VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

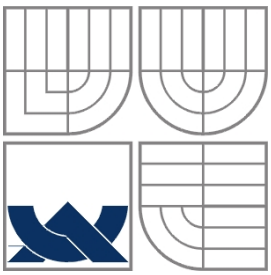
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE OSOBNÍHO PLÁNOVAČE

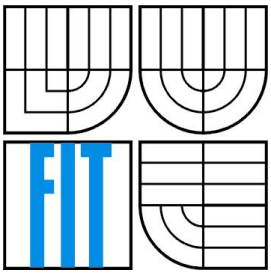
BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE JAKUB FILÁK
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

IMPLEMENTACE OSOBNÍHO PLÁNOVAČE

IMPLEMENTATION OF PERSONAL SCHEDULER

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

JAKUB FILÁK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. ROMAN LUKÁŠ, PH.D.

BRNO 2007

Implementace osobního plánovače

Zadání

1. Prostudujte detailně různé funkce osobních plánovačů pro plánování pracovních záležitostí a jiných soukromých úkolů.
2. Navrhněte rozhraní vlastního osobního plánovače a strukturu databáze, která bude tyto různé úkoly evidovat.
3. Daný osobní plánovač naimplementuje pomocí programovacího jazyka C#.
4. Porovnejte Váš osobní plánovač s ostatními již existujícími osobními plánovači na trhu a diskutujte jeho přednosti a nedostatky. Diskutujte další možnosti rozšíření software o užitečné funkce.

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Cílem této bakalářské práce je prozkoumat možnosti plánovaní času a vytvořit nástroj, který toto plánovaní jednodušší a urychlí. Vytvořený nástroj bude určen nejen k plánování osobních událostí a úkolů, ale také k plánování schůzek, které bude usnadněno sdílením uživatelských diářů a vyhledáváním vhodných termínů s jinými uživateli. V textu dokumentu jsou rozebrány použité technologie, požadavky na nástroj pro plánování času a postup implementace nástroje. Na závěr jsou zhodnoceny výsledky práce a schopnosti vytvořeného nástroje.

Klíčová slova

Plánovač, informační systém, databáze, .NET, upomínky

Abstract

The goal of this bachelor´s thesis is to explore the possibilities of the time scheduling and to create a tool for its simplification. Besides managing personal tasks and events, the tool can also be used for planning appointments with other users, mainly due to a user diaries sharing and a searching tool for appropriate dates. The first part of the thesis deals with the technologies employed, followed with a part concerning the requirements for the time scheduling tool and a description of an implementation of the tool. In the last part the programme capabilities are evaluated, along with the results of the work.

Keywords

Scheduler, information system, database, .NET, reminders

Citace

Jakub Filák: Implementace osobního plánovače, bakalářská práce, Brno, FIT VUT v Brně, 2007

Implementace osobního plánovače

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Romana Lukáše
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Filák
15.5. 2007

Poděkování

Děkuji vedoucímu Romanu Lukášovi za pomoc při vytváření této práce.

© Jakub Filák, 2007.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

1 Úvod

V dnešní době, ač je to k nevíře, je nejvíce nedostatkovým zbožím čas. I když existuje spousta vymožeností, které nám čas šetří, stále neexistuje žádná, která čas "vyrábí". A tak je čas stále jen spotřebováván a proto je nutné jej co nejefektivněji využívat. Rozhodl jsem se proto prozkoumat stávající možnosti plánování času a na základě svého zkoumání stanovit požadavky na nástroj k plánování času. Tento nástroj poté implementuji a ověřím si zda s jeho pomocí dokáži efektivně svůj čas naplánovat.

Nástrojů, které nám umožňují plánovat čas, jsou dnes spousty. A proto by měl být výsledek této práce něčím výjimečný, výjimečný svou schopností ulehčit plánování. Výjimečnost softwaru netrvá dlouho, protože pokud některý umí něco víc než ostatní, budou to brzy umět i ty ostatní a budou to umět lépe. A tak se mohlo stát, že během vývoje mého nástroje již funkce, které označuji za výjimečné, jsou dnes zcela běžné.

Můj nástroj bude v podstatě informační systém. Uživatel toho systému nebude potřebovat přístup k internetu, neboť se bude jednat o osobní informační systém, který bude provozován na uživatelském PC. Systém však bude mít schopnost komunikace po síti s centrální databází. Pokud bude uživatel chtít, bude si moci po registraci do centrální databáze ukládat svá data. Existence centrální databáze přinese uživatelům několik výhod.

Jednou z hlavních výhod je sdílení naplánovaných aktivit mezi uživateli, díky čemuž bude možné sledovat co jiní uživatelé dělají a kdy mají čas. Systém tohoto sdílení aktivit bude umět využívat při tvorbě schůzek, a to tak že bude umět najít vhodný termín pro schůzku více uživatelů.

Další výhodou je také dostupnost naplánovaných aktivit všude tam kde bude mít uživatel přístup k internetu. A také bude díky tomu možná synchronizace mezi různými PC, na nichž bude uživatel tento nástroj provozovat.

Existuje mnoho technologií použitelných k vytvoření nástroje pro plánování času. Já jsem si zvolil prostředí operačního systému Microsoft Windows a platformu .NET s programovacím jazykem C#. Volil jsem z několika důvodů. Jedním z nich je, že pro operační systémy UNIX existuje opravdu kvalitní plánovač času KOrganizer. I pro Microsoft Windows existují plánovače času, ale většinou nejsou příliš kvalitní a nebo se za ně musí platit. Vypadá to, že se ve Windows nedá čas pořádně plánovat, ale naštěstí tomu tak není. Existují i použitelné plánovače, nejsou však příliš známy nebo jsou ještě ve vývoji.

2 Informační systém

Informační systém je celá řada zdrojů a prostředků sloužících ke sběru, uchování, zpracování a poskytování informací. Toto vše je prováděno za účelem zlepšení výkonnosti uživatelů informačního systému.

2.1 Informace

Informace je jakýkoliv údaj, který má nějaký význam. Ačkoliv se dnes hovoří o informačním věku, není termín informace jednoznačně definován. Je však jisté, že informace jsou data, která mají sémantiku. Sémantiku dat je velice obtížné definovat, protože se jedná o jejich význam, který se v kontextu uživatele může a taky často liší.

2.1.1 Data

Jsou vyjádření skutečnosti a popisují nějaký jev. Data lze získat měřením nebo pozorováním. Jsou schopná přenosu, uchování či interpretace. Z hlediska počítačů jsou to hodnoty datových typů. Data nemusejí a taky většinou nemají žádnou sémantiku. Data samotná, bez dalšího zpracování, nemají pro uživatele velký význam.

2.1.2 Znalost

Znalosti jsou informace poskládané do určitých souvislostí. Znalosti nám umožňují porozumět skutečnosti.

2.2 Obecný systém

Systém je účelově definovaný soubor komponent, mezi kterými existují určité vztahy, a které splňují nějaký cíl. Obecný systém se skládá ze vstupní a výstupní části, kudy do systému vstupují a vystupují zdroje. Mezi vstupem a výstupem dochází k transformaci těchto zdrojů.

3 Databáze

Databáze je určitá uspořádaná množina informací (perzistentních dat) uložená na paměťovém médiu. Paměťové médium, na kterém je databáze uložena nemusí být nutně harddisk počítače, ale mohou to být papírové kartotéky nebo děrné štítky. Databáze slouží k evidování a shromažďování dat. Základními prvky databáze jsou data a program pro práci s nimi. Najdeme je dnes téměř v každém odvětví lidské činnosti.

3.1 Historie

Předchůdcem dnešních databází byly papírové kartotéky. Umožňovaly uspořádání dat podle různých kritérií a zatřiďování nových položek. Veškeré operace s nimi prováděl přímo člověk. Správa takových kartoték byla v mnohém podobná správě dnešních databází. Dalším krokem při vývoji databázi bylo převedení úkolů člověka na stroje.

3.1.1 50. léta

V této době bylo vše v programu. V programu byla jak data tak i jejich definice. Díky tomu, že jsou data součástí jednoho procesu, nemohla být sdílena s jinými procesy.

3.1.2 60. léta

Od poloviny 60. let bylo možné zpracování dat více procesy najednou. Data jíž více nebyla součástí procesů, ale jejich definice stále součástí procesů zůstávala. Zpracování dat více procesy bylo možné díky existenci systému pro zpracování souborů. Objevil se však problém s možnou nekonzistencí při přístupu více procesů k jednomu souboru. Problém spočívá v tom, že proces mohl začít provádět nějakou akci na základě stavu jistých dat přičemž jiný proces mohl tyto data v průběhu provádění akce změnit.

3.1.3 70. léta

V této době vznikl systém řízení báze dat (SŘBD), který umožnil osvobození procesů od definice dat. Data jsou nyní uložena v databázi a mohou být sdílena více procesy bez nebezpečí nekonzistence dat díky transakcím.

Systém řízení báze dat je soustava programů fungujících jako rozhraní mezi uživatelem a databází. Slouží k definici a konstrukci databáze a k manipulaci s databází. SŘBD umožnil vznik transakcí. Transakce je tvořena posloupností databázových operací a její hlavní vlastností je atomičnost.

V této době se také objevily relační databáze. Název relační nemají díky vztahům mezi tabulkami, nýbrž proto že každá tabulka je relací. Tabulky obsahují sloupce a řádky. Sloupce odpovídají jednotlivým vlastnostem (atributům). Každý sloupec má jednoznačně stanoven název, typ a rozsah neboli doménu. Řádky jsou n-tice, které obsahují informaci. Typy hodnoty v řádcích odpovídají typu sloupce, ve kterém se nacházejí.

3.1.4 80. léta

Vývoj se zaměřuje na výzkum distribuovaných databázových systémů, tento vývoj však neměl příliš velký dopad na obor databází.

3.1.5 90. léta

V této době se začínají objevovat objektově orientované databáze. Mají úspěch ve specializovaných databázích, kde se pracuje s více komplexními daty, ale relační model je stále nejhojnější. Některé ideje objektově orientovaných databází byly použity v relačních databázích a tak vznikly objektově-relační databáze.

3.1.6 Současnost

V současnosti se stávají moderními XML databáze, které odstraňují tradiční rozdíl mezi dokumenty a daty. Stále jsou však nejpoužívanější relační databázové systémy.

3.2 SQL

SQL (Structured Query Language) je standardizovaný dotazovací jazyk používaný pro práci s daty v relačních databázích. První standard byl přijat v roce 1986, ale obsahoval nedostatky a tak byla v roce 1992 vydána opravená verze s názvem SQL92, která se používá dodnes. SQL je deklarativní programovací jazyk, což znamená, že musíme používat interpret jazyka. Jazyk SQL se skládá z několika skupin.

Skupina DDL (Data Definition Language) je určena pro specifikaci schématu databáze. Schéma je možné vytvářet, doplňovat a mazat. Skupina DML (Data Manipulation Language) je určena pro manipulaci s daty. Jsou to příkazy pro získávání dat z databáze a pro jejich úpravy. Obsahuje ještě další skupiny jako jsou DCL (Data Control Language), která slouží pro řízení přístupových práv nebo SDL (Storage Definition Language), která slouží pro definici způsobu ukládání tabulek.

3.3 MySQL

Je to multiplatformní relační databázový systém. Je také volně šiřitelná a díky své rychlosti a výkonnosti je hojně používaná. Jelikož se databázový systém MySQL snažil být rychlý a výkonný, postrádal v dřívějších verzích spousty standardů relačních databázových systémů. Během vývoje byly tyto standardy doplňovány, ale ani v nejnovějších verzích nejsou všechny obsaženy.

3.4 SQLite

Je relativně malý relační databázový systém obsažený v jedné knihovně napsané v jazyce C. Její použití je na rozdíl od databázových systémů typu klient-server, kdy server běží jako samostatný proces, mnohem lehčí. Stačí ji pouze přilinkovat k aplikaci a pomocí jednoduchého rozhraní začít používat. Nic víc k použití databázového systému SQLite není potřeba. SQLite databáze jsou založeny na souborech. Každý soubor je jedna databáze. Má schopnost pracovat s databází o velikosti až 2 terrabyty. Po vytvoření databáze není potřeba dalšího administrátorského nastavování, vše se nastaví samo.

4 Microsoft .NET Framework

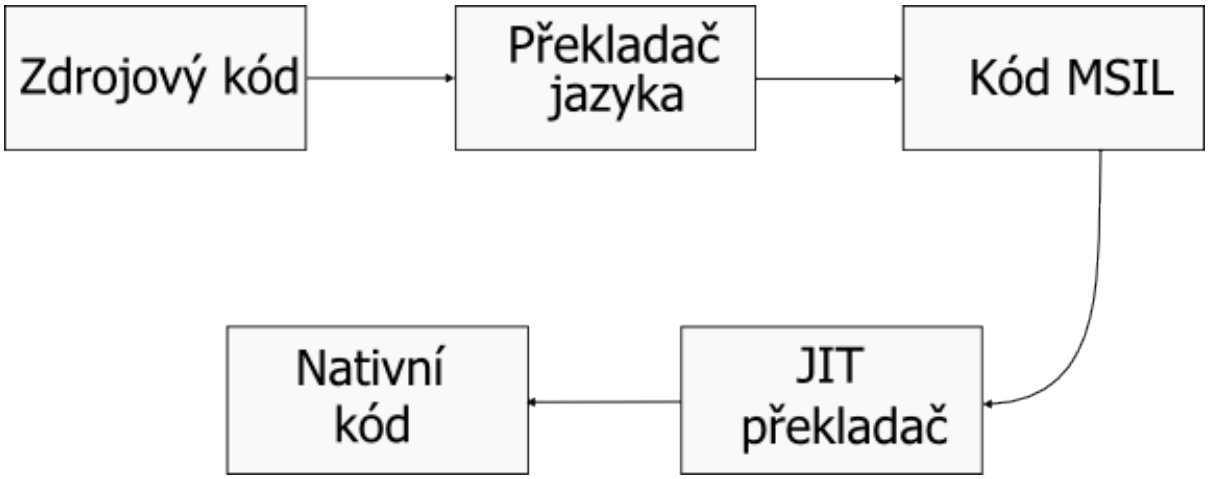
Prostředí .NET Framework je soubor technologií vyvinutých firmou Microsoft. Obsahuje knihovnu, která je stejně rozsáhlá jako rozhraní Windows API. Je možné ji používat k volání stejných funkcí, které dříve poskytoval samotný operační systém. Kromě toho je také možné ji používat i v novějších oblastech jako je přístup k databázím, připojení k internetu nebo přístup k webovým službám.

Prostředí .NET Framework nabízí rovněž operační prostředí (.NET runtime), v němž jsou programy spouštěny. Při spouštění kódu založeného na platformě .NET Framework to bude právě operační prostředí .NET, které náš kód spustí. Operační prostředí zajistí správu spouštěných vláken a podpůrných služeb. Prostředí .NET Framework nám z jistého pohledu zobecňuje operační systém.

4.1 CLR

Nejdůležitější částí prostředí .NET Framework je jeho operační prostředí označované jako modul **CLR** (Common Languagle Runtime). Předtím než kód v prostředí .NET spustíme, je třeba kód přeložit. Překlad probíhá ve dvou krocích. Nejprve je kód přeložen do jazyka MSIL a následně je přeložen modulem CLR na kód specifický pro danou platformu. Kód spouštěný v prostředí .NET je často označován jako spravovaný kód.

První krok v překladu se provádí po vytvoření aplikace. Tehdy se přeloží do jazyka MSIL, ve kterém je dále distribuována. Druhý krok překladu se provádí při spuštění aplikace a při každém volání metody. Tento postup překladu se může zdát zdlouhavý, ale uvidíme, že to přináší výhody, které zdlouhavost překladu kódů vynahradí.



Obrázek 1: Schéma překladu v .NET

4.2 MSIL

Výsledkem kompilátoru jazyka schopného generovat řízený kód je MSIL - MicroSoft Intermediate Language. MSIL je procesorově nezávislý jazyk podobný assembleru. Oproti asembleru je však mnohem vyspělejší. Umí pracovat s objekty, volat virtuální metody, pracovat s prvky pole nebo zpracovávat výjimky. Důvodem pro zavedení tohoto jazyka je snaha o jednoduché přenášení existujícího kódu mezi různými platformami. V současné době neexistuje procesor, na kterém by šlo provádět instrukce MSIL, proto se používá modul CLR. Hlavní výhodou použití intermediárního jazyka je platformní nezávislost.

4.3 JIT

Tento pojem znamená překlad v době potřeby (Just-In-Time). Kód v Jazyce MSIL je překládán až tehdy jeli potřeba. To znamená, že se celý program nepřekládá při startu, ale jsou překládány jednotlivé metody při svém volání. Vývojáři firmy Microsoft tvrdí, že velká část aplikace se v době běhu nevyužije a tak se tato strategie vyplatí a zrychlí start aplikace. Při překladu v době potřeby je také možno provádět spoustu optimalizací, neboť překladač ví na jakém procesoru běží a tak jej může efektivně využívat. Výsledek potom může vypadat tak, že kód spuštěný v .NET Framework je rychlejší než kód překládaný přímo na cílovou platformu. Překladače totiž dopředu neví na jakém procesoru bude aplikace spouštěna a nemohou kód tak dobře optimalizovat.

4.4 CTS

Common Type System je typový systém prostředí .NET, který je nezávislý na použitém jazyku. Tento typový sytém je objektový, unifikovaný a komponentně orientovaný. Je hierarchicky uspořádaný a podporuje dědičnost napříč jazyky. Každý jazyk používaný v prostředí .NET musí mít typový systém, který je podmnožinou CTS. Díky tomu, že jsou typové systémy různých jazyků podmnožiny CTS a že je CTS nezávislý na zvoleném jazyce, je možná interoperabilita jazyků.

4.5 Výhody .NET Framework

Hlavními výhodami jsou nezávislost na platformě a interoperabilita jazyků. Díky prvnímu překladu do jazyka MSIL a poté překladu modulem CLR je možné vyvíjet multiplatformní aplikace, protože kód je spouštěn v .NET a toto prostředí může fungovat na různých platformách a stejně tak potom můžou námi vytvořené aplikace pracovat na těchto platformách. Kód ve vyšším programovacím jazyce může být vytvořen na jedné platformě a poté přeložen do jazyka MSIL může být spuštěn na jiné.

Nezávislost prostředí .NET Framework je zatím značně omezené, neboť je toto prostředí v plné funkčnosti dostupné pouze pro operační systémy Microsoft Windows. Na přenesení .NET Framework na jiné operační systémy se již pracuje v podobě projektu MONO.

Výhoda interoperability jazyků spočívá v tom, že prostředí .NET lze používat mnoha programovacími jazyky.

Prostředí též zajišťuje automatickou správu životnosti pomocí Garbage Collection. Vývojáři se tak nemusí zabývat uvolňováním paměti. Nepoužívané objekty jsou po čase z paměti odstraněny prostředím.

4.6 Jazyk C#

Je nový objektově orientovaný jazyk šitý na míru prostředí .NET Framework. Jazyk C# je odvozený od jazyka C++. Jazyk C# je čistě objektově orientovaný, což znamená, že všechno jsou třídy, které dědí z jedné hlavní třídy. Dědičnost v tomto jazyce je pouze jednoduchá. Jazyk je case senstitive. Rozlišuje tedy velká a malá písmena.

Přestože je vytvořen speciálně pro prostředí .NET Framework není jeho součástí. V prostředí .NET existují funkce, které v jazyce C# nejsou dostupné. Stejně tak i jazyk C# obsahuje funkce, které v prostředí .NET dostupné nejsou.

4.6.1 Jmenné prostory

Programy v jazyce C# jsou rozděleny zpravidla do několika jmenných prostorů zvaných namespace. Každý jmenný prostor může obsahovat další jmenné prostory, definice nových typů a metadata. Jmené prostory představují způsob jak logicky strukturovat program. Každý typ v jazyce patří do jediného jmenného prostoru a přistupuje se k němu pomocí tečkové notace.

4.6.2 Typ

Je šablona pro monžinu objektů. Objekt je vlastně instance typu. Z jednoho typu můžeme vytvořit několik instancí. Typ je tedy způsob popisu množiny objektů. Typy v jazyce C# se dělí do dvou základních skupin. Hodnotové a referenční typy.

Proměnná hodnotového typu představuje hodnotu. Tato hodnota je uložena někde v paměti, takže proměnná hodnotového typu je vlastně pojmenovaná část paměti. Paměť pro hodnotové typy je alokována na zásobníku.

Proměnná referenčního typu pojmenovává místo na zásobníku, kde je silně typovaný ukazatel na hodnotu v paměti uloženou na haldě. Tento silně typovaný ukazatel není nic jiného než číslo označující místo na haldě. Halda je paměť přidělená programu po jeho spuštění. Umísťují se do ní proměnné vytvořené za běhu programu.

4.6.3 Třídy

Jsou to základní referenční složené datové typy. Složené typy to jsou proto, že jsou složeny z dalších složek a metod. Složky představují proměnné uvnitř třídy a metody jsou funkce třídy.

Třídy mají u všech svých složek a metod modifikátory viditelnosti. Modifikátory viditelnosti slouží k řízení viditelnost složek vně třídy. Modifikátor *private* umožňuje přístup k dané složce pouze v rámci třídy. Modifikátor *protected* umožňuje přístup ke složkám v rámci třídy a tříd od ní odvozených. Modifikátor *public* umožňuje přístup odkudkoliv. Nakonec modifikátor *internal* znamená všem v rámci jedné *assembly*.

Dalšími možnými modifikátory jsou modifikátory *static,* kterými deklarujeme statický člen dostupný pouze v deklaraci třídy a ne v objektech této třídy, modifikátor *const,* kterým deklarujeme konstatní statický člen a modifikátor *readonly,* kterým se definuje člen pouze pro čtení. Členu *readonly* lze přiřadit hodnotu v konstruktoru třídy.

Definice třídy začíná slovem *class* a dále následuje dvojice složených závorek. Mezi těmito závorkami je tělo třídy. V jazyce C# je možné vytvářet třídy, které jsou rozděleny do několika souborů. Toho dosáhneme tím, že před klíčové slovo *class* napíšeme další klíčové slovo *partial*. V deklaraci třídy je možné ještě použít klíčové slovo *abstract,* pro vytvoření abstraktní třídy nebo *sealed*, pro zakázaní dědění z této třídy.

V jazyce C# je možné deklarovat u tříd vlastnosti. Vlastnost není nic jiného než dvojice metod, které se navenek chovají jako jedna proměnná. U každé vlastnosti můžeme deklarovat metodu *get* pro návrat hodnoty a metodu *set* pro nastavení hodnoty. V těle metody *set* reprezentuje předávanou hodnotu klíčové slovo *value.* Vlastnost musí obsahovat alespoň jednu z těchto metod. Vynecháním jedné metody získáme *read only* respektive *write only* vlastnost.

4.6.4 Struktury

Struktury jsou uživatelsky definované hodnotové typy. Jsou podobné třídám, ale podstatně se od nich liší. Proměnné struktury jsou uloženy na zásobníku. Struktury také nepodporují dědičnost. Jsou logickým sdružením atributů, metod, indexerů, operátorů, vlastností a případně dalších vnořených typů.

4.6.5 Výčtové typy

Výčtový typ je množina pojmenovaných konstant. Typickým příkladem je množina dnů. Jednotlivé elementy jsou implicitně číslovány od nuly. V případě, že je to potřeba, můžeme omezit rozsah tohoto typu. Standardně je typ konstant uvnitř výčtového typu *int*.

4.6.6 Delegáti

V jazyce C existuje ukazatel na metodu. Obdobný mechanismus poskytuje i jazyk C#. Jde o delegáty. Oproti ukazatelům na funkce jsou však delegáti typově bezpeční. Delegáta deklarujeme pomocí klíčového slova *delegate*.

4.6.7 Rozhraní

Rozhraní je z praktického hlediska čistě abstraktní třída obsahující pouze veřejné složky. Jazyk C# umožňuje pouze jednoduchou dědičnost, ale díky rozhraním jsme schopni třídám přidávat vlastnosti nezávisle na hierarchii tříd. Třídy z rozhraní nedědí, ale třídy implementují rozhraní. Mohou implementovat i více než jedno rozhraní.

Definice rozhraní začíná slovem *interface* a jeho tělo se píše stejně jako u tříd mezi dvě složené závorky. Uvnitř rozhraní jsou uvedeny pouze jména metod s jejich parametry a návratovými typy. K metodám se nepíší modifikátory viditelnosti, neboť jsou všechny veřejné.

4.6.8 Dědičnost

Jak již bylo několikrát zmíněno v C# je možná pouze jednoduchá dědičnost. Dědičnost v definici třídy vyjádříme dvojtečkovou konvencí.

Chceme-li předefinovat veřejnou metodu, kterou třída dědí, použijeme klíčové slovo *new* při deklaraci nové metody. Za běhu programu poté záleží na typu reference, jaká metoda se zavolá.

Chceme-li realizovat polymorfismus, použijeme virtuální metody. Virtuální metodu deklarujeme za pomocí klíčového slova *virtual*. V odvozené třídě požijeme klíčové slovo *ovveride* pokud chce tuto virtuální metodu přepsat. Za běhu programu nezáleží na typu reference, ale je vždy volána metoda daného typu.

4.6.9 Reflexe

Mnohé ze služeb platformy .NET (jako pozdní vazba, serializace, vzdálené řízení, atributy a podobně) závisejí na přítomnosti metadat. Vytvářené programy mohou využívat tato metadata a rozšiřovat je o nové informace. *Reflexí* je označováno prozkoumávání existujících typů prostřednictvím metadat. Uskutečňuje se prostřednictvím sady typů v jmenném prostoru `System.Reflection`. Reflexe představuje procházení a manipulování s objektovým modelem, který představuje nějakou aplikaci včetně všech jejích elementů kompilace a běhu.

5 Počítačová síť

Počítačová síť je souhrnné označení prostředků, které umožňují spojení a výměnu informací mezi počítači. Počítačové sítě byli vyvinuty pro možnost sdílení prostředků mezi počítači. Dnes se využívají i k dalším účelům, jako je hraní her nebo komunikace uživatelů.

5.1 Internet

Internet je celosvětová počítačová supersíť. Jedná se o propojení mnoha menších sítí pomocí sady protokolů TCP/IP. Vznikl z experimentální sítě ARPANET, která byla vytvořena v roce 1969. Stejně jako počítačové sítě slouží ke sdílení, komunikaci a mnoha dalším účelům. V síti internet mají všechny počítače jednoznačnou IP adresu. IP adresa je 32 bitové číslo pro IP adresy verze 4. IP adresy verze 6 mají délku 128 bitů. Základními protokoly jsou SMPT, FTP, HTTP, SNMP, DNS.

5.2 HTTP

HTTP (Hyper Text Transfer Protokol) je počítačový protokol, určený původně k přenášení hypertextových dokumentů ve formátu HTML. Hypertextový dokument je strukturovaný elektronický text, obsahující odkazy na jiné dokumenty, obrázky, videa, animace a na další zdroje.

Protokol HTTP je založen na zasílání zpráv, které slouží jako dotazy a také jako odpovědi. Protokol pracuje způsobem dotaz-odpověď. Tyto zprávy mají přesně definovaou strukturu, kdy se každá správa skládá z hlavičky, za kterou může následovat tělo. Komunikace poté probíhá tak, že klient (většinou prohlížeč) zašle požadavek na server. V tomto požadavku specifikuje dokument, který požaduje, informace o svých schopnostech apod. Server na tento požadavek odpoví zprávou.

5.2.1 Metoda GET

Je to nejpoužívanější metoda v protokolu HTTP. V metodě je uveden požadavek na objekt. Server na tento požadavek odpoví odesláním toho objektu, pokud je dostupný.

5.2.2 Metoda POST

Metoda post je další hojně používanou metodou protokolu HTTP. Tato metoda je určena k odesílání uživatelských dat na server. Metoda se používá například při odesílání formulářů.

5.3 PHP

PHP je rozšířený univerzální skriptovací jazyk s volně dostupným zdrojovým kódem, který je obzvláště vhodný pro vývoj webových aplikací a lze jej zapouzdřit do HTML. PHP - původně Personal Home Page vzniklo v roce 1996. Od té doby prošlo velkými změnami a nyní tato zkratka znamená Hypertext Preprocessor.

Při využití PHP pro tvorbu stránek jsou skripty vykonávány na straně serveru a uživateli se tak posílá jen výsledek jejich práce. Díky tomu, že je PHP interpretovaný jazyk, je nezávislé na operačním systému. Programátorům poskytuje rozsáhlé knihovny funkcí téměř pro každou operaci potřebnou při programování. Syntaxe jazyka kombinuje několik programovacích jazyků (Perl, C, Pascal a Java).

6 Návrh

V této části dokumentu se budu zabývat požadavky na nástroj pro plánování času. Rozeberu zde různé funkce a vlastnosti, které by měl nástroj pro plánování času mít. V části specifikace cílů budou uvedny a rozebrány konkrétní funkce mého nástroje, které budou implementovány.

6.1 Analýza požadavků

Moderní nástroje pro plánování času musí mít spoustu funkcí, na které jsou uživatelé již zvyklí. Pro plánování nejsou důležité pouze funkce určené k plánování času, ale také funkce, které toto plánování usnadňují.

Dnes je zcela běžné, že mají lidé více počítačů a dalších mobilních zařízení, které obsahují nástroje pro plánování času. Tito lidé často řeší problém jak mít své plány stále u sebe. A tak jsou kromě funkcí pro plánování nutné funkce pro synchronizaci mezi počítači nebo přenesení informací z jednoho stroje do jiného a většinou také do jiného nástroje pro plánování.

6.1.1 Upomínky

Lidé často zapomínají na své naplánované události a proto jsou zde plánovače, které plány nezapomínají. Ale pamatovat si naplánované události nestačí, jelikož uživatel často nemá čas sledovat hodiny a čekat na událost. Je nutné, aby mu takovou událost někdo připomenul. Podle mého soudu patří funkce připomínání k těm nejdůležitějším a nejužitečnějším funkcím všech plánovačů. Pokud některý plánovací nástroj tuto funkci postrádá, je to pro něj velikým handicapem.

6.1.2 Synchronizace

Díky tomu, že uživatelé používají více počítačů a provozují na nich své plánovače, je nutnou funkcí synchronizace mezi těmito počítači a jejich plánovači. Tato synchronizace by měla probíhat nejlépe transparentně, a to tak, že uživatelé při změně počítače a po spuštění svého plánovače budou mít dostupné všechny své naplánované věci, aniž by se o to museli jakkoliv starat.

6.1.3 Import/Export

Jelikož již existují jiné plánovací nástroje, je nutné umožnit uživatelům migraci mezi těmito nástroji. Nejlepším způsobem jak toto uživatelům umožnit, je schopnost plánovače importovat a exportovat různé formáty dat jiných plánovačů.

Dnes existuje několik formátů uložení dat plánovačů. Tím podle mě dnes nejpoužívanějším a nejznámějším je formát softwaru Microsoft Outlook. V tomto formátu jsou data uložena v jedno textovém souboru. Uvnitř toho souboru jsou jednotlivá data oddělena čárkou. Formát souboru je

nazýván CSV. CSV je standardní formát, který využívá spousta aplikací k ukládání dat. CSV znamená hodnoty oddělené čárkou.

Jelikož je formát Outlooku tak hojně používán, je nejvhodnějším kandidátem pro implementaci. Při schopnosti plánovače importovat a exportovat formát CSV bude také zaručena možnost uživatelů migrovat z jiných plánovačů na můj. Také bude zaručena možnost přenosu informací z PC do mobilních telefonů a PDA. To díky tomu, že software v těchto zařízeních většinou formátu CSV rozumí.

Existují také další používané formáty. Jedním ze známějších formátů je například standard *iCalendar*. Tyto další formáty nejsou tak rozšířeny a proto není nutné implementovat schopnost tyto formáty exportovat nebo importovat. To však neznamená, že nepřijde doba, kdy budou rozšířeny natolik, že schopnost exportovat a importovat tyto formáty nebude nepodstatná. Již dnes je vhodné se na takovouto situaci připravit.

6.1.4 Přehlednost

Přehlednost je jedna z nejdůležitějších vlastností nástroje pro plánování času. Nejde jen o přehlednost a intuitivnost ovládání, ale také o přehlednost naplánovaných akcí. Plánovač musí být schopen poskytovat různé pohledy na uživatelské diáře tak, aby měli uživatelé jasný a úplný přehled o svém čase. Tím mám namysli pohled z hlediska času, tedy časového úseku, který pohled reprezentuje. Jsou to například pohled denní, týdenní, měsíční a další užitečné časové úseky.

6.2 Specifikace cílů

Podle požadavků na plánovací nástroj je možné takový nástroj vytvořit, avšak takovýto nástroj by nepřinesl nic nového. Proto můj nástroj bude obsahovat nové funkce a vlastnosti, které jiný nástroj pro plánování času neobsahuje. V této části budou uvedeny funkce a vlastnosti, které jsou dle mého názoru nové nebo se s nimi v plánovačích často nesetkáváme. Tyto funkce a vlastnosti bude můj plánovač obsahovat.

6.2.1 Základní funkce

Z analýzy jsou znát základní funkce, které žádnému plánovači času nesmí chybět. Jsou to funkce pro synchronizaci, export/import různých formátů, připomínání, a přehlednost. Bez nich by byl program pro plánování v dnešní době téměř nepoužitelný. Proto tyto funkce bude obsahovat i můj plánovač. Toto však nejsou úplně všechny funkce, které považuji za základní a které bude můj plánovač obsahovat.

6.2.2 Události

Hlavním úkolem plánovačů je vytvářet události a tyto vytvořené události poté uživateli zobrazovat. Stejně tak jako všechny i můj plánovač bude vytvářet události. Každá vytvořená událost bude zařazena do kategorie. Zařazování událostí je standardní funkcí a je velice užitečná. Další vlatností události, která ji zařazuje do určitého druhu kategorie, je kalendář, do kterého náleží. Tato vlastnost je využívána při tvorbě přehledů, kdy si můžeme zvolit, které kalendáře chceme vidět. Neposledními vlastnostmi jsou priorita a viditelnost události. Viditelnost je zde pro řízení viditelnosti detailů naplánované události jinými uživateli.

Událostem bude také možné přiřadit opakování. Opakování budou denní nebo týdenní. V běžném životě se události často opakují. Pokud by v plánovači nebyla možnost nastavit opakování museli by uživatelé vytvářet každý výskyt události samostatně, což by bylo v některých případech nerealizovatelné.

6.2.3 Roční přehled

Při zkoumání vlastností a funkcí jiných nástrojů pro plánování jsem také zkoumal možnosti nastavení přehledů kalendářů. Všechny mají samozřejmě denní, týdenní a měsíční pohled, ale většina z nich nemá možnost zvolit přehled, který by uživateli zobrazil celý rok. Tuto schopnost považuji za velice důležitou, neboť v ročním přehledu se uživatel může velice jednoduše orientovat a získá jím unikátní pohled na využití svého času v průběhu roku.

Roční přehled je tedy důležitý pro orientaci a přehled o využití času v průběhu roku, nicméně jeho rozsáhlost neumožňuje zobrazit všechny důležité informace ke každému dni tak, aby nepřestal být přehledným. V ročním přehledu nám tedy bude stačit označení dní, které obsahují nějakou událost. Detailnější informace budou například poskytovány při vystavení kurzoru myši nad daným den.

6.2.4 Synchronizace

Pro možnost synchronizace plánovačů na dvou různých PC bude na internetu možno si svá data ukládat na internet. Pro možnost ukládání dat na internet se budou muset uživatelé nejdříve registrovat. Registrace bude probíhat uvnitř aplikace. Ukládání dat na internet se bude poté provádět vždy, když bude uživatel přihlášen. Toto ukládání nebude muset uživatel explicitně vyžadovat, bude se provádět automaticky.

6.2.5 Sdílení dat

Jelikož si budou moci uživatelé ukládat svá data na internet, bude díky tomu možné implementovat funkci sdílení těchto dat. Uživatelé plánovače registrovaní na internetu budou moci sledovat události a plány jiných uživatelů. Stačí být pouze registrovaným uživatelem a vybrat si ze seznamu ostatních

registrovaných uživatelů. Díky této funkci budou mít přehled o jejich času a tak si s nimi budou moci lépe plánovat schůzky nebo telefonáty apod.

6.2.6 Vyhledávání vhodných termínů

Plánovač bude obsahovat, podle mého názoru, velice zajímavou a jedinečnou funkci. Tou bude funkce vyhledávání vhodných termínů událostí. Tato funkce bude uživatelům usnadňovat vyhledávání volných termínů, kam bude možno umístit událost. Tuto funkci uživatelé nejvíce využijí a ocení při vytváření schůzek s jinými uživateli, se kterými sdílí naplánované události.

K vytvoření schůzky jim bude stačit vybrat uživatele, se kterými chtějí uskutečnit schůzku. Po vybrání účastníků setkání vybere délku trvání, minimální hodinu začátku, maximální hodinu začátku, rozsah dat a dny, ve kterých se může schůzka konat. Program poté vyhledá v kalendářích všech účástníků datum a čas, který bude vyhovovat všem účastníkům. Těchto termínů nabídne uživateli několik a dá mu na výběr jeden z nich.

6.2.7 Správa úkolů

Nezbytnou součástí každého plánovače je správa úkolů. Mají ji všechny plánovače a ani ten můj nebude výjimkou. Správa úkolů však bude vylepšená o možnost tvoření podúkolů k úkolům. To umožní rozdělení složitého úkolu do menších a méně složitých částí.

Ke každému úkolu si budou uživatelé moci psát poznámky. S možností psát si poznámky k úkolům jsem se ještě v žádném plánovačí nesetkal, i když je to podle mého názoru velice užitečná vlastnost. Kam jinam si psát poznámky, které se vztahují k úkolu, něž přímo k úkolu samotnému. S touto funkcí již nebudeme muset pátrat po ztracených poznámkách k úkolům.

6.2.8 Efektivní využívaní plochy

Tato funkce souvisí s přehledností kalendáře. Obsahují ji všechny plánovače. Ostatní plánovače však nedokáží využívat plochu tak efektivně jako můj plánovač.

Při plánování mohou a často také nastávají situace kdy se jednotlivé časové události kryjí. Řešením je rozdělení šířky podle počtu kryjících se událostí a poskládat tyto události vedle sebe, čímž vzniknou jakési sloupce. Pokud není vhodně navržený algoritmus, může to vést k vytvoření zbytečně velkého počtu sloupečků. Funkce bude provádět uspořádání událostí v kalendáři tak, že z minimalizuje počet takovýchto sloupečků a využije každé místo k tomu, aby bylo možné zobrazit co největší detaily.

6.2.9 Adresář

Adresář není standardní funkcí, kterou plánovač času potřebuje. Já jsem se však rozhodl tuto funkci do svého plánovače přidat. Jednak si myslím, že není špatné mít všechno pohromadě v jedné aplikaci a také to přinese výhodu při vytváření schůzek.

Domnívám se, že při plánování času se často plánují schůzky s jinými lidmi, nikoli jen události týkající se pouze vlastníka plánovače. A je zřejmé, že pokud si naplánujeme s někým schůzku chceme vědět s kým to je. Proto si do událostí, které plánujeme jako schůzky přidáváme jejich účastníky. Pokud máme všechny tyto účastníky již v adresáři, můžeme je při každé nové schůzce pouze vkládat.

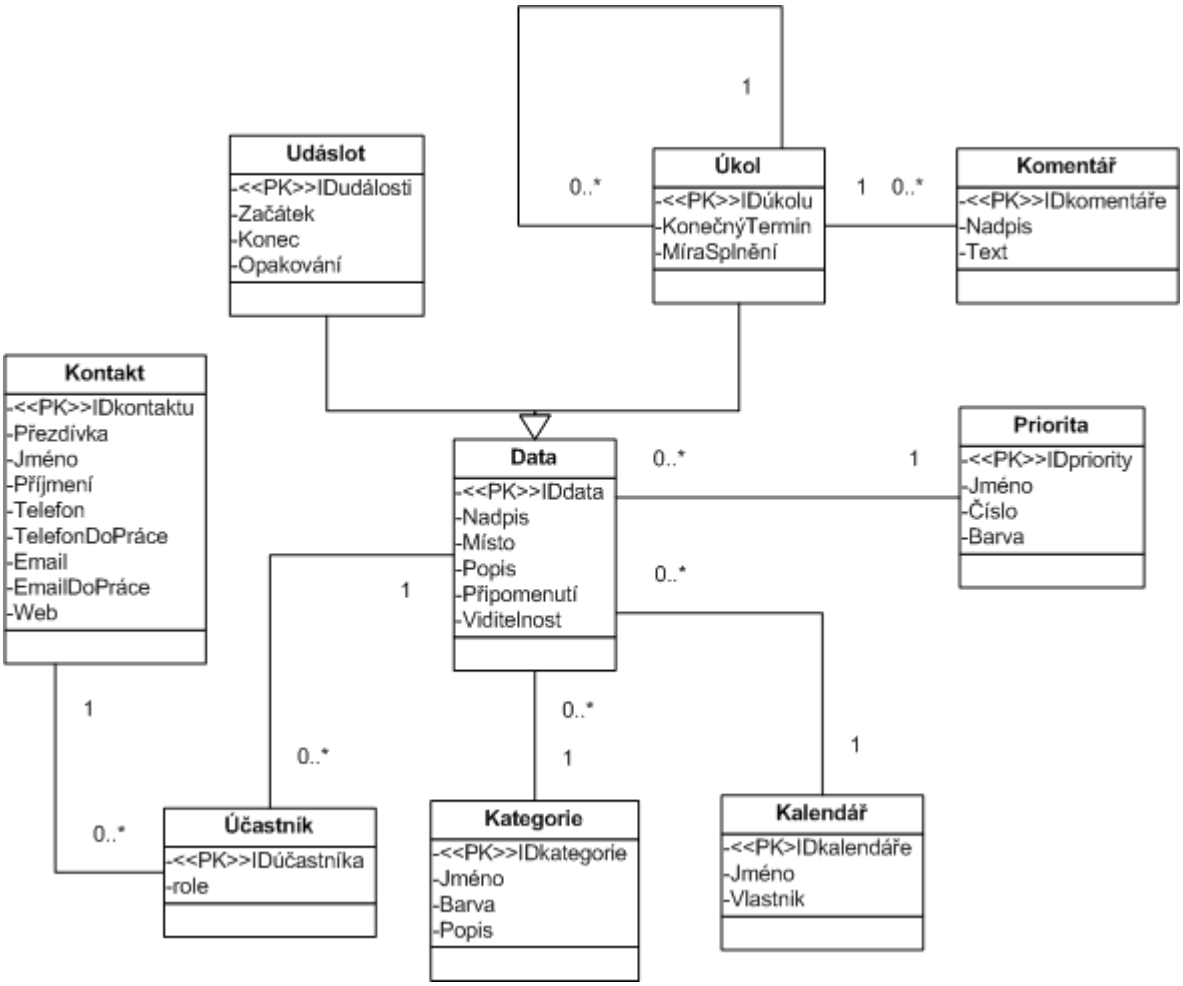
7 Implementace

V této části dokumentu bude popsána implementace plánovače podle specifikace z předchozí kapitoly.

Celý projekt je rozdělen do dvou dynamických knihoven a spustitelného souboru. Ke svému běhu potřebuje program kromě dvou vlastních knihoven ještě knihovny pro práci s databází a knihovnu pro práci s formátem CSV.

7.1 E-R diagram

Na obrázku je vidět E-R diagram mého systému. V tomto diagramu je možné vidět, že události a úkoly mají z větší části stejné atributy. Toho bude využito v celé aplikaci.



Obrázek 2: E-R diagram

7.2 Databáze

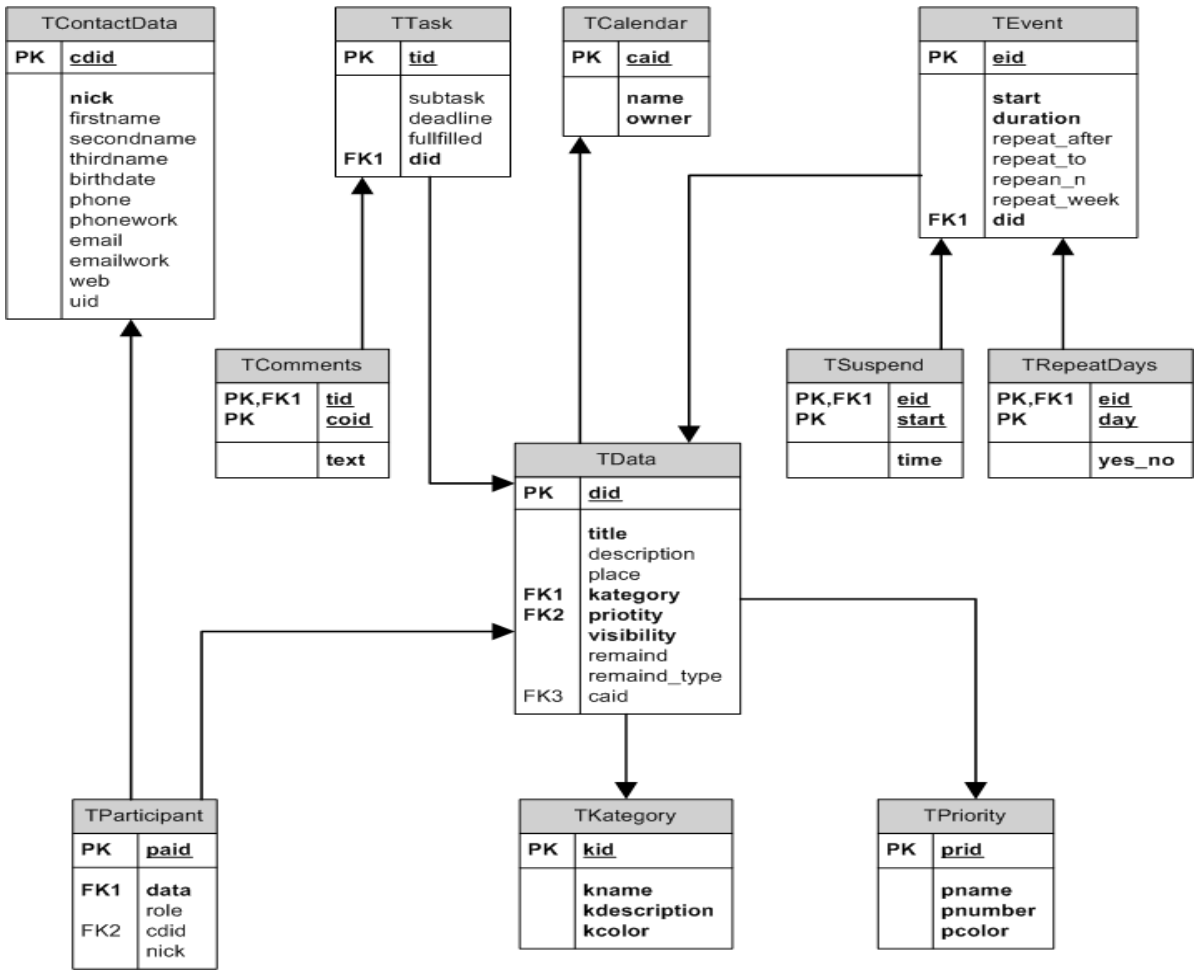
Systém plánovače bude zahrnovat dvě samostatné databáze. Jedna bude uložena na lokálním stroji uživatele a druhá na serveru. Lokální databáze bude používat databázový systém SQLite a databáze na serveru databázový systém MySQL. Tyto databáze jsou si hodně podobné. Obsahují stejné tabulky.

Databáze na serveru obsahuje navíc tabulku s registrovanými uživateli. Všechny tabulky na serveru mají navíc jeden sloupec, jímž je sloupec vlastníka záznamu. Tento sloupec slouží jako diskriminátor ve složeném primárním klíči pro každou tabulku. Primární klíč každé tabulky je tedy složen z ID položky a ID vlastníka položky. Každá tabulka v lokální i serverové databázi obsahuje sériové číslo položky. S každou změnou se toto číslo zvyšuje.

Pro práci s lokální aplikací je použita volně šiřitelná knihovna SQLite .NET Wraper. Na straně serveru pracují PHP skripty.

7.2.1 Schéma databáze

Na tomto obrázku je vidět schéma lokální databáze.



Obrázek 3: schéma databáze

7.3 Objektový model

Objektový model aplikace je navržen pro jednoduchou práci s databází. Základními stavebními kameny celé aplikace jsou databázové třídy.

Základní třída se jmenuje *Table*. Z této třídy dále dědí třídy pro práci s tabulkami celého systému. Třída table abstrahuje jednu tabulku z databázi. Poskytuje základní metody pro práci se záznamy v tabulce. Obsahuje kolekci struktur *TableColumn*, která obsahuje definice sloupců tabulky. Struktura *TableColumn*, je definicí jednoho sloupce. Struktura udržuje data potřebná k definici sloupce, a navíc uživatelské jméno tohoto sloupce a metodu pro čtení řetězcových hodnot z databáze.

Třída *TableRow* abstrahuje jeden řádek v databázi. Uvnitř třídy je slovník, který sdružuje jméno sloupce a hodnotu tohoto sloupce v tomto řádku. Třída neposkytuje přístup k atributům a může tedy být použita pro jakékoliv databázové schéma. Slouží jako bázová třída pro třídy událostí, úkolů, kalendářů a dalších. Ve spoustě metod, které zahrnuje, je nejzajímavější dvojice metod *ToXmlNode()* a *FromXmlNode()*. První z uvedených převádí řádek tabulky do formátu XML. Ve formátu XML je každá hodnota sloupce uzavřena mezi značkami nesoucími jméno tohoto sloupce. Druhá metoda pak formát XML převádí zpět na řádek tabulky. Zajímavou metodou je také metoda *GetPanel()*, která vytvoří grafickou reprezentaci řádku. V grafické reprezentaci jsou uvedeny v jednom sloupci jména atributů a v druhém jsou textová políčka obsahují hodnoty těchto atributů. Hodnoty atributů je možné v grafické reprezentaci také měnit.

Další třídou pro práci s databází je třída *RelationShip.* Tato třída reprezentuje jeden vztah mezi tabulkami. Po svém vytvoření se přidá do seznamů vztahů zúčastněných tabulky. Tato třída poté poskytuje práci se vztahem. Umožňuje získávání řádků ve vztahu a řeší integritu dat při mazání primárních klíčů.

Nejdůležitější třídou pro práci s databází je třída *Database*. Tato třída ve svém konstruktoru otevře soubor databáze a zjistí za obsahuje všechny tabulky, které obsahovat má. Před kontrolou jsou vytvořeny instance všech tříd reprezentující tabulky systému, po jejich vytvoření se poté kontroluje zda jsou v databázi vytvořeny a pokud v databázi nejsou vytvoří je. Třída udržuje seznam všech tabulek v systému po celou dobu běhu aplikace a na požádání poskytuje jejích instance. Třída je vytvořena podle návrhového vzoru *Singleton*, což znamená že v celé aplikaci existuje pouze jedna instance této třídy. Jediná instance je dostupná přes statickou vlastnost třídy *Databse*.

Aplikace pracuje s databázovými tabulkami v paměti. Po vytvoření instance tabulky jsou z databáze přečteny všechny její řádky a instance tabulky je po zbytek běhu aplikace udržuje tyto záznamy v paměti. Nově vytvářené záznamy jsou přidávány do seznamu záznamů tabulky a mazané jsou z toho seznamu odebírány. Aby se při každé operaci se záznamem nemusela volat explicitně metoda tabulky má třída *TableRow* definovány tři události. Každá událost je vyvolána při specifické operaci se záznamem. Událost *InserEvent* je volána při ukládání nového záznamu. Událost *UpdateEvent* je volána při ukládání změn v záznamu. Poslední událostí je *DeleteEvent,* která je

volána při mazání existujícího záznamu. Po vytvoření instance záznamu přidá tabulka všem těmto událostem chování. Tyto události poté usnadňují práci se záznamem a udržují seznam záznamů v paměti shodný s obsahem tabulky v databázi.

Důležitou třídou aplikace je třída *UnixTimeStamp*. Jedná se o statickou třídu se dvěma nejdůležitějšími metodami *To* a *From.* Tyto metody převádí typ *DateTime* na unixovou časovou známku a naopak. Unixová časová známka je počet sekund od počátku Unixové Epochy 1. lednu 1970. Její číselná reprezentace přináší mnohé výhody při práci s časem, a proto je používána v databázi místo řetězců reprezentujících čas.

7.4 Komunikace po síti a synchronizace

Pro komunikaci jsem vytvořil knihovnu Network.dll. Obsahuje třídy poskytující připojení k serveru, odesílání dat na server a příjímání dat ze serveru. Komunikace probíhá pomocí protokolu HTTPS. Server a klient si navzájem posílají XML data.

Komunikaci po sítí zprostředkovávají třídy platformy .NET framework. Pro komunikaci je použita třída *WebClient*. Třída komunikuje se serverem protokolem HTTP. Umí také komunikovat pomocí protokolu HTTPS, který je v aplikaci použit pro bezpečnost přenášených dat. Pro komunikaci je však potřeba nastavit síťová bezpečnostní pravidla, neboť server požaduje autorizaci pomocí http autorizaci. Pro nastavení těchto pravidel stačí vytvořit instanci *NetworkCrendetial* v jejímž konstruktoru jsou jí předány přihlašovací údaje. Tato instance je poté předána instanci webového klienta. Před začátkem komunikace je však ještě potřeba nastavit odpověď na dotaz, zda chceme komunikovat se serverem. Tato otázka je kladena webovým klientem kvůli použití protokolu HTTPS. Odpověď je vyřešena předáním anonymního delegáta, který nedělá nic jiného, než že pokaždé vrací hodnotu *true.*

Synchronizace se provádí po přihlášení a v daném intervalu. Před synchronizací klient zjistí sériové číslo databáze na internetu. Pokud je číslo vyšší než lokální, je lokální databáze zastaralá a bude obnovena ze serveru. Pokud je číslo nižší, je naopak zastaralá databáze na serveru. Po zjištění sériových čísel se přistupuje k samotné synchronizaci. Synchronizace je rozdělena do dvou částí. První část synchronizuje data lokálního uživatele, která se nesdílejí. V druhé části jsou již synchronizována všechna data. Při synchronizaci klient postupně požádá server o sériová čísla řádků všech tabulek. Tato čísla poté porovnává se sériovými čísly v lokální databázi. Pokud se sériová čísla liší, je chování stejné jako u sériových čísel databází.

Při komunikaci pracují na straně serveru PHP skripty. Tyto skripty umí vytvořit z jakéhokoliv řádku v každé tabulce jeho XML reprezentaci, stejnou jakou vytváří třída řádku *TableRow.* Také umí z této reprezentace vytvořit příkaz, jímž řádek uloží nebo obnoví nebo nahradí ve své databázi.

7.5 Import/Export

Jako formát pro import a export byl zvolen CSV formát z aplikace MS Outlook. Pro čtení toho formátu jsou v prostředí .NET dostupné technologie. Nastal však problém, neboť ve standardním CSV se jako oddělovač dat používá znak ';' a ve formátu z MS Outlook se používá jako oddělovač znak ','. Samotný rozdíl mezi znaky by nebyl problémem, protože je možné pomocí souboru *schema.ini* změnit nastavení ovladače, a tak mu změnit oddělovací znak. Bohužel změna oddělovače nebyla možná, neboť ovladač pro čtení CSV se bránil použití znaku ','. Proto je ke čtení formátu CSV využívána volně dostupná knihovna *CsvReader*. Pro zápis není potřeba speciálních technologií. Formát CSV je obyčejný textový soubor obsahující na prvním řádku názvy položek oddělených oddělovacím znakem, a na dalších řádcích jsou data oddělena stejným oddělovacím znakem.

Import začíná vybráním souboru, který chceme importovat. Po vybrání souboru přebírá řízení instance třídy *CSVExporterImpoter*, která provádí samotný import. Postupně zpracovává řádky souboru, dokud není soubor celý přečten. Každý řádek se pak pokouší převést na jednu událost. Pokud se při převodu vyskytne chyba, u povinných položek není převod této události dokončen a pokračuje se převáděním další události. Pokud jsou chyby v položkách, které nejsou povinné, pokračuje se v převádění události a nakonec je uložena.

Export začíná vybráním kalendáře k exportu. Po vybrání tohoto kalendáře je řízení předáno instance třídy *CSVExporterImpoter*, která provádí samotný export. Nejdříve si vyžádá všechny události v kalendáři, který se má exportovat. Poté každou událost převede na řetězec obsahující data události oddělená čárkou. Převedené řetězce jsou zapisovány do souboru na samostatné řádky. Před zápisem řádku do souboru je zapsán řádek s názvy položek.

7.6 Upomínky

Důležitou funkcí plánovače jsou upomínky. Upomínky budou uživatelům připomínat jejich naplánované události. Funkce je v mém plánovači řešena pomocí druhého vlákna aplikace. Toto vlákno každou minutu kontroluje události a úkoly. Kontrola úkolů je jednoduchá, neboť úkoly se nemohou opak, a tak jejich připomenutí má jeden přesný termín. Kontrola událostí je složitější.

Události se mohou opakovat, díky čemuž není jednoduché určit přesné datum připomenutí. Vypočítání data připomenutí provádí události automaticky. Tento výpočet je založen na prozkoumání všech možných termínů. Zkoumání termínů začíná v prvním dni výskytu a končí jakmile najde datum připomenutí, které je buď v budoucnosti nebo právě teď.

Když vlákno upomínek objeví událost nebo úkol, který se má připomenout, musí volat speciální metody hlavního okna, které vyvolají zobrazení dialogu připomenutí a zablokování hlavního okna. Hlavní okno je blokováno tak dlouho, dokud nejsou ukončeny všechny otevřené dialogy

upomínek. Speciální metody volané vláknem jsou použity z důvodu bezpečné komunikace mezi vlákny. Tyto metody volají jiné požadované metody nyní již bezpečně.

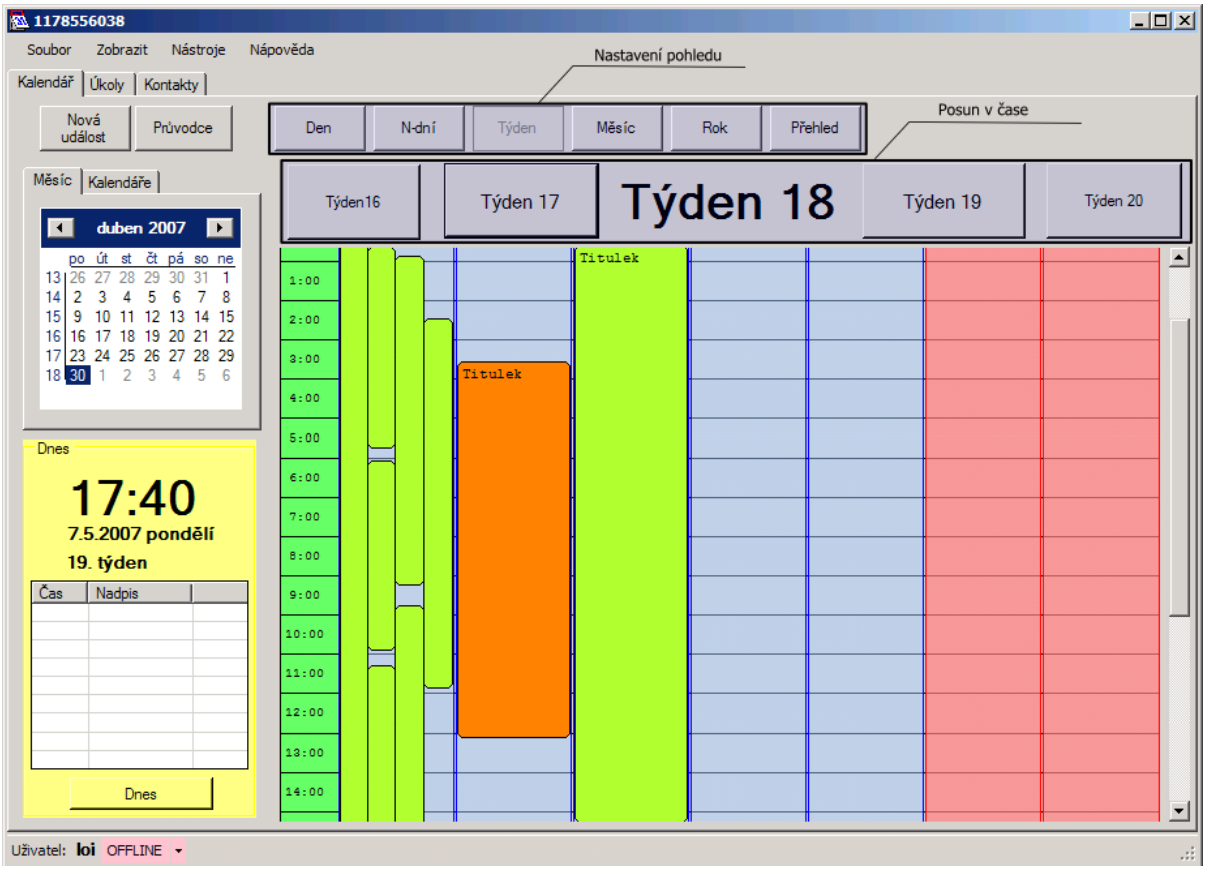
7.7 Uživatelské rozhraní

Pro tvorbu uživatelského rozhraní jsem využíval namespace *System.Forms*. Uživatelské rozhraní se skládá z hlavního okna a několika dialogů.

Hlavní okno je rozděleno pomocí záložkového layoutu do tří záložek. V první záložce jsou zobrazeny události. Obsahuje také ovládací prvky pro změnu přehledů, posouvání v čase a práci s událostmi.

V záložce druhé se jménem Úkoly najdeme správu úkolů. Obsahuje kompletní sadu nástrojů pro práci s úkoly a zobrazuje strom úkolů na jedné straně a na druhé straně je zobrazen detail označeného úkolu. U detailu úkolu je zobrazen seznam komentářů.

Na poslední záložce se jménem kontakty je situována správa kontaktů. V jedné části je úplný seznam a v části druhé je zobrazen detail jednoho kontaktu.



Obrázek 4: Uživatelské rozhraní

7.7.1 Uživatelská nastavení

Uživatelé si mohou aplikaci přizpůsobovat tak, aby jim vyhovoval její vzhled. Toto nastavení je ukládáno v podobě XML do souboru *config.xml*. Tento soubor je uložen v adresáři specifickém přihlášenému uživateli, a tak může mít každý uživatel své vlastní nastavení.

7.7.2 Uspořádávání plochy

Na obrázku je vidět týdenní pohled a úplně vlevo vedle hodin jsou poskládány zelené obdélníčky. Tyto zelené obdélníčky reprezentují události a jsou poskládány tak, aby efektivně využívaly šířku dne.

Algoritmus pro tuto funkci vypadá následovně:

serad události vyskytující se v tomto dni;

foreach (události v tomto dni *)*
udělej z události graficky obdélníky;
foreach (vytvořené obdélníky z událostí *)*
if (překrývají-li se *)*
přidej nový do překrývajících se k již vytvořenému;

V této chvíli jsou vytvořeny všechny obdélníčky reprezentující události v jednom dni.

foreach (obdélníky z událostí *)*
if(obdélník není vyřešen *)*
/* odstraní nežádoucí sousedy
nežádoucí sousedé jsou takový, kteří mohou být umístěni tak, že neovlivní mou šířku */
foreach (překrývající se obdélníky *)*
foreach (předcházející obdélníky *)*
if (předcházející se nekryje s vybraným překrývajícím)
/* zjistí, zda se je místo pod předcházejícím obdélníčkem */
foreach (obdélníky pod předcházejícím *)*
if (kryje se)
přejdi na další předchozí obdélník;

Nyní má každý obdélník jen nezbytný počet obdélníků v seznamu překrývajících se.

foreach (obdélníky z událostí *)*

if (nemá nastavenou šířku)

zjisti nejdelší cestu k přes obdélníčky;

délkou cesty poděl možnou šířku a výsledek nastav jak svou vlastní šířku;

vypočti zbylou šířku a volej nastavení šířky a pozici pro všechny své překrývající obdélníky;

V této chvíli je algoritmus dokončen a obdélníky událostí jsou nastaveny.

7.7.3 Vyhledávání volných termínů

V hlavním okně na záložce Kalendář se nachází tlačítko s nápisem Průvodce. Po stisknutí tlačítka se objeví dialog, na němž jsou ovládací prvky, určené pro nastavení parametrů vyhledávání volného termínu. Uživatelé si zde mohou vybrat účastníky, nastavit rozmezí dat, vybrat dny v týdnu, rozmezí hodin ve dni a délku trvání události. Po tomto nastavení stisknutím tlačítka OK se vyhledají volné termíny a jsou nabídnuty uživateli k výběru.

Algoritmus vyhledávání volného termínu po požadavku na vyhledání postupně prochází den po dni, a v každém z těchto dní vyhledá události uživatele a účastníků. Po vyhledání všech událostí ve dni převede události do časových úseků. Kryjící se události se spojují v jeden časový úsek. Tyto úseky jsou uloženy v seznamu využitých časových úseků. Seznam použitých časových úseků je poté převeden na seznam volných časových úseků. Po převedení jsou odstraněny krátké úseky. Zbytek volných úseků je seřazen podle své délky a začátku a je nabídnut uživateli k výběru.

8 Závěr

Navržený plánovač jsem vytvořil a otestoval jeho schopnosti. Jako nástroj pro plánování času se mi jeví použitelným. Má podle mého soudu všechny potřebné schopnosti nástroje pro plánování. Navíc má funkci vyhledávání volných termínů, a jeho možnosti synchronizace jsou na vyšší úrovní než u ostatních plánovačů. Jeho další předností je přehlednost, díky efektivnímu využívání plochy a ročnímu přehledu. Díky jeho integrované správě kontaktů ulehčuje plánování schůzek. Správa úkolů má také vlastnosti, které nejsou příliš časté. Je to možnost vytváření podúkolů a možnost vytváření komentářů k úkolům.

Jako další funkce by bylo vhodné implementovat import a export dalších formátů. Také by bylo vhodné vylepšit vlastnosti sdílení, především bezpečnost dat. V současném stavu je možné sledovat události kteréhokoliv registrovaného uživatele. Uživatelé by měli mít možnost řídit přístup ke svým datům, a to buď pomocí seznamu povolených uživatelů, nebo přístupovými právy podobnými s těmi v operačních systémech.

Literatura

[1] Robinson, S. *C# Programujeme* profesionálně. Computer Press 2003

[2] Hruška, T.; Křivka, Z., *Informační systémy Studijní opora* Verze: prosinec 2006 Dokument dostupný na URL https://www.fit.vutbr.cz/study/courses/WAP/private/opory/OporaIIS2PISPojemDataProcesyTransakce.pdf

[3] Zendulka, J.; Rudolfová, I., *Databázové systémy Studijní opora* Verze: 18. 7. 2006. Dokument dostupný na URL https://www.fit.vutbr.cz/study/courses/IDS/private/IDS_predn.pdf

[4] Osovský, M., *Programování v .NET.* Dokument dostupný na URL https://www.fit.vutbr.cz/study/courses/MW5/private/MW5_1.ppt

[5] Běhálek, *M. Programovací jazyk C#* Dokument dostupný na URL http://www.cs.vsb.cz/behalek/vyuka/pcsharp/texty/

[6] Wikipedia, The Free Encyclopedia, *Databse* 8. května 2007. Dokument dostupný na URL http://en.wikipedia.org/wiki/Database

[7] Wikipedia, The Free Encyclopedia, *Informační systém* 8. května 2007. Dokument dostupný na URL http://cs.wikipedia.org/wiki/Informa%C4%8Dn%C3%AD_syst%C3%A9m

[8] SQLite, *SQLite,* Dokument dostupný na URL http://www.sqlite.org/

[9] The PHP Group, *PHP,* Dokument dostupný na URL http://www.php.net/

Seznam příloh

Příloha 1: Instalace

Aplikace nevyžaduje instalaci. Pro svůj běh však potřebuje operační prostředí .NET 2.0. Po spuštění aplikace jsou vytvořeny všechny potřebné soubory, a aplikace je připravena k použití.

Příloha 2: Manuál

Tento manuál je určen k základnímu seznámení s funkcemi aplikace plánovače. Naučí uživatele události a úkoly vytvářet, upravovat a mazat. Také jej seznámí s dalšími užitečnými funkcemi. Uživatelské rozhraní je však velmi intuitivní a přehledné, takže není potřeba příliš podrobného manuálu.

Vytvoření události

Uživatelé mají několik možností jak vytvořit událost. Tou nejsnazší možností je kliknout na tlačítko "Přidat událost". Po kliknutí na tlačítko se zobrazí dialog pro přidání události. Stejný dialog se uživateli zobrazí i při dvojkliku v kalendáři nebo kliknutím pravým tlačítkem do volného místa v kalendáři a vybrání položky "Přidat událost" z kontextového menu. V předchozích možnostech se vytvoří událost s délkou trvání jednu hodinu. Pro vytvoření události s jinou délkou trvání a časem začátku je možné vytvořit událost natažením v kalendáři. Stačí stisknout levé tlačítko myši a táhnout, v kalendáři se zobrazí světlý obdélník, který reprezentuje novou událost. Po puštění tlačítka se opět otevře dialog pro přidání události.

Úprava události

Možností jak upravit událost máme také více. Je možné ji otevřít pomocí dvojkliku na událost v kalendáři. Také je zde možnost kliknout pravým tlačítkem na událost v kalendáři a zvolit z menu "Editovat". Opět se otevře dialog, který se otevírá při přidávání události.

Pokud se událost, kterou chceme upravit, nachází v dnešním dni, je uvedena v seznamu v levé části aplikace pod měsíčním kalendářem. Zde je možné na událost dvakrát kliknout nebo si událost vybrat a stisknout tlačítko "Enter".

Pro změnu času začátku je možné událost posunout přímo v grafickém kalendáři. Stačí vystavit kurzor myši nad událost a stisknout levé tlačítko. Poté je možné s událostí různě pohybovat.

Smazání události

Při otevření události je v levém dolním rohu vidět tlačítko "Smazat", kterým je možné událost odstranit. Je také možné, stejně jako při otevření události, kliknout pravým tlačítkem na událost a v menu vybrat položku "Smazat".

Změna pohledu

Změnu pohledu je možné provést pomocí kliknutí na jedno ze šesti tlačítek, ve vrchní části záložky “Kalendář“. Po kliknutí na některé z nich dojde ke změně zobrazovaného časového úseku.

Posun v čase

Přesun v času je možné provést pomocí tlačítek přímo nad plochou kalendáře. Je také možné přesunovat se v čase pomocí měsíčního kalendáře u levého okraje aplikace.

Výběr zobrazovaných kalendářů

Každou událost je možné zařadit do kalendáře. Aplikace poté umožňuje výběr kalendářů, jejichž události se mají zobrazit na ploše kalendáře. Tento výběr je možné uskutečnit v záložce pojmenované “Kalendáře“ . Záložka se nachází nad měsíčním kalendářem. Po přechodu na ni je měsíční kalendář skryt a je zobrazen seznam s kalendáři. U každého kalendáře je zatrhávací tlačítko, jehož zatržením dojde k zobrazení kalendáře.

Vytvoření úkolu

Vytvoření úkolu je možné na záložce “Úkoly“. Na této záložce se také nachází seznam všech úkolů a jejich detaily. Vytvoření úkolu je možné kliknutím na tlačítko “Nový úkol“. Po kliknutí na něj se otevře stejný dialog jako při vytváření nové události.

Úprava úkolu

Opět je několik možností jak začít upravovat úkol. Všechny tyto možnosti vedou k otevření dialogu s možností úpravy úkolu. Tou první možností je vybrat v seznamu a kliknout na tlačítko “Upravit úkol“. Dále je tu možnost dvojkliku na vybraný úkol, nebo stisknutí klávesy “Enter“. Procentuální splnění úkolu je možné měnit při vybrání úkolu. Po vybrání úkolu se zobrazí jeho detail, a zde je možné změnit počet procent.

Smazání úkolu

Tuto operaci je možno provést při úpravě úkolu nebo vybráním úkolu ze seznamu, a s stisknutím tlačítka “Delete“.

Podúkoly

Každý úkol se může skládat z částečných úkolů. Tyto částečné úkoly lze k úkolům kdykoliv přidat. Přidání je možné kliknutím na tlačítko “Přidat podúkol“. Nebo je také možné pomocí myši přetáhnout již vytvořený úkol nad jiný a tím z něj udělat úkol podřízený.

Komentáře k úkolům

Pokud máme v seznamu úkolu vybrán nějaký úkol, je možné si k němu uložit neomezený počet komentářů. Komentáře lze libovolně upravovat a mazat. Tyto úkony je možné provádět pomocí tlačítek v sekci “Komentáře“.

Úprava kategorií

K úpravě kategorií je možné přistupovat pouze přes položku menu v sekci “Nástroje“. Po zobrazení sekce a kliknutí na položku “Kategorie“, se otevře dialogové okno se správou kategorií.

Úprava kalendářů

K úpravě kalendářů je možné přistupovat pouze přes položku menu v sekci “Nástroje“. Po zobrazení sekce a kliknutí na položku “Kalendáře“, se otevře dialogové okno se správou kalendářů.

Úprava priorit

K úpravě priorit je možné přistupovat pouze přes položku menu v sekci “Nástroje“. Po zobrazení sekce a kliknutí na položku “Priority“ se otevře dialogové okno se správou priorit.

Import

Ovládací prvek, jimž je možno provést import dat, je umístěn v menu aplikace v sekci “Soubor“. Po kliknutí na položku “Import“, se aplikace dotáže na jméno souboru, který se má importovat. Po zadání jména souboru je proveden import. Po importu dat je zobrazeno dialogové okno s výsledkem předchozí akce.

Export

Ovládací prvek, jimž je možno provést export dat, je umístěn v menu aplikace v sekci “Soubor“. Po kliknutí na položku “Export“ se aplikace dotáže na jméno kalendáře, který chceme exportovat. Pro zvolení je nutné kalendář označit a kliknout na tlačítko “OK“. Po dokončení exportu je zobrazeno dialogové okno s výsledkem akce.

Zobrazení sdíleného kalendáře

Pro možnost zobrazení kalendáře jiného uživatele je nutné se nejdříve zaregistrovat. Registrace se provádí v dialogovém okně možností. Do tohoto okna se dostaneme kliknutím na položku “Možnosti“ v sekci menu “Nástroje“. Po otevření okna možností přejdeme na záložku “Synchronizace“. Na této záložce jsou tři textová políčka do kterých vyplníme registrační údaje. Všechny tyto položky jsou povinné.

První položkou pro registraci je login, jedná se o uživatelské jméno. Login musí mít minimálně tři znaky. Jakmile máme login ve správném tvaru můžeme přejít na první zadání hesla. Heslo musí obsahovat minimálně osm znaků. Po zadání korektního hesla je možné přejít na kontrolní zadání hesla. Když jsou hesla stejné, povolí se tlačítko “Registrovat“ a po kliknutí na něj se provede registrace. Pokud je registrace úspěšná bude možné se přihlásit.

Když jsme přihlášeni můžeme si přidat kontakt z internetu mezi své kontakty. Přidání se provede po kliknutí na položku “Přidat kontakt“ v sekci menu “Nástroje“. Po otevření dialogu přidání kontaktu označíme kontakt, který chceme přidat, a klikneme na tlačítko “Přidat“. Pokud již nechceme přidávat další kontakt můžeme opustit dialog.

Teď je vše připraveno na zobrazení cizího kalendáře. Stačí již jen zvolit zobrazovaný kalendář, se jménem přidaného kontaktu (viz. Výběr zobrazovaných kalendářů) .

Vyhledání volného termínu

Funkce vyhledávání volných termínů se skrývá pod tlačítkem “Průvodce“. Po kliknutí na toto tlačítko se objeví dialog, ve kterém se nastaví parametry vyhledávání. Nejdříve jsou vybrány kontakty, se kterými chceme schůzku. Poté se volí délka trvání této akce. Dále je možné nastavit rozmezí hodin, rozmezí dat a dny v týdnu, ve kterých se akce může konat. Po kliknutí na tlačítko “OK“ aplikace vyhledá volné termíny a nabídne je k vybrání. Zde je možné vybrat si jeden termín. Po vybrání je otevřen dialog přidání nové události.