Západočeská univerzita v Plzni

Fakulta aplikovaných věd

Katedra informatiky a výpočetní techniky

Bakalářská práce

Plánování VFR letů – Webová aplikace s využitím veřejně dostupných geografických služeb

Plzeň, 2012 Martin Bydžovský

Prohlášení

Prohlašuji, že jsem bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 1. 5. 2012 Martin Bydžovský

Poděkování

Na tomto místě bych rád poděkoval vedoucímu práce, panu Ing. Kamilu Ekšteinovi, Ph.D., za přínosné rady a pomoc při řešení práce. Dále pak mé rodině za podporu při studiu a v neposlední řadě také mojí přítelkyni Evě Janouškové za korekturu.

Abstract

The main target of this thesis is to create a modern, usable web application that will help pilots to prepare their flights according to the Visual Flight Rules (VFR) specification. VFR is one of ways how to manage flights, using only visual reference to the ground.

To achieve this, my first objective was to learn and understand methods of the VFR ground preparation so that the application reflects and meets requirements of pilots and therefore can be a suitable assistant to them.

Second goal was to explore various technologies and gather information about available services for displaying and handling interactive, user-editable maps on the Internet.

Finally, last part of the thesis was about choosing the most suitable technology and developing the application according to the previously analyzed situation and knowledge obtained.

Obsah

1. Úvod

Cílem práce je seznámit se s různými volně dostupnými knihovnami a nástroji pro zobrazování interaktivních map na internetu a vybrat tu, která se bude nejvíce hodit k vývoji webové aplikace, která usnadní pilotům naplánovat trasu letu pro let za VFR, čili letu podle srovnávací navigace, nikoliv podle přístrojů.

První část práce popisuje základní principy létání podle pravidel VFR, obeznamuje s jednotlivými pojmy a vysvětluje důležité aspekty tohoto druhu létání. Dále se práce soustředí na nalezení a prozkoumání geografických služeb, možnosti jejich využití, srovnání jejich API a výběr té nejvhodnější. Spolu s výběrem knihovny pro zobrazování map bylo nutné vyřešit otázku, jaký serverový jazyk a databázi použít, případně zda nasadit nějaký webový framework, který by zjednodušil práci při implementaci méně zajímavých částí kódu.

Poslední část práce se věnuje detailům implementace jednotlivých částí webové aplikace, přehledu důležitých komponent a ukázkám zajímavých částí zdrojového kódu. Také jsou rozebrány konkrétní algoritmy, které byly při práci použity.

2. Visual Flight Rules

Visual Flight Rules (VFR) [1] [2] je soubor pravidel, podle kterých pilot provozuje let na základě vizuální reference. Musí mít výhled z kokpitu letadla, aby byl schopen vizuálně určit, kde se letadlo nachází, kam směřuje, případně reagovat na situaci ve vzduchu bez použití přístrojů. Řídící orgány stanoví zvláštní předpisy, za kterých je VFR létání povoleno. Jedná se zejména o požadavky na počasí – minimální viditelnost, vzdálenost od mraků, povětrnostní podmínky nebo nadmořská výška. Meteorologické podmínky splňující požadavky pro VFR létání se nazývají vizuální meteorologické podmínky (VMC – Visual Meteorological Conditions). V opačném případě je nutné létat za podmínek IFR – Instrument Flight Rules, tj. létání podle přístrojů. V závislosti na kategorii vzdušného prostoru, ve kterém se let provádí, může být po letadle požadováno, aby bylo vybaveno odpovídačem (transpondérem). To pomůže stanovišti řízení letového provozu (ATCC – Air Traffic Control Center) lépe identifikovat letadlo na radaru a věž může dávat letounu specifické rady ohledně VFR.

Na následujícím obrázku je vidět výhled z kokpitu VFR pilota:

Obrázek 1

2.1. Příprava a plánování letu

Před samotným provedením VFR letu je nutné vykonat navigační přípravu. Výstupem této přípravy může být navigační štítek nebo letový plán, případně oba dokumenty.

2.1.1. Letový plán

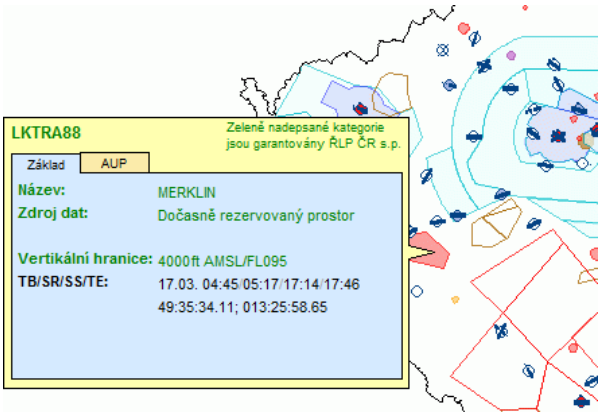
Letový plán je oficiální dokument, který pilot může a nemusí podat středisku řízení letového provozu (ŘLP). Jsou v něm uvedeny údaje o letadle, jeho parametry, počet pasažérů a plánovaný čas odletu a příletu. Dále lze uvést kontrolní body, přes které se poletí. Pilot se poté hlásí ŘLP jak při průletu jednotlivými body, tak při přistání. ŘLP průběžně kontroluje, zda se pilot hlásí, a pokud by došlo k nějakému problému a pilot se neozval více jak 30 minut od plánu, vyhlásí po něm pátrání. Tento dokument je tedy důležitý hlavně pro ŘLP.

2.1.2. Navigační štítek

Na rozdíl od letového plánu, navigační štítek je dokument určený pouze pro pilota. Jsou na něm zaznamenány body, přes které poletí, azimuty a vzdálenosti mezi jednotlivými body a frekvence leteckých radiostanic, kolem kterých se letí. Právě na základě údajů ze štítku pilot provádí navigaci za letu.

2.1.3. Tvorba navigačního štítku

Pilot si nejprve rozmyslí trasu letu. Při tom musí vzít v úvahu zóny s omezeným letovým provozem – vzdušné prostory, kam smí s letadlem vletět pouze s povolením od některého stanoviště letového provozu. Některé zóny jsou aktivní jen někdy, některé platí pouze v určitých letových hladinách (nadmořských výškách) a do určitých zón pilot nesmí vletět vůbec (např. vojenské výcvikové oblasti). Aktuální přehled zón lze nalézt na webu ŘLP [3] – *http://aisview.rlp.cz/*. Jak aplikace vypadá je vidět na obrázku 2. Aktivace zón se vyhlašuje vždy na další den ve 14:00.



Obrázek 2

Pilot musí při plánování bodů trasy (tzv. otočných bodů či fixů) brát v úvahu omezení vyplývající z existence těchto zón. Poté, co má v mapě zakresleny jednotlivé body, musí vypočítat azimuty a vzdálenosti mezi otočnými body. Azimut je vodorovný úhel mezi vertikální rovinou obsahující směrový vektor narýsované úsečky a rovinou místního poledníku. Změřený azimut se zapíše do navigačního štítku. Na základě znalosti měřítka mapy se určí vzdálenost mezi body, která se také zapíše do štítku. Pokud pilot navíc ví, jak rychle poletí, může si do štítku zanést i informaci o době, jakou mezi body poletí. V neposlední řadě si pilot do štítku zapíše informace o blízkých radiostanicích nacházejících se na trase letu v případě, že by se dostal do nesnází a potřeboval by pomoc.

3. Dostupné geografické služby

Tato sekce se věnuje nalezeným a prozkoumávaným veřejně dostupným geografickým službám, jejich možnostem a jejich srovnání. Bylo potřeba najít službu, která poskytuje kvalitní a dostatečně zdokumentované API pro interaktivní práci s mapou, jako např. přidávání uživatelských bodů, tvorbu vlastních tras i mimo silnice, a která usnadní práci při navigaci – GPS souřadnice, měření vzdáleností, počítání azimutů, apod. Velký plus při výběru služby by představovala existence nějaké komunity vývojářů, případně fórum, kam se může vývojář obrátit s problémem, který by mu zainteresovaní lidé pomohli vyřešit. Dále, vzhledem k tomu, že se plánuje reálné nasazení této aplikace, by se mělo jednat o stabilní službu, ve které se API nebude měnit každý týden a jejíž běh se očekává v řádu let.

3.1. OpenStreetMap

OpenStreetMap je služba, která se inspirovala projekty jako např. Wikipedia [4]. Jedná se o službu zdarma, která je navíc uživatelsky editovatelná, takže každý může přispívat a rozšiřovat ji. Nabízí velice podrobné informace o adresách (ulice jsou často uvedeny včetně čísel popisných a orientačních). Bohužel není k dispozici satelitní pohled, který se k VFR plánování velice hodí, a hlavně nedisponuje žádným API, přes které by šla mapa dále modifikovat. Pro účely této práce je tedy nevhodná.

3.2. Bing Maps

Bing Maps jsou kvalitní mapy od společnosti Microsoft s velmi podrobnou dokumentací ve stylu MSDN [5]. Podle dokumentace umožňuje jejich API všechny požadované akce, zobrazování tzv. Pushpinů – uživatelsky definované body, jejich spojování do Polyline – křivka spojující vzdušnou čarou jednotlivé Pushpiny. Práce se souřadnicemi GPS je samozřejmostí. K tomu, aby mapy na webové stránce běžely, je třeba získat od společnosti Microsoft tzv. *API KEY*. Získání tohoto klíče je podmíněno registrací na webu Microsoft Bing Maps (je nutný *Live* účet), kde se vyplní základní údaje o aplikaci.

Mezi slabé stránky této služby patří poměrně malá podrobnost satelitní mapy pro Českou republiku. Více znepokojujícím faktem je však to, že tutoriál pro seznámení se s programovým rozhraním nefungoval v době psaní této práce v prohlížeči Opera. Při posouvání mapy se hýbal i celý kontejner s mapou a začal překrývat všechen text

kolem. Přestože Opera nepatří mezi majoritní prohlížeče, tento nedostatek mě, jakožto vývojáře webové aplikace, zaskočil. I přes tento nedostatek se jedná o jednu z vhodných služeb k realizaci této práce.

3.3. Mapy.cz

Mapy.cz zastupují geografickou službu od společnosti Seznam [6]. Pro uživatele z České republiky určitě skvělá volba. Jedná se o kvalitní a podrobné satelitní mapy včetně podrobného pokrytí České republice s lokalizací do češtiny. Na vysoké úrovni je také podrobnost adres a čísel popisných, což ovšem nemá pro potřeby VFR velký význam.

Bohužel, při pohledu do API jsem zjistil, že služba nesplňuje požadavek na kvalitní dokumentaci s dostatkem ukázkových příkladů. Dokumentace je psaná neformálně, oficiální fórum, které je vedené jako jedno vlákno kdesi na serveru *lide.cz*, nepůsobí jako robustní služba se silným pozadím. Dále styl, jakým je napsán programový kód (tzv. coding standards), a hlavně jednotnost napříč API a dokumentací této služby, je na nižší úrovni, než bych si u takové služby představoval. Z těchto důvodů jsem označil službu *mapy.cz* za nevhodnou.

3.4. Google Maps

Google Maps je asi nejznámější mapová služba, která pochází od společnosti Google [7]. Podobně jako Bing Maps nabízí její API (které je mimochodem, až na terminologii, velice podobné tomu od společnosti Microsoft) vše potřebné pro realizaci této práce. Obsahuje kvalitní dokumentaci, velké množství ukázkových příkladů a silnou komunitní základnu. Tyto argumenty dělají ze služby Google Maps dalšího vhodného kandidáta.

Navíc, u společnosti tohoto formátu se dá předpokládat, že i v případě vydání nové verze API, zůstane stará verze ještě delší dobu podporována. Ostatně toto je vidět i nyní: Aktuální verzí je Google Maps API v3, které vyšlo v létě 2009. Současně s tím bylo API v2 označeno za zastaralé. Přesto jsou stávající aplikace běžící na této verzi stále funkční.

3.5. Google Earth

Jedná se o službu opět od společnosti Google, která umožňuje zobrazovat plně interaktivní 3D mapy tak, jak se zobrazují ve stejnojmenném programu [8]. Vzhledem k tomu, že trojrozměrné zobrazení přináší spoustu možností, je i API této služby velice rozsáhlé. Protože rozsah a zaměření této práce nemá pro trojrozměrné mapy využití, tato služba je příliš komplexní, a tudíž nevhodná.

3.6. Srovnání Google Maps a Bing Maps

Obě služby disponují srovnatelnými možnostmi, jak zobrazovat interaktivní mapu a jak s ní pracovat. Zde je srovnání základních metod pro práci s mapou [5] [7]:

- Inicializace mapy:

```
Bing Maps:
var mapOptions = {
  credentials:"APIKEY", height: 400, width: 400, showMapTypeSelector: false,
  mapTypeId: Microsoft.Maps.MapTypeId.road
};
var map = new Microsoft.Maps.Map(document.getElementById("mapDiv"),
      mapOptions);

Google Maps:
var mapOptions = {
  zoom: 12, center: pilsen, mapTypeId: google.maps.MapTypeId.SATELLITE
};
var map = new google.maps.Map(document.getElementById('map_canvas'),
      mapOptions);
```

- Přidání uživatelského bodu do mapy:

```
Bing Maps:
var center = map.getCenter()
var pin = new Microsoft.Maps.Pushpin(center, {
  height:50, width:50, anchor:new Microsoft.Maps.Point(lat,lng),
  draggable: true
});
map.entities.push(pin);

Google Maps:
var locationMarker = new google.maps.Marker();
locationMarker.setOptions({
  draggable: true, map: map, title: ,"Point",
  position: new google.maps.LatLng(latFloat, lngFloat);
});
```

- Spojování bodů do trasy:

```
Bing Maps:
var loc1 = new Microsoft.Maps.Location(20, 20);
var loc2 = new Microsoft.Maps.Location(40, 20);
var loc3 = new Microsoft.Maps.Location(60, 20)
var line = new Microsoft.Maps.Polyline(new Array(loc1, loc2, loc3));
map.entities.push(line);
```

```
Google Maps:
var latlngs = new google.maps.MVCArray();
latlngs.push(new google.maps.LatLng(10, 20));
latlngs.push(new google.maps.LatLng(20, 20));
latlngs.push(new google.maps.LatLng(30, 20));
var path = new google.maps.Polyline({
  map: map, path: latlngs
});
```

Jak je vidět z těchto ukázek kódů, obě API jsou téměř totožná, liší se víceméně jen názvem entit, pořadím parametrů a několika dalšími, převážně kosmetickými rozdíly. Z tohoto důvodu byla konečná volba, kterou službu využít, čistě subjektivní záležitostí.

3.7. Výběr služby

Po analýze služeb, které přicházely v úvahu, bylo třeba vybrat tu, na které bude nakonec aplikace postavena. V úvahu tedy přicházely dvě: Google Maps a Bing Maps. Obě nabízejí srovnatelné (navíc velmi podobné) komunikační rozhraní, zázemí silné společnosti i kvalitní dokumentaci.

Když se vezmou v úvahu zápory popsané u Bing Maps, vychází jako nejvhodnější kandidát služba Google Maps.

4. Architektura aplikace

4.1. Kód v prohlížeči

Všechny výše uvedené služby, Google Maps nevyjímaje, poskytují své API v jazyce JavaScript. To znamená, že o vykreslení mapy a veškerou práci s ní (přidávání bodů, výpočet azimutu, délky trasy atd.) se stará kód běžící u klienta (ve webovém prohlížeči uživatele naší aplikace). To je vhodné hned z několika důvodů. Jednak tím, že jakákoliv práce s mapou nezatěžuje server, kde aplikace běží – klient komunikuje a načítá mapové podklady přímo ze serverů Google, a pak faktem, že se většina kódu vykonává v jeho prohlížeči, což zaručuje vysokou interaktivitu aplikace bez zbytečných prodlev.

4.2. Serverová část

Existují operace, které se nemůžou provádět na klientské straně, ale musí se o ně postarat server. Jedná se například o autentizaci konkrétních uživatelů, ukládání jejich map a finální export.

4.2.1. Programovací jazyk

Na rozdíl od programovacího jazyka uživatelské části, kde bylo víceméně rozhodnuto poskytovatelem služby, na straně serveru jsem měl možnost volby. V úvahu přicházely všechny tradiční serverové jazyky: Java, PHP, C#, Python, atd.

Vzhledem k tomu, že podstatnou část této práce tvoří seznamování se s metodologií plánování VFR létání a prozkoumávání geografického aparátu, což je pro mě velká neznámá, rozhodl jsem se tuto skutečnost vyrovnat tím, že serverovou část napíši v jazyce, se kterým mám dlouholeté zkušenosti – a sice PHP. Dalším důvodem, proč zvolit právě tento jazyk, je jeho rozšířenost na různých webových hostinzích. Hosting pro PHP + MySQL lze nalézt mnohem snáze (a často i levněji) než např. pro hosting s podporou .NET + MSSQL databází.

PHP (aktuálně ve verzi 5.3) [12] je skriptovací, objektově orientovaný, dynamicky typovaný jazyk. Původně byl navržen pro tvorbu jednodušších webových aplikací, nicméně v současnosti je v něm napsáno velké množství projektů. Nejnovější verze tohoto jazyka přináší mimo několika oprav dvě zásadní novinky: podporu jmenných prostorů (které budou v práci použity) a anonymní funkce (ve velké míře známé například z jazyka JavaScript).

4.2.2. Databáze

Data uživatelů, jejich mapy a osobní nastavení je třeba uchovávat. Pro co nejjednodušší propojení s jazykem PHP a kvůli nejrozšířenější podpoře na hostingových serverech jsem použil databázi MySQL [13]. V novějších verzích podporuje vlastnosti známé z velkých podnikových databází jako například referenční integrita, uložené procedury, triggery a podobně.

4.2.3. Webový framework

Pro zjednodušení a pomoc s psaním rutinního kódu jako je např. práce se *sessions*, abstrakce a bezpečnější práce s databází, autentizace uživatelů, šablonovací systém a další (ať už PHP specifické nebo obecné) záležitosti, které musí programátor psát v každém projektu stále dokola, jsem se rozhodl nasadit framework, který mě od tohoto úskalí odstíní. Tím zbyde více času a prostoru na implementaci vlastní aplikace a ladění drobností, které dělají aplikaci více uživatelsky příjemnou a použitelnou.

Protože poslední dobou je v prostředí (hlavně českého) internetu stále více informací, článků, ale i nabídek k zaměstnání poptávající znalost Nette Frameworku [9], rozhodl jsem se ho tedy po krátkém zamyšlení použít jako kostru této aplikace. Splňuje prakticky všechny výše zmíněné aspekty, které jsou od moderního frameworku očekávány.

Nette je projekt s otevřeným zdrojovým kódem (tzv. open source) od známé osobnosti českého internetu Davida Grudla. Je celý psaný objektově a v nejnovější verzi využívá možnosti jazyka PHP 5.3. Celý kód je přehledně členěn do jmenných prostorů. Kód vychází z návrhového vzoru MVC a svojí strukturou nutí programátora, aby tyto principy také dodržoval.

V úvahách nad tím, který framework zvolit, jsem se také rozmýšlel nad použitím systému Drupal, se kterým mám díky mému předchozímu zaměstnání bohaté zkušenosti. Drupal, ač velice flexibilní a moderní aplikace, je spíše než framework redakční systém vhodný pro publikování článků, novinek a inzerátů (obecně se těmto entitám říká uzly). Sice by šlo nastavit vytváření nových map principiálně stejně jako publikace článků, ale Drupal již v základu nabízí velké množství funkcí a možností, přičemž většinu bych stejně nevyužil. Spíše naopak. Jeho filozofie procedurálního programování by mi v tomto případě znesnadňovala práci.

5. Implementace

5.1. Zobrazení mapy a vykreslování bodů

Jak již bylo zmíněno, veškerá práce s mapou, přidávání a úprava bodů trasy probíhá přes JavaScript API poskytované společností Google. Samotné vytvoření mapy potom vypadá takto:

```
var pilsen = new google.maps.LatLng(49.755098, 13.386841);
var mapOptions = {
  zoom: 10,
  center: pilsen,
  mapTypeId: google.maps.MapTypeId.SATELLITE
};

map = new google.maps.Map(document.getElementById('map_canvas'), mapOptions);
google.maps.event.addListener(map, 'click', function(event) {addPoint(event)})
```

Jak je vidět, API je velice jednoduché a poměrně snadno čitelné. Nejvíce času zabralo na začátku studování dokumentace a procházení ukázkových příkladů. Ve

vývojovém prostředí se mi bohužel nepovedlo zprovoznit našeptávání pro práci s JavaScriptovým kódem, který se týkal map, takže veškeré hledání metod a názvů jednotlivých parametrů znamenalo práci s dokumentací.

Na posledním řádku kódu je pak vidět přiřazení funkce jakožto listeneru k akci „kliknutí myší na plochu mapy“. Práce s listenery je velice podobná té v jazyce Java, pouze s tím rozdílem, že jako listener figuruje pouze samostatná funkce, na rozdíl od registrace celé třídy (v Javě).

Následující kus kódu demonstruje, jak probíhá samotné přidání nového bodu do mapy.

```
function addLocation(lat, lng, title) {
  var newLocation = new google.maps.LatLng(lat, lng);
  var marker = createLocationMarker(newLocation, title);
  markers.push(marker);
  latlngs.push(newLocation);
  displayPath.setPath(latlngs);
  recalculateMarkers();

  return marker;
}
```

5.2. Vzdálenost dvou bodů

Určit nejkratší vzdálenost mezi dvěma body na povrchu Země není úplně triviální záležitost.

Za prvé proto, že Země je kulatá, je nutné veškeré výpočty provádět ve sférických souřadnicích. Z toho vyplývá, že nejkratší vzdálenost dvou bodů na povrchu Země není přímka, ale tzv. ortodroma. Jedná se o křivku opisující povrch Země, jejíž střed splývá se středem Země. Její tvar je kružnice, respektive její část.

Bohužel, nevýhoda této křivky spočívá v tom, že azimut, který určuje směr letu, se při pohybu po této křivce mění. Pro zjednodušení se z tohoto důvodu častěji využívá loxodroma, která je sice delší (ovšem do vzdálenosti 1000 km je rozdíl naprosto zanedbatelný), ale má tu vlastnost, že všechny poledníky protíná pod stejným úhlem.

Za druhé, veškeré předchozí úvahy předpokládaly Zemi jako ideální kouli, což samozřejmě neplatí, a proto jsou výše uvedené postupy zatíženy chybou už od samého počátku.

5.2.1. Metoda haversinu

Metoda, která vypočte vzdálenost mezi dvěma body na kulové ploše, vychází z výpočtu ortodromy [10] [11] (algoritmus Great-Line Circle) a její vzorec vypadá následovně:

$$a = sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) * \cos(lat_2) * sin^2\left(\frac{\Delta lon}{2}\right)$$

$$vzdálenost = 2 * R * atan2(\sqrt{a}, \sqrt{1-a})$$

Kde:

- *lat* znamená zeměpisnou šířku, *lon* zeměpisnou délku.
- atan2 je funkce, která bere dva parametry: p1, p2 a z nich spočítá hodnotu funkce arkus tangens v bodě p1/p2. Navíc je speciálně definovaná pro případy, kdy se parametr p2 rovná nule.

5.3. Výpočet azimutu mezi dvěma body

Azimut je veličina hojně používaná (nejen) v letecké navigaci. Jedná se o orientovaný úhel, který svírá určitý směr k cílovému (pozorovanému) objektu od přímého směru na sever. Záleží u něj na směru měření – trochu netradičně (vzhledem k měření úhlů v matematice) se měří po směru hodinových ručiček. K měření se používá buzola a měří se ve stupních. V letectví se k popisu azimutu používá trojice cifer, které definují daný úhel (může nabývat hodnot od 0° – směr na sever, přes 180° – směr na jih, 270° – přímo na západ až po 360°), Přičemž nevýznamové nuly zleva se vždy píší. Konkrétní azimut tedy může vypadat třeba jako 055 nebo 258.

Jak již bylo zmíněno výše, při pohybu po ideální křivce (ortodromě) se azimut během letu mění. Změna tohoto azimutu mezi počátečním a koncovým bodem může být opravdu značná. Např. při trase z Bagdádu: 35°N, 45°E do Osaky: 35°N, 135°E bude počáteční azimut 060 a koncový azimut dokonce 090.

Vzorec pro azimut v počátečním bodě (někdy též označován jako dopředný azimut) vypadá následovně [10]:

$$y = sin(\Delta lon) * cos(lat2)$$
$$x = cos(lat1) * sin(lat2) - sin(lat1) * cos(lat2) * cos(\Delta lon)$$
$$azimut = atan2(y, x)$$

Většina letounů používaných při VFR letech je vybavena základními navigačními přístroji, které nejsou schopny tuto odchylku opravovat. Naopak větší letadla jsou vybavena vyspělými navigačními přístroji, které jsou schopné na základě informací ze satelitů GPS, integrovaných gyroskopických zařízení, apod. tuto nepřesnost korigovat a k odchýlení nedochází.

Naštěstí při VFR letech po České republice (let z jednoho konce republiky na druhý) může dojít k maximální odchylce cca 5 metrů. Proto je použití těchto metod pro účely této aplikace dostatečně přesné.

5.4. Export do PDF

V aplikaci je třeba dynamicky generovat PDF dokumenty obsahující navigační štítek. Pro tyto účely byla, po rychlém průzkumu, vybrána knihovna mPDF. Jedná se o objektově psanou knihovnu, díky které lze na základě několika příkazů jednoduše vygenerovat PDF dokument. Nepotřebuje ani žádné speciální rozšíření nebo moduly do PHP. Pouze je vyžadována verze PHP 4.3 a vyšší se zapnutým rozšířením *mb_string* (které je stejně obsaženo v téměř každé instalaci PHP na serveru). Navíc díky její licenci (GNU) není nejmenší problém jejího použití pro jakýkoliv účel. Knihovna si poradí jak s vykreslením základního dokumentu, tak i složitějších elementů. Dle dokumentace zvládá HTML tabulky, obrázky, odrážky a číslování, CSS pozicování. Velkou zajímavostí je potom schopnost vytvářet čárové nebo dokonce QR kódy. K tomu přidává automatické zalamování listů, tvorbu obsahu a další užitečné funkce (které bohužel v této prácí nenajdou využití). Její použití je velice jednoduché, jak je vidět na následujícím příkladě:

```
$mpdf = new mPDF( 'en-GB', 'A4', 9, 'dejavusans' );
$mpdf->debug = true;
$mpdf->WriteHTML( $html );
$mpdf->Output('map.pdf', "D");
```

5.5. Databázový model

Struktura a množství tabulek jsou vzhledem k rozsahu dat, která se budou ukládat, poměrně malá – celkem čítá dvě tabulky.

Tabulka *users* obsahuje definici uživatelů, jejich přihlašovací jméno, otisk hesla a jednoznačný identifikátor (číselný primární klíč).

Tabulka *maps* obsahuje definice jednotlivých map, které si uživatelé v aplikaci vytvořili. Mapa je identifikována číselným primárním klíčem. Zároveň obsahuje cizí klíč z tabulky *users* jakožto autora, který danou mapu vytvořil. Dále obsahuje pole: *created_time, updated time, title, markers* a *points.*

Zajímavá jsou pole *markers* a *points*. *Markers* obsahuje definici všech otočných bodů, které si uživatel uložil do mapy. *Points* jsou pomocné ATC body v dolní části navigačního štítku. Jedná se o textová pole obsahující JSON (JavaScript Object Notation), což je efektivní popis datových struktur, který se často využívá při přenosu dat (zde mezi databází a aplikací). Jedná se o obdobu popisovacího jazyka XML, která je ovšem úspornější co se velikosti výsledného dokumentu týče (v XML většinu zaberou samotné elementy). Dále neexistuje obdoba DTD nebo XSD, který by přesně popisoval strukturu vyměňovaných dat.

Dokument JSON obsahující otočné body jedné mapy potom vypadá takto:


```
[
  {
    "lat":"49.72296711730795",
    "lng":"13.45284479943848",
    "title":"LKPL",
    "distance":"0",
    "cumulativeDistance":"0",
    "azimuth":"068",
    "ident":"",
    "freq":"",
    "alt":""
  },
  {
    "lat":"49.858418517572154",
    "lng":"13.762114227691654",
    "title":"Zbiroh",
    "distance":"27",
    "cumulativeDistance":"27",
    "azimuth":"076",
    "ident":"",
    "freq":"",
    "alt":""
  },
  {
    "lat":"49.8907084815098",
    "lng":"13.887384117523197",
    "title":"Točník",
    "distance":"10",
    "cumulativeDistance":"37",
    "azimuth":"083",
    "ident":"",
    "freq":"",
    "alt":""
```

```
  },
{
    "lat":"49.93963468142287",
    "lng":"14.188167036453251",
    "title":"Karlštejn",
    "distance":"22",
    "cumulativeDistance":"59",
    "azimuth":"183",
    "ident":"",
    "freq":"",
    "alt":""
  }
]
```

Struktura dokumentu je následující: Jedná se o pole objektů (pole je ohraničeno závorkami [ ], jednotlivé elementy jsou v něm odděleny čárkou). Každý element představuje jeden otočný bod na mapě. První tři atributy jsou mapové informace o tomto bodě (GPS pozice a uživatelský název bodu). Další představují vypočtené údaje o daném bodě (přenáší se při exportu do tabulky). Poslední tři atributy představují dodatečné údaje, které si uživatel aplikace může vyplnit v navigačním štítku. Jedná se např. o maximální/minimální letovou výšku.

Největší výhoda použití tohoto formátu v aplikaci je ta, že tato pole je třeba po načtení z databáze vykreslit ve stránce jako JavaScriptové pole (a při zobrazování mapy postupně přidat všechny otočné body tak, jakoby je vytvářel uživatel). Takto upravený obsah pole markers stačí vzít a přímo vypsat do šablony. Pokud by se body ukládaly ve speciální tabulce s vazbou 1:N s tabulkou maps, znamenalo by to další dotaz do databáze a následné složité skládání výsledného dotazu do JavaScriptového pole.

6. Instalace aplikace

K zprovoznění této aplikace na vlastním stroji je zapotřebí mít nainstalováno několik programů:

- Webový server Apache (alespoň verze 2.2)
- Nakonfigurovaná podpora PHP skriptů (alespoň PHP 5.3)
- Databáze MySQL (verze 5 a vyšší)
- Pokud jsou vyžadovány tzv. hezké URL, je zapotřebí povolit Apache rozšíření *mod_rewrite*

Nejprve je třeba nakopírovat všechny soubory aplikace do kořenového adresáře webového serveru (případně do kořenového adresáře vytvořeného virtuálního stroje virtual host). Jako další přichází na řadu založení nové databáze na MySQL serveru. Poté je třeba nad touto databází spustit přiložený skript *create.sql*, který se postará o vytvoření všech potřebných tabulek pro aplikaci. Pro tuto akci budou potřeba databázová oprávnění CREATE/ALTER TABLE. Pro samotný chod aplikace potom stačí oprávnění CRUD (create/read/update/delete).

Dále je třeba aplikaci říct, kde se databázový server nachází. V souboru /app/config.neon je v sekci common/services/database třeba správě vyplnit adresu stroje, kde se databáze nachází, její název (jedná se o stejný název, který byl vytvořen v minulém kroku) a nakonec jméno a heslo.

Správnost výše popsaných kroků lze ověřit jednoduše zadáním adresy, kde má aplikace běžet, do prohlížeče. Pokud se objeví úvodní stránka aplikace, je vše v pořádku. V opačném případě se objeví chybová hláška s detailním popisem chyby. Častým problémem bývá nemožnost aplikace zapisovat do logovacích a dočasných (cache) adresářů. Řešením je změnit přístupová práva na adresářích */log* a */temp* pro zápis skupiny, do které patří Apache uživatel (případně nastavit Apache jako vlastníka tohoto adresáře). Na Unixovém stroji k tomu poslouží příkazy *chmod* a *chown*.

7. Ovládání

Aplikace nabízí potřebné funkce pro vytváření, ukládání, prohlížení stávajících a export naplánovaných mapových tras. Všechny tyto položky jsou dostupné z hlavního menu, které se nachází v horní části obrazovky. Aplikace je schopná rozlišovat jednotlivé uživatele na základě jimi vytvořených profilů. Uživatel se standardně (jako u většiny aplikací) identifikuje jménem a heslem. Odkazy na založení nového účtu nebo přihlášení lze nalézt opět v hlavním menu.

7.1. Registrace a přihlášení uživatele

Registrovaní a přihlášení uživatelé mají v aplikaci několik výhod. Např. mohou vidět jimi vytvořené mapy pohromadě na jedné stránce a pak mohou tyto mapy dodatečně editovat. K zaregistrování nového uživatele nebo přihlášení do aplikace slouží odkaz v hlavním menu „Přihlásit/Registrovat“. Stránka pak obsahuje standardní formulář se jménem a heslem. O úspěšném vytvoření účtu nebo přihlášení je uživatel informován potvrzující zprávou v horní části obrazovky.

7.2. Vytvoření nové mapy

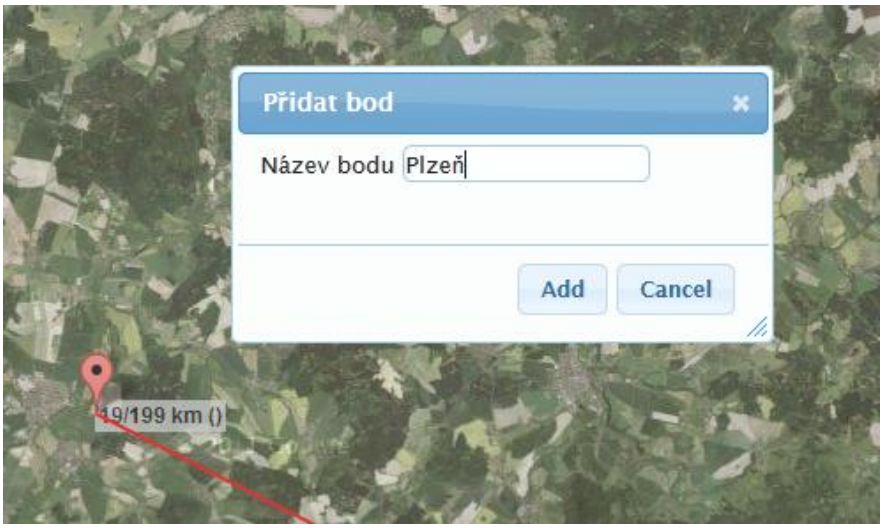
Všechny důležité ovládací prvky, které se používají při tvorbě nové i úpravě stávající mapy, se nachází v plovoucím panelu, který se standardně zobrazuje u pravého okraje stránky. Tento panel vypadá následovně:

Obrázek 3

7.2.1. Přidávání bodů

Vytvořit novou mapu lze jako přihlášený i nepřihlášený uživatel. Pokud má ovšem uživatel v plánu editovat mapu, kterou dříve vytvořil, musí být přihlášen. K tvorbě nové mapy slouží odkaz v menu „Nová mapa“. Na této stránce se nachází plátno s prázdnou mapou a panel s ovládacími prvky.

Nové body lze přidávat jednoduše klikáním na požadované místo, kam má být otočný bod umístěn. Po kliknutí na mapu se objeví dialogové okno se vstupním polem, kam uživatel zadá název nově vytvářeného bodu. Potvrzením OK (nebo stisknutí klávesy Enter) je bod přidán. Tuto proceduru opakuje uživatel tak dlouho, dokud chce přidávat další body. Dialog pro přidání nového bodu vypadá následovně:



Obrázek 4

Druhou možností, jak přidat nový bod, je přes ovládací panel. Nachází se zde textové pole, kam uživatel zadá GPS souřadnici a kliknutím na tlačítko „Přidat na“ je bod vložen přesně na zadanou souřadnici. Pokud by se uživatel chtěl pouze podívat na cílové místo (bez přidání bodu), lze kliknout na tlačítko „Jít na“. Aplikace umí automaticky uzavřít křivku (spojí aktuálně poslední bod s počátečním bodem). Toho se dosáhne kliknutím na tlačítko „Uzavřít křivku“.

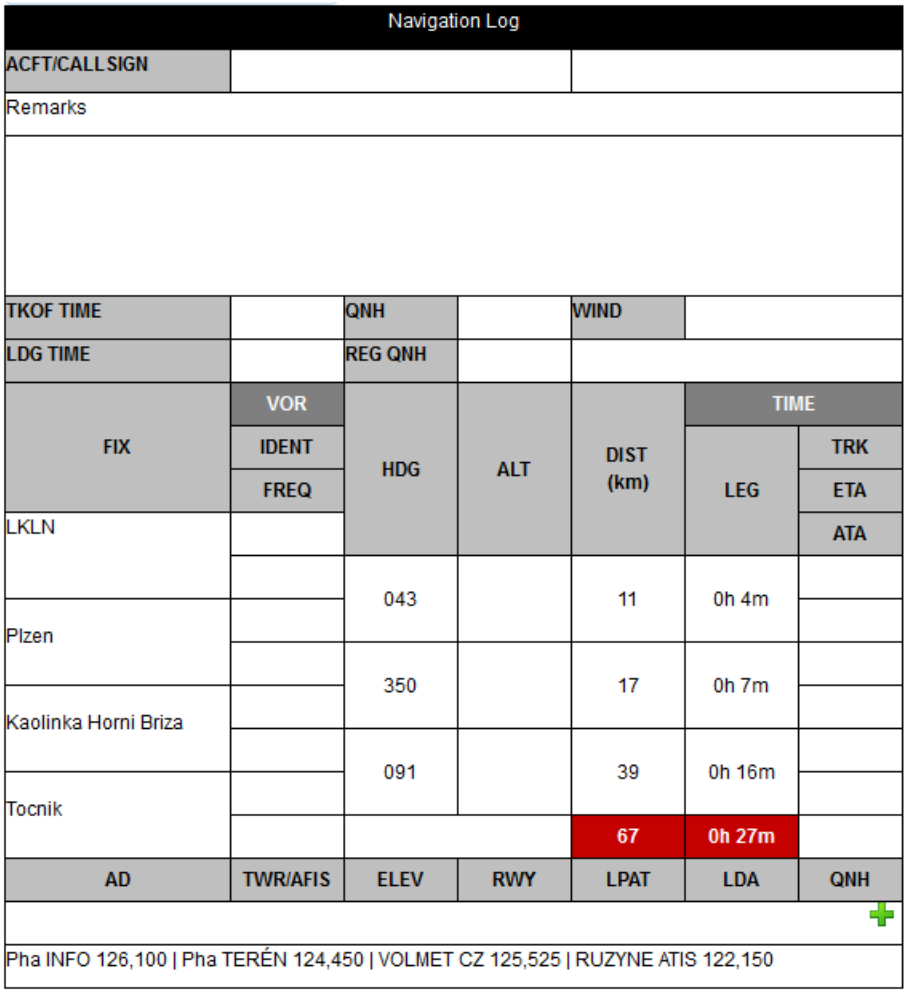
Všechny již vytvořené body lze v mapě dodatečně přemisťovat. Stačí kliknout myší na nějaký bod a držet tlačítko. Tažením se potom bod přesouvá, přičemž se okamžitě dynamicky přepočítávají sousední body (azimuty a vzdálenosti). Po uvolnění tlačítka myši se bod umístí na novou pozici. Obyčejným kliknutím na libovolný bod lze potom vyvolat podrobné informace o daném místě (včetně přesné pozice GPS).

Aplikace také umí počítat přibližnou dobu plánovaného letu. K tomu musí znát odhadovanou rychlost letadla. Tu může pilot zadat v kolonce „Průměrná rychlost“. K tomu se váže nastavení jednotek rychlosti. Aplikace umí pracovat jak v metrické, tak imperiální soustavě. Přepínání mezi těmito jednotkami se děje také na ovládacím panelu, hned nad polem pro zadávání průměrné rychlosti. Veškeré vzdálenostní/rychlostní údaje jsou potom zobrazovány v odpovídajících jednotkách.

7.2.2. Vyplnění navigačního štítku

Poté, co uživatel zadal všechny plánované otočné body, se lze přepnout do zobrazení navigačního štítku. Toho se dosáhne kliknutím na stejnojmenné tlačítko v ovládacím panelu. Zobrazí se (zatím pouze náhled), jak bude výsledný navigační

štítek vypadat. Jsou předvyplněna pole o otočných bodech, azimuty a vzdálenosti. Ostatní pole, jako např. maximální výška nebo údaje o pomocných leteckých radiostanicích, jsou prázdná a kliknutím na ně myší (editovatelnému poli se po najetí myši změní kurzor) se buňka změní na textové pole, kde lze dovyplnit patřičné údaje, a zmáčknutím tlačítka Enter potvrdit. Text se potom objeví v buňce a promítne se i do exportovaného štítku. Náhled takového štítku vypadá následovně:



Navigation Log						
ACFT/CALLSIGN						
Remarks						
TKOF TIME		QNH		WIND		
LDG TIME		REG QNH				
FIX	VOR	HDG	ALT	DIST (km)	TIME	
	IDENT				LEG	TRK
	FREQ					ETA
LKLN						ATA
		043		11	0h 4m	
Plzen						
		350		17	0h 7m	
Kaolinka Horni Briza						
		091		39	0h 16m	
Tocnik						
				67	0h 27m	
AD	TWR/AFIS	ELEV	RWY	LPAT	LDA	QNH
Pha INFO 126,100 | Pha TERÉN 124,450 | VOLMET CZ 125,525 | RUZYNE ATIS 122,150						

Obrázek 5

V průběhu vyplňování štítku se lze přepnout zpět do mapového pohledu stisknutím tlačítka „Zpět k mapě“, kde lze provést jakékoliv dodatečné úpravy, přidat/odebrat libovolné body a po úpravách se vrátit zpět do náhledu štítku. Tlačítkem „Navigační štítek“ se lze poté opět přepnout do editace štítku.

7.3. Uložení mapy

Uživatel má možnost si mapu uložit pro pozdější (opakovaný) export (nebo také proto, aby ji mohl sdílet s přáteli). Uložením se rozumí jak zachování všech přidaných bodů v mapě, tak i informace, které si uživatel vyplnil v navigačním štítku – maximální nebo minimální výška, frekvence radionavigačních prostředků, atd. Mapa se uloží tlačítkem „Uložit mapu“. Aplikace se zeptá na jméno, pod jakým bude mapa dostupná, a poté ji uloží. Následně se mapa zobrazuje na stránce „Všechny mapy“. Pokud byl uživatel přihlášen, zobrazí se mu i na stránce s jeho mapami (odkaz „Moje mapy“).

Nepřihlášený uživatel má také možnost uložení mapy. Ta bude viditelná mezi ostatními mapami a kdokoliv si ji bude moct zobrazit, ale nikdo nebude mít možnost ji upravit.

Každý uživatel má při prohlížení jakékoliv (i cizí) mapy možnost vytvořit novou kopii této mapy a tu si uložit pod svým účtem (princip „Uložit jako“, známý z drtivé většiny kancelářských aplikací). Uživatel tedy může cizí mapu poupravit, přidat/odebrat nějaké body, upravit informace v navigačním štítku a uložit si vlastní kopii této upravené mapy. Tímto lze obejít nemožnost editace map od nepřihlášených uživatelů.

7.4. Export mapy

Poté, co má uživatel vytvořenou mapu se správnými body a vyplněný navigační štítek, lze provést export. Aby bylo možné mapu exportovat, musí být zobrazen navigační štítek (ne samotná mapa). V tomto zobrazení je aktivní tlačítko „Export do PDF“. Kliknutím na něj se začne na serveru generovat PDF soubor a následně je uživateli nabídnut ke stažení.

7.5. Procházení existujících map

Vytvořené mapy od všech uživatelů lze nalézt na stránce „Všechny mapy“, kam se lze dostat odkazem z hlavního menu. Pokud je uživatel přihlášen, vidí odkaz „Moje mapy“, kde jsou zobrazeny jen mapy vytvořené právě jím.

8. Závěr

V této práci jsem prozkoumal jednak metodiku plánování letů za viditelnosti (VFR) a s tím související matematické výpočty nutné k správné orientaci ve vzduchu, a jednak analyzoval veřejně dostupné geografické služby.

S těmito získanými znalostmi jsem naprogramoval aplikaci, která je použitelná pro piloty menších letadel na plánování jejich VFR letů. Pilot si jednoduše klikáním myši vytvoří požadovanou trasu, aplikace mu vypočte nezbytné údaje potřebné pro nadcházející let a na závěr vygeneruje tzv. navigační štítek, který si s sebou pilot bere do letadla.

S Google Maps API se pracovalo dobře, na internetu bylo k dispozici velké množství tutoriálů a řešených příkladů, díky kterým bylo řešení všech problémů velice jednoduché. Znalost tohoto API může do budoucna představovat velkou výhodu.

Na druhou stranu mě velice zklamal Nette Framework. Jednak pan Grudl má (nebo alespoň měl, v době psaní této aplikace) velký zmatek ve verzování – tehdy byla stabilní verze 2.0-beta. Už zde je vidět první rozpor – stabilní beta (jako beta se obyčejně označují verze v testovacím stadiu). V rámci této verze bylo vydáno několik revizí (naneštěstí všechny se stejným označením), které obsahovaly zásadní změny v API Nette, v syntaxi konfiguračního souboru a další, které ovšem nikde nebyly zdokumentované. Jediným místem, kde hledat pomoc, byly přímo zdrojové kódy nebo fórum. Oficiální dokumentace je obecně dost chaotická, část se vztahuje na předchozí verzi (s označením 0.9), část na novou 2.0. Ani komentáře v kódu na tom nejsou o moc lépe. Chybějící PHPdoc, který pomáhá vývojovému prostředí (IDE) v našeptávání návratového typu funkce nebo popis parametrů, velice znesnadňuje práci. Věřím, že pokud bych se opravdu ponořil do nitra Nette a pracoval s ním delší dobu, určitě by toto nebyl problém. Bohužel to není ta situace, proč programátor nasazuje externí framework. Kvůli těmto důvodům se pokusím Nette Frameworku do budoucna co nejvíce vyhnout.

Seznam zdrojů

[1] *Letecká informační služba: Předpisy* [online]. [2012] [cit. 2012-02-18]. Dostupné z: http://lis.rlp.cz/predpisy/predpisy/index.htm

[2] *IVAO Czech Division* [online]. [2012] [cit. 2012-02-18]. Dostupné z: http://www.ivao.cz/

[3] *AisView* [online]. [2012] [cit. 2012-03-14]. Dostupné z: http://aisview.rlp.cz/

[4] *OpenStreetMap* [online]. [2012] [cit. 2012-02-25]. Dostupné z: http://www.openstreetmap.org/

[5] *Microsoft.Maps API Reference* [online]. [2012] [cit. 2012-02-25]. Dostupné z: http://msdn.microsoft.com/en-us/library/gg427611.aspx

[6] *Dokumentace API Mapy.cz* [online]. [2012] [cit. 2012-02-25]. Dostupné z: http://api.mapy.cz/static?doc=index

[7] *Google Maps Javascript API V3 Reference* [online]. [2012] [cit. 2012-02-26]. Dostupné z: https://developers.google.com/maps/documentation/javascript/reference

[8] *Google Earth API Developer's Guide* [online]. [2012] [cit. 2012-02-26]. Dostupné z: https://developers.google.com/earth/documentation/index

[9] *Nette Framework: Dokumentace* [online]. [2012] [cit. 2012-03-14]. Dostupné z: http://doc.nette.org/cs/

[10] VENESS, Chris. *Movable Type Scripts* [online]. [2012] [cit. 2012-01-14]. Dostupné z: http://www.movable-type.co.uk/scripts/latlong.html

[11] *Haversine Formula* [online]. [2012] [cit. 2012-01-14]. Dostupné z: http://en.wikipedia.org/wiki/Haversine_formula

[12] DOYLE, Matt. *Beginning PHP 5.3*. Indianapolis, IN: Wiley Pub., c2010, 801 s. Wrox beginning guides. ISBN 04-704-1396-4.

[13] *MySQL profesionálně: optimalizace pro vysoký výkon*. Vyd. 1. Brno: Zoner Press, 2009, 712 s. Encyklopedie webdesignera. ISBN 978-80-7413-035-9.