**ZÁPADOČESKÁ UNIVERZITA V PLZNI**
**FAKULTA ELEKTROTECHNICKÁ**

**KATEDRA ELEKTROENERGETIKY A EKOLOGIE**

# BAKALÁŘSKÁ PRÁCE

## Aplikace pro mobilní zařízení pro výpočet uhlíkové stopy s využitím GPS

**Vladimír Krichťák** **2014**

*Originál (kopie) zadání BP/DP*

## Abstrakt

Předkládaná bakalářská práce se zabývá tématem vývoje aplikací pro mobilní zařízení, zejména pak pro Windows Phone, pro jejich využití při ochraně životního prostředí.

## Klíčová slova

Windows Phone, mobilní zařízení, vývoj mobilních aplikací, uhlíková stopa, C#, XAML, MVVM

## Abstract

The bachelor's thesis deals with a topic of development applications for mobile devices especially for Windows Phone and how it can be used as way of protecting the environment.

## Key words

Windows Phone, mobile device, mobile application development, carbon footprint, C#, XAML, MVVM

# Prohlášení

Prohlašuji, že jsem tuto diplomovou/bakalářskou práci vypracoval samostatně, s použitím odborné literatury a pramenů uvedených v seznamu, který je součástí této diplomové práce.

Dále prohlašuji, že veškerý software, použitý při řešení této bakalářské/diplomové práce, je legální.

.....

podpis

V Plzni dne 9.6.2014 Vladimír Krichťák

# Obsah

# Seznam symbolů a zkratek

API ..... Application Programming Interface

$CO_2$ ..... oxid uhličitý

CLR ..... Common Language Runtime

GPS ..... Global Positioning System

GUID ..... Globally unique identifier

IDE ..... Integrated Development Environment

JSON ..... JavaScript Object Notation

LINQ ..... Language-Integrated Query

MVC ..... Mode-View-Controller

MVVM ..... Model-View-ViewModel

OS ..... Operační systém

SDK ..... Software Development Kit

WPF ..... Windows Presentation Foundation

XAML ..... Extensible Application Markup Language

XML ..... Extensible Markup Language

# Úvod

Předkládaná práce se zabývá problematikou vývoje takových mobilních aplikací, které jsou využitelné při ochraně životního prostředí. Je zaměřená na sledování emisí oxidu uhličitého, takzvané uhlíkové stopy, produkovaných zařízeními se spalovacími motory. Zároveň popisuje vyvinutou aplikaci, která je součástí této práce v digitální podobě pod názvem ECOlog.

Text práce je rozdělen do tří částí. První část popisuje nejpoužívanější mobilní platformy a zaměřuje se na zařízení s operačním systémem Windows Phone. Dále pak popisuje možnosti vývoje pro tuto platformu a používané technologie, které byly využity při tvorbě aplikace ECOlog.

Druhá část vysvětluje, co je uhlíková stopa a co ji způsobuje. Zároveň popisuje způsoby jak redukovat produkci nežádoucích skleníkových plynů, kde je nejvíce zastoupený právě oxid uhličitý.

Ve třetí části je pak popsaná aplikace ECOlog, která díky využití systému GPS, zaznamenává produkci oxidu uhličitého, který je produkován dopravním prostředkem se spalovacím motorem. Obsahuje jak detailní popis architektury aplikace, tak i postup, kterým byla vyvíjena. Zároveň obsahuje krátký popis použitých knihoven a rozšíření třetích stran využitých v aplikaci.

# 1 Vývoj aplikací pro mobilní zařízení

Mobilní zařízení v kontextu této práce jsou elektronická přenosná zařízení s vlastním napájením a vlastním operačním systémem specializovaným pro tato zařízení. Zejména se jedná o mobilní telefony.

## 1.1 Přehled nejběžněji používaných platforem

Současný trh s mobilními zařízeními ovládají především tyto platformy:

- **Android**

Android je v současnosti nejrychleji rostoucím operačním systémem pod vedením společnosti Google. Popularitu si u uživatelů získal především nízkou cenou zařízení používajících tento OS. U vývojářů potom velice mírnými podmínkami pro umístění aplikace do prodejního katalogu (v případě androidu do tzv. Google Play). Což způsobilo veliký nárůst aplikací včetně nekvalitních a nebezpečných.

- **BlackBerry OS**

Operační systém vyvinutý firmou Research In Motion speciálně pro mobilní zařízení společnosti BlackBerry. Tato platforma je známá především svými podnikovými funkcemi, zejména pak podporou firemní korespondence.

- **iOS**

Stále ještě nejrozšířenější OS pro mobilní zařízení. Používají jej zařízení firmy Apple. Poprvé se objevil na prvním zařízení iPhone, tehdy ještě pod názvem iPhone OS. Na rozdíl od jeho přímého konkurenta, platformy Android, je umístění aplikací do prodejního katalogu (v tomto případě App Store) složitější. Aplikace zde totiž procházejí schvalovacím procesem, tím je zaručeno, že se k uživateli dostanou bezpečné aplikace.

- **Java ME**

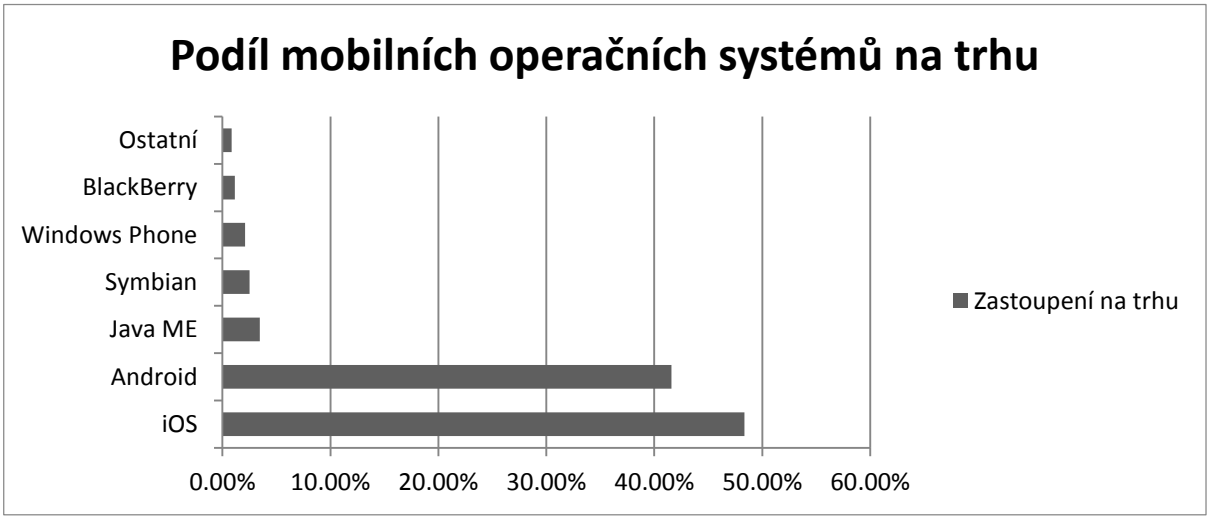
Java ME znamená Java Micro Edition a jedná se o jednu ze základních platforem založených na platformě Java. Je velice univerzální, na rozdíl například od iOS, nebo Windows Phone.

- **Symbian**

Kdysi velice populární systém, zejména na telefonech značky Nokia. Nyní je jeho další vývoj již ukončen.

- **Windows Phone**

Nejmladší z operačních systémů je nástupce systému Windows Mobile. Největší zásluhu na rozšíření toho systému má společnost Nokia, která jej začala používat namísto Symbianu. Windows Phone se vyznačuje především velice přísnými požadavky na hardware zařízení. Na druhou stranu je pro tento hardware optimalizovaný a běh tohoto systému je velice plynulý, čemuž zajisté napomáhá i jednoduché grafické rozhraní. Stejně jako Android a iOS má i Windows Phone svůj obchod s aplikacemi, nazývaný Windows Phone Store.




*Obr. 1.1 Podíl jednotlivých operačních systémů na trhu, květen 2014*[1]

## 1.2 Vývoj pro systém Windows Phone

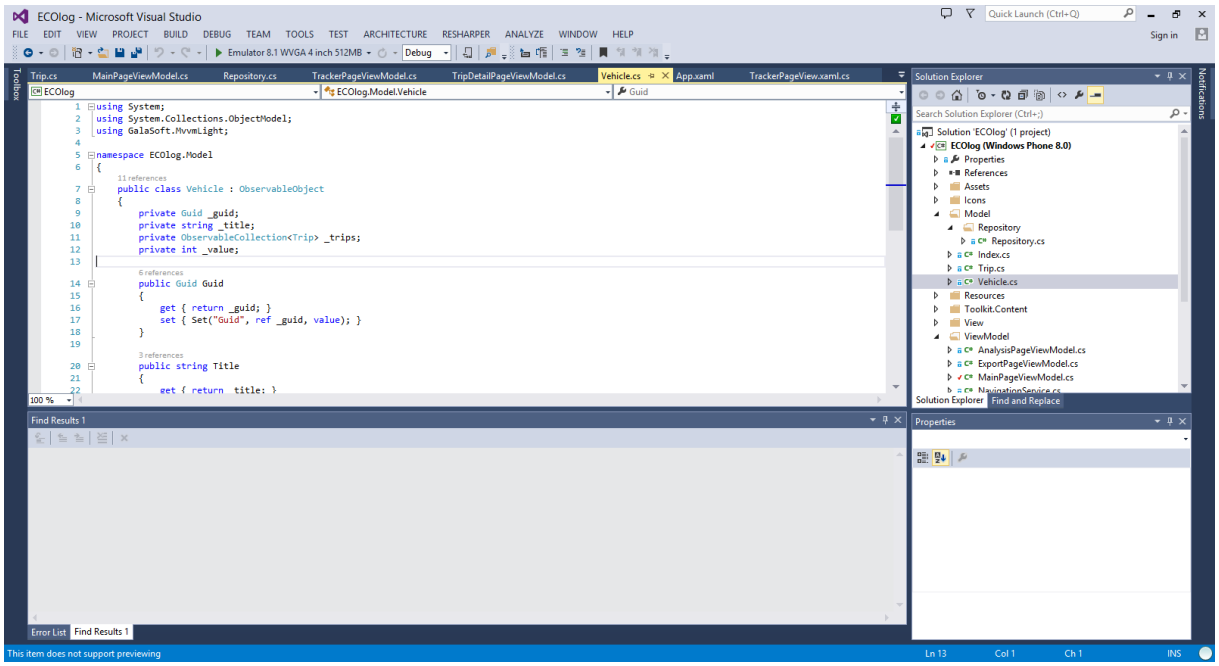
Společnost Microsoft poskytuje takzvaný Software Development Kit neboli SDK. Jedná se o sadu softwarových nástrojů a knihoven určených pro vývoj aplikací. Součástí je i odlehčená verze (Express) povedeného editoru, který se nazývá Visual Studio. V editoru je rovnou k dispozici emulátor mobilních zařízení a základní předdefinované šablony, jako například

[1] Podíl jednotlivých operačních systémů na trhu: Květen 2014. In: *Net market share: Market Share Statistics for Internet Technologies* [online]. 2009 [cit. 2014-06-09]. Dostupné z: https://www.netmarketshare.com

stránka na výšku, či na šířku. Specializované knihovny obsažené v SDK se nazývají API, neboli Application Programming Interface, tedy rozhraní pro programování aplikací. V dokumentaci se pak nachází popis tohoto rozhraní.

## 1.3 Visual Studio

Vývojové prostředí, takzvané IDE (Integrated Development Environment), neboli integrované vývojové prostředí, umožňuje efektivní vývoj aplikací díky připraveným nástrojům a editoru uzpůsobenému pro vývoj kódu. Dále obsahuje kompilátor a debugger.




*Obr. 1.2 Visual Studio*

## 1.4 Podporované technologie pro vývoj aplikací určených pro Windows Phone

Pro vývoj nativních aplikací jsou k dispozici technologie C++ a .NET.

- **C++**

Hlavní výhoda použití jazyka C++ v tomto případě Visual C++ 2012 je snadná portace aplikace na jednotlivé platformy. Dále, na rozdíl od .NET, umožňuje využití Direct3D.

- **.NET Framework**

.NET Framework je technologie, která podporuje vybudování a provoz aplikací nové generace. Je navržen tak, aby splňoval tyto požadavky:

- Poskytnout konzistentní objektově orientované vývojové prostředí ať už je kód uložen a spouštěn lokálně, spouštěn lokálně ale distribuovaný po internetu anebo rovnou spouštěn vzdáleně.
- Zajistit prostředí pro spuštění kódu, které minimalizuje požadavky na nasazování software a konflikty mezi jeho verzemi.
- Obstarat prostředí, které umožní bezpečné spuštění kódu, včetně kódu vytvořeného neznámými nebo ne úplně důvěryhodnými třetími stranami.
- Eliminovat problémy s výkonem, které se vyskytují u skriptů nebo interpretovaných prostředí.
- Umožnit využití zkušeností s frameworkem napříč širokým spektrem aplikací od programů založených na platformě Windows, přes mobilní aplikace, po webové služby.
- Provádět veškerou komunikaci splňující průmyslové standardy pro zajištění kompatibility s ostatními systémy.

.NET Framework se skládá takzvaného Common Language Runtime (CLR) a knihovny tříd.

Common Language Runtime si lze představit jako agenta, který spravuje kód v čase, kdy je spuštěn, a poskytuje služby jádra, jako je správa paměti, správa vláken a vzdálená komunikace. Zároveň také zajišťuje striktní typovost kódu a další požadavky na přesný kód, které podporují robustnost a bezpečnost. Koncept řízení kódu, je základním principem běhového prostředí.

Knihovna tříd je komplexní, objektově orientovaná kolekce znovupoužitelných typů, které lze využít pro vývoj aplikací od klasického příkazového řádku, či grafického uživatelského rozhraní, po XML webové služby.

Pro využití technologií .NET je možno vybrat si jazyk VB.NET nebo C#.

## 1.5 Jazyk C#

Jazyk C# je objektově orientovaný, vysokoúrovňový programovací jazyk. Syntaxe využívající složených závorek, je snadno čitelná pro každého, kdo je obeznámen s jazyky C, C++ nebo Java. Vývojáři, kteří mají zkušenosti s některým s těchto jazyků, jsou zpravidla schopni začít pracovat v C# ve velmi krátkém čase. Syntaxe jazyka C# zjednodušuje mnohé složitosti, které sebou přináší C++ a poskytuje mocné funkce, jako jsou hodnotové typy umožňující nabývat hodnoty NULL, výčty, delegáty na funkce, lambda výrazy, ale také umožňuje i přímý přístup do paměti. Dále podporuje generické třídy a metody, které zvyšují bezpečnost a výkon aplikací. Umožňuje využití iterátorů, pomocí kterých lze snadno implementovat vlastní chování při procházení kolekcí. Velice silným nástrojem obsaženým v jazyce C# jsou LINQ (Language-Integrated Query) výrazy, které vytváří silně typové dotazy.

Jako objektově orientovaný jazyk podporuje zapouzdření, dědičnost a polymorfismus. Všechny proměnné a metody, včetně metody Main, která je vstupním bodem aplikace, jsou zapouzdřeny v rámci definice třídy. Každá třída sice může dědit pouze od jediné rodičovské třídy, ale může implementovat libovolný počet rozhraní. Metody, které přepisují virtuální metody rodičovské třídy, vyžadují použití klíčového slova override, aby nedošlo k náhodnému předefinování. V jazyce C# je struktura (struct) jako odlehčená třída – je alokovaná na zásobníku, může implementovat rozhraní, ale nepodporuje dědičnost.

## 1.6 Windows Presentation Foundation

WPF je prezentační systém poslední generace pro vytváření grafického rozhraní. Jádro je nezávislé na rozlišení s vektorově orientovaným vykreslovacím enginem, který je uzpůsoben pro využití moderního grafického hardwaru. Součástí WPF je také komplexní sada aplikací pro vývoj, která mimo jiné umožnuje data binding, dále obsahuje Extensible Application Markup Language (XAML), ovládací prvky neboli kontrolky (Controls), 2D a 3D grafiku, animace styly, šablony, dokumenty, média a text. WPF je součástí .NET Framework a existuje jako podmnožina jeho typů, které jsou převážně umístěné v namespace System.Windows. Pro usnadnění vývoje obsahuje další konstrukce, které rozšiřují vlastnosti a události. Jedná se o tak zvané dependency properties (závislé vlastnosti) a routed events (přidružené události).

# 2 Uhlíková stopa

Slovo stopa nám evokuje to, odkud pocházíme a kam směřujeme. Ale zatímco lidská stopa nabízí informace o velikosti, váze, nebo rychlosti, uhlíková stopa měří, jaké množství oxidu uhličitého produkujeme každý náš běžný den. Jízdou do práce, zapnutím světla, letem na dovolenou, spalováním fosilních paliv jako ropy, uhlí a plynu. Když fosilní paliva hoří, produkují skleníkové plyny jako $CO_2$, které přispívají ke globálnímu oteplování.

## 2.1 Jak vzniká uhlíková stopa a co to je

*"Uhlíková stopa je měřítkem dopadu lidské činnosti na životní prostředí a na klimatické změny. Uhlíková stopa se zaměřuje na množství skleníkových plynů, které produkujeme naším každodenním životem, například spalováním fosilních paliv pro výrobu elektřiny nebo tepla, dopravou atd. Vyjadřuje se v ekvivalentech oxidu uhličitého ($CO_2$), udává se v hmotnostních jednotkách – kilogramech a tunách. Jednoduše řečeno, uhlíková stopa je množství oxidu uhličitého a ostatních skleníkových plynů, uvolněné během životního cyklu produktu, který používáme, poskytnutím služby, cestováním, a podobně."*[2]

Jinými slovy: Když člověk řídí auto, motor spaluje palivo, které vytváří určité množství oxidu uhličitého, v závislosti na složení paliva a délce jízdy. Když člověk vyhřívá svůj dům naftou, plynem nebo uhlím, tak produkuje oxid uhličitý. Dokonce, i když vyhříváme náš dům elektrickou energií, její výroba může produkovat určité množství $CO_2$. Každé jídlo, nebo zboží, které kupujeme, bylo vytvořeno se vznikem $CO_2$.

## 2.2 Způsoby jak zredukovat skleníkové plyny

- Jíst lokální potraviny, vegetariánské nebo organické jídlo, snížit množství konzumovaného hovězího masa, zvýšit konzumaci kuřecího masa.

- Chodit pěšky, jezdit na kole, využívat hromadnou dopravu cestovat společně nebo používat ekologicky šetrné automobily.

- Malé domy potřebují méně energie.

[2] Uhlíková stopa. In: *Uhlíková stopa* [online]. 2010 [cit. 2014-06-08]. Dostupné z: http://www.uhlikovastopa.cz/

- Výměna klimatizací v komerčních budovách za solární odrazové materiály.
- Využívání nízkotlakých sprch a kohoutků.
- Vypnutí počítače, televize a dalších elektrických zařízení když nejsou používány.
- Vybrat si energeticky efektivní osvětlení.
- Nakupovat produkty s minimální obalem.

# 3 Aplikace ECOlog

Název je tvořen slovy ECO, z anglického ecological neboli ekologický, a log, což znamená záznam. A jak název vypovídá, jejím hlavním účelem je zaznamenávání ekologických následků jejího uživatele, přesněji zaznamenávání produkce oxidu uhličitého vyprodukovaného při cestování dopravním prostředkem, který využívá spalovací motor. Je zaměřena především na snadné a rychlé použití, aby uživatele zbytečně neobtěžovala. Před cestou stačí vybrat dopravní prostředek a spustit záznam. Během cesty se uživateli přehledně zobrazuje jeho průměrná rychlost, vzdálenost, kterou urazil, množství vyprodukovaného $CO_2$ a čas strávený na cestě. Zároveň na mapě vidí svoji trasu. Při ukončení cesty a zastavení záznamu má uživatel možnost tyto údaje, včetně souřadnic, uložit do svého mobilního zařízení. ECOlog zároveň tyto záznamy vyhodnocuje a přehledně prezentuje. Pro ukládání dat je zvolen formát JSON, který umožňuje snadné parsování a další zpracování těchto dat. Všechny své záznamy si uživatel může nahrát přímo z aplikace do svého cloudového uložiště OneDrive.

## 3.1 Architektura aplikace

Aplikace ECOlog je vystavěna na architektuře MVVM neboli Model-View-ViewModel. Základem pro MVVM byla MVC tedy Model-View-Controller.

- **MVC**

  Účelem mnoha aplikací je načtení dat z úložiště a zobrazit je uživateli. Poté, co uživatel provede změny v datech, musí aplikace tato data zpracovat a opět uložit. Vzhledem k tomu, že dochází k toku informací mezi úložištěm dat a uživatelským rozhraním, může mít vývojář sklony k tomu, aby svázal tyto dvě části dohromady, ať již kvůli tomu, nebo aby snížil množství tříd a kódu, nebo aby zlepšil výkon aplikace. Nicméně tento zdánlivě přirozený přístup má několik nezanedbatelných problémů. Jedním z nich je, že změny v uživatelském rozhraní jsou zpravidla častější, než změny v systému ukládání dat. Další problém s přímým spojením dat a uživatelského rozhraní je, že aplikace zpravidla obsahují logiku, která obstarává více než jen pouhý přenos dat.

  Zpravidla je potřeba zobrazit stejná data různými způsoby přesto, že se aplikace nezmění. Jednou například jako tabulku hodnot, jindy je potřeba zobrazit tato data ve formě grafu. Jedná se tedy o různé pohledy na stejná data. Pokud však dojde ke změně například v tabulce, je žádoucí, aby se tato změna provedla i v příslušném grafu. Je tedy potřeba

vyřešit problém, jak v reálném čase zobrazit tyto změny ve všech pohledech. Dále je běžnou praxí, že na jedné aplikaci pracuje současně tým vývojářů a je výhodnější mít specialisty na vzhled uživatelského rozhraní, kteří nemají tak hluboké technické znalosti jako například vývojáři aplikační logiky, ale zase jsou zdatnými grafiky.

Řešením těchto problémů je právě použití architektury MVC. Model-View-Controller odděluje datový model od prezentace dat a akcí na základě vstupů od uživatele do tří samostatných skupin.

- Model má na starosti uchovávat, aktualizovat a poskytovat data.
- View se stará o zobrazení požadovaných informací.
- Controller pak zpracuje požadavky uživatele, o změně v datech informuje Model a zároveň upraví zobrazovaná data.

Je důležité si uvědomit, že jak View tak Controller přímo závisí na modelu. Avšak model nezávisí ani na jednom z nich. Toto je jednou z hlavních výhod tohoto oddělení, neboť toto rozdělení umožňuje vystavět model, který je nezávislý na prezentaci dat.

Výhody:

- Podpora více pohledů. Vzhledem k tomu, že View je odděleno od modelu, není mezi nimi žádná přímá závislost.
- Snadná změna vzhledu. Požadavky na uživatelské rozhraní mají tendenci provádět změny vzhledu častěji než například změny v modelu.

Nevýhody:

- Složitost. MVC představuje novou úroveň nepřímého přístupu a tím pádem vyžaduje komplexnější řešení problému. Také vyžaduje událostmi řízený kód v uživatelském rozhraní, což znesnadňuje ladění programu.

- **MVVM**

Architekturu MVVM vyvinula společnost Microsoft s ohledem na událostmi řízené programování, kterého využívá Windows Presentation Foundation, tedy WPF. Funguje tak,

že obsahuje pro každé View třídu uchovávající si stav aplikace (ViewModel). Toto View se pak dotazuje svého ViewModelu a přizpůsobuje podle něj ovládací prvky. A naopak, pokud uživatel zadá data v uživatelském rozhraní, tak se automaticky zpropagují zpět do patřičného ViewModelu.

### 3.1.1 Model

Model aplikace je velice jednoduchý, skládá se ze tří tříd: Vehicle, Trip, Index a Repository.

- **Vehicle**

Obsahuje interní identifikátor vozidla, který slouží například pro identifikaci v indexu. Tvoří jej GUID, což je Globally Unique IDentifier, česky pak globálně jedinečný identifikátor. Dále je v této třídě uložena kolekce cest, tedy kolekce instancí třídy Trip. Potom také základní informace o dopravním prostředku jako je jeho označení a produkce oxidu uhličitého v gramech na jeden kilometr. Stejně jako spotřeba, je tento údaj udáván výrobci automobilů.

- **Trip**

Uchovává kolekci souřadnic, po kterých se uživatel při dané cestě pohyboval, jeho průměrnou rychlost, vzdálenost kterou urazil, množství vyprodukovaného oxidu uhličitého a také čas a datum začátku a konce cesty.

- **Index**

Slouží k tomu, aby se při spuštění aplikace nemuseli vyhledávat a procházet všechny soubory s dopravními prostředky a jejich cestami. Uchovává tedy informace o názvu dopravního prostředku, GUID pro jeho snadné dohledání a jednoznačnou identifikaci. Podobně je index využit při analýze dat. Aby se nemuseli načítat všechny soubory s dopravními prostředky, jsou v indexu pro každé vozidlo uloženy údaje o jeho průměrné rychlosti, uražené vzdálenosti. Dále pak je zde uložena doba strávená na cestách a celkové množství vyprodukovaného $CO_2$ tímto dopravním prostředkem. Samozřejmě obsahuje název souboru, ve kterém jsou serializována data. Při spuštění aplikace se pak pouze načte jeden soubor, ten se deserializuje na kolekci instancí třídy Index. Z těchto dat se sestaví menu, ze kterého uživatel vybere požadované vozidlo, a až po té se pro něj načte soubor

s daty, který se deserializuje na instanci třídy Vehicle obsahující kolekci cest. Tímto je celý start aplikace značně urychlen, obzvláště v případě, že má uživatel za sebou již větší množství cest.

- **Repository**

Třída Repository jako jediná část modelu neuchovává žádné další informace, ale slouží k načítání a ukládání ostatních objektů modelu.

### 3.1.2 View

View neboli pohled je uživatelské rozhraní. Je vytvořeno technologií Windows Presentation Foundation. Pro přístup k datům se v aplikaci využívá takzvaný binding, česky vazba, namísto defaultního code-behind (kód v pozadí), což právě umožnuje využití architektury MVVM. Tato jednotlivá View pak tvoří stránky aplikace.

- **AnalysisPageView**

Stránka obsahující kontrolku (Control) Pivot, která umožnuje mít na jedné stránce více podstránek a snadné listování mezi nimi. V tomto případě jsou to stránky s grafy ukazujícími uživatelovu produkci oxidu uhličitého.

- **ExportPageView**

Zde se nachází ovládací prvky k přihlášení ke cloudové službě OneDrive (dříve SkyDrive) kterou poskytuje Microsoft automaticky zdarma k uživatelskému účtu, nahrání a stažení souborů s daty, uloženými v univerzálním formátu json.

- **MainPageView**

Jak již název napovídá, toto je hlavní strana aplikace, tedy menu. Odtud může uživatel přejít na analýzu (AnalysisPageView), případně na stránku se správou svých cloudových záloh (ExportPageView). V menu je dále přehledně zobrazen seznam uživatelských dopravních prostředků s jejich dosavadní produkcí $CO_2$ a tlačítko pro přidání nového, po jehož stisku se zobrazí formulář (VehicleDetailPageView). Pokud uživatel vybere z menu požadovaný dopravní prostředek, je přesměrován na stranu se záznamem jeho cesty (TrackerPageView).

- **TrackerPageView**

Stránka tvořená kontrolkou Pivot. Má dvě podstránky. Na první se nachází mapa s tlačítkem pro spuštění záznamu. Pokud záznam trasy již běží, tak jsou zde mimo mapy zobrazeny i aktuální informace o cestě (produkce $CO_2$, uražená vzdálenost, průměrná rychlost a čas) a tlačítko pro ukončení záznamu. Na druhé podstránce je zobrazena historie cest.

- **TripDetailPageView**

V tomto pohledu jsou k dispozici informace o konkrétní cestě, která byla vykonána, tedy opět produkce $CO_2$, uražená vzdálenost, průměrná rychlost a čas, ale také datum a čas začátku a konce cesty.

- **VehicleDetailPageView**

Na této straně je uživateli zobrazen formulář, ve kterém vyplní název dopravního prostředku a jeho emise oxidu uhličitého v gramech na jeden kilometr. Tento rozměr byl vybrán záměrně, neboť jej udávají výrobci automobilů. Dále se zde nachází tlačítka pro uložení a smazání.

### 3.1.3 ViewModel

- **AnalysisPageViewModel**

Logika pro zpracování naměřených dat a jejich transformování pro zobrazení ve formě grafu. Dále pak zajišťuje výpočet nároků na odbourání celkové produkce zaznamenaného oxidu uhličitého a převádí je objem smrkového dřeva, který naváže jeho stejné množství. Podobné srovnání uvádí na případě vázaného oxidu uhličitého ve dřevě využitého na nízkoenergetický dřevěný dům.

- **ExportPageViewModel**

Obsluha tlačítek pro nahrání nebo staženi souborů s daty na úložiště OneDrive. A vyvolání příslušných akcí v Repository.

- **MainPageViewModel**

ViewModel k menu, nenachází se zde žádná složitější logika. Slouží hlavně pro obsloužení živých dlaždic z MainPageView. Také zajišťuje vyvolání formuláře pro přidání nového dopravního prostředku a přechod na stránky s analýzou a zálohováním zaznamenaných dat.

- **TrackerPageViewModel**

Zde se nachází nejvíce aplikační logiky, obstarává obsluhu senzoru GPS, sleduje změnu polohy uživatele a v reálném čase vypočítává čas, který uplynul od začátku měření, uraženou vzdálenost, kterou počítá ze souřadnic GPS, průměrnou rychlost a produkci oxidu uhličitého pomocí známého množství na jeden kilometr uvedeného u dopravního prostředku. Dále pak reaguje na události vyvolané výběrem záznamu z tabulky na podstránce s historií cest a události vyvolané tlačítky pro editaci a smazání dopravního prostředku.

- **TripDetailPageViewModel**

Pohled, na kterém je uživateli zobrazena mapa s absolvovanou cestou včetně data a času jejího zahájení a ukončení Dále pak obsahuje informace o době trvání této cesty, množství vyprodukovaného oxidu uhličitého, průměrné rychlosti a vzdálenosti, kterou uživatel urazil.

- **VehicleDetailPageViewModel**

Zajišťuje uložení dat po jejich editaci.

### 3.1.4 Další části aplikace

- **ViewModelLocator**

Třída sloužící jako Dependency Injection (DI) kontejner. To znamená, že pokud je v aplikaci potřeba instance ViewModelu, je dotázána tato třída, která ji dodá. Zde se tak děje v případě jednotlivých View. Každé má definované, který ViewModel využívá a pokud se toto View načte, je právě tento kontejner dotazován a poskytne onu instanci.

- **Resources**

Resources, česky zdroje, slouží k vyčlenění části aplikace mimo kód. Zpravidla se využívají pro textové řetězce, kdy lokalizace celé aplikace je řešená pouze načtením jiného zdroje. V aplikaci ECOlog jsou tyto řetězce uloženy v souboru AppResources. Dále je zde využito zdrojů pro definici cest k binárním souborům, jako jsou například obrázky. Tyto cesty se pak nacházejí v souboru FilesPathsResource. Přístup k těmto zdrojům poskytují třídy LocalizedStrings pro přístup k AppResources. V případě zdroje FilesPathsResource se jedná o třídu FilesPaths.

- **App**

Vstupní bod aplikace. Obstarává inicializaci a životní cyklus aplikace.

### 3.1.5 Rozšíření a knihovny třetích stran

- **AppBarUtils**

Jedná se o rozšíření standartní lišty aplikace umožňující například binding obrázků ikon z Resources.

- **Json.NET**

Povedená knihovna, která umožnuje nejen serializaci a deserializaci objektů do formátu JSON.

- **Live SDK**

SDK pro přístup a práci s cloudovou službou OneDrive.

- **MVVM Light Toolkit**

Sada nástrojů usnadňující využití MVVM. Obsahují například dependency injection container a předka pro ViewModely, který má metody usnadňující implementaci rozhraní INotifyPropertyChanged.

- **WPCharting**

Nástroje pro vytváření grafů. V aplikaci jsou využity pro grafické zobrazení naměřených hodnot.

## 3.2 Postup při vývoji

Ještě než jsem začal se samotným vývojem aplikace, rozmyslel jsem si její celkovou architekturu. Při zvážení všech výše zmíněných výhod modelu MVVM padla volba na něj.

Nejprve bylo nutné rozmyslet, jaká data je potřeba uchovávat a zpracovávat. Bylo tedy třeba navrhnout model. Samozřejmostí bylo vytvoření dopravního prostředku, u kterého je důležitá hodnota jeho emisí v závislosti na uražené vzdálenosti. Vznikla tedy třída Vehicle s vlastností pro uložení hodnoty produkce oxidu uhličitého na jeden kilometr. Dále bylo jasné, že uživatel musí poznat, se kterým dopravním prostředkem aktuálně pracuje. Třídě Vehicle, přibyla vlastnost Title pro uložení jeho názvu. Další informace, která se musí uchovat, je informace o uživatelových cestách. Samozřejmě se dalo předpokládat, že uživatel vykoná s vozidlem více než jednu cestu. Třída Vehicle musí obsahovat kolekci cest. Tím se dostáváme k další entitě a to je cesta, kterou popisuje třída Trip. Zde se musí uchovat informace o hodnotě vyprodukovaného oxidu uhličitého. Dále je každá cesta popsaná vzdáleností, kterou uživatel urazil, průměrnou rychlostí a časem který na ní strávil. Zajímavou informací také je, co to bylo za cestu a kde byl, proto se zde ukládá i kolekce souřadnic po kterých se pohyboval a datum jejího absolvování.

Když už bylo jasné, co se ukládá, bylo nutné tato data zobrazit. Začal jsem tedy s rozmýšlením jednotlivých obrazovek aplikace. Základem je menu, které uživatel uvidí ihned po spuštění aplikace. Pro větší přehled tvoří menu velké živé dlaždice, které zobrazují nejen název dopravního prostředku, ale i celkové množství vyprodukovaného oxidu uhličitého daným dopravním prostředkem. Bohužel ve standardních ovládacích prvcích se žádná podobná kontrolka nenachází, proto jsem ji musel vytvořit sám. Z menu musí mít uživatel možnost zvolit si dopravní prostředek, případně přidat nový. Tlačítka pro další ovládání aplikace jsem chtěl umístit na aplikační lištu. Zároveň jsem ale nechtěl cesty k ikonám psát přímo jako řetězec v pohledu. Mnohem přehlednější a pro případné další úpravy výhodnější je mít tyto informace uloženy ve zdrojích (Resources). Musel jsem využít rozšíření třetí strany a to AppBarUtils. Toto rozšíření umožňuje mimo jiné právě využít bindingu, v mém případě ze zdrojů, pro cestu k ikoně tlačítka. Na navigační lištu jsem umístil tlačítka, pomocí kterých má uživatel možnost spustit analýzu dat, jejich import a export pro zazálohování, případně další zpracování. Pro zálohu dat jsem zvolil cloudové úložiště OneDrive, protože je automaticky součástí live účtu, který uživatel potřebuje pro plnohodnotné využívání Windows Phone. Navíc služba live

poskytuje SDK pro práci s cloudem. Další pohled, který jsem vytvořil, byl právě ten, který se zobrazí po zvolení dopravního prostředku. Zde je ovládání záznamu cesty, které je tvořené mapou, výpisem zaznamenávaných hodnot a tlačítkem pro spuštění záznamu. Uživatele bude zajisté zajímat i historie jeho cest, proto jsem přidal podstránku pro jejich zobrazení. V případě, že by uživatel zjistil, že zadal špatně údaje o dopravním prostředku tak je zde tlačítko pro jeho smazání. Případně by mohl chtít tyto údaje pouze upravit. Vytvořil jsem tedy další obrazovku s formulářem sloužící k vyplnění údajů o vozidle. Bylo zbytečné mít dva stejné formuláře, proto se využívá tato obrazovka jak po stisknutí tlačítka pro editaci tak pro vyplnění údajů o nově přidaném vozidle. Abych zachoval jednotu systému, pro význačné barvy v aplikaci jako jsou živé dlaždice, barvy pozadí či různé zvýrazněné části, využívám akcentů, které si uživatel vybral v nastavení telefonu. Jsou tedy stejné jako barvy v systémových aplikacích a nabídkách.

Nyní jsem již věděl jaká data je potřeba zpracovávat a jak je zobrazit. Bylo tedy potřeba vytvořit aplikační logiku aplikace. Začal jsem vytvořením defaultních dopravních prostředků, které bude mít uživatel k dispozici po nainstalování aplikace. Díky tomu jsem mohl začít s výpisem hodnot pro menu. Kdy jsem si pro jednoduchost pomocí bindingu předal patřičnému View kolekci dopravních prostředků. Po zvolení jednoho z nich, bylo třeba přejít na stránku s ovládáním záznamu cesty. Toto má Microsoft vyřešeno velice nešťastně a to hned z několika důvodů. Jednak při MVVM je vcelku problém dostat se k ovládání přechodu mezi stránkami a vývojář je tlačen k tomu udělat to v code behind. To jsem obešel vytvořením třídy, která se jmenuje NavigationService a získá si přístup k takzvanému hlavnímu snímku aplikace. Dále se mi hodně nelíbí, že View, na které je potřeba přejít, musí být zadáno řetězcem jako cesta k souboru. Což mi přijde jako obrovský nesmysl, obzvlášť když má .NET dobře zpracovanou genericitu. Vyřešil jsem to tak, že jsem si vytvořil konstanty s cestami k souborům. Ty jsem umístil do dependency injection containeru. Ještě bylo potřeba vyřešit předávání informací mezi jednotlivými View. Když už jsem měl vyřešené i přecházení mezi stránkami, musel jsem se potýkat s problémem, jak si předávat informace při přecházení mezi jednotlivými pohledy. To byl jeden z důvodu, proč jsem se rozhodl využít rozšíření třetí strany a to sice MMVM Light Toolkit. Ten nejenže má již zpracované mechanismy pro vytvoření dependency injection containeru, tak navíc má připravenou abstraktní třídu jako předka ViewModelu. Ta pomáhá s implementací rozhraní INotifyPropertyChanged, které je nezbytné pro využití data bindingu a aktualizaci dat mezi View a ViewModelem. Přechod mezi stránkami je vyřešen tak, že dependency injection container poskytne instanci ViewModelu, na který se má přejít. Zde se nastaví předávaná informace jako vlastnost. Poté se pomocí NavigationService na tuto stránku

přejde. Nyní jsem již byl schopen přejít na stránku se záznamem cesty. Pro využití GPS senzoru, bylo nutné si v manifestu aplikace vyžádat povolení pro přístup k pozici uživatele. Dále jsem vytvořil patřičnou metodu pro výpočet trasy, vzdálenosti, rychlosti a produkce oxidu uhličitého v závislosti na změně polohy. Aby tyto výpočty probíhaly jen v případě, že se uživatel pohybuje, je tato metoda zaregistrovaná jako obslužná rutina při změně polohy. Obrazovka jenom s textovými výpisy vypočítaných údajů vypadala nudně, takže jsem přidal mapu se zobrazením uražené cesty a aktuální polohy. Control pro zobrazení mapy je udělaný opravdu špatně z hlediska využití MVVM a neumožnuje binding čáry, která znázorňuje cestu ani změnu středu mapy v závislosti na změně polohy. Po dlouhém výzkumu jsem nakonec rezignoval a obsluha mapy je jako jediná část v code-behind. Omezil jsem ji však na naprosté minimum a nezpracovává žádná data. Je zaregistrovaná jako samostatná rutina při změně polohy bez ohledu na výpočty a ukládané informace. Pro výpočet kurzu, kterým se uživatel pohybuje, je využito tříd, které jsem našel na stránkách MSDN Microsoftu s ukázkami kódu. Dále zbývalo získaná data analyzovat. Pro grafické znázornění hodnot je využito nástrojů WPCharting, které umožňuji elegantně zobrazit data ve formě grafů.

Aby bylo možné data zálohovat bylo nutné použít vhodný formát pro jejich uložení. Proto jsem zavrhl možnost využít databázi, jelikož by běžný uživatel nemohl snadno naměřená data z binárního souboru databáze zpracovávat. Zvolil jsem formát JSON, který je standardní, přehledný a umožňuje případné další zpracování napříč platformami. Pro serializaci a deserializaci je využito knihovny Json.NET. Informace o každém dopravním prostředku jsou ukládány do samostatných souborů. Proto, aby se při spuštění aplikace nemuseli všechny tyto soubory načítat a deserializovat, byla vytvořena třída Index, která obsahuje o každém dopravním prostředku základní informace a souhrnné statistiky. S vozidlem je spojena pomocí unikátního identifikátoru GUID. Samozřejmě nese informaci o tom, v jakém souboru je vozidlo uloženo. Kolekce všech indexů je uložena v jednom souboru ve formátu JSON. Při spuštění aplikace pak stačí načíst a deserializovat tento jeden soubor. Změnil jsem tedy binding v menu na kolekci entit třídy Index. Pro načítání a ukládání dat slouží třída Repository, která má na starosti jak serializaci a deserializaci, tak zápis a čtení souboru, i ukládání a stahování dat OneDrive.

Jako poslední krok mě čekala lokalizace. Vzhledem k tomu, že jsem veškeré textové řetězce umístil do Resources, tak mi pouze stačilo přidat nový zdroj a v nastavení projektu povolit další jazyk.

## 3.3 Závěr

Možnosti vývoje aplikací pro mobilní zařízení byly naznačeny v první kapitole. Zároveň zde byly objasněny technologie využívané pří vývoji pro platformu Windows Phone. Je zde popsáno jak WPF, které bylo v aplikaci využito pro vytvoření uživatelského rozhraní tak jazyk C#, který byl využit při vývoji logiky. Tyto nástroje jsou součástí rozsáhlého .NET Framework, jemuž se první kapitola věnuje také.

Mobilní zařízení jsou výbornými kandidáty, pro využití při sledování dopadu působení člověka na životní prostředí. Jejich hlavní výhodou je, kompaktnost při možnostech jakých nabízejí, jako například dostatek výpočetního výkonu, senzor GPS, dostatečně velký display pro zobrazování dat a mnohé další. Tyto výhody se dají využít pro záznam uhlíkové stopy produkované uživatelem při využívání dopravního prostředku se spalovacím motorem. Jak je ve druhé kapitole popsáno, uhlíková stopa popisuje dopad lidské činnosti na životní prostředí. Proto je vhodné ji sledovat a snažit se minimalizovat. S tímto se snaží uživateli pomáhat aplikace ECOlog, která byla vyvinuta v rámci této práce. Při vývoji aplikace jsem se zaměřil zejména na jednoduchost jejího používání. Chtěl jsem, aby ji uživatel mohl využívat v běžném životě. Z tohoto důvodu bylo rozhodující, její rychlé spuštění a minimalizace počtu kroků provedených uživatelem pro spuštění záznamu. Nezbytné zároveň bylo, sebraná data přehledně prezentovat a názorně vysvětlit potřebné kroky pro odbourání oxidu uhličitého vyprodukovaného uživatelem. ECOlog předkládá uživateli grafy zobrazující nejen jeho uhlíkovou stopu produkovanou všemi jeho dopravními prostředky, ale také vzdálenosti a průměrné rychlosti právě pro tyto dopravní prostředky. Zajímavým pohledem je i graf zachycující dobu jejich využívaní. Pro názornost jsou požadavky na odbourání uhlíkové stopy prezentovány jako objem smrkového dřeva, které by toto množství $CO_2$ bylo schopno na sebe navázat. Při dlouhodobém využívání aplikace uživatel ocení přepočet na množství dřevěných domů, jejichž množství dřeva je schopno navázat oxid uhličitý produkovaný uživatelem.

Ve třetí části je popsán způsob, jakým tato aplikace vznikala a problémy které musely být vyřešeny při vývoji pro mobilní zařízení s operačním systémem Windows Phone. Nejvíce jsem byl zaskočen překážkami, které musí vývojář překonat, pokud se chce držet architektury MVVM, která byla navíc přímo vyvinuta společností Microsoft stejně jako tento operační systém.

Windows Phone jako operační systém pro mobilní telefon je naprosto vynikající, odladěný a přehledný. Z těchto důvodů jsem si jej oblíbil pro osobní používání. Dalším plusem je možnost vývoje v jazyce C#, který patří k mým oblíbeným. Zároveň je pro něj oproti systémům iOS a Android mnohem méně aplikací, tedy menší konkurence pro vývojáře. Proto jsem pro vývoj aplikace zvolil právě Windows Phone. Ale opravdu mě překvapilo, jak nedomyšlené jsou nástroje pro vývojáře, jak jsem zmiňoval výše. Vzhledem k tomu, že je to mladá platforma, lze doufat, že se tyto neduhy do budoucna zlepší.

# Seznam literatury a informačních zdrojů

[1] Windows Phone Dev Center [online]. © 2014 [Cit. 1.6.2014]. Dostupné z: http://dev.windowsphone.com

[2] Center for Sustainable Systems, University of Michigan. 2013. *Carbon Footprint Factsheet.* Pub. No. CSS09-05.

[3] MSDN Library [online]. © 2014 [Cit. 1.6. 2014]. Dostupné z: http://msdn.microsoft.com/en-us/library/

[4] Windows Phone samples [online]. © 2014 [Cit. 1.6. 2014]. Dostupné z: http://code.msdn.microsoft.com/

[5] Podlahy a interiér [online]. © 2014 [Cit. 1.6. 2014]. Dostupné z: http://www.az-podlahy.cz

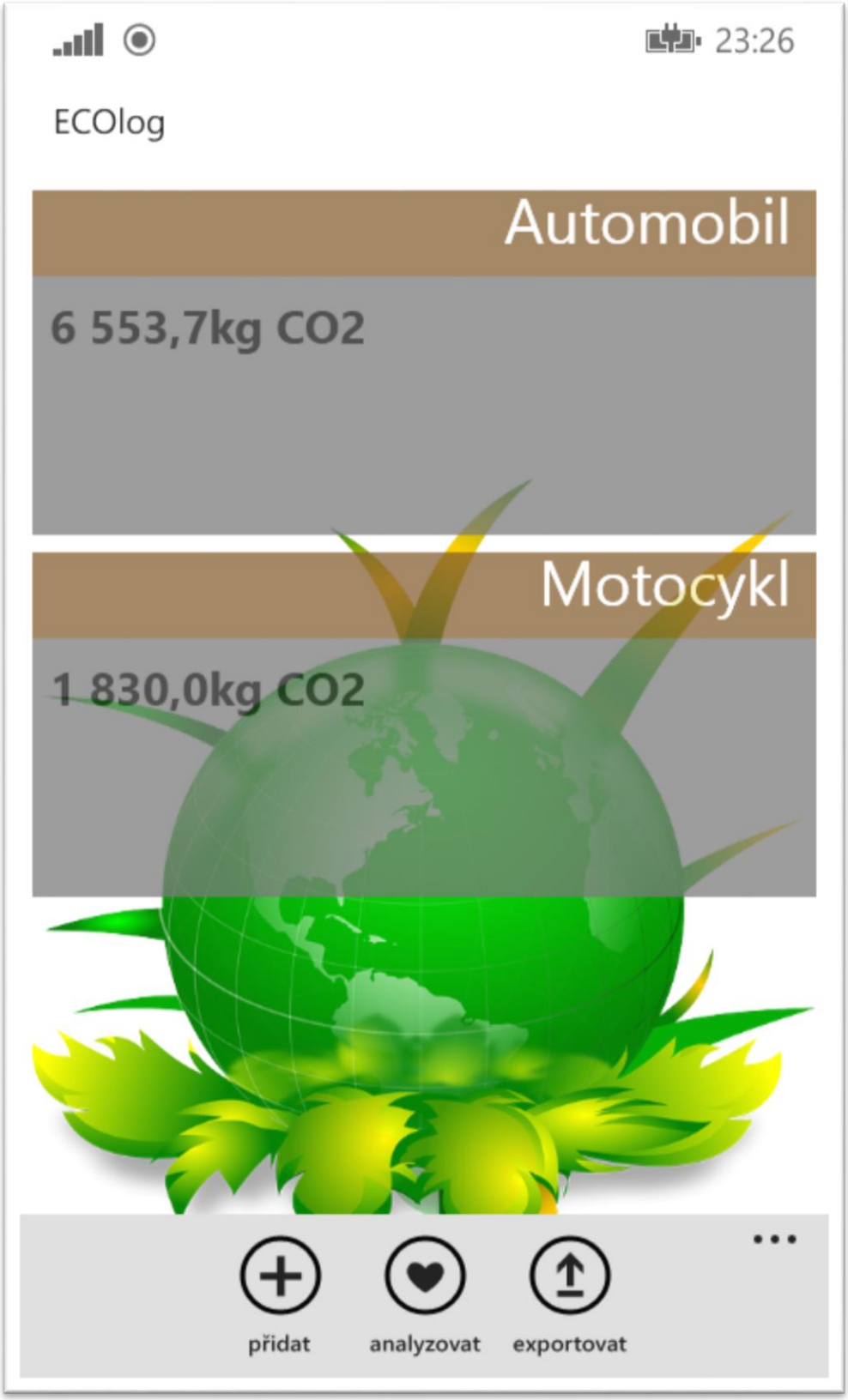
[6] Netmarketshare [online]. © 2014 [Cit. 1.6. 2014]. Dostupné z: https://www.netmarketshare.com

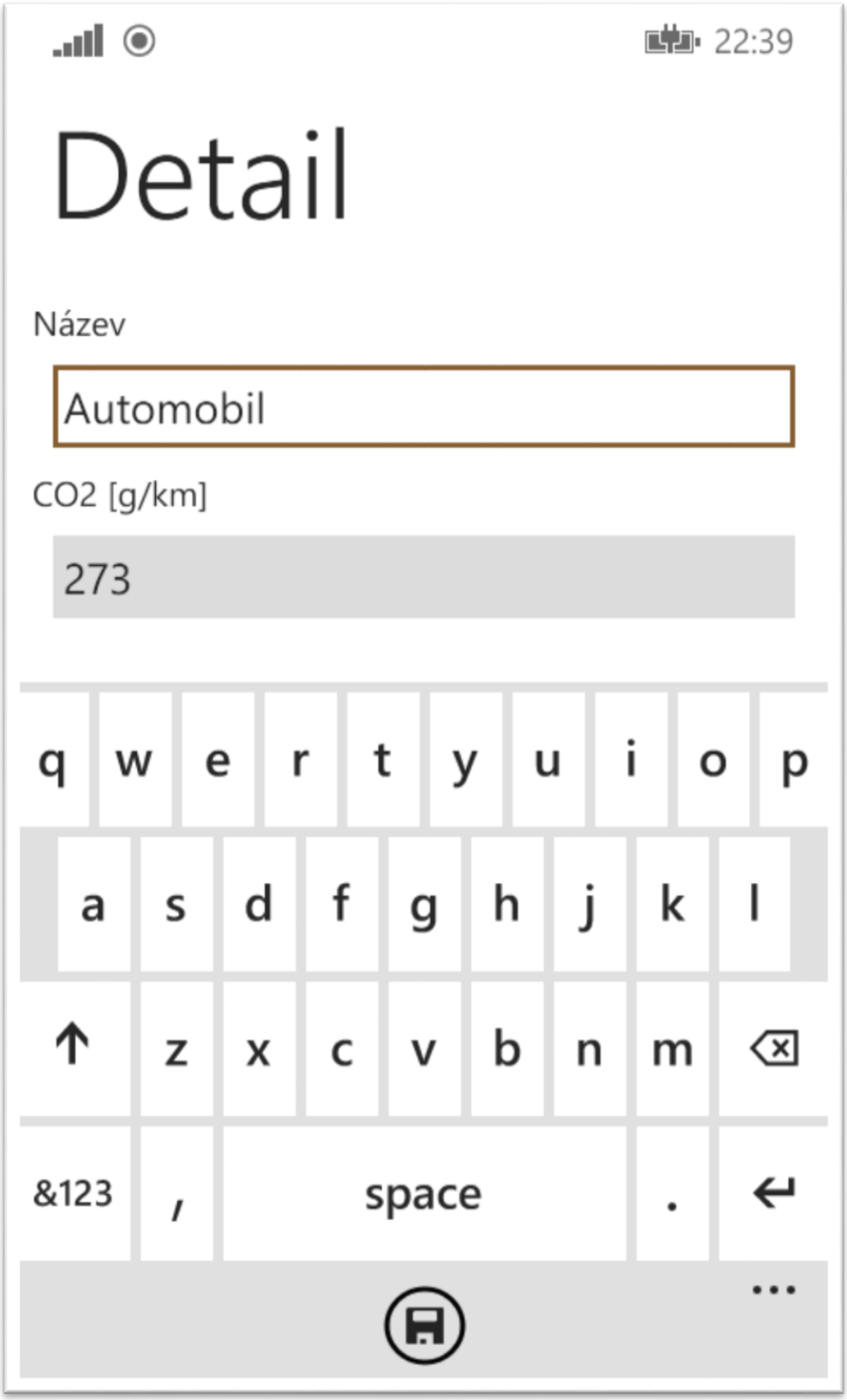
[7] Uhlíková stopa [online]. © 2014 [Cit. 1.6. 2014]. Dostupné z: http://www.uhlikovastopa.cz

[8] Hofman, P., Novák, V.: Zpracování exhalací a odpadů, Praha, Vydavatelství ČVUT, 2002, ISBN 80-01-02439-3.

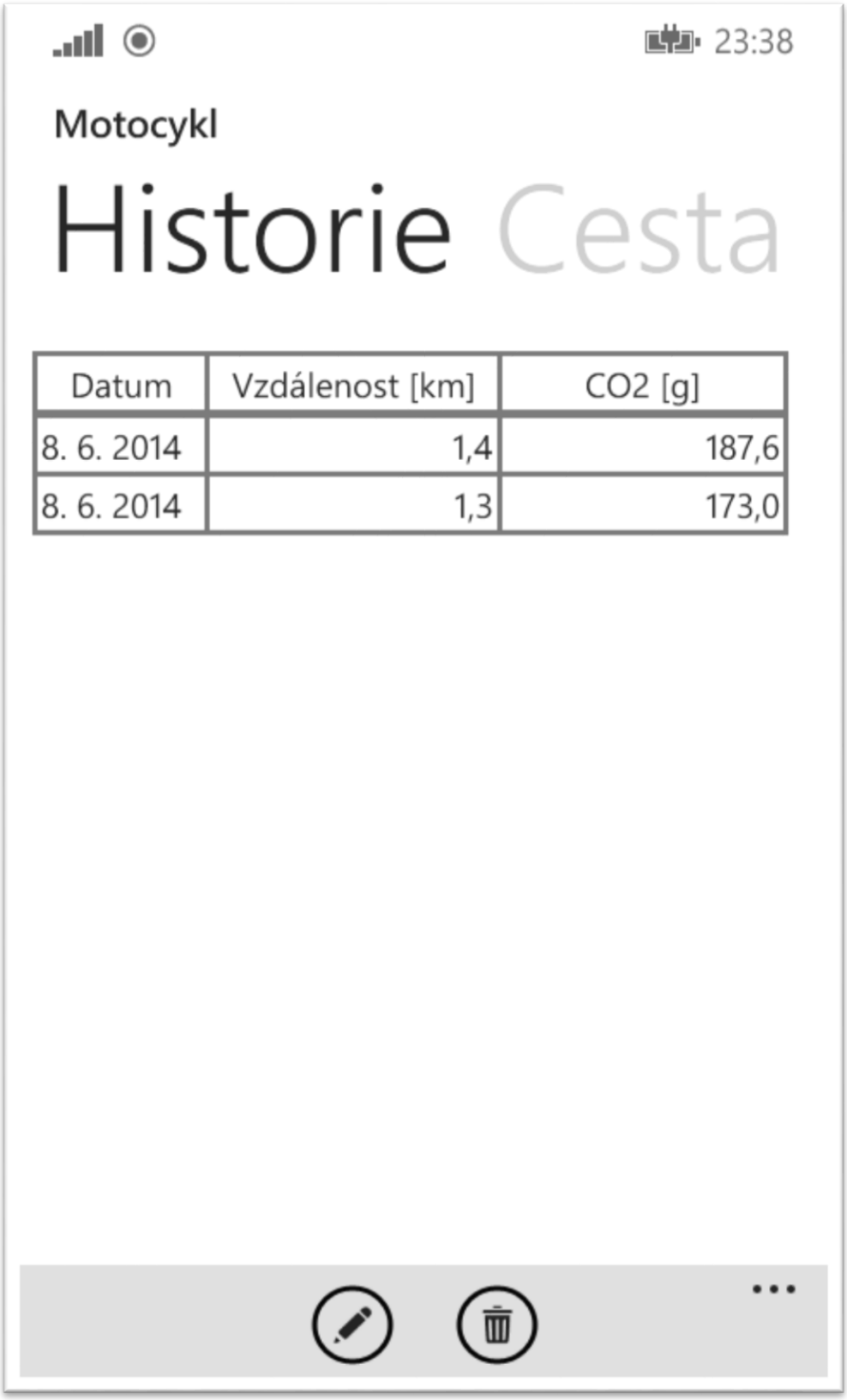
# Přílohy

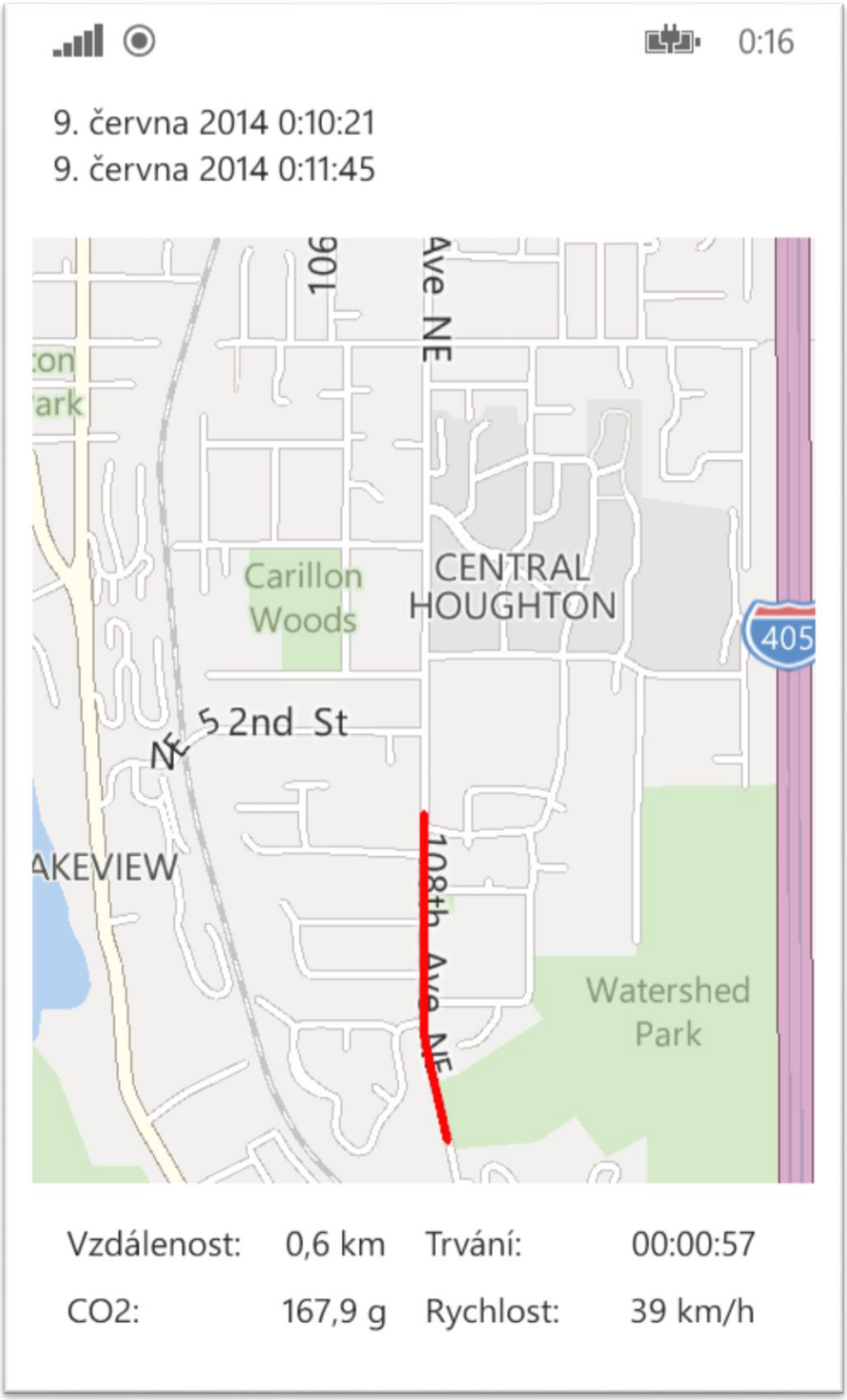
## Příloha A - Snímky obrazovek aplikace, Český jazyk, světlé téma vzhledu

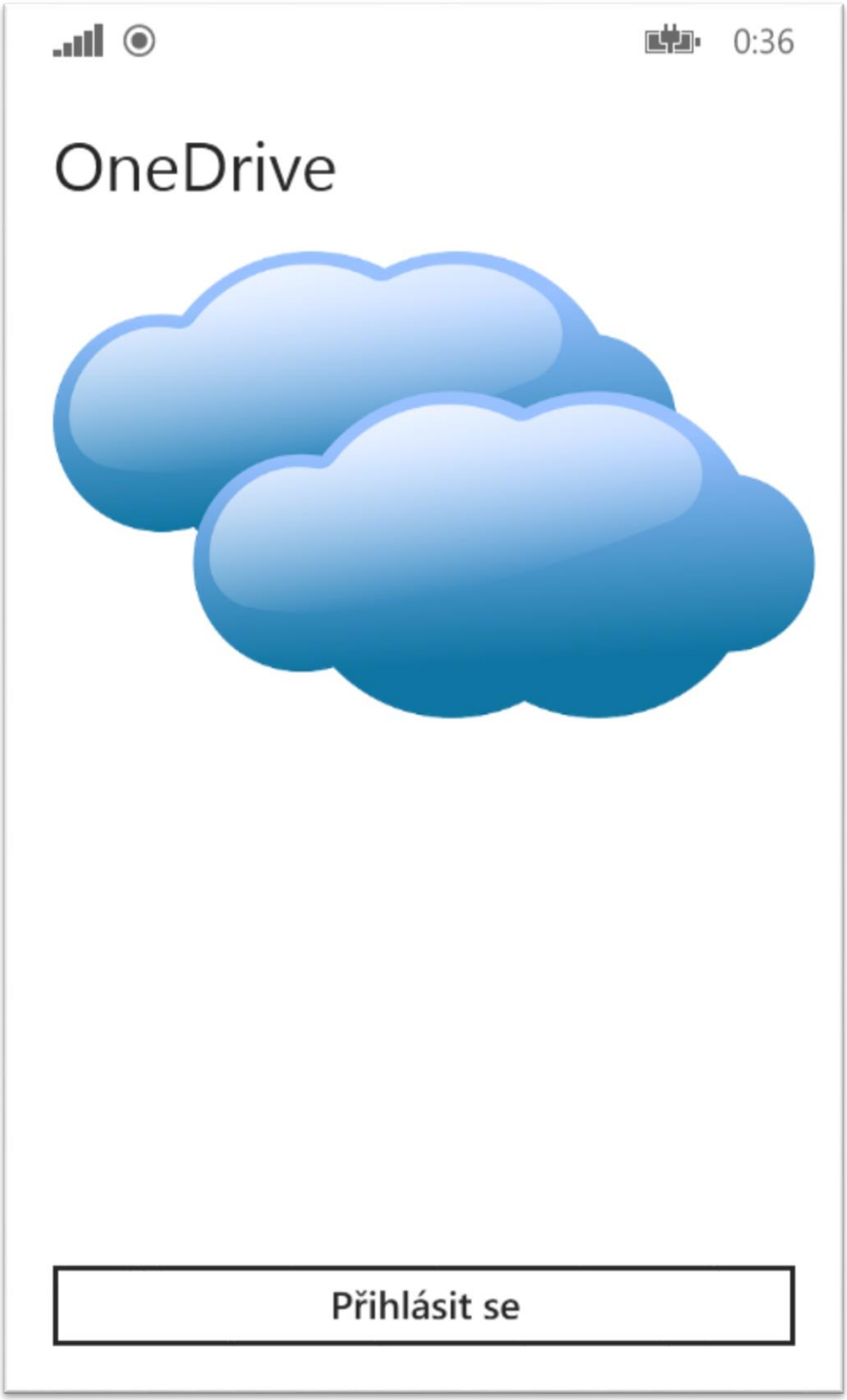


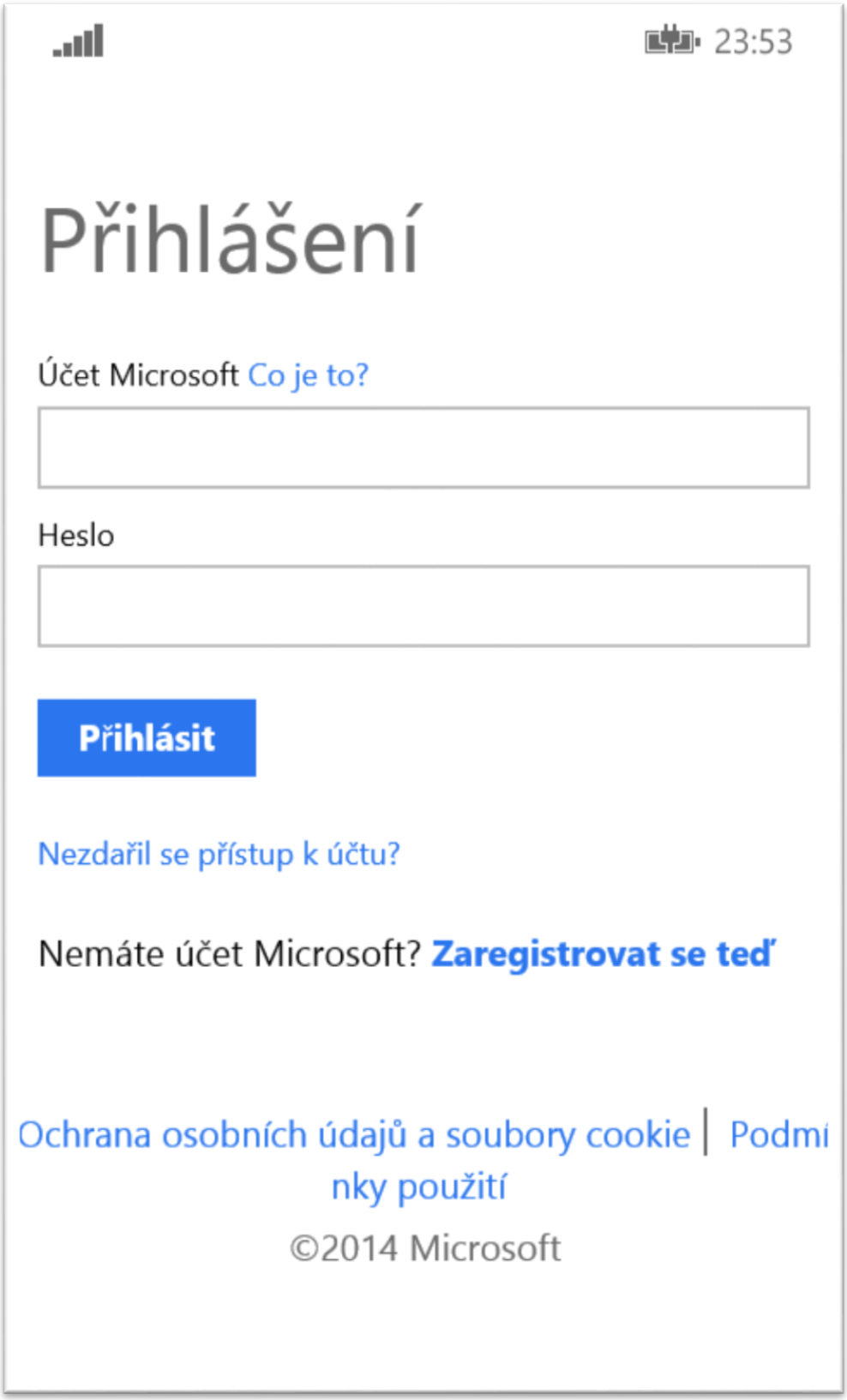

22:39
Detail
Název
Automobil
CO2 [g/km]
273
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
&123
space

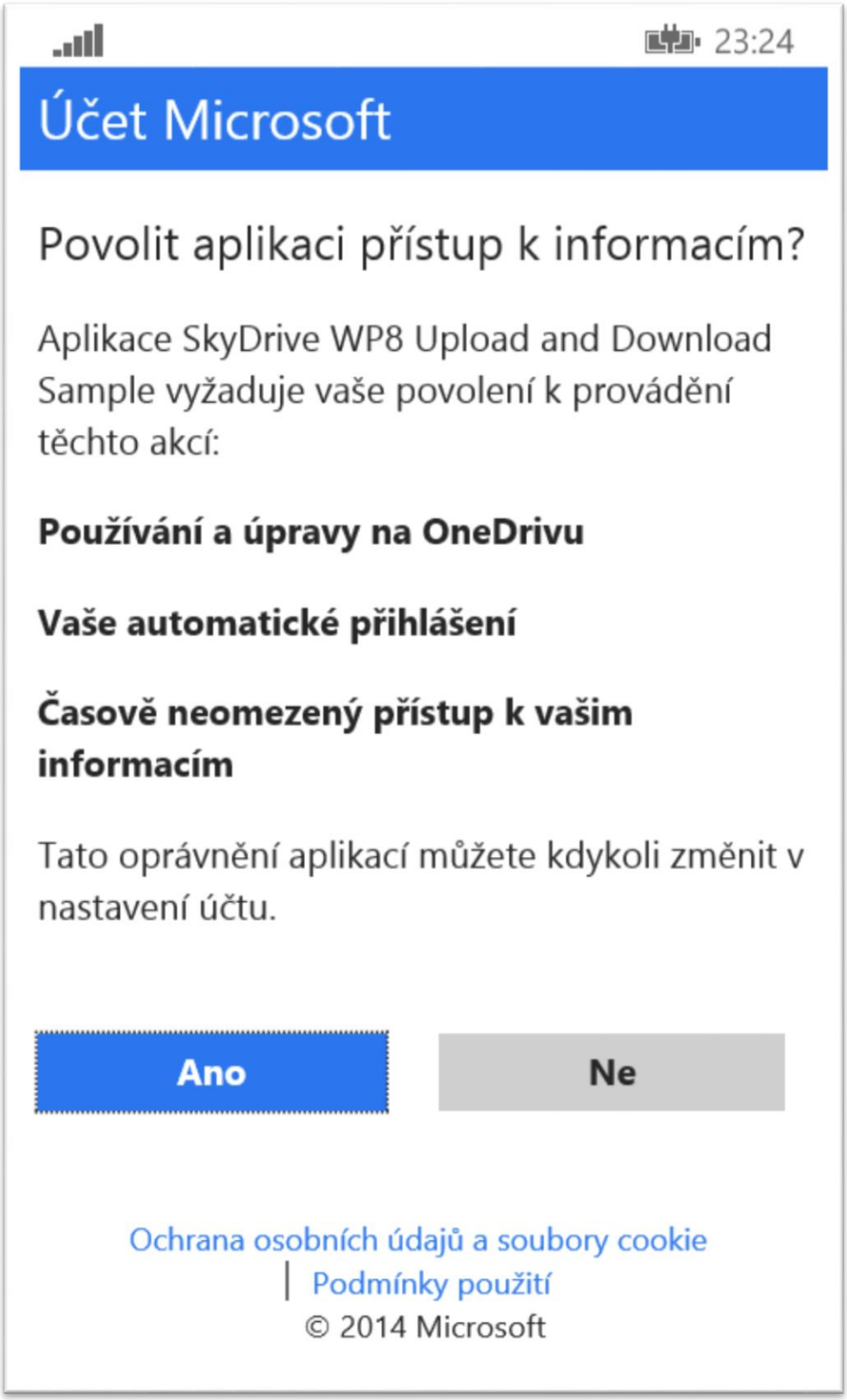

22:40
Automobil
Cesta Historie
Carbon River
Carbon River Rd
Start

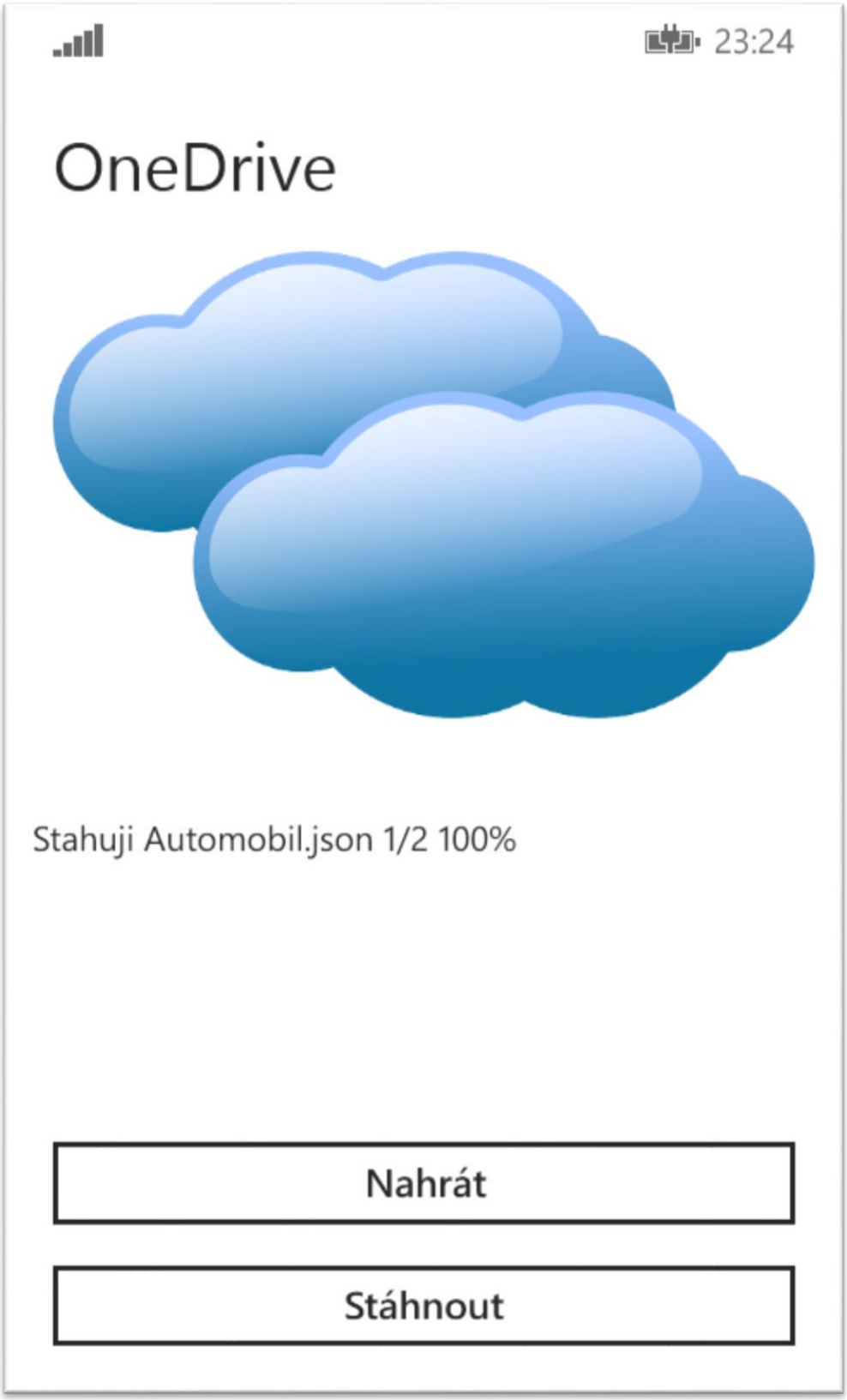

23:38
Motocykl
Historie
Cesta
Datum
Vzdálenost [km]
CO2 [g]
8. 6. 2014
1,4
187,6
8. 6. 2014
1,3
173,0

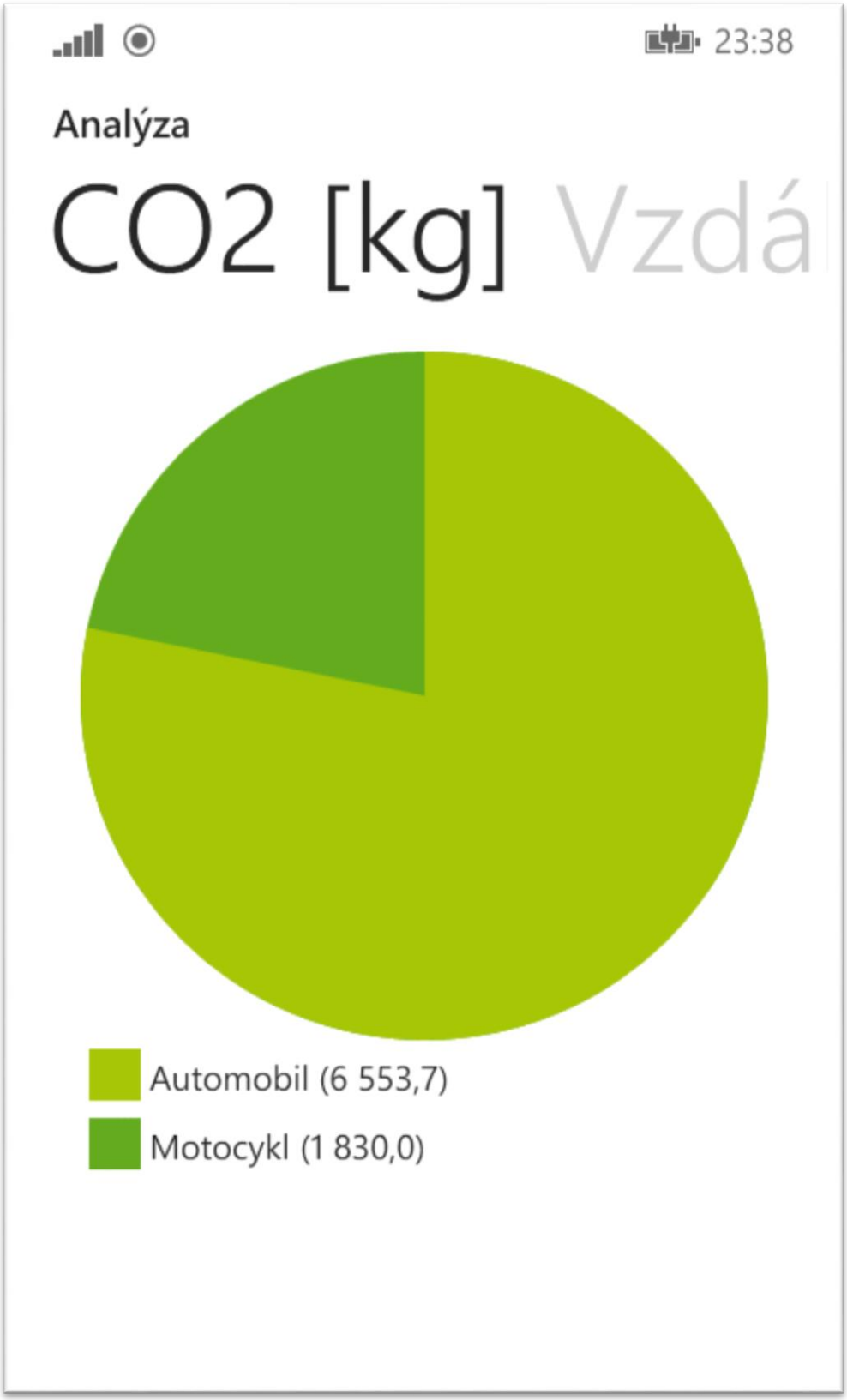

0:16
9. června 2014 0:10:21
9. června 2014 0:11:45
106
Ave NE
CENTRAL
HOUGHTON
Carillon
Woods
405
2nd St
AKEVIEW
108th Ave NE
Watershed
Park
Vzdálenost: 0,6 km
Trvání: 00:00:57
CO2: 167,9 g
Rychlost: 39 km/h

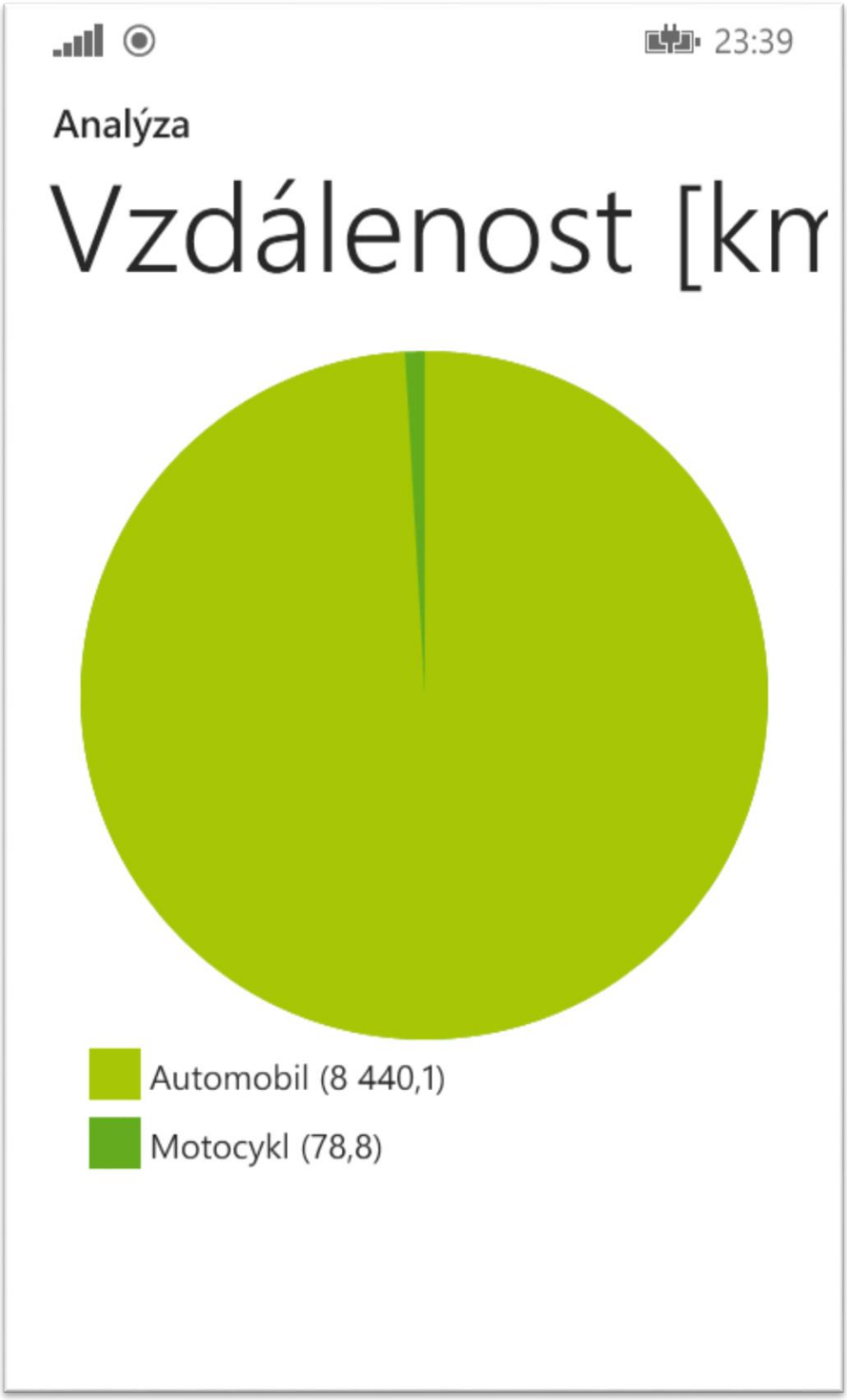

0:36
OneDrive
Přihlásit se

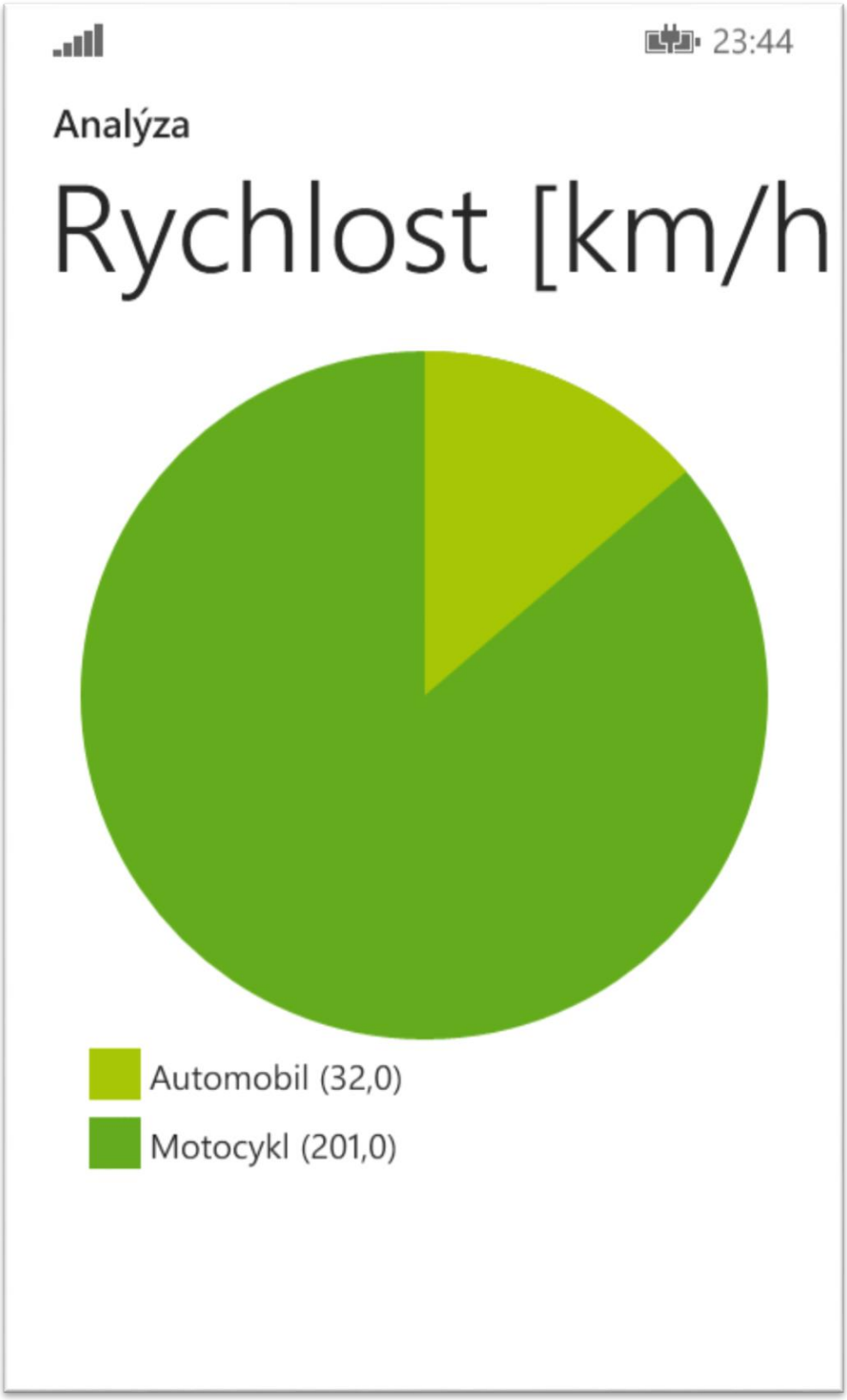

23:53
Přihlášení
Účet Microsoft Co je to?
Heslo
Přihlásit
Nezdařil se přístup k účtu?
Nemáte účet Microsoft? Zaregistrovat se teď
Ochrana osobních údajů a soubory cookie | Podmí
nky použití
©2014 Microsoft

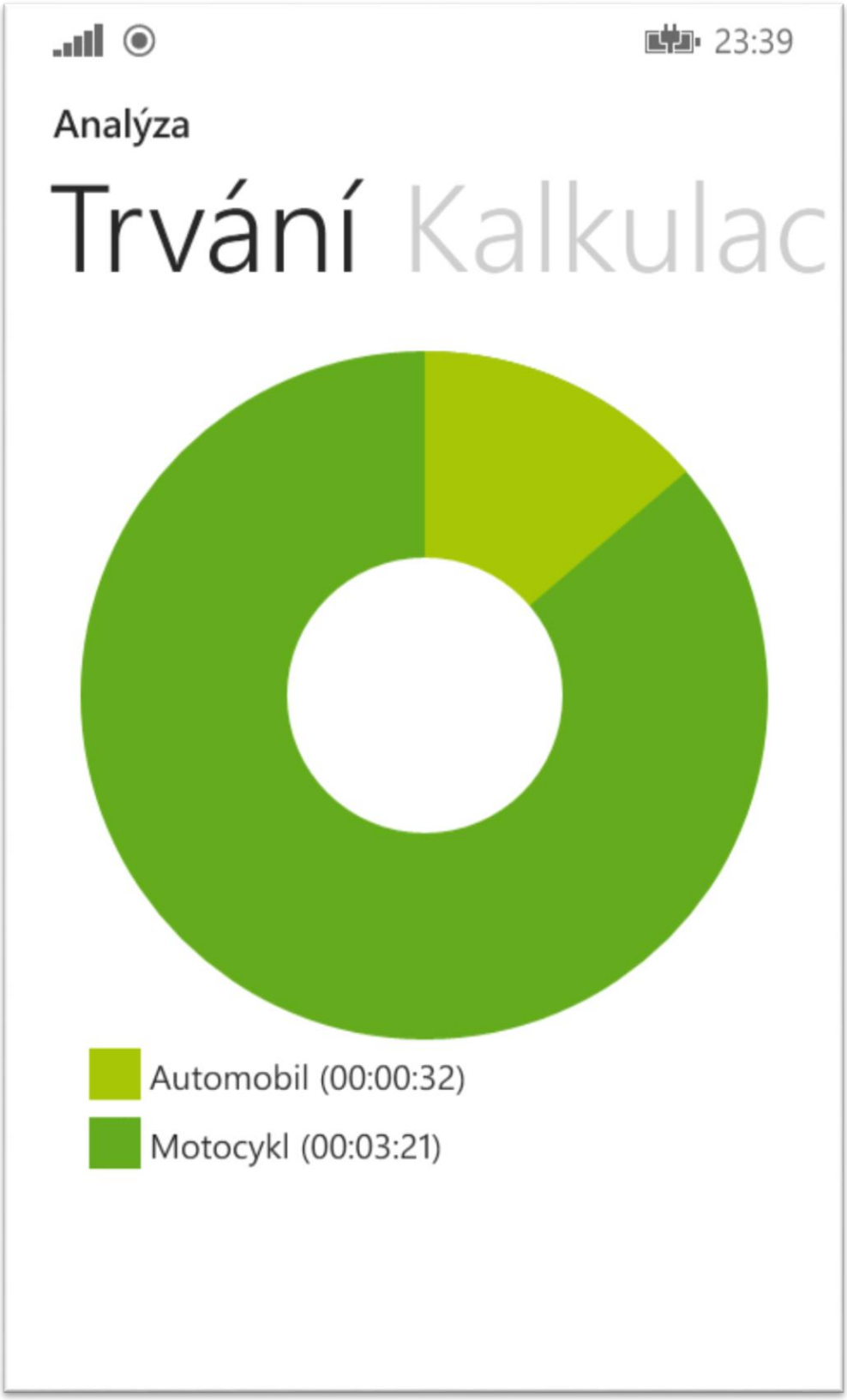

23:24
Účet Microsoft
Povolit aplikaci přístup k informacím?
Aplikace SkyDrive WP8 Upload and Download Sample vyžaduje vaše povolení k provádění těchto akcí:
Používání a úpravy na OneDrivu
Vaše automatické přihlášení
Časově neomezený přístup k vašim informacím
Tato oprávnění aplikací můžete kdykoli změnit v nastavení účtu.
Ano
Ne
Ochrana osobních údajů a soubory cookie
Podmínky použití
© 2014 Microsoft


23:24
OneDrive
Stahuji Automobil.json 1/2 100%
Nahrát
Stáhnout

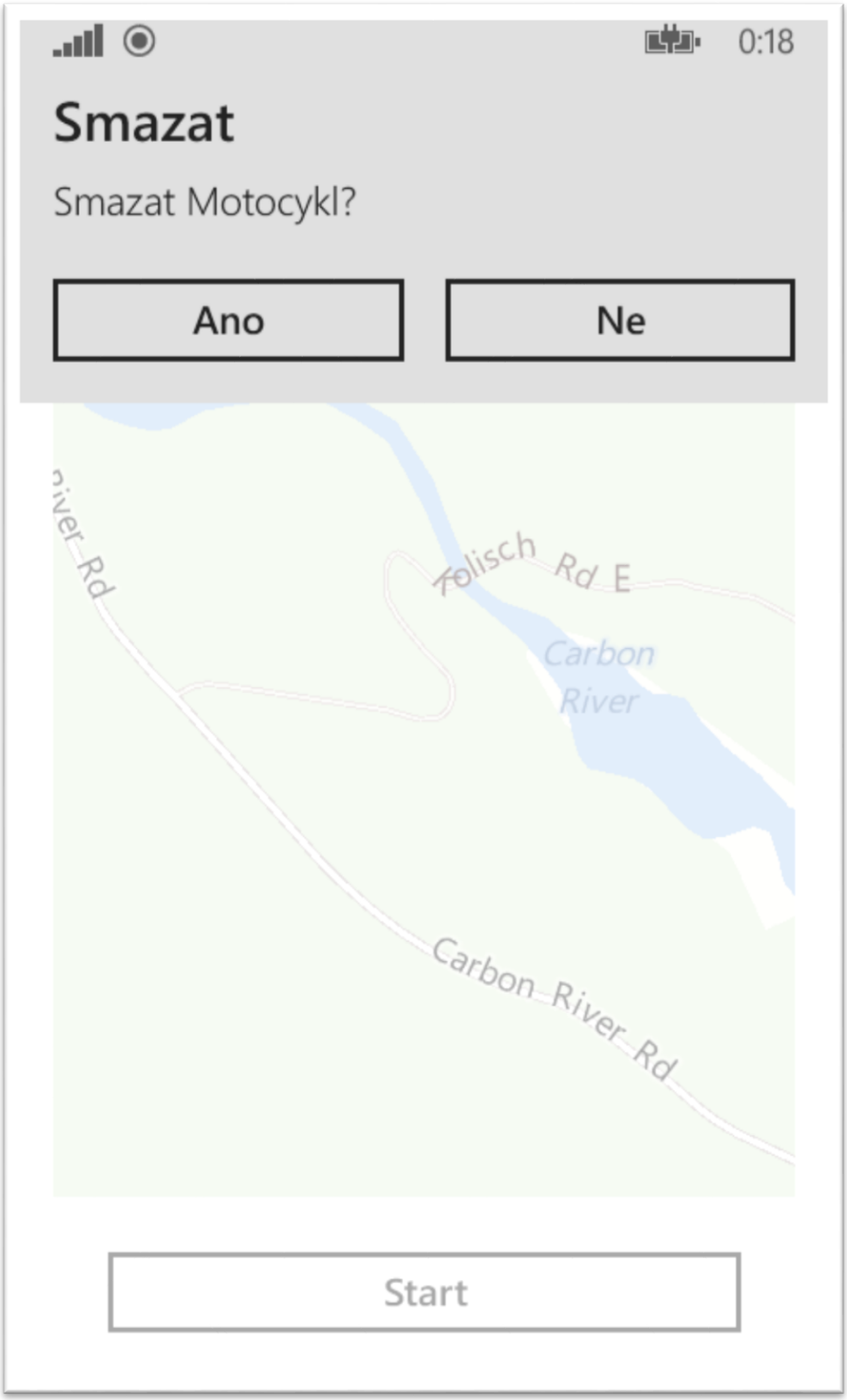

23:38
Analýza
CO2 [kg] Vzdá
Automobil (6 553,7)
Motocykl (1 830,0)


23:39
Analýza
Vzdálenost [km
Automobil (8 440,1)
Motocykl (78,8)

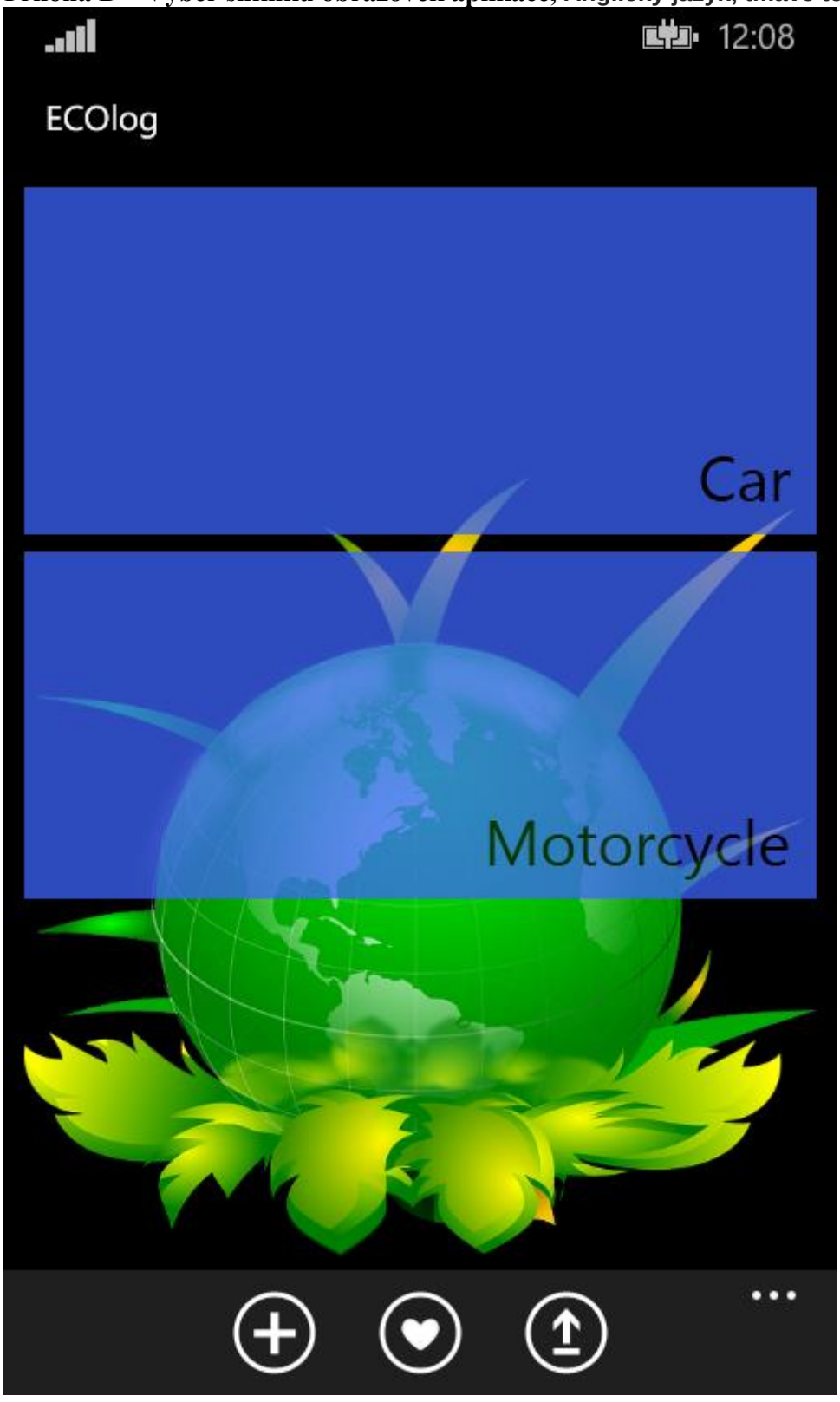

23:44
Analýza
Rychlost [km/h
Automobil (32,0)
Motocykl (201,0)

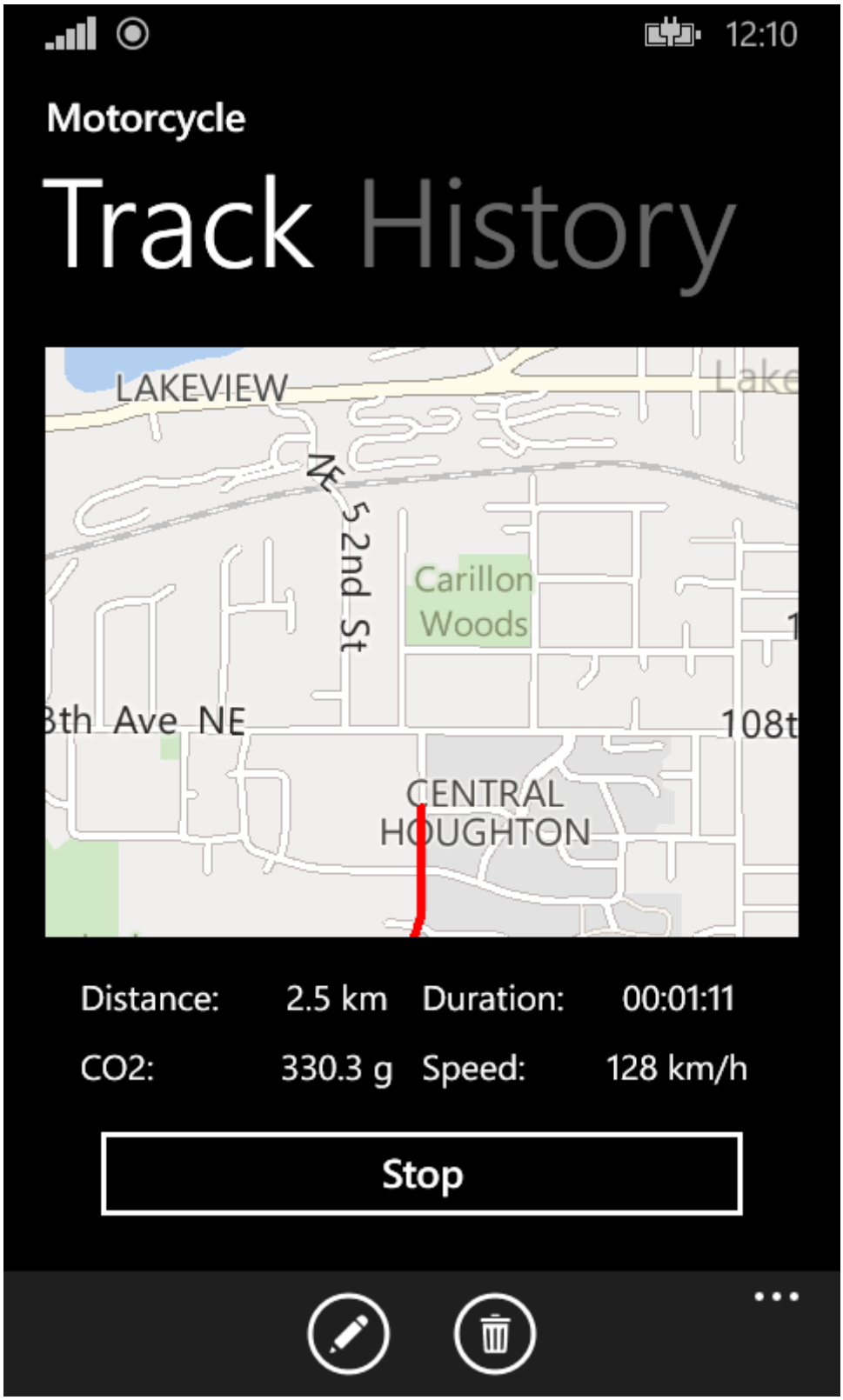

23:39
Analýza
Trvání Kalkulac
Automobil (00:00:32)
Motocykl (00:03:21)

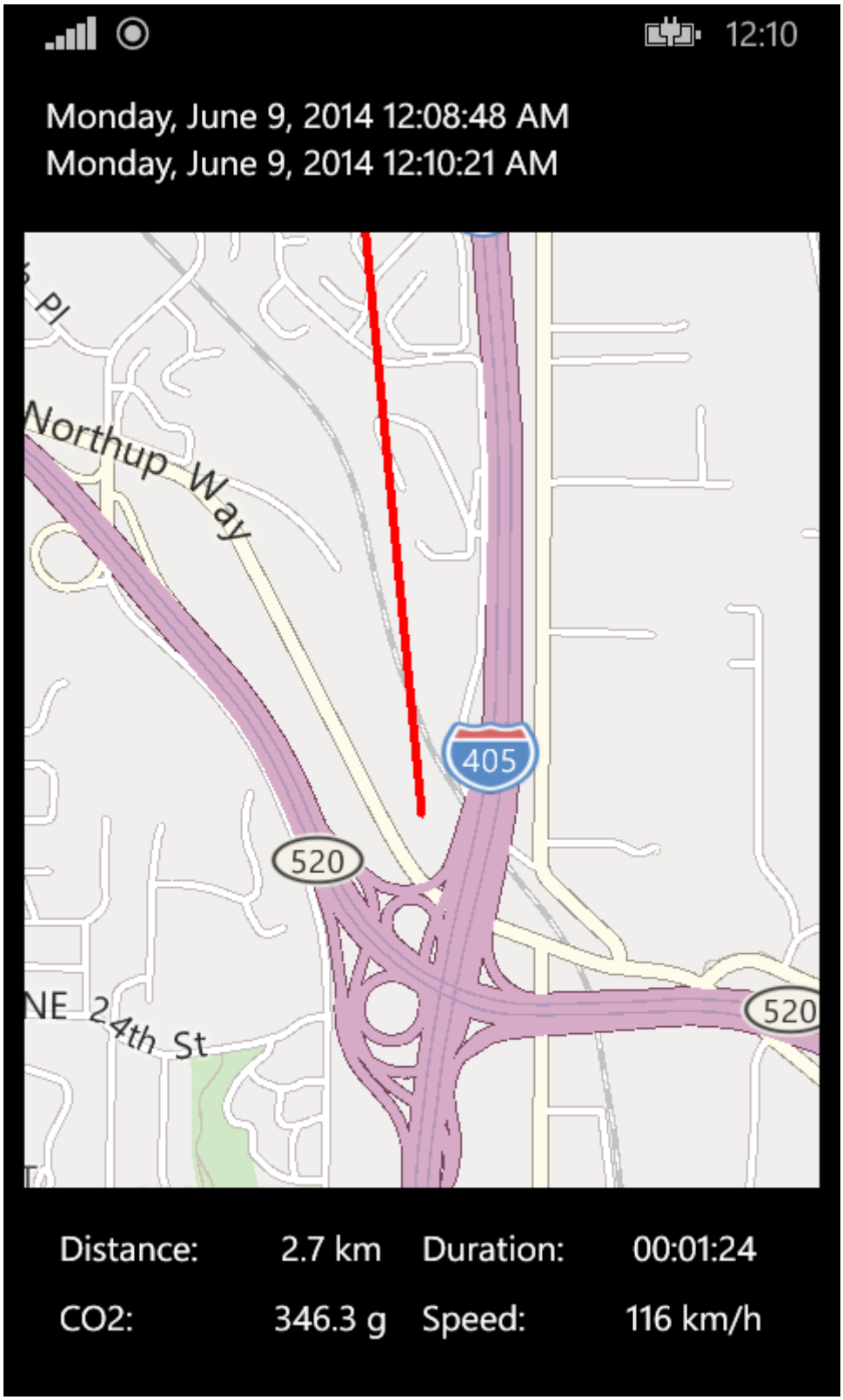

23:27
Analýza
Kalkulace CO2
9,53 metrů krychlových smrkového stavebního dřeva by navázalo Vaši produkci oxidu uhličitého.
Pro kompenzaci Vaší uhlíkové stopy je potřeba postavit 0,10 nízkoenergetických dřevěných domů namísto "běžných".

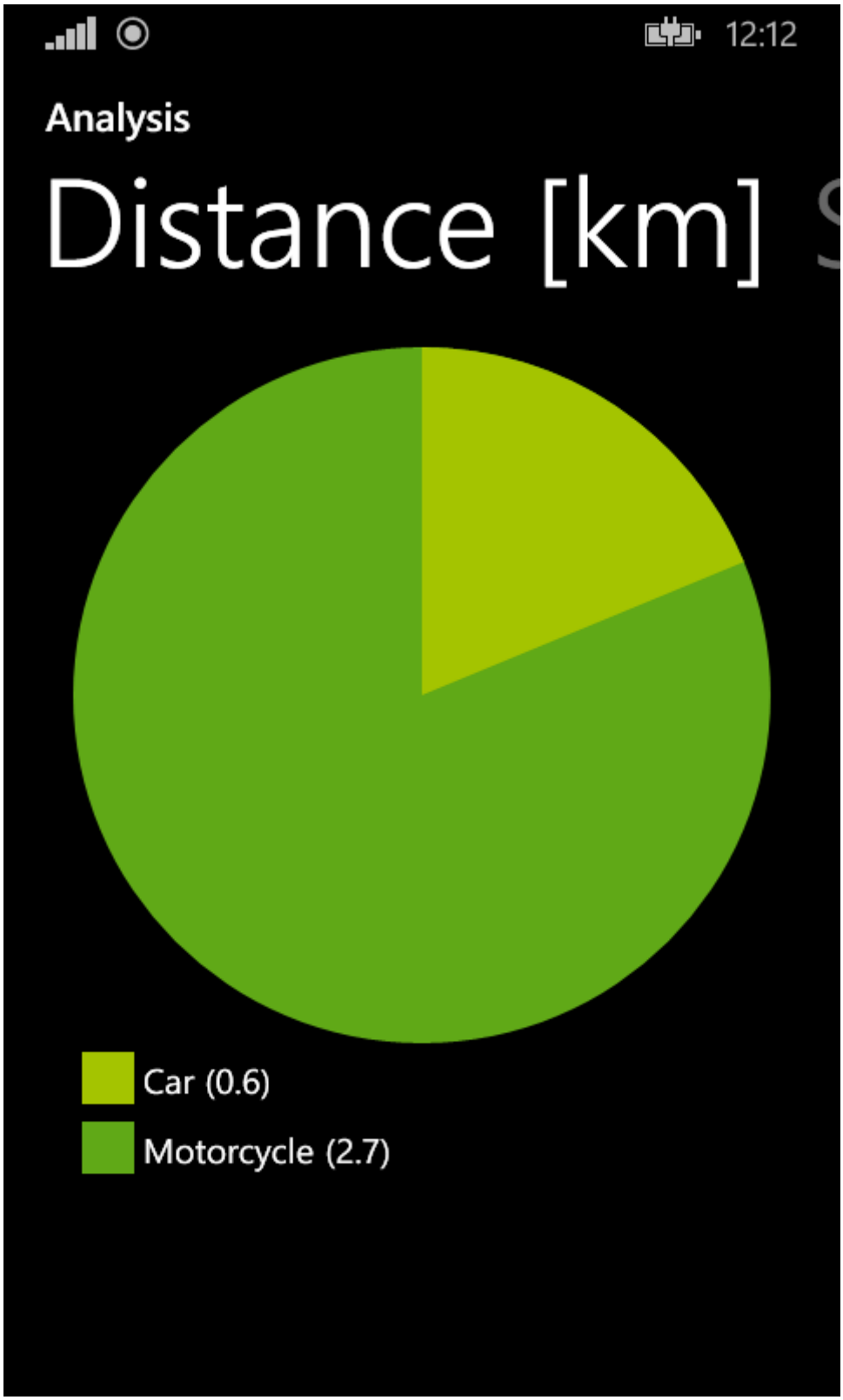

0:18
Smazat
Smazat Motocykl?
Ano
Ne
Carbon River
Carbon River Rd
Start

**Příloha B – Výběr snímků obrazovek aplikace, Anglický jazyk, tmavé téma vzhledu**




12:10
Motorcycle
Track History
LAKEVIEW
NE 52nd St
Carillon Woods
CENTRAL HOUGHTON
Distance: 2.5 km
Duration: 00:01:11
CO2: 330.3 g
Speed: 128 km/h
Stop


12:10
Monday, June 9, 2014 12:08:48 AM
Monday, June 9, 2014 12:10:21 AM
Northup Way
405
520
520
NE 24th St
Distance: 2.7 km
Duration: 00:01:24
CO2: 346.3 g
Speed: 116 km/h


12:12
Analysis
Distance [km]
Car (0.6)
Motorcycle (2.7)