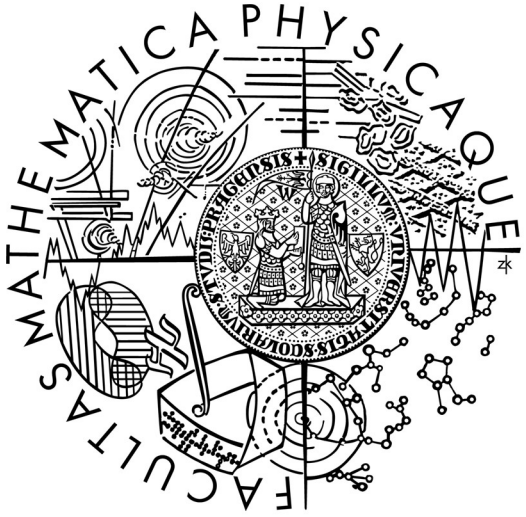
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Jiří Florián

Skupinový diář

Katedra softwarového inženýrství

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

Studijní program: Informatika , programování

2010

Na tomto místě bych ráda poděkoval RNDr. Michalu Žemličkovi, Ph.D. za odborné vedení, cenné připomínky a rady při zpracování bakalářské práce. A současně tímto děkuji i všem, kteří mne po celou dobu mého studia podporovali.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce.

V Praze 5. 8. 2010 Jiří Florián

Obsah

Název práce: Skupinový diář

Autor: Jiří Florián

Katedra (ústav): Katedra softwarového inženýrství

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

e-mail vedoucího: Michal.Zemlicka@mff.cuni.cz

Abstrakt: Ať si to chceme či nechceme připustit, celý náš život se skládá z plnění rozličných úkolů a událostí. A právě vytvoření návrhu aplikace pro snadné plánování úkolů a 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: týmové plánování, diář, skupinové plánování

Title: Group Time Manager

Author: Jiří Florián

Department: Department of Softwre Engineering

Supervisor: RNDr. Michal Žemlička, Ph.D.

Supervisor's e-mail address: Michal.Zemlicka@mff.cuni.cz

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

Keywords: team planning, diary, group scheduling

1 Úvod

1.1 Motivace

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 tohoto „blahobytu“ kladou na zaměstnance, řídící pracovníky, ale třeba i na personalisty neustále se zvyšující nároky na jejich znalosti, dovednosti a schopnosti, vyžadují stálé učení se něčeho nového – souhrnně řečeno, globalizace s sebou rovněž přináší nutnost plnění stále většího množství různých úkolů. A tyto úkoly je třeba nějakým způsobem organizovat. Dříve byly k tomuto účelu 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 nahradily 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 celé skupiny. V tom případě již ale nehovoříme o elektronickém diáři, nýbrž o plánovací aplikaci.

A právě vytvoření návrhu aplikace pro snadné plánování úkolů a 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. Uvedená aplikace zohlední rovněž práci v kolektivu, na níž je dnes všeobecně kladen značný důraz, a to prostřednictvím implementace řady funkcí, jež podporují týmové plánování.

V implementaci se zaměříme na použití systému práv pro zveřejňování informací o akcích. Praktické zhodnocení navrhnutého a implementovaného systému práv je dalším cílem předkládané práce.

1.2 Struktura práce

V předchozím textu jsme si vymezili cíl této práce a rovněž jsme poskytli zdůvodnění, proč se zkoumanou problematikou vůbec zabýváme. Nyní stručně nastíníme, jak budeme při řešení našeho problému postupovat.

Ve druhé kapitole se, vzhledem k výše stanovenému cíli práce, budeme zabývat analýzou současné nabídky softwarů. Uvedeme zde jejich členění podle tří nejčastěji zohledňovaných kritérií – podle finančního hlediska, podle potřeby instalace a podle jejich technické stránky.

Ve třetí kapitole pak naši analýzu dále prohloubíme zkoumáním některých konkrétních, na trhu dostupných, aplikací. Zaměříme se zde především na popis jejich funkčnosti.

Čtvrtá kapitola je následně věnována popisu charakteristiky ideálního plánovacího softwaru, včetně hlavních atributů takovéhoto softwaru. Kromě toho je zde rovněž věnován prostor vymezení s problematikou souvisejících pojmů.

Ideální plánovací aplikace musí být zabezpečena určitými technickými prostředky. Způsoby technické realizace takové aplikace se zabývá pátá kapitola práce.

Šestá kapitola se zabývá logikou návrhu aplikace s detailním popisem přístupu k právům uživatelů.

Volbou architektury, volbou klienta a dalšími implementačními rozhodnutími se zabýváme v sedmé kapitole.

Osmá kapitola je následně věnována konkrétní implementaci serverové části. Popisujeme zde přístupné třídy a jejich vztahy mezi tabulky databáze.

Popisem realizace databáze se zabývá devátá kapitola. Zejména zde poukazujeme na zajímavé funkce a triggery.

Desátá kapitola je věnována popisu implementace klienta. Jsou zde rozebírány všechny v práci využité pluginy.

Jedenáctá kapitola se následovně zabývá popisem komunikace mezi klientem a serverem.

Poslední dvanáctá kapitola shrnuje výsledky práce a její přínosy.

2 Analýza současné nabídky softwarů

Vzhledem k tomu, že součástí cíle této práce je vytvořit aplikaci pro snadné plánování úkolů a událostí, bude tato kapitola věnována analýze stávající nabídky takovýchto softwarů. Těch existuje v současné době na trhu velké množství a lze je dělit podle nejrůznějších kriterii. V následujícím textu uvádíme jejich členění podle tří nejčastěji zohledňovaných kritérií.

2.1 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., 2010a), GNU GPL (Open Source I., 2010b), MIT (Open Source I., 2010c), 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 omezena či žádná. Díky „otevřenému kódu“ je možné aplikace doplnit o potřebnou funkcionalitu, ale při potřebě složitějších funkcí 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. Jejich rozsáhlost neumožňuje snadné doimplementování specifických funkcí. Některé pak navíc dávají přednost vzhledové stránce před stránkou funkční.

- Aplikace placené měsíčně.

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

Některé tyto aplikace nenabízejí větší množství funkcí než konkurence.

- 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.

2.2 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.

  Toto 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. Zde pak hraje roli spíše otázka obtížnosti a nákladnosti vývoje takovéto aplikace.

2.3 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 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 je 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 plánovaní 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 zodpovědnost za poskytnutá data. Oba zmíněné typy ujednání nenutí poskytovatele činit kroky k ochraně dat. Co je zde však překvapující, to je skutečnost, že podobná ujednání lze vidět i u placených služeb.

3 Analýza konkrétních typů softwarů

V této kapitole navážeme na předchozí část práce a naši analýzu zaměříme již na konkrétní typy aplikací, jež se v současné době nacházejí na trhu. Přehled, který zde uvádíme, zahrnuje aplikace, jež jsou podle našeho názoru nějakým způsobem zajímavé nebo jsou běžně rozšířené.

3.1 Volně dostupné webové aplikace

- Google Calendar (Google, 2010):

  Jedná se o aplikaci určenou spíše k soukromým účelům, z tohoto důvodu také nedisponuje větší podporou týmového plánování. Dostačující podporu týmového plánování lze zajistit, ale pouze za cenu ztráty soukromí uživatelů. (V průběhu psaní této bakalářské práce byly v této aplikaci některé formy týmové spolupráce zavedeny – pozn. autora).

- Yahoo! Calendar (Yahoo!, 2010):

  Jedná se o funkčně podobný produkt jako je výše uvedený Google Calendar, bez přidané podpory týmového plánování.

- Vcalendar (Vcalendar, 2010):

  Volně dostupná aplikace, zajímavá a poměrně jednoduše naprogramovaná. Neobsahuje však podporu týmového plánování. Ani struktura ukládání dat není příliš vhodná pro přidání funkcí podpory týmového plánování.

- Ostatní webové kalendáře například PHP iCalendar(PHP iCalendar,2010), nebo phpScheduleIt(phpScheduleIt, 2010):

  Volně dostupné aplikace, nevyznačující se něčím příliš zajímavým, od ukládání dat až po způsob a kvalitu zobrazení. Dle názoru autora jsou

vhodné spíše do webové prezentace, k prezentování událostí, než-li k práci s nimi. Některé tyto aplikace různým způsobem využívají protokol CalDEV specifikovaný v RFC 4791 (Daboo, et al., 2007). Který probereme dále v textu.

3.2 Placené webové aplikace (poskytující službu)

Je velmi obtížné popsat takovéto placené aplikace, neboť ty, jež mají volně dostupné demo nebo trial účty, bývají zpravidla nezajímavé, ničím se nelišící od kvalitních volně dostupných aplikací. A ty, které nabízejí demo účty na vyžádání, projevily nechuť poskytnout tyto účty ke studijním účelům. Z dostupných informací jde především o aplikace zahrnující celé kancelářské balíky.

3.3 Dostupné desktopové aplikace

- Microsoft Office Outlook (Microsoft Office, 2010):

  Desktopová aplikace přímo dostupná pouze pro operační systémy Microsoftu. Zahrnuje mailového klienta, kalendář a „úkolníček“. Při použití Microsoft Exchange server. Podporuje týmové plánování i publikaci volného času.

- Lightning pro Thunderbird(Lightning, 2010):

  Jedná se o plug-in do mailového klienta Thunderbird, je pod MPL/GPL/LGPL licencí, tudíž je modifikovatelný a zdrojové kódy jsou dostupné. Nemá však podporu týmového plánování. Tu je možné simulovat povolením přístupu do jednotlivých kalendářů (vrstev). Je dostupný pro všechny využívanější operační systémy.

- IBM Lotus (Lotus Softwere, 2010):

  Jedná se o placený „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 komplet funkcionalitu, kterou měla tato komponenta. Dle našeho hodnocení se jedná o rozsáhlý, kvalitně zpracovaný a smyslně provázaný balík aplikací podporující týmové plánování na vysoké úrovni.

3.4 Shrnutí

Jak vyplývá z výše uvedeného, v současné době existuje mnoho plánovacích aplikací, které se svého úkolu zhostily s různou měrou úspěšnosti. Řada z nich implementuje zajímavé a unikátní funkce. Ať k přístupu k týmovému plánování, nebo k přístupu propojení úkolů a událostí. Implementováním všech těchto funkcí a dodáním nových by mohl vzniknout optimální plánovací software. Charakteristice takového softwaru je věnována následující kapitola.

4 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. Poté jsme učinili závěr, že v podobě, v jaké se tyto aplikace nyní nacházejí, neodpovídají našim potřebám. V této kapitole si proto vymezíme ideální plánovací aplikaci. Jaké charakteristiky a atributy by tedy taková aplikace měla mít? Odpovědi poskytuje následující text.

4.1 Uživatelská přívětivost

O tom, že každá aplikace by měla být uživatelsky přívětivá, panuje obecná shoda. Pokud se ale jedná o aplikaci, kterou by měl uživatel využívat denně, pak je velmi vhodné, aby byla rovněž přehledná, snadno se ovládala a současně byla srozumitelně konfigurovatelná. Pro kvalitní využití plánovací aplikace se předpokládá, že ji bude uživatel navštěvovat mnohokrát denně a konzultovat s aplikací stávající čas. Aplikace by tedy 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 různou grafikou dialogů, aby uživatel snadno rozlišil konkrétní dialog,
- Aplikace by měla mít různou grafiku potvrzovacích dialogů, aby

uživatel snáze rozlišil „důležité a nedůležité“ potvrzení. Míra důležitosti se odvíjí od důležitosti položky, kterou potvrzuje a zda manipulace s danou položkou neovlivní i jiné uživatele,

- Obsahovat předvyplňování dat, která bývají většinou stejná, aby uživatel nebyl zatěžován jejich neustále se opakujícím vyplňováním,
- Uživatel po přihlášení by měl mít aplikaci nastavenou stejně, jako když ji při regulérním odhlášení opustil. Nastavení viditelnosti vrstev kalendáře, nastavený pohled kalendáře (denní/týdenní/měsíční se zobrazením grafické/textové),
- V místech, kde se zobrazuje velký seznam dat, by uživatel měl mít možnost tato data snadno filtrovat a snadno vyhledávat.

4.2 Bezpečnost a soukromí

Aplikace, jíž svěřujeme své osobní údaje i údaje našich obchodních partnerů, by měla být bezpečná. Proti všem možným útokům, jak z fyzického světa, tak ze světa binárního. Proto by komunikace měla probíhat v šifrované podobě. Přihlašování k jednotlivým účtům by mělo být realizováno přes bezpečné heslo, či jiný bezpečnostní prvek, který jistě rozezná 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:

- Přes bezpečnostní prvek (heslo, čipová karta atd.) jistě rozeznat unikátního uživatele. Při opakovaném pokusu o neoprávněné přihlášení informovat o této události zodpovědné osoby, či zablokovat účet do doby vyřešení této situace,
- Komunikace by měla být šifrována, aby ji nebylo možné 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 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.

4.3 Klíčová slova

Předtím než začneme uvažovat, jak budeme naše plány zobrazovat a pracovat s nimi, měli bychom si nejdříve určit, z čeho se naše plány vůbec skládají. Rovněž bychom měli být schopni používat relevantní termíny. Naplnění tohoto kroku zajišťují následující odstavce.

Pokud plánujeme schůzku či návštěvu divadla, známe čas začátku a dokážeme odhadnout dobu trvání, musíme si vyhradit dostatečně dlouhou dobu v určitém čase. Takto vyhrazený čas můžeme nazvat **událostí**.

Máme-li vykonat určitou činnost do určité doby a nezáleží-li nám přitom na tom, kdy tuto činnost vykonáme, pokud jí dokončíme do dané doby, takovou to činnost pak nazveme **úkolem**.

Úkol i událost mohou byt spojeny s jednou činností. Úkol může být například naučit se látku na zkoušku, událostí je pak tato samotná zkouška začínající od určité hodiny. V tom případě řekneme, že úkol je **spjatý** s událostí.

Jeden úkol může také záviset na druhém úkolu. Dle úspěchu prvního úkolu se poté modifikují vlastnosti druhého úkolu. Proto o tomto spojení řekneme, že úkoly jsou na sobě **závislé** nebo, že jsou na sebe **vázané**. Vázané však nemusejí být pouze úkoly, ale mohou to být rovněž i události.

Stálé skupině lidí, která potřebuje koordinovat akce v závislosti na ostatních členech, budeme pro účely této práce říkat **tým**.

Funkce aplikace, které pracují s plány více osob potom nazveme **týmové funkce**.

O některých akcích víme, že jsou důležitější než jiné, proto je vhodné zavést stupnici důležitosti, takovéto stupnici budeme říkat **priorita** akce.

Časová vytíženost je ukazatel, udávající kolik procent minut v hodině je volných. Tento indikátor se používá pro zobrazení rezervovaného času pro úkoly.

4.4 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.

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 na týdenní a ještě později na malé diáře, kde byly rozepsány jednotlivé dny, a to dokonce na hodiny. Důležité je 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. Je to pochopitelné, pokud plánujeme událost, plánujeme ji s další osobou, proto na ni potřebujeme mít také příslušný kontakt. Proč tedy institut papírového diáře, který již úspěšně funguje mnoho let nepřevést do digitální podoby.

Mnoho lidí si píše na papírky poznámky, aby na něco nezapomněli. Otázkou je, jestli mnoho takovýchto poznámek nemá charakter úkolu s nízkou důležitosti. Některé ano, jako například vyřídit, vytisknout, atd. Některé jsou sice úkoly, ale nikoliv s nízkou prioritou. Jedná se o rutinní úkoly, které děláme, aniž bychom si je museli poznamenávat.

Poznamenáváme si pouze obsah tohoto úkolu, který se mění jen v detailech. Ukázkový příklad je nákup potravin. Děláme jej téměř každý den, má specifikace úkolu, ale jako úkol ho nebereme. Poznamenáváme si pouze, co máme v příslušný den nakoupit. Má naše aplikace právo a moc nutit uživatele, ať přestane používat poznámky a místo toho začne používat technicky správnější úkoly? Nebylo by smysluplné ničit něco, co funguje a zásadně ničemu nevadí. Přidáme-li poznámkám nevyžadované atributy jako má úkol, může uživatel sám dojít k názoru, že to jsou úkoly.

Korespondence byla 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 mailu, se stává e-mail jednou z hlavních komunikačních cest. Mnohé schůzky a jednání jsou 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 jinou rozsáhlejší institucí můžeme komunikovat s více různými lidmi, kteří s námi řeší stejný problém. Bylo by žádoucí mít možnost vybrat maily pouze z konkrétní instituce.

Považujeme-li e-mail za hlavní komunikační cestu, potom VOIP a IM musíme považovat ze vedlejší komunikační cesty. Máme prostor pro zapojení i těchto vedlejších komunikačních cest. Jedním kliknutím na telefon v kartě adresáře se okamžitě vytáčí 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 končit neměla. Později můžeme mít potřebu zjišťovat, jestli jsme se dané události zúčastnili, a jak dopadla. Z některých typů události (jednání, schůze, konference) vznikne další materiál jako je zápis 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: harmonogram akce, pozvánku na akci, 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 je v jiné složce, protože v plánovači je okamžitě uvidí.

Co bylo výše napsáno o událostech, platí stejně i o úkolech, kde můžeme mít zájem zaznamenávat konkrétní rozsáhlejší zadání, vyhodnocení úkolu a nebo dokonce i samotné jeho řešení.

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 nikoliv. 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 na jiné osoby. Potřebujeme informace o tom, zda ony 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. Nastává tak duplicita některých akcí, což může vést k nechuti uživatelů zapisovat dvě stejné akce do různých diářů. To může mít pak další následky, například v podobě zapomenutí zaznamenat akci do jednoho z nich. 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, ale 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, ale 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, jako nadbytečné.

Máme-li naplánované akce, zjistíme, že o některých událostech chceme mít přehled, ale nechceme se jich účastnit. Proto je dobré tyto události rozlišovat. Také je dobré znemožnit přidání akce, které se chceme zúčastnit, pokud by časově překrývala jinou událost, které se chceme zúčastnit.

Při využívání diáře a zaznamenávání mnoha akcí se 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, půjde 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, vedoucímu pracovníkovi).

Hledáme-li vhodný čas na schůzku s jinou osobou, která nebude používat naší aplikaci, můžeme jí poskytnout svůj kalendář, kde budou zvýrazněna pouze místa, kde máme volno. Pokud jdeme na pracovní jednání, je vhodné aby kolegové, sekretářka nebo vedoucí věděli/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 ze seznamů. Seznam by měl být ve formátu umožňujícím takovýto přístup (XML, nebo jiný přesně definovaný formát). Příklady těchto seznamů mohou byt: seznam státních svátků, harmonogram školního roku atd.

Uživatel chce mít aplikaci co nejjednodušší, a v našem životě děláme mnoho akci 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 nedostaneme čtrnáctého, 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 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, kde se uživatel rozhodl této události zúčastnit. Aby aplikace byla co nejpřívětivější, stačí když uživatel zadá místo konaní akce, popřípadě i místo, kde se pravděpodobně bude nacházet než pojede na danou událost. Aplikace pak sama vyhodnotí dobu trvání cestování. K tomu bude ještě zapotřebí znát případný způsob dopravy (MHD, pěšky, autem atd.). 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.

Některé události nekončí pouze jejich konáním. Především pokud organizujeme jednání, konferenci atd., potřebujeme si vyhradit určitý čas na přípravu (místa konání atd.) a také čas po konci na dokončení, zhodnocení atd.

K plánování patří i úkoly. Můžeme předpokládat, že úkolů bude mnoho. Je proto nutné zavést filtrování a vyhledávání úkolů. Filtrování by mělo být reprezentováno seřazením podle jednoho z určujících faktorů úkolu (např. časová vytíženost, datum konce úkolu, atd.). Vyhledávání muže být vytvořeno fulltextovým vyhledáváním přes název úkolu a jeho popis.

O rozdílu mezi úkolem a událostí jsme se již dříve v textu zmiňovali. Nesmíme při plánování zapomenout, že vypracování úkolu trvá určitý netriviální čas. Tento čas, i když není přesně určen na vypracování daného úkolu, musí být rezervován, a to proto, aby nemohla nastat situace, kdy je plánování zmařeno nedostatkem času na vypracování úkolu.

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 samostatné plánování, ale potřebujeme plánovat pouze svůj kalendář a 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.

4.5 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,
- E-mail, VOIP a IM,
- Kontrola časové vytíženosti,
- 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.

5 Technická realizace ideální plánovací aplikace

Po určení toho, co by měla ideální plánovací aplikace umět, je třeba následně uvažovat o způsobech její technické realizace. Těmi se zabývá tato kapitola.

5.1 Volba síťové architektury

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

Nabízí se možnost Client-queue-client. Funkce serveru jsou 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 (Wikipedie, 2010a). 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 pracování s vrstvou, kterou může vidět více uživatelů.

Druhá možnost je architektura client-server. 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 (Wikipedie, 2010a). Výhodou této architektury pro naši aplikaci je především dostupnost všech potřebných dat na jednom místě. Tato výhoda se ale také může 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.

5.2 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 implementací vrstvy jiné 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. Jedná se většinou o databázový systém, nemusí to však být pravidlem.

5.3 Datová vrstva

Data chceme 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, a proto se do ní autor této práce nebude pouštět. Pro naše účely se tedy omezíme na konstatování, že pro ideální plánovací aplikaci je žádoucí rychlý přístup k datům dle klíče a rozsahu klíče.

Autorův práce se domnívá, že relační databáze v tomto případě nepředstavuje špatnou volbu.

5.4 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 databáze. 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, které budou v datové vrstvě uloženy.

Odpověď není snadná. Zde záleží už na konkrétní implementaci použitého nástroje pro prezentační vrstvu a v neposlední řadě i nástroje pro datovou vrstvou. Na základě výše uvedeného se zdá, že je nejvhodnější přenášet mezi vrstvami minimální obsah potřebných dat.

5.5 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 (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 zase 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.

5.6 Bezpečnost

V naší ideální plánovací aplikaci hraje bezpečnost značnou roli.

Důležitá je především bezpečnost komunikace mezi jednotlivými součástmi aplikace. Nesmí se ale opomenout také bezpečnost fyzická (tj. serverové části). A to, jak před úmyslným poškozením, tak proti poškození náhodnému. Rozebírat možnosti fyzického zabezpečení serverové části je nad rámec této práce, proto se na místo toho pojďme věnovat bezpečnosti komunikace jednotlivých komponent aplikace.

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 (Sun Microsystems, 2010).

SSL se dá využít v těchto dvou stavech:

- **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) (Sun Microsystems, 2010).

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

5.7 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 popis práce s nimi) 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.[1]

Výše zmíněné formáty a protokoly neumožňují snadnou implementaci systému oprávnění, kterou požadujeme. Z tohoto důvodu nejsou pro naše potřeby zcela vhodné. Jejich přehled zde uvádíme pouze pro získání komplexnosti přehledu.

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

6 Logický návrh

V aplikaci se především pracuje se základními komponentami, jimiž jsou: událost (event), úkol (task), vrstva (layer), uživatel (user), skupina (group) a zpráva (message).

Komponenty obsahují logické hodnoty vyplývající z jejich názvů. Konkrétní instance jednotlivých komponent musíme být schopni jednoznačně rozeznat, k tomu slouží číselný identifikátor. Jednotlivé komponenty jsou v aplikaci specifikovány objekty. Tyto objekty korespondují s tabulkami v naší relační databázi. Některé objekty jsou spojené přímo a jiné pomocí kolekční tabulky.

6.1 Návrh přístupu k právům

Problematika viditelnosti jednotlivých úkolů a událostí je jedním z cílů práce, proto si zde detailněji popíšeme návrh jednotlivých práv.

Právo vytvářet uživatele a skupiny má pouze uživatel, který má přiřazená administrátorská práva. Uživatel s administrátorskými právy má přehled o všech skupinách a všech uživatelích. (Ve smyslu správy nad údaji o uživatelích a účasti v jednotlivých skupinách. Nikoli ve smyslu přehledu o uživatelských aktivitách).

Protože zastáváme restriktivní přístup k právům, omezujeme i právo přístupu k informacím jednotlivých uživatelů. Toto omezení budeme aplikovat tak přísně, že omezíme přístup k informacím o uživatelích i ostatním uživatelům. Pro umožnění práce v týmu uživatelů stanovíme pravidla, za jakých okolností bude možno informace o uživateli získat. Práce v týmu je v aplikaci realizována skupinou, proto právo k přístupu k informacím konkrétního uživatele, bude mít jen uživatel, přiřazen do stejné skupiny.

Takto restriktivní přístup je v některých případech značně omezující

(např. uživatel není přiřazen do stejné skupiny s jiným uživatelem a rád by mu poskytl ke sdílení některé své události). Takovýto typ problému lze v naší aplikaci vyřešit vytvořením nové skupiny, do které budou uživatelé přiřazeni. Toto řešení však není systémově vhodné.

Vhodným řešením by bylo použití řetězce (na způsob veřejného klíče) s nutností potvrzení druhého uživatele.

Hlavním nositelem práv k informacím o akcích (úkol, událost) je vrstva.

Uživatel vůči vrstvě může mít tato práva (seřazeny hierarchicky, každé další právo obsahuje i možnosti toho předešlého):

- viditelnost – uživatel má informace o dané vrstvě, ale nemůže do této vrstvy přidávat další akce, ani s již zadanými akcemi manipulovat,
- spolupráce - uživatel může do dané vrstvy přidávat nové akce a své již zadané akce ve vrstvě upravovat,
- správa – uživatel může upravovat nastavení dané vrstvy.

Další nejemnější členění práv k informacím o akcích, nese daná akce s dvěma nastaveními.

- zveřejnění – hodnota tohoto nastavení určuje způsob zveřejnění informací o akci ve veřejném kalendáři. Toto nastavení je navrženo z důvodu možnosti generování veřejných kalendářů ze všech vrstev daného uživatele,
- zveřejnění do skupiny – hodnota tohoto nastavení určuje způsob zveřejnění informací do vrstvy, v které je akce přiřazena. Toto nastavení je navrženo pro možnost neposkytnutí informací o akci ostatním uživatelům přiřazeným k vrstvě.

Výše uvedená formulace „zveřejnění do skupiny“ může být zavádějící název S přidáním možnosti připojovat k vrstvám uživatele z jiných skupin pak tato formulace ztrácí výstižnost.

Zveřejnění (veřejné i do skupin) může být nastaveno na:

- žádné – informace o dané akci nebude ostatním uživatelům přístupná,
- zaneprázdněn – ostatním uživatelům bude přístupná pouze informace, že v dané době máte blíže nespecifikovanou akci,
- zaneprázdněn kde – ostatním uživatelům bude přístupná pouze informace, že v dané době máte akci specifikovanou pouze místem konání,
- základní informace - ostatním uživatelům bude přístupná informace o názvu, místě konání a prioritě akce,
- detailní informace – ostatním uživatelům budou přístupné všechny zadané informace o akci,
- povolit i upravovat – ostatním uživatelům budou přístupné všechny zadané informace o akci s možností danou akci upravovat.

6.2 Návrh přístupu k vrstvě

Jako vrstva se dá považovat diář, který využíváme, pouze na určité věci. Takto k vrstvě také budeme přistupovat jedná se o prostor, kam vkládáme pouze spolu související akce. Tento prostor můžeme otevřít i jiným uživatelům. To učiníme tím že vrstvu nasdílíme jiným uživatelům pod určitými právy, popsané v předchozí podkapitole.

7 Implementační rozhodnutí

Vyvinout ideální plánovací aplikaci je nad rámec bakalářské práce. Proto přistoupíme k vývoji 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 naší aplikaci jsme již popisovali, 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, 2009). Proto by zobrazení naší aplikace mělo být ve všech typech stejné. Toto rozhodnutí nám dovolí splnit požadavek na interaktivní uživatelské rozhraní, který byl popsán v kapitole o uživatelské přívětivosti ideální plánovací aplikace.

Třetím rozhodnutím 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 5.2 s rozšířením PDO (*PHP Data Objects)* pro práci s datovou vrstvou, která bude reprezentována relační databází PostgresSql verze 8.2. Tuto volbu jsem provedli z důvodu vhodnosti pro vývoj webových aplikací, kterou náš klient je. Toto ale neznamená, že 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.

8 Implementace serverové části

Server přijímá komunikaci od klienta na čtyřech vstupních bodech, specifikovatelných v konfiguraci. 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:

- ajaxEvent – zpracovává požadavky na události,
- ajaxTask – zpracovává požadavky na úkoly,
- ajaxLayer – zpracovává požadavky na vrstvu,
- ajaxAdmin – zpracovává požadavky na autorizaci, uživatele, zprávy.

Ke zpracování požadavků serverová část aplikace využívá jednotlivé třídy.

8.1 Práce s událostí

Událost je v aplikační vrstvě reprezentována třídou Event a v datové vrstvě reprezentována tabulkou event. Proměnné třídy Event korespondují se sloupci tabulky event. Třída Event poskytuje veřejné metody, které umožňují pracovat s událostmi v požadované míře. Jedná se především o následující metody:

- loadEvent – načte událost,
- newEvent – vytvoří novou událost,
- editEvent – upravuje událost,
- deleteEvent – maže událost,
- moveEvent – upravuje začátek a konec události.

Pro práci s více událostmi (především pak načítaní více událostí) slouží

třída eventRow. Tato třída umožňuje načíst kolekci událostí v rozmezí dvou dat.

8.2 Práce s úkolem

Úkol je v aplikační vrstvě reprezentován třídou Task a v datově vrstvě pak tabulkou task. Proměněné třídy Task korespondují se sloupci tabulky task. Třída Task poskytuje pro práci s úkoly tyto veřejné metody:

- loadTask – načte úkol,
- newTask – vytvoří nový úkol,
- editTask – upravuje stávající úkol,
- deleteTask – maže z databáze úkol,
- chnageLayer – mění přiřazení úkolu k vrstvě,
- changePriority – mění prioritu úkolu.

Pro práci s více úkoly slouží třída taskRow. Tato třída umožňuje načíst kolekci úkolů dle zadaného filtru.

8.3 Práce s vrstvou

Vrstva je v aplikační vrstvě reprezentována třídou Layer a v datové vrstvě tabulkou layer a tabulkou inLayer. Proměnné třídy Layer korespondují se sloupci tabulky layer. A tabulka inlayer obsahuje informace specifické pro spojení uživatele s vrstvou (práva, barva, atd.). Třída Layer poskytuje pro práci s vrstvou tyto veřejné metody:

- loadLayer – načte vrstvu,
- newLayer – vytvoří novou vrstvu,
- editLayer – upravuje vrstvu,
- deleteLayer – přiřazuje příznak o odstranění přiřazení uživatele k vrstvě (probíhá v tabulce inLayer),

- changeColor – upravuje barvu vrstvy pro uživatele (probíhá v tabulce inLayer).

Pro načtení všech vrstev přihlášeného uživatele slouží třída layerRow.

8.4 Práce s uživatelskými účty

Uživatelský účet je v aplikační vrstvě reprezentován třídou User a v datové vrstvě je reprezentován tabulkou user. Proměnné třídy User korespondují se sloupci tabulky user. Třída User poskytuje pro práci s uživatelskými účty tyto veřejné metody:

- newUser – vytváří nového uživatele,
- loadUser - načte uživatele,
- selfEdit – upravuje informace o přihlášeném uživateli,
- getAllUser – poskytne číselník (identifikátor a jméno) všech uživatelů.

8.5 Práce se skupinou

Skupina je v aplikační vrstvě reprezentována třídou Group a v datové vrstvě reprezentována tabulkami group a ingroup. Proměnné třídy Group korespondují se sloupci tabulky group. Tabulka ingroup obsahuje informace o přiřazení uživatele do skupiny a jeho práva ve skupině. Třída Group poskytuje pro práci se skupinou tyto veřejné metody:

- getInfoGroup – vrací informace o skupinách, do nichž je přihlášený uživatel přiřazen,
- incGroup – vytváří a upravuje vrstvy, zároveň do vrstev přiřazuje uživatele,
- getAdminGroup – pokud má přihlášený uživatel administrátorská práva, vrací seznam všech skupin.

8.6 Práce se závislostí

Události i úkoly na sobě mohou být závislé. Tuto závislost v aplikační

vrstvě reprezentuje třída dependence a v datové vrstvě tabulka dependence. Třída dependence má tyto veřejné metody:

- newDependence – pokud neexistuje žádná předchozí závislost tohoto úkolu, vytváří novou závislost, pokud existuje předchozí závislost tak jí pouze upravuje,
- getDependence - pokud existuje závislost, tak jí vrací, jinak vrací prázdný řetězec.

8.7 Práce se zprávami

Aby byl uživatel informován o nabídce akce ke sdílení, nebo změně sdílené akce, je k dispozici třída message vykonávající tuto činnost v aplikační vrstvě, v datové vrstvě je třída message reprezentovaná tabulkami message a mesagetitle, kde sloupce tabulky message korespondují s proměnnými třídy message. Tabulka mesagetitle je pouhý číselník, který obsahuje text zprávy.

Třída message má tyto veřejné metody:

- dataMessage – vrací informace o zprávě,
- acceptMessage – informuje, že uživatel danou zprávu (nabídku sdílení) přijal,
- rejceptMessage – informuje, že uživatel danou zprávu (nabídku sdílení) odmítl.

8.8 Práce s databází

K přístupu k databázi je v celé aplikaci využita třída my_db, která obsahuje PHP rozšíření PDO (PHP Data Objects). Jedná se o abstrakci nad databází. V naší aplikaci je tato abstrakce využita na odchytávání výjimek z databáze a převod chybových hlášení do uživatelsky pochopitelné formy.

Třída my_db obsahuje tyto veřejné metody:

- query – položí sql dotaz na databázi, vrací odpověď databáze,

- getMessage – vrátí případnou chybu databáze upravenou do uživatelsky pochopitelné formy.

8.9 Obecně o implementovaných třídách

Třídy obsahují veřejnou metodu *result()*. Tato metoda vrací řetězec ve formátu JSON (JavaScript Object Notation). Jeho obsah se mění dle příznaku lazy, který obsahuje většina tříd a je definován parametrem konstruktoru. Ten může nabývat hodnot:

- 0/„small“ - třída s tímto příznakem je využívána pouze na požadavky, které informují server o akcích uživatele. Třída ve funkci *result()* vrací pouze hodnotu *statusClass* a identifikátor konkrétní instance třídy,
- 1/„long“ - třída s tímto příznakem korektně pracuje se všemi vnitřními proměnnými třídy, na funkci *result()* vrací všechny naplněné hodnoty,
- 2/„full“ - třída s tímto příznakem též korektně pracuje se všemi proměnnými třídy, ale na funkci *result()* vrací všechny hodnoty.

Třídy dále obsahují příznak ClassStat, tento příznak určuje stav třídy vůči klientovi nebo k databázi. Nabývá těchto hodnot:

- „ok“ – požadavek proběhl v pořádku,
- „error“ - požadavek nebyl vykonán,
- „db“ - hodnoty třídy jsou stejné jako hodnoty databáze,
- „new“ - jsou zadané nové hodnoty, které dosud nejsou v databázi,
- „edit“ - jsou zadané nové hodnoty upravující stávající hodnoty v databázi identifikované identifikátorem (id),
- „delete“ - je požadováno vymazání dat z databáze, mazání probíhá přes hodnotu identifikátoru (id).

Třída též obsahuje proměnnou errorMessagess. Jedná se o pole řetězců chybových hlášení pro uživatele. Tato proměnná není prázdná pouze v případě, že příznak classStat je „error“.

Třídy obsahují neveřejnou metodu saveClass(), která slouží k vykonání databázové operace dle příznaků classStat („new“, „edit“ a „delete“).

9 Databáze

Jak již bylo napsáno, databázová část je tvořena databázovým serverem PostgresSQL. Část logiky z aplikace byla převedena do této vrstvy. Jedná se především o testy integrity dat a testy přístupu k určitým datům. V této kapitole nebudeme popisovat jednotlivé tabulky a jejich význam, popíšeme zde pouze zajímavé funkce a triggery. Konkrétní návrh databáze je zobrazen v příloze C.

9.1 Funkce triggerů

Triggery jsou v naší databázi na tabulkách task a event. Vykonávají následující činnosti:

- Kontrola korektnosti vkládaných dat,
- Kontrola smysluplnosti vkládaných dat,
- Kontrola práv uživatele upravovat/vytvářet nové záznamy,
- Případné vytváření zpráv pro uživatele, kterým byla akce nabídnuta,
- Případné vytváření zpráv pro uživatele, sdílející upravované/mazané sdílené akce.

9.2 Funkce

Z důvodu neexistujících triggerů na výběr z databáze. Jsme rozhodli přistoupit k vytvoření databázových funkcí na výběr záznamu z tabulek task a event (tyto tabulky reprezentují logické informace o úkolu a události). Tyto funkce modifikují vrácená data dle hodnoty u možnosti zveřejnění. Jedná se o funkce:

- getEventRow(*integer, timestam, timestamp*) – funkce vrací všechny události přístupné danému uživateli (specifikovanému prvním parametrem) v časovém rozmezí (specifikovaném druhým a třetím

parametrem). Tyto události jsou na výstupu modifikovány, dle práv přístupu daného uživatele k informacím (možnosti zveřejnění) dané události,

- taskbyusertolayer(*integer*) – funkce vrací všechny úkoly přístupné danému uživateli (specifikovanému parametrem). Tyto úkoly jsou na výstupu modifikovány dle práv přístupu daného uživatele k informacím (možnosti zveřejnění) daného úkolu.

K datům databáze chceme přistupovat především přes námi specifikované funkce, proto byly vytvořeny další funkce. Pro příklad zde zmíníme tyto:

- deleteevent(*integer,integer*) – uživateli (specifikovanému prvním parametrem) vymaže záznam o události (specifikované druhým parametrem),
- changecolor(*integer, integer, varchar*) – uživateli (specifikovanému prvním parametrem) změní barvu dané vrstvy (specifikované druhým parametrem) na barvu (specifikovanou třetím parametrem ve formátu hexadecimálního zápisu s mřížkou na začátku).

10 Implementace klienta

Klient je vystavěn nad grafický webový prohlížeč s podporou JavaScriptu pomocí JavaScriptového frameworku jQuery. Tomuto framework jsem dali přednost před jinými alternativami (např. Prototype, Dtojo, Ext), a to z důvodu dobrých výsledků ve srovnávacích testech (Velichkov, 2009) a v neposlední řadě také proto, že plugin kalendáře, který používáme, je v tomto frameworku napsaný.

10.1 Použité pluginy

Ve vývoji klientské části jsme využili množství pluginů napsaných pro jQuery. Jejich seznam a stručný popis jsou uvedeny zde:

- FullCalendar (Shaw A, 2010a) – zobrazuje a pracuje se zobrazením kalendáře (stručný popis je uveden v samostatné podkapitole),
- colorPicker (Petre S, 2009) – umožňuje nám vybírat barvu přiřazenou k vrstvě pomocí podobného dialogu známého z deskopových aplikací,
- IdleChecker (Lint K, 2009) – zajišťuje automatické odhlášení dlouhou dobu neaktivních uživatelů,
- jQuery.datapicker – vytváří a spravuje malý kalendář sloužící k výběru dat v dialozích a v hlavním pohledu nastavení dat pro pohled poskytující pluginem FullCalendar,
- jQuery.diaog – vytváří a spravuje dialogy, využitý na dialogy v klientovi,
- jQuery.button – vytváří a spravuje definovatelná tlačítka, využitý na tlačítka v klientovi,
- jQuery.slider – vytváří a spravuje posuvník, který je použit v dialozích úkolů, na možnost nastavení hodnoty vypracovanosti úkolu,
- jQuery.tabs – vytváří a spravuje záložky, záložky jsou použity v

některých dialozích klienta,

- jQuery.AJAX – vytváří komunikaci AJAX (*Asynchronous JavaScript and XML)*.

V aplikaci jsme také vytvořil vlastní pluginy:

- taksManager – vytváří a spravuje seznam úkolů,
- LayerManager – odvozen od pluginu taskManager, vytváří a spravuje seznam vrstev.

10.2 Plugin FullCalendar

Tomuto pluginu pro zobrazení kalendáře jsme dali přednost před možnými alternativami z důvodů kvalitního kódu, dobře napsané dokumentace, skinovatelnosti a v neposlední řadě i také kvůli dobré komunitní podpoře.

Tento plugin má velké množství nastavení pro lokalizaci (formát data, formát času atd.).

Na akce uživatele v tomto pluginu reaguji callback funkce, my zde využíváme tyto:

- select (*start, end, allDay*) – reaguje na akci výběru uživatele (provedeným kliknutím a možným tažením).
  - *start* – JavaScriptový objekt date reprezentující začátek místa výběru,
  - *end* – JavaScriptový objekt date reprezentující konec místa výběru,
  - *allDay* – zda byl výběr proveden v pohledu podporující celodenní pohled.
- EventDrop (*event, dayDelta, minuteDelta, allDay, revertFunc*) – reaguje na akci drag&drop provedenou uživatelem na události.
  - *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 znázorňující událost) provedenou uživatelem na události.
  - *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.
  - *calEvent* – konkrétní instance události, na kterou bylo kliknuto.
- viewDisplay(*view*) – tato funkce reaguje na změnu pohledu. My toto volání využíváme k přenesení změny zobrazení na datapicker, aby mohl zobrazovat datum pohledu.
  - *view* – aktuální instance pohledu.

Detailní dokumentace pluginu FullCalendary (Shaw A, 2010b)

10.3 Vlastní pluginy

Pro práci s vrstvou a úkolem jsme si vyvinuli pluginy LayerManager a

taskManager.

TaskManger má dvě veřejné metody, a to:

- reload () – znovunačte úkoly odpovídající filtru ze serveru,
- imputTask(*id*, *reform*) – vyvolá vkládací dialog pro úkol, tato metoda je v aplikaci volána při přijmutí nabízeného úkolu z dialogu message.
    - *id* – pokud je null, vytváří se nový úkol, pokud je identifikátor platný, načtou se do dialogu data tohoto uživatele (pokud přihlášený uživatel má k těmto informacím oprávnění),
    - *reform* – příznak akceptování z nabízených událostí (dialog message).

Ostatní metody těchto pluginů jsou neveřejné.

10.4 Změna vzhledu

Za použití CSS (Cascading Style Sheets) (W3C, 2010). Můžeme měnit vzhled uživatelského rozhraní aplikace. Abychom toho docílili, museli jsme ve vyvíjených částech dodržet pravidla jQuery UI Theming (jQuery UI Theming, 2010). Nový vzhled můžeme vytvořit pomocí jQuery ThemeRoller (jQuery UI ThemeRoller, 2010) nebo zde nalézt již vytvořené 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.

11 Komunikace klient-server

Klient komunikuje se serverem ve formátu JSON (JavaScript Object Notation). Ke komunikaci využívá plugin jQuery.AJAX s nastavenou synchronní komunikací.

Komunikace probíhá v modelu:

- klient odešle požadavek ve formátu JSON, s hodnotou operation (specifikující konkrétní požadavek) dále pak hodnoty, které jsou nutné ke zpracování daného požadavku,
- server obdrží daný požadavek, pracuje ho a opět klientovi odesílá řetězec ve formátu JSON. Tento řetězec obsahuje příznak *ClassStat*, zda byl požadavek korektně vykonán, pokud ne, má tento příznak hodnotu „error“. Také obsahuje ErrorMessages, kde jsou případné varovné zprávy pro uživatele. A dále obsahuje hodnoty dle požadavku,
- Klient se do vyřešení požadavku ze serveru stane neaktivním (použitá synchronní komunikace). Po přijmutí výsledku požadavku se aktivuje a pracuje s daty výsledku.

12 Výsledky a shrnutí práce

Po seznámení se s existujícími plánovacími aplikacemi a rozmyšlením, 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 tento názor lišit uživatel od uživatele.

Druhý stanovený cíl práce, implementovat týmovou plánovací aplikace a s její pomocí prakticky odtestovat návrh systému práv z pohledu uživatele, nebyl zcela dosáhnut. Hlavním důvodem byla absence implementace funkcí, které byly nad rozsah bakalářské práce. I s těmito nedostatky jsme zjistili, že striktně omezovat viditelnost mezi uživateli, kteří nejsou přiřazeni do stejné skupiny, je pro uživatele nepříjemně omezující. Dále jsme také zjistili, že stálé vyplňování hodnot pro konkrétní způsob zveřejnění je pro uživatele obtěžující. Také možnost zveřejňování pouze do vrstvy a veřejně, není dostačující.

Hodnocení systému práv je tedy dle názoru autora takové, že tento systém lze označit jako vyhovující pro týmové plánování v pracovních kolektivech, nikoli už však jako dostatečně vhodný pro účely soukromého využívání.

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. 2010-08-03]. 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. 2010-08-03]. Dostupný z WWW: http://tools.ietf.org/html/rfc5545.

GNU OPERATING SYSTEM. 2007. GNU General Public License. GNU Operating System. 2007. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.gnu.org/licenses/gpl.html.

GOOGLE. 2010. Google Kalendář. Google. 2010. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.google.com/calendar.

JQUERY UI THEMING. 2010. jQuery UI Theming. 2010. [on-line], [cit. 2010-08-02]. Dostupné z WWW: http://jqueryui.com/docs/Theming/API.

JQUERY UI THEMEROLLER. 2010. jQuery UI ThemeRoller. 2010. [on-line], [cit. 2010-08-02]. Dostupné z WWW: http://jqueryui.com/themeroller/.

LINT K. 2009. IdleChecker. Kevin Lint. 2009. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.kevinlint.com/IdleChecker/.

LOTUS SOFTWERE. 2010. Lotus Softwere. IBM. 2010. [on-line], [cit. 2010-08-02]. 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. 2010-08-03]. Dostupný z WWW: http://tools.ietf.org/html/rfc3283.

MICROSOFT OFFICE. 2010. 10 hlavních důvodů, proč vyzkoušet aplikaci Outllook 2010. Microsoft Office. 2010. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://office.microsoft.com/cs-cz/outlook/.

OPEN SOURCE I. 2010a. The BSD License. Open Source Initiative. 2010.

[on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.opensource.org/licenses/bsd-license.php.

OPEN SOURCE I. 2010b. The GNU General Public License. Open Source Initiative. 2010. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.opensource.org/licenses/gpl-2.0.php.

OPEN SOURCE I. 2010c. The MIT License. Open Source Initiative. 2010. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.opensource.org/licenses/mit-license.php

PETER S. 2009. Color Picker. Stefan Peter. 2009. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://www.eyecon.ro/colorpicker/.

PHP ICALENDAR. 2010. PHP iCalendar. PHP iCalendar. 2010. [on-line], [cit. 2010-08-02]. Dostupné z WWW: http://phpicalendar.net/.

PLZAK, et al. 1999. FYI on Questions and Answers. Answers to Commonly Asked "New Internet User" Questions. Network Working Group. 1999. [on-line], [cit. 2010-08-03]. Dostupný z WWW: http://tools.ietf.org/html/rfc2664.

PHPSCHEDULEIT. 2010. PhpScheduleIt. PhpScheduleIt. 2010. [on-line], [cit. 2010-08-03]. Dostupné z WWW: http://php.brickhost.com/index.php.

ROYER, et al. 2005. Calendar Access Protocol (CAP). Network Working Group. 2005. [on-line], [cit. 2010-08-03]. Dostupný z WWW: http://tools.ietf.org/html/rfc4324.

SHAW A. 2010a. FullCalendar Introduction. Adam Shaw. 2010. [on-line], [cit. 2010-08-03]. Dostupný z WWW: http://arshaw.com/fullcalendar/.

SHAW A. 2010b. FullCalendar Documentation. Adam Shaw. 2010. [on-line], [cit. 2010-08-03]. Dostupný z WWW: http://arshaw.com/fullcalendar/docs/.

SUN MICROSYSTEMS. 2010. Introduction to SSL, Sun Microsystems. 2010. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://docs.sun.com/source/816-6156-10/contents.htm.

VCALENDR. 2010. Vcalendar. UltraApps. 2010. [on-line], [cit. 2010-08-02]. Dostupné z WWW: http://www.vcalendar.org/.

VELICHKOV P. 2009. Dojo vs JQuery vs MooTools vs Prototype Performance Comparison. Peter Velichkov. 2009. [on-line], [cit. 2010-08-02]. Dostupný z WWW: http://blog.creonfx.com/javascript/dojo-vs-jquery-vs-mootools-vs-prototype-performance-comparison.

W3C. 2010. Cascading Style Sheets. 2010. [on-line], [cit. 2010-08-03]. Dostupný z WWW: http://www.w3.org/Style/CSS/.

WIKIPEDIE. 2010a. Klient-server. Internetová encyklopedie Wikipedie. 2010. [on-line], [cit. 2010-08-02]. Dostupný z WWW:

http://cs.wikipedia.org/wiki/Klient-server#Srovn.C3.A1n.C3.AD_s_Client-queue-client_architekturou.

YAHOO!. 2010. Yahoo! Calendar. Yahoo. 2010. [on-line], [cit. 2010-08-02]. Dostupné z WWW: http://www.yahoo.com/calendar

Příloha A – uživatelská dokumentace

Instalace:

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))

Změněním konfiguračního souboru configuration.php v domovském adresáři aplikace, kde nastavíme domain name stroje, na kterém aplikace poběží. Zde též můžeme v proměnné $timeOutAfter nastavit možnou dobu neaktivity uživatele před automatickým odhlášením v milisekundách.

Přístup k databázi konfigurujeme v /Class/Setting/db.php, kde máme proměnné $hostname, $username, $password, $dbname, které nastavíme dle hodnot nastavení konkrétní databáze, kam aplikaci instalujeme.

Instalace struktur databáze musí být zajištěna uživatelem s potřebnými oprávněními, proto jsme zvolili instalací pomocí dávkových souborů.

Po vykonání všech úkonů potřebných k aplikaci, se můžeme do aplikace přihlásit do administrátorského účtu aplikace pod uživatelským jménem „admin“ a heslem „demo“.

Uživatelská dokumentace

Podporované prohlížeče Mozilla Firefox 3.2, Internet explore 7.0 a Opera 8.0

Pro využívání aplikace je doporučené minimální rozlišení 1280 x 720 pixelu.

Obrázky v dokumentaci odpovídají zobrazení na platformě Windows XP SP2 za použití prohlížeče Mozillla Firefox verze 3.5. s nastaveným rozlišením 1280 x 960 pixelu.

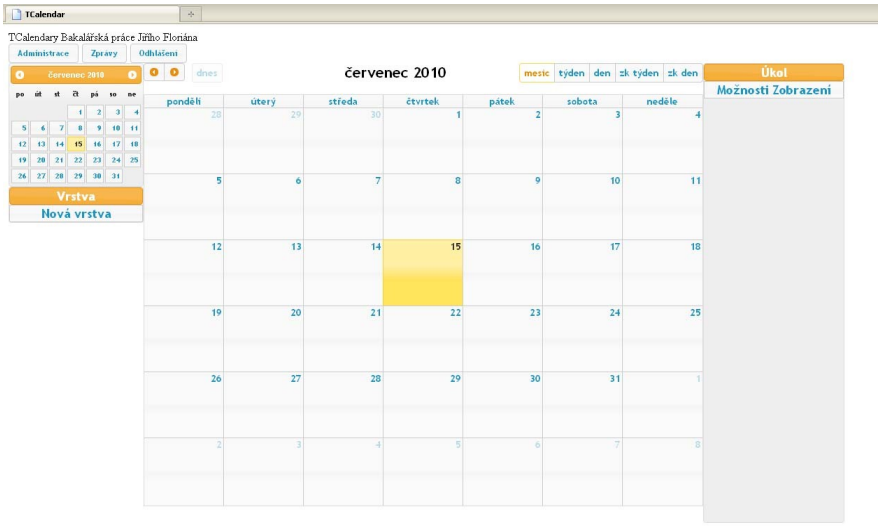
Aplikace byla testována na těchto konfiguracích:

Ubuntu 9.4, Mozilla Firefox verze 3.4 s nastavením rozlišením 1200 x 760 pixelů

Windows XP SP2, Mozilla Firefox verze 3.5. s nastavením rozlišením 1280 x 960 pixelů

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 s zvýrazněním datumu dnešního dne.



obr. 1 - Základní rozhraní

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

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

- zk den – (zkrácený) zobrazení bez osy, úkoly 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 levém horním rohu se nachází tři tlačítka správy klientského účtu Administrace, Zprávy, Odhlášení, která jsou popsány v 2.kapitole.

Pod těmito tlačítky nalezneme zmenšený kalendář, který slouží k rychlému přesunu na hledaný den.

A ještě níže nalezneme sekci Vrstva (a tlačítko **Nová vrstva)**, která nám slouží k zobrazení a ke správě jednotlivých vrstev. Podrobněji probereme v kapitole 3.

V pravé část zobrazení se nachází sekce s názvem Úkol a dvěma tlačítky **Možnosti** a **Zobrazení**, tato část je popsána v kapitole 5.

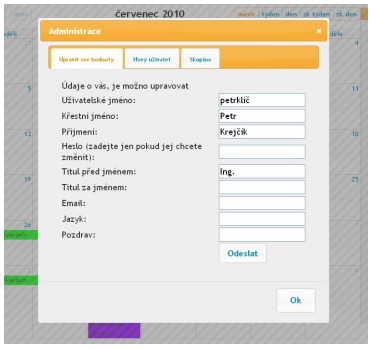
Správa klientského účtu

Ke správě klientského účtu slouží tlačítka Administrace, Zprávy, Odhlášení.

- Administrace

Po stisknutí tlačítka se nám otevře dialog se třemi záložkami. Podle přidělených práv, při vytváření klientského účtu, máme přístup k jedné, dvou nebo třem záložkám.

- **Upravit své hodnoty** (obr. 2), kde nalezneme základní údaje o uživateli tak, jak jsou uloženy v databázi. Zde má uživatel možnost upravit tyto údaje, včetně změny hesla. Provedené změny budou uloženy do databáze stisknutím tlačítka **Odeslat**. O úspěšnosti odeslání budeme informováni prostřednictvím upozorněni.



Obr. 2 – dialog úpravy údajů

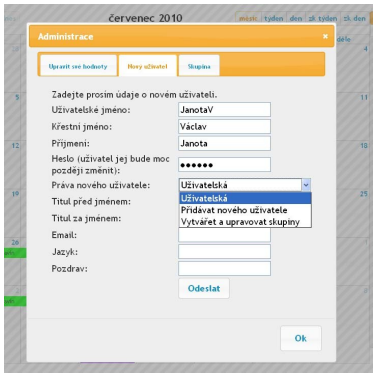
- **Nový uživatel** (obr. 3), tento dialog slouží k přidání nového uživatele. Poté, co vyplníme základní údaje a přiřadíme prvotní heslo, můžeme změnit práva vytvářeného uživatele. K volbě jedné ze tří možných úrovní práv využijeme výběrové menu. Možnost přidělit nejvyšší stupeň práv mají ti uživatelé, kteří sami disponují tímto stupněm.

1.1 **Uživatelské** – přednastavené právo umožňující přístup pouze k dialogu s úpravami vlastního účtu,

1.2 **Přidávat nového uživatele** – právo, které zpřístupní dialog přidávání nového uživatele,

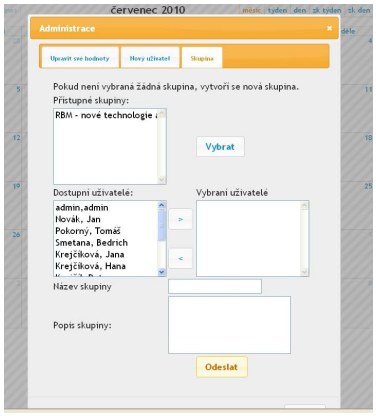
1.3 **Vytvářet a upravovat skupiny** – právo zpřístupňující dialog správy skupin.

K vytvoření nového uživatele dojde po stisknutí tlačítka **Odeslat**, o uložení do databáze budeme informováni.



Obr. 3 – dialog přidání nového uživatele

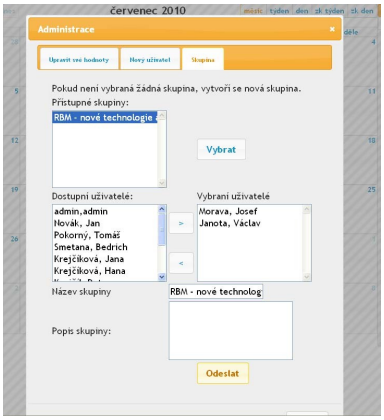
- **Skupina** (obr. 4), tento dialog se zpřístupňuje pouze pokud má uživatel nejvyšší stupeň práv. Slouží ke správě skupin uživatelů. V okně Přístupné skupiny nalezneme všechny skupiny, ke kterým jsme v současné době přiřazeni. V okně Dostupní uživatelé jsou v abecedním pořadí zobrazeni všichni uživatelé vytvoření na tomto serveru. K vyhledání konkrétního uživatele v tomto seznamu můžeme použít scrollbar nebo stisknout klávesu s písmenem, jímž začíná příjmení hledaného uživatele.



Obr. 4 – dialog správy skupin

Ukázkový příklad přidání a odebrání uživatele ze skupiny:

V prvním okně vybereme skupinu, se kterou chceme pracovat a stiskneme tlačítko **Vybra**t. V okně Vybraní uživatelé se nám zobrazí uživatelé, kteří jsou přiřazeni k této skupině (obr. 5). V tomto seznamu vybereme uživatele, kterého chceme odebrat a klikneme na šipku doleva viz. (obr.6).



Obr. 5 – vybrání skupiny

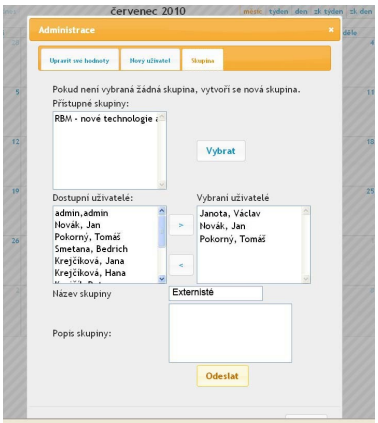
Obr. 6 – odebrání uživatele

Uživatele, kterého chceme do skupiny přidat vybereme ze seznamu Dostupných uživatelů a stisknutím šipky vpravo se provede přidání do skupiny. Pokud jsme spokojeni s výsledky, uložíme změny stisknutím tlačítka **Odeslat**.

Zde můžeme provést změnu jména skupiny a to tak, že skupinu vybereme popis viz. výše a v kolence Název skupiny provedeme požadovanou úpravu a opět změnu uložíme tlačítkem **Odeslat**.

Ukázkový příklad vytvoření nové uživatelské skupiny

Pro vytvoření nové skupiny využijeme zmiňovaný formulář. Vyplníme kolonku Název skupiny a přidáme uživatele, které chceme přidělit do této skupiny (obr. 7). Pro vytvoření skupiny stiskneme tlačítko **Odeslat**.



Obr. 7 – vytvoření nové skupiny

!! Upozornění: z důvodu bezpečnosti nenarušení soukromí jednotlivých skupin muže spravovat skupinu pouze její člen. Proto musíme zabezpečit, aby se vždy ve skupině nacházel alespoň jeden uživatel s těmito právy. !!

- Zprávy

Toto tlačítko slouží ke zobrazení zpráv, které vytváří aplikace v momentě, kdy nám je jiným uživatelem poslán úkol ke sdílení. Vysvětleno bude níže.

- Odhlášení

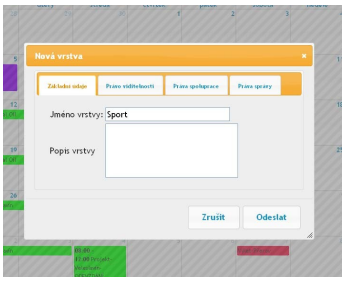
Tlačítko sloužící k odhlášení uživatele. K automatickému odhlášení dochází podle času nastaveného v konfiguraci aplikace na serveru.

2 Vrstvy

Veškeré události a úkoly, vytvářené v této aplikaci, musí náležet některé z vrstev. A proto se nyní seznámíme s fungováním vrstev.

- Nová vrstva

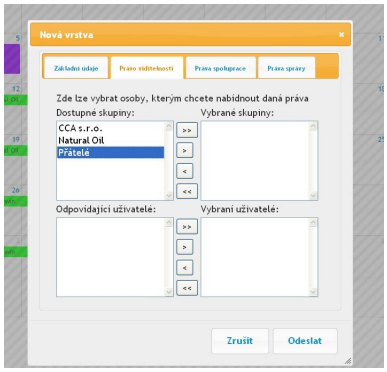
K vytvoření vrstvy slouží tlačítko **Nová vrstva**, po jeho stisknutí se nám zobrazí dialog (obr. 8). V první záložce vyplníme jméno vrstvy a rovněž můžeme připojit její popis. Poté můžeme přidat dalším uživatelům práva, která vybereme pomocí dalších záložek.



Obr. 8 – dialog nové vrstvy

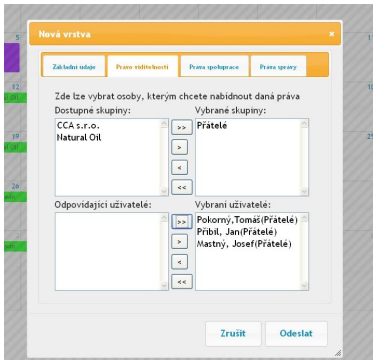
Druhá záložka je určena pro právo viditelnosti. Po vybrání se nám otevře

dialog, ve kterém se zobrazí seznam Dostupných skupin(skupiny ve kterých jsme členy). Abychom přidělili uživateli či skupině uživatelů právo musíme je vybrat. To provedeme tak, že označíme skupinu ze které chceme vybírat a stiskneme tlačítko šipky (obr. 9). Máme možnost opakovat s další skupinou případně vybrat všechny skupiny najednou dvojitou šipkou.



Obr. 9 – dialog úpravy práv vrstvy

Po vybrání skupin/y se zobrazí seznam relevantních uživatelů, se který pracujeme stejným způsobem. Pokud jsme spokojeni s výběrem (obr.10) můžeme přejít na přidělení vyšších práv nebo formulář odeslat k uložení stisknutím tlačítka **Odeslat**.



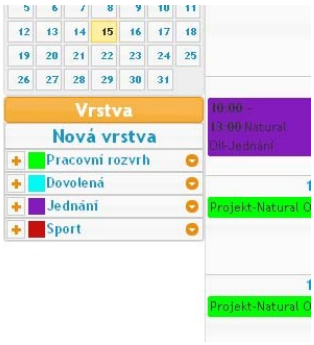
Obr. 10 – dialog úpravy práv vrstvy

Takto vybraným uživatelům se zobrazí tato vrstva, ale bez jakékoliv dalších možností úpravy.

Třetí záložka slouží k přidělení práv spolupráce, což umožňuje vybraným uživatelům přidávání nových úkolů a událostí do této vrstvy.

Čtvrtá záložka slouží k přidělení práv uživatelům, kteří budou v budoucnu oprávněni upravovat vrstvu, a to včetně změny názvu a práv.

Takto vytvořená vrstva se přidá k již existujícím vrstvám v levé části základního rozhraní (obr. 11).



Obr. 11 – existující vrstvy

Jak je patrné z výše uvedeného obrázku, zobrazuje se zde název vrstvy, před kterým je barevné políčko a znaménko +. Za názvem vrstvy je šipka. Nyní si popíšeme co jednotlivé symboly znamenají.

Symbol + nám ukazuje, že se vrstva zobrazuje do rozhraní. Pokud na něj klikneme, změní se na symbol -, 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 vrstva zobrazuje. Pokud na barevné políčko klikneme, otevře se dialog pro změnu barvy. Provedení změny barvy potvrdíme tlačítkem v pravém dolním rohu.

Pokud klikneme na symbol šipky otevře se nám menu. Pokud nemáme právo na správu této vrstvy zobrazí se nám pouze položka Detail, která po stisknutí vypíše informace o vrstvě. V případě plných práv k vrstvě, se nám otevře menu s položkami Detail, Editace, Smazat. Položka Editace nám otevře dialog s editováním vrstvy, který je shodný s dialogem pro tvorbu nové vrstvy. S tím rozdílem, že jsou vyplněni již zadané parametry. Pro uložení změny je potřeba stisknout tlačítko **Odeslat.** Položka Smazat nám

nenávratně odstraní přístup k této vrstvě, včetně všech úkolů a událostí nacházející se v této vrstvě.

3 Události

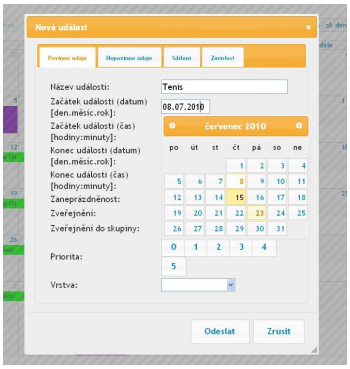
Pod pojmem událost je v této aplikaci myšlena akce, která má definovaný čas začátku a konce. V této kapitole si povíme vše, co nám tato akce nabízí.

- Tvorba události

K vytvoření nové události můžeme využít několik metod. Základní metodou je že klikneme na kterýkoliv den kalendáře, čímž se nám otevře dialog Nová událost s předvyplněným datumem vybraného dne (obr. 12).

Obr. 12 – dialog vytvoření nové vrstvy

Vyplníme název události a dále stanovíme časové parametry vytvářené události. Kliknutím do kolonky s datumem se nám otevře navigační kalendář (obr. 13), ve kterém vybereme námi požadované datum začátku události. Zadáme čas začátku události v požadovaném formátu. Stejně postupujeme u zadání datumu a času ukončení události.



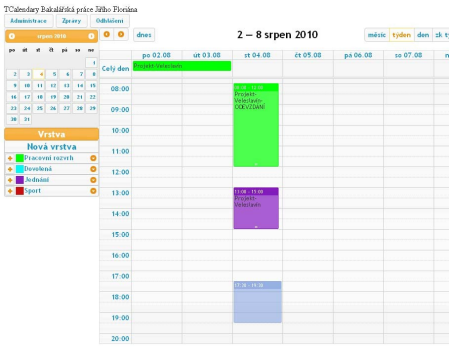
Obr. 13 – vybrání dne pomocí navigačního kalendáře

Komfortnější metodou je vybrání kliknutím na požadovaný den, což vede k předvyplnění správného datumu do dialogu Nová událost. 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 datumu začátku a konce události.

Při poslední metodě využijeme týdenního/denního zobrazení, což vede k lepšímu přehledu o ostatních událostech. Výhoda této metody se projeví tehdy, kdy je 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. Pomocí stisknutí levého tlačítka a tažením myší, vybereme požadovaný čas nové události (obr. 14). Otevře se nám dialog Nová událost s předvyplněným dnem i časem nové události (obr. 15)



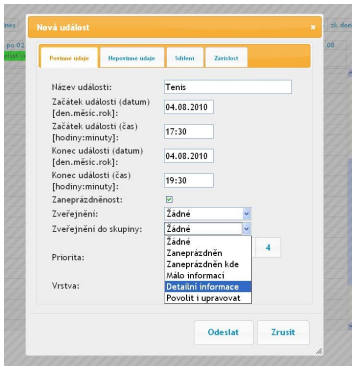
Obr. 14 – vytvoření události pomocí zvoleného časového úseku

Obr. 15 – dialog pro vytvoření nové události pomocí zvoleného časového úseku

Tímto jsme si ukázali, jak je možné zadat časové parametry různými způsoby.

V dialogu Nové události pod časovými údaji nalezneme zaškrtávací políčko zaneprázdněnost, kterým potvrzujeme účast na události.

Dále můžeme nastavit množství informací o události, které budu zveřejňovany pro všechny uživatele a pro zvolenou skupinu uživatelů (obr. 16).



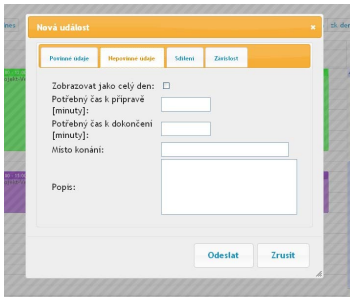
Obr. 16 – nabídka možného zveřejnění

Pozn. : doporučujeme neudělovat nejvyšší stupeň pro veřejné uživatele.

Dalším nastavením je stupeň priority, který určuje pořadí zobrazování více událostí ve stejný okamžik.

Posledním povinným údajem je přiřazení nově vznikající události k vrstvě. To se provede pomocí výběrového menu, kde se zobrazí všechny dostupné vrstvy u kterých máme právo spolupráce(viz. výše kapitola Vrstvy). Po vyplnění první záložky, máme možnost vytvořit událost stisknutím tlačítkem **Odeslat** nebo nastavit další údaje, k čemuž slouží další záložky.

Po kliknutí na druhou záložku se nám zobrazí formulář (obr. 17), kde můžeme vyplnit další informace o události jako je čas k přípravě a dokončení, místo konání, popis a rovněž zde máme možnost zaškrtnou, 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 toho to políčka závisí na typu zvolené metody vytváření nové události, pokud použijeme posledně jmenovanou metodu, políčko je nezaškrtnuté v ostatních případech platí opak.



Obr. 17 – formulář s nepovinnými údaji

Další záložkou je sdílení, zde můžeme vytvářenou událost poskytnout dalším uživatelům, výběr se provádí stejným způsobem, jak byly vybírani uživatelé v kapitole o vrstvách.

Poslední záložkou je závislost, které umožňuje nastavení závislosti vytvářené události na jiné dostupné události. V prvním seznamu vybereme událost, na které má být tato závislá. Vybereme, která činnost s vybranou událostí a jakou změnu vyvolá u námi vytvářené události.

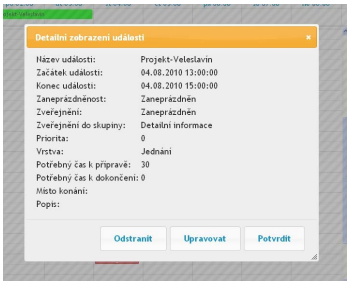
!! Upozornění:

Novou událost lze uložit pouze pokud jsou správně vyplněné údaje. Není-li tomu tak, zobrazí se nám varování s upozorněním na špatně popř. nevyplněný údaj.

- K prvnímu typu varovaní dojde pokud nevyplníme název události,
- Dalším typem varování je špatně zadaný časový údaj, toto varovaní se zobrazí pokud zadaný časový úsek již proběhl nebo jeli datum/čas zadán ve špatném formátu,
- Poslední typ varování nás upozorňuje, že jsme nevybrali prioritu nebo vrstvu
- Zobrazení detailů události

Pokud si chceme prohlédnout údaje o události, klikneme na vybranou událost a aplikace nám otevře dialog Detailní zobrazení události (obr. 18.). Okno zavřeme symbolem X v pravém horním rohu nebo klikneme na

tlačítko **Potvrdit**.



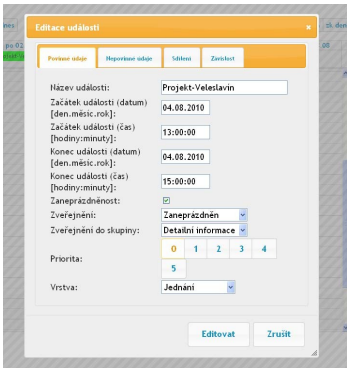
Obr. 18 – dialog se zobrazením údajů o události

- Editace události

Aplikace nám též umožňuje upravovat již vytvořené události, a to tak, že využijeme tlačítka v dialogu zobrazení události (obr. 18).

Pokud chceme událost vymazat stiskneme tlačítko **Odstranit**, což nám vyvolá dotaz, zdali opravdu chceme vybranou událost smazat. **P**o stisknutí **Ano**, je událost nenávratně odstraněna.

Pro upravování události stiskneme tlačítko **Upravovat**. Po stisknutí se nám otevře dialog editace události (obr. 19), který má stejnou strukturu jako vytváření nové události a platí pro něj stejná pravidla. Pro potvrzení změn stiskneme tlačítko **Editovat**, tlačítko **Zrušit** slouží k opuštění dialogu beze změn.



Obr. 19 – dialog se zobrazením údajů o události

4 Úkoly

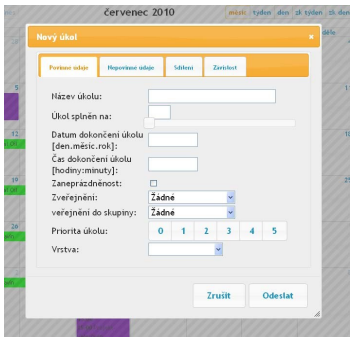
V této kapitole si popíšeme práci s úkoly, což je akce, která má pouze termín ukončení. Veškerá správa úkolů se nachází v sekci napravo základního rozhraní. Pod názvem sekce se nachází dvě tlačítka **Možnosti** a **Zobrazení** a níže se nám pak zobrazují již existující úkoly.

4.1.1 Možnosti

Po stisknutí tlačítka **Možnosti**, se nám otevře menu se třemi položkami Nový úkol, Znova načíst, Rozbalit všechny detaily (obr. 20).

Obr. 20 – sekce úkol po stisknutí tlačítka Možnosti

Stisknutím položky Nový úkol se nám otevře dialog Nový úkol (obr. 21), kde vyplníme požadované údaje. Pokud jsou časové údaje vyplněny ve špatném formátu, či některá z položek na tomto formuláři není vyplněna, zobrazí se nám chybové hlášení.

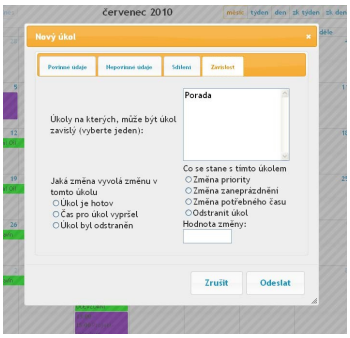


Obr. 21 – dialog pro zadání nového úkolu

Ve druhé záložce máme možnost vyplnit další popisné údaje.

Třetí záložka slouží k na sdílení vytvářeného úkolu ostatním uživatelům. Pracuje se obdobně jako s jinými záložkami sdílení v této aplikace, které jsou popsány výšše.

Poslední záložka slouží k nastavení závislosti na jiném úkolu (obr. 22). Po vybrání úkolu, na kterém má být nově vytvářený úkol závislý, se vybere jaká změna úkolu nám vyvolá požadovanou změnu na vytvářeném úkolu.



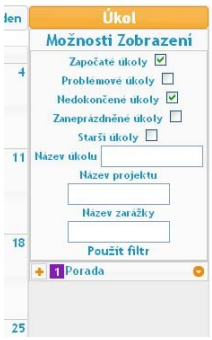
Obr. 22 – dialog pro nastavení závislosti

Stisknutím Znovu načíst se obnoví sekce úkoly s novými nastaveními.

Položka Rozbalit všechny detaily nám slouží k rychlému zpřístupnění všech detailů u právě zobrazených existujících úkolu.

4.1.2 Zobrazení

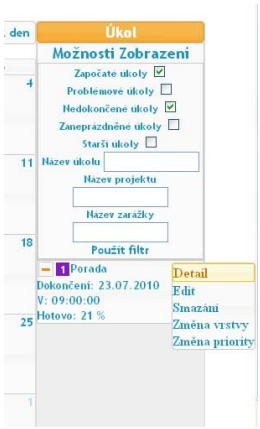
Stisknutí tlačítka **Zobrazení** otevře menu, kde se na staví jaké úkoly se mají zobrazovat (obr. 23). Využívá se k tomu zaškrtávacích políček, popřípadě můžeme využít přesného zadání názvu úkolu či projektu. Nové nastavení se aktivuje stisknutím tlačítka **Použít filtr**.



Obr. 23 – sekce úkol po stisknutí tlačítka Zobrazení

4.1.3 Existující úkoly

Níže se nám zobrazují existující úkoly. Na řádku se nachází symbol +, který po stisknutí rozbalí informace o úkolu. Vedle symbolu je situováno barevné políčko s číslicí. Číslice udává prioritu a barva je shodná s barvou vrstvy, ke které je úkol přiřazen. Napravo na řádku je šipka, která po stlačení vyvolá menu (obr. 24).



Obr. 24 – zobrazení existující úlohy s rozbalenými detaily a vyvolaným menu

Příloha B – Obsah přiloženého CD

- \TCalendar – adresář obsahující zdrojové kódy implementace,
- \Instalace – adresář soubory k instalaci, především pak soubory SQL k vytvoření databáze. Adresář též obsahuje textový soubor s podrobnými pokyny k instalaci,
- \doc – adresář obsahující generovanou dokumentaci k PHP třídám,
- \Text.pdf – elektronická verze bakalářské práce,
- \Uzivatelska_dokumentace.pdf – elektronická verze uživatelské dokumentace,
- \index.html – úvod pro obsah CD.

Příloha C – návrh databáze