VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

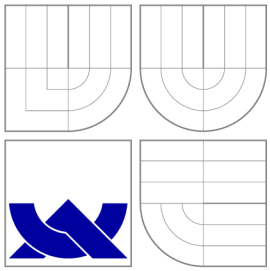
CYKLISTICKÝ/BĚŽECKÝ TRÉNINKOVÝ DENÍK VYUŽÍVAJÍCÍ GPS DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

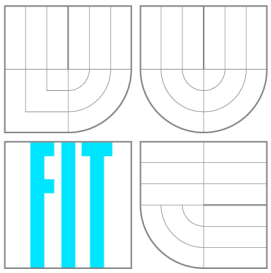
AUTOR PRÁCE
AUTHOR

Bc. MARTIN SKALICKÝ

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

CYKLISTICKÝ/BĚŽECKÝ TRÉNINKOVÝ DENÍK VYUŽÍVAJÍCÍ GPS DATA

CYCLING/RUNNING TRAINING DIARY USING GPS DATA

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MARTIN SKALICKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK KUBÍČEK

BRNO 2011

Abstrakt

Cílem praktické části práce je vytvořit aplikaci s vhodným grafickým uživatelským rozhraním, která bude umožňovat import tréninkových dat z GPS zařízení. Dále bude vytvářet grafické a statistické výstupy dosažených výsledků s možností jejich exportu do HTML a formátu podporovaného tabulkovými procesory. Teoretická část práce se týká úvodu do vedení a zaznamenávání tréninku, krátkého popisu funkce GPS systému, popisu formátu pro ukládání získaných GPS dat a návrhu aplikace.

Abstract

This master's thesis practical goal is to create an application with usefull graphical users interface, which allows to import training data from GPS device. Also it will generate graphical and statistical outputs of achived results with export option to HTML and tabular processors format. Theoretical part of this thesis presents introduction to creating of a training diary, short description of GPS system function, as next it describes GPS data storage formats and application design.

Klíčová slova

Tréninkový deník, import GPS dat, GPX, KML, zobrazování statistik, export grafů, Open Street Maps, Google Maps, C#, .NET, Windows Presentation Foundation

Keywords

Training diary, import GPS data, GPX, KML, stats presentation, graphs export, Open Street Maps, Google Maps, C#, .NET, Windows Presentation Foundation

Citace

Martin Skalický: Cyklistický/běžecký tréninkový deník využívající GPS data, diplomová práce, Brno, FIT VUT v Brně, 2011

Cyklistický/běžecký tréninkový deník využívající GPS data

Prohlášení

Prohlašuji, že jsem tento diplomový projekt vypracoval samostatně pod vedením pana Ing. Radka Kubíčka.

.....
Martin Skalický
25. května 2011

Poděkování

Zde bych chtěl poděkovat vedoucímu práce Ing. Radkovi Kubíčkovi za určování směru práce a odbornou pomoc při její tvorbě.

Obsah

Kapitola 1

Úvod

Každý sportovec, který se naplno věnuje své zálibě, touží po zlepšování svých výkonů. K tomuto účelu jsou vhodným nástrojem tréninkové deníky sloužící k zaznamenávání sportovních aktivit a jejich následné analýze, ze které se dají vyvodit vhodné postupy pro růst výkonu a kvalitní příprava k závodům. Klasické ruční zapisování do sešitů a následné dohledávání dat je zdlouhavá záležitost odrazující od vedení takového deníku. Naštěstí dnešní doba umožňuje efektivní zaznamenávání tréninkových jednotek pomocí systému GPS a přehlednou analýzu pomocí grafů a tabulek.

Tato diplomová práce má za cíl vytvoření vhodné aplikace pro vedení tréninkového deníku s intuitivním grafickým uživatelským rozhraním, která umožní uživatelům snadný záznam tréninkových jednotek a jejich následnou analýzu podle zvolených kritérií. Práce je rozdělená na teoretickou a praktickou část.

Teoretická část práce je zaměřena na úvod do vedení běžeckého deníku, kde je vysvětlena motivace pro jeho vedení, jaké základní údaje by měl obsahovat a jaké přináší výhody a nevýhody. Dále se práce věnuje popisu již existujících aplikací pro vedení tréninkového deníku s výčtem možností, které přinášejí uživatelům. Krátkému úvodu do funkce a rozdělení systému GPS se pak věnuje kapitola 3. Následuje přehled zařízení a aplikací pro získávání GPS dat, tato část obsahuje i popis formátů pro úkládání takto získaných GPS dat. Následuje analýza a návrh aplikace s diagramy užití, nasazení, tříd a E-R diagramu s návrhem databáze. Poslední 5. kapitolou práce je pak návrh exportu dat z aplikace.

Praktická část práce popisuje implementaci aplikace podle návrhu. Aplikace je implementována v jazyce C# .NET s využitím Windows Presentation Foundation pro vytvoření graficky bohatého uživatelského rozhraní popsaného v 6.1. Práce obsahuje úvod do použitých technologií a pokouší se vystihnout jejich výhody a nevýhody. V části 6.2 je popsána knihovna *Dynamic Data Display*, která umožňuje tvorbu grafů a zobrazování mapového zdroje z *Open Street Maps*. K práci s datovými kolekcemi aplikace využívá technologie LINQ, což je blíže popsáno v části 6.3. Následuje popis vlastních tříd sloužících pro chod aplikace. Poslední částí práce je pak návrh možných dalších rozšíření aplikace a jejího reálného nasazení.

Kapitola 2

Vedení běžeckého tréninkového deníku

2.1 Motivace pro sport

Běh, cyklistika nebo sport obecně dává jeho vyznavačům hodně. Na prvním místě fyzickou kondici a s ní spojené lepší zdraví, či potřebné odreagování od problémů [14].

Pravidlem jedna je tedy stavovit si jasné cíle podle toho, co je momentálně důležité. Například:

- zhubnout několik kilogramů,
- uběhnout v kuse 10 kilometrů,
- uběhnout půlmaraton a
- nakonec uběhnout maraton.

Aby se těchto cílů dosáhlo, musí se také něco obětovat. Nezáleží na tom, jak moc rádi sportujeme, každému se může stát, že z určitého důvodu není nálada. Abychom chtěli sportovat, musí nás to bavit, protože to, co nepřináší výsledky nebo upadá do stereotypu, časem omrzí.

Opravdová zkouška odhodlání běhat nastává na začátcích, kdy tělo nezvyklé na stálý pohyb a zátěž bolí, nestíhají se plnit dávky, které jsme si předsevzali nebo přijde zranění znemožňující delší dobu trénink. Nejdůležitější je v této fázi vydržet, protože nečinností se cíle splnit nedají.

Pokud ale jednoho krásného dne dojde ke zjištění, že už sport netěší tak jako na začátku, je vhodná doba pro změnu terénu, kolektivu, intenzity nebo jenom změnit hudbu, která při běhu hraje.

2.1.1 Motivace při zranění

Jedním z nejtěžších testů vůle je případné zranění, které může znemožnit trénink po delší dobu. Jenom malé množství sportovců nikdy nepotkal některý z důvodů, který je na nějaký čas donutil, aby vysadili tréninkové dávky. Období bez tréninku donutí část mysli k tomu, že si zvykne na život bez tréninku, disciplíny, zdravého stravování a všeho ostatního se sportem spojeného.

V průběhu zranění se doporučuje jiný sport, který by nás udržel v zavedeném režimu tréninků (např. při zlomené noze lze posilovat ruce). Takže při vhodně zvoleném náhradním sportu se bude člověk při návratu cítit mnohem lépe, než když by nedělal nic a litoval se.

Po zranění nelze hned začít s takovou zátěží jako v době před zraněním, kdy bylo tělo zvyklé trénovat a mělo fyzickou kondici, kterou nabralo při trénincích. Důležité je nevzdat se předsevzatých cílů, ale nesnažit se jich dosáhnout hned, protože by mohlo snadno dojít k obnovení zdravotního problému nebo dalšímu zranění.

Další věc, která může běžce po vynucené pauze limitovat, je strach z obnovení zranění. Toho se dá pozitivně využít a zátěž zvyšovat jenom s velkou opatrností, aby se tomuto zamezilo, což je při uzdravení prioritou.

Z vlastních zkušeností doporučuji neuspěchat návrat k maximální zátěži. I když se už člověk cítí v pořádku, tak i malá nezahojená trhlinka ve svalu způsobí při nevhodném tréninku nejen velkou bolest, ale i mnohem delší následnou léčbu. Může se tak stát, že léčba normálně trvající 2 týdny se protáhne na měsíce.

2.2 Běžecký deník

Pokud existuje dostatečná motivace k tréninku běhu, pak je vhodné zavést si běžecký deník. Ten si lze vést na papíře nebo použít vhodnou aplikaci, která se tak stává určitým osobním trenérem. V této části je čerpáno převážně z [1].

2.2.1 Základní údaje

Základem deníku bývá *uběhnutá vzdálenost*, *čas* a *klidový srdeční tep*, *jenž ukazuje stav fyzické kondice*. Tyto údaje jsou potřebné pro určování zvyšování objemu a intenzity tréninku. Dalším údajem v deníku může být i cíl, kterého chceme dosáhnout (viz 2.1).

2.2.2 Další údaje

Zapisovat by se měly i údaje o fyzickém stavu těla, např. kde co bolí, i kdyby to byla jenom maličkost. Tímto způsobem se dá lépe předcházet vzniku vážných zranění, která mohou začínat nenápadně. Dalším atributem v deníku by měl být i popis pocitu z vykonaného tréninku, kdy např. běžec může mít pocit, že se po dvaceti minutách běhu dostavila "krize", ale údaje vypovídají o tom, že běh byl dobře rozložený, pak se jednalo pouze o pocit a příčinou může být třeba počátek virového onemocnění nebo příznaky přetrénování (v tomto případě je dobré na krátkou dobu vysadit). Možnost zapisování převýšení při běhu je také vhodná, právě z důvodu špatného pocitu z běhu v určité fázi, kdy opticky převýšení nevnímáme, ale dochází k němu.

2.2.3 Výhody

Největší výhodou vedení běžeckého deníku je možnost data analyzovat a sledovat, jaké objemy a intenzita vedly k zlepšování fyzické kondice a ze získaných informací plánovat další postup v tréninku.

2.2.4 Nevýhody

Skrytou nevýhodou vedení deníku může být to, že se člověk stane jeho otrokem, a to tak, že si toho ani nevšimne. Potom dochází k situacím, že trénuje i v době, kdy by bylo mnohem

vhodnější odpočívat, jenom z toho důvodu, aby neměl prázdnou stránku či kolonku.

2.3 Existující aplikace

2.3.1 MyTreneek

Jedna z webových aplikací (nutná registrace) pro vedení běžeckého deníku, která poskytuje velké množství možností vytváření tréninkových aktivit a jejich správy [15]. Velkou nevýhodou však je chybějící možnost importu dat z GPS. Existuje také možnost placené verze a nákupu předtvořeného tréninkového plánu.

Tato webová aplikace poskytuje svým uživatelům zajímavé možnosti. Správu deníků pro hlavní trénink, intervaly, posilování, plavání, závody, výsledky a doplňky výživy.

Jedná se většinou o přehledné tabulky s možností její správy a filtrování podle zadaných parametrů. V neplacené verzi aplikace také lze zaznamenávat další údaje:

- Laboratorní vyšetření – formulář pro zadání výsledků laboratorního vyšetření a tabulka s přehledem vyšetření.
- Trasy – formulář pro vyplnění uběhnuté trasy a tabulka s přehledem tras.
- Náčiní – formulář pro zadání vlastního náčiní a tabulka s přehledem náčiní.
- Vlastní doplňky výživy – formulář pro zadání vlastních doplňků výživy a tabulka s jejich přehledem.
- Grafy – zobrazení grafů s kilometry s možnostmi filtrování podle vybraných parametrů.
- Statistika – zobrazení statistik podle měsíců s možnostmi filtrování podle vybraných parametrů.

V placené verzi lze pak dále spravovat vlastní tréninkové prvky, systémy tréninkových prvků nebo tréninkový plán, kde se nabízí možnost zakoupení některého z předdefinovaných tréninkových plánů.

2.3.2 RunKeeper

Další z testovaných aplikací, která má jak placenou, tak volně dostupnou verzi pro mobilní a webové rozhraní, je RunKeeper [16]. Zajímavou vlastností této aplikace je možnost propojení se sociální sítí Facebook.

Mobilní aplikace

Funguje tak, že se do mobilního telefonu podporujícího záznam GPS polohy nainstaluje aplikace, která zaznamenává pohyb. Po vykonání úkonu má uživatel možnost záznam aktivity uložit. V případě uložení aktivity se dají data odeslat na server. Aplikace umožňuje také sledovat polohu přímo na mapě, aktuální tempo a spálené kalorie.

Webová aplikace

Po registraci a přihlášení pod svým uživatelským účtem k serveru se uživateli zobrazí hlavní strana s tzv. "FitnessFeed", což je přehled posledních aktivit uživatele i jeho kolegů. Aplikace umožňuje vyhledávat trasy, závody a fitness kurzy.

Uživatel může sledovat a filtrovat podle vlastního výběru aktivity, trasy, tým kamarádů, fyzickou kondici, závody, fitness kurzy a může ručně vložit novou aktivitu.

Kapitola 3

GPS (Global Positioning System)

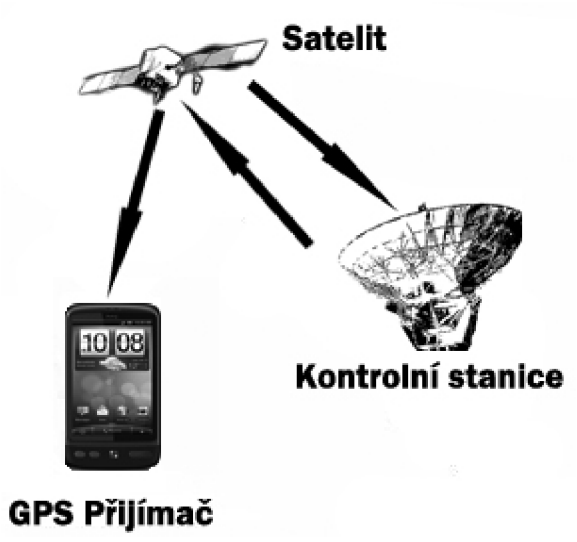
3.1 Co je GPS?

Jedná se o globální družicový polohový systém vlastněný Spojenými státy americkými. Poskytuje svým uživatelům poziční, navigační a časovací služby. Systém se skládá ze tří segmentů:

1. Kosmického, který potřebuje pro *plnou operační schopnost* 24 satelitů (současný počet je 32 satelitů [8]) vysílajících jednosměrný signál určující aktuální pozici GPS satelitu a čas.

2. Kontrolního, skládajícího se z celosvětových monitorů a kontrolních stanic, které udržují satelity v jejich správné dráze díky občasným korekčním manévrům a nastavují hodiny satelitu.

3. Uživatelského, který se skládá z GPS přijímače dostávajícího signály z GPS satelitů a používajícího přenesené informace k výpočtu pozice uživatele v třírozměrném prostoru a čase.



Obrázek 3.1: GPS systém.

Vývoj, údržbu a řízení kosmického i kontrolního segmentu obstarává Letectvo Spojených států amerických. V této části je čerpáno převážně z [7] a [12].

3.1.1 Uživatelský segment

Vzhledem k tomu, že GPS systém byl vyvinut a je udržován Leteckými silami Spojených států amerických, slouží primárně k vojenským účelům. Uživatele využívající systém GPS můžeme rozdělit do dvou skupin:

Autorizovaní uživatelé

Mají k dispozici dekódovací klíče pro vojenský `P kód` vysílaný na frekvencích *L1* (1575,42 MHz) a *L2* (1227,62 MHz). Více informací o tématu v [13]. Díky dekódovacím klíčům mohou využívat službu PPS (Precise Positioning Service) a mají zaručenou vyšší přesnost. Toto se týká vojenského sektoru USA a vybraných spojeneckých armád např. pro podporu velení a vojáků v poli, dopravu, navádění zbraňových systémů, vojenské geodézie, mapování a přesného času ($< 10^{-7}$s).

Ostatní uživatelé

využívají C/A kódu na frekvencích *L1* tzv. SPS (Standard Positioning Service). Více vysvětleno v [13]. Tyto služby využívá především civilní sektor pro dopravu, geologii, geofyziku, geodézii, geografické informační systémy, archeologii, lesnictví, zemědělství, turistiku, zábavu a přesný čas ($< 10^{-6}s$).

Zařízení vyrobená v USA nesmí být exportována, pokud nemají omezení pro výšku (do 18 km) a rychlosti (do 515 m/s) z důvodu prevence možného zneužití pro navádění balistických střel s plochou dráhou letu.

3.2 Zařízení pro záznam GPS

Navigační GPS zařízení je každé zařízení, které přijímá GPS signály k účelu zjištění aktuální polohy zařízení na planetě. GPS zažízení poskytuje údaje o zeměpisné šířce a délce. Některá vypočítávají nadmořskou výšku, ale tyto údaje nejsou natolik přesné a spojité (ztráta signálu, blokování signálu atd.). GPS zařízení se používají v armádě, letectví, námořních silách a v civilním sektoru.

GPS zařízení mohou také poskytovat další služby jako:

- Mapy čitelné uživatelem v textovém nebo grafickém formátu.
- Navigaci z aktuálního místa na určenou pozici pomocí textu nebo hlasu.
- Navigaci přímo autonomnímu zařízení jako je robotický průzkumník.
- Informace o hustotě dopravy a případné alternativní cesty.
- Informace o službách (restaurace, benzinové stanice atd.) v blízkosti aktuální polohy uživatele.

Zařízení se pak dělí na:

- Navigační (TomTom, Garmin atd.).

- GPS moduly (Nokia LD-3W atd.).
- Vestavěné (GPS zařízení např. v mobilních telefonech).

K práci bylo využíváno zařízení HTC Desire (tzv. chytrý telefon) s vestavěným GPS zařízením. Operačním systémem zařízení je Android.

3.3 Aplikace pro záznam GPS

Vzhledem k tomu, že pro práci je využíváno zařízení s operačním systémem Android, je velký výběr aplikací pro záznam GPS. Jednou z těchto aplikací je *MyTracks* [6] pro zaznamenávání trasy v podobě GPS signálu a zobrazování aktuálních statistik jako je čas, rychlost, vzdálenost a převýšení při jízdě na kole, běhu a dalších venkovních aktivitách. Po ukončení nahrávání záznamu trasy ji může uživatel aplikace sdílet pomocí Google Spreadsheets, zobrazit na Google My Maps nebo exportovat ve vybraných formátech (GPX (3.4.1), KML (3.4.2), CVS nebo TCX). Další alternativou může být např. aplikace *Big Planet Tracks*, která také umožňuje záznam GPS souřadnic trasy.

3.4 Formáty záznamu GPS

Jednou z vhodných možností záznamu GPS tras je využití formátů vycházejících ze značkovacího jazyku XML (jednoduchý a volně přístupný formát pro uchovávání dat).

3.4.1 GPX

GPS eXchange Format [4] je XML datový formát pro uložení informací o GPS souřadnicích.

Může být používán k popisu cestovních bodů, tras a cest. Jedná se o otevřený formát a může být využíván bezplatně. Obsahuje zeměpisnou šířku, délku, výšku a čas, lze jej tedy snadno použít pro výměnu dat mezi zařízeními GPS nebo mezi GPS zařízením a aplikací.

Některé počítačové aplikace umoňují uživateli např. zobrazení trasy na mapě (Google Earth), komentovat místa na mapě nebo označovat fotografie pomocí geolokace v metadatech EXIF. Ukázka dokumentu ve formátu GPX je na příkladu 1.

Příklad 1 Dokument ve formátu GPX.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<?xml-stylesheet type="text/xsl" href="details.xsl"?>
<gpx version="1.0" creator="My Tracks for the G1 running Android"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.topografix.com/GPX/1/0"
  xmlns:topografix="http://www.topografix.com/GPX/Private/TopoGrafix/0/1"
  xsi:schemaLocation="http://www.topografix.com/GPX/1/0 .....">
  <trk>
    <name>
      <![CDATA[Track 1]]></name>
      <desc>
        <![CDATA[]]></desc>
        <number>1</number>
        <topografix:color>c0c0c0</topografix:color>
        <trkseg>
          <trkpt lat="49.750851" lon="16.457058">
            <ele>489.0</ele>
            <time>2010-10-04T14:25:48Z</time>
          </trkpt>
          <trkpt lat="49.750884" lon="16.457042">
            <ele>490.0</ele>
            <time>2010-10-04T14:26:16Z</time>
          </trkpt>
        </trkseg>
    </trk>
</gpx>
```

3.4.2 KML

Keyhole Markup Language [5] je XML schéma pro vyjádření geografického popisu a vizualizace v rámci internetových dvourozměrných a třírozměrných světových map.

KML bylo vyvinuto pro užívání v aplikaci Google Earth (původní název byl Keyhole Earth Viewer). Schéma bylo vytvořeno společností Keyhole, Inc. najatou Googlem v roce 2004. Název "Keyhole"je pocta vojenským KH průzkumným satelitům vypuštěným v roce 1976.

KML je mezinárodní standard z Open Geospatial Consortium. Google Earth byl prvním programem schopným zobrazovat a graficky upravovat KML soubory.

Soubor ve formátu KML určuje sadu vlastností (značky míst, obrázky, polygony, 3D modely, textové popisy atd.) pro zobrazení v aplikaci Google Earth, mapách, mobilních zařízeních a každé další aplikaci používající KML formát. Každé místo má vždy svoji zeměpisnou šířku a délku. Další data jako náklon, směr a nadmořská výška mohou dále specifikovat zobrazení a definují pohled pomocí kamery. Ukázka dokumentu ve formátu KML je na příkladu 2.

Příklad 2 Dokument ve formátu KML.

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://earth.google.com/kml/2.0"
     xmlns:atom="http://www.w3.org/2005/Atom">
  <Document>
    <atom:author>
      <atom:name>My Tracks running on Android</atom:name>
    </atom:author>
    <name><![CDATA[Track 1]]></name>
      <description>
        <![CDATA[]]></description>
        <Style id="track">
          <LineStyle>
            <color>7f0000ff</color>
            <width>4</width>
          </LineStyle>
        </Style>
        <Style id="sh_green-circle">
          <IconStyle>
            <scale>1.3</scale>
            <Icon>
              <href>http://maps.google.com/mapfiles/kml/paddle/grn-circle.png</href>
            </Icon>
            <hotSpot x="32" y="1" xunits="pixels" yunits="pixels"/>
          </IconStyle>
        </Style>
        <Placemark>
          <name><![CDATA[(Start)]]></name>
            <description>
              <![CDATA[]]></description>
              <styleUrl>#sh_green-circle</styleUrl>
              <Point>
                <coordinates>16.457058,49.750851</coordinates>
              </Point>
            </Placemark>
            <Placemark>
              <name><![CDATA[Track 1]]></name>
                <description>
                  <![CDATA[]]></description>
                  <styleUrl>#track</styleUrl>
                  <MultiGeometry>
                    <LineString>
                      <coordinates>
                        16.457058,49.750851,489.0
                        16.457058,49.750851,489.0
                        16.457042,49.750884,490.0
                      </coordinates>
                    </LineString>
                  </MultiGeometry>
                </Placemark>
          </Document>
        </kml>
```

Kapitola 4

Analýza a návrh aplikace

Důležitou součástí tvorby aplikací je i vhodná analýza požadavků a návrh systému, kterým se programátor řídí. Pokud se tato část podcení, je velmi pravděpodobné, že se čas vývoje systému prodlouží a prodraží.

V této kapitole bude použto UML diagramů:

- případů užití,
- nasazení,
- tříd a
- E-R diagramu pro návrh databáze.

4.1 Neformální specifikace

Aplikace bude mít pro uživatele *více úrovní práv*.

Základní požadavky na *uživatelskou část* aplikace jsou popsány v sekcích 2.2.1 a 2.2.2. Dalé by pak měla aplikace umožňovat import GPS dat ze zařízení ve zvoleném formátu *GPX* nebo *KML*. Pro lepší orientaci v dosažených tréninkových výsledcích bude existovat možnost grafického zobrazení statistik a map podle uživatelem zadaných kritérií, s možností jejich exportu do HTML nebo formátu podporujícího zpracování v tabulkovém procesoru.

Vyšší úroveň práv bude obsahovat stejné možnosti jako uživatelská a dále umožňovat tvorbu a správu tréninků, např. volbu trasy a náročnosti tréninku.

Nejvyšší úroveň práv bude obsahovat stejné možnosti jako všechny nižší úrovně a umožňovat správu a tvorbu uživatelských účtů i možnost přidávání typů aktivit.

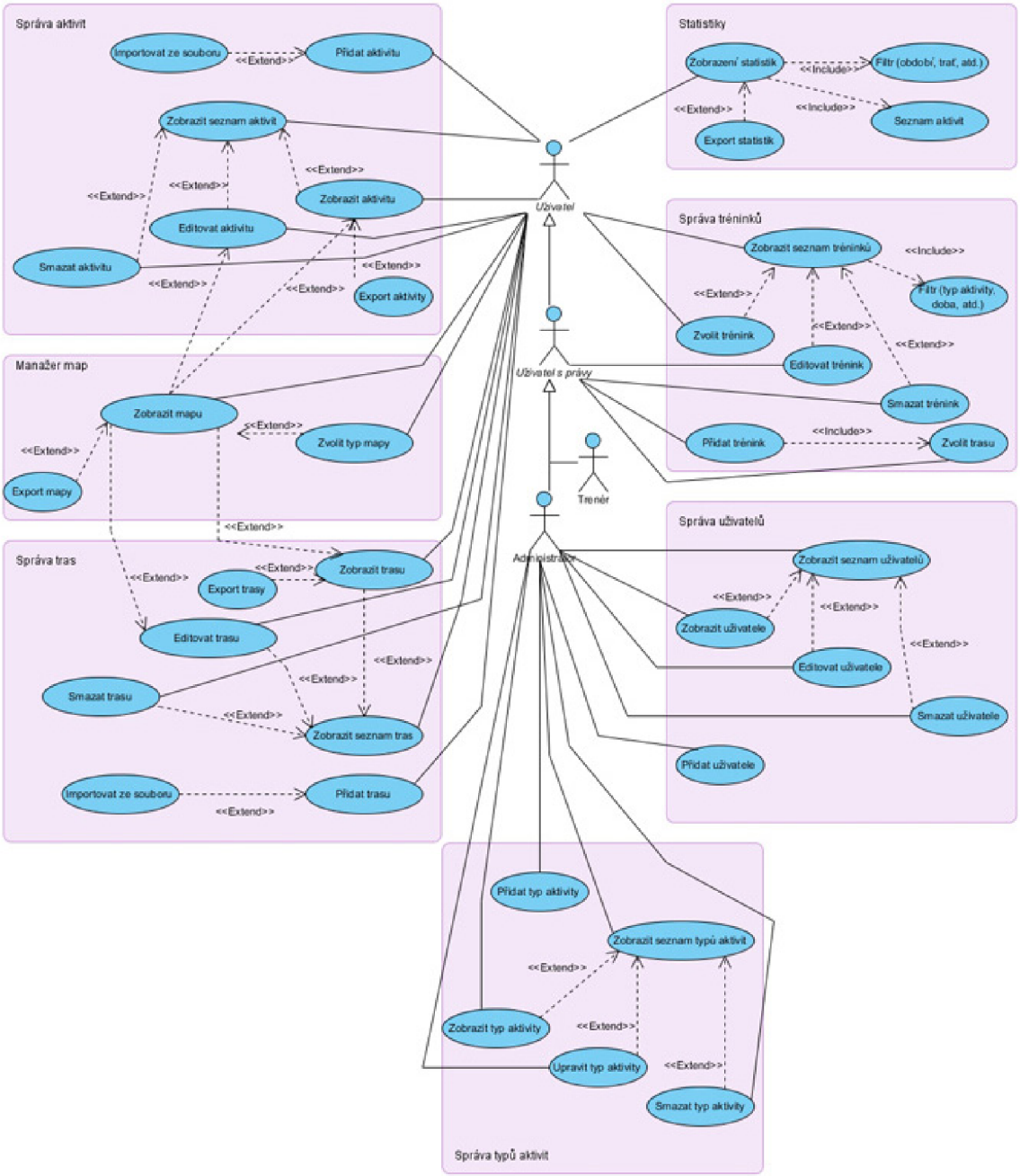
4.2 Analýza požadavků

Z neformální specifikace plynou následující požadavky na systém (viz obrázek 4.1):

- Abstraktní aktéři:
  - Uživatel – reprezentuje nejobecnějšího aktéra systému, který bude mít základní možnosti vedení tréninkového deníku. Mezi tyto možnosti patří správa aktivit, tras, výběr tréninku, zobrazení a export statistik podle zvolených filtrů a grafické zobrazení aktivit či tras na mapě.

  - Uživatel s právy – reprezentuje aktéra, který má možnost správy tréninků. Konkrétním potomkem je aktér Trenér.

- Speciální aktéři:
  - Administrátor – je aktérem s nejvíce právy v systému (komunikuje se všemi případy užití). Má možnost spravovat uživatele systému a spravovat typy aktivit.



Obrázek 4.1: Případy užití.

4.3 Návrh architektury

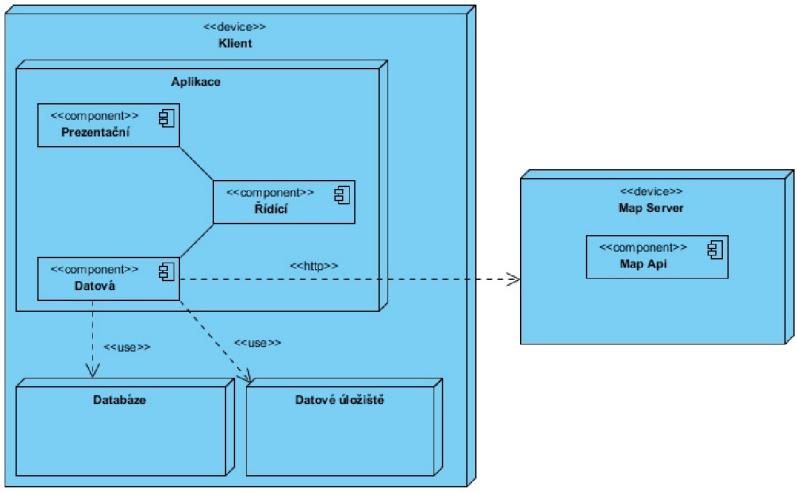
Z analýzy plyne, že existuje více možností návrhu architektury systému. Jednou z možností je využívat databázi na lokálním počítači, další je využívat databázi na vzdáleném počítači a poslední je propojení obou architektur se synchronizací databází. Všechny tyto architektury pak počítají s možností získávání map z webového serveru.

4.3.1 Architektura s databází na lokálním počítači

Aplikace na lokálním počítači bude mít tři hlavní komponenty:

- Prezentační, která bude poskytovat grafické rozhraní.
- Řídící, která bude řídit chod aplikace (reakce na akce uživatele v grafickém rozhraní).
- Datová, která bude získávat data z databáze na lokálním počítači, zpracovávat soubory pro import dat z lokáních disků a spolupracovat pomocí protokolu HTTP s mapovým serverem. Na lokálním počítači bude také umístěna databáze pro uchování získaných dat a soubory s daty pro import. Na vzdáleném počítači bude API s mapami, viz obrázek 4.2.

Výhodou architektury je rychlost komunikace s lokální databází.
Nevýhodou této architektury je globálnější rozšíření systému s možností spolupráce uživatelů a možnost ztráty spojení se vzdáleným počítačem obsahujícím API s mapami.



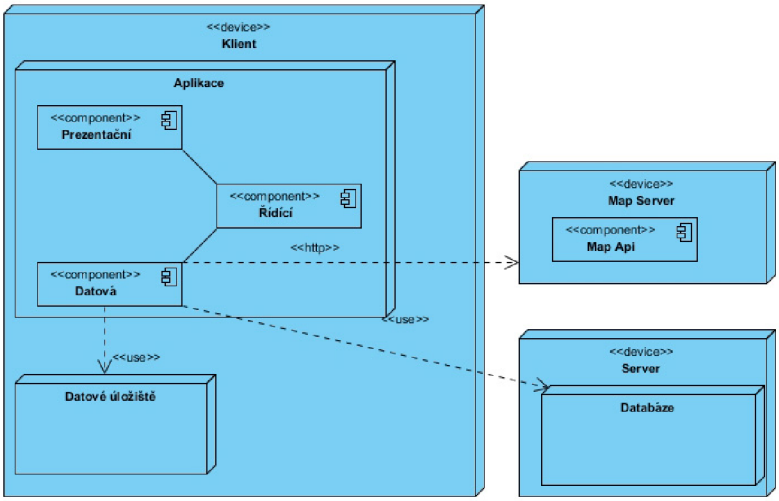
Obrázek 4.2: Architektura s databází na lokálním počítači.

4.3.2 Architektura s databází na vzdáleném počítači

Aplikace na lokálním počítači bude mít tři hlavní komponenty: prezentační, řídící a datovou. Ta bude získávat data z databáze na vzdáleném počítači, zpracovávat soubory pro import dat z lokáních disků a spolupracovat pomocí protokolu HTTP s webovým serverem. Na vzdáleném počítači bude umístěna databáze pro uchování získaných dat. Soubory s daty pro import pak budou uloženy na lokálních discích. Na vzdáleném počítači bude API s mapami, viz obrázek 4.3.

Výhodou architektury je možnost rozšíření systému mezi více uživatelů a jejich vzájemná spolupráce.

Nevýhodou této architektury je možnost ztráty spojení se vzdáleným počítačem obsahujícím API s mapami nebo se vzdáleným počítačem s databází.



Obrázek 4.3: Architektura s databází na vzdáleném počítači.

4.3.3 Architektura se synchronizací databází

Aplikace na lokálním počítači bude mít také tři hlavní komponenty: prezentační, řídící a datovou. Ta bude získávat data z databáze na vzdáleném počítači nebo lokálním počítači a synchronizovat záznamy. Dále pak zpracovávat soubory pro import dat z lokáních disků a spolupracovat pomocí protokolu HTTP s webovým serverem. Na vzdáleném nebo lokálním počítači bude umístěna databáze pro uchování získaných dat. Na lokálních discích budou uchovány soubory s daty pro import. Na vzdáleném počítači bude API s mapami, viz obrázek 4.4.

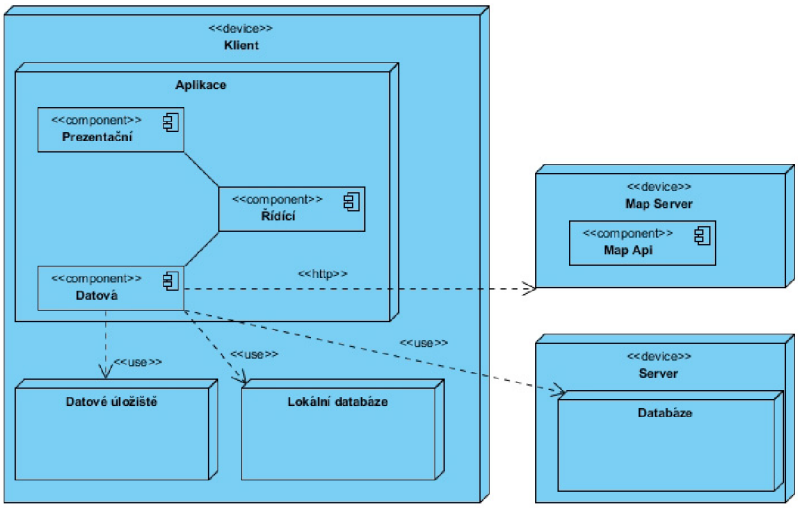
Výhodou architektury je možnost rozšíření systému mezi více uživatelů a jejich vzájemná spolupráce. I při ztrátě spojení se vzdáleným serverem je možné pracovat s databází.

Nevýhodou této architektury je možnost ztráty spojení se vzdáleným počítačem obsahujícím API s mapami.

4.4 Návrh databáze

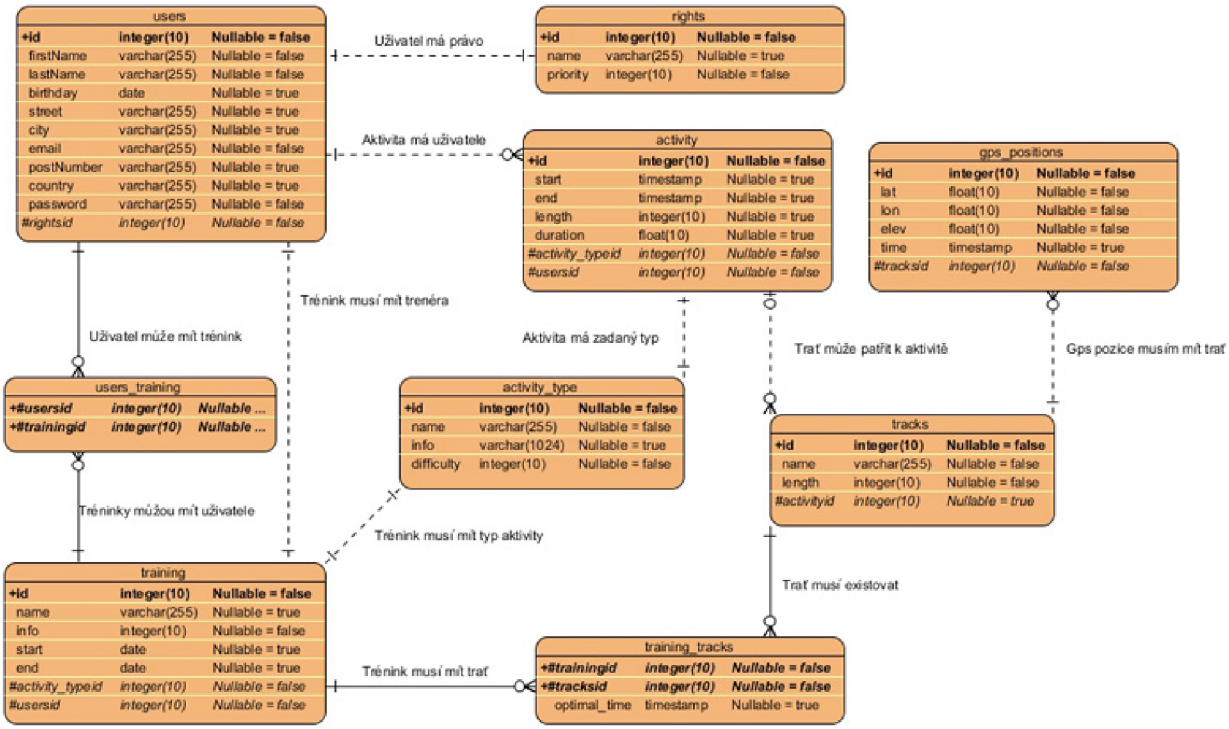
Návrh databáze je graficky prezentován pomocí entitně relačního diagramu, viz obrázek 4.5 (dále jen ER diagramu). Entity a jejich vztahy budou stručně popsány v této kapitole.

Základní entitou je *users*, která obsahuje údaje o uživatelích a úroveň jejich práv (entita *rights*). Uživatelé mohou vlastnit aktivity (entita *activity*), které jsou popsány atributy začátek, konec, délka tratě, trvání a jsou propojeny s entitou *activity_type* (popis typu aktivity) a s uživatelem, kterému daná aktivita náleží. Záznam v entitě *activity* nemůže existovat bez platného uživatele. Aktivity mohou mít zadanou trasu (entita *tracks*), ta obsahuje atributy jméno, délka v km a odkaz na aktivitu, ke které trasa náleží. Záznam v entitě trasa může existovat i bez aktivity. Entita *tracks* je propojena s entitou *gps_positions* uchovávající data pozic z GPS pomocí atributů latence, délky, nadmořské výšky, času a trasy,



Obrázek 4.4: Architektura se synchronizací databází.

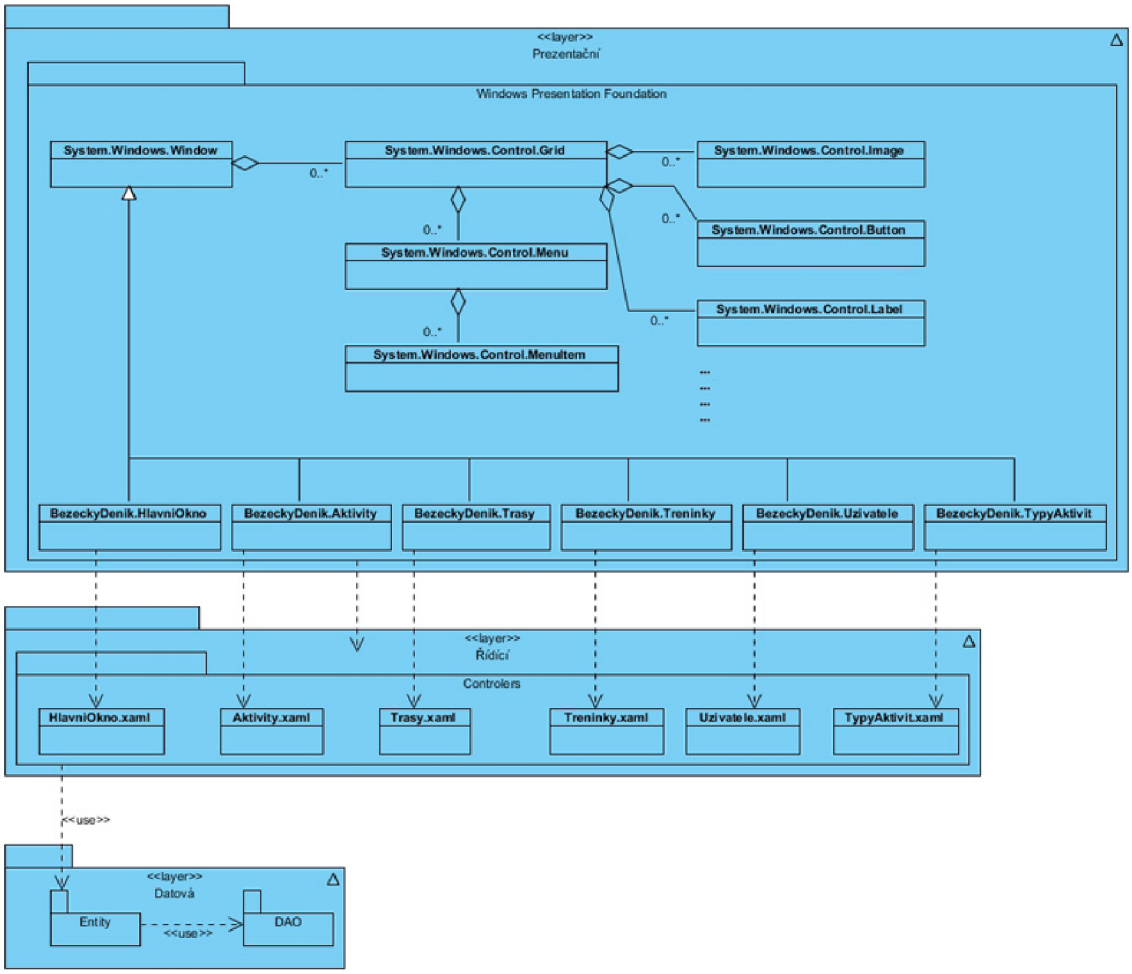
v níž se pozice nachází. Záznam v entitě *gps_positions* nemůže existovat bez platné trasy, ke které by byl přiřazen. Poslední tři entity popisují trénink (entita *training*), jeho vztah k trasám (entita *training_tracks*) a uživateli (entita *users_training*). Trénink má atributy jméno, popis tréninku, datum začátku, datum konce a je propojený s typem aktivity a uživatelem, který ho vytvořil. Záznam v entitě trénink nemůže existovat bez platného uživatele, trasy a typu aktivity specifikující, o jakou trasu a náročnost se jedná.



Obrázek 4.5: ER diagram.

4.5 Diagram návrhových tříd

Rozdělení aplikace je prezentováno na obrázku 4.6. Aplikace bude mít tři vrstvy: prezentační, řídící a datovou.



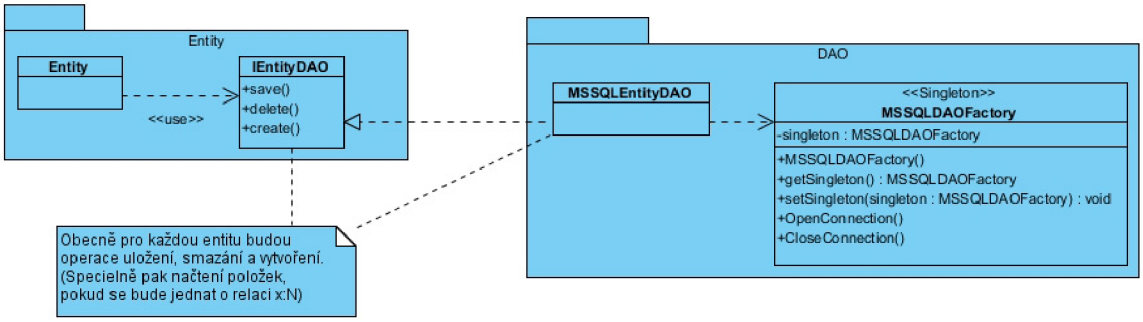
Obrázek 4.6: Rozdělení aplikace.

Prezentační vrstva aplikace se bude starat o grafické uživatelské rozhraní (dále jen *GUI*) všech vytvořených oken aplikace. GUI bude definováno pomocí komponent *Windows Presentation Foundation* využívající značkovací jazyk XAML (bude popsáno spolu s WPF v dalších kapitolách). Výčet komponent v diagramu není kompletní z důvodu jejich vysokého počtu.

Řídící vrstva bude obstarávat správu událostí komponent GUI. Každé okno aplikace bude mít svoji třídu.

Datová vrstva bude mít třídy vytvořené analogicky podle entit z návrhu databáze 4.5. Třída *Tracks* pak bude navíc obsahovat metodu pro import dat ze souboru a každá třída bude pomocí rozhraní komunikovat s databází a provádět v ní požadované změny, viz obrázek 4.7.

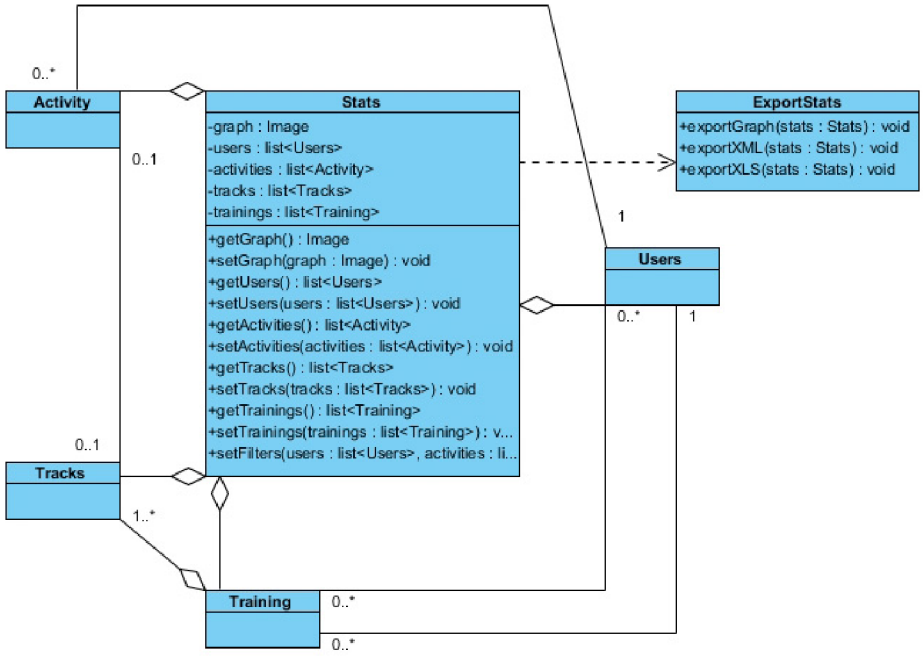
Výjimkami jsou třídy pro export starající se o převod dat do XML, které lze pomocí XSLT transformovat do dalších formátů a formátu tabulkového procesoru (konkrétně



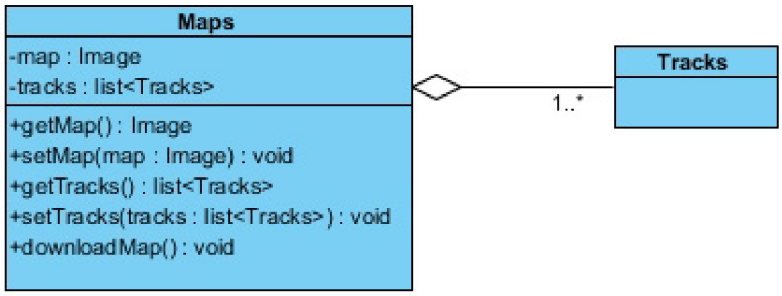
Obrázek 4.7: Komunikace entity s databází.

pak XLS). Další výjimky tvoří třídy:

- Stats – bude se skládat z seznamů objektů tříd *Users*, *Activity*, *Tracks* a *Training*. Seznamy se budou plnit objekty podle kritérií zadaných metodou *setFilters*, viz obrázek 4.8.
- Maps – bude obsahovat objekt typu obrázek, který se vykreslí v aplikaci, a seznam objektů typu *Tracks*, který bude obsahovat jednu nebo více tras k vykreslení do obrázku. Dále bude komunikovat pomocí protokolu HTTP s rozhraním mapového serveru, ze kterého bude stahovat potřebná data pro tvorbu mapy, viz obrázek 4.9.



Obrázek 4.8: Třída pro statistiky.



Obrázek 4.9: Třída pro mapy.

Kapitola 5

Návrh vhodného formátu pro export

5.1 Kritéria pro filtrování statistik

Vhodnými kritérii, ze kterých se budou dát získat relevantní data, jsou *vzdálenost*, *čas*, *období*, *náročnost aktivity* a *zvolený tréninkový plán*.

Podle vybraných kritérií pak může uživatel zjišťovat mimo jiné:

- Podle zvolené vzdálenosti nejrychlejší, průměrný a nejhorší čas. V kombinaci s kritériem období lze vytvořit např. křivka časů v různých obdobích pro sledování růstu (poklesu) výkonu v zadaném intervalu.
- Podle času lze určit nejkratší, nejdelší a průměrnou uraženou vzdálenost.
- Podle náročnosti aktivity v kombinaci s obdobím lze vytvořit křivku růstu nebo poklesu náročnosti v zadaném intervalu.

Pomocí uvedených údajů lze vytvořit mnoho dalších kombinací mimo těch uvedených výše, ty lze považovat za nejčastěji používané.

5.2 Export výsledných statistik

Export výsledků filtrování bude proveden do *XML* a do *XLS* z důvodu snadné implementace pomocí jazyka C#, který poskytuje vhodné knihovny pro práci s těmito soubory. XML se dá poté pomocí XSLT (viz 5.2.1) dále transformovat do různých dalších formátů.

5.2.1 XSL transformace 1.0

XSL je jazyk založený na XML sloužící k převodu XML dokumentů na dokumenty jiného typu nebo stejného typu, je-li potřeba odstranit nekompatibilitu např. mezi verzemi návrhu XML dokumentů.

XSLT [9] nabízí tři odlišné modely programování: model založený na vzorech, procedurální model a deklarativní model.

Kapitola 6

Popis implementace aplikace

V této kapitole je popsána praktická část diplomové práce, která je tvořena ve vývojovém prostředí Microsoft Visual Studio 2010 v jazyce C# s využitím technologie Windows Presentation Foundation, knihovny pro mapy a grafy Dynamic Data Display, technologie LINQ pro dotazy nad datovými strukturami a databází MSSQL.

6.1 Windows Presentation Foundation

Pro vytvoření vhodného GUI je využívána technologie Windows Presentation Foundation (dále jen WPF), která dovoluje využití bohaté škály grafických prvků a slouží k oddělení grafického popisu aplikace od logiky aplikace.

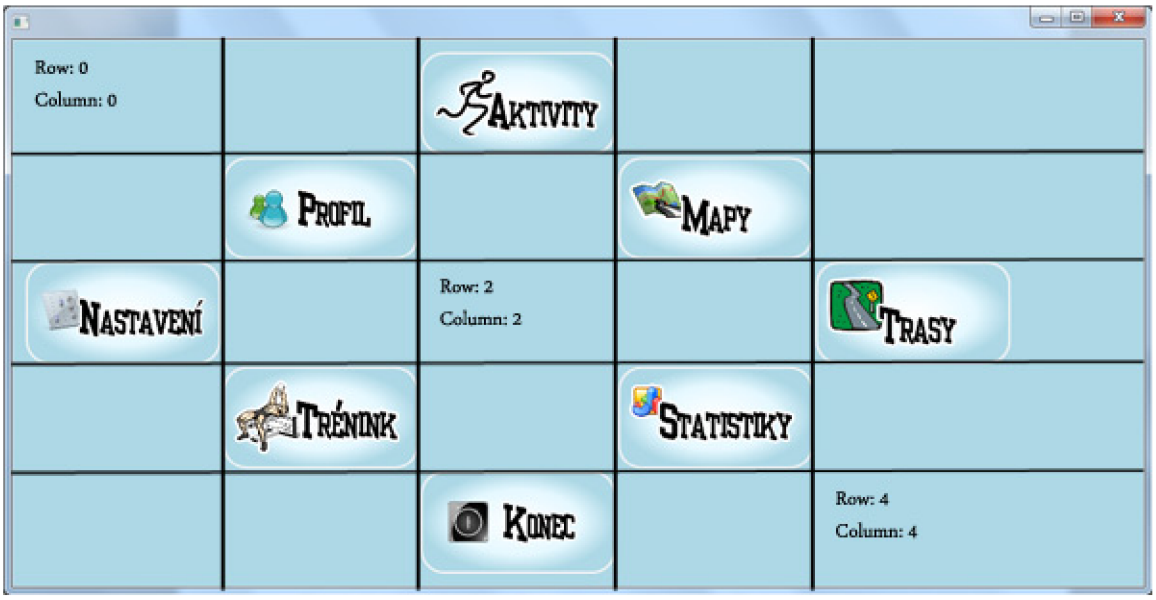
Jádro WPF je nezávislé na rozlišení a založeno na vektorovém vykreslovacím modulu pro využití moderního grafického hardware. WPF je tvořeno rozšiřitelným aplikačním značkovacím jazykem XAML (eXtensible Application Markup Language), ovládacími prvky, datovými vazbami, grafickým rozložením, 2D a 3D grafikou, animacemi, styly, šablonami, dokumenty, médii, textem a typografií. WPF je součástí Microsoft .NET Framework, takže lze kombinovat i s dalšími prvky této knihovny.

6.1.1 Využití prvků WPF v aplikaci

Tvorba aplikace pomocí WPF je jednoduchá, dobře zdokumentovaná a přístupná, takže budou stručně popsány jen klíčové prvky.

Pro okno aplikace je využita třída *Window*, jejíž obsah tvoří objekt, který je potomkem třídy *Page*. V jednom případě je využito pro ovládání obsahu aplikace třídy *TabControl* pro využití záložek. Pro grafické rozložení prvků v aplikaci je používáno tříd:

- *Border* s okraji a efekty jejich překreslení.
- *Grid* pro rozložení dalších prvků do sloupců a řádků, jak je ukázáno na obrázcích 6.1 a 6.2.



Obrázek 6.1: Okno aplikace.

```
<Grid Name="MenuGrid"
        DockPanel.Dock="Left"
        HorizontalAlignment="Left"
        RadioButton.Click="selectedMenuChanded"
        Margin="10" VerticalAlignment="Top" Visibility="Visible">
    <Grid.ColumnDefinitions>
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
        <RowDefinition />
    </Grid.RowDefinitions>
```

Obrázek 6.2: Rozložení pomocí XAML.

V aplikaci je využito stylů pro grafický vzhled některých prvků. Styly obecně umožňují vývojářům a designérům vytvořit jednotný vzhled výsledné aplikace. WPF poskytuje silný model pro stylování prvků.

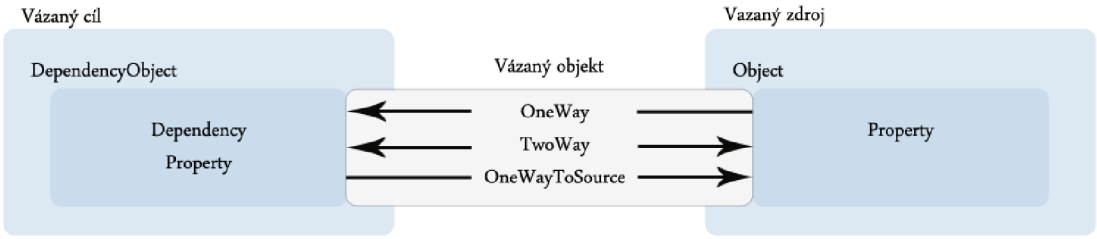
Animace jsou v aplikaci také využity pro zpřehlednění změn obsahu oken, aby uživatel viděl, že se ”něco děje”.

6.1.2 Data Binding

Data binding (vázání dat) je proces, který vytváří spojení mezi uživatelským rozhraním aplikace a aplikační logikou. Pokud jsou vazby správně nastaveny a data poskytují správné oznámení, pak při změně hodnot v datech se automaticky mění i prvky, které jsou na tyto data vázány. *Data binding* také může fungovat tak, že při změně hodnoty prvku se provedou změny ve vázaném datovém zdroji. Datový tok mezi vázaným zdrojem a cílem je pro ilustraci ukázán na obrázku 6.3, kde:

- *OneWay binding* (jednosměrná vazba) způsobuje automatickou aktualizaci cílového objektu podle zdrojového, ale na samotný zdrojový objekt vliv nemá.
- *TwoWay binding* (obousměrná vazba) způsobuje automatické změny jak v cílovém, tak zdrojovém objektu.
- *OneWayToSource binding* (jednosměrná vazba ke zdroji) způsobuje automatickou aktualizaci zdrojového objektu podle cílového objektu.

Data binding využívá aplikace pro zobrazení dat z databáze a při kontrolování správnosti zadaných dat v prvcích (například při zadávání hodiny je možné zadat pouze číslo od 0–23).



Obrázek 6.3: Datový tok mezi zdrojem a cílem.

6.2 Dynamic Data Display

Dynamic Data Display (dále jen DDD) je knihovna WPF prvků pro dynamickou vizualizaci dat v aplikacích a její zdrojový kód je volně ke stažení[1]. Je vybavena efektivními vazebními mechanismy a real-time interaktivitou schopnou mapovat miliony datových bodů. Současná verze (0.3) dovoluje flexibilní vykreslování čárových a bodových grafů.

Použití prvků DDD je stejné jako používání ostatních prvků WPF. Jakákoli data mohou být použita jako zdroj pro body grafu. Pro ovládání grafu se používá myši nebo klávesnice. Jednou z možností grafu jsou mapové podklady[2], kde jsou osami grafu zeměpisná délka a šířka.

Pro použití knihovny DDD v projektech je potřeba dodržet následující kroky [2] :

- Je nutné mít nainstalovaný Microsoft .NET Framework alespoň verze 3.5.
- Stáhnout poslední verzi DDD a rozbalit soubor DynamicDataDisplay.dll do vybraného adresáře.
- Přidat do projektu referenci na tento soubor.

[1] K 11. 5. 2011 z adresy http://dynamicdatadisplay.codeplex.com/.
[2] Open Street Maps (http://www.openstreetmap.org/).

- Pro kombinaci s jazykem C# je potřeba přidat jmenný prostor pomocí *using Microsoft.Research.DynamicDataDisplay;*
- Pro kombinaci s XAML je potřeba přidat XML jmenný prostor do hlavičky XAML souboru pomocí *xmlns:d3="http://research.microsoft.com/DynamicDataDisplay/1.0"*. Nyní jsou prvky DDD přístupné pomocí prefixu *d3:*.

Součástí archivu s knihovnou jsou i ukázky aplikací.

6.2.1 Využití prvků DDD v aplikaci

Prvky DDD jsou v aplikaci využity pro zobrazení mapových podkladů z Open Street Maps a pro zobrazení statistických grafů. V obou případech se k jejich vytvoření používá třída *ChartPlotter*, což je vlastně prostor pro další prvky grafu a pro graf samotný.

6.2.2 Mapa pomocí DDD

Pomocí třídy *Map* je vytvořen objekt pro prostor obrazu mapy a pomocí třídy *OpenStreetMapServer* určujeme, odkud se budou čerpat obrazová data. V DDD je použita mapa ze serveru Open Street Maps, kdy DDD umožňuje přibližování a ovládání mapy pomocí myši. Mapa je vlastně součástí grafu, kde osami jsou zeměpisná délka a šířka. Do grafu se dají vykreslovat přímky, body a tzv. "tahací bod (draggable point)", který je využíván pro změnu polohy určitého bodu v trase. Pro zlepšení orientace na mapě umožňuje DDD přiblížení podle okrajových bodů vykreslované trasy, kdy se podle nich v prostoru pro mapu nastaví viditelná část.

Protože Země není plochá, ale jedná se o geoid, je potřeba použít pro výpočet polohy bodu na mapě a pro výpočet osy zeměpisné šířky Merkatorovo zobrazení. Jeho implementaci lze nalézt ve třídě *MercatorTransform* v knihovně DDD.

Merkatorovo zobrazení

Mercatorovo zobrazení je druh válcového mapového zobrazení, které navrhl roku 1569 vlámský kartograf Gerhard Mercator (1512 – 1594). Základem zobrazení je válec v normální poloze (tedy rovnoběžný se zemskou osou). Po zobrazení povrchu koule na válec a po rozvinutí pláště válce do roviny vznikne pravoúhlá síť poledníků a rovnoběžek. Rovnoběžky od sebe nejsou stejně vzdálené a vzájemné rozdíly se od rovníku zvětšují v poměru sekanse zeměpisné šířky. I přes to, že Merkatorovo zobrazení je nepoužitelné v blízkosti rovníků, protože se blíží nekonečnu, tak pro navigaci do $\pm 85.05113°$ je stále základním zobrazením.

Následující rovnice 6.1–6.2 určují x a y souřadnice bodu na Merkatorově mapě z jejich latitudy ϕ, longitudy λ (s λ_0 jako středem mapy) a poloměru koule R:

$$x = R \cdot (\lambda - \lambda_0) \tag{6.1}$$

$$y = R \cdot ln\left(\arctan\left(\frac{\pi}{4} + \frac{\phi}{2}\right)\right). \tag{6.2}$$

Rovnice 6.3–6.4 pro opačný výpočet z bodů na mapě jsou pak:

$$\lambda = x + \lambda_0 \tag{6.3}$$

$$\phi = \frac{\pi}{2} - 2 \cdot \arctan\left(e^{\frac{y}{R}}\right). \tag{6.4}$$

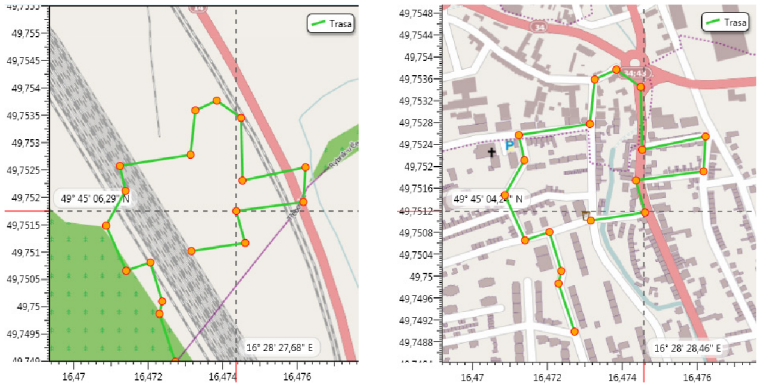
Původní Merkatorovo zobrazení v DDD nefungovalo správně a body byly umístěny jinde, než by ve skutečnosti měly být, viz obrázek 6.4a. Tento problém byl vyřešen nastavením přesnější konstanty pro výpočet bodu a změnou tří metod pro výpočet souřadnic ve třídě *MercatotProjection*, viz příklad 3 a obrázek 6.4b. Čerpáno z [10].

Příklad 3 Upravené metody ve třídě MercatorProjection.

```
// Proměnná s konstantou
private double maxLatitude = 85.06255;

// Metoda pro výpočet "poloměru" mapy s maximální latitudou z konstanty.
private void CalcScale(double maxLatitude)
{
  double maxLatDeg = maxLatitude;
  double maxLatRad = maxLatDeg * Math.PI / 180;
  scale = maxLatDeg / Math.Log(Math.Tan(maxLatRad / 2 + Math.PI / 4));
}

// Metoda pro výpočet pozice bodu na 2D mapě.
public sealed override Point DataToViewport(Point pt)
{
  double y = pt.Y;
  if (-maxLatitude <= y && y <= maxLatitude)
  {
    double latRad = pt.Y * Math.PI / 180;
    y = scale * Math.Log(Math.Tan(Math.PI / 4 + (latRad / 2)));
  }
  return new Point(pt.X, y);
}
// Metoda pro získání zeměpisné šířky a délky z bodu na 2D mapě
public sealed override Point ViewportToData(Point pt)
{
  double y = pt.Y;
  if (-maxLatitude <= y && y <= maxLatitude)
  {
    y = 90 - Math.Atan(Math.Exp(-y / scale)) / Math.PI * 360;
  }
  return new Point(pt.X, y);
}
```



(a) Špatný výpočet souřadnic. (b) Správný výpočet souřadnic.

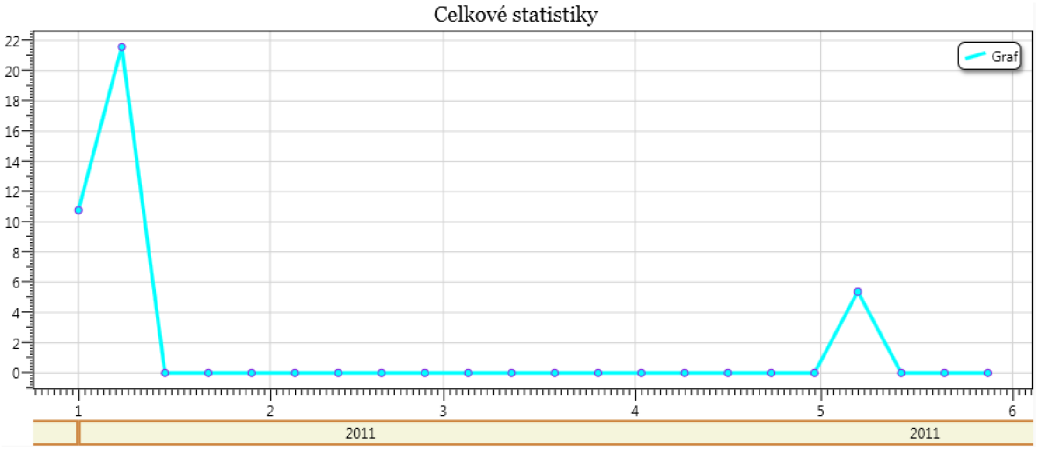
Obrázek 6.4: Porovnání vykreslení trasy pomocí (a) původního, (b) opraveného Mercatorova zobrazení.

6.2.3 Graf pomocí DDD

Prostor pro graf a jeho body v aplikaci tvoří třída *ChartPlotter*. Pro přidání grafu a jeho bodů do prostoru se používá metoda *AddLineGraph*, kdy pro potřeby aplikace používá parametry:

- datový zdroj bodů,
- grafická definice štětce pro vykreslení čar do grafu,
- grafická definice bodů grafu a
- textový popis vykreslovaného grafu.

Ukázka grafu vytvořeného pomocí DDD je na obrázku 6.5. Jako zdroj bodů se dají používat libovolná data, čehož je v aplikaci využito a bude blíže popsáno v části 6.4.



Obrázek 6.5: Graf v DDD.

6.3 LINQ

LINQ (zkratka z anglického Language Integrated Query) přináší nový způsob dotazování nad daty. Jedná se o poměrně obecný nástroj, takže je možné jeho pomocí manipulovat s různorodými daty. LINQ umožňuje například dotazy:

- pro kolekce objektů (LINQ to Objects),
- pro knihovnu ADO.NET (LINQ to ADO.NET), zahrnující jazyk SQL (LINQ to SQL), datové sady (LINQ to DataSets), entity (LINQ to Entities) a
- pro jazyk XML (LINQ to XML).

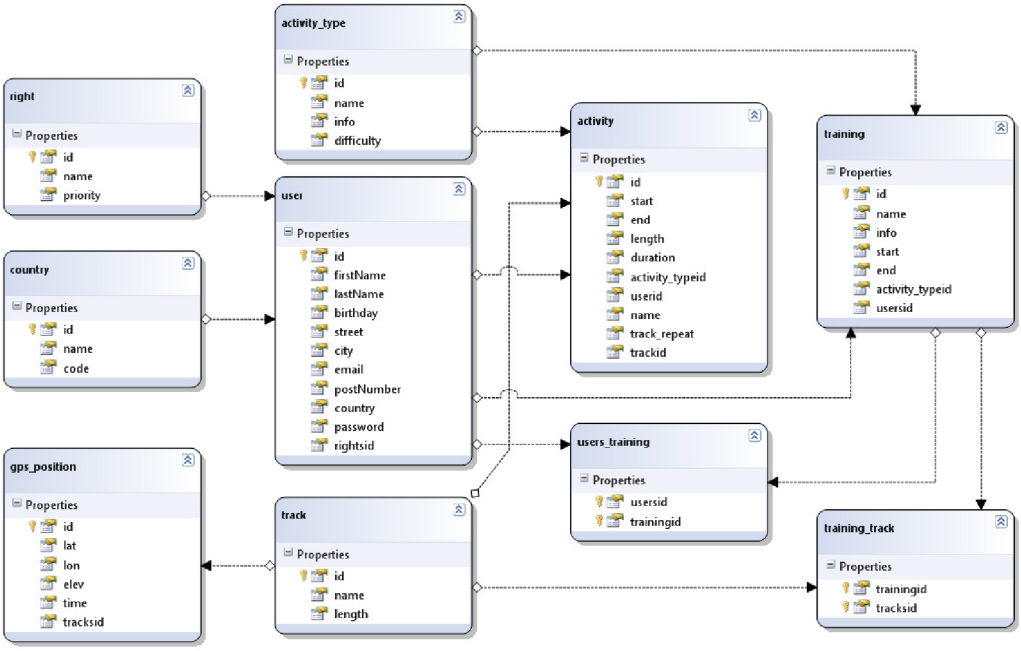
Na internetu jsou dostupné i další implementace LINQu.

6.3.1 Využití LINQ v aplikaci

Databáze je navržena pro Microsoft SQL Server 2005. Práce s databází je postavena na technologii LINQ to SQL. Pro naše účely využíváme komponenty LINQ, a to nejen pro práci s databází, ale i pro práci s kolekcemi.

Činnost komponenty LINQ to SQL je rozdělena do tří vrstev: vrstva Aplikace, LINQ to SQL vrstva a vrstva SQL Serveru. Dotazy píšeme v aplikační vrstvě pomocí syntaxe jazyku C#, ty jsou pak přeloženy do standardního SQL a odeslány serveru. Server odešle výsledek v podobě řádků tabulky a předá ho vrstvě SQL to LINQ, zde se výsledky převedou na objekty a jsou předány vrstvě aplikace, kde se s nimi může pracovat dále.

Pro automatizované vytvoření datových tříd podle návrhu databáze slouží ve vývojovém prostředí Visual Studia 2010 nástroj ORM designer. Do prostředí ORM designeru přeneseme tabulky z databáze MSSQL a designer pak automaticky vygeneruje třídy s příslušnými metodami pro reprezentaci atributů z tabulek. Dále se designer postará i o vygenerování vztahů mezi tabulkami a datového kontextu pro zpřístupnění tříd. Na obrázku 6.6 je pohled na ORM designer při realizaci návrhu databáze.



Obrázek 6.6: Návrh databáze pomocí OR Designeru.

Dotazy nad kolekcemi založené na metodách

Kolekcí, která může být použita v LINQ, je kterákoli kolekce, jež implementuje generické rozhraní *IEnumerable*⟨*T*⟩. S příchodem C# 3.0 bylo rozhraní *IEnumerable* rozšířeno o bohatou množinu nových metod, které slouží právě k provádění dotazů nad daty. Tato množina metod je označována také jako *standard query operators*. V těchto metodách je často nutné použít instanci jednoho z delegátů s názvem *Func*, které jsou formou generických argumentů a pomocí nich je určován typ vstupních argumentů a typ návratové hodnoty. Instance těchto delegátů jsou nejčastěji tvořeny pomocí lambda výrazů, viz příklad 4 demonstrující použití v aplikaci.

Příklad 4 Příklad dotazu nad kolekcemi založený na metodách.

```
// Výpočet průměrné vzdálenosti na aktivitu v určitém časovém intervalu, kdy výběr
// z určitého časového intervalu proběhne pomocí metody Where a pomocí
// metody Count získáme počet aktivit v zadaném intervalu.
dist = dist / activities.Where(act =>
  DateTime.Compare(
    (DateTime)act.start, new DateTime(tmp.Ticks + ts.Ticks * days)
  ) <= 0 &&
  DateTime.Compare(
    (DateTime)act.start, tmp) >= 0
  ).Count();
```

Dotazy nad kolekcemi pomocí klíčových slov

Použití dotazů je poměrně přehledný způsob, jak vybírat určité prvky z kolekcí, ale díky takzvaným *query keywords*, což je vlastně skupina nových klíčových slov pro realizaci LINQ

dotazů, lze vybírat ještě jednodušeji. *Query keywords* silně připomínají SQL s tím rozdílem, že projekce končí klíčovým slovem *select* a začíná slovem *from*, viz příklad 5 demostrující použití v aplikaci. Při výběru pomocí klíčových slov nemusí být explicitně určen datový typ a je stejně jako v případě klíčového slova *var* odvozen v době kompilace.

Příklad 5 Příklad dotazu nad kolekcemi pomocí klíčových slov

```
// Výběr všech aktivit podle ID uživatele.
var activities = from a in db.activities
                 where a.userid == userID
                 select a;
```

Vkládání dat (insert) do databáze pomocí LINQ

Vkládání jednoho prvku do tabulky se provádí pomocí metody *InsertOnSubmit* nad vybranou tabulkou z datového kontextu, kde vstupním parametrem je objekt podle datové třídy vybrané tabulky. Pokud je potřeba do tabulky vložit více prvků, použije se metoda *InsertAllOnSubmit*, kde vstupním parametrem je kolekce objektů podle datové třídy vybrané tabulky. Změny v datovém kontextu jsou pak potvrzeny pomocí metody *SubmitChanges*. Vkládání v aplikaci, viz příklad 6.

Příklad 6 Příklad vkládání dat do tabulky.

```
// Vložení nového řádku do tabulky s uživateli.
db.users.InsertOnSubmit(newUser);
db.SubmitChanges();
```

Úprava dat (update) v databázi pomocí LINQ

Pokud se vybraný objekt podle datové třídy změní a chceme změny uchovat, pak musí být potvrzeny pomocí metody *SubmitChange*.

Mazání dat (delete) v databázi pomocí LINQ

Pokud je potřeba vymazat z tabulky data, používají se metody *DeleteOnSubmit* pro vymazání jednoho prvku a *DeleteAllOnSubmit* pro vymazání více prvků nad vybranou tabulkou z datového kontextu. Vstupními parametry jsou pak vybraný objekt podle datové třídy tabulky nebo více takových objektů, pokud chceme mazat více prvků. Jsou-li odstraňovány záznamy z tabulky, kde tabulka, ze které mažeme, je v relaci s jinou tabulkou, pak musíme nejdříve vymazat záznam z této tabulky. Změny v datovém kontextu jsou pak potvrzeny pomocí metody *SubmitChanges*. Příklad použití v aplikaci, kde je mazán záznam z tabulky *tracks* (pro trasy), který je v relaci s tabulkou *gps_positions* (pro GPS souřadnice bodu trasy), je ukázán na příkladu 7.

Příklad 7 Příklad mazání dat z tabulky.

```
// Pokud má vybraná trasa gps souřadnice tak je odstraň a následně
// odstraň i samotnou trasu a ulož změny.
if (((DB.track)tracksGridData.SelectedItem).gps_positions != null){
    db.gps_positions.DeleteAllOnSubmit(
      ((DB.track)tracksGridData.SelectedItem).gps_positions
    );
}
db.tracks.DeleteOnSubmit(((DB.track)tracksGridData.SelectedItem));
db.SubmitChanges();
```

6.4 Zobrazení bodů trasy, statistik a možnost exportu

Jak již bylo popsáno v 6.2.1, pro zobrazení trasy a jejich bodů se využívá třídy *ChartPlotter* a metody *AddLineGraph*. V této sekci bude popsáno, jak jsou tvořeny datové zdroje pro grafy tras, statistik a jejich bodů pomocí vlastních tříd.

6.4.1 Třída PointByTime

Tato třída slouží jako datový zdroj pro uchování nějaké hodnoty v čase u celkových statistik. Například jaká vzdálenost byla uražena určitý den.

6.4.2 Třída StatPoint

Třída *StatPoint* je používaná jako datový zdroj pro uchování statistických dat určité aktivity, kdy se zadává zeměpisná šířka, zeměpisná délka, výška, vzdálenost, čas, tempo, rychlost, počáteční čas a aktivita, ke které se data vztahují.

6.4.3 Třída GPS

Data potřebná pro třídu GPS

Aby třída *GPS* mohla správně fungovat, potřebuje znát body trasy (datová třída *gps_position*) a aktivitu (datová třída *activity*), ke které se body vztahují.

Body trasy

Třída *GPS* je základní třídou pro tvorbu datového zdroje s body trasy, které se budou zobrazovat na mapě. K tomuto účelu slouží metoda *getMapPoints*, která vrací objekt typu *IPointDataSource* z knihovny DDD obsahující souřadnice grafu pro osy X a Y. Pro zrychlení tvorby metoda vytvoří dvě vlákna, kde se každé stará o body jedné z os.

Výpočet vzdálenosti mezi body trasy

K výpočtu vzdálenosti mezi jednotlivými body trasy používá aplikace rovnici *haversine* [11], což je nejkratší kruhová vzdálenost mezi body zemského povrchu ignorující převýšení. Nepřesnost této metody spočívá v tom, že bere jako model Země kouli, ale skutečný tvar Země se od tvaru koule mírně liší. Poloměr zeměkoule je aproximován jako $R = 6371km$

a pro výpočet vzdálenosti jsou použity rovnice 6.5–6.9, kde d je vzdálenost mezi body a θ zastupuje Δlat a Δlon.

$$\Delta lat = lat_2 - lat_1 \tag{6.5}$$

$$\Delta long = long_2 - long_1 \tag{6.6}$$

$$haversin\left(\frac{d}{R}\right) = haversin(\Delta lat) + \cos(lat_1) \cdot \cos(lat_2) \cdot haversin(\Delta long) \tag{6.7}$$

$$haversin(\theta) = \sin^2\left(\frac{\theta}{2}\right) \tag{6.8}$$

$$haversin\left(\frac{d}{R}\right) = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2\left(\frac{\Delta long}{2}\right). \tag{6.9}$$

Pro výpočet vzdálenosti mezi body podle daného poloměru je použita inverzní rovnice *haversine* 6.10–6.11.

$$d = R \cdot haversin^{-1}\left(haversin\left(\frac{d}{R}\right)\right) \tag{6.10}$$

$$d = R \cdot 2 \cdot arcsin\left(haversin\left(\frac{d}{R}\right)\right). \tag{6.11}$$

Na příkladu 8 je ukázána implementace rovnice *haversine* pomocí jazyku C# ve třídě *GPS* v metodě *distHaversine*, která očekává na vstupu body zeměpisné šířky a délky, aby z nich dokázala vypočítat vzdálenost.

Příklad 8 Implementace haversine vzorce pomocí C#.

```
Double R = 6371;
Double dLat = DegreeToRadian(lat2 - lat1);
Double dLon = DegreeToRadian(lon2 - lon1);
Double a = Math.Sin(dLat / 2) * Math.Sin(dLat / 2) +
           Math.Cos(DegreeToRadian(lat1)) * Math.Cos(DegreeToRadian(lat2)) *
           Math.Sin(dLon / 2) * Math.Sin(dLon / 2);
Double c = 2 * Math.Asin(Math.Sqrt(a));
Double d = R * c;
```

Délka trasy a statistiky bodů trasy

Výpočet délky trasy a statistika bodů trasy probíhá v metodě *countLength*. Celková délka trasy se vypočítá jako součet vzdáleností mezi jednotlivými body. Dále se vypočítávají údaje pro statistické body trasy (podle třídy *StatPoint*). Rychlost mezi dvěma body se počítá dvěma způsoby, pokud je všem bodům trasy přidělený čas, tak vždy rychlost v km/h mezi dvěma body, nebo pokud není přidělený čas, tak v km/h podle údajů z aktivity (celkové trvání aktivity a uložená délka aktivity). Tempo se pak vypočítá jako počet minut na kilometr. Zbytek údajů pro statistické body se vypočítávat nemusí a pouze se přidělí.

Vyhlazené body do grafu statistiky

Vzhledem k tomu, že výsledné hodnoty rychlosti a tempa pro body statistických grafů jsou velice rozdílné a graf byl nepřehledný, bylo potřeba hodnoty vyhladit pro větší přehlednost. K tomuto účelu byla implementována metoda *getAntiAliasedStatPoints*. Pro vyhlazení grafu rychlosti a tempa aktivity využívá aplikace *prostorového klouzavého průměru* [3], který nahrazuje původní hodnoty pro každý bod váženým průměrem okolních bodů. Přidělované váhy jsou jednoduché a bez přímého vztahu k velikosti okolí. Základní vztah pro výpočet nové hodnoty bodu je pak:

$$z_i = \frac{\sum_{j-1}^{n} w_{ij} \cdot z_j}{\sum_{j-1}^{n} w_{ij}}, \tag{6.12}$$

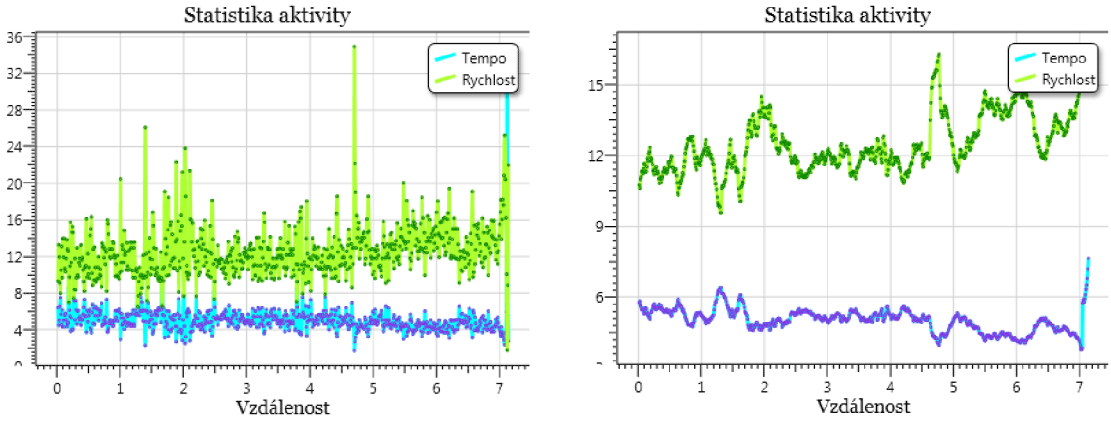
kde

- z_j – původní hodnoty sousedních bodů,
- w_{ij} – váha hodnoty sousedního bodu j pro výpočet z místa i,
- j – index vymezující sousední body,
- z_i – vyhlazená hodnota z v bodě i.

Samotná implementace *prostorového klouzavého průměru* v aplikaci neřeší vážené průměry, protože se nedá určit, která statistická hodnota je významnější. Algoritmus v jazyce C# pak je ukázán na příkladu 9. Rozdíl mezi vyhlazeným a nevyhlazeným grafem je ukázán na obrázku 6.7.

Příklad 9 Implementace prostorového klouzavého průměru.

```
// Proměnná pro rychlost
double speed = 0;
// Proměnná pro tempo
double pace = 0;
// Proměnná pro určení velikosti okolí
double distance = 5;
// V cyklu proběhne suma rychlostí a temp z okolí
for (int j = -((distance - 1) / 2); j <= (distance - 1) / 2; j++)
{
  speed += points[realDistance + j].Speed;
  pace += points[realDistance + j].Pace;
}
// Sumy se vydělí sumou váženého průměr bodů z okolí.
pace = pace / distance;
speed = speed / distance;
```



(a) Nevyhlazený graf.

(b) Vyhlazený graf.

Obrázek 6.7: Rozdíl mezi (a) nevyhlazeným a (b) vyhlazeným grafem.

Import aktivity

V části 3.4.1 a 3.4.2 byly popsány dva možné typy XML formátů (GPX a KML) pro uchovávání dat z GPS. Pomocí metod *importKML* a *importGPX* aplikace získá potřebné GPS údaje o trase aktivity, které bude možné dále uložit do databáze.

Export aktivity

Aplikace umožňuje export aktivity do souboru ve formátu XML a XSL. Popis výstupního XML souboru je v příloze B.

6.4.4 Třída OveralStats

Tato třída slouží k výpočtu bodů, které se vykreslují na grafu pro celkové statistiky aktuálně přihlášeného uživatele. V aplikaci je možné zobrazit:

- Celkovou uraženou vzdálenost uživatele podle dnů, týdnů a měsíců.
- Průměrnou uraženou vdálenost na aktivitu uživatele podle dnů, týdnů a měsíců.
- Průměrnou rychlost na aktivitu uživatele podle dnů, týdnů a měsíců.
- Průměrné tempo na aktivitu uživatele podle dnů, týdnů a měsíců.

Data potřebná pro třídu OveralStats

Aby třída *OveralStats* mohla správně fungovat, potřebuje znát identifikační číslo přihlášeného uživatele, podle kterého pak vybere všechny jeho aktivity.

Výběr aktivit a výpočet statistických bodů

V konstruktoru je vytvořeno pole objektů třídy *GPS* (pro každou trasu a její aktivitu) a pole vláken, kdy každé vlákno obsluhuje vykonání metody *countLength* (viz část 6.4.3) ze třídy *GPS* z toho důvodu, že výpočet může probíhat paralelně a zrychluje tím chod aplikace.

Výpočet celkových statistik

Výpočet celkových statistik z bodů aktivit probíhá tak, že se podle uživatelova výběru nastaví časové období, ve kterém chce statistiky sledovat (např. měsíčně), vytvoří se kolekce objektů podle třídy *PointByTime* (viz část 6.4.1) a takto vytvořená kolekce pak bude datovým zdrojem pro graf a jeho body.

Export statistiky

Aplikace umožňuje export celkové statistiky do souboru ve formátu XML a XSL. Popis výstupního XML souboru je v příloze C.

6.4.5 Třída TrainStats

Tato třída slouží k výpočtu bodů, které se vykreslují na grafu pro zjištění dodržení zvoleného tréninku aktuálně přihlášeného uživatele.

Data potřebná pro třídu TrainStats

Aby třída *TrainStats* mohla správně fungovat, potřebuje znát zvolený trénink, podle kterého pak vybere všechny trasy a zadané atributy tréninku, jako je doporučená vzdálenost, rychlost a tempo.

Výpočet bodů grafu

Pomocí metody *countPoints* se vypočítají statistické body třídy *StatPoint*, které pak umožňují snadné zobrazení na výsledném grafu.

6.5 Třídy pro konvertory

Tyto třídy slouží ke kontrole a případné změně hodnot zadávaných nebo zobrazovaných v prvcích aplikace. Každý konvertor je vytvořen pomocí rozhraní *IValueConverter* a implementuje metody *Convert* a *ConvertBack*, které složí pro převody mezi zdrojem a cílem. Ukázka jednoduchého konvertoru pro kontrolu správnosti zadaného čísla bez desetinné čárky je na příkladu 10.

6.5.1 ToHourRangeConverter

Slouží ke kontrole správnosti zadaného čísla, které musí být v rozmezí 0–23. Používá se při zadávání hodiny u času.

6.5.2 ToMinSecRangeConverter

Slouží ke kontrole správnosti zadaného čísla, které musí být v rozmezí 0–60. Používá se při zadávání minut nebo sekund u času.

6.5.3 ToFloatValueConverter

Slouží ke kontrole správnosti zadaného čísla, které musí být číslem s desetinnou čárkou. Používá se při zadávání čísel s desetinnou čárkou, například doporučená rychlost u tréninků.

6.5.4 ToIntValueConverter

Slouží ke kontrole správnosti zadaného čísla, které musí být číslem bez desetinné čárky. Používá se při zadávání čísel bez desetinné čárky, například počet opakování trasy.

6.5.5 RoundConverter

Složí k zaokrouhlení zobrazovaného čísla na dvě desetinná místa.

6.5.6 DateTimeConverter

Složí k zobrazení datumu ve vybraném formátu.

Příklad 10 Příklad konvertoru v aplikaci pro kontrolu čísla bez desetinné čárky.

```
public class ToIntValueConverter : IValueConverter
{
    public object Convert(object value, Type targetType, object parameter,
        System.Globalization.CultureInfo culture)
    {
        String val = (String)value;
        try
        {
            Int32.Parse(val);
        }
        catch
        {
            return 0;
        }
        return value;
    }

    public object ConvertBack(object value, Type targetType, object parameter,
        System.Globalization.CultureInfo culture)
    {
        return value;
    }
}
```

Kapitola 7

Plán dalších rozšíření aplikace

7.1 Databáze

Jak bylo napsáno v 4.3.3, ideální architekturou by byla architektura se synchronizací místní a vzdálené databáze. Toto je jedním z možných dalších rozšíření aplikace, protože by se dala používat na jakémkoliv počítači a uživatel by měl přístup ke všem svým datům. Aby bylo možné tohoto dosáhnout, bude nutné zařídit server s MSSQL, který bude přístupný přes internet a doprogramovat metody pro synchronizaci databází.

7.2 Další údaje o aktivitách

Při tréninku je také důležité sledovat srdeční tep, který se dá automaticky zjišťovat a sledovat pomocí *sporttesterů*. Pro účely aplikace by bylo vhodné sehnat takový *sporttester*, který dokáže sledovat tep, zaznamenávat GPS souřadnice a umožňovat export těchto statistik do formátu, ze kterého by se dala tato data získat. Byla snaha o zapůjčení sporttesteru, což se bohužel nakonec nezdařilo.

7.3 Další aktivity

Posledním z možných rozšíření by mohlo být i zadávání jiných aktivit, což už nesouvisí přímo s běžeckým deníkem, ale spíše s tréninkovým deníkem obecně, kdy by se mohly zadávat aktivity z posilovny nebo z plavání.

Kapitola 8

Závěr

Cílem práce bylo vytvoření uživatelsky příjemné aplikace pro vedení běžeckého deníku s možností analýzy vložených dat.

Úvod teoretické části práce je věnován seznámení s běžeckým deníkem, vlastnostmi, které by měl obsáhnout, přehledu a popisu již existujících aplikací pro vedení běžeckého deníku. Poté následuje kapitola s krátkou charakteristikou systému GPS, zařízení pro záznam pozice GPS a vybraných aplikací poskytujících tuto službu. Další sekce se věnuje formátům pro uložení tras ze zařízení zaznamenávajících GPS pozici. Důležitou kapitolou je návrh aplikace, kde jsou sepsány a graficky prezentovány požadavky na její funkčnost. Podle předchozích požadavků následuje popis rozdělení rolí uživatelů v systému, různých možností architektury aplikace, návrh vhodné databáze a tříd. Poslední kapitolou teoretické části práce je návrh vhodného formátu pro export dat z aplikace.

V popisu praktické části práce je jak krátký popis použitých technologií, tak samotné postupy a příklady implementace aplikace. Za pomoci *Windows Presentation Foundation* (dále jen *WPF*) je vytvořeno uživatelské rozhraní aplikace, které umožňuje snadné a intuitivní ovládání aplikace. Při tvorbě uživatelského rozhraní nedocházelo k větším komplikacím hlavně díky výborné dokumentaci a velkému množství příkladů, tutorialů a relevantních příspěvků v internetových diskuzích. S daty z databáze a datových kolekcí je v aplikaci manipulováno pomocí technologie *LINQ*, která je stejně jako *WPF* kvalitně zdokumentována a je k dispozici mnoho příkladů, návodů jak postupovat a relevantních příspěvků v diskuzích, tudíž ani zde nedocházelo k větším problémům. K vykreslování grafů pro statistiky uživatele, bodů trasy a mapového zdroje je použita volně šiřitelná knihovna *Dynamic Data Display* (dále jen *DDD*), která naopak není příliš zdokumentována[1] a i některé její metody bylo třeba upravit pro lepší funkčnost. Po prozkoumání příkladů, které byly ke knihovně *DDD* k dispozici, ji bylo možné kvalitně využít ve vlastní aplikaci. Klíčovou částí aplikace jsou výpočty bodů trasy, statistik a možnost jejich exportu do formátu XML a XLS, k čemuž byly vytvořeny vlastní třídy, ty jsou popsány v části 6.4. Tato část byla nejzajímavější, protože se naskytla možnost prozkoumat různé nové algoritmy a využít možností moderního jazyku C#, jako jsou konvertory, snadná práce s vlákny a další. Závěr práce je pak věnován možným dalším rozšířením vhodných k doplnění aplikace pro její větší využití.

[1] Tvrzení platné k 14.3.2011.

Literatura

[1] Baběrád, P.: Běžecký deník. [online], [cit. 16.10.2010].
URL: <http://www.sportsite.cz/motivace/kdyz-potrebuji-povzbudit/bezecky-denik.html>

[2] Brinchuk, M.: WPF Dynamic Data Display. [online], [cit. 14.3.2011].
URL: <http://dynamicdatadisplay.codeplex.com/>

[3] Doc. Dr. Ing. Horák, J.; Dr. Ing. Horáková, B.: Prostorové vyhlazování dat s areálovou reprezentací. [online], 2003.
URL:
<http://gis.vsb.cz/pan-old/Skoleni_Texty/TextySkoleni/ProstorVyhlaz.pdf>

[4] Foster, D.: GPX: the GPS Exchange Format. [online], [cit. 18.12.2010].
URL: <http://www.topografix.com/gpx.asp>

[5] Google, I.: KML Documentation Introduction. [online], [cit. 22.12.2010].
URL: <http://code.google.com/intl/cs/apis/kml/documentation/>

[6] Google, I.: My Tracks for your Android. [online], [cit. 26.12.2010].
URL: <http://mytracks.appspot.com/>

[7] National Space-Based Positioning, N.; Office, T. C.: Global Positioning System. [online], [cit. 1.11.2010].
URL: <http://www.gps.gov/systems/gps/>

[8] OBSERVATORY, U. S. N.: BLOCK II SATELLITE INFORMATION. [online], [cit. 27.11.2010].
URL: <ftp://tycho.usno.navy.mil/pub/gps/gpsb2.txt>

[9] Skonnard, A.; Gudgin, M.: *XML – pohotová referenční příručka*. Grada Publishing, a.s., 2006.

[10] Snyder, J. P.: Map Projections: A Working Manual. [online], [cit. 24.4.2011].
URL: <http://pubs.er.usgs.gov/publication/pp1395>

[11] Veness, C.: Movable Type Scripts. [online], [cit. 14.2.2011].
URL: <http://www.movable-type.co.uk/scripts/latlong.html>

[12] Wikipedia: Global Positioning System. [online], [cit. 14.11.2010].
URL: <http://cs.wikipedia.org/wiki/Global_Positioning_System>

[13] Woessner, M.; Koehne, A.: GPS explained. [online], [cit. 3.1.2011].
URL: <http://www.kowoma.de/en/gps/index.htm>

[14] WWW stránky: Běh a motivace v tréninku – web o běhání sportsite.cz. [online], [cit. 18.10.2010].
URL: <http://www.beh.sportsite.cz/motivace/>

[15] WWW stránky: MyTreneek – nejen tréninkový deník. [online], [cit. 22.10.2010].
URL: <http://www.ondrej-vojtechovsky.cz/mytreneek>

[16] WWW stránky: RunKeeper. [online], [cit. 24.10.2010].
URL: <http://runkeeper.com>

Příloha A

Obsah DVD

Na DVD jsou zdrojové kódy aplikace, projekt pro Visual Studio 2010, instalační balíček aplikace a krátký manuál k aplikaci.

Příloha B

XML schéma pro export aktivity

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="activity">
    <xs:complexType>
      <xs:sequence >
        <xs:element name="description">
          <xs:complexType>
            <xs:all>
              <xs:element name="name" nillable="true">
                <xs:complexType>
                    <xs:attribute name="text" type="xs:string" />
                </xs:complexType>
              </xs:element>
              <xs:element name="start" nillable="true">
                <xs:complexType>
                    <xs:attribute name="time" type="xs:dateTime"/>
                    <xs:attribute name="lat" type="xs:double"/>
                    <xs:attribute name="lon" type="xs:double"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="end" nillable="true">
                <xs:complexType>
                    <xs:attribute name="time" type="xs:dateTime"/>
                    <xs:attribute name="lat" type="xs:double"/>
                    <xs:attribute name="lon" type="xs:double"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="length" nillable="true">
                <xs:complexType>
                    <xs:attribute name="km" type="xs:double"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="trackRepeat" nillable="true">
                <xs:complexType>
                    <xs:attribute name="times" type="xs:integer"/>
                </xs:complexType>
              </xs:element>
              <xs:element name="activityType" nillable="true">
                <xs:complexType>
```

```
                    <xs:attribute name="text" type="xs:string"/>
                  </xs:complexType>
                </xs:element>
              </xs:all>
            </xs:complexType>
          </xs:element>
          <xs:element name="track">
            <xs:complexType>
              <xs:sequence>
                <xs:element name="trackPoint" nillable="true" maxOccurs="unbounded">
                  <xs:complexType>
                      <xs:attribute name="time" type="xs:dateTime"/>
                      <xs:attribute name="lat" type="xs:double"/>
                      <xs:attribute name="lon" type="xs:double"/>
                      <xs:attribute name="elev" type="xs:double"/>
                      <xs:attribute name="speed" type="xs:double"/>
                      <xs:attribute name="pace" type="xs:double"/>
                      <xs:attribute name="dist" type="xs:double"/>
                  </xs:complexType>
                </xs:element>
              </xs:sequence >
            </xs:complexType>
          </xs:element>
        </xs:sequence >
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

Příloha C

XML schéma pro export celkových statistik

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="overalstats">
    <xs:complexType>
      <xs:sequence >
        <xs:element name="points">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="statPoint" nillable="true" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:attribute name="time" type="xs:dateTime"/>
                    <xs:attribute name="value" type="xs:double"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence >
          </xs:complexType>
        </xs:element>
      </xs:sequence >
    </xs:complexType>
  </xs:element>
</xs:schema>
```