Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

# BAKALÁŘSKÁ PRÁCE


Jiří Florián

Skupinový diář

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Michal Žemlička, Ph.D.

Studijní program: Informatika

Studijní obor: programování

Praha 2013

Na tomto místě bych chtěl poděkovat vedoucímu mé bakalářské práce RNDr. Michalovi Žemličkovi, Ph.D. za odborné vedení, trpělivost, cenné rady a připomínky při zpracovávání této práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 2. srpna 2013 podpis

Název práce: Skupinový diář

Autor: Jiří Florián

Katedra / Ústav: Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. Michal Žemlička, Ph.D., Katedra softwarového inženýrství

Abstrakt:

Ať si to chceme připustit či nikoli, celý náš život se skládá z plnění rozličných úkolů a povinností. A právě vytvoření návrhu aplikace pro snadné plánování úkolů a především událostí je cílem této bakalářské práce. Konkrétně půjde o aplikaci, která bude vhodná pro spojení osobního i pracovního plánování, aniž by přitom došlo k jakémukoli narušení soukromí.

Klíčová slova: skupinové plánování, diář, oprávnění ke sdílení

Title: Time Manager

Author: Jiří Florián

Department: Department of Software Engineering

Supervisor: RNDr. Michal Žemlička, Ph.D., Department of Software Engineering

Abstract:

Whether we are able to admit it or not, our whole life consists of the performance of various tasks and obligations. And just a draft application for easy scheduling of tasks and especially events is the aim of this thesis. It will be a program that is suitable for both personal and work related planning occurred without any invasion of privacy.

Keywords: Team Planning, Time Manager, Sharing Permissions

# Obsah

# Úvod

Jedním z největších fenoménů posledních let se stala globalizace. Ta s sebou přináší jak pozitiva, tak i řadu negativ, dají-li se takto označit rostoucí nároky a požadavky, které se spolu s globalizací prosazují téměř všude. K pozitivním stránkám globalizace patří především propojování, ke kterému dochází jak ve výrobě, distribuci, logistice, tak například i v komunikaci. Tyto procesy spotřebitelům přináší větší možnosti výběru a zrychlením téměř všeho usnadňují jejich život. Na druhé straně však pro dosažení této životní úrovně kladou na zaměstnance, řídící pracovníky, ale třeba i na personalisty neustále se zvyšující nároky na jejich znalosti, dovednosti a schopnosti, vyžadují od nich stálé učení se něčeho nového, přizpůsobování se měnícím se požadavkům doby. Jinak řečeno, globalizace s sebou rovněž přináší nutnost plnění stále většího množství stále rozmanitějších úkolů. Tyto úkoly je třeba nějakým způsobem organizovat, a to zejména proto, aby nedocházelo ke „ztrácení se v nich“ tím, jak jejich množství i složitost narůstají.

Dříve byly k účelu uspořádání si jednotlivých prací, úkolů a aktivit používány papírové diáře. S dostupností moderních technologii (jež rovněž souvisí s globalizací) se však vše převádí do digitální formy, což postihlo i výše zmíněné papírové diáře. Ty dnes u mnoha lidí vystřídaly diáře elektronické. Elektronické diáře mají oproti papírovým tu výhodu, že nad (či s) vloženými údaji mohou pracovat, samozřejmě za předpokladu, že je nedegradujeme pouze na úroveň papírového zápisníku. Elektronický diář nám pak může pomoci snadněji plánovat náš čas. Propojíme-li diáře více osob, mohou nám dokonce pomoci koordinovat čas i v rámci určité skupiny, např. v rámci pracovního týmu. V tom případě již ale nehovoříme o elektronickém diáři, nýbrž o plánovací aplikaci. Taková aplikace skýtá nejen všechny výhody výše uvedeného elektronického diáře, ale nabízí svým uživatelům i další možnosti. A právě vytvoření návrhu aplikace pro snadné plánování událostí je cílem předkládané bakalářské práce. Konkrétně se bude jednat o aplikaci, jež bude vhodná pro spojení osobního i pracovního plánování, aniž by přitom docházelo k jakémukoli narušení soukromí. V implementační části se autor zaměří zejména na použití systému práv pro zveřejňování informací o událostech. Praktické zhodnocení navrženého a implementovaného systému práv je pak jedním z dalších cílů předkládané práce.

Pokud jde o strukturu bakalářské práce, tak ta bude následující. V kapitole bezprostředně navazující na tento úvod se, vzhledem k výše stanovenému cíli práce,

budeme zabývat analýzou současné nabídky aplikací. Uvedeme zde jejich členění podle tří nejčastěji zohledňovaných kritérií – podle finančního hlediska, podle potřeby instalace a podle jejich technické stránky. Ve druhé kapitole pak naši analýzu dále prohloubíme zkoumáním některých konkrétních, na trhu dostupných, aplikací. Zaměříme se zde především na popis jejich funkčnosti. Třetí kapitola je následně věnována popisu charakteristiky ideálního plánovacího software, včetně hlavních atributů takovéhoto software. Takováto ideální plánovací aplikace musí být zabezpečena určitými technickými prostředky. Způsoby technické realizace takové aplikace se zabývá kapitola čtvrtá předkládané práce. Volba architektury, volba klienta a další implementační rozhodnutí jsou předmětem kapitoly páté. Šestá kapitola se následně zabývá detailním popisem přístupu k právům uživatelů. Sedmá kapitola je poté věnována návrhu logiky aplikace. Osmá kapitola je poté věnována konkrétní implementaci serverové části. Popisujeme zde přístupné třídy konkrétní implementaci serverové části. Devátá kapitola popisuje implementaci klienta. Výběru vhodných pluginu a jejich modifikaci pro naší aplikaci. Poslední, desátá, kapitola shrnuje výsledky práce.

# 1 Analýza současné nabídky aplikací

Jak již bylo zmíněno v úvodní kapitole, cílem práce je navržení a vytvoření aplikace pro snadné týmové plánování, proto tuto kapitolu věnujeme analýze stávající nabídky podobných aplikací. Těch existuje v současné době na trhu mnoho a lze je dělit podle nejrůznějších kritérii. V následujícím textu uvádíme jejich členění podle tří nejčastěji zohledňovaných kritérií – členění podle finančního hlediska, podle potřeby instalace a podle technické stránky.

## 1.1 Členění podle finančního hlediska

- *Aplikace volně šiřitelné bez záštity velké komunity či korporace*

Takovéto aplikace jsou šiřitelné pod „volnými“ licencemi. Jedná se například o: BSD (Open Source I., 2013a), GNU GPL (Open Source I., 2013b), MIT (Open Source I., 2013c), a další. Uvedené aplikace nezřídka kdy nabízejí zajímavé funkce, objem těchto funkcí je však bohužel značně omezen. Také jejich vývoj či pomoc uživatelům při problémech s nimi, bývá omezená či dokonce žádná. Díky „otevřenému kódu“ je možné aplikace doplnit o potřebnou funkcionalitu, ale při potřebě složitějších funkcí mnohdy narážíme na problém nevyhovujícího návrhu datových struktur.

- *Aplikace volně šiřitelné se záštitou velké komunity či korporace*

Tyto aplikace bývají většinou rozsáhlejší, s dostatečnou komunitní podporou. Bohužel ve většině případů nejsou zdrojové kódy dostatečně popsané, proto jejich rozsáhlost neumožňuje snadné doplnění specifických funkcí. Některé pak navíc dávají přednost vzhledu (vizuální stránce) před funkčností.

- *Aplikace placené měsíčně*

U takovýchto nabídek se většinou uživatel nemusí zabývat technickou stránkou (tedy instalaci, hostingem, apod.) ani žádnými reklamami.

- *Aplikace placené za zřízení a udržování*

Tento způsob platby využívají velké projekty, které nabízejí mnoho funkcí se stálou podporou. Cena může být poměrně vysoká a rovněž bývá závislá na počtu uživatelů a zakoupených funkcích.

## 1.2 Členění podle potřeby instalace

- *Aplikace, které se nemusejí instalovat (webové aplikace)*

Takové aplikace mívají většinou pomalejší odezvu než aplikace desktopové. Řada z nich nevyužívá kódované spojení, a z tohoto důvodu je nelze považovat ani za zcela bezpečné. Pro práci s těmito aplikacemi je nutné být připojen na internet. Toto však nemusí být pouze nevýhoda, neboť k přístupu nám stačí jakékoliv zařízení s připojením na internet, které podporuje dané využité standardy a protokoly.

- *Aplikace, které se instalují jako pluginy do jiných aplikací*

Tyto aplikace mají výhody i nevýhody desktopových aplikací. Pokud na zařízení není nainstalována či nejde nainstalovat aplikace, nemáme možnost s aplikací pracovat. Uvedené aplikace pracují s rychlejší odezvou než webové aplikace. Lze zde většinou pracovat i v režimu „offline“. Jejich výhoda spočívá v tom, že pluginy se nemusejí zabývat platformou, jejich rozdílnost většinou řeší aplikace, do které se instalují.

- *Aplikace, které se instaluji jako samostatná desktopová aplikace*

Uvedený způsob instalace využívají velké a rozsáhlé balíky sdružující více aplikací dohromady. Některé jsou závislé na platformě a nejsou přenositelné. Pokud aplikace nemá jinou možnost přístupu k datům, potom je velmi pravděpodobné, že aplikaci nebude možno využívat na menších mobilních zařízeních.

- *Aplikace, na nichž si uživatel instaluje pouze klienta této aplikace (tzv. server-client aplikace)*

Výhod tímto způsobem zpracovaných aplikací – při dostupnosti odpovídajících klientů pro různá zařízení – je mnoho. Takovéto aplikace totiž mohou kombinovat klienta ve webovém prohlížeči, stejně tak, jako i desktopovou aplikaci, nebo v současné době velmi rozšířené mobilní aplikace. Zde pak hraje roli spíše otázka obtížnosti a nákladnosti vývoje takovéto aplikace.

## 1.3 Členění podle technické stránky (instalace, hosting, databáze)

Toto členění je velmi významné, neboť v plánování sdělujeme informace, jak o vlastní osobě, tak o práci, na níž pracujeme. Jde o takové informace, které by se

neměly dostat na veřejnost. Některé tyto informace mohou mít dokonce charakter obchodního tajemství.

- *Vlastní instalace, správa databáze a hardwaru*

Tato varianta nám dává přehled o tom, kdo k datům muže přistupovat a rovněž, co se s nimi děje. Nevýhoda spočívá naopak v tom, že potřebujeme vlastní hardware a taktéž kvalifikovaného správce pro jeho správu.

- *Vlastní instalace na cizím hostingu*

Nevýhoda tohoto způsobu spočívá v tom, že nemáme přehled o osobách, jež mohou přistupovat k našim datům, ani o samotném fyzickém zabezpečení hardwaru.

- *Pouhé využívání služby*

Společností, zabývajících se poskytováním webových aplikaci pro plánování, existuje mnoho. V případě rozhodnutí se pro tuto variantu, je dobré si přečíst licenční ujednání. V licenčních ujednáních je obsažena část věnující se ochraně poskytnutých dat. Je důležité věnovat zvýšenou pozornost zejména těm ujednáním, u nichž si poskytovatel vyhrazuje právo poskytnout informace třetí straně, případně tam, kde poskytovatel nenese žádnou odpovědnost za poskytnutá data. Oba zmíněné typy ujednání nenutí poskytovatele činit kroky vedoucí k ochraně dat. Co je zde však překvapující, je skutečnost, že podobná ujednání lze vidět i u placených služeb.

# 2 Analýza konkrétních typů softwarů

V této kapitole navážeme na předchozí část práce a naši analýzu zaměříme již na konkrétní existující typy aplikací, jež se v současné době vyskytují na trhu. Přehled, který zde uvedeme, bude zahrnovat aplikace, jež jsou dle názoru autora práce určitým způsobem zajímavé či něčím výjimečné. Jedná se jmenovitě o tyto aplikace: Google Calendar, PHP iCalendar, phpScheduleIt, Smart PHP Calendar, Microsoft Office Outlook, Lightning pro Thunderbird, IBM Lotus. Uvedené aplikace budou dále analyzovány dle následujícího klíče: (i) volně dostupné webové aplikace, (ii) placené webové aplikace a (iii) desktopové aplikace. Na následujících řádcích práce se tedy nejprve pojďme věnovat webovým aplikacím, jež jsou uživatelům volně k dispozici.

## 2.1 Volně dostupné webové aplikace

- *Google Calendar* (Google, 2013):

Google Calendar ztělesňuje snad nejrozšířenější aplikaci kalendáře vůbec. Rovněž představuje jeden z uživatelsky nejpřívětivějších software tohoto druhu, v tomto směru za sebou nechává dokonce i mnoho desktopových aplikací. Aplikace Google Calendar je určena spíše k soukromým účelům, z tohoto důvodu také nedisponuje větší podporou týmového plánování. Dostatečnou podporu týmového plánování lze zajistit, avšak pouze za cenu ztráty soukromí uživatelů, či při nedostatečném množství informací pro komplexní týmové plánování.

- *Další webové kalendáře – např. PHP iCalendar* (PHP iCalendar, 2013), *phpScheduleIt* (phpScheduleIt, 2013):

Aplikace typu PHP iCalendar či phpSchedulelt představují volně dostupné aplikace nevyznačující se něčím příliš zajímavým, toto konstatování platí počínaje ukládáním dat až po způsob a kvalitu zobrazení. Dle názoru autora práce jsou uvedené aplikace vhodné spíše do webové prezentace, k prezentování událostí nežli k samotné práci s nimi. Některé aplikace uvedeného typu různým způsobem využívají protokol CalDEV, specifikovaný v RFC 4791 (Daboo, et al., 2007), který bude podrobněji rozebrán dále v textu.

## 2.2 Placené webové aplikace (poskytující službu)

Tuto skupinu aplikací je možné ještě dále rozčlenit do dvou kategorií, první

z nich jsou velké kancelářské balíky s možností kalendáře a týmového plánování. Z této skupiny nemůžeme popsat mnoho zajímavých aplikací, neboť u řady z nich jsou demo účty pouze na vyžádání, přičemž většina projevila nechuť poskytnout tyto účty ke studijním účelům. Druhou kategorii pokrývají aplikace, které se v mnohém neliší od volně dostupných. Z tohoto důvodu v této kapitole uvedeme pouze jeden konkrétní produkt, a sice aplikaci Smart PHP Calendar.

- *Smart PHP Calendar* ( Smart PHP Calendar, 2013)

Tento projekt je zajímavý přímou podporou týmového plánovaní. Bohužel jde směrem „direktivního“ plánování a malé granularity oprávnění v týmech. Implementuje zajímavé funkce a je uživatelsky přívětivý, bohužel tato aplikace trpí na cross-site scripting[1].

## 2.3 Dostupné desktopové aplikace

- *Microsoft Office Outlook* (Microsoft Office, 2013):

Microsoft Office Outlook je desktopová aplikace přímo dostupná pouze pro operační systémy Microsoft. Aplikace zahrnuje mailového klienta, kalendář a „úkolníček“. Při použití Microsoft Exchange server podporuje týmové plánování i publikaci volného času.

- *Lightning pro Thunderbird* (Lightning, 2013):

V tomto případě se jedná o plugin do mailového klienta Thunderbird, který je pod MPL/GPL/LGPL licencí, což znamená, že je modifikovatelný a zdrojové kódy jsou dostupné, v jeho neprospěch pro naše využití však hovoří fakt, že nedisponuje podporou týmového plánování. Tu je možné simulovat povolením přístupu do jednotlivých kalendářů (vrstev). Je dostupný pro všechny využívanější operační systémy.

- *IBM Lotus* (Lotus Softwere, 2013):

IBM Lotus představuje placený, dalo by se říci, že až „mamutí“ balík kancelářských aplikací, zahrnující v sobě kalendář, “úkolníček“, adresář osob, mailového klienta, tabulkový a textový procesor. IBM poskytuje trial verze svých produktů pro všechny platformy, je možné instalovat i jednotlivé komponenty. Každou nedoinstalovanou komponentou ztrácí produkt jako celek funkcionalitu, kterou měla

[1] Cross-site scripting představuje určitý způsob narušení.

tato komponenta. Dle našeho hodnocení se jedná o rozsáhlý, kvalitně zpracovaný a smyslně provázaný balík aplikací podporující týmové plánování na vysoké úrovni.

## 2.4 Shrnutí

Jak vyplývá z výše uvedeného přehledu, v současné době na trhu existuje řada plánovacích aplikací, které se svého úkolu zhostily s různou měrou úspěšnosti. Některé z nich implementují zajímavé a unikátní funkce, ať pro přístup k týmovému plánování nebo pro přístup k propojení úkolů a událostí. Implementováním všech těchto funkcí a dodáním nových by mohl vzniknout optimální plánovací software. Charakteristice takového softwaru je věnována následující kapitola práce.

# 3 Ideální plánovací software

Předchozí dvě kapitoly nám poskytly přehled o aplikacích, jež jsou v současné době uživatelům reálně k dispozici. Získali jsme z nich podstatné informace o funkčnosti těchto aplikací a seznámili jsme se rovněž s jejich hlavními přednostmi a nedostatky. V závěru předchozí kapitoly jsme učinili dílčí závěr, že v podobě, v jaké se tyto aplikace nyní nacházejí, neodpovídají našim potřebám. V této kapitole si proto vymezíme takzvanou ideální plánovací aplikaci. Jaké charakteristiky a atributy by tedy taková aplikace měla mít? Odpovědi poskytuje následující text.

## 3.1 Uživatelská přívětivost

O tom, že každá aplikace by měla být uživatelsky přívětivá, snad není pochyb, můžeme snad říci, že v této otázce panuje mezi laickou veřejností i akademickou a odbornou sférou obecná názorová shoda. Pokud se ale jedná o aplikaci, kterou by měl uživatel využívat denně, pak jsou na ni přirozeně kladeny i další nároky. V takovém případě bude vhodné, aby byla rovněž přehledná, snadno se ovládala a současně byla srozumitelně konfigurovatelná. Pro plné a zároveň kvalitní využití plánovací aplikace se předpokládá, že ji bude uživatel navštěvovat mnohokrát denně a konzultovat s aplikací stávající i budoucí časové možnosti. Z těchto důvodů jsou v aplikaci žádoucí jisté atributy. Aplikace by měla být:

- interaktivní s krátkými prodlevami zapříčiněnými načítáním dat,
- graficky přehledná,
- využívající metody drag&drop k editaci událostí,
- každý uživatel by měl mít možnost nastavit grafiku aplikace tak, aby jemu osobně vyhovovala,
- hlavní „řídící“ panely by měly být snadno přístupné z jakéhokoli stavu aplikace,
- vybavena různým vzhledem dialogů, aby uživatel snadno rozlišil konkrétní dialog,
- měla by mít různý vzhled potvrzovacích dialogů, aby uživatel snáze rozlišil „důležité“ a „nedůležité“ potvrzení. Míra důležitosti se odvíjí od významu položky, kterou potvrzuje a zda manipulace s danou položkou neovlivní i jiné uživatele,
- každý uživatel by měl mít po přihlášení do aplikace stejně nastavené prostředí,

jako když aplikaci opouštěl,

- snadné uživatelské vyhledávání a filtrování případných rozsáhlejších dat.

## 3.2 Bezpečnost a soukromí

Aplikace, jíž svěřujeme své osobní údaje i údaje našich obchodních partnerů či kolegů, by měla být bezpečná proti všem možným útokům, tedy jak z fyzického světa, tak i z toho binárního. Z tohoto důvodu je žádoucí, aby komunikace probíhala v šifrované podobě. Přihlašování k jednotlivým účtům by mělo být realizováno přes bezpečné heslo, či jiný bezpečnostní prvek, který dokáže bezpečně rozeznat majitele účtu. K bezpečnosti neodmyslitelně patří i zachování soukromí. Pokud chceme uživateli nabídnout aplikaci takovou, kterou bude využívat v osobní i pracovní části svého života, potom by tato aplikace měla umět následující:

- zabezpečit v ní obsažená fyzická data proti ztrátě a odcizení;

  - Ke ztrátě dat dochází především poškozením datového nosiče, na kterém jsou data uložena, proto by aplikace měla umožňovat snadnou duplikaci dat a snadné zpětné nahrání, případně on-line replikaci.

  - Proti fyzickému odcizení se můžeme bránit ukládáním dat na bezpečné úložiště s použitím dostatečného šifrování uložených dat.

- zabezpečit data ve zpracování. Ideální plánovací aplikace bude s největší pravděpodobností aplikací využívající vícevrstvé architektury, proto by měla být zajištěna spolehlivá komunikace mezi jednotlivými vrstvami a zabezpečení samostatných vrstev;

- přes bezpečnostní prvek (heslo, čipová karta atd.) jistě rozeznat unikátního uživatele. Při opakovaném pokusu o neoprávněné přihlášení informovat o této skutečnosti zodpovědné osoby, či zablokovat účet do doby vyřešení takové situace;

- komunikace by měla být šifrována, aby ji nebylo snadné odposlouchávat;

- automatické odhlášení déle neaktivních uživatelů. Automatické odhlašování by nemělo příliš obtěžovat uživatele, proto je nutné určit interval, kdy se jedná o vyžádanou přestávku interakce mezi aplikaci a uživatelem, a kdy už se jedná o bezpečnostní riziko. Při automatickém odhlášení a zpětném přihlášení uživatele by se neměla znehodnotit práce, na níž uživatel před automatickým odhlášením

pracoval (např. z některých internetových mail klientů jsou známé případy, kdy předtím, než uživatel dopsal rozsáhlý e-mail, byl automaticky odhlášen, rozepsaný e-mail při odhlášení ale nebyl uložen);

- zachování soukromí by mělo být realizováno přístupem „raději méně než více“. Jedná se o přístup, kdy budou sdělovány jen nezbytné informace pro potřebný chod ostatních částí aplikace, pokud uživatel nenastaví, co vše může být zveřejněno a komu.

## 3.3 Výchozí předpoklady

Abychom mohli vytvořit aplikaci, kterou lidé budou používat k plánování, je třeba uvažovat, co všechno lidé používají, když plánují, a to proto, abychom všechny tyto prostředky mohli implementovat do jedné optimální aplikace (tzv. ideální plánovací aplikace). Za tímto účelem provedeme na následujících řádcích stručný exkurz do historie toho, jakým způsobem lidé dříve organizovali svůj čas.

V dobách před příchodem moderních technologii byl k plánování využíván nástěnný roční kalendář. Pro kvantitativní plánování byla granularita takového kalendáře nedostačující, proto se přešlo na kalendáře měsíční, později týdenní a ještě později na malé diáře, kde byly rozepsány jednotlivé dny, a to dokonce po hodinách. Pro člověka je důležité mít diář vždy při ruce, aby bylo možné si do něj zapsat novou událost nebo úkol. Dnešní diáře obsahují i kolonky na telefonní čísla, e-maily a adresy či disponují volným místem pro vepsání potřebné poznámky. Je to pochopitelné, pokud plánujeme událost, plánujeme ji zpravidla s další osobou, proto na ni potřebujeme mít také určitý (zpravidla telefonický či mailový) kontakt. Institut papírového diáře, jak je patrné z pultů obchodů s papírnictvím, prověřila doba a ukazuje se, že úspěšně funguje mnoho let, zdá se tedy vhodné převést jej určitým způsobem do elektronické podoby.

Korespondence představovala před mnoha lety základní kámen komunikace. Jak se náš život stává čím dál tím rychlejší, pošta se stává čím dál tím pomalejší. Také s příchodem nových technologii umožňujících ověřovat originalitu e-mailu, se stává e-mail jednou z hlavních komunikačních cest. Mnohé schůzky a jednání jsou dnes dojednávány prostřednictvím e-mailu, proto se dá říci, že lidé e-mail využívají k plánování. Chceme-li využít adresář osob a e-mail, jednoduché provázání těchto částí se již nabízí. Máme také část s kalendářem, zde je provázání obtížnější, ale část konkrétní komunikace probíhá za účelem specifikovat podrobnosti úkolu nebo události. Proč tedy nemoci přiřazovat konkrétní e-maily ke konkrétním akcím, ke

kterým patří? Při komunikaci s rozsáhlejší institucí můžeme jednat s více lidmi, kteří s námi řeší stejný problém, bylo by proto žádoucí mít možnost vybrat e-maily pouze z konkrétní instituce.

Považujeme-li e-mail za hlavní komunikační kanál, potom VOIP a IM musíme považovat za vedlejší komunikační kanály. Máme prostor pro zapojení i těchto vedlejších komunikačních cest. Jedním kliknutím na telefon v kartě adresáře se okamžitě začne vytáčet dané telefonní číslo na VOIP telefonu. Tento telefonát by mohlo být možné i zaznamenávat. Minimálně by bylo možno zaznamenat, že dané osobě bylo voláno a uživatel by mohl textově zadat důvod takového telefonátu a připojit ho k události nebo úkolu, kvůli kterému byl telefonát veden. Podobné možnosti by mohly být provedeny i u IM, kde ukládání rozhovorů je běžnou praxi u většiny klientů IM.

Naplánováním a zaznamenáním události by práce plánovacího softwaru ovšem rozhodně končit neměla. Později můžeme mít potřebu zjišťovat, jestli jsme se dané události zúčastnili, a jak dopadla. Z některých typů události (jednání, schůze, konference) vznikne následně další materiál, jako je zápis, záznam či poznámky. Tento konkrétní materiál se vztahuje ke konkrétní události, kterou plánovací software naplánoval. Proč tedy nepřipojit materiál k této události, ale zakládat novou složku s těmito materiály, ať už digitální anebo fyzickou? Ukládáme-li výstupní materiály z akcí, které jsme naplánovali, nebylo by vhodné ukládat i vstupní materiály (těmito materiály jsou myšleny např. harmonogram akce, pozvánka na akci, prezenční listina účastníků atd.)? Uživatel tak bude mít větší přehled, jak o události samotné, tak o materiálech, které na danou událost vypracoval, vyhne se tak nebezpečí, že je zapomene v jiné složce, protože v plánovači je okamžitě uvidí.

Komunikaci, kterou popisujeme v předešlých dvou odstavcích, musíme s ohledem na bezpečnost svěřit pouze aplikacím a protokolům, kterým důvěřujeme nebo u nichž jsme schopni zajistit dostatečné šifrování přenášených zpráv. U e-mailu jsme tohoto schopni dosáhnout pomocí šifrování, pro IM se pak nabízí třeba otevřený protokol xmpp/jabber (XMPP, 2013), který umožňuje jak šifrování konkrétních zpráv, tak komunikaci pomocí SSH, případně je možné spravovat vlastní server.

Pokud ukládáme dokumenty, je zajímavé uvažovat o tom, jak se tyto dokumenty vytvoří (konkrétně zde máme na mysli textové editory). Pokud budeme pokračovat v přístupu, který jsme doposud pro definování pojmů používali, řekneme, že dokumenty patří k plánování, proto do naší aplikace přidáme i textový procesor. Takto postupují i

stávající velké aplikace. Dalo by se diskutovat, zda je to správně či nikoli. Protože textových procesorů je velké množství a formátů, ve kterých pracují, taktéž, a každý uživatel může využívat jiný, je zbytečné, aby plánovací aplikace v sobě implementovala textový procesor, může jej mít jako doplněk, ale funkce by na něm neměly být nikterak závislé. Řešení nám může přinést využití vhodného verzovacího nástroje.

Jak jsme uvedli již výše, plánování se provádí s ohledem (v závislosti) na jiné osoby. Primárně potřebujeme informace o tom, zda tyto osoby mají čas, či čím ho mají zaplněný. Některé skupiny kooperující spolu (pracovníci v zaměstnání, rodiny, aj.) používají jeden společný diář, kde zapisují všichni dohromady. Pokud ale do tohoto společného diáře nechtějí sdělit i osobní plány, musejí si vést i svůj soukromý osobní kalendář či zápisník. Nastává tak duplicita některých akcí, což může vést k nechuti uživatelů zapisovat dvě stejné akce do různých diářů (konec konců v dnešní době je čas stále vzácnějším statkem). To může mít pak i další následky, například v podobě zapomenutí zaznamenat určitou akci vůbec. Právě z tohoto důvodu jsou důležité *funkce podporující týmové plánování se zachováním soukromí*. Pokud tyto funkce budou implementovány, je možno používat odděleně svůj osobní plánovač, přičemž změny v něm se promítnou i do týmového plánovače. Pokud pracujeme s více týmy, je to velmi užitečná pomůcka, která eliminuje několikanásobné zaznamenávání stejné události. Můžeme tedy učinit závěr, že je vhodné implementovat týmové funkce a plánovače. V tu chvíli ale vyvstává otázka další, a sice, co by tyto funkce měly umět? *Hledání společného volného času* – tato funkce nalezne a uspořádá v relativním pořadí vhodné termíny, kdy mají členové týmu čas. Primární využití této funkce je na plánování schůzek a konzultací ve vyhovujícím čase všech zúčastněných. Plánování události, na kterou jsou zváni nebo o ní mají být informováni i ostatní členové týmu, funguje jako sdílení událostí. Na některé události (jednání, komise) je vhodné mít přesný počet účastníků, proto je vhodné zavést *sdílenou událost s potvrzením účasti*, kde správce této události může přihlášené uživatele odhlašovat, například jako nadbytečné.

Máme-li naplánované akce, nezřídka kdy zjišťujeme, že o některých z nich chceme mít pouze přehled, ale ve skutečnosti se jich nechceme účastnit, proto je dobré události určitým způsobem rozlišovat. Také je dobré znemožnit přidání akce, které se chceme zúčastnit, pokud by časově překrývala jinou událost, které se účastnit chceme.

Při využívání diáře a zaznamenávání mnoha akcí se snadno dostaneme do stavu, kdy zobrazení pro uživatele může být velmi nepřehledné. Ku pomoci by měly

v takovém případě nastoupit *vrstvy*. Určité akce mají společného jmenovatele a právě tyto akce lze spojit do jedné vrstvy, přičemž by mělo být možné vypínat zobrazení konkrétních vrstev. Vrstvy mohou být nastavené s určitým stupněm *zveřejnění*.

*Zveřejnění* – již dříve jsme diskutovali o určitém zachování soukromí, přesto některé akce chceme nebo musíme zveřejnit v určitém měřítku a určité skupině uživatelů (týmu, veřejnosti, vedoucím pracovníkům, atd.). Hledáme-li vhodný čas na schůzku s jinou osobou, která nebude používat naší aplikaci, můžeme jí poskytnout svůj kalendář, kde budou zvýrazněna pouze ta místa, přesněji řečeno, časové úseky, kde máme volno. Pokud jdeme na pracovní jednání, je vhodné, aby kolegové, sekretářka nebo vedoucí věděl/a, kde se právě nacházíme, na druhou stranu, nikdo nemusí vědět, že každý pátek chodíme hrát golf. Naopak přátelům, se kterými golf hrajeme, chceme sdělit, zda se daný týden zúčastníme, či nikoli, proto jim tuto událost zveřejníme.

Máme-li naplánováno a zveřejněno, pak by se vše mohlo zdát vyhovující, naše plánovací aplikace ale není jediná, kterou ostatní uživatelé mohou využívat. Jak jsme již mnohokrát v textu zmínili, aplikací je mnoho. Protože bychom chtěli mít interakci i s ostatními uživateli používajícími jiné plánovací aplikace, je vhodné přistoupit *k importům a exportům*. Podle možností druhé aplikace se může jednat o *interaktivní nebo statický import/export*.

Některá data, která bychom chtěli promítnout do své aplikace (kalendáře) nemusejí být dostupná ve formě kalendáře, ale mohou být dostupná v různých seznamech, proto je vhodné mít možnost zpracovávat a zobrazovat informace i z těchto seznamů. Seznam by měl být ve formátu umožňujícím takovýto přístup (jedná se např. o XML, nebo jiný přesně definovaný formát). Příkladem seznamu, který bychom mohli mít potřebu promítnout do naší aplikace, je třeba seznam státních svátků, harmonogram školního roku našeho potomka a podobně.

Uživatel chce mít aplikaci co nejjednodušší, a v našem životě děláme mnoho úkonů opakovaně, proto ulehčíme uživateli práci s opakovaným zadáváním dat a umožníme mu u akcí nastavovat *opakování* (denní, týdenní, měsíční, určitý den v týdnu, atd.). Některé akce se ale v určitých situacích nedějí (např. výplatu či důchod nedostaneme čtrnáctého, přestože je to standardně den vyplácení dané dávky, pokud by toto datum připadlo na víkend), proto je nutné k opakování událostí přiřadit, za jakých podmínek je opakování žádoucí. Místo podmínek můžeme importovat externí kalendáře s takto vyznačenými dny.

Za některými sjednanými událostmi také musíme cestovat. Cestování přitom zabírá určitou dobu. S tímto časem vyhrazeným na cestování tak při plánování musíme počítat. Proto je užitečné počítat s tímto časem i v aplikaci plánovače. Tento čas by se měl určovat pouze u událostí, u nichž se uživatel rozhodl jich zúčastnit. Aby aplikace byla co nejpřívětivější, stačí, když uživatel zadá místo konaní akce, popřípadě i místo, kde, se pravděpodobně bude nacházet bezprostředně předtím, než pojede na danou událost. Aplikace pak sama vyhodnotí dobu trvání cestování. K tomu bude ještě zapotřebí znát případný způsob dopravy (MHD, pěšky, autem). Aplikace, které vyhodnocují nejlepší cestu a dobu strávenou při jejím použití, již existují a mají svá rozhraní. Proto by plánovač mohl pouze využívat tato rozhraní.

Dříve již také bylo zmíněno, že některé události nekončí pouze jejich konáním. Především pokud organizujeme jednání, konferenci apod., potřebujeme si vyhradit určitý čas na přípravu (např. místa konání) a také čas po konci, na dokončení, zhodnocení a jiné.

Uživatel musí být informován o všem, co se s jeho plánovačem děje. Upozornění na nabídku událostí, upozornění, že nestihl udělat tento úkol, atd. *Upozornění* by měla být viditelná na první pohled, pokud se však nejedná o velmi důležitá upozornění, neměla by uživatele zbytečně obtěžovat.

Pokud nás omrzí plánování vlastními silami, ale přesto potřebujeme plánovat alespoň pouze svůj kalendář (nikoli týmový), můžeme dát jinému uživateli přístup do svého kalendáře. Tento druhý uživatel může mít nastavené různé možnosti zveřejnění (např. zveřejnění pouze pracovní doby, zveřejnění pouze pracovních vrstev) a také jen omezená *práva k editaci*. Tato možnost je vhodná, pokud potřebujme, aby nám rozvrh částečně mohla plánovat například asistentka či jiná k tomu povolaná osoba, ale současně přitom zůstalo zachováno naše soukromí.

Následující subkapitola poskytuje stručné shrnutí výše uvedeného textu, zpracované v přehledné bodové podobě.

## 3.4 Bodové shrnutí ideální plánovací aplikace

- Kalendář s režimem zobrazení: denní, týdenní, měsíční a možností grafického nebo textového režimu;
- „úkolníček“ s fulltextovým vyhledáváním a různými možnostmi řazení;
- adresář osob;

- poznámkový blok;
- propojení na bezpečné aplikace poskytující e-mail, VOIP a IM;
- události: informativní a s plánovanou účastí;
- více vrstev;
- ukládání dokumentů k akcím (úkol/událost);
- import a export kalendářů;
- možnost importovat a pracovat s externím kalendářem;
- opakující se události;
- plánování cestování;
- čas potřebný na přípravu a dokončení události;
- „uživatelsky přívětivé chování“;
- zveřejňovaní části kalendáře;
- týmové plánování;
    - plánování schůzek,
    - sdílení a nabízení události s možností správy těchto událostí,
    - plánování třetí osobou.

# 4 Technická realizace ideální plánovací aplikace

Po určení toho, co by měla ideální plánovací aplikace umět (viz předchozí kapitola), je třeba nyní uvažovat o způsobech technické realizace. Jimi se zabývá tato kapitola.

## 4.1 Volba síťové architektury

Pokud chceme podporovat týmové plánování, je nutné nějak komunikovat s více přístroji.

Nabízí se možnost Client-queue-client. V tom případě jsou funkce serveru omezeny na pasivní předávání zpráv a dat a klienti neplní roli serveru, ale implementují snadnější komunikační protokol. Architektura sestává ze dvou či více klientů, kteří si vyměňují data a zprávy přes nezávislý třetí uzel, tzv. pasivní frontu (EXFORSYS, 2007). Takto jsou implementovány některé stávající kalendáře, které využívají e-mailové komunikace k přenosu dat. Takovýto způsob má i své výhody, pro naši aplikaci je však nevhodný z důvodu obtížného nelezení dat pro některé funkce týmového plánování, jako je vyhledávání schůzky týmu a velká komunikace při práci s vrstvou, kterou může vidět více uživatelů.

Druhá možnost je architektura client-server. Při jejím využití každá instance klienta může posílat žádost o data jednomu nebo více připojeným serverům. Na druhé straně, servery mohou akceptovat tyto žádosti, zpracovat je a vrátit klientovi požadovanou informaci. Tento koncept může být použit více různými způsoby, avšak základ zůstává v zásadě stejný (EXFORSYS, 2007). Výhodou této architektury pro naši aplikaci je především dostupnost všech potřebných dat na jednom místě. Tato výhoda se ale také může snadno stát nevýhodou, a to v případě velkého množství dat. Obecné nevýhody této architektury, jako je centralizace a možnost přetížení, naše aplikace žádným způsobem nemůže ovlivnit. Tyto nevýhody se dají zmírnit vhodným rozdělením na více fyzických strojů či linek.

Z tohoto důvodu bychom tedy měli zvolit architekturu client-server. Vhodnost této volby dokazuje i skutečnost, že i většina masově využívaných plánovacích aplikací volí právě tuto architekturu.

## 4.2 Aplikační architektura

Aplikace muže být členěna na vrstvy, kde každá vrstva zajišťuje specifickou funkcionalitu. Vrstvy spolu spolupracují přes definovaná rozhraní. Proto je možné implementaci jedné vrstvy nahradit jinou implementací téže vrstvy beze změn ostatních vrstev. Většina moderních aplikací používá architekturu tvořenou třemi vrstvami. Jedná se o tyto vrstvy:

- prezentační vrstva (User interface) – uživatelské rozhraní a jeho logika (grafické zpracování dat, možná kontrola vstupního interfacu);
- aplikační vrstva (Business Logic) – aplikační logika, zde se provádějí výpočty a zpracování požadavků od prezentační vrstvy;
- datová vrstva (Data Access) – vrstva uchovávající a zpřístupňující data, je žádoucí, aby zajišťovala jejich konzistenci.

## 4.3 Datová vrstva

Data chceme bezpečně uchovávat a snadno s nimi pracovat. Existuje více způsobů, jak toho dosáhnout. Problematika ukládání dat je však natolik rozsáhlá, že by vydala na samostatnou bakalářskou práci, předmětem předkládané bakalářské práce je však řešení jiného úkolu, proto problematika bezpečného ukládání dat je nad rámec naší práce a nebude zde dále podrobně rozebírána. Pro naše účely se tedy omezíme na pouhé konstatování, že pro ideální plánovací aplikaci je žádoucí rychlý přístup k datům dle klíče a rozsahu klíče. Se splněním podmínek pro soukromí a bezpečnost popisovaných v subkapitole 3.2.

## 4.4 Aplikační vrstva

Prezentační vrstva naší aplikace by měla umět zpracovávat požadavky předávané na server od klienta a opačně. Také by měla být schopna získávat data z datové vrstvy. Nyní si musíme položit otázku, zda logiku aplikace budeme mít v prezentační vrstvě nebo v datové vrstvě (za předpokladu vhodného nástroje v datové vrstvě). Je zřejmé, že aplikace bude dělat funkce nad kolekcí dat, která budou v datové vrstvě uloženy.

Odpověď na uvedenou otázku není snadná. Zde záleží už na konkrétní implementaci použitého nástroje pro prezentační vrstvu a v neposlední řadě i nástroje pro datovou vrstvu.

## 4.5 Prezentační vrstva

U klient-server aplikací je možno mít více klientů (resp. prezentačních vrstev). Zde záleží na konkrétním typu zařízení a způsobu komunikace se serverem. Musí se počítat s různými omezeními konkrétního zařízení (např. malá paměť, malý výkon, pomalá přenosová rychlost, apod.).

Pro naši ideální plánovací aplikaci by bylo nejlepší mít pro každý typ zařízení svého klienta. Tento požadavek se v reálném světě splňuje jen velmi obtížně. Proto by bylo vhodné mít alespoň dva druhy klientů. A sice tyto:

- *tlustý* – na straně klienta se vykonává větší část aplikační logiky, to vyžaduje větší hardwarové i softwarové nároky na stranu klienta, ale naopak menší nároky na stranu serveru a především komunikace s ním;
- *tenký* – na straně klienta dochází jen k zobrazování dat, je zde minimální logika, ta se všechna řeší na serveru. Minimální nároky na klienta, velké nároky na server.

## 4.6 Bezpečnost

V naší ideální plánovací aplikaci hraje bezpečnost význačnou roli. Důležitá je především bezpečnost komunikace mezi jednotlivými součástmi aplikace. Nesmí se ale opomenout také bezpečnost fyzická (tzn. serverové části), a to, jak před úmyslným poškozením, tak proti poškození náhodnému. Diskutovat možnosti fyzického zabezpečení serverové části je nad rámec této práce, proto se na místo toho pojďme věnovat bezpečnosti komunikace jednotlivých komponent aplikace.

Jak už jsme řekli dříve, komunikace by měla probíhat v šifrované podobě. Jednou z využívaných metod je použití protokolu SSL – tedy Secure Sockets Layer. To je protokol, respektive vrstva, vložená mezi vrstvu transportní a aplikační, která poskytuje zabezpečení komunikace šifrováním a autentizací komunikujících stran (Oracle, 2013).

SSL se dá využít v těchto dvou stavech (Oracle, 2013):

- **jednosměrná SSL autentizace** (one-way SSL authentication) – umožňuje SSL klientovi ověřit identitu SSL serveru, ale SSL serveru neumožňuje ověřit identitu SSL klienta. Tento způsob SSL autentizace je využíván při komunikaci prostřednictvím protokolu HTTPS. Koncový uživatel svoji identitu serveru potvrzuje až na aplikační vrstvě;

- **Dvousměrná SSL autentizace** (two-way SSL authentication nebo mutual SSL authentication) – umožňuje SSL klientovi ověřit identitu SSL serveru a zároveň SSL serveru ověřit identitu SSL klienta. Tento typ autentizace se nazývá klientskou autentizací, protože SSL klient při něm prokazuje svoji identitu SSL serveru klientským certifikátem. Autentizace klientským certifikátem může vhodně doplnit nebo nahradit klasické metody autentizace (zadávání hesla a jména) ( Oracle, 2013).

Naše aplikace by mohla využívat SSL šifrování, protože je rozšířené a obecně považované za bezpečné.

## 4.7 Datové struktury a standardy

Jak bylo psáno již na začátku této práce, aplikací poskytujících různé metody plánování a různě zpracované kalendáře, je dnes mnoho. Proto vznikl standard iCalendar **(**Internet Calendaring and Scheduling Core Object Specification). Specifikován byl v RFC 2445 (Dawson, Stenerson, 1998) a RFC 5545 (Desruisseaux, 2009) (jedná se pouze o specifikaci datového formátu, nikoli o popis práce s ním – pozn. autora). Práci s daty ve formátu iCalendar, popisují následující protokoly:

- iTIP (iCalendar Transport-Independent Interoperability) specifikován v RFC 2664 (Plzak, et al., 1999);
- IMIP (iCalendar Message-based Interoperability Protocol) specifikován v RFC 2447 (Dawson, et al., 1998);
- CAP (Calendar Access Protocol) specifikovan v RFC 4324 (Royer, et al., 2005);
- Guide to Internet Calendaring v RFC 3283 (Mahoney, et al., 2002). Jedná se o standard, který specifikuje vztahy mezi formátem iCalendar a jinými standardy.

Na standard iCalendar navazuje protokol CalDAV (Calendaring Extensions to WebDAV) specifikovaný v RCF 4791 (Daboo, et al., 2007), kde je specifikována práce s iCalendar přes http a DAV protokol, za využití iTIP. Je napsáno mnoho serverů pro protokol CalDAV, například DAViCal (viz: http://www.davical.org/index.php), a tento přístup ke kalendáři má implementováno velké množství klientů. Za zmínku stoji Google Calendar čí Lightning.[2]

Výše zmíněné formáty a protokoly neumožňují snadnou implementaci systému oprávnění, kterou požadujeme, z tohoto důvodu nejsou pro naše potřeby zcela vhodné.

2 Částečný přehled je možné nalézt na: http://wiki.davical.org/w/CalDAV_Clients.

Jejich přehled zde uvádíme pouze pro získání uceleného a komplexního přehledu.

# 5 Implementační rozhodnutí

Vyvinout ideální plánovací aplikaci je dlouhodobý úkol, proto přistoupíme k vývoji určité její části s možností pozdějšího rozšiřování. Budeme brát na zřetel účel práce, kterým je vyzkoušet navržený systém práv. Na začátku provedeme některá rozhodnutí, která ovlivní způsob implementace a možnosti dalšího rozšíření.

Prvním takovým rozhodnutím je volba architektury klient-server. Výhodnost dané architektury pro naši aplikaci jsme již vysvětlili, proto se zde odkazujeme pouze na výše uvedený text.

Druhým rozhodnutím je volba klienta. Jako klienta zvolíme grafický webový prohlížeč s podporou JavaScriptu. Výhodou této volby je dostupnost grafického webového prohlížeče na většině systémů. Zobrazení v grafickém webovém prohlížeči podléhá standardům W3C (W3C, 2011), proto by zobrazení naší aplikace mělo být ve všech typech stejné. Toto rozhodnutí nám umožní splnit požadavek na interaktivní uživatelské rozhraní, který byl popsán v kapitole o uživatelské přívětivosti ideální plánovací aplikace.

Třetím rozhodnutím, které je třeba učinit, je formát komunikace mezi klientem a serverem. Zvolíme JSON (JavaScript Object Notation), specifikovaný v RFC 4627 (Crockford, 2006). Protože je tento formát velmi dobře zpracováván JavaScriptem, je považován za nezávislý datový formát a je podporován v mnoha jiných programovacích jazycích. Výběrem tohoto nezávislého datového formátu umožníme tvorbu jazykově nezávislých klientů.

Čtvrtým rozhodnutím je volba serverové části. Zvolíme server Apache, doplněný o PHP verze 5.5 pro práci s datovou vrstvou, která bude reprezentována relační databází PostgresSql verze 9.1 Tuto volbu jsme provedli z důvodu vhodnosti pro vývoj webových aplikací, kterou náš klient je. Toto ale neznamená, že tyto prostředky jsou nevhodné i pro jiné zpracování klientů. Druhým důvodem pro tuto volbu byla dostupnost daných prostředků a snadno čitelných zdrojových kódů v daných jazycích, což je pro ukázkovou implementaci důležité.

# 6 Návrh přístupu k právům

Doposud jsme popisovali pouze to, jak by měla vypadat ideální plánovací aplikace, a provedli jsme stručný přehled dostupných software. Ty většinou vycházejí ze standardů a protokolů podobněji popisovaných v subkapitole 4.7, proto nesplňují všechna naše očekávání, především s ohledem na týmové plánování. U tohoto plánování je třeba klást větší důraz na granularitu oprávnění. Proto se v naší aplikaci pokusíme navrhnout a otestovat jiný přístup k oprávněním a sdílení.

To, co chceme, je, aby každý uživatel mohl své i sdílené události dále sdílet, maximálně však v rozsahu viditelnosti (viz dále), která mu přísluší. V tomto „volném“ přístupu se odlišujeme od direktivního přístupu, jež můžeme vidět například u Smart PHP Calendar (Smart PHP Calendar, 2013), uvedeného v analýze současných aplikací v kapitole 1. Proto nosný prvek sdílení v naší aplikaci bude *entita události*, nikoli entita vrstvy, jak je tomu u současných aplikací. Sdílení událostí umožníme ve stavech: *"Editace", "Vidět vše", "Vidět název", "Vidět obsazenost"*. Každý z těchto stavů má přirozeně přístup k začátku l konci události.

Také chceme, aby uživatel mohl některá data události měnit, zejména nastavení účasti, abychom docílili stavu, kdy pro jednoho uživatele má událost pouze informativní charakter a druhý se jí chce reálně účastnit (např. odevzdávání bakalářských a diplomových prací je důležité zvláště pro studenty posledních ročníků příslušného stupně studia, kteří se ho budou účastnit, studenty ostatních ročníků by tato informace však mohla také zajímat například proto, aby v daný den zbytečně nechodili na studijní oddělení, kde bude pravděpodobně fronta). Zde nastává otázka, jak se mají nasdílené události měnit v případě jejich editace. Můžeme zvolit direktivní přístup, při němž událost direktivně upravíme všem uživatelům, kteří jí sdílejí. Jako vhodnější přístup se nám však jeví, ponechat událost tak, jak je a informovat o změně uživatele, ať si událost případně upraví sám, případně mu poskytnout nástroj pro synchronizaci.

Je vhodné sdílet i jednotlivé vrstvy. To umožňujeme v těchto režimech: *"Upravovat vlastnosti", "Přidávat události", "Pouze nahlížet"*.

Zde vyvstává podobná otázka, jako v předchozím případě, tedy, jak se chovat v případě editace. Pokusíme se opět ponechat volnost uživateli. To ale povede k neintuitivnímu chování z pohledu „správce“ vrstvy. Protože pokud odebereme sdílení vrstvy uživateli, lze předpokládat, že ztratí všechny události v této vrstvě. To se ale

nestane, a to z důvodu obrany uživatele, který tuto vrstvu sdílí, mohl mít v dané vrstvě události, kterých se musí zúčastnit a jejich odebráním by mohl přijít o důležité informace. Protože *události jsou nositeli sdílení*, můžeme na vrstvu nahlížet pouze jako na kolekci se společným jmenovatelem.

Budeme tedy v naší aplikaci pracovat s pojetím vrstvy, které se zásadně liší od ostatních aplikací. Tradiční pojetí vrstev staví na standardech popsaných dříve v textu (viz kapitola 4.7), kde se nejedná o vrstvy jednoho kalendáře, ale o samostatné kalendáře, které jsou pouze společně prezentované.

Pokud se s vrstvami pracuje jako s jednotlivými kalendáři, není možná kontrola „volného termínu“. Také, pokud dochází ke sdílení jednoho tohoto kalendáře (vrstvy), dochází ke sdílení všech událostí. To také není vždy žádoucí, protože se přenášejí všechny údaje, dokonce i o účasti. To je jeden z výše popsaných nedostatků, ale také to je nevhodné pro sloučení kalendářů více oddělených skupin (např. pracovního a rodinného). Ale, pokud nositelem informací o sdílení je samotná událost a na vrstvu se nahlíží pouze jako na kolekci událostí se společným jmenovatelem, můžeme s vrstvami pracovat jako s vrstvami jedné instance kalendáře. To nám umožní přesně to, co potřebujeme, tedy ve vrstvě členit události se společnými vlastnostmi. Tyto vlastnosti poté mohou být předvyplňovány k událostem vrstvy, ale událost není na těchto pravidlech závislá a může si nastavit vlastní.

# 7 Logika aplikace

Celou aplikaci jsme navrhli tak, aby hlavní datové entity z aplikační vrstvy odpovídaly stejným entitám z vrstvy prezentační (jedná se o entity vrstvy a události). Takto navržené entity se nám budou dobře kontrolovat a rozšiřovat.

## 7.1 Datová vrstva

Reprezentovaná relační databází PostgresSql, zde představuje pouze prosté úložiště, konkrétní schéma nalezneme v příloze č. 3. Logika nad daty je zpracována v aplikační vrstvě. Tímto rozhodnutím částečně ztrácíme implementační nezávislost datové vrstvy, kterou nám ale balancuje použití aplikační databázové vrstvy (dibi (NETTE, 2013a)). Autor k tomuto rozhodnutí dospěl po neúspěšné implementaci plánovací aplikace, kdy větší část logiky celé aplikace byla právě převedena do datové vrstvy. V takovéto implementaci je velmi obtížné zapracovávat funkcionalitu, se kterou se v návrhu nepočítalo, nebo se opomněla. Při takovémto použití se nabízí otázka přínosu relační databáze. S datovou vrstvou komunikuje aplikační vrstva přes rozhraní aplikační databázové vrstvy pomocí přímého volání SQL dotazů.

## 7.2 Aplikační vrstva

Aplikační vrstva reprezentovaná webovým serverem, komunikuje s prezentační vrstvou pomocí synchronního volání AJAX (asynchronou JavaScript and XML) a data jsou přenášena ve formátu JSON (JavaScript Object Notation).

Komunikace probíhá v modelu:

- prezentační vrstva pošle požadavek ve formátu JSON, s hodnotou operation (specifikující konkrétní požadavek) dále pak hodnoty, které jsou nutné ke zpracování daného požadavku,
- aplikační vrstva obdrží daný požadavek, zpracuje ho a opět klientovi odesílá řetězec ve formátu JSON,
- prezentační vrstva se do vyřešení požadavku aplikační vrstvou stane neaktivní (použitá synchronní komunikace), po přijmutí výsledku požadavku se aktivuje a pracuje s daty výsledku.

Aplikační vrstva přijímá tyto požadavky od prezentace události.

- Load (timestamp start, timestamp end) – požadavek na všechny události v daném termínu pro přihlášeného uživatele, je vraceno pole objektů Event ošetřených podle práv pro viditelnost přihlášeného uživatele;
- Save (event) – požadavek na uložení/změnu události, požadavek je zpracován, provede se kontrola „volného termínu“, propagace do sdílených událostí a poté je vrácena chyba, nebo data změněná událost;
- Move (event.id, timestamp start, timestamp end) – požadavek vyvolaný pomocí drag&drop, provede se kontrola „volného termínu“, propagace do sdílených událostí a poté je vrácena chyba, nebo data změněné události, mohla by být použita funkcionalita save, ale zbytečně by docházelo k zatěžování při nezbytných kontrolách;
- GetDialog (event.id, timestam start, timestamp end) – požadavek na získání dat a vykreslení dialogu pro vytvoření/editaci/náhled události, dojde k načtení daných dat a vytvoření html obsahu pro vykreslení v dialogu, který je odeslán prezentační vrstvě;
- getLayerRule (Layer.id) – požadavek vyvolaný změnou vrstvy v dialogu událostí, slouží pro dynamické načtení přednastavených pravidel vrstvy.

Požadavky od prezentace vrstvy:

- load() - požadavek na načtení všech vrstev přihlášeného uživatele;
- save(layer) - požadavek na uložení/editaci, provede se případná propagace do sdílených vrstev, případně i do sdílených událostí sdílených vrstev a jsou vrácena upravená data vrstvy;
- delete(layer.id) – požadavek na smazání vrstvy;
- changeColor (Layer.id, string color) – požadavek na změnu barvy vrstvy;
- getDialog(layer.id) – požadavek na získání dat a vykreslení dialogu pro vytvoření/editaci/náhled události, dojde k načtení daných dat a vytvoření html obsahu pro vykreslení v dialogu, který je odeslán prezentační vrstvě.

Požadavky od prezentace uživatele:

- login (email, passsword) – požadavek na zalogování uživatele;
- logout () – požadavek na odhlášení uživatele

## 7.3 Prezentační vrstva

Prezentační vrstva je reprezentována JavaSciptovou aplikací běžící ve webovém prohlížeči. Přijímá události od uživatele, především click a drag&drop, pokud jsou tyto události nad ovládacími prvky, prezentační vrstva to rozpozná, a pokud má dostatečné údaje a typ události to dovoluje, vyřeší jí sama a překreslí uživatelský výstup, pokud jsou údaje nedostačující, dojde k dotazu na aplikační vrstvu jedním z požadavků. Kompletní popis implementace prezentační vrstvy nalezne čtenář v samostatné kapitole.

# 8 Implementace serverové části

Pro implementaci nebyl zvolen žádný PHP framework. Autor práce se domnívá, že pro ukázkovou implementaci s možností pozdějšího rozvoje by nebylo vhodné framework použít, i když se částečně inspiruje MVC architekturou. Základní myšlenkou MVC architektury je oddělení logiky od výstupu, celá aplikace je rozdělena na komponenty 3 typů. Model, View, Controller (Fowler, M., 2006).

Autor se snažil aplikaci napsat tak, aby byla snadno rozšířitelná, proto jsou všechny interní funkce ve stavu protected a v Controllerech jsou využívané prázdné třídy pouze s dědičností na implementovanou třídu. To umožní snadné přetížení a upravení dané funkce. Je to také jedna z cest, která umožní přidávání serverových pluginu.

## 8.1 Controller

Server přijímá komunikaci od klienta na více vstupních bodech. Nevyužíváme zde běžný způsob komunikace pouze s jedním bodem z důvodů možnosti oddělit práci jednotlivých součásti klienta. To nám poskytne možnost tyto součásti implementačně měnit. Rozdělení také zpřehledňuje názornost komunikace. Tyto vstupy jsou:

- *controller.Event.php* – zpracovává požadavky na události,
- *controller.Layer.php* – zpracovává požadavky na vrstvu,
- *controller.User.php* – zpracovává požadavky na uživatele.

Ke zpracování požadavků serverová část aplikace využívá jednotlivé modely

## 8.2 Modely

Všechny modely k přístupu k databázi využívají databázovou abstraktní vrstvu dibi (Nette, 2013a), která je také komponentou známého českého PHP Frameworku Nette (Nette, 2013b).

- *Event* - event je datový model, který spravuje vše kolem událostí v databázi, je reprezentován tabulkou Event. Třída Event poskytuje veřejné metody, které umožňují pracovat s události a událostmi v požadované míře;
- *Layer* - layer je datový model reprezentující vrstvu, v databázi jsou její data rozdělena do tabulek Layer a LayerJoin, kde v tabulce Layer jsou uložena data určující

danou vrstvu a v tabulce LayerJoin obsahuje informace specifické pro spojení uživatele s vrstvou (práva, barva, atd.);

- *Layers* – model, který umožňuje snadnější práci s více vrstvami jednoho uživatele;
- *User* – jednoduchý datový model reprezentující uživatele, v databázi je reprezentován tabulkou user.

## 8.3 View

Aplikace komunikuje s klientskou částí pomocí protokolu JSON, proto mnoho pohledů není. Existuje pouze pohled pro přihlášení a registraci nového uživatele.

# 9 Implementace klienta

Klient je vystavěn nad grafický webový prohlížeč s podporou JavaScriptu pomocí JavaScriptového frameworku jQuery s rozšíření jQuery UI. Tomuto frameworku jsme dali přednost před jinými alternativami (např. Prototype, Dtojo, Ext) z důvodu velkého rozšíření a podpory.

Jak bylo napsáno již dříve, uživatelská přívětivost je velmi důležitá. Při zběžném prohlédnutí stávajících aplikací (Google Calenda, phps, atd.), zjistíme, že mají více méně velmi podobný vzhled a základní funkční prvky, které jsme popsali v naší představě o ideální plánovací aplikaci (viz subkapitola 3.1). Protože se jedná o rozšířené požadavky, podíváme se na vhodnost použití jQuery pluginů, které by umožňovaly zrychlit vývoj.

- **jQuery FullCalendar** (Shaw A, 2013a) – obsahuje kalendář s více možnostmi režimu pohledu, drag&drop, podporu jQuery UI Theme, kvalitní dokumentaci funkcí, komunitní podporu;
- **jQuery Frontier Calendar**[3] – obsahuje pouze kalendář s měsíčním režimem, drag&drop, kvalitní dokumentaci;
- **jQuery jMonthCalendar**[4] **– o**bsahuje pouze kalendář s měsíčním režimem, drag&drop, průměrnou dokumentaci;
- **jQuery Week Calendar**[5] – obsahuje pouze kalendář s týdenním režimem zobrazení, drag&drop, průměrnou dokumentaci;
- **jQuery wdCalendar**[6] **–** obsahuje kalendář s více možnostmi režimu pohledu, drag&drop, téměř žádnou dokumentaci.

Z této nabídky zajímavých pluginů, které by nám mohly pomoci, jsme vybrali jQuery FullCalendar, a to z důvodů podpory více režimů zobrazení (což byl jeden z našich požadavků na ideální plánovací aplikaci), kvalitní dokumentace (Shaw A, 2013b) a komunitní podpory. Zároveň umožňuje rozšíření o některé požadavky uživatelské přívětivosti popsané v kapitole 3.1, zejména drag&drop, grafická přehlednost a skinovatelnost pomocí jQuery UI Theme. Plugin se pouze zabývá

3 K nahlédnutí zde: http://ifsocial.dlbtampa.com/monthcalendar/calendar.html
4 K nahlédnutí zde: http://code.google.com/p/jmonthcalendar/
5 K nahlédnutí zde: https://github.com/robmonie/jquery-week-calendar
6 K nahlédnutí zde: http://www.web-delicious.com/jquery-plugins/#calendar

vykreslováním předaných dat a vrací callback funkce na některé události. Nezabývá se tedy správou dat, jejich vytvářením, či ukládáním, to vše je potřeba doimplementovat, také se nezabývá správou vrstev a nemá pro ni podporu, rovněž neimplementuje tzv. mini kalendář (data picker) pro rychlý posun v kalendáři.

## 9.1 Rozšíření pro FullCalendar

Jak bylo napsáno, plugin FullCalendar ze zabývá pouze vykreslováním předaných dat a vrací callback funkce, tyto funkce je potřeba doimplementovat. V naší aplikaci jsme využili následující funkce:

- *select* (start, end, allDay) – reaguje na akci výběru uživatele (provedeným kliknutím a možným tažením), zde je doplněno vyvolání serverové operace getDialog a inicializování dialogu události;
  - *start* – JavaScriptový objekt date reprezentující začátek místa výběru,
  - *end* – JavaScriptový objekt date reprezentující konec místa výběru,
  - *allDay* – informuje, zda byl výběr proveden v pohledu podporující celodenní pohled,
- *EventDrop* (event, dayDelta, minuteDelta, allDay, revertFunc) – reaguje na akci drag&drop provedenou uživatelem na události. Tuto funkci jsme doplnili o vyvolání serverové operace move;
  - *event* - konkrétní instance události,
  - *dayDelta* – určuje počet dní rozdílu mezi změněným a původním počátkem události,
  - *minuteDelta* - určuje počet minut rozdílu mezi změněným a původním počátkem události,
  - *allDay* - zda je změněná událost nastavena na celodenní zobrazování,
  - *revertFunc* - funkce vracející událost na původní čas.
- *EventResize* (event,dayDelta,minuteDelta,revertFunc) – reaguje na akci resize (změnění velikosti pole znázorňujícího událost) provedenou uživatelem na události. Tato funkce byla také doplněna o vyvolání serverové operace move;

- *event* - konkrétní instance události,
- *dayDelta* – určuje počet dní rozdílu mezi změněným a původním počátkem události,
- *minuteDelta* - určuje počet minut rozdílu mezi změněným a původním počátkem události,
- *allDay* - zda je změněná událost nastavena na celodenní zobrazování,
- *revertFunc* - funkce vracející událost na původní čas.

- *eventClick(calEvent)* – reaguje na kliknutí uživatele na událost. Doplněno o vyvolání serverové operace getDialog a inicializování dialogu události ze získaných dat,
  - *calEvent* – konkrétní instance události, na kterou bylo kliknuto.
- *eventRender(event)* – reaguje na požadavek vykreslení jakékoliv události. Museli jsme doplnit o podporu vrstev, pokud je vrstva skrytá události jí náležející se nevykreslují, také zde musíme nastavit barvu vykreslení události,
  - *calEvent* – konkrétní instance události, kterou vykreslujeme.
- *viewDisplay(view)* – tato funkce reaguje na změnu pohledu. My toto volání využíváme k přenesení změny zobrazení na data picker, aby mohl zobrazovat datum pohledu,
  - *view* – aktuální instance pohledu. Také obsahuje množství konstant, které je nutné předvyplnit:

Lokalizační konstanty – bohužel plugin neumí pracovat s lokalizačním souborem jQuery UI, proto je nutné vyplňovat všechny lokalizační konstanty, od překladů měsíců a dnů, přes po formáty vykreslení času a data, až po nastavení prvního dne v týdnu.

Konstanty pro zobrazení – při inicializaci je potřeba nastavit, jak má vypadat hlavička kalendáře, které všechny režimy pohledu mají být dostupné, také od kterého dne a do jakého zobrazovacího dne se má kalendář po inicializaci zapnout, na jakou hodinu má být nastavené první přepnutí na týdenní či denní zobrazení. Tyto konstanty jsou popsány v dokumentaci (viz Shaw A, 2013b), bohužel v tomto ohledu dokumentace zaostává a mnohé principy nastavení musely být vyčteny ze zdrojových kódů.

Dialog události je vytvořen za pomocí pluginu dialog z rozšíření jQuery UI, příjme data od uživatele, poté je odešle ke zpracování a případnou kladnou odpověď musí vypropagovat zpět FullCalendary. Nové vykreslení událostí se provede funkci renderEvent, která vykreslí změnu do kalendáře.

## 9.2 Mini kalendář pro rychlý posun

Mini kalendář je v naší implementaci reprezentován pluginem data picker z rozšíření jQuery UI. Zde bylo potřeba nastavit českou lokalizaci a při výběru data vyvolat změnu v instanci pluginu FullCalendar. Pro přesun na určené datum má FullCalendar veřejnou metodu goDate.

## 9.3 Vrstvy

Plugin FullCalendar nepodporuje vrstvy, nechěli jsme podporu implementovat přímo do tohoto pluginu, protože by byla omezená možnost upgradu na novější verzi. Proto jsme se rozhodli, že vyvineme vlastní plugin LayrList.

LayerList má pouze 3 veřejné metody:

- *init* – dojde k získání dat ze serveru a vykreslení základního layoutu;
- *newLayer* – dojde k vyvolání dialogu pro vložení nové vrstvy, tato metoda musí být veřejná, abychom umožňovali vytváření vrstev i mimo tento plugin. My tuto vlastnost využíváme, když umožňujeme vytvářet novou vrstvu z dialogu události;
- *GetCalendaryRender (layerId)* – tato metoda je vyvolávaná z metody fullCalendar.eventRender, pro získání informací pro vykreslení událostí. To jsou informace, jestli vrstva není skrytá a jestli se tedy událost nemá vykreslovat, případně jakou barvo se má událost vykreslit, pokud nemá nastavenou svojí vlastní barvu;
- *LayerId* – identifikátor vrstvy.

## 9.4 Barvy

Každá vrstva i událost může mít nastavenou vlastní barvu, pro dialog nastavení barvy používáme plugin colorPicker (Petre S, 2009). Tento plugin jsme vybrali, protože jako jeden z mála obsahuje více způsobů výběru barvy a je plně funkční. Bohužel jQuery UI nedisponuje žádným colorPickerem, i když na jeho vývoj je vytvořena skupina, už přes 4 roky.

## 9.5 Změna vzhledu

Za použití CSS (Cascading Style Sheets) (W3C, 2011) můžeme měnit vzhled uživatelského rozhraní aplikace. Abychom toho docílili, museli jsme ve vyvíjených částech dodržet pravidla jQuery UI Theming (jQuery UI Theming, 2013). Nový vzhled můžeme vytvořit pomocí jQuery ThemeRoller (jQuery UI ThemeRoller, 2013) nebo použít již vytvořené, našim potřebám vyhovující vzhledy. Nový vzhled do aplikace implementujeme přehráním původního vzhledu. Zde se nabízí možnost pro vylepšení aplikace, a to, že by si každý uživatel mohl vybrat vzhled z předem vybraných a přístupných vzhledů.

# 10 Výsledky a shrnutí práce

V tuto chvíli se nacházíme před posledním úkolem, provedenou práci určitým způsobem zhodnotit. V následujících odstavcích rozdělíme hodnocení do několika částí, zaměříme se na hodnocení implementace, porovnání s ideální plánovací aplikací, systém práv, nedostatky použité metody a celkové hodnocení, včetně přínosu práce pro autora samotného. Začneme-li tedy dle shora uvedeného pořadí a budeme v prvé řadě hodnotit implementaci, pak můžeme konstatovat, že ta se zdařila po stránce vizuální a interaktivní, na úkor rozsahu funkčnosti. To ale nebránilo splnění cíle v praktickém odtestování návrhu systému práv.

Pokud budeme porovnávat implementovanou aplikaci s návrhem naší ideální plánovací aplikace, tak můžeme říci, že následující body se nám podařilo implementovat:

- kalendář s režimem zobrazení denní, týdenní, měsíční a možností grafického nebo textového režimu;
- události informativní a s plánovanou účastí;
- kontrola časové vytíženosti.
- více vrstev,
- uživatelsky přívětivé chování;
- sdílení a nabízení události s možností správy těchto událostí;
- plánování třetí osobou.

Při implementaci byl brán zřetel i na ostatní body z původního seznamu, proto by mělo být možné aplikaci rozšířit i o tyto ostatní body, s výjimkou čistého exportu a importu do externích kalendářových aplikací pomocí popsaných standardu, a to z důvodu nekompatibility práv. Tento nedostatek lze obejít, ale nikdy tak nebudeme schopni docílit 100% synchronizace.

Hodnocení navrhovaného systému práv je rozporuplné. Uživatel získá velkou svobodu ve sdílení, ale v některých situacích se práva chovají jinak, než by uživatel očekával.(například pokud zrušíme sdílení vrstvy, tak bychom mohli předpokládat, že uživatel přijde o všechny stávající události, ale tomu tak není, uživatel pouze nebude

získávat nové události a ztratí přehled o účasti). Také díky právům přijdeme o možnost 100% exportu a importu do externího kalendáře, jak bylo již zmíněno v předchozím odstavci. Toto má za následek znemožnění plného využití jiných aplikací a uživatel je tak plné odkázán jen na naši aplikaci.

Se získanými zkušenostmi by autor pro implementaci v budoucnu volil nerelační databázi MongoDB a k implementaci klienta by zvolil JavaScript MVC framework Angular.Js. Odvedená práce autorovi přinesla kromě velkého přehledu o dané problematice (standardy, protokoly, současné aplikace) i jiný pohled na plánování vlastního času.

# Závěr

Po seznámení se se stávajícími plánovacími aplikacemi jsme zjistili, že žádná nesplňuje naše očekávání. Proto jsme analyzovali a sepsali požadavky na ideální plánovací aplikaci. Po rozmyšlení si, co k plánování používáme, jsme zjistili, že ideální plánovací aplikace by měla spolupracovat s rozhraními jiných aplikací, které při plánování přímo, či nepřímo využíváme. Tato aplikace by rovněž měla být uživatelsky příjemná a dostupná na většině zařízení. Pokud jde o názor na to, jaké nadstandardní funkce by měla ideální plánovací aplikace mít, bude se tento názor lišit uživatel od uživatele. Z pohledu autora této práce, jsou žádoucími funkcemi především zachování soukromí při plné funkčnosti aplikace, nerestriktivní pravidla pro sdílení a kontrola „volného termínu“.

Po teoretickém zkonstruování ideální plánovací aplikace v první části práce, jsme ve druhé její části přistoupili k technické realizaci naší vlastní aplikace. Za tímto účelem bylo nejprve třeba učinit některá implementační rozhodnutí. Dále byl diskutován a následně navržen přístup k právům v naší aplikaci, a vysvětlena a zdůvodněna její logika. V další části práce již byla věnována pozornost implementačním záležitostem, zejména implementaci serverové části a implementaci klienta. Na závěr autor provedl hodnocení provedené práce. Z tohoto hodnocení je patrné, že ačkoli se autorovi nezdařilo naplnit stanovený cíl práce na sto procent, přináší práce vysokou přidanou hodnotu v podobě reálného pohledu na problematiku programování aplikace tohoto typu, včetně vymezení faktických úskalí s jejím programováním spojených. Předložená bakalářská práce tak otevírá prostor pro realizaci dalších podobných aplikací, například s využitím jiných nástrojů.

# Seznam zdrojů:

CROCKFORD D. 2006. The application/json Media Type for JavaScript Object Notation (JSON). Network Working Group. 2007. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://tools.ietf.org/html/rfc4627.

DABOO, et al. 2007. Calendaring Extensions to WebDAV (CalDAV). Network Working Group. 2007. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://tools.ietf.org/html/rfc4791.

DAWSON, et al. 1998. iCalendar Message-Based Interoperability Protocol (iMIP). Network Working Group. 1998. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://tools.ietf.org/html/rfc2447.

DAWSON, F.; STENERSON, D. 1998. Internet Calendaring and Scheduling Core Object Specification (iCalendar). Network Working Group. 1998. [on- line], [cit. 2013-08-01]. Dostupný z WWW: http://tools.ietf.org/html/rfc2445.

DESRUISSEAUX, B. 2009. Internet Calendaring and Scheduling Core Object Specification (iCalendar). Network Working Group. 2009. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://tools.ietf.org/html/rfc5545.

GNU OPERATING SYSTEM. 2007. GNU General Public License. GNU Operating System. 2007. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://www.gnu.org/licenses/gpl.html.

GOOGLE. 2013. Google Kalendář. Google. 2013. [on-line], [cit. 2013-08- 01]. Dostupný z WWW: http://www.google.com/calendar.

JQUERY UI THEMING. 2013. jQuery UI Theming. 2013. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://api.jqueryui.com/.

JQUERY UI THEMEROLLER. 2013. jQuery UI ThemeRoller. 2013. [on- line], [cit. 2013-08-01]. Dostupné z WWW: http://jqueryui.com/themeroller/.

LIGHTNING. 2013. Mozilla Lightning 2013. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://www.mozilla.org/projects/calendar/lightning/

LOTUS SOFTWERE. 2013. Lotus Softwere. IBM. 2013. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://www-01.ibm.com/software/lotus/.

MAHONEY, et al. 2002. Guide to Internet Calendaring. Network Working Group. 2002. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://tools.ietf.org/html/rfc3283.

MICROSOFT OFFICE. 2013. Aplikace pro e-mail a kalendář. Microsoft Office. 2013. [on-

line], [cit. 2013-08-01]. Dostupný z WWW: http://office.microsoft.com/cs-cz/outlook/.

OPEN SOURCE I. 2013a. The BSD License. Open Source Initiative. 2013. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://www.opensource.org/licenses/bsd-license.php.

OPEN SOURCE I. 2013b. The GNU General Public License. Open Source Initiative. 2003. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://www.opensource.org/licenses/gpl-2.0.php.

OPEN SOURCE I. 2013c. The MIT License. Open Source Initiative. 2013. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://www.opensource.org/licenses/mit-license.php

EXFORSYS. 2007. Peer-to-Peer and Client-Queue-Client Architecture. Exforsys Inc. 2007. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://www.exforsys.com/tutorials/client-server/peer-to-peer-and-client-queue-client-architecture.html.

FOWER M. 2006. GUI Architectures. Martin Fowler. 2006. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://martinfowler.com/eaaDev/uiArchs.html#ModelViewController

NETTE. 2013a. Dibi is Database Abstraction Library for PHP 5. Nette foundation. 2013. [on-line], [cit. 2013-08-01]. Dostupná z WWW: http://dibiphp.com/

NETTE. 2013b. Nette Framework .Nette foundation. 2013. [on-line], [cit. 2013-08-01]. Dostupná z WWW: http://nette.org/

PETER S. 2009. Color Picker. Stefan Peter. 2009. [on-line], [cit. 2013-08- 01]. Dostupný z WWW: http://www.eyecon.ro/colorpicker/.

PHP ICALENDAR. 2013. PHP iCalendar. PHP iCalendar. 2013. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://www.softaculous.com/apps/calendars/phpicalendar.

PLZAK, et al. 1999. FYI on Questions and Answers. Answers to Commonly Asked "New Internet User" Questions. Network Working Group. 1999. [on- line], [cit. 2013-08-01]. Dostupný z WWW:

http://tools.ietf.org/html/rfc2664.

PHPSCHEDULEIT. 2013. PhpScheduleIt. PhpScheduleIt. 2013. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://php.brickhost.com/index.php.

ROYER, et al. 2005. Calendar Access Protocol (CAP). Network Working Group. 2005. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://tools.ietf.org/html/rfc4324.

SHAW A. 2013a. FullCalendar Introduction. Adam Shaw. 2013. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://arshaw.com/fullcalendar/.

SHAW A. 2013b. FullCalendar Documentation. Adam Shaw. 2013. [on- line], [cit. 2013-08-01]. Dostupný z WWW: http://arshaw.com/fullcalendar/docs/.

SMART PHP CALENDAR. 2013. Smart PHP Calendar. 2013 [on-line], [cit. 2013-08-01] Dostupný z WWW:http://smartphpcalendar.com/

Oracle. 2013. Introduction to SSL, Oracle. 2013. [on-line], [cit. 2013-08-01]. Dostupný z WWW: http://docs.oracle.com/cd/E13222_01/wls/docs81/secmanage/ssl.html.

W3C. 2011. Cascading Style Sheets. 2011. [on-line], [cit. 2013-06-03]. Dostupný z WWW: http://www.w3.org/TR/2011/NOTE-css-2010-20110512/.

XMPP. 2013. Protocols The XMPP Standards Foundation. 2013. [on-line], [cit. 2013-08-01]. Dostupné z WWW: http://xmpp.org/xmpp-protocols/

# Příloha č. 1 – Uživatelská dokumentace

## Instalace

Běh aplikace vyžaduje webový server Apache ( se standardní konfigurací) s rozšířením PHP 5.4 a rozšíření php5-pgsql, php-pdo, php-pdo-pgsql. S tímto rozšířeními se muže vyskytnout problém se staršími verzemi Apache, že nedochází k dobré automatické konfiguraci potřených rozšíření, tyto konfigurace jsou nutné provést ručně (závisí na operačním systému a jeho distribuce).

Instalace probíhá způsobem stejným jako zveřejnění webové prezentace. Na každém systému se tento způsob muže lišit. (např. nakopírováním adresáře daných zdrojových souborů do adresáře /var/www (kořenový adresář webu) ). V zásadě stačí mít veřejně přístupné soubory index.php a soubory v adresáři Controller.

Změněním konfiguračního souboru configuration.php v domovském adresáři aplikace, kde nastavíme konstantu ROOT_URL, zde vyplníme absolutní url na které je veřejně dostupná webová aplikace. Dále zde můžeme nastavit různé cesty k souborům, ale standardně cesty měnit nemusíme. Dále se v souboru nachází pole $dbSetting, jedná se o konfiguraci spojení pro rozšíření dibi, kde také naleznete případné detaily. Nám bude stačit vyplnit správně položky: host, username, password, database

Instalace struktur databáze musí být zajištěna uživatelem s potřebnými oprávněními, proto je nutné aby databáze byla vytvořena ručně. Obsah databáze po spuštění vytvoří script install.php . Předvytvořeny jsou uživatelé „Demo“ s heslem „demo“ a uživatele „Share“ s heslem „Share“ a „share1“ s heslem „share1“.

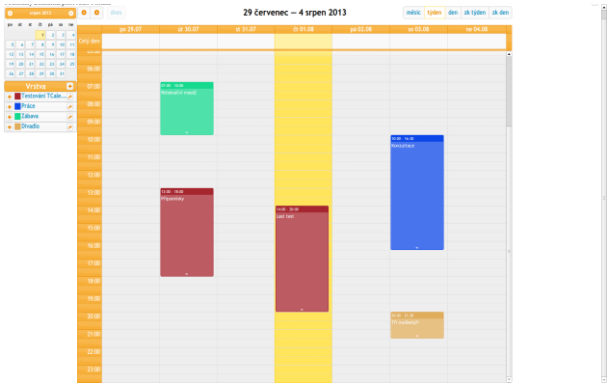
## Odinstalace

Aplikace si nevytváří žádné dočasné soubory ani jiné struktury, proto je odinstalování velmi jednoduché, stačí smazat nahrané soubory a odstranit přidanou databázi.

## Uživatelská dokumntace

### 1. Základní rozhraní TCalendar

Po přihlášení pod přiděleným uživatelským jménem a heslem se nám zobrazí základní rozhraní kalendáře – viz obr. 1. Kalendář se otevře na aktuálním měsíci se zvýrazněním data dne, který právě je.



**Obr. 1 - Základní rozhraní**

Implicitně je nastaveno měsíční zobrazení, které je možno měnit pěti tlačítky, která jsou situována vpravo nahoře, aktuální nastavení je zvýrazněno.

Tlačítka umožňující měnit nastavení v základním rozhraní:

- měsíc – zobrazení celého měsíce,
- týden – zobrazení týdne s časovou osou dělenou po 30 minutách,
- den – zobrazení dne s časovou osou dělenou po 30 minutách,
- zk týden - (zkrácený) zobrazení bez osy, události řazeny za sebou,
- zk den – (zkrácený) zobrazení bez osy, události řazeny za sebou.

Šipky vlevo nám slouží k navigaci v zobrazení. Pokud opustíme zobrazení dnešního dne, zpřístupní se nám tlačítko dnes, které slouží k rychlému návratu na dnešní den v aktuálním zobrazení.

V pravém horním rohu je umístěno odhlašovací tlačítko.

V levém rohu nalezneme zmenšený kalendář, který slouží k rychlému přesunu na hledaný den.

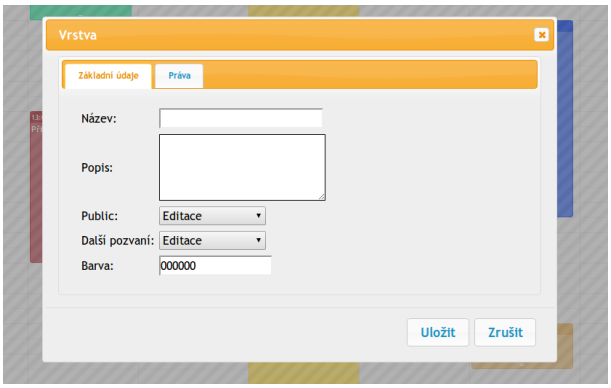
Pod zmenšeným kalendářem (viz předchozí bod) nalezneme sekci Vrstva, která nám slouží k zobrazení a ke správě jednotlivých vrstev – podrobněji viz následující sekce uživatelské dokumentace.

# 2. Vrstvy

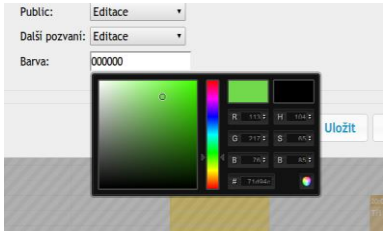
Veškeré události náleží některé z vrstev, proto se nyní seznámíme s fungováním vrstev.

K vytvoření vrstvy slouží tlačítko „+“ umístěné vedle textu „Vrstva“. Po jeho

stisknutí se nám zobrazí dialog (viz obr. 2). V první záložce dialogového okna vyplníme údaje o vrstvě, tj. jméno vrstvy a rovněž můžeme připojit její popis a změnit barvu pomocí dialogu změny barvy (viz obr. 3 níže). Poté můžeme přidat dalším uživatelům práva, která vybereme pomocí záložky „Práva“.

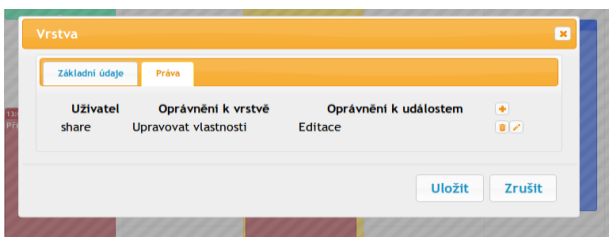



**Obr. 2 – dialog vrstvy**



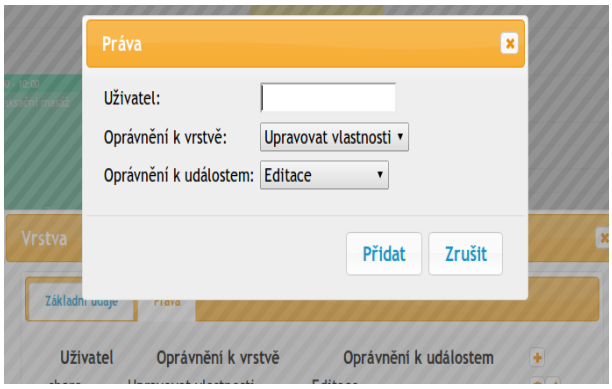

**Obr. 3 – dialog výběru barvy z dialogu vrstvy**

Druhá záložka (záložka „Práva“) je určena pro práva sdílení vrstvy, ve které se zobrazí seznam uživatelů, se kterými danou vrstvu sdílím a jejich práva k vrstvě a přednastavené právo k událostem (blíže viz obr. 4). Pomocí tlačítka „+“ můžeme přidat práva pro sdílení dalšímu uživateli. Pokud již vrstvu sdílíme s jiným uživatelem, máme možnost zrušit sdílení pomocí ikony popelnice, nebo upravit režim sdílení pomocí ikonky tužtičky.



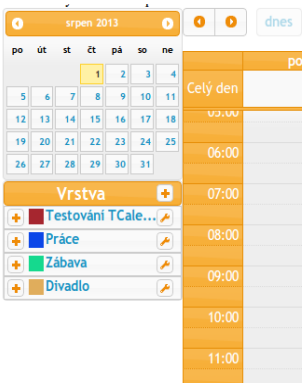

**Obr. 4 – záložka práv sdílení vrstvy**

Dialog pro přidání práva sdílení dalšího uživatele (obr. 5), nám umožňuje přidat právo sdílení danému uživateli. Pro uskutečnění tohoto úkonu musíme znát uživatelské jméno uživatele, kterého chceme přidat, následně mu nastavíme práva pro sdílení a při potvrzení se vracíme zpět na dialog vrstvy do záložky práv, kde již vidíme přidaného uživatele. Pokud už máme vše vyplněno tak, jak si přejeeme, můžeme vrstvu uložit, a to stisknutím tlačítka „Uložit“.




**Obr. 5 – dialog pro přidání práva sdílení dalšího uživatele**

Takto vytvořená vrstva se přidá k již existujícím vrstvám v levé části základního rozhraní (viz obr. 6).




**Obr. 6 – existující vrstvy**

Jak je patrné z výše uvedeného obrázku (viz obr. 6), zobrazuje se zde název vrstvy, před kterým je umístěno barevné políčko a znaménko „+“. Za názvem vrstvy je klíč. Nyní si popíšeme, co jednotlivé symboly znamenají.

Symbol „+“ nám ukazuje, že se vrstva zobrazuje do rozhraní. Pokud na toto tlačítko klikneme, změní se na symbol „-“ (mínus), což značí, že vrstva není aktivní. Po opětovném stisknutí tohoto tlačítka se obnoví symbol „+“ a vrstva se zviditelní.

Symbol barevného políčka nám značí, jakou barvou se zobrazují události z této

vrstvy s nezměněnou barvou. Pokud na barevné políčko klikneme, otevře se dialog pro změnu barvy (obr 7). Provedení změny barvy potvrdíme tlačítkem v pravém dolním rohu.


**Obr. 7 – dialog pro změnu brvy ze základního rozhraní**

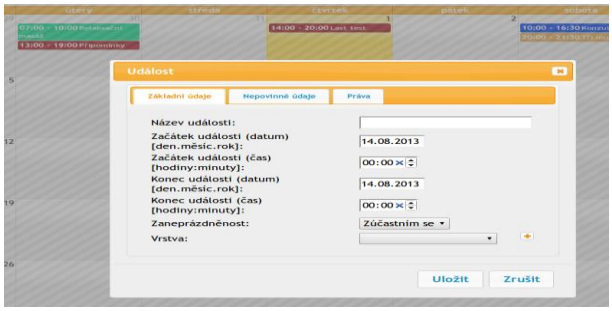
Pokud klikneme na symbol klíče, otevře se nám stejný dialog, jako je u vytváření nové vrstvy s vyplněnými údaji o vrstvě. Máme-li právo na editaci, uvidíme příslušná ovládací tlačítka v pravém spodním rohu.

# 3 Události

Pojmem „Událost“ je v této aplikaci označována akce, která má definovaný čas začátku a konce. V této části se seznámíme s tím, co nám tato akce nabízí.

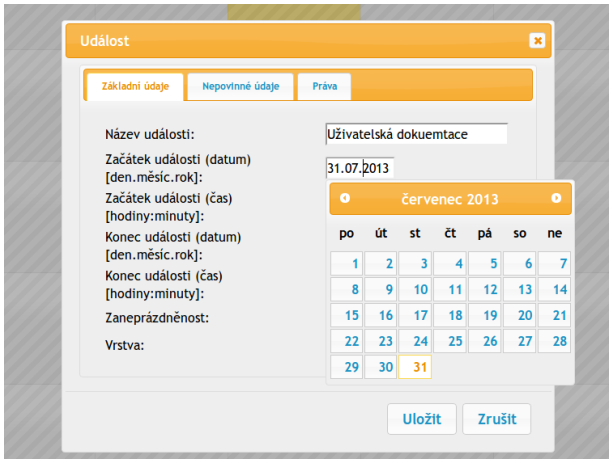
### Tvorba události

K vytvoření nové události můžeme využít několik postupů. Základním je kliknutí na kterýkoli den kalendáře, čímž se nám otevře dialog „Událost“ s předvyplněným datem vybraného dne (názorně viz obr. 8).




**Obr. 8 – dialog: vytvoření nové události**

Vyplníme název události a dále stanovíme časové parametry vytvářené události. Kliknutím do kolonky s datem se nám otevře navigační kalendář (viz obr. 9), ve kterém vybereme námi požadované datum začátku události. Zadáme čas začátku události v požadovaném formátu. Stejně postupujeme u zadání data a času ukončení události.

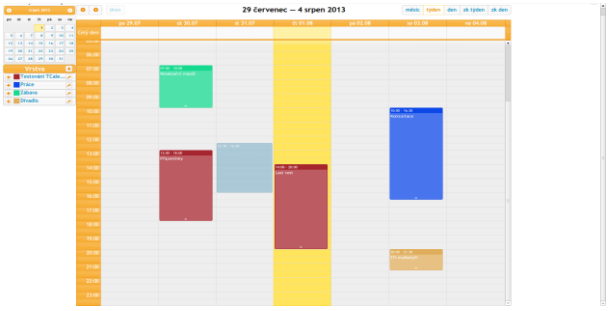



**Obr. 9 – vybrání dne pomocí navigačního kalendáře**

Komfortnější metodu vytvoření nové události představuje vybrání kliknutím na požadovaný den, což vede k předvyplnění správného data do dialogu události. Pro zobrazení požadovaného dne využijeme navigační šipky či navigační kalendář v základním rozhraní. Pokud se událost opakuje v po sobě následujících dnech, můžeme dny vybrat stisknutím a tažením myši přes požadované dny, což vede k předvyplnění data začátku a konce události.

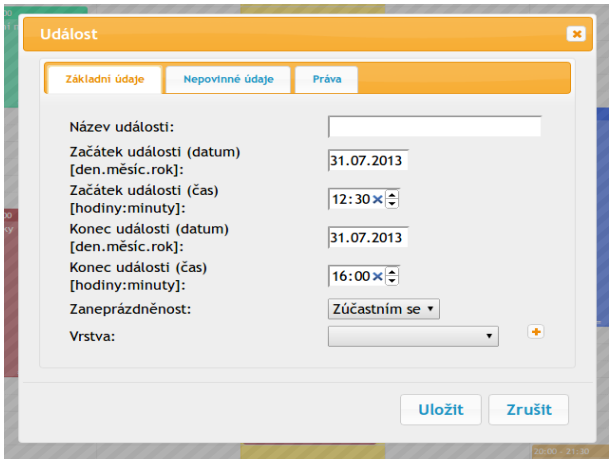
Poslední možností, jak vytvořit novou událost, je výběr týdenního/denního zobrazení, což vede k lepšímu přehledu o ostatních událostech. Výhoda této metody se projeví tehdy, kdy je v určitém dni naplánováno větší množství událostí.

Ukázka vytvoření nové události pomocí posledně zmiňované metody: Nejprve přepneme zobrazení do týdenního a pomocí navigačního kalendáře vyhledáme požadovaný týden (den). Aplikace nám zobrazí vybraný den s již vytvořenými událostmi. Stisknutím levého tlačítka a tažením myší vybereme požadovaný čas nové události (viz obr. 10), otevře se nám dialog události s předvyplněným dnem i časem



nové události (názorně viz obr. 11).

**Obr. 10 – vytvoření události pomocí zvoleného časového úseku**

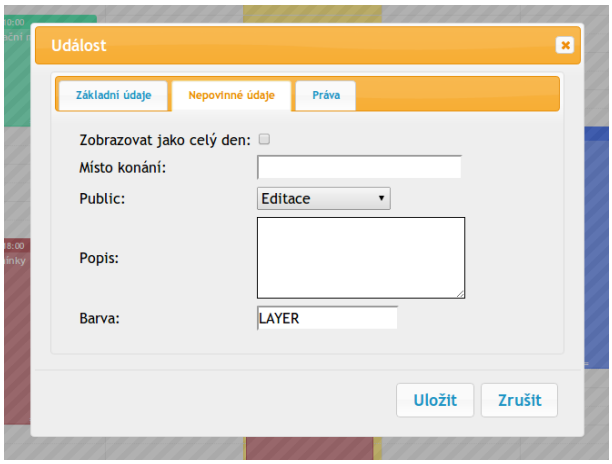



**Obr. 11 – dialog pro vytvoření nové události pomocí zvoleného časového úseku**

Tímto jsme si ukázali, jak je možné zadat časové parametry různými způsoby. V dialogu události pod časovými údaji nalezneme výběrové menu „Zaneprázdněnost“, kterým můžeme potvrdit svou účast na dané události.

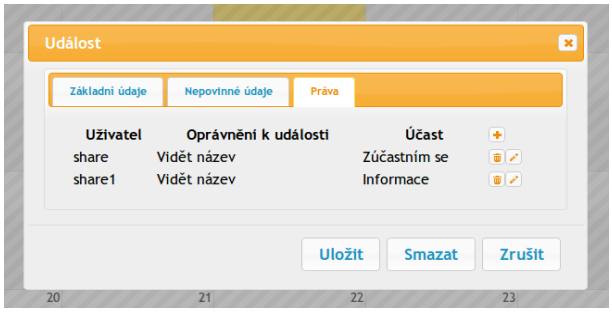
Posledním povinným údajem je přiřazení nově vznikající události k vrstvě. Toho dosáhneme pomocí výběrového menu, kde se zobrazí všechny dostupné vrstvy (viz výše část věnována vrstvám). Po vyplnění první záložky, máme možnost vytvořit událost stisknutím tlačítkem „Uložit“, nebo nastavit další údaje, k čemuž slouží další záložky.

Po kliknutí na druhou záložku s nepovinnými udají události (viz obr. 12), kde můžeme vyplnit další informace o události, jako je místo konání, popis události, barvu, zde máme také možnost zaškrtnout, zda se událost bude zobrazovat jako celodenní, bez ohledu na čas začátku a konce události. Pokud máme událost, která zabírá větší část dne, je pro lepší přehlednost vhodné mít toto políčko zaškrtnuté. Základní nastavení tohoto políčka závisí na typu zvolené metody vytváření nové události. Pokud použijeme posledně jmenovanou metodu, políčko je nezaškrtnuté, v ostatních případech platí opak. Pokud nevybereme žádnou barvu události, barva bude odpovídat barvě přiřazené vrstvy.

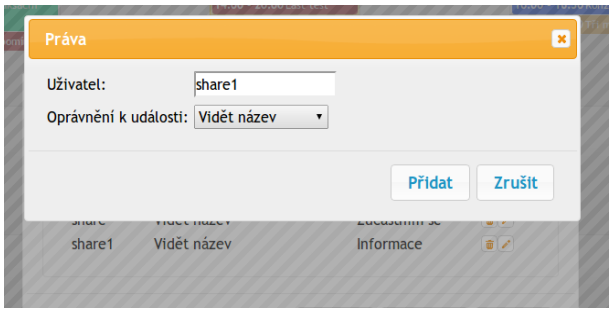



**Obr. 12 – nepovinné údaje události**

Další záložkou jsou „Práva sdílení“. Zde můžeme vytvářenou událost poskytnout dalším uživatelům a vidíme zde účast ostatních pozvaných uživatelů (názorně viz obr. 13). Ovládací prvky jsou zde stejné jako u záložky práv vrstvy, liší se pouze editační dialog (obr. 14).




**Obr. 13 – záložka práva události**




**Obr. 14 – dialog pro přidání/editaci práv na sdílení události**

Poslední záložkou je záložka „Závislost“, která umožňuje nastavení závislosti vytvářené události. Novou událost lze uložit, pouze pokud jsou správně vyplněné údaje. Není-li tomu tak, zobrazí se nám varování s upozorněním na špatně vyplněné údaje.

### Zobrazení a editace události

Pokud si chceme prohlédnout údaje o události, klikneme na vybranou událost a aplikace nám otevře dialog události, pokud máme práva na editaci, vidíme v pravém dolním rohu ovládací tlačítka umožňující danou událost editovat.

### Mazání události

Pokud chceme událost vymazat, stiskneme tlačítko „Odstranit“, což nám vyvolá dotaz, zda opravdu chceme vybranou událost smazat. Po stisknutí potvrzovacího tlačítka (tlačítko „Ano“), je událost nenávratně odstraněna.

# Příloha č. 2 – Obsah přiloženého CD

- \TCalendar – adresář obsahující zdrojové kódy implementace,
- \Text.pdf – elektronická verze bakalářské práce,
- \index.html – úvod pro obsah CD.

# Příloha č. 3 – Schéma databáze