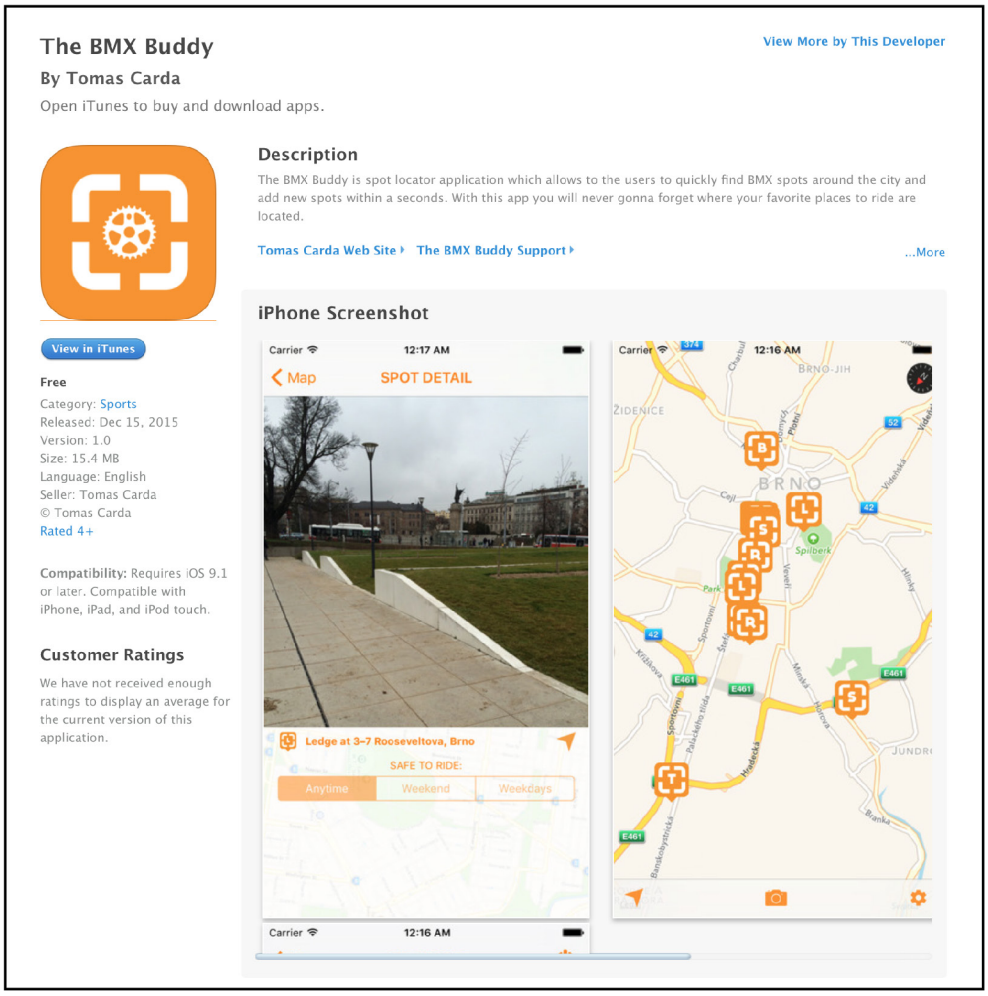
Jakmile jsou vytvořeny všechny potřebné profily a certifikáty je možné přistoupit k nahární aplikace do iTunes Connect. Aplikace v Xcode nesmí obsahovat chyby, musí mít vyplněné všechny základní údaje v profilu aplikace a musí obsahovat všechny ikony a splash screen[1]. Pokud aplikace splňuje všechny tyto požadavky může dojít k archivaci aplikace pro upload.

Z menu Product stačí vybrat položku Archive, která se postará o archivaci aplikace. Tento archiv je poté možné nalézt v okně Organizer, kde jsou zobrazeny všechny archivy dané aplikace. Požadovaný archiv lze poté vybrat a pokud jsou všechna nastavení v pořádku také nahrát na App Store.

1 Obrazovka zobrazená po spuštění aplikace. Tvoří zástupné okno, které nahrazuje aplikaci počas jejího spouštění.

### 4.8.5 Validace

Po úspěšném nahrání lze nalézt tento archiv v profilu aplikace na iTunes Connect. Nyní aplikace čeká na ruční ověření, které může v závislosti na vytíženosti týmu, který se stará o testování nových aplikací trvat až 10 dní. O procesu ověřování je vývojář informován prostřednictvím emailů a změnou statusu aplikace v iTunes Connect. Pokud dojde k schválení aplikace je aplikace umístěna na App Store, kde je dostupná volně ke stažení.




Obr. 15 - The BMX Buddy na App Store

# 5. Tvorba propagačního videa

Video je v současnosti jednou z nejúspěšnějších metod marketingu. Propagační videa zaplavují internetové stránky, sociální sítě, reklamní plochy a samozřejmě televizi. Ve formě explainer videí[1] navíc také slouží k rychlé prezentaci funkcionality daného produktu, jeho předností, možností využití a jiných vlastností. Proto bylo pro účely propagace aplikace The BMX Buddy vytvořeno propagační video, jehož účelem je zaujmout a oslovit diváka ke stažení aplikace do svého mobilního zařízení.

Jelikož cílová skupina pro aplikaci The BMX Buddy je velmi specifická a tvoří jí pouze BMX jezdci, kteří mají zájem se podílet na tvorbě této sítě spotů, je specifické i video pomocí kterého je aplikace prezentována. Inspirací pro toto video bylo především lifestyle video značky OSS[2] a propagační video doplněné o informační grafiku pro kampaň Modré Brno[3].

Tvorba audiovizuálního díla je obvykle rozdělena do tří fází kterými jsou pre-produkce, produkce a post-produkce. Tyto fáze však nemohou fungovat samostatně a naopak je žádoucí, aby byly navzájem provázány. [10]

## 5.1. Pre-produkce

Pre-produkcí se rozumí prvotní fáze plánování, která obsahuje sestavení námětu, scénáře, storyboardů, tvorbů harmonogramů, sestavení rozpočtu, herceké obsazení, použitou techniku a další úkony nutné pro následnou úspěšnou produkci. Již v této fázi musí být ale brán zřetel na další fáze, na stanovení reálných termínů a vyhrazení dostatečného rozpočtu na jednotlivé fáze. Této fázi be měl být věnován dostatek pozornosti, jelikož chyby v pre-produkci mohou vést k fatálním chybám v dalších fázích.

### 5.1.1 Námět

Jezdec projíždí městem a uživá si proplétání ulicemi a mezi kolemjdoucími lidmi. V průběhu této projížďky provádí triky, které podtrhují jeho pocit z jízdy městem. Jezdec si sedá, vytahuje telefon a hledá v aplikací místo, kam by se vydal jezdit. Tato interakce s mobilním telefonem je doplněná o informační grafiku, která podtrhuje prováděnou operaci.

---

1 Krátké video sloužící k vysvětlení produktu či služby.

2 BROC RAIFORD | ONSOMESHIT 2014 na webových stránkách www.youtube.com/watch?v=p3ItkzbWTCs&app=desktop

3 Modré Brno na webových stránkách www.youtube.com/watch?v=ZoCg88UE3VE

Vyráží na místo, projíždí městem a v průběhu si fotografuje spot, který cestou objevil. Ukládání nového spotu je rovněž doplněno o informační grafiku. Tyto interakce s mobilním zařízením jsou ale v tuto chvíli potlačeny a hlavním bodem zájmu je jezdcovo projíždění městem, jelikož zatím nechceme odhalit podstatu videa. Jezdec poté přijíždí na místo, které si našel a předvádí obtížný trik, kterým video vrcholí.

Video končí leteckým záběrem, který odkrývá pomocí informační graky podstatu aplikace a do záběru jsou zakomponovány ikony spotů. Záběr poté plynule přechází do čistě grafické části, která představuje samotnou aplikaci, kde je na trojrozměrném modelu mobilního zařízení ukázána aplikace a vedle tohoto modelu jsou typograficky prezentovány hlavní přednosti a informace o dostupnosti této aplikace na App Store.

### 5.1.2 Výběr postav a lokací

Za hlavní postavu propagačního videa byl vybrán Michal Šmelko, který je nejúspěšnějším českým jezdcem, známým nejen v České republice ale i ve světě. Michal je navíc týmovým jezdcem společnosti TBB-bike.cz pro kterou je tato aplikace určena a která je se svou dvacetiletou tradicí největším prodejcem BMX komponentů v České republice a na Slovensku. TBB-bike.cz navíc sídlí v Brně a proto bylo toto město vybráno za lokaci pro natáčení. Dalším důvodem je velké množství kvalitních spotů v této moravské metropoli.

## 5.2 Produkce

Natáčení tohoto propagačního videa probíhalo jeden den na dvou lokacích v Brně. Pro realizaci byla použita digitální zrcadlovka[1] Canon, set objektivů s různými ohnisky[2] 8 mm, 16 mm, 24-70 mm, zařízení pro stabilizaci obrazu Flycam, stativ, slider[3] a dálkově ovládaná quadrocoptéra[4] DJI Phantom osazená kamerou GoPro Hero 4.

Pro účely tohoto propagačního videa byly všechny záběry natáčeny s vysokou snímkovací frekvencí 60 fps[5] pro následné docílení zpomaleného pohybu. Záběry byly poté ve většině případů zpomaleny na polovinu původní rychlosti z důvodu atipického pohledu na danou scénu. Jedinou výjimkou je

1 Typ fotoaparátu, který je vybaven zrcadlem pro lom obrazu směrem do hledáčku.
2 Vzdálenost čočky objektivu od jejího ohniska. Společně s velikostí senzoru určuje zorný úhel zachycené scény.
3 Mechanický pojezd pro plynulý pohyb kamery po horizontální ose.
4 Dálkově ovládaný model se čtyřmi vrtulemi.
5 Frames per second - počet snímků, který kamera zaznamená za jednu sekundu.

závěrečný záběr z quadrocoptéry, který ja natočen standradní snímkovací frekvencí 30 fps, ale naopak oproti ostatním záběrům je natočen do 4K[1] rozlišení, které umožňuje následné lepší zpracování při post-produkci.

## 5.3 Post-produkce

Post-produkční fáze obsahuje vše od střihu, tvorby zvukových a vizuálních efektů, titulků, barevných korekcí a dalších procesů vedoucích k finálnímu dílu. Post-produkční fáze by měla být brána vždy v potaz v průběhu pre-produkční fáze, kdy musí být zohledněny náklady a časová náročnost post-produkce a v pruběhu produkce, kdy by mělo být dohlíženo na správnou přípravu materiálu. [10]

### 5.3.1 Střih

Střihem je označován postup skladby jednotlivých filmových stop za sebe a vytvoření z těchto úseků ucelený celek. Název této činosti je odvozen od původní metody, při které docházelo k reálnému střihu filmového pásu a následného spojování těchto části. V současné době se ke střihu používá softwarových nástrojů, které umožňují střih a další práci se surovým filmovým materiálem v digitální podobě. Mezi nejčastěji používaný software patří střihové programy Adobe Premiere, Sony Vegas či Final Cut pro Mac.

Pro střih propagačního videa pro aplikaci The BMX Buddy byl využit software Adobe Premiere, který navíc umožňuje snadnou komunikaci s dalšími programy z balíku Creative Cloud, které jsou využívány v průběhu post-produkčního procesu.

### 5.3.2 Vizuální efekty

Tvorba vizuálních efektů slouží primárně k vytvoření uričté iluze v daném snímku. Do snímku mohou být zakomponovány další prvky, jako kresby, fotografie, 3D objekty či informační grafika a na tvůrcích těchto efektů je co nejvěrohodněji tyto objekty zakomponovat do scény. Hlavní vlastností úspěšného VFX[2] specialisty je především schopnost řešit problémy, jelikož každá práce je ve většinou specifická a je nutné k ní přistupovat odlišně. [12]

Pro tvoru vizuálních efektů existuje celá řada programů a VFX specialisti většinou používají jejich kombinaci, jelikož každý z nich se hodí pro spe-

1 Dvojnásobné rozlišení oproti standardnímu rozlišení FullHD. Snímky mají rozlišení 4096 × 2304 pixelů.

2 Zkratka pro Visual Effects - vizuální efekty.

cifický úkon a má určité přednosti. Existují speciální programy na komponování scén mezi které patří například Nuke nebo Shake, které patří mezi špičkuv této oblasti. Na trhu je celá řada programů umožňujících tvorbu 3D grafiky jako například 3ds Max, Maya, Blender či Cinema 4D a každý z těchto programů je využíván v určitých odvětvích, kde projevuje své přednosti.

Pro tvorbu vizuálních efektů pro účely propagačního videa The BMX Buddy byl využit software Adobe After Effects, který je rozsálým nástrojem pro komponování, tvorbu efektů, barevnou korekci a tvorbu informační grafiky. Právě z důvodů přílišné rozsáhlosti tohoto softwaru není příliš využíván v profesinoální filmové produkci, kdy je využíváno specifických programů na jednotlivé úkony, ale je dokonalým nástrojem pro středně velké produkce zaměřující se na tvorbu propagačních videií. Z toho důvodu patří After Effects mezi nejrozšířenější post-produkční nástroj.

Pro přípravu 3D grafiky pro tento projekt byl použit program Cinema 4D, který patří v současnosti mezišpičkové programy využívané při produkci reklamních spotů a vzhledem k jeho zaměření je podporována jeho kooperace s programem After Effects pro následné komponování a od verze After Effects CC je dokonce součástí tohoto balíku ve verzi Lite[1]. Vzhledem k provázanosti programů Adobe Premiere, Adobe After Effects a Cinema 4D je kombinace tohoto softwaru hojně využívána reklamními produkcemi.

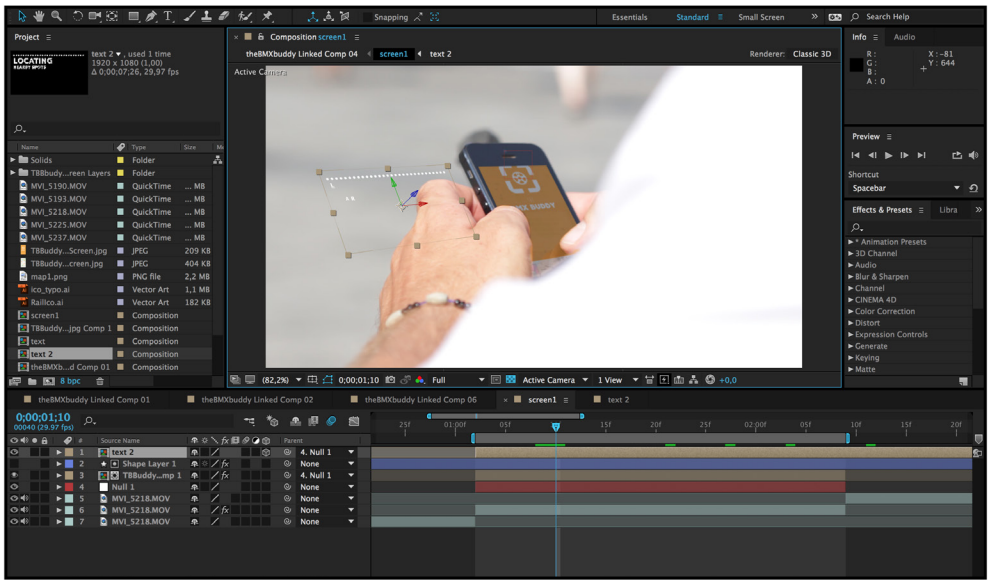
Obr. 16 - Pracovní prostřední programu Cinema 4D

1 Verze s omezenou funkcionalitou oproti standardní verzi.

#### 5.3.2.1 Informační grafika

V případě propagačního videa The BMX Buddy je na třech místech použita informační grafika doplňující daný záběr o dodatečné informace. V prvním případě informuje o použití mobilního telefonu k lokalizaci spotů v uživatelově okolí a v druhém případě infromuje o přidání nového spotu do databáze The BMX Buddy a v třetím případě jde o zasazení ikon do finálního leteckého záběru.

Pro první scenu bylo nutné provést trackování[1] záběru, kterým byly získány informace o pohybu mobilního zařízení. Jelikož automatické trackovací metody v případě této scény selhaly a byly nepřesné, muselo dojít ke kombinaci automatického a ručního trackování pro získání přesnějšího výsledku. Výsledná data o pohybu byla poté použita pro zakomponování grafických prvků do scény.




Obr. 17 - Komponování grafiky v prostředí Adobe After Effects

U této scény je také na displej mobilního telefonu umístěno grafické rozhraní aplikace, jelikož samotné rozhraní bylo v původním záběru prakticky nerozlišitelné. Obrazovka je do scény umístěná jako vrstva v trojrozměrném prostoru pro dodržení správné perspektivy. K zakomponování byla poté

1 Metoda sledování pohybu objektů ve scéně.

použita získaná trackovací data, tak aby obrazovka přesně sledovala pohyb mobilního telefonu. Jelikož je obrazovka ve scéně překrývána jezdcovo rukou muselo dojít k rotoscopingu[1] pohybu této ruky a dalších částí, které displej překrývaly.

Informační grafika umístěná ve scéně byla připravena jako samostatná scéna, která obsahuje průbeh animace této grafiky. Tato scéna pak byla umístěna do scény jako v trojrozměrném prostoru stejně jako u obrazovky mobilního telefonu, tak aby dodržovala správnou perspektivu a pomocí trackovacích dat byl umožněn stejný pohyb grafiky jako mobilního telefonu pro docílení iluze připoutání grafiky k zařízení.

Stejným způsobem je informační grafika zakomponována i do druhé scény. Automatické trackování v tomto případě také selhalo a vzhledem k složitému pohybu bylo nutné provést manuální trackování. Vrstva s grafikou byla poté umístěna do trojrozměrného prostoru do správné perspektivy a poté byla v průběhu scény manuálně upravována rotace této vrstvy pro dodržení správné perspektivy.

#### 5.3.2.2 3D model

Závěrečná scéna propagčního videa končí zakomponováním leteckého záběru do obrazovky trojrozměrného modelu mobilního zařízení iPhone 6. V grafickém softwaru je tento model simulován v trojrozměrném prostředí, aby mohl následně vypočítat viditelné body ze kterých je tvořen výsledný obraz. Model je tvořen body, spojení dvou bodů vytváří hrany a uzavření těchto hran tvoří polygony, ze kterých je vytvořen povrch modelu. Modely mohou být v těchto programech tvořeny pomocí těchto objektů nebo pomocí primitivních těles jako jsou krychle, koule a další jednoduchá tělesa. K vytváření komplexních a organických tvarů pak může docházet metodou dělení těchto těles a jejich vyhlazování, která je známá jako subdivision surface modeling.

Objektům mohou být poté přiřazeny materiály a textury[2], které simulují materiály reálného světa. Do scény mohou být umístěna světla a kamery, které reagují s prostředím a v kombinaci těchto prvků může být docíleno i naprosto věrohodné fotorealistické simulace. [13]

Pro potřeby tohoto propagačního videa byl vybrán volně dostupný model mobilního telefonu iPhone 6 [Zdroj: c4depot.com]. Tento model byl poté v programu Cinema 4D nasvícen a byla vytvořena kamera pro účely ná-

1 Metoda při které je snímek po snímku vytvářená maska, která sleduje a zakrývá určité části scény. Tím je docílené realistické komponování prvků do scény.

2 Bitmapový obrázek nanesený na povrch objektu.

sledné animace. Animováním pozice modelu v prostoru byl vytvořen dojem oddálění obrazovky mobilního zařízení od kamery, čím byla vytvořena iluze běhu videa na tomto mobilním zařízení. Dále byly některým objektům scény přiřazeny speciální značky (tagy), které následně umožňují práci s těmito objekty v kompozičním programu.

Posledním krokem před začátkem komponování v After Effects je vyrenderování této scény. Proces renderování umožňuje vykreslit tato obrazová data a uložit je v podobě obrazové sekvence či video souboru na výstup. [16]

Pro účely další práce a komponování byl v tomto případě vygenerován i After Effects projekt, který obsahuje všechny podstatné informace o scéně jako je umístění kamer, světel a dalších objektů.

Finální fází je komponování závěrečné scény v After Effects. Data o kameře a poloze obrazovky mobilního zařízení umožňují nahradit obrazovku záběrem z quadrocoptéry a následně rozhraním aplikace The BMX Buddy. Scéna je nakonec doplněna o typografickou animaci s informacemi o aplikaci.

### 5.3.3 Finální úpravy

Po dokončení práce na vizuálních efektech jsou provedeny finální úpravy celého videa. Je provedena barevná korekce, která umožňuje vizuální sjednocení všech záběrů a video je v konečné fázi vyexportováno v cinemascope[1] formátu s poměrem stran 2.35:1. Tento poměr stran tak dává videu více filmovější vzhled.

Obr. 18 - Finální scéna propagačního videa The BMX Buddy

1 Extrémně širokoúhlý formát zavedený společností 20th Century Fox.

# 6. Závěr

Cílem této práce bylo seznámit čtenáře s procesem vývoje mobilní aplikace The BMX Buddy pro mobilní zařízení iPhone za využití vývojového prostředí Xcode a programovacího jazyka Swift a následné tvorby propagačního videa s využitím moderních postupů v této oblasti.

Po uvedení aplikace na App Store byla aplikace uvedena na webových stránkách společnosti TBB-bike.cz kde se setkala s velkým ohlasem a v současné chvíli má přes 300 uživatelů nejen v České republice ale i zahraničí.

Jelikož aplikace si získala oblibu u cílové skupiny na kterou mířila její vývoj bude nadále pokračovat. V návaznosti na tuto první verzi aplikace vzniká prostor pro přidání nové funkcionality, úpravu grafického rozhraní či migraci této aplikace na jiné platformy

V následující fázi dojde k vytvoření další verze, která poskytne podporu i operačnímu systému iOS 8.0. Po ověření funkcionality poté bude aplikace prezentována i na zahraničních webových portálech, pro získaní většího počtu zahraničních uživatelů.

## Literatura


[1] MILLER, BJ. Swift in 24 Hours, Sams Teach Yourself. Vyd. 2. Sams Publishing, 2014. 448 str. ISBN-13: 978-0-672-33724-6

[2] iPhone Market Share [online], aktualizace 1. 10. 2015, [cit 2. 1. 2016], Bgr. http://bgr.com/2015/10/01/iphone-market-share-q3-2015-android/

[3] App Store Apps [online], aktualizace 1. 1. 2016, [cit 2. 1. 2016], Statista. http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[4] Apple Inc. App Programming Guide for iOS. Apple Inc, 2015. 118 stran.

[5] VAN HECKE, William. Learning iOS Design: A Hands-On Guide for Programmers and Designers. Vyd. 1. Pearson Education Inc, 2013. 317 str. ISBN-13: 978-0321887498

[6] RAY, John. iOS 5 Application Developement in 24 Hours, Sams Teach Yourself. Vyd. 2. Sams Publishing, 2012. 785 str. ISBN-13: 978-0-672-33576-1

[7] PITRE, Boisy G. Swift for Beginners: Develop and Design. Vyd. 1. Peachpit Press, 2015. 304 str. ISBN-13: 978-0-134-04470-5

[8] BIRANI, Bhanu. Application Development with Parse using iOS SDK. Vyd. 1. Packt Publishing Ltd., 2013. 304 str. ISBN 978-1-78355-033-3

[9] VÁVRŮ, Jiří. iPhone vývoj aplikací. Vyd. 1. Grada Publishing a.s., 2012. 192 str. ISBN 978-80-247-4457-5

[10] Postproduction hand book, A guide to finishing your film. Vyd 1. 2009. 108 str.

[11] Navigation Controller [online], akt. 15. 11. 2014, [cit 2. 1. 2016], Apple. https://developer.apple.com/library/ios/documentation/WindowsViews/Conceptual/ViewControllerCatalog/Chapters/NavigationControllers.html

[12] GRESS, Jon. [digital] Visual Effects and Compositing. Vyd. 1. Grada Publishing a.s., 2015. ISBN 13: 978-0-321-98438-8

[13] KERLOW, Isaac. Mistrovství 3D animace. Vyd. 1. Computer Press a.s., 201. ISBN: 978-80-251-2717-9

## Příloha

Na přiloženém flash-disku jsou umístěny všechny zdrojové soubory použité při vývoji aplikace a při realizaci propagačního videa aplikace The BMX Buddy. Flash disk je rozdělen do tématických složek usnadňující orientaci v této příloze.