VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

KALENDÁŘNÍ APLIKACE S EFEKTIVNÍM UŽIVATELSKÝM ROZHRANÍM

CALENDAR APPLICATION WITH USER FRIENDLY INTERFACE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **EVA NAVRÁTILOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Dr. Ing. PETR PERINGER**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19465/2016/xnavra54

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Navrátilová Eva**
Obor: Informační technologie
Téma: **Kalendářní aplikace s efektivním uživatelským rozhraním**
Calendar Application With User Friendly Interface
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s problematikou aplikací typu kalendář. Prostudujte uživatelské rozhraní a funkčnost podobných aplikací. Zaměřte se na kalendář pro systémy firmy Palm a zdokumentujte konkrétní rozhraní použité v zařízení *Tungsten E*.
2. Navrhněte kalendářní aplikaci s kvalitním uživatelským rozhraním inspirovaným kalendářem na systémech Palm. Aplikace musí umožňovat sdílení záznamů s ostatními kalendářními programy.
3. Aplikaci implementujte v C++ s použitím Qt5. Proveďte testování aplikace na vhodně zvolených příkladech.
4. Zhodnoťte dosažené výsledky a navrhněte možné směry dalšího vývoje aplikace.

Literatura:

- Dokumentace na WWW stránkách projektu Qt (http://doc.qt.io/).
- Další dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních dvou bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Peringer Petr, Dr. Ing.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav inteligentních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato práce se zabývá tvorbou kalendářní aplikace s efektivním a uživatelsky přívětivým rozhraním. Nejprve je provedena analýza existujících kalendářních aplikací na různých platformách. Je prozkoumán standard ukládání kalendářních dat a dobré praktiky pro návrh uživatelského rozhraní. Dále je popsán návrh kalendářní aplikace, který zahrnuje návrh kalendářní knihovny, vzhledu grafického rozhraní a struktury rozhraní. Výsledná aplikace je implementována v jazyce C++ za využití frameworku Qt a podrobena testům.

Abstract

This thesis describes the process of creating a calendar application featuring an effective and user friendly interface. The first step is an analysis of existing calendar applications on various platforms. Then the calendar data standard is explored, together with good practices for user interface design. The analysis is followed by designing the calendar application. The design process consists of calendar library design, graphical interface design and structural interface design. The resulting application, implemented in C++ using Qt framework, is tested.

Klíčová slova

Kalendář, uživatelské rozhraní, objektové programování, C++, Qt, přenositelnost

Keywords

Calendar, user interface, object-oriented programming, C++, Qt, portability

Citace

NAVRÁTILOVÁ, Eva. *Kalendářní aplikace s efektivním uživatelským rozhraním.* Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Peringer Petr.

Kalendářní aplikace s efektivním uživatelským rozhraním

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Dr. Ing. Petra Peringera. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Eva Navrátilová
15. května 2017

Poděkování

Děkuji vedoucímu své práce, panu Dr. Ing. Petru Peringerovi, za cenné rady při konzultacích a za zapůjčení literatury.

Obsah

Kapitola 1

Úvod

Lidstvo odjakživa pociťovalo potřebu členit čas a uspořádávat ho do uchopitelných celků. Vznikaly tak různé kalendáře vyhovující různým civilizacím. Kalendářem se řídilo a dodnes řídí zemědělství, jsou zde zaznamenány náboženské svátky a stojí na něm hospodářský systém. Kalendářní systém také umožňuje jednotlivcům udržovat si přehled o svých plánovaných akcích. S rozvojem techniky začal přirozeně vzrůstat i zájem o aplikace poskytující funkci kalendáře.

Cílem této práce je navrhnout takovou kalendářní aplikaci, která bude uživateli umožňovat evidenci a zobrazování jeho událostí, a to prostřednictvím uživatelsky přívětivého rozhraní jehož obsluha bude rychlá a nenáročná. Aplikace bude také přenositelná, což uživatelům umožní využívat aplikaci nezávisle na typu jejich zařízení.

První část práce (Kapitola 2) popisuje standard pro sdílení kalendářních dat. Dále se věnuje průzkumu kalendářové aplikace Calendar na zařízení *Tungsten E* od firmy Palm, která je hlavním vzorem pro výsledné uživatelské rozhraní kalendáře. Jsou zde podrobně zdokumentovány jednotlivé obrazovky kalendáře a popsána jeho funkcionalita. Prozkoumány jsou i další často používané kalendářní aplikace. Kapitola je uzavřena souhrnem dobrých zásad pro programování uživatelských rozhraní.

Druhá část práce (Kapitola 3) už se věnuje návrhu uživatelského rozhraní kalendářní aplikace, který spojuje všechny klady implementace kalendáře na zařízení *Tungsten E* s přínosnými prvky ostatních prozkoumaných implementací a zároveň ctí zásady dobrého návrhu uživatelských rozhraní prozkoumané v sekci 2.4.

Třetí část práce (Kapitola 4) popisuje již samotnou implementaci kalendářní aplikace a jejího uživatelského rozhraní v jazyce C++ za použití frameworku Qt.

Čtvrtá část (Kapitola 4) pak popisuje průběh testování aplikace na zvolených příkladech.

V závěru práce (Kapitola 5) je výsledná aplikace zhodnocena. Jsou navrženy možné směry dalšího vývoje aplikace.

Kapitola 2

Analýza existujících aplikací a standardů

Tato kapitola se zabývá průzkumy provedenými za účelem zmapování problematiky kalendářních aplikací. Základním předpokladem pro porozumění této problematice je analýza aktuálního standardu pro ukládání kalendářních dat, standardu *iCalendar*, který je využíván většinou dnešních aplikací kalendářního typu.

V první řadě je ovšem rozebráno uživatelské rozhraní zařízení *Tungsten E* od firmy Palm implementující mimo jiné aplikaci Calendar, která je dobrým vzorem kalendáře s uživatelsky přívětivým rozhraním. Kromě avizované aplikace jsou zde rozebrány i jiné kalendářní programy, a to především se zaměřením na jejich uživatelská rozhraní. Kapitola je uzavřena zevrubnou analýzou dobrých a špatných praktik používaných při návrhu uživatelského rozhraní.

2.1 Standard *iCalendar*

Standard *iCalendar* slouží k výměně kalendářních dat mezi zařízeními. Umožňuje tak celé řadě produktů od různých výrobců kompatibilní způsob sdílení dat. Soubory formátu *iCalendar* mají obvykle přípony „.ical“, „.ics“, „.ifb“ nebo „.icalendar“.

iCalendar byl poprvé jakožto standard definován v roce 1998 jako RFC 2445 [8] společností Internet Engineering Task Force (IETF). Jeho autorem byl Frank Dawson ze společnosti Lotus Notes (dnes vlastněna IBM) a Deryk Stenerson ze společnosti Microsoft. Spolupráce zaměstnanců konkurenčních firem podtrhla záměr vytvořit všeobecně využitelný standard. Tato myšlenka byla zachována dodnes rozšiřováním podpory pro nové technologie jako jsou například online webové aplikace nebo aplikace pro chytré telefony a tablety.

Definice standardu *iCalendar* byla upravena v roce 2009 prostřednictvím RFC 5545 [4]. Bernard Desruisseaux ze společnosti Oracle odstranil ze standardu nejednoznačnosti a zredukoval jej o již nepotřebné funkcionality. RFC 5545 je dnes považováno za *iCalendar* standard a nahradilo původní RFC. V roce 2016 bylo vydáno RFC 7986, které rozšiřuje *iCalendar* RFC například o podporu konferenčních systémů. *iCalendar* je dnes využíván jako prostředek k impotrování a synchronizaci kalendářních událostí na rozličných platformách, mezi něž patří i chytré telefony a webové aplikace.

Jednotlivé datové objekty jsou na základě standardu *iCalendar* ohraničeny počátečním klíčovým slovem `BEGIN` a koncovým slovem `END`. Samotný *iCalendar* objekt je pak označen klíčovým slovem `VCALENDAR`. Uvnitř tohoto objektu již mohou být definovány uživatelské

události, označované jako `VEVENT` a upozornění na ně uložená pod klíčovým slovem `VALARM`. V rámci těl událostí a jejich upozornění pak mohou být definovány další doplňující informace jako například název události, datum konání události či relativní čas spuštění upozornění.

```
BEGIN:VCALENDAR
VERSION:2.0
PRODID:-//hacksw/handcal//NONSGML v1.0//EN
BEGIN:VEVENT
UID:19970610T172345Z-AF23B2@example.com
DTSTAMP:19970610T172345Z
DTSTART:19970714T170000Z
DTEND:19970715T040000Z
SUMMARY:Bastille Day Party
END:VEVENT
END:VCALENDAR
```

V úseku nad tímto odstavcem je příklad záznamu standardu Ze záznamu výše tak například vyčteme, že událost s názvem „Bastille Day Party“ začala 14. července 1997 v 17 hodin a skončila o den později ve 4 hodiny.

2.2 Aplikace Calendar na zařízení *Tungsten E*

Série zařízení *Tungsten* byla představena v říjnu roku 2002. Jedná se o zařízení typu PDA – osobní digitální pomocník (personal digital assistant).

Zařízení *Tungsten E* patří k nejprodávanějším zařízením této úspěšné řady. Hardwarové jádro je tvořeno 126 MHz procesorem Texas Instruments OMAP311[9] a 32 MB nevolatilní flash paměti. Zařízení je vybaveno dotykovou TFT obrazovkou o rozměrech 54 x 54 mm s rozlišením 320 x 320 pixelů a barevnou hloubkou 16 bitů. Ta umožňuje vynikající čitelnost jak v interiéru tak na přímém slunci.

Obrázek 2.1: Zařízení *Tungsten E* od firmy Palm

Text zařízení přijímá prostřednictvím dedikované Graffiti plošky, kam uživatel načrtává jednotlivá písmena, která se v zařízení převádí do digitální podoby.

Zařízení je dále vybaveno operačním systémem Palm OS 5.2.1 [13] a několika základními aplikacemi. K těm patří například Contacts, Notepad, Memos, Calendar nebo Tasks. Tato sestava ale nemusí být nikoli finální, množství softwaru lze rozšířit o aplikace nainstalované na SD kartě.

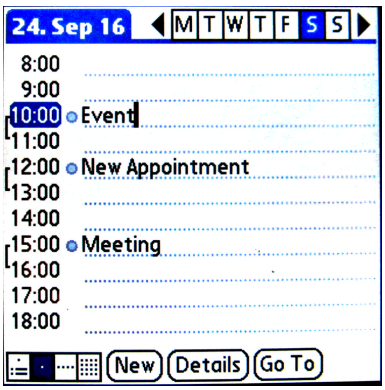
Specialitou tohoto konkrétního zařízení byla právě vylepšená aplikace Tasks, určená přímo ke zobrazování agendy. Ta byla uzpůsobena pro rychlé zobrazení nadcházejících úkolů a událostí a tvořila tak jakousi zkratku do aplikace Calendar.

Pro účely této práce je klíčové prozkoumat na zařízení *Tungsten E* právě aplikaci Calendar. Strukturovaná analýza její funkcionality a uživatelského rozhraní je základní prerekvizitou a inspirací pro návrh efektivního rozhraní kalendářní aplikace. Rozhraní i aplikace samotná byly za tímto účelem zevrubně zdokumentovány.

Přehled dne

Tato obrazovka se vybírá prostřednictvím ikony s tečkou uprostřed, znázorňující jeden den. Po jejím výběru se zobrazí přehled konkrétního dne, a to implicitně od 8 do 18 hodin. Je zde možné prohlížet, editovat a přidávat aktivity.

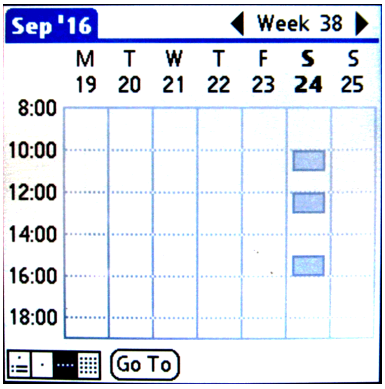
Přidávání aktivit je velmi efektivní a uživatelsky přívětivé. Provádí se prostřednictvím kliknutí na čas, kdy má událost začínat, a vepsáním jejího názvu. Délka události je pak automaticky nastavena na jednu hodinu. Editace události probíhá obdobně. Uživatel vybere událost k editaci a přímo na aktuální obrazovce je mu umožněno měnit její název. Pokročilou editaci umožňuje tlačítko Details (viz podsekci 2.2)



Obrázek 2.2: Přehled dne s naplánovanými událostmi

Přehled týdne

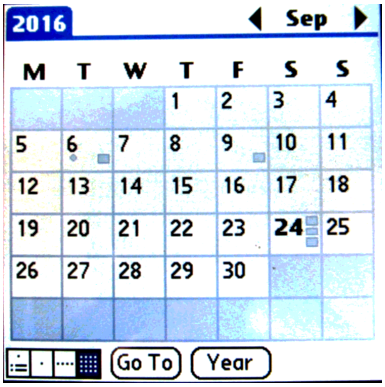
Toto zobrazení se vybírá prostřednictvím ikony s několika vodorovnými tečkami graficky vyjadřujícími týden. Uživateli umožňuje prohlížení a plánování událostí ve větším časovém intervalu. Uživatel si udržuje přehled nad naplánovanými aktivitami a jejich posouváním může jednoduše upravovat čas jejich konání. I toto zobrazení navíc umožňuje rychlé vkládání a úpravy aktivit, podobně jako zobrazení předchozí.



Obrázek 2.3: Přehled týdne s naplánovanými událostmi

Měsíční přehled

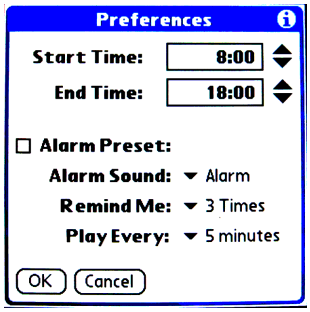
Na tuto obrazovku se uživatel dostane po vybrání ikony s tečkami vykreslenými do matice, které ilustrují měsíc. Uživateli se zde naskýtá možnost prohlížet aktivity v ještě rozsáhlejším časovém intervalu. Jsou-li na určitý den naplánovány nějaké aktivity, zobrazí se vedle jeho čísla chronologicky seřazený odpovídající počet obdélníčků. Aktuální den je zvýrazněn tučným písmem.



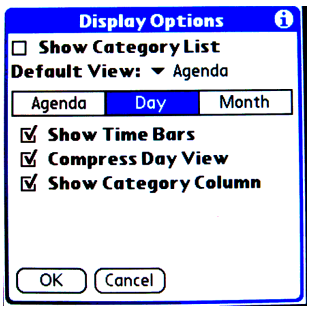
Obrázek 2.4: Měsíční přehled

Obecná nastavení aplikace

Pro zpříjemnění uživatelské zkušenosti lze aplikaci přizpůsobit potřebám uživatele prostřednictvím obecných nastavení. Zde lze nastavit například zobrazovaný časový interval případně detaily týkající se zobrazování aplikace.



(a) Nastavení preferencí

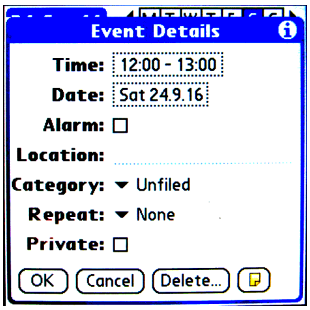


(b) Nastavení zobrazení

Obrázek 2.5: Obrazovky nastavení aplikace

Pokročilá nastavení aktivity

Po vytvoření události lze upravit její detaily prostřednictvím pokročilých nastavení. Ta umožňují například zapnutí budíku pro událost, nastavení místa konání, přidání události do určité kategorie či případné nastavení opakování události. I zde je samozřejmě možné upravit datum a čas konání události.



Obrázek 2.6: Pokročilá nastavení aktivity

Shrnutí průzkumu aplikace Calendar

Aplikace Calendar vyniká především svým uživatelským rozhraním, které je velice efektivní a intuitivní. Tohoto stavu bylo dosaženo především omezením nastavitelných parametrů akcí, což přispělo k lepší přehlednosti a nevznikla tak potřeba složitých nastavovacích dialogů.

Výsledkem takového návrhu je tak kompaktní aplikace, která je z hlediska každodenního používání velice přívětivá. K vytváření události se člověk často propracuje jediným kliknutím. Velkou část informací lze k události nastavit už ve výchozím zobrazení, okno s pokročilými nastaveními je tedy potřeba otevřít pouze zřídka. Uživatelské rozhraní je zde tedy skutečně prostředníkem, nikoliv překážkou, mezi uživatelem a funkcí aplikace, kterou chce využít.

Velkým nedostatkem této aplikace je ovšem její zastaralost. Aplikace již není kompatibilní s dnešními mobilními zařízeními ani se současným standardem ukládání kalendářních dat.

2.3 Další kalendářní aplikace

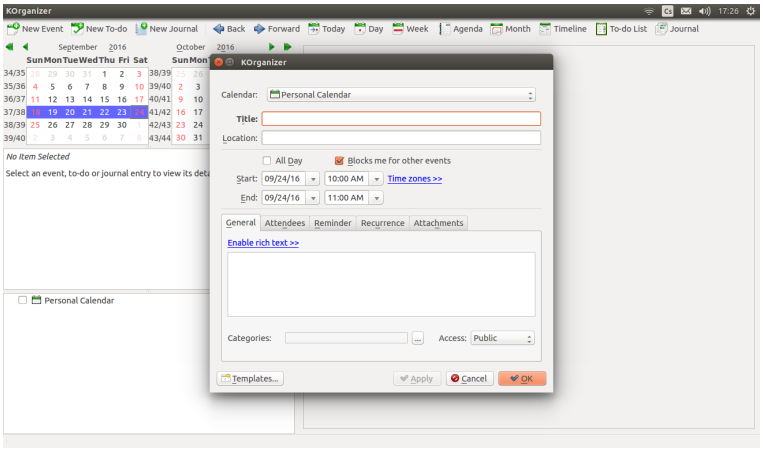
Možnost plánování událostí je dnes považována takřka za základní funkcionalitu, kterou by mělo poskytovat každé univerzální elektronické zařízení. Není tedy divu, že je na trhu celá řada kalendářních aplikací přizpůsobených na míru všemožným zařízením i operačním systémům. Ojedinělejší jsou však již aplikace vytvořené na míru pro uživatele.

Tato sekce se zabývá vybranou množinou všeobecně rozšířených kalendářních aplikací a to především jejich rozborem s důrazem především na jejich uživatelské rozhraní.

2.3.1 KOrganizer

KOrganizer [5] je volně dostupná aplikace pro zaznamenávání událostí do kalendáře. Je dostupná pro všechny platformy využívající hlavní distribuce operačních systémů GNU/Linux i pro operační systémy Windows. Patří mezi komponenty soupravy Kontact [6], která uživateli zprostředkovává management osobních informací a zahrnuje i aplikace pro správu e-mailů, vytváření adresářů a další. Není tedy žádným překvapením, že je KOrganizer velice komplexní aplikace.

Při vytváření či úpravách události umožňuje KOrganizer kromě základních nastavení i celou řadu velmi pokročilých upřesnění, například nastavení účastníků události, přidávání příloh či aplikaci šablon. Tuto variabilitu jistě ocení náročnější uživatelé, ale jako vše i ona má svou stinnou stránku. Kvůli velkému množství nastavitelných možností vzrostla členitost nejednoho dialogového okna. Přestože většina uživatelů těchto nadstandardních funkcionalit nejspíše nikdy nevyužije, nevyhnou se při každodenním vytváření i editaci jednoduchých událostí tomuto složitému prostředníkovi. Používání takto komplexní aplikace daleko překračující jejich požadavky může negativně ovlivnit jejich spokojenost s aplikací.



Obrázek 2.7: Vytváření nové události v aplikaci KOrganizer

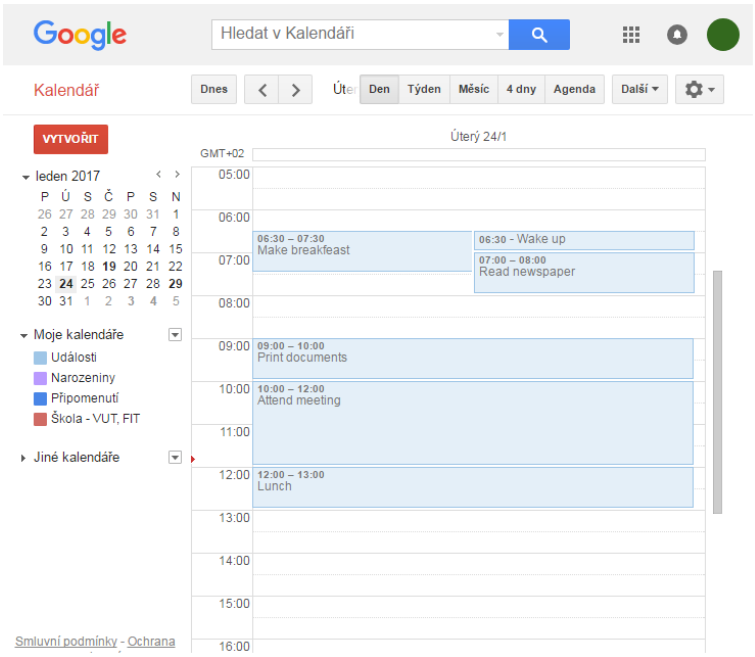
2.3.2 Kalendář Google

Kalendář od firmy Google je podobně jako KOrganizer (podsekce 2.3.1) součástí celé sítě aplikací pro správu osobních dat uživatele. Je k dispozici jako webová aplikace, ale existují i jeho verze pro chytrá mobilní zařízení. Tato analýza se zabývá konkrétně webovou aplikací.

Obrovskou výhodou tohoto kalendáře je možnost vytvářet události jediným kliknutím. Vytvoří se tak událost s přednastavenou standardní délkou, jejíž název je možné vyplnit okamžitě bez nutnosti využití komplexnějšího okna. To je zde ovšem stále k dispozici pro

pokročilejší uživatele. Toto považuji za ideální přizpůsobení složité aplikace široké škále uživatelů. Online správa událostí navíc zajišťuje automatickou synchronizaci stavu kalendáře na všech zařízeních uživatele. Tato funkčnost je jistě velice užitečná pro uživatele, kteří chtějí využívat Google kalendář i na svém chytrém telefonu.

Zdálo by se, že Google kalendář je ideální volbou, má ovšem také svůj handicap. Ten tkví právě ve webovém rozhraní jako takovém. K využívání tohoto kalendáře je totiž potřeba mít připojení k internetu. To může být překážkou pro uživatele, kteří jsou zvyklí se svým počítačem cestovat a postrádají tak jistotu neustálého připojení. Bez něj totiž nemohou události na počítači ani vytvářet, ani zobrazovat.



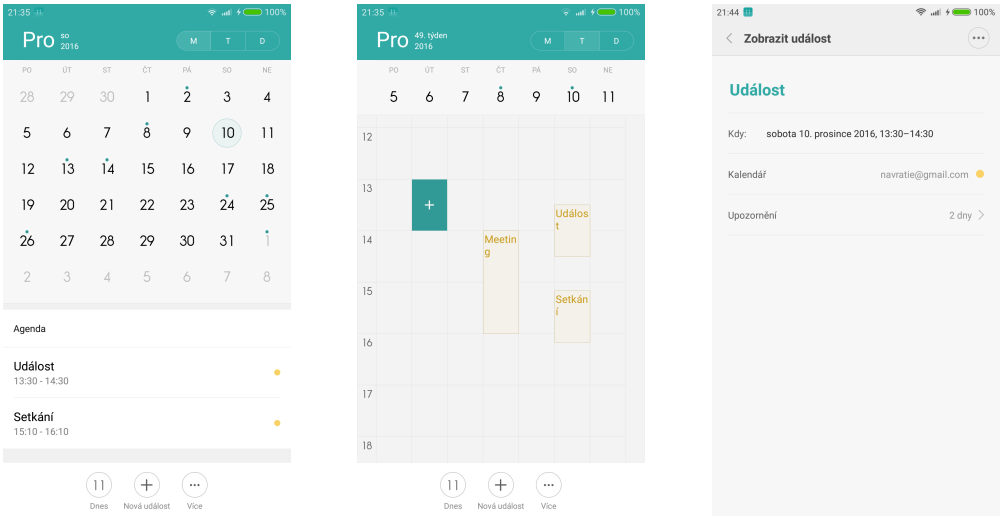
Obrázek 2.8: Webové rozhraní kalendáře Google

2.3.3 Kalendář na platformách MIUI

MIUI je uživatelské rozhraní pro chytré telefony od firmy Xiaomi [1]. Je založeno na Androidu 2.3 a 4.0 a nabízí uživatelům velké množství uzpůsobitelných prvků. Také s sebou nese několik základních aplikací, mezi něž patří například i kalendář, který je zde uveden jako zástupce kalendářního rozhraní pro mobilní platformu. Jeho analýzou se bude zabývat tato kapitola.

Rozhraní této aplikace nabízí moderní a úsporné zobrazení nadcházejících aktivit, které je poměrně vhodně přizpůsobené cílové platformě. Umožňuje také sdílení dat s kalendářem Google (podsekce 2.3.2) a vzájemnou synchronizaci, což je pro mobilní zařízení klíčová funkce.

Rozhraní je strohé, ale moderní; neobsahuje rušivé prvky. Zobrazování událostí je v rámci možností dané platformy přehledné. Hůře je na tom však vytváření a editace událostí. Úspěšnému vytvoření události předchází příliš mnoho úkonů. Otevření editace je zdlouhavé a uživatelsky málo přívětivé. Při zobrazení detailu události je pak nevhodně distribuován prostor; nepříliš důležité informace pak vystupují výrazněji než ty významné.



(a) Zobrazení měsíce (b) Zobrazení týdne (c) Zobrazení události

Obrázek 2.9: Přehledová obrazovka kalendáře na platformách MIUI

2.4 Zásady pro tvorbu uživatelských rozhraní

Uživatelské rozhraní předurčuje, jak bude aplikace vnímána a přijata okolním světem. Z pohledu uživatele je rozhraní dokonce synonymem aplikace jako celku. Je tedy vhodné věnovat jeho návrhu značné úsilí. Aby bylo uživatelské rozhraní úspěšné, je vhodné řídit se několika pravidly, shrnujícími základní zásady návrhu uživatelského rozhraní.

Konzistence
: Určitý objekt by měl v rámci jedné aplikace mít vždy stejnou (konzistentní) funkci či sémantiku.

Viditelnost
: Uživatel by měl být vizuálně informován o všech možnostech, které mu rozhraní nabízí.

Zpětná vazba
: Uživatel by měl dostávat zpětnou vazbu o tom, že jeho akce byly programem zaznamenány.

Jednoduchost
: Rozhraní aplikace by mělo být právě tak jednoduché, jak to aplikace umožňuje.

Tolerance vůči chybám
: Rozhraní by mělo implementovat ochranu uživatele před jeho vlastními chybami.

Přístupnost
: Aplikace by měla být přizpůsobena osobám trpícím zrakovými, pohybovými či jinými zdravotními potížemi.

Standardy a směrnice
: Při návrhu rozhraní je vhodné držet se ověřených směrnic a standardů, na něž jsou již uživatelé zvyklí.

V praxi se tato pravidla projevují širokým spektrem principů, na něž je vhodné při návrhu uživatelských rozhraní dbát. Prvním a zásadním krokem ovšem je zpracování návrhu uživatelského rozhraní ještě před započetím implementace [12]. Pokud by toto nebylo dodrženo, mohl by být programátorův náhled na vzhled rozhraní negativně ovlivněn jeho znalostí implementace.

Obecná pravidla pro návrh rozhraní

Prvním z řady těchto pravidel je *výběr vhodné terminologie*. Terminologie reprezentuje aplikaci jako takovou, je tedy vhodné její výběr nezanedbat. Terminologie by měla být poplatná modelovanému procesu, nikoliv použitému modelu. Dalším důležitým aspektem je pak samotný výběr použitých slov. Ta by měla odpovídat oboru zaměření aplikace a měla by být uzpůsobena tak, aby jim uživatelé dobře rozuměli.

Další pravidlo už se již týká vizuálního vzezření aplikace. Apeluje přitom na *rozumné využívání barev*. Barva je jednoduchý prostředek pro zvýraznění objektů. Nesmíme ovšem zapomínat, že až 10% mužské populace trpí nějakou formou barvosleposti, a vybrané barvy tak pro ně nemusí být dostatečně kontrastní. Barva je ovšem také emocionální záležitostí, a tak se může stát, že barva bude ne vždy interpretována tak, jak bylo původně zamýšleno. Neposledním problémem je také nadužívání barev. Ideální cestou tedy je využívat barev střídmě a výmluvně a podávat tutéž informaci i jinou cestou.

Dobrým krokem při vytváření uživatelského rozhraní je také *podpora ovládání aplikace myší i klávesnicí*. Mnozí uživatelé totiž považují jedno z těchto zařízení za jednodušeji ovladatelné. Je tedy vhodné zpřístupnit všechny možnosti aplikace jak pro uživatele myši, tak pro uživatele klávesnice, a to například nastavením klávesových zkratek.

Užitečná je i zásada, že *pamatovat by si měla aplikace, ne uživatel*. Týká se to například nastavení, nedávno použitých položek či pozicování a velikosti okna. S tímto pravidlem souvisí i doporučení zavést *tlačítko „Zpět“* pro zrušení poslední provedené akce.

Pravidla pro nabídky menu

Důležitou součástí rozhraní každého programu je i *struktura menu aplikace*. To by mělo především odpovídat zavedeným standardům. Nemělo by být členěno na příliš mnoho kaskád a každá položka by se v celém menu měla vyskytovat přesně jedenkrát, jinak se uživatel bude zbytečně snažit porozumět neexistujícímu rozdílu mezi dvěma totožnými položkami na různých místech. Menu by mělo být především logicky uspořádáno a mělo by co nejpřesněji korespondovat s prací uživatele, pro niž je aplikace určena.

Některé položky v nabídce menu se chovají jinak než ostatní. Zatímco jedny pouze provedou akci, jiné otevřou dialog. Je vhodné uživatele *předem upozornit na chování jednotlivých položek*. Pokud například volba otevírá dialogové okno a vyžaduje tak další interakci s uživatelem, je vhodné doplnit její název třemi tečkami.

Důležitou součástí většiny aplikačních rozhraní jsou *panely nástrojů*, takzvané toolbary. Zde je vhodné uživatelům poskytnout všechny nejčastěji používané položky. Ty se následně doporučuje rozdělit to souvisejících celků prostřednictvím rozdělovačů. V panelech nástrojů také přicházejí ke slovu obrázkové ikony. Ty tvoří z velké části vizuální stránku aplikace. I u ikon je ovšem potřeba zachovávat konzistenci sémantiky.

Pravidla pro formuláře

Formuláře jsou v případě mnoha aplikací místem, kde uživatelé stráví nejvíce času. Při jejich návrhu je tedy na místě zvážit jak často a za jakých podmínek budou využívány a přizpůsobit tomu jejich vzhled a chování.

Jelikož každý formulář slouží ke konkrétnímu účelu, měla by být jeho *funkce zmíněna v titulku*. Kromě funkce je někdy vhodné do titulku zahrnout například i název souboru, k němuž se akce dialogu vztahuje.

Důležitým předpokladem pro přehlednost rozhraní je *vizuální odlišení dialogů od hlavního okna* aplikace. Tohoto lze docílit například nastavením modality pro dialogy. Naopak hlavní okno nesmí být nikdy modální.

Používá-li aplikace dialogy, je vhodné řídit se zásadou *nepoužívat dialogů příliš mnoho*. Velice jednoduše totiž může dojít k tomu, že se z dialogů stane rušivý element. Nedoporučuje se tedy požadovat po uživateli dialogové potvrzování neinvazivních akcí, informovat ho o provedeném procesu není-li to nezbytně nutné, nebo ho informovat o chybách s nimiž se dokáže program bez problémů vypořádat sám.

Jedním z dilemat při návrhu rozhraní je výběr vhodného způsobu *ukládání dat*. Tím je míněna otázka, zda by měl data uživatel ukládat manuálně, či zda by to měla být automatická akce samotné aplikace. Na základě vybraného schématu je pak nutné přizpůsobit i chování aplikace.

Aplikace také musí řešit problém *nalezení a zobrazení dat*, jež chce uživatel zhlédnout. Druhá úloha je většinou vyřešena otevřením dialogového okna s kýženými daty. Co se týče nalézání, pro jednodušší zprostředkování množiny dat k prohledání byl vyvinut mechanismus mřížky, který uspořádává data do logických oddělených celku a uživateli tak zjednodušuje orientaci ve zobrazovaných datech.

Pravidla pro ovládací prvky

Naprosto zásadní je pravidlo využít pro danou úlohu *vhodný ovládací prvek*. Většinou se zde jedná o řešení dilematu zda využít set přepínačů, vysouvací nabídku nebo zaškrtávací políčka. Při řešení tohoto dilematu se dá řídit následujícími zásadami. Set přepínačů je vhodný pro neměnné sady vzájemně se vylučujících voleb. Vysouvací nabídka se naopak hodí pro případy, kdy se počet možností k výběru dynamicky mění. Zaškrtávací políčko se pak využije v případech, kdy se volí mezi možnostmi ano/ne.

Jelikož není vhodné uživateli po výběru akce zahlásit, že akce momentálně není dostupná, musí existovat jiná cesta jak toto zprostředkovat. Ta se řídí pravidlem *povolit uživateli přístup jen k těm akcím, které může provést*. Ikony či titulky ostatních akcí je vhodné zobrazit jako neaktivní.

Další zásadní pravidlo se týká nastavení. Aplikace by měla uživateli umožňovat *nastavit standardní hodnoty* pro často prováděné akce. Toto pravidlo navazuje na zásadu zapamatovávání si uživatelových voleb.

Jistě není žádným překvapením, že *související ovládací prvky by měly být seskupeny*. V nejlepším případě by měla být každá skupina pojmenovaná a ohraničená boxem. Uživatel se pak bude dobře orientovat i ve složitějších částech programu.

Shrnutí zásad pro návrh uživatelských rozhraní

Vytvoření uživatelsky přívětivého rozhraní nestojí na pouhé znalosti všech doporučovaných zásad, ale na jejich aplikaci. Pro navržení skutečně dobrého rozhraní je nutné pochopit, jak

lidé vidí, chápou, a přemýšlejí [10]. Je nutné prozkoumat cílovou skupinu uživatelů stejně jako porozumět oboru, který má aplikace zaštiťovat. Teprve potom má smysl pouštět se do samotného návrhu rozhraní.

Kapitola 3

Návrh kalendářní aplikace

Cílem této práce je navrhnout a vytvořit kalendářní aplikaci. Tato kapitola se bude zabývat právě jejím návrhem. Bude zde popsán návrh vizuální podoby aplikace, návrh kalendářní knihovny a v neposlední řadě i návrh struktury uživatelského rozhraní.

3.1 Specifikace požadavků

Před započetím jakéhokoliv návrhu je vhodné zaměřit se na požadavky kladené na produkt. Těmto požadavkům by měl být celý proces návrhu podřízen. V tomto případě je kladen důraz především na efektivitu kalendářového rozhraní. To by mělo uživateli umožňovat svižnou a intuitivní interakci s aplikací. Doporučena je inspirace rozhraním kalendáře na zařízeních od firmy Palm osvěžená o přínosné prvky z jiných úspěšných kalendářních aplikací.

Výsledná aplikace by měla umožňovat zobrazování a správu uživatelových událostí a upozornění na ně. Dále by měla poskytovat možnost sdílení dat s ostatními kalendářními programy, a to prostřednictvím souborů podléhajících standardu *iCalendar*.

Aplikace měla být vytvořena v jazyce C++11 za využití frameworku Qt5, což by mělo vést k přenositelnosti výsledné aplikace. Primárně by však aplikace měla být uzpůsobena pro linuxové operační systémy.

3.2 Grafický návrh uživatelského rozhraní

Na základě analýzy ze sekce 2.4 je nutné začít právě návrhem uživatelského rozhraní, které v této aplikaci hraje hlavní roli. Mým cílem bylo dosáhnout toho, aby rozhraní umožňovalo uživateli především svižné a intuitivní ovládání aplikace. Tomuto cíli pak byl podřízen celý návrh aplikace.

Vytvoření nové události by mělo být ideálně výsledkem jednoho uživatelova kliknutí, stejně tak její mazání. Úpravy záznamů by měly být přehledné a data předvyplněna na základě uživatelem definovaných výchozích hodnot. Zobrazení kalendáře by mělo být přehledné a celistvé.

Na základě těchto elementárních požadavků jsem v několika fázích tvořila návrh rozhraní. Rozhodla jsem se pro jednoduchý a čistý vzhled. Jako barevné téma aplikace jsem zvolila výchozí šedou a bílou, které jsem doplnila pastelovou modrou barvou, jejímž účelem je především optické zvýraznění důležitých prvků. Výběr modré barvy rozhodně nebyl

náhodný. Modrá barva totiž navozuje relaxaci, mír, veselost a komfort [15]. Má také uklidňující účinek. Modrá barva v uživatelském rozhraní podle výzkumů koncentruje uživatele na jeho úkol.

Inspirace při návrhu tohoto rozhraní byla čerpána především z již existujících aplikací popsaných v sekcích 2.2 a 2.3. Výsledné grafické uživatelské rozhraní spojuje přednosti těchto aplikací do jednoho celku.

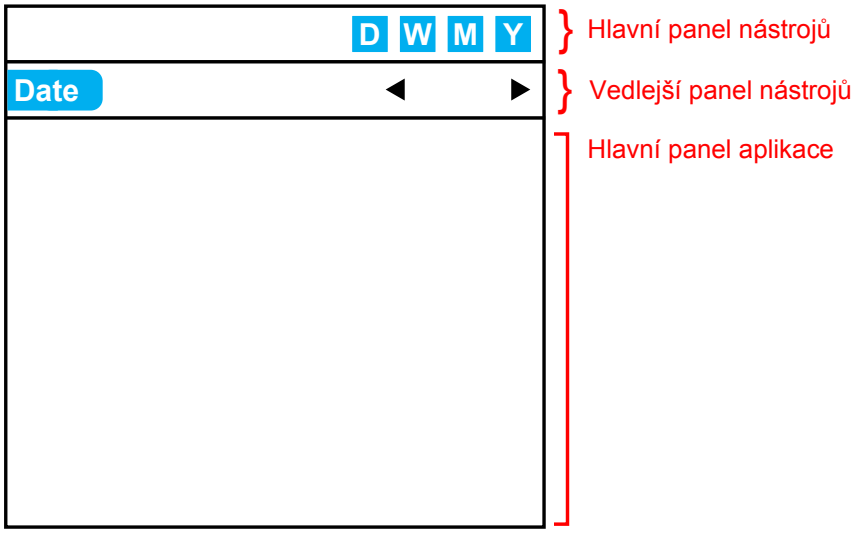
3.2.1 Obecný grafický návrh

Výsledná podoba návrhu úvodní obrazovky kalendáře je demonstrována na obrázku 3.1. Snažila jsem se navrhnout co nejefektivnější uživatelské rozhraní zvýrazňující důležité prvky. Vysoký důraz jsem při návrhu kladla na zachování konzistence rozhraní. Bylo totiž zřejmé, že základní obrazovka aplikace bude poskytovat čtyři různá zobrazení; dne, týdne, měsíce a roku.

Z obrázku je patrné, že navržené rozhraní se dělí na tři hlavní vertikálně navazující celky. Prvním z nich je hlavní panel nástrojů, obsazující horizontální pruh v horní části okna. Ten obsahuje ikony s písmeny *D*, *W*, *M* a *Y*, která odkazují na anglické obdoby výrazů den, týden, měsíc a rok. Ikony jsou podbarvovány modrou barvou, a to na základě toho, které zobrazení je právě vybrané. Přepínání mezi jednotlivými ikonami hlavního panelu nástrojů upravuje podobu vedlejšího panelu nástrojů a mění informace zobrazované v hlavním panelu aplikace.

Další prvek, vedlejší panel nástrojů, je tvořen horizontálním pruhem pod hlavním panelem. Tento vedlejší panel nástrojů slouží uživateli především k upřesnění informací ohledně vybraného zobrazení a pro pohyb v jeho rámci. Na levé straně panelu je ve většině zobrazení modrý rámeček upřesňující informace ohledně aktuálně vybraného časového intervalu. Na základě vybraného zobrazení je zde vypsáno celé datum, nebo třeba jen aktuálně prohlížený rok. Na pravé straně jsou pak šipkové přepínače sloužící k posunu v čase v rámci vybraného zobrazení. Mezi přepínači jsou pak doplňující informace k aktuálně vybranému času, např. den v týdnu či název měsíce, a to opět v závislosti na vybraném zobrazení.

Hlavní panel aplikace slouží ke zobrazení hlavního obsahu. Jeho rozložení se kompletně mění na základě volby provedené v hlavním panelu nástrojů. V hlavním panelu aplikace jsou uživateli vizuálně zprostředkována kalendářní data či přehledně zobrazeny měsíce pro efektivní orientaci a posun v čase. Jedná se o prostor, kde bude uživatel pracovat většinu času. Zde může vytvářet nové události a prohlížet, upravovat či mazat ty stávající.



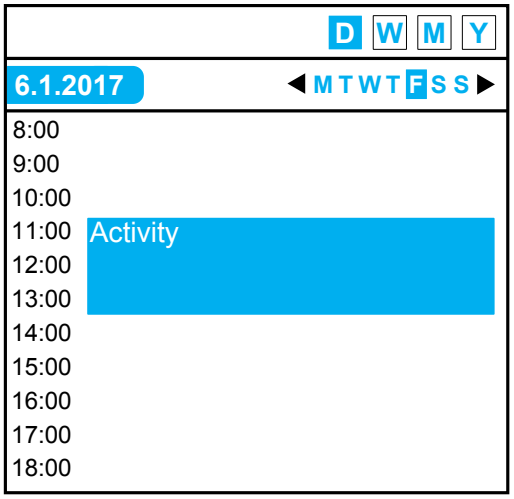
Obrázek 3.1: Obecný návrh rozhraní kalendáře

3.2.2 Zobrazení dne

Zobrazení dne je předpokládaná nejpoužívanější obrazovka této aplikace. Slouží k vykreslení přehledu událostí naplánovaných na konkrétní den. Kromě prohlížení umožňuje i manipulaci s těmito událostmi. Návrh této obrazovky je znázorněn na obrázku 3.2. V hlavním panelu nástrojů, který se za běhu aplikace téměř nemění, je v tomto případě vybrána ikona s písmenem D označující den. Pro zvýraznění volby je podbarvena modře.

Vedlejší panel nástrojů zprostředkovává v tomto zobrazení mnoho informací. Na jeho levé straně se uživateli v modrém zaobleném rámečku zobrazuje kompletní datum aktuálně prohlíženého dne. Napravo jsou pak k dispozici přepínače mezi jednotlivými dny. Mezi přepínači se nachází řada sedmi klikacích ikon reprezentujících jednotlivé dny v týdnu. Ikona aktuálního dne v týdnu je podbarvena modře.

V hlavním panelu aplikace se pro tento případ, zobrazení dne, se vykreslí vertikální časová osa rozdělená na hodiny. Pokud pro vybrané datum existují nějaké události, jsou zobrazeny v příslušném časovém okně jako modré obdélníky obsahující příslušný název. Při kliknutí na konkrétní čas se vytvoří nová událost začínající ve vybranou hodinu. Trvání události se v takovém případě automaticky nastaví na základě uživatelem stanovené výchozí hodnoty. Při pohybu myší nad jednotlivými událostmi se v plovoucí nápovědě zobrazují doplňující informace. Kliknutím do prostoru obdélníku může uživatel název události editovat. Pro pokročilejší editaci události stačí dvakrát kliknout do jejího prostoru, načež se otevře editační okno. Smazaní události je pak z hlediska uživatele stejně časově efektivní jako její vytvoření. Pokud se kurzor myši ocitne v prostoru události, zobrazí se v jejím pravém horním rohu ikona koše. Při kliknutí na ni je pak událost odstraněna.



Obrázek 3.2: Návrh rozhraní kalendáře: Zobrazení dne

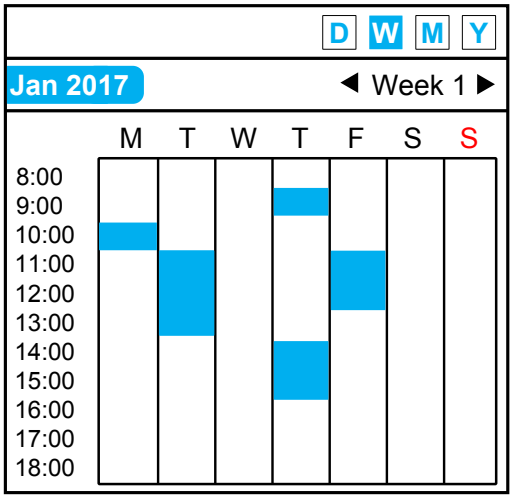
3.2.3 Zobrazení týdne

Rozhraní jsem se snažila udržet co nejkonzistentnější. Zobrazení týdne se tedy od zobrazení dne příliš neliší. Jeho vzhled je prezentován v obrázku 3.3. Jeho hlavní funkcí je umožnit uživateli přehled nad jeho harmonogramem v rámci delšího časového úseku než je jeden den.

Hlavní panel nástrojů zůstává zachován beze změn. Jedinou změnou je zde zvýraznění ikony s písmenem W symbolizující týden.

Ve vedlejším panelu nástrojů už je patrných více změn. V modrém rámečku zmizelo konkrétní datum, zobrazen je pouze aktuální měsíc a rok. Na pravé straně je pak místo přepínání mezi dny k dispozici přepínání mezi jednotlivými týdny. Mezi přepínači je zobrazeno číslo aktuálního týdne.

Nakolik oba panely nástrojů zůstaly bez výraznějších změn, hlavní panel aplikace zcela upravil svoji podobu tak, aby vyhovoval nárokům na zobrazení přehledu týdne. Vertikální časová osa složená z hodin zůstala zachována, přibyla k ní ovšem ještě horizontální osa, která zprostředkovává informace o konkrétních datech a dnech v týdnu. Samozřejmě nechybí náhled událostí v podobě modrých obdélníků, tentokrát je ovšem z prostorových důvodů ochuzen o vepsaný název. Ostatní funkce, jako jsou například editace či mazání, už nejsou nijak omezeny.



Obrázek 3.3: Návrh rozhraní kalendáře: Zobrazení týdne

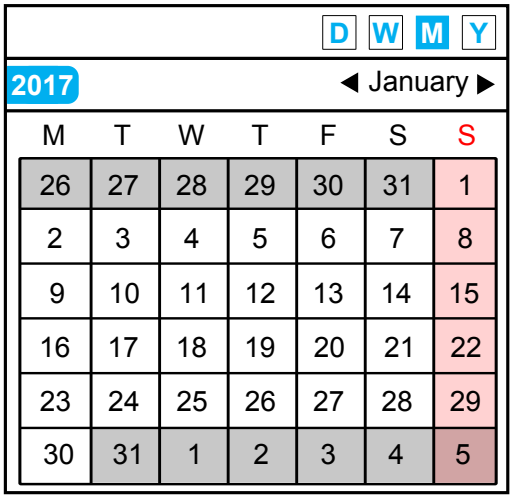
3.2.4 Zobrazení měsíce

Kromě dne a týdne bylo samozřejmě navrženo i zobrazení měsíce. Tento návrh je k nahlédnutí na obrázku 3.4. Uživateli může sloužit například jako rychlý prostředek k posunu o více dní v rámci jednoho či více měsíců.

Stejně jako v obou předešlých zobrazeních zůstává i zde v rámci podpory konzistence rozhraní hlavní panel nástrojů beze změny; opět je pouze upraveno zvýraznění. Modře je podbarvena ikona s písmenem M, odkazující na měsíc.

Vedlejší panel nástrojů se opět změnil jen symbolicky. V modrém rámečku nalevo je uveden již jen aktuální rok. Nalevo jsou vykresleny šipky pro přepínání mezi jednotlivými měsíci a mezi nimi je vepsán název aktuálního měsíce.

Hlavní panel aplikace tentokrát obsahuje tradiční zobrazení kalendářního měsíce. To umožňuje uživateli interaktivní práci; při pohnutí kolečkem myši se měsíce přepínají i bez nutnosti použití vedlejšího ovládacího panelu. Při poklepání na konkrétní den se aplikace automaticky přepne do zobrazení aktuálního dne.



Obrázek 3.4: Návrh rozhraní kalendáře: Zobrazení měsíce

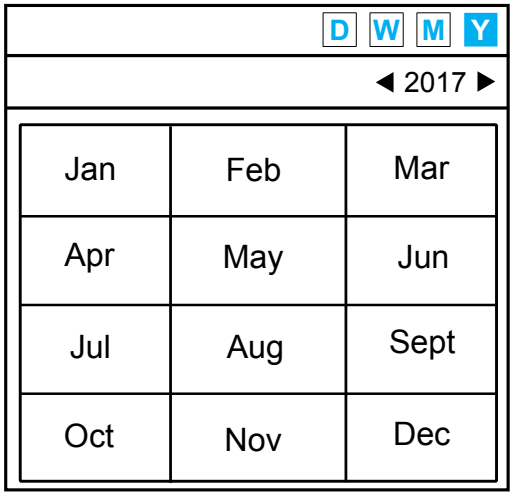
3.2.5 Zobrazení roku

Posledním z hlavních zobrazení, které bylo pro aplikaci navrhnuto, je zobrazení roku. To je v mnohém podobné zobrazení měsíce. Jeho primární funkcí je zprostředkovat uživateli rychlý grafický přehled nad vybraným rokem a jednotlivými měsíci. Může také sloužit jako přehledná podoba funkce „Přejít na datum“. Návrh je k nahlédnutí na obrázku 3.5.

Hlavní panel nástrojů nepřekvapivě zůstává beze změn, opět se jen upravuje zvýrazněná ikona. Tentokrát je modře podbarveno písmeno Y, symbolizující rok.

Vedlejší panel nástrojů změny opět prodělal. Ve zobrazení roku už zde chybí modrý rámeček se specifikovaným datem. Na levé straně však opět funguje přepínání, tentokrát mezi roky. Mezi přepínači je vepsán aktuálně zobrazovaný rok.

Hlavní panel je tvořen mřížkou o třech, případně čtyřech sloupcích v závislosti na aktuální šířce okna aplikace. Po řádcích jsou zde vykresleny náhledy jednotlivých měsíců. Po poklepání na konkrétní den je uživatel opět aplikací přesměrován do jeho zobrazení.



Obrázek 3.5: Návrh rozhraní kalendáře: Zobrazení roku

Shrnutí grafického návrhu

Výsledná vzhledová podoba aplikace byla v mnohých směrech inspirována kalendářní aplikací na zařízení *Tungsten E*, popsané v sekci 2.2. Vedlejší panel nástrojů byl převzat téměř beze změn, hlavní panel nástrojů byl inspirován ikonami v levé dolní části původní aplikace. I vzhled obsahu hlavního panelu aplikace byl v určitých směrech inspirován vzorovou aplikací, její vliv zde už ovšem není tolik patrný.

Další inspirací se pro propracovanost svého rozhraní stal i kalendář od firmy Google, popsaný v podsekci 2.3.2. Podle jeho vzoru jsou události vykreslovány jako barevné obdélníky a řadí se do mřížky podle časů konání. Stejně tak existuje pro každou kategorii událostí různá barva, kterou se obdélník události příslušné kategorie vykresluje.

Kromě inspirace z již existujících řešení se na vzhledu rozhraní podepsaly i zbrusu nové prvky. Mezi ně patří například mazání prostřednictvím tlačítka přímo v těle události. Odstranění události je tak jednodušší a uživatelsky přívětivější.

3.3 Kalendářní knihovna

Vhodně navrhnutá kalendářní knihovna je nezbytným prvním krokem na cestě za funkční kalendářní aplikací. Jejím prostřednictvím probíhají veškeré operace s kalendářními daty. Slouží také jako evidence naplánovaných událostí a upomínek a je tak jádrem celé aplikace. Diagram tříd kalendářní knihovny a jejich vzájemné provázání je k nahlédnutí v příloze B jako obrázek B.3.

Knihovna *icalendarlib*

Při návrhu kalendářní knihovny bylo nezbytné zajistit kompatibilitu se standardem *iCalendar*, popsaným v sekci 2.1. Dle specifikace požadavků v sekci 3.1 bylo totiž nutné umožnit aplikaci přijímat a ukládat data v tomto formátu.

Rozhodla jsem se tedy za tímto účelem využít knihovnu *icalendarlib* od Julia Gonery [11]. Ta zvládá jak načítání událostí ze souboru, tak i export evidovaných událostí. Kromě toho dokáže pracovat i s upomínkami a zajišťuje tak kompletní pokrytí této potřeby. Další důležitou vlastností, kterou jsem od knihovny vyžadovala, byla její multiplatformnost. I toto kritérium knihovna splňuje.

Tato knihovna se také v mnohých směrech stala inspirací pro vlastní kalendářní knihovnu. To je nejvíce patrné ve struktuře proměnných, která se ve vzájemně si odpovídajících třídách příliš nemění. Velkou inspiraci jsem odtud získala například i při vytváření struktury pro zaznamenávání pravidla opakování pro opakující se události, popsané v podsekci 3.3. Využila jsem také výčtový typ `TimeUnit` definovaný v této knihovně.

Aby tato knihovna co nejlépe splňovala kladené požadavky, bylo v ní ovšem třeba provést několik změn, například přidání metod. Také v ní byla odhalena chyba, která musela být opravena.

Úvahy o rozšíření této existující knihovny namísto vytváření vlastní knihovny však byly zamítnuty na základě nevhodně navržené struktury pro ukládání data a času. Z toho plynula další velká nevýhoda knihovny. To byla z tohoto pohledu i skutečnost, že by bylo nutné provést v ní mnoho změn za účelem jejího přizpůsobení nové funkci. Nakonec jsem se tedy rozhodla tuto knihovnu použít pouze jako prostředníka mezi vlastní kalendářní knihovnou a formáty definovanými standardem *iCalendar*.

Třída `DateTime`

Třída `DateTime` je první ze čtyř tříd zajišťujících správnou funkci kalendářní knihovny. Jak již vyplývá z jejího názvu, slouží pro evidenci data a času a pro operace s nimi. Je tak zároveň i pilířem všech ostatních tříd a aplikace jako takové. Při jejím návrhu jsem se řídila hlavně tím, aby byla co nejkompatibilnější se vstupním a výstupním formátem dat, tedy se standardem *iCalendar*.

Hlavním dilematem při návrhu této třídy se stalo rozhodnutí, zda využít šesti instančních atributů, jednoho pro každou časovou jednotku, nebo jediného atributu, v němž bude uložen počet sekund, který uplynul od 1.1.1970 00:00:00.

Hlavním pozitivem první možnosti je skutečnost, že uložená hodnota zcela odpovídá zobrazované hodnotě; není tedy třeba před zobrazením provádět výpočet. Tento způsob také kopíruje formát, jímž jsou data uložena ve formátu *iCalendar*. Existuje ale i mnoho negativ. Atributy zabírají více místa v paměti a jejich vzájemné porovnávání je algoritmicky složitější. Nakonec jsem jako ideální zvolila první možnost s šesti proměnnými, a to především kvůli kompatibilitě se standardem *iCalendar*. Díky jejímu využití je také možné vyhnout se nutnosti častého přepočítávání jediné hodnoty na pro člověka pochopitelný čas, ke kterému by muselo docházet při činnosti programu velice často. Toto byl také důvod, proč jsem se rozhodla nevyužít standardní C++ knihovnu chrono, která čas ukládá právě do jedné proměnné, evidující čas uběhlý od epochy. I při využití této knihovny by navíc bylo nutné implementovat kompletní kalendářní správu.

Třída tedy obsahuje proměnné evidující rok, měsíc, den, hodinu, minutu a sekundy. Definuje celkem dva konstruktory. Bezparametrový konstruktor nastaví čas na aktuální čas a konstruktor s šesti nastaví hodnoty všech instančních proměnných, přičemž při potřebě

nastavení pouze data je možné využít výchozích nulových hodnot nastavených pro časové proměnné a definovat pouze hodnoty data.

Hodnoty těchto proměnných lze získat prostřednictvím `Get` metod a jejich nastavení probíhá výhradně prostřednictvím `Set` metod. Ty zajišťují korekci zadané hodnoty. Je-li jejich prostřednictvím například zadáno datum 41.3.2017, je jeho podoba upravena na 10.4.2017 a analogicky. Tato funkčnost umožňuje jednoduché relativní posouvání se v čase prostřednictvím přidání či odebrání konkrétního počtu jednotek, o něž se má datum či čas posunout.

Porovnávání hodnot probíhá prostřednictvím redefinovaných operátorů. Operátory <, >, <=, >=, == a != porovnávají datové i časové složky instancí, zato operátory $\ll$ a $\gg$ porovnávají pouze jejich datové složky, uložený čas tedy výsledek neovlivňuje. Pro porovnání dat bez časů na rovnost slouží metoda `IsSameDate()`.

Třída umí pracovat i se dny v týdnu. Algoritmus byl navržen na základě algoritmu od Michaela Keitha a Toma Cravera [14], který zajišťuje převod data na den v týdnu prostřednictvím jediného výrazu v jazyce C.

Třída na uchovávání data a času samozřejmě musí podporovat i práci s přestupnými roky. Za tímto účelem byla navržena metoda `LeapYear()`, která kontroluje, zda se současné datum nachází v přestupném roce či nikoliv. Užitečná je i metoda `SetToNow()`, která přepíše proměnné aktuálním časem.

Poslední zajímavostí této třídy je tzv. „prázdné datum“. Je to takové datum, jehož všechny proměnné jsou naplněny nulovými hodnotami. Je používáno například jako návratová hodnota v případě, kdy nemůže být vygenerováno žádně platné datum.

Třída `cEvent`

Tato třída zaštiťuje veškerou správu jednotlivých naplánovaných událostí. Ukládá data o události, například čas začátku a konce události, název události, její kategorii, příznak opakování události a v neposlední řadě i seznam upomínek na událost.

V této třídě existuje rozhraní pro komunikaci s knihovnou *icalendarlib*. Umožňuje převod instancí událostí z knihovny *icalendarlib* na události typu `cEvent` kalendářní knihovny a naopak, díky této skutečnosti je tedy možné provádět import a export událostí z a do souboru. Je tak naplněn jeden z hlavních požadavků na aplikaci.

Třída samozřejmě také umožňuje vkládání nových událostí přímo uživatelem. Taková událost je pak určena počátečním časem a předdefinovaným trváním. Její název je automaticky nastaven jako prázdný.

Třída samozřejmě obsahuje celou řadu `Get` a `Set` metod, ty druhé jsou však přístupné pouze třídě samotné a spřátelené třídě `Calendar`. V této kalendářní aplikaci existují dva typy událostí. Jednorázové události a opakující se události. Jednorázová událost je taková událost, která se vyskytuje pouze jednou, a to v časovém okně, které je definováno jejím začátkem a koncem. Opakující se události se naopak vyskytují vícekrát. Jejich první výskyt je definován stejně jako u jednorázové události začátkem a koncem jejich konání. Čas každého jejich dalšího výskytu je pak definován časem prvního výskytu zvětšeným o opakovací pravidlo vynásobené počtem opakování, pro než chceme hodnotu zjistit.

Skutečnost, že existují tyto dva typy událostí se samozřejmě musí projevit i v návrhu třídy, která s nimi pracuje. S opakovanými událostmi souvisí speciální metoda `OccursOn(DateTime *)`, která zjišťuje, zda se nějaké opakování události vyskytuje v datu zadaném parametrem. Odpovídá pak příslušnou booleovskou hodnotou.

Je zřejmé, že nakolik jsou si oba tyto typy událostí podobné, není možné spravovat oba stejným způsobem. Aby bylo možné tyto typy odlišit, existuje metoda `IsRecurrent()`, která vrátí boolevskou hodnotu 0 v případě, že se jedná o jednorázovou událost a hodnotu 1 v případě, že se událost opakuje. Skutečné odlišování těchto typů událostí však probíhá až v režii třídy `Calendar`.

Třída `cAlarm`

Tato třída slouží k zaznamenávání informací o upomínkách na události. Stejně jako se podle standardu *iCalendar* nacházejí záznamy o upomínkách jen v tělech událostí, je i tato třída navržena tak, aby její instance byly pevně spjaty s třídou `cEvent`. Tato skutečnost se projevuje například tak, že třída kopíruje přístup standardu k uložení času upomínky. Čas je ukládán v relativní podobě vzhledem k datu a času začátku události.

Podobně jako třída uchovávající informace o událostech i tato třída poskytuje rozhraní pro komunikaci s knihovnou *icalendarlib*, odkud tak může čerpat a také skrze ni zpětně ukládat data podléhající standardu *iCalendar*.

Instance této třídy jsou navrženy tak, aby si uchovávaly informace o času spuštění upomínky, který je relativní k času události, příznak o aktivitě alarmu, ale také odkaz na událost, které se upomínka týká. Díky tomuto odkazu je například možné prostřednictvím metody `TriggerTime()` dopočítat absolutní čas, kdy má být upomíka spuštěna. Kromě toho i tato třída obsahuje celou řadu `Get` a `Set` metod.

Třída `Calendar`

Tato třída tvoří hlavního prostředníka mezi uživatelem knihovny a jejími metodami. Ze své podstaty zaštiťuje všechny zde obsažené třídy s jedinou výjimkou v podobě samostatně využitelné třídy `DateTime`. Její pevné sepětí s ostatními třídami je stvrzeno tím, že je deklarována jako přítel třídy `cEvent`.

Myšlenka hlavní kalendářní třídy sdružující v sobě všechny evidované události je jádrem celé knihovny. Aby bylo možné docílit efektivní spolupráce všech komponent, bylo nejprve nutné pozastavit se nad návrhem datových struktur, v nichž budou potřebné záznamy evidovány. Této problematice se budu věnovat v několika následujících odstavcích.

Pro evidenci událostí jsem zvolila datový typ seznam odkazů na události, tedy `list<*cEvent>`. Předností datového typu seznam [3] je totiž skutečnost, že umožňuje vkládání a odstraňování prvků z jakéhokoliv místa ze seznamu a to s lineární časovou složitostí. Jelikož právě vkládání a odstraňovaní je obdobou přidávání a odebírání událostí, jevilo se toto řešení jako ideální, a bylo použito i pro evidenci upomínek.

Události jsou tedy evidovány ve dvou seznamech, do kterých jsou roztříděny dle svého typu při startu aplikace. Jeden seznam je určen pro události jednorázové, druhý pro ty opakující se. Jedno mají ale oba seznamy společné. Události jsou zde seřazeny vzestupně dle data a času konce, v případě opakujících se dle konce jejich posledního výskytu. Procházení těchto seznamů však probíhá reverzně, události jsou tedy procházeny směrem od budoucnosti k minulosti.

Tento přístup přináší hned několik výhod. Primárně je to možnost prohledávání po překročení hledaného data jakožto data ukončení události přerušit. To má pozitivní efekt na výkon aplikace. Dá se také předpokládat, že na budoucnost bude v kalendáři naplánovaných méně událostí, než je evidovaných v minulosti. Minulé události se díky tomuto přístupu nemusí procházet, efektivita tedy netrpí ani z tohoto pohledu. Tato výhoda by se ovšem neprojevila v případě, kdy by se uživatel chtěl podívat na své záznamy v dávné minulosti.

Stále ovšem vyvstává otázka, proč rozdělovat evidenci událostí do dvou seznamů. Toto rozhodnutí bylo během návrhu učiněno především z toho důvodu, že se na každý typ události musí nahlížet jiným způsobem. Zatímco okno výskytu jednorázové události končí jejím datem ukončení, u opakujících se události nemusí končit dokonce nikdy. I při zjišťování, zda se konkrétní událost v konkrétní den vyskytuje, je nutné přistupovat k procházení obou seznamů jiným způsobem. Jednorázovou událost stačí otestovat pouze jednou, zato u opakujících se je nutné prozkoumávat všechny opakování, dokud není hledaný čas překročen.

Kromě seřazených seznamů evidujících naplánované události obsahuje třída ještě třetí seznam. Ten eviduje upomínky naplánované na aktuální datum. Ty jsou opět seřazeny vzestupně dle času, kdy mají proběhnout. Už ovšem na rozdíl od událostí nejsou procházeny reverzně, nýbrž od začátku do konce. Tento seznam se stejně jako oba předchozí tvoří při startu aplikace, kromě průběžných aktualizací je však při změně data (například o půlnoci) jeho obsah zcela odstraněn a seznam je znovu naplněn upomínkami odpovídajícími novému aktuálnímu datu. S tímto seznamem souvisí i proměnná evidující alarm, který je další na řadě ke spuštění, a příznak, zda takový alarm existuje.

Datové struktury ovšem nejsou zdaleka to jediné, čím třída knihovně přispívá. Právě jejím prostřednictvím je totiž prováděn import a export z nebo do souboru standardu *iCalendar*. Toto probíhá příznačně pojmenovanými metodami `Import(string)` a `Export(string)`. Zajímavostí těchto metod je skutečnost, že je možné je volat i bez poskytnutí parametru. V takovém případě je pak pro datový vstup či výstup využit výchozí soubor. Tímto způsobem tak probíhají všechna interní uložení i načítání uložených záznamů při spuštění aplikace. V případě zadání parametru je pak využit v něm jmenovaný soubor.

Tato třída neobsahuje typické `Get` a `Set` metody, zato však obsahuje celou řadu metod pro vkládání a odstraňování událostí a upomínek z jejich seznamů. Z důvodu nutnosti zachování konzistence řazení seznamů a rozdělení záznamů dle jejich typů je také nutné provádět úpravy v proměnných událostí prostřednictvím rozhraní této třídy. Z tohoto důvodu vznikla skupinka metod s prefixem `Modify`, jejichž prvním parametrem je vždy právě odkaz na modifikovanou událost.

Poslední skupina metod souvisí právě se seznamem upomínek pro aktuální den. Existuje tak metoda pro zjištění, zda na aktuální den ještě nějaká upomínka existuje, metoda pro získání aktuální upomínky, která je v současné době první na řadě, či metoda, která po uskutečnění upomenutí nastaví další upomínku v seznamu jako aktuální. V neposlední řadě je k dispozici i metoda, která provádí vyprázdnění a znovu naplnění seznamu upomínek při přechodu na další den.

3.4 Třída pro uživatelská nastavení

Návrh této třídy byl dalším krokem k uživatelsky přívětivé aplikaci. Každý uživatel totiž jistě ocení možnost nastavit si výchozí trvání nově vytvářené události, výchozí dobu upomínky na událost, či možnost nastavit, zda se má při vytvoření události automaticky přidat i výchozí upomínka.

Kromě těchto uživatelem specifikovaných parametrů jsem se však rozhodla nezávisle na vůli uživatele evidovat i informace o poloze a velikosti okna, aby je byla aplikace při dalším spuštění s to automaticky reprodukovat.

Abych tyto informace v aplikaci zpřístupnila, rozhodla jsem se vytvořit speciální třídu, která se bude zabývat jejich správou. Při jejím návrhu jsem využila návrhový vzor singleton. K tomuto kroku jsem se rozhodla především z toho důvodu, že jsem chtěla zabránit

předčasné inicializaci této třídy. Ve prospěch využití tohoto návrhového vzoru se přikláněla i skutečnost, že se v celé aplikaci bude vždy vyskytovat jediná instance této třídy.

Za účelem zachování informací evidovaných v této třídě i po ukončení běhu aplikace však bylo nutné využít „.ini“ souboru. Ten je s to data do dalšího spuštění aplikace bezpečně uchovat. S využitím souboru ovšem vyvstala i potřeba knihovny, která bude schopna data v něm interpretovat a upravovat. Touto knihovnou se nakonec stala knihovna *RudeConfig*.

Knihovna *RudeConfig*

Jedná se o knihovnu vyvíjenou společností RudeServer, která má na kontě i mnoho dalších knihoven se zaměřením například na Databáze nebo sockety. Tato knihovna se ovšem zabývá právě čtením a úpravou ini souborů.

Hlavním požadavkem na tuto knihovnu byla kromě schopnosti spravovat .ini soubory také multiplatformnost. Tu knihovna *RudeConfig* splňuje; je totiž dostupná jak pro Windows, tak pro Linux. Při kompilaci tato knihovna sice vypisuje několik varování, ta ovšem nejsou nijak zásadní, nebere se tedy na ně zřetel.

Skrze přívětivé rozhraní tato knihovna umožňuje získávat i vkládat do souboru rozmanité parametry a jejich hodnoty. Pro účely této aplikace není zdaleka využit celý její potenciál, přesto je neocenitelnou pomocnicí.

3.5 Struktura uživatelského rozhraní

V této sekci se zmíním o základní struktuře návrhu uživatelského rozhraní. Tentokrát ovšem ne z pohledu jeho grafického návrhu, který již byl rozebrán v sekci 3.2, nýbrž z pohledu strukturálního návrhu. Diagramy dále zmiňovaných tříd jsou k nahlédnutí v příloze B jako obrázky B.1 a B.2.

Hlavním problémem z pohledu návrhu bylo nalezení ideálního způsobu propojení rozhraní s kalendářní knihovnou a třídou pro nastavení. Pro efektivní práci je totiž potřeba po jedné instanci od třídy `Calendar` i třídy pro nastavení. Tohoto stavu jsem docílila využitím globálních proměnných.

Obecná struktura rozhraní

Prvním krokem při návrhu struktury rozhraní bylo vytvoření třídy, která zaštítí prvky sdílené napříč mezi všemi obrazovkami aplikace. Touto třídou se stala třída `MainWindow`.

Jednou z pravomocí této třídy je například kompletní správa hlavního panelu nástrojů aplikace. Upravuje podbarvení ikon na základě vybraného zobrazení. Kromě toho se stará i o vykreslení vybraného prvku do zbytku rozhraní.

Tato třída dále sdružuje nabídky menu a jejich obsluhu. To zahrnuje například i okno s nastaveními, interagující se třídou pro nastavení popsanou v sekci 3.4. To umožňuje uživateli přizpůsobit určité parametry aplikace svým potřebám. Kromě nastavení jsou zde k dispozici i možnosti jako je hromadné mazání událostí, importování či exportování kalendářních dat či skok na zobrazení vybraného dne.

Poslední, ovšem neméně důležitou pravomocí této třídy je správa časovače. Jeho úkolem je především správa upomínek. Obsluha tohoto časovače se spouští každou vteřinu. Zajišťuje, aby byly upomínky provedeny ve specifikovaném čase a aby po jejich proběhnutí byla korektně nastavena další upomínka. Kromě toho také tento časovač kontroluje, zda se ak-

tuální datum nezměnilo a není tedy třeba ve třídě `Calendar` upravit obsah pole evidujícího upomínky na aktuální den. Pokud ano, je zavolána příslušná metoda, která toto zajišťuje.

Zobrazení dne

Zobrazení dne je hlavním zobrazením celé aplikace. Očekává se, že bude nejčastěji používaným zobrazením vůbec. Kromě toho je ovšem také velice komplexní. Obsahuje totiž mnoho prvků, které je nutné na základě vstupu od uživatele překreslovat.

Prvním prvkem, který toto zobrazení spravuje, je vedlejší ovládací panel. Ten dle vzhledového návrhu zobrazuje datum, pro jehož vykreslení byl vybrán prvek `QLabel`. Také je zde k dispozici přepínač mezi dny v týdnu. Ten je vyřešen sadou sedmi vygenerovaných tlačítek založených na typu `QPushButton`, odpovídajících jednotlivým dnům v týdnu. Při poklepání na tlačítko se zobrazení přepne do vybraného dne. Podbarvuje se opět aktuálně vybraný den. Tato tlačítka jsou dále z každé strany ohraničena dalším prvkem typu `QPushButton` s ikonou šipky. Tyto prvky opět slouží pro přepínání mezi dny.

Největším oříškem při návrhu tohoto zobrazení se ovšem stal hlavní panel aplikace. První komplikací byl výběr vhodné Qt komponenty, která by umožnila vhodným způsobem vykreslit časovou mřížku s událostmi. Bylo nutné vybrat takovou komponentu, která přizpůsobí svou velikost obsahu. Také bylo vhodné, aby tato komponenta umožňovala zobrazování objektu přes více buněk, v tomto případě události do počtu buněk závislém na době jejího trvání. Těmito vlastnostmi vyniká právě Qt objekt `QGridLayout`, který organizuje vkládané objekty do mřížky.

Dalším řešeným dilematem byl výběr vhodného objektu pro vykreslování událostí. Na ten byly celkem tři hlavní požadavky. Objekt musel umožňovat dynamickou změnu své velikosti; události totiž musí zabírat prostor odpovídající délce jejich trvání. Dalším požadavkem byla schopnost zobrazit a editovat text, tedy název události. Posledním kritériem pro výběr objektu byla schopnost objektu vykreslit své pozadí barevně. Zprvu se jako vhodný jevil již dříve používaný objekt `QLabel`. Ten ovšem neumožňuje jednoduše editovat a získat editovaný text. Z tohoto důvodu jsem se při návrhu nakonec přiklonila k objektu `QTextEdit`, který skvěle splňuje všechny požadavky.

V tomto objektu bylo dále nutné při vznášení kurzoru nad jeho prostorem zobrazit tlačítko koše, které by záznam o události při poklepání odstranilo. Tlačítko bylo opět navrženo za využití objektu `QPushButton`. Pro jeho vykreslování a skrývání byla navržena reimplementace metod `enterEvent(QEvent*)` a `leaveEvent(QEvent*)`.

Posledním řešeným problémem byl návrh algoritmu pro správné vykreslování událostí. Ten bylo třeba navrhnout tak, aby se vypořádal i s časově se překrývajícími událostmi. Návrh algoritmu prošel několika iteracemi než se ustálil na současné podobě. V té se postupně iteruje po událostech nalezených na specifický den. V počáteční fázi každé iterace jsou využity dvě metody. První opatří index první buňky, kam se má událost vykreslit, druhá spočítá rozpětí události do buněk. Následuje algoritmus, který postupně prochází všechny sloupce aplikace. Pokud jsou potřebné buňky ve sloupci již nějakou událostí obsazeny, prohledává následující sloupec. Jsou-li volné, je událost umístěna a algoritmus pokračuje další iterací.

Aktualizace zobrazení probíhají téměř po každé interakci uživatele s rozhraním. Jedinou výjimkou, na niž se aktualizace nevztahuje, je snad jen použití posuvníku. Aktualizace probíhá tak, že se Odstraní celý obsah hlavního panelu aplikace s výjimkou časové osy. Poté jsou znovu vyhledány události vyskytující se v zobrazovaném dni a jsou vloženy na odpovídající místa v časové ose. Toto překreslování zajišťuje korespondenci zobrazovaných

a faktických záznamů. Dochází k němu totiž například ve chvíli, kdy je přidána či smazána událost.

Zobrazení týdne

Již z grafického návrhu je patrné, že zobrazení týdne je v mnohém velice podobné zobrazení dne, popsanému v podsekci 3.5. Při jeho návrhu se tedy již počítalo s tím, že zde budou využity prvky z tohoto zobrazení.

Prvním navrhovaným prvkem byl vedlejší panel nástrojů. Ten se použitými objekty příliš neliší od panelu ve zobrazení dne. Jediným rozdílem je zde náhrada tlačítek, reprezentujících dny v týdnu, štítkem typu `QLabel`, který obsahuje pořadové číslo aktuálně prohlíženého týdne.

Hlavní panel aplikace pak sestává z osmi sloupců. První je naplněn hodnotami časové osy, dalších sedm pak příslušnými událostmi vyskytujícími se v jednotlivých dnech v týdnu. Všechny tyto prvky jsou převzaty ze zobrazení dne. V hlavičce každého sloupce je pak štítek `QLabel` s označením dne v týdnu a jeho data.

Aby bylo zobrazení použitelné, bylo nutné zajistit možnost přidávání událostí. Ta byla nakonec navržena tak, že uživatel nejprve vybere den, tedy sloupec, kam chce událost umístit, poklepáním na jeho záhlaví. Následně stejně jako ve zobrazení dne vybere příslušnou hodinu, na kterou chce událost naplánovat.

Aktualizace tohoto rozhraní funguje obdobným způsobem jako ve zobrazení dne. Je pouze doplněna o několik dodatečných kroků, které například zajistí aktuální informace v hlavičkách jednotlivých dní a proiterují všemi dny ke zobrazení.

Zobrazení měsíce

Zobrazení měsíce patří spíše k těm, u nichž není předpokládáno časté využití. Slouží spíše k přehledu nad delším časovým intervalem, kde už není možné uživateli zpřístupnit kontrolu nad naplánovanými událostmi, pouze mu je nějakým způsobem zobrazit. Hlavní funkcí tohoto zobrazení pak je rychlé přesouvání se mezi velkými časovými intervaly a přehled nad časovým schématem především nadcházejících měsíců.

Vedlejší ovládací panel se v tomto zobrazení příliš neliší od panelu ve zobrazení týdne. Jedinou změnou je fakt, že místo týdnů jsou zde šipkovými tlačítky přepínány měsíce.

V hlavním panelu aplikace se nachází objekt `QCalendarWidget`. Ten uživateli zobrazuje vybraný měsíc. Za využití signálu tohoto objektu byla navržena i funkcionalita, zajišťující přepínání mezi měsíci prostřednictvím kolečka na myši. To probíhá zachycováním signálu `currentPageChanged(int year, int month)` a příslušnou úpravou parametrů, následovanou aktualizací zobrazení.

Hlavním účelem tohoto zobrazení je především rychlý přesun na zobrazení dne po vybrání hledaného data. K tomuto přesunu dojde po poklepání na datum ve zobrazovaném kalendáři. Aplikace se tak přepne do zobrazení dne, kde vykreslí záznamy pro vybrané datum.

Zobrazení roku

Stejně jako zobrazení týdne přebírá prvky ze zobrazení dne, i zobrazení roku využívá prvky navržené pro zobrazení měsíce.

Vedlejší panel nástrojů je zde opět obdobný jako u zobrazení týdne či měsíce; jediným rozdílem je chybějící štítek pro datum. Přepínač zde tentokrát slouží k listování mezi roky.

Je to také jediný způsob přepínání tohoto zobrazení; na rozdíl od zobrazení měsíce už zde neexistuje podpora kolečka myši.

Hlavní zobrazení obsahuje 12 kalendářních objektů typu `QCalendarWidget` nadepsaných štítkem typu `QLabel` s názvem měsíce. Tyto objekty mají stejné schopnosti jako kalendářní objekt ve zobrazení měsíce; při poklepání na konkrétní datum tedy uživatele přesměrují na zobrazení vybraného dne. Všechny tyto objekty jsou zmenšeny na minimum, aby bylo zobrazení co možná nejúspornější.

Pěkným rysem, který byl pro toto zobrazení navržen, je přeskládávání kalendářních objektů do tří či čtyř sloupců na základě poměru aktuální výšky a šířky okna aplikace. Za tímto účelem je využito sledování Qt události typu `QResizeEvent`.

Shrnutí návrhu struktury uživatelského rozhraní

Při návrhu struktury uživatelského rozhraní byl mnohdy největším dilematem výběr objektu vhodného pro daný účel. V mnoha případech došlo k situaci, že na počátku návrhu byl prvek reprezentován zcela odlišným objektem, než jaký byl vybrán ve finále. Při výběru Qt komponent splňujících kladené požadavky mi byla velkým pomocníkem online dokumentace frameworku Qt 5 [2]. Dalším neocenitelným zdrojem informací byla kniha [7] popisující tvorbu rozhraní ve starší verzi Qt 4.

Kapitola 4

Implementace a testování

Program *Calendar* byl implementován v jazyce C++11 za využití frameworku Qt verze 5.8. Tato kombinace podporuje přenositelnost aplikací. Jelikož přenositelnost patří k požadavkům na aplikaci, byl na ni při implementaci brán nemalý zřetel. Implementace probíhala především na 64 bitovém operačním systému Ubuntu 14.04 LTS. Byly ovšem provedeny takové kroky, které umožnily běh aplikace kromě primárního linuxového systému i na systémech z rodiny Windows. Aplikace byla úspěšně spuštěna na 32 i 64 bitovém systému Windows 7.

4.1 Výsledná podoba aplikace

Výsledný vzhled i funkcionalita aplikace odpovídají provedenému návrhu. Aplikace poskytuje čtyři hlavní zobrazení, zobrazení dne, týdne, měsíce a roku. Ta jsou dostupná prostřednictvím ikon v hlavním panelu nástrojů aplikace. Hlavní okno aplikace podporuje změny velikosti a pamatuje si svou pozici od předchozího spuštění. Aplikace se dokonce dokáže vyrovnat i se změnou počtu monitorů; svou pozici přizpůsobí jak za běhu, tak i při novém spuštění.

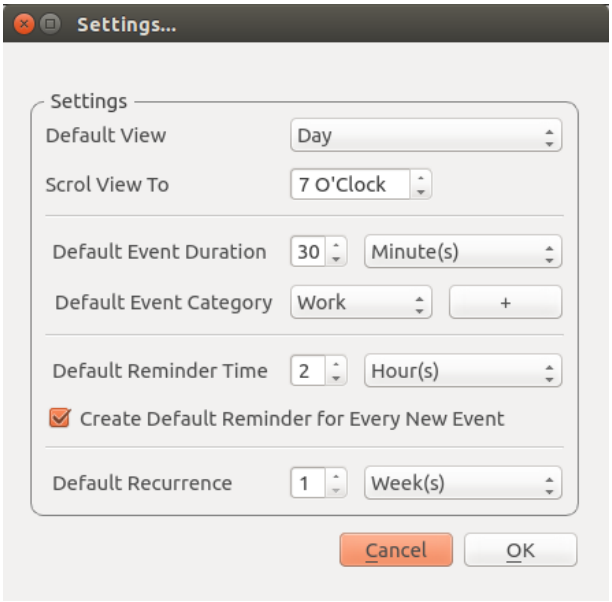
Čtveřici hlavních ikon zobrazení doplňuje ikona pro zobrazení dnešního dne. Při jejím výběru je čas, zobrazovaný na aktuální obrazovce upraven tak, aby odpovídal dnešnímu dni. Tato akce je přístupná i z nabídky menu, společně s akcí přesměrovávající zobrazení aplikace na jakékoliv datum vybrané uživatelem.

Z nabídky menu aplikace jsou přístupné i funkce jako Import či Export kalendářního souboru s událostmi. Také je zde k dispozici funkce hromadného mazání událostí, a to například na základě časového intervalu.

Aplikace je lokalizována do anglického jazyka.

Nastavení aplikace

Posledním prvkem spustitelným z nabídky menu je nastavení aplikace. To uživateli umožňuje ručně nastavit parametry jako například na jakou počáteční hodinu se mají posunovat zobrazení s časovou osou, jaká je výchozí délka události, případně zda se pro novou událost má automaticky vytvořit i upozornění.

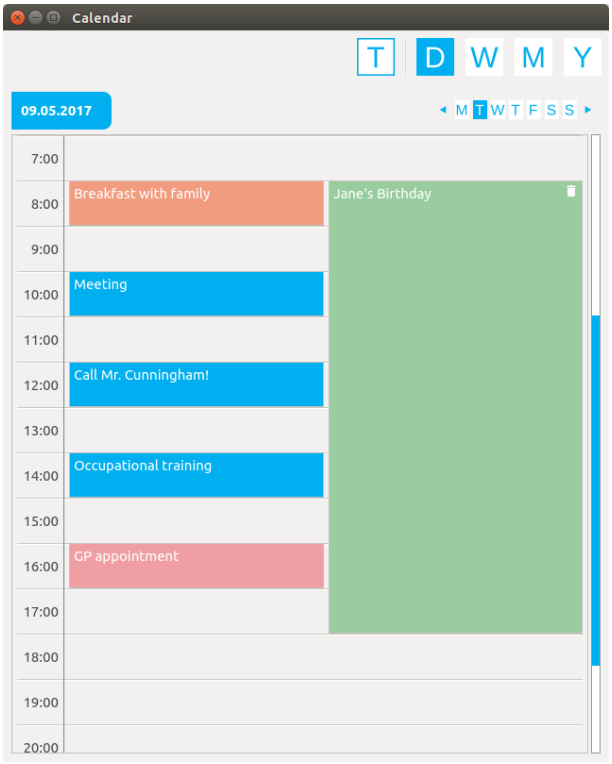


Obrázek 4.1: Aplikace Calendar, nastavení aplikace

Zobrazení dne

Zobrazení dne se nijak neliší od svého návrhu, a to jak vzhledem, tak funkčností. Události se vykreslují na plochu, která odpovídá délce jejich trvání. Rozlišení je 15 minut, přičemž krátká událost se vždy vykreslí alespoň do 30 minutového pole. Události jsou podbarveny barvou odpovídající jejich kategorii. Kategorie má vždy automaticky přiřazenu jednu konkrétní barvu, kterou jsou vykresleny všechny pod ni spadající události.

Událost lze přejmenovávat přímo v tomto zobrazení. Pro pokročilejší editaci lze otevřít dialog dvojitým kliknutím na štítek konkrétní události. Mazání probíhá prostřednictvím ikony odpadkového koše, která se objeví v pravém horním rohu štítku, pokud se kurzor myši nachází nad štítkem události. Novou událost lze přidat kliknutím na konkrétní čas v časové ose.



Obrázek 4.2: Aplikace Calendar, zobrazení dne

Zobrazení týdne

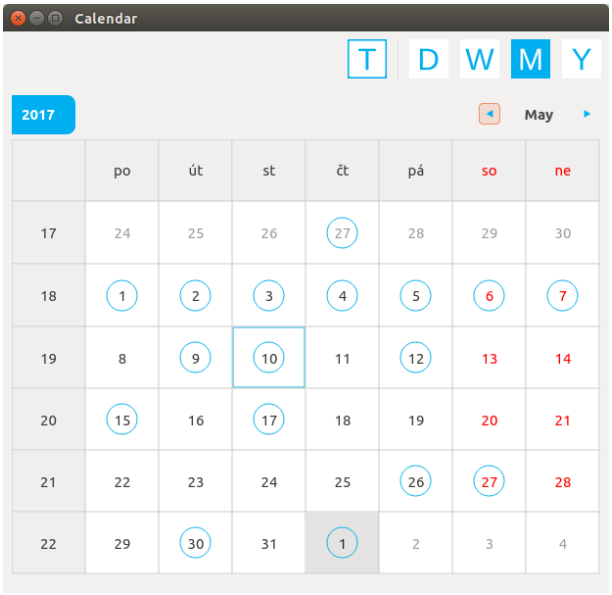
Zobrazení týdne má mnoho společného se zobrazením dne. Jeho ovládání probíhá velmi analogicky, jen výběr editovaného dne se prování kliknutím do jeho záhlaví. Dvojitým kliknutím na záhlaví se pak vybraný den zobrazí ve výchozím zobrazení. Dále už se s obrazovkou i událostmi manipuluje stejným způsobem jako ve zobrazení předchozím.

Tato obrazovka se trochu liší od svého návrhu. Dle něj se ve štítcích událostí neměly zobrazovat jejich názvy a nebyla tudíž možná přímá editace názvu. Od této myšlenky bylo při implementaci upuštěno a názvy byly zobrazeny. Obrazovka je díky tomuto kroku mnohem využitelnější, jelikož uživatel může kromě časového rozvrhu udržovat přehled i nad konkrétními událostmi.

Obrázek 4.3: Aplikace Calendar, zobrazení týdne

Zobrazení měsíce

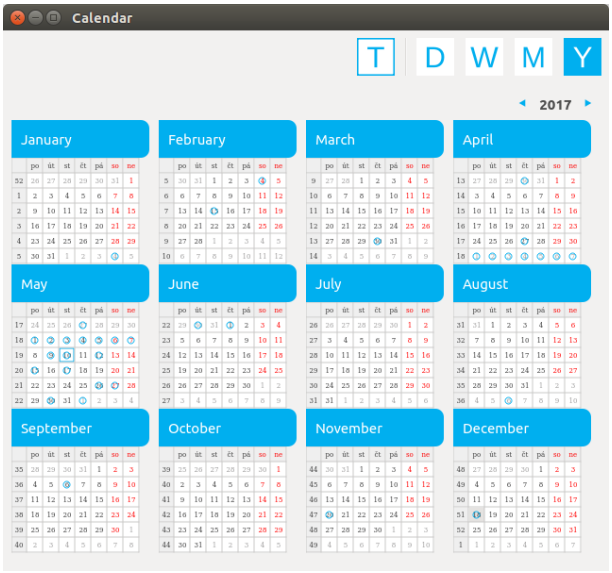
I zobrazení měsíce odpovídá původnímu návrhu. Jedná se o klasickou tabulku znázorňující dny v konkrétním měsíci. Dnešní den je v tomto zobrazení ohraničený modrým čtverečkem. Dalším specifikem je označení dní, na něž jsou naplánovány nějaké události, modrým kroužkem vykresleným kolem čísla dne. Při dvojitém kliknutí na konkrétní den se provede buď přeskok na příslušný měsíc, nebo se vybraný den zobrazí ve výchozí obrazovce.



Obrázek 4.4: Aplikace Calendar, zobrazení měsíce

Zobrazení roku

Zobrazení roku je v mnohém podobné zobrazení měsíce. I zde jsou vykreslovány kružnice kolem dnů z událostmi, stejně tak funguje i výběr konkrétního dne. Specifikem tohoto zobrazení však je schopnost přizpůsobit svůj obsah poměru šířky a výšky aplikace. Matice obsahující měsíce tak dynamicky mění počet sloupců a řádků.



Obrázek 4.5: Aplikace Calendar, zobrazení roku

4.2 Zátěžové testování

Testování aplikace probíhalo na výše zmíněném operačním systému Ubuntu. Referenční stroj disponoval 2.9 GiB operační paměti a procesorem Intel Core 2 Duo, každé jádro s frekvencí 2.10 GHz. Bylo provedeno několik testů zaměřených především na dobu inicializace aplikace a dobu aktualizace nejsložitějšího zobrazení, zobrazení týdne, v závislosti na objemu dat. Čas inicializace byl měřen prostřednictvím linuxového příkazu time, čas strávený aktualizováním zobrazení byl měřen pomocí knihovny chrono přímo v aplikaci. Data pro tyto testy byla generována prostřednictvím kalendářní knihovny.

Nejprve byla aplikace spuštěna bez poskytnutí souboru obsahujícího data. Aplikace se tedy spouštěla bez jakéhokoliv načítání vstupních dat. K načtení rozhraní aplikace došlo okamžitě bez jakékoliv prodlevy. Odezva aplikace na uživatelské akce byla velmi rychlá.

Další test představovalo spuštění aplikace s předpokládaným obvyklým až větším množstvím dat, která ovšem byla střádána po dlouhou dobu. Data obsahovala celkem 5 000 událostí. Ty byly rozloženy do období téměř čtyř let takovým způsobem, že každý pracovní den obsahoval pět událostí.

Načtení těchto dat zpomalilo spouštění aplikace o přibližně 3 sekundy. Za běhu nebylo žádné zpoždění pozorováno. Odezva aplikace zůstala zachována stejná jako v případě aplikace neobsahující žádná načtená data. Zátěžovým testem pak bylo spuštění aplikace s enormními daty. Tato data obsahovala celkem 10 000 událostí. Ty byly v kalendáři rozmístěny tak, že na každý den včetně víkendů po dobu přibližně 3 let bylo naplánováno deset událostí.

Spouštění aplikace s těmito daty sice trvalo na stroji přibližně 17 sekund, samotný běh aplikace však byl již velice plynulý. Rychlost odezvy na akci uživatele se nijak nelišila od rychlosti odezvy aplikace bez jakýchkoliv dat. Smazání přibližně 6 000 událostí za běhu nemělo na výkon aplikace taktéž žádný vliv.

Počet událostí	**Doba spouštění**	**Doba odezvy**
0	0.3 s	99 ms
5 000	3.1 s	127 ms
10 000	16.6 s	168 ms

Tabulka 4.1: Tabulka znázorňující rychlost odezvy aplikace v závislosti na objemu načítaných a uchovávaných dat

Zajímavý je poznatek, že doba načítání nemá lineární závislost na objemu načítaných dat. Tato skutečnost je zapříčiněna způsobem vkládání událostí do seznamu událostí. Vkládání má lineární složitost odvíjející se od počtu prvků v seznamu. Postupným narůstáním tohoto počtu se zvyšuje i složitost vkládání a tudíž i doba načítání aplikace.

Aplikace si se zátěžovými testy poradila poměrně uspokojivě. Jediná člověkem indikovatelná časová prodleva nastala při spouštění aplikace. Při běhu aplikace už žádná zpoždění nebyla detekována. Jelikož z hlediska uživatelského pocitu je mnohem přijatelnější chvíli počkat při spouštění aplikace, než se vyrovnávat se zpožděnou odezvou, jsem přesvědčena, že výsledky zátěžových testů tak vrhají výkon aplikace jako celku do pozitivního světla.

Kapitola 5

Závěr

Cílem této bakalářské práce bylo prozkoumat existující kalendářní aplikace a na základě získaných vědomostí navrhnout a vytvořit takovou kalendářní aplikaci, která uživateli zprostředkuje kalendářní funkce skrze co nejefektivnější uživatelské rozhraní. Bylo nastudováno několik kalendářních aplikací z různých platforem, přičemž jako příklad velmi vhodného uživatelského rozhraní byla zvolena aplikace Calendar na zařízení *Tungsten E*. Dále byl analyzován aktuální standard pro ukládání kalendářních dat *iCalendar*, který aplikaci umožňuje komunikaci s podobnými druhy softwaru.

V souladu s analýzou kalendářních aplikací a vhodných praktik při návrhu uživatelských rozhraní bylo navrhnuto grafické rozhraní aplikace. Za účelem maximální efektivity byly také využity vhodné prvky ze zkoumaných aplikací. Následoval návrh kalendářní knihovny, zajišťující podporu nutných funkcí pro činnost kalendáře a návrh struktury grafického rozhraní.

Aplikace byla implementována v jazyce C++11 za využití frameworku Qt 5. Kromě standardních knihoven byly využity i externí knihovny, zajišťující podporu standardu *iCalendar*. Na závěr byla aplikace podrobena testování.

Výsledná aplikace disponuje efektivním uživatelským rozhraním, díky němuž najde uplatnění u širokého spektra uživatelů. Tato skutečnost je podpořena i přenositelností aplikace na různé operační systémy.

Další vývoj aplikace může směřovat k zavedení podpory dalších prvků podporovaných kalendářním standardem, jako jsou například seznamy úkolů. Rozšíření by však neměla být prováděna na úkor efektivity rozhraní. Aplikace by pak mohla sloužit více účelům.

Literatura

[1] *MIUI*. [Online; navštíveno 3.5.2017].
URL http://en.miui.com/a-39.html

[2] *Qt Documentation*. [Online; navštíveno 15.2.2017].
URL http://doc.qt.io/qt-5/

[3] *Standard C++ Library reference*. [Online; navštíveno 15.2.2017].
URL http://www.cplusplus.com/reference/

[4] *The iCalendar Standard*. [Online; navštíveno 15.1.2017].
URL https://icalendar.org/uncategorised/

[5] *KOrganizer*. 2013-07-05 [cit. 2017-04-11], [Online; navštíveno 15.1.2017].
URL https://userbase.kde.org/KOrganizer

[6] *Kontact*. 2016-02-29 [cit. 2017-04-11], [Online; navštíveno 15.1.2017].
URL https://userbase.kde.org/Kontact

[7] BLANCHETTE, J.; SUMMERFIELD, M.: *C++ GUI Programming with Qt 4*. Trolltech ASA, 2008, ISBN 0-13-235416-0.

[8] DAWSON, F.; STENERSON, D.; DESRUISSEAUX, B.: *Internet Calendaring and Scheduling Core Object Specification (iCalendar)*. 2009-09-01 [cit. 2017-01-15], [Online; navštíveno 15.1.2017].
URL https://tools.ietf.org/html/rfc5545

[9] GADE, L.: *Palm Tungsten E [online]*. 2003-10-01 [cit. 2016-12-11], [Online; navštíveno 11.12.2016].
URL http://www.mobiletechreview.com/palm_Tungsten_E.htm

[10] GALITZ, W. O.: *The Essential Guide to User Interface Design: An Introduction to GUI Design*. Wiley, 2007, ISBN 0470053429.

[11] GONERA, J.: *A simple C++ library for the iCalendar (.ics) format*. [Online; navštíveno 15.2.2017].
URL https://github.com/jgonera/icalendarlib

[12] GRANOR, T. E.: *Best Practices for User Interfaces*. [Online; navštíveno 21.1.2017].
URL https://pdfs.semanticscholar.org/d05b/0bb6df30032e779fa28a4bd6b585f77c5238.pdf

[13] HUDEC, J.: *Nový Palm Tungsten E: hodně muziky za málo peněz.* 2003-09-22 [cit. 2016-12-11], [Online; navštíveno 11.12.2016].
URL http://www.mobilmania.cz/clanky/novy-palm-tungsten-e-hodne-muziky-za-malo-penez/sc-3-a-1105329/default.aspx

[14] KEITH, M.; CRAVER, T.: *The ultimate perpetual calendar?* Journal of Recreational Mathematics, *ročník 22, č. 4, 1990: s. 280–282, ISSN 0022-412X.*

[15] KLAŠKA, M.: Vliv barvy uživatelského rozhraní na výkon uživatele. Diplomová práce, Brno: Masarykova univerzita. Fakulta sociálních studií, 2008.

Přílohy

Příloha A

Obsah přiloženého paměťového média

- `README.TXT` – Obsahuje informace o aplikaci a instalaci
- `doc/`
  - `thesis/` – Text této práce a latexové zdrojové kódy
  - `doxygen/` – Dokumentace kódu vygenerovaná prostřednictvím Doxygenu
- `src/` – Makefile
  - `calendar/` – Zdrojové kódy aplikace a použitých knihoven
- `bin/` – Binární soubory
  - `linux/` – Spustitelný soubor pro linux (generuje se příkazem `make` v `src/`)
  - `windows/` – Spustitelný .exe soubor pro Windows

Příloha B

Diagramy tříd

Obrázek B.1: Diagram tříd rozhraní `DayWidget`, `WeekWidget`, `DayLayout`

Obrázek B.2: Diagram tříd rozhraní `MonthWidget`, `YearWidget`, `MainWindow`

Obrázek B.3: Diagram tříd kalendářní knihovny