VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

JAVA EE ORGANIZÉR – MODUL KALENDÁŘ

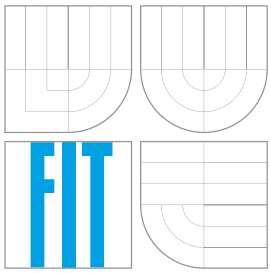
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN KOVÁŘ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

JAVA EE ORGANIZÉR – MODUL KALENDÁŘ

JAVA EE ORGANIZER – CALENDAR MODULE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

JAN KOVÁŘ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. RADEK KOČÍ, Ph.D.

BRNO 2009

Abstrakt

Tato práce pojednává o analýze stávajících organizérů a přístupů, které jsou s těmito aplikacemi spojeny. Další část práce popisuje model třívrstvé architektury a její části. Jsou zde také popisovány některé základní technologie, které jsou obsaženy v platformě Java EE a nebo jsou s touto platformou úzce spjaty. Další část práce je věnována popisu implementace a návrhu aplikace Java EE Organizér, což je aplikace, jejímž úkolem je správa událostí, úkolů a kontaktů.

Abstract

This thesis analyses existing organizers and approaches associated with these applications. Next part of the work describes three tier architecture and its individual parts. It also described some of the basic technologies which are included in the Java EE platform, or which that are closely related to this platform. Last part of the work is dedicated to describing the implementation and design of Java EE Organizer. It is an application which target is administration of event, task and contact manager.

Klíčová slova

Java EE, Java, modul kalendář, organizér, databázová aplikace, síťová aplikace, Glassfish, Firebird

Keywords

Java EE, Java, calendar module, organizer, databaze aplikation, network aplikation, Glassfish, Firebird

Citace

Jan Kovář: Java EE Organizér – modul kalendář, bakalářská práce, Brno, FIT VUT v Brně, 2009

Java EE Organizér – modul kalendář

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Radka Kočího, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jan Kovář
20. května 2009

Obsah

Kapitola 1

Úvod

Tato bakalářská práce představuje teoretický základ pro program Java EE Organizér, který byl vytvořen v rámci tříčlenného týmu. Tvůrčí činnost na programu byla rozdělena následujícím způsobem: Pavel Palát – architektura a návrh databáze, Petr Černý – modul úkoly a Jan Kovář – modul kalendář. Účelem výsledného organizéru je usnadnění správy času a úkolů, protože právě ta získává v dnešní době stále více na důležitosti. Existuje mnoho možností, jak zaznamenávat a spravovat úkoly, a to od nejzákladnějších a nejstarších jako tužka a papír, přes jednoduché organizéry v mobilních telefonech až po multifunkční a složité online aplikace určené pro počítače, notebooky a „chytré telefony“, které v podstatě představují druh počítačů, např. MDA.

Řazení kapitol v práci vychází z logické následnosti. Nejdříve jsou popsána současná již realizovaná řešení a následující kapitoly popisují naši práci od nejobecnějších rozhodnutí po konkrétní práci na tvorbě programu. V kapitole 2 Analýza stávajících řešení jsou popisovány stávající přístupy k tvorbě organizérů a zahrnuty jsou také rozbory několika již vytvořených organizérů, u kterých jsou popsány i jejich funkce a působení na uživatele. Kapitola 3 Architektura se zabývá architekturou naší aplikace a také obecně vysvětluje architekturu využívající trojvrstvý model. Kapitola 4 Technologie popisuje a vysvětluje technologie používané při tvorbě aplikací postavených na Java Platform, Enterprise Edition (Java EE) a také popisuje výhody a nevýhody jednotlivých technologií. Kapitola 5 Návrh a implementace je zaměřena na námi vytvořený organizér Java EE Organizér. V této kapitole jsou podrobně popsány i některé složitější nebo důležitější algoritmy. Závěrečná kapitola 6 obsahuje celkové zhodnocení práce a obecné shrnutí zkušeností získaných při použití těchto technologií.

Kapitola 2

Analýza stávajících řešení

Tato kapitola popisuje tři základní přístupy, které se využívají při tvorbě elektronických organizérů. Jedná se o online aplikace, dále o aplikace se synchronizací a o lokální aplikace bez synchronizace. V kapitole jsou také blíže popsány některé již existující organizéry a jejich možnosti.

Dnešní hektická společnost se bez organizace času neobejde, proto se organizátory stávají součástí mnoha technických i klasických pomůcek, jejichž cílem je pomoci uživatelům zorganizovat a efektivněji využívat jejich čím dál dražší čas. V současnosti jsou, dle mého názoru, nejčastěji využívanými i vytvářenými organizátory času elektronické organizéry, které se nacházejí v celé škále zařízení (např. v smartphonu, notebooku, mobilním telefonu, apod.). To je jedním z důvodů proč do následné analýzy nejsou zahrnuty i klasické prostředky, jako je např. papír a tužka. U elektronických organizérů se využívá tří základních přístupů, a to online aplikace, aplikace se synchronizací a lokální aplikace bez sdílení dat.

2.1 Obecné rozdělení přístupů

2.1.1 Online aplikace

U tohoto přístupou jsou data uložena na serveru a k těmto datům se dá přistupovat z celé škály zařízení (smartphony, PDA, MDA, notebooky, apod.). Na žádost klienta jsou data ze serveru poslána na klienta, který s nimi poté pracuje. Klientská aplikace může data zaznamenávat a využívat i lokálně, ale pokud si přeje uživatel data změnit, je nutné data změnit na serveru a znovu stáhnout aktualizovaná data. Tento přístup také řeší možnost kolizí na klientovi. U online aplikací se využívá buď aplikačních klientů a appletů, nebo webového rozhraní.

Webové rozhraní má tu výhodu, že uživatel nemusí instalovat přídavnou aplikaci a k serveru se může připojit z jakéhokoli zařízení, jenž má připojení na internet a webový prohlížeč, naopak jeho nevýhodu představuje větší přenos dat. Aplikační servery mají nižší přenos dat, ale je nutné nainstalovat aplikaci, což se ne každému uživateli chce. Uživatel také nemusí mít vždy práva na instalaci aplikace (např. při návštěvě internetové kavárny, knihovny, nebo i ve škole či zaměstnání). Další nevýhodou je možnost zpomalení aplikace z důvodu čekání na data, která se teprve stahují ze serveru, a také možnost chyby „Single point of failure“. Nevýhodou online přístupu je nutnost připojení na internet. Ale v dnešní době vysokého internetového pokrytí a snižování cen za internet se tento problém zmenšuje a dle mého názoru se tento přístup bude stávat stále více využívaným.

2.1.2 Aplikace se synchronizací

Tento přístup je možný jak lokálně, tak i online. Lokální synchronizace se využívá hlavně v kombinaci mobilní zařízení – počítač, přičemž se využívá především technologií Blootooth a USB. Online synchronizace využívají servery, na které se nahrávají data z organizérů, tyto data se dají poté stáhnout na jiné zařízení. U tohoto přístupu jsou hlavní problémy se zaznamenáváním změn od minulé synchronizace a také možnost vzniku kolizí, u kterých je mnohdy nutný zásah uživatele. Naopak mezi výhody lze zařadit například offline přístup, díky čemuž je rychlost práce aplikace dána pouze rychlostí zařízení a není ovlivněna rychlostí internetového připojení. Při synchronizaci dat se serverem je přenos dat také menší než u online aplikací.

2.1.3 Lokální aplikace bez sdílení dat

Tyto aplikace jsou většinou velmi amatérské, nebo zastaralé. Jejich výhodou je vysoká rychlost (stejně jako u synchronizace je aplikace omezena pouze rychlostí zařízení). Tyto aplikace jsou ale velmi zřídka využívané, protože se data o úkolech, jenž jsou v nich zaznamenány, nedají nijak předat na další zařízení, což snižuje jejich využitelnost. Současně nabízejí aplikace se synchronizací stejné a někdy i lepší možnosti.

2.2 eGroupWare

Tento groupware je velmi silným nástrojem pro správu času, událostí a projektů. Obsahuje celou škálu možností, jako je například sdílení úkolů mezi více uživateli, správu zdrojů (kopírka, projektor, atd.), propojování různých zdrojů či úkolů k určitým projektům, správu souborů, wikipedii, milníky projektů, apod. Obvyklé možnosti u organizérů, jako je zobrazení událostí dne či úkolů, jsou samozřejmostí, stejně jako upozorňování na události a úlohy nebo rozdělování úkolů a událostí do kategorií. Všechny možnosti jsou přehledné a graficky velmi vyvedené. Tento groupware podporuje jak online přístup, tak možnost synchronizace s nějakým zařízením. Ze všech dále uvedených aplikací je tato nejpropracovanější, obsahuje největší podporu pro uživatele a navíc je zdarma.

2.3 Microsoft Exchange

Microsoft Exchange je groupware s celou škálou nadstandardních možností, jako je zvýšená ochrana dat, přehrávání hlasových zpráv zaslaných na server exchange, automatická rezervace zdrojů, ... Základní funkce, jako je správa úloh a úkolů, jsou opět samozřejmostí, stejně jako dělení do kategorií, upozorňování na události nebo úkoly pomocí zasílání e–mailů. K aplikaci se přistupuje pomocí webového prohlížeče, nebo přes aplikaci Microsoft Outlook. K aplikaci je sice možné přistupovat i např. z programu Evolution, ale nastavení propojení je poměrně obtížné. Tento groupware je bohužel komerční, a proto byl zařazen až za eGoupWare.

2.4 Rainlendar

Rainlendar je jednoduchá lokální aplikace, jenž je určena pro zaznamenávání a práci s událostmi a úkoly. Aplikaci lze využívat se třemi hlavními operačními systémy (Microsoft Windows, Linux, Mac OS). Railendar se dá online synchronizovat s aplikací Google Calendar.

Grafické rozhraní je umístěno na ploše, což je dle mého názoru uživatelsky velmi přívětivé. Úkoly a události takto zobrazené jsou přehledné a díky zobrazení na ploše i snadno dostupné. Aplikace podporuje dělení úkolů a událostí do kategorií a upozorňování na tyto úkoly nebo události. Jako výhodu u Rainlandaru považuji možnost nastavení cesty k zaznamenaným datům, díky čemuž je možné nastavit ve více operačních systénech stejnou cestu k jednomu souboru, čímž zařídíme, že pokud změníme data v jednom operačním systému, tak se nám tato změna projeví i v druhém. Rainlendar byl zařazen na třetí místo, protože je uživatelsky velmi přívětivý, propracovaný a snadno pochopitelný. Bohužel neobsahuje správu projektů, což poněkud snižuje jeho použitelnost.

2.5 Google Calendar

Google Calendar je online aplikace s možností synchronizace. Umožňuje jednoduchou správu úkolů a událostí, rozdělení do kategorií, sdílení kalendářů s dalšími uživateli, upozorňování na události a úkoly pomocí sms, nebo e – mailů, apod. Uživatelsky je aplikace velmi povedená, přehledná a snadno pochopitelná. Velmi se mi také líbí měsíční, týdenní, nebo i roční výpis událostí, což může uživatelům ulehčit plánování. Aplikace je uvedena za Rainlendar organizérem z důvedu nižší uživatelské přívětivosti.

2.6 Windows Live Calendar

Jedná se o online aplikaci s webovým rozhraním. Zatím se jedná o beta verzi s celou škálou méně důležitých nedostatků, jako je např. chybějící export do externího souboru, apod. Aplikace obsahuje výpisy a správu úkolů a událostí, vertikální zobrazení přehledu dne, rozdělování úkolů a událostí do kategorií, kterým se dají pro přehlednost nastavit rozdílné barvy. Také má zabudované upozorňování na události a úlohy, a to ať už pomocí e – mailů, tak i pomocí SMS. Po grafické stránce je aplikace propracovaná a přehledná. Tato aplikace obsahuje také týdenní a měsíční přehledy událostí. V podstatě je možné říci, že se jedná o obdobu Google Calendar s novým kabátem.

2.7 Evolution

Aplikace Evolution, která je automaticky nainstalovaná v operačním systému Linux Ubuntu, je po grafické stránce v některých ohledech nepropracovaná a „kostrbatá“, také po uživatelské stránce nepřívětivá a neintuitivní. Aplikace umožňuje tvorbu kategorií, výpis a tvorbu událostí a úkolů, zaznamenávání kontaktů, dále obsahuje e – mailového klienta a možnost synchronizace s Google Calendar. Aplikace je uvedena až nakonci seznamu aplikací, protože uživatelské prostředí je dle mého názoru nejméně přehledné.

Kapitola 3

Architektura

Tato kapitola popisuje základní architekturu naší aplikace, dále je zde obecně definována třívrstvá architektura a jsou zde podrobněji popsány i její jednotlivé vrstvy.

Výsledný program je vytvořen na platformě Java EE s využitím architektury klient – server, přesněji řečeno na základě třívrstvého modelu. Pracuje jako online aplikace založená na protokolu TCP/IP, díky čemuž je zajištěn spolehlivý přenos dat bez změny pořadí paketů. Program obsahuje i lokální cache pro plynulejší práci aplikace. Pokud je prováděna úprava v záznamech, tato změna se odešle na server, který provede aktualizaci dat a přepošle nová data na klienta. Činnosti vykonané nad serverem se provádí transakčně, tzn. musí se provést všechny části operace, jinak nedojde k žádnému zákroku.

Protože organizér pracuje přes interent, je nezbytné, aby byla dostatečně vyřešena otázka zabezpečení komunikace. Údaje, které jsou součástí organizéru a právě kvůli kterým jsou tyto aplikace důležité, jsou pro uživatele velmi cenné a často je považuje za důvěrné. Je velmi jednoduché mnoha způsoby zneužít uložená data, např. osobní údaje, ale také při sociálním inženýrství představují právě údaje, které tvoří záznamy organizéru, výhodu pro útočníka, který může jednoduše zneužít znalosti života uživatel. To, že by se údaje v organizéru mohly dostat do rukou nepovolané osoby, ale nemusí způsobit jen potenciální ohrožení přes počítač a internet, i „offline“ zloději uvítají informace o plánovaných schůzkách a dovolených a adresy, které k těmto informacím patří. Protože se budou databázový a aplikační server nacházet na stejném počítači nebo budou propojeny pomocí zabezpečené sítě, nemusíme spojení mezi aplikačním a databázovým serverem složitě zabezpečovat.

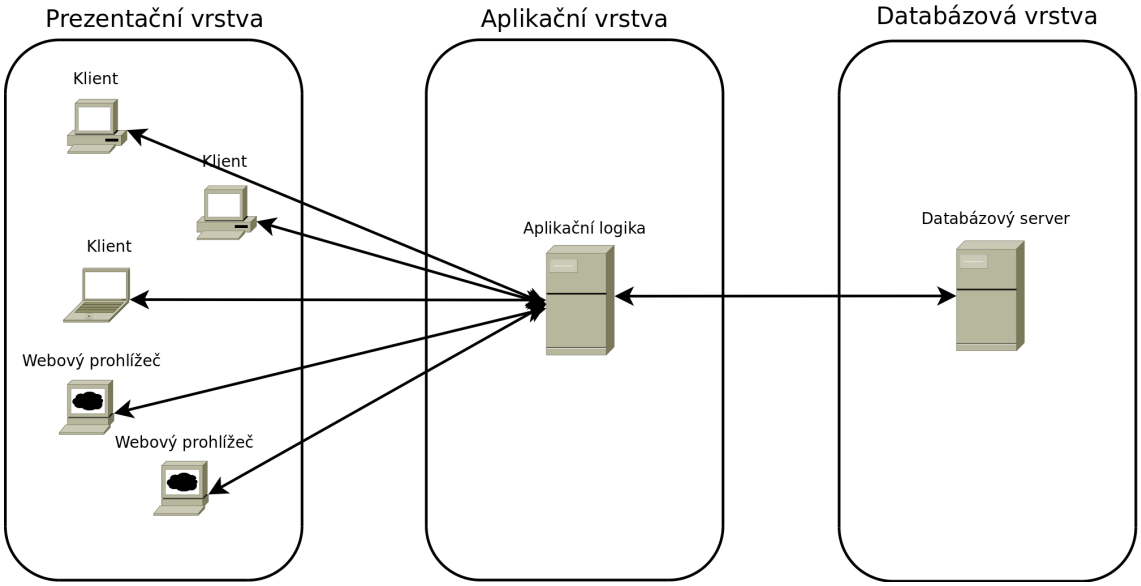
Třívrstvý model (viz obr. 3.1) se skládá z následujících částí:

- *Prezentační vrstva* – Obsahuje metody pro práci s grafickým prostředím, zpracováním informací od uživatele a reakcemi na události, jako je např. pohyb myši, kliknutí na položku, apod. Někdy je vytvořeno více prezentačních vrstev pro různé platformy a zařízení [9]. V našem případě nebylo nutné zavést větší množství těchto vrstev, protože celá aplikace pracuje na Java Virtual Machine (JVM), čímž je poskytnuto jednotné rozhraní, na kterém aplikace běží. Velmi důležitá je také volba graficko – uživatelského rozhraní (GUI), která může rozhodnout o úspěšnosti, nebo neúspěšnosti celé aplikace. Z tohoto důvodu je nutné přizpůsobit aplikaci uživatelům, vytvořit uživatelsky přívětivé a snadno pochopitelné prostředí, ve kterém bude uživatel rád pracovat každý den. Tato vrstva komunikuje pouze s aplikační vrstvou, nemůže se spojit přímo s databázovou vrstvou.

- *Aplikační vrstva* – Tato vrstva obsahuje aplikační logiku a odděluje prezentační vrstvu od databázové. Aplikační logika se stará o příjem požadavků od klientů, zpracování těchto požadavků a odeslání případné odpovědi klientovi. Aplikační logika se také stará o získávání dat z databázového serveru či o zapisování dat na něj.

- *Databázová vrstva* – Úkolem této vrstvy je ukládání a načítání perzistentních dat, tedy takových dat, která jsou trvale uložena a která je potřeba zachovat i po vypnutí serveru [13] . Tato vrstva komunikuje pouze s aplikační vrstvou a nespojuje se přímo s prezentační vrstvou.

Jednou z výhod třívrstvé architektury je, že aplikační server může být umístěn na jiném stroji, než se nachází databázový server, čímž je rozděleno vytížení stroje na dva samostatné stroje. Obvykle se také implementují dohodnutá rozhraní, pomocí nichž spolu vrstvy komunikují. Díky tomuto přístupu je poté snazší případný přechod na jiný databázový či aplikační server [9].



Obrázek 3.1: Třívrstvý model

Kapitola 4

Technologie

V následujících podkapitolách jsou podrobněji popsány technologie, které byly brány v úvahu při vytváření výsledného organizátoru. Nejdříve je pozornost věnována pro účel této práce zásadním technologiím, které obsahuje Java EE. Poté jsou rozepsány technologie, které jsou s Javou EE kompatibilní a které rozšiřují možnosti této platformy.

4.1 Java Platform, Enterprise Edition (Java EE)

Pro tvorbu Java EE Organizéru jsme využili platformu Java EE, což je platforma Java SE (Java Platform, Standard Edition) rozšířená o technologie pro vícevrstvé aplikace [8]. Java EE prošla od svého vzniku významnými změnami. Z počátku obsahovala širokou škálu možností, které ale byli obtížně využitelné. V současné době se projevuje snaha o zjednodušení a zvýšení použitelnosti nabízených možností. Významné je i úsilí o modularizaci, která by měla být obsažena již ve verzi Java EE 6 [7].

Výhody Javy EE [7]:

- Zjednodušení vývoje aplikací,
- velké množství knihoven a funkcionalit,
- aplikace je nezávislá na operačním systému, a to serveru i klienta,
- škála možných aplikačních serverů, stejně jako databázových serverů,
- podpora bezpečnosti,
- správa aplikací přes internet.

Z tohoto výčtu je patrné, že využití vývojové platformy Java EE je velmi výhodné, protože zjednodušuje práci programátora, který je díky popsané podpoře schopen vytvářet aplikace kvalitněji, bezpečněji a také rychleji, což je základní požadavek na vývojovou platformu. Programátor se také nemusí zabývat obšírným studiem technologií, protože Java EE tyto technologie „zaobaluje“ pro co nejsnažší použití. Pro více informací o platformě Java EE viz [4].

Základní přehled Java EE technologií:

- **Technologie webové vrstvy:**
  - Java Servlet: Java Servlet je technologie poskytující programátorům jednoduchý a konzistentní nástroj pro tvorbu a rozšiřování webových serverů [5]. Java servlety nemají žádné GUI (graficko – uživatelské rozhraní), ale mohou generovat HTML kód. Tyto servlery nejsou nijak omezeny platformou serveru, ale pouze tím, že vyžadují Java Virtual Machine (JVM) a java-enabled-server (server podporující Javu) [1]. Java Servlety mohou zpracovávat formuláře, zasílat e – maily, ... Zpracované formuláře mohou být naopak uloženy na databázový server. Této technologie využívají i následující dvě technologie.
  - JavaServer Pages (JSP): JSP v podstatě představují klasické HTML dokumenty obohacené o dynamické prvky uzavřené do speciálních tagů, které obsahují zdrojový kód napsaný v programovacím jazyku Java. Pokud přijde na server požadavek na JSP, je tento kód zpracován a pomocí servletu je vygenerován HTML kód, který je zaslán na klienta.
  - JavaServer Faces (JSF): Aplikace postavené na technologii JSF propojují Java servlety a JSP. Tvorba JSF je podobná jako tvorba GUI v Java Swing, kdy jsou dány určité komponenty, které jsou následně uspořádány na stránce. Při komunikaci s klientem je pak generován přes servelety kód (obvykle v HTML). JSF reaguje na události pomocí *listenerů*. Jednotlivé komponenty mohou být vnořeny do sebe (např. textové pole je vnořeno do formuláře).
  - JavaServer Pages Standard Tag Library (JSTL): Jedná se o rozšíření JSP, které přidává určité tagy pro práci se základními obecnými akcemi, akcemi pro přístup do relační databáze apod. [6]
- **Technologie webových služeb:**
  - The Java API for XML-based Web Services (JAX – WS): JAX-WS je určena pro tvorbu webových služeb a klientů založených na XML (eXtensible Markup Language).
  - The Java API for XML Binding (JAXB): JAXB slouží pro mapování tříd vytvořených v jazyce Java a převod těchto tříd do XML a zpět. JAXB je zvláště užitečný při složité a měnící se specifikaci [11].
  - The Streaming API for XML (StAX): Jedná se o API pro čtení a zápis XML dokumentů v programovacím jazyku Java [12].
  - The SOAP with Attachments API for Java (SAAJ): SAAJ umožňuje vytvářet a zpracovávat zprávy, které jsou v souladu se specifikací SOAP [10].
- **Enterprise Java Beany (EJB):** EJB komponenty reprezentují aplikační logiku. Na žádost klienta provádí určitou činnost (např. načítání dat z databáze, zpracování a zaslání těchto dat na klienta). V současné době se používají především dvě základní komponenty:
  - Session beans: Session beany existují ve dvou variantách, a to stateless a stateful. Stateless beany si neuchovávají stavy z předchozích volání, zatím co stateful ano. Většinou jsou doporučovány stateless beany, které mají nižší režii, ale v některých případech je výhodnější využít stateful.

  - Message – driven beans: Tyto beany využívají asynchronní komunikace, což někdy může být výhodné. Jsou využívány hlavně v případě žádosti na akci, která nevyžaduje odpověď od serveru.

- **Perzistence:** Java Persistance API (JPA) je framework, který umožňuje mapování dat do relační databáze a také zpětné zpřístupnění těchto dat. Tento framework se využívá hlavně u vícevrstvých aplikací, ale je možné ho využít i u lokálních aplikací. Framework využívá pro vytváření dotazů na subjekty uložené v relační databázi jazyk Java Perzistence Query Language (JPQL). Tento jazyk má podobnou syntaxi jako jazyk Structured Query Language (SQL), ale liší se v tom, že dotazy nevztahuje k tabulkám a jejich atributům v databázi, ale k entitním třídám a jejich atributům, které jsou do databáze na tyto tabulky namapovány. Třídy, které mají být mapovány se nazývají entitními třídami.

  Třída je označena jako entitní pomocí anotací, které nejen určují, že je třída entitní, ale také na jakou tabulku má tato třída být mapována, které atributy třídy budou odpovídat daným atributům tabulky, co bude primárním klíčem tabulky a případně jaký generátor bude využit pro generování nových primárních klíčů. Entitní třídy musí splňovat několik kritérií, jako je implicitní konstruktor, get a set metody pro každý atribut. Třída a stejně tak i její metody či perzistentní data nesmějí být deklarovány jako final a třída musí být serializovatelná. JPA také umožňuje pro entitní třídy vytvořit *Named Queries* (pojmenované dotazy) v jazyce JPQL.

- **Zabezpečení:** Java EE obsahuje několik mechanizmů pro zabezpečení, a to jak spojení mezi serverem a klientem, tak samotného připojení k serveru. Java EE obsahuje také knihovny pro tvorbu klíčů a certifikátů. Součástí Java EE API je i framework Java Authentication and Authorization Service (JAAS), který slouží pro autentizaci a autorizaci uživatelů. Přenos z prezentační vrstvy na aplikační bývá zabezpečen pomocí Virtual private network (VPN), nebo Secure Shell (SSH).

4.2 Java Swing API

Java Swing API (dále jen Swing) je knihovna tříd a komponent pro tvorbu graficko – uživatelského rozhraní (GUI). Swing je rozšířením knihoven AWT, které sloužily pro tvorbu GUI dříve. AWT využívalo komponenty systému, což mu propůjčovalo stejný vzhled, jako měl zbytek systému, naproti tomu Swing využívá svých komponent, a proto může vypadat i jinak než ostatní aplikace. U aplikací napsaných s využitím Swing je možné nastavit, aby komponenty využily programátorem nastaveného vzhledu, vzhledu operačního systému, nebo vzhledu, který byl nastaven pro všechny Java aplikace. AWT má oproti Swingu výhodu v rychlejším chodu aplikace, ale výhodou Swingu je čistší kód, větší flexibilita, vyšší podpora tvorby aplikací pro nevidomé, podpora práce s 2D grafikou a obsahuje i složitější komponenty, jako jsou např. `JTable` nebo `JColorChooser`.

Aplikace vytvářené s využitím knihovny Swing využívají základního principu tvorby GUI, a to řízení programu událostmi. Každá komponenta může vyvolat širokou škálu možných událostí v závislosti na druhu komponenty. Události jsou většinou generovány pomocí zásahu uživatele (např. pohyb myši, stisknutí tlačítka, apod.). Každá komponenta může mít přiřazen jeden až n *listenerů*[1] událostí. Existuje několik druhů událostí např.

[1] posluchačů

KeyEvent, MouseEvent, ActionEvent, apod. K těmto výjimkám existuje několik druhů rozhraní *listenerů*, jako je `ActionListener`, `MouseListener`, ... Pro přidání *listeneru* je nutné daný *listener* registrovat k určité komponentě (`addMouseListener()`) a také implementovat všechny metody daného rozhraní. V případě, že by takto *listener* nebyl nepřidán, nebude se daná výjimka „odchytávat“ a zpracovávat. Metody daného rozhraní se nazývají *handlary* (obslužné rutiny), tyto metody zpracovávají danou výjimku. Díky tomuto principu se dá vytvořit uživatelsky velmi přívětivá a interaktivní aplikace, která může teoreticky reagovat na každý pohyb myši nebo zmáčknutí klávesy.

Další důležitou částí Swingu jsou právě komponenty a kontejnery. Základními kontejnery jsou `JWindow`, `JDialog` a `JFrame`. Většina aplikací, které jsou tvořeny s pomocí Swing knihoven, se skládá z jednoho `JFramu` a několika `JDialogů`. Pro `JFrame` a `JDialog` je možné nastavit titulek, velikost, umístění, apod. K těmto kontejnerům přistupujeme pomocí kontejneru `ContentPane`, který je automaticky přiřazěn k těmto kontejnerům. Swing obsahuje i další kontejnery např. `JPanel`, `JScrollPanel`, `JInternalFrame`, apod. `JInternalFrame` dále například umožňuje tvorbu vnitřního framu uvnitř hlavního framu aplikace. Swing obsahuje základní komponenty, jako jsou `JButton`, `JMenu`, `JTextArea`, `JPasswordField`, atd. Tyto komponenty vlastně tvoří základní vstup i výstup aplikací. Další kategorií jsou neměnitelné informační komponenty, což jsou `JToolTipText`, `JLabel` a `JProgressBar`. Další důležitou součástí Swingu jsou *Layout managery*. Tito managery mohou být přiřazeny k určitému kontejneru. Existuje několik druhů layout managerů, např. `FlowLayout`, `GridLayout`, `BoxLayout`, `GridBagLayout`, atd. Při přiřazení Layout manageru kontejneru se poté tento manager stará o grafické rozmístění komponent v tomto kontejneru. Například `FlowLayout` umístí komponenty postupně zleva doprava na jeden řádek, při konci řádku přejde na nový řádek a pokračuje dál.

4.3 Glassfish

Glassfish [3] je open source aplikační server od firmy Sun Microsystems pro platformu Java EE. Aktuální dokončená verze je v2, a proto jsme se pro ni rozhodli. Nová verze v3 je zatím v distribuci pouze v prelude verzi, a proto je pro nás nepoužitelná. Tato nová verze bude podporovat Javu EE 6, která zatím také není dokončena. Tento aplikační server byl pro nás jasným „favoritem“, protože je zdarma a pochází od firmy Sun Microsystems, která je také tvůrcem Javy EE, díky čemuž je velmi pravděpodobné, že budou spolehlivě spolupracovat. K tomuto aplikačnímu serveru je několik alternativ, jako je například Oracle Application Server nebo RedHat Jboss, ale, jak jsem se již zmínil, tyto aplikační servery jsou na rozdíl od aplikačního serveru Glassfishe komerční, a proto jsou pro nás nevyhovující.

4.4 Firebird

Firebird [2] je multiplatformní relační databázový server, který je spravovaný neziskovou organizací Firefird Foundation. Tento databázový server má dobrou podporu od vývojových nástrojů, a to ať uz se jedná o Javu EE, nebo .NET. Nový kód je vydáván pod licencí Initial Developer's Public License, která umožňuje využívat Firebird zdarma. Další výhoda Firebirdu spočívá v plné podpoře ACID[2] transakcí, v poměrně malých nákladech na systémové požadavky, snadném zálohování, apod. Firebird může být v nejbližší budoucnosti využíván v následujících čtyřech režimech:

[2] Atomicity – Consistency – Isolation – Durability

- **Firebird Embedded** – Tato varianta obsahuje některé prvky neslučitelné s naším projektem. Jedná se například o nutnost mít aplikační server a databázový server na jednom zařízení.
- **Firebird Superserver** – Vhodné využít při větším počtu spojení. Každý proces má více paralelních vláken. Klienti sdílí jednu společnou cache. Při chybě se přeruší všechny spojení.
- **Firebird Classic** – Výhodné aplikovat při malém počtu připojení. Každý klient má samostatný proces a cache. Při přerušení nebo chybě při procesu je ovlivněn pouze klient, kterému daný proces náleží.
- **Firebird Superclassic** – Bude podporovaná až od Firebirdu v2.5, která ještě nevyšla a bude slučovat vlastnosti režimů Firebird Superserver a Firebird Classic.

Z tohoto přehledu je patrné, že pro naše účely jsou použitelné režimy Firebird Classic a Firebird Superserver. Mezi těmito variantami se můžeme rozhodnout podle toho, kolik očekáváme připojených klientů. Pokud jich bude málo, pak použijeme variantu Firebird Classic, a pokud naopak budeme očekávat více jak dvacet klientů, které budou využívat operačního systému Microsoft Windows, pak zvolíme variantu Firebird Superserver.

Kapitola 5

Návrh a implementace

Tato kapitola se zabývá návrhem a implementací Java EE Organizéru a zaměřuje se na modul kalendář. Je zde popsán návrh a implementace aplikační logiky pro události a kontakty, a také návrh a implementace části prezentační vrstvy, která mi byla přidělena. Také zde budou blíže popsány alternativy k použitým algoritmům a postupům. V rámci implementace byla práce v týmu rozdělena následovně: Pavel Palát vytvářel databázovou vrstvu, entitní třídy, dotazy do databáze, rozhraní do aplikační logiky a aplikační logiku pro správu uživatelů. Petr Černý měl za úkol implementovat aplikační logiku pro moduly úkoly a kontakty. Implementoval také hlavní okno klienta potřebné třídy pro moduly kontakty a úkoly. Mým úkolem byla implementace aplikační logiky pro modul kalendář a modul kategorie a také implementace modulu kalendář a modulu kategorie v prezentační vrstvě. Dále jsem implementoval přihlašovací okno a několik dalších tříd, které jsou podrobněji zmíněny v podkapitole 5.2. Celá aplikace je naprogramována pomocí vývojového prostředí NetBeans, které nám usnadnilo rozmisťování komponent v kontejnerech a práci jako takovou vůbec.

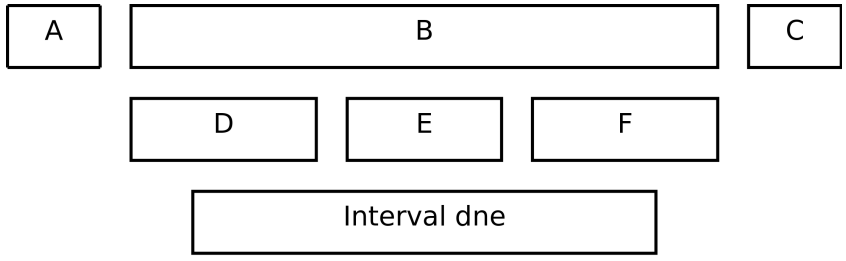
5.1 Aplikační vrstva

5.1.1 Návrh aplikační vrstvy

V aplikační vrstvě bylo nutné vytvořit EJB komponenty, které se starají o zpracování žádostí od klienta vztahujících se ke kalendářovým událostem a kategoriím. Tyto komponenty také získávají nebo ukládají perzistentní data k těmto událostem nebo kategoriím. Aby komponenty mohly správně fungovat, musí být propojeny s komponentou, která spravuje informace o uživatelích. Toto propojení je nezbytné kvůli ověřování a identifikaci uživatele. Těchto informací se poté využívá při práci s perzistentními daty (např. žádá se pouze o data, která přísluší danému uživateli). U EJB komponenty starající se o události bylo v metodě, která vrací události k určitému datu, nutné vytvořit podmínku, která ověřuje, zda interval události spadá do intervalu hledaného dne. Obrázek 5.1 ukazuje, jaké jsou možnosti intervalů událostí. Z tohoto obrázku je patrné, že existují dva případy, ve kterých interval události nespadá do intervalu dne. Ty se poznají tak, že začátek intervalu události je vyšší než konec intervalu dne nebo že konec intervalu události je nižší než začátek intervalu dne.

5.1.2 Implementace aplikační vrstvy

V aplikační vrstvě bylo potřeba implementovat dvě EJB komponenty `CalendarManagerBean` a `CategoryManagerBean`. Tyto komponenty jsou téměř totožné, a proto popíši pouze kom-



A, B, C, D, E, F - Intervali událostí
A,C - Události, které nespadají do dne
B, D, E, F - Události, které spadají do dne

Obrázek 5.1: Možnosti intervalů

ponentu `CalendarManagerBean`, která obsahuje jednu rozsáhlejší a komplikovanější metodu. `CalendarManagerBean` implementuje rozhraní `CalendarManager` vytvořené Pavlem Palátem. Pomocí anotace `@Stateless` bylo určeno, že se jedná o EJB komponentu, která neuchovává stavy mezi voláními. Dalším nutným krokem je vytvoření referenci na EJB komponentu `UserManagerLocal`, což se provede pomocí anotace `@EJB`. Pomocí anotace `@PersistenceContext` bylo nezbytné určit, že budeme využívat perzistentních dat, která jsou přístupná přes `EntityManager`. Tento `EntityManager` je něco jako cache objektů, které jsou jednou za určený čas obnoveny z databáze. Třída samotná obsahuje několik základních metod (např. metody pro následující operace: přidání, odebrání a úpravu úkolů).

Další důležitou a složitou metodou je `getDayItems`. Této metodě jsou předány dvě informace, a to informace o uživateli a informace datu (D), ke kterému se události mají vybrat. Metoda vrací utříděný seznam kalendářových položek, které obsahují informace o události, datu a času, kdy daná událost začíná.

Následuje posloupnost kroků, které metoda provádí:

1. Ověření, zda daný uživatel existuje.
2. Vznesení dotazu na perzistentní data – dotaz vrací seznam událostí (U), které by mohly nastat.
3. Postupné procházení U.
   (a) Ověření, zda má daná událost stejné datum jako je D. V případě, že se daná data shodují, je událost přidána do nového seznamu (S), který již bude obsahovat i startovní datum události.
   (b) Pokud se jedná o opakující se událost, je k datu startu události přičtena hodnota odpovídající periodě opakování. Poté se ověří, zda interval události spadá do intervalu dne D, pokud ano, pak událost přidáme do S i s novým datem startu. Toto přičítání period je opakováno, dokud se nedojde k datu konce události.
4. Nakonec seřadíme události a vrátíme S.

5.2 Prezentační vrstva

5.2.1 Návrh prezentační vrstvy

`Main`, `LoginFrame`

Třída `Main` se spouští automaticky při startu aplikačního klienta. Třída si vytváří reference na EJB komponenty, díky čemuž poté stačí volat tyto komponenty přes třídu `Main` a není nutné vytvářet tyto reference ve všech třídách, které s těmito komponentami chtějí pracovat. Obecně preferovaný přístup je sice obrácený, ale my jsme tuto metodu zvolili kvůli zjednodušení práce a zvýšení přehlednosti. Dále třída obsahuje jako statický atribut informace o uživateli, který je přihlášený. Základním úkolem třídy `Main` je zobrazení okna, které slouží pro přihlášení. V tomto okně uživatel zadává svůj login a heslo. Tyto informace se ověří a poté jsou do atributu třídy `Main` nastaveny informace o uživateli. Při této operaci se uzavře okno pro přihlášení a otevře se hlavní okno klienta. Okno pro přihlášení obsahuje i status bar, na který jsou případně vypsána chybová hlášení. Další možností, jak provádět přihlášení uživatele, je, že se spustí hlavní okno klienta, přičemž je zakázané zobrazení vnitřních framů okna a jediné dvě povolené položky menu způsobí ukončení klienta, a nebo zobrazení okna pro přihlášení. Po úspěšném přihlášení jsou poté zpřístupněny všechny položky menu a je umožněno zobrazení vnitřní framů.

`CzechCalendar`

Tuto třídu jsme původně vytvořili kvůli modulu kalendář, který je velmi složitý, a práce s gregorianským kalendářem, který obsahuje Java EE API, by byla obtížná. Tato třída obsahuje několik konstruktorů, díky čemuž jsme schopni vytvořit nový kalendář skoro pro jakýkoli parametr předávaný v konstruktoru (např. samostatné položky určující datum, gregoriánský kalendář, apod.). `CzechCalendar` uschovává ve svých atributech jak základní informace o datu a času, tak i rozšířené informace, které se hojně využívají právě v modulu kalendář. Jedná se například o informace o prvním dnu v měsíci (jedná-li se o pondělí, úterý, atd.). Součástí třídy je také pole řetězců obsahující české názvy měsíců a české zkratky dnů. Pole se zkratkami názvů dnů je využito v komponentě pro zobrazení dnů v měsíci. Práce s touto třídou je intuitivní a snadná. Také bylo vhodné vytvořit výjimku, která nastává při nastavení chybného dne, měsíce nebo roku. Díky vytvoření této výjimky je poté možné přímo určit, že tato chyba vznikla z důvodu zadání nesprávného data.

`PopupMenu`

Důležitou součástí GUI je *popup menu*. Jedná se o „vyskakující“ nabídku, která je nejčastěji implementována tak, aby se zobrazila při kliknutí pravého tlačítka myši. Jelikož využíváme *popup menu* při několika rozdílných příležitostech, vytvořili jsme si samostatnou třídu rozšiřující komponentu `JPopupMenu`. Alternativou k *popup menu* je přidání položek z popup to hlavního menu klienta, ale tato metoda je uživatelsky méně přívětivá. Další možností je kombinace těchto dvou přístupů. Tuto možnost jsme nezvolili, protože tuto metodu považujeme za méně přehlednou.

Modul kalendář – zobrazení měsíce

Tato část modulu kalendář se zabývá zobrazením výpisu dnů v měsíci. Tento výpis je standardně zobrazen v levém dolním rohu hlavního okna klienta. Výpis bylo nutné samo-

statně doimplementovat, protože komponenty se stejnou funkčností jsou bohužel komerční. Tato komponenta je rozdělena do tří tříd. Instance první třídy zastupují jednotlivé buňky představující dny v měsíci, zkratky českých názvů dní nebo dny, které do měsíce nepatří. Druhá třída obsahuje rozložení těchto buněk a metody pro změnu tohoto výpisu (např. při změně roku ve třetí třídě je nutné změnit i rok v této třídě, aby se správně změnilo rozmístění buněk). Třetí třída poté přidává do kontejneru další komponenty, např. *spinnery* pro změnu roku a měsíce. Ve třídě, která se stará o rozložení buněk, je využito správce rozložení `GridLayout`, který využívá tabulkového rozložení. Velikost tabulky je sedm sloupců a sedm řádků. Tato velikost je spočítána následovně: týden obsahuje sedm dní (sedem sloupců) a do měsíce může spadat maximálně šest týdnů (šest řádků). K počtu týdnů je nutné přičíst ještě jeden řádek, který bude obsahovat zkratky dnů.

Modul kalendář – zobrazení událostí dne

Zbývající část modulu kalendář se zabývá zobrazením událostí vybraného dne. Výběr dne je proveden pomocí výpisu dnů v měsíci (viz 5.2.1). Výpis událostí dne je standardně zobrazen v pravé části hlavního okna. Pro větší uživatelskou přívětivost bylo rozhodnuto, že výpis bude obsahovat nejen zobrazení událostí, ale i mřížku znázorňující hodiny a půlhodiny daného dne. Jednou z možností, jak toho dosáhnout, bylo vytvořit výpis jako tabulku a nastavovat postupně pro každou buňku takový rámeček, aby v událostech rámeček chyběl, ale jinak byl zobrazen. Druhou možností bylo využít kontejneru `JLayeredPain`. Tento kontejner funguje na principu vrstev, které jsou zobrazeny přes sebe. Tato metoda nám přišla efektivnější, a proto jsme jí využily.

Jelikož jsme museli vyřešit proměnlivé rozložení komponent, zvolili jsme `GridBagLayout`. Tento layout manager je nejsložitější, ale poskytuje nejvíce možností. Manager využívá mřížky, ale jednotlivé komponenty mohou obsadit i více řádků nebo sloupců. Jednotlivým komponentám jde také individuálně nastavit umístění, roztažitelnost, velikost, apod. Tento manager spolupracuje s třídou `GridBagConstraints`, jejíž instance obsahují informace o jednotlivých komponentách, které jsou rozmisťovány. Jedná se například o informace o umístění, velikosti, apod. Pro vytváření mřížky je využita hlubší vrstva než pro výpis událostí, díky čemuž jsou poté komponenty představující událost vykresleny nad mřížku. Komponenty tvořící mřížku jsou umisťovány tak, že nejdříve jsou umístěny komponenty vyjadřující půlhodiny a poté jsou na levou stravu vytvořeny komponenty, které obsahují hodnotu vyjadřující hodiny dne. Tyto komponenty pro hodiny jsou roztaženy přes dva řádky.

Také byla vytvořena třída, které se stará o práci s tímto výpisem. Tato třída je jakýmsi „správcem“ této části. Některé metody tohoto „správce“ poté zprostředkovaně přes třídu `Main` volají metody z EJB komponenty v aplikační vrstvě. Při vznesení požadavku na výpis událostí dne je zavolána metoda, která získá tyto informace a zavolá metodu pro obnovení obsahu výpisu událostí, přičemž jako atribut přidá informace o událostech v daný den. Metoda poté musí ze seznamu událostí postupně projít všechny tyto události a vypočítat umístění těchto událostí. Tento výpočet není nejtriviálnější z důvodu nutného počítání překrývání událostí a řešení těchto kolizí. Pokud interval události spadá do intervalu dne částí kratší než je půl hodiny, je interval prodloužen. Při výpočtu umístění jsou události seřazeny podle startovního času události a poté procházeny. Pro každou událost je spočítán počet kolizí a maximální počet je zaznamenáván. Tato hodnota následně udává maximální počet událostí zobrazených vedle sebe. V dalším kroku je postupně procházeno pole s událostmi a je ověřováno, zda na daném místě není už jiná událost. Pokud není, pak je na toto místo

událost přidána.

Třída, jejíž instance jsou komponenty představující událost, je velmi jednoduchá. Třída obsahuje atributy, které vyjadřují barvu, informace o události a pozici. Třída také obsahuje *listener*, který reaguje na kliknutí myši. Při kliknutí levého tlačítka je zobrazeno dialogové okno obsahující formulář s předvyplněnými údaji o dané události. Při změně těchto údajů jsou tyto nové údaje přeposlány až do aplikační logiky, která se postará i o změnu těchto údajů v relační databázi. Při kliknutí pravého tlačítka myši na událost je zobrazeno *popup menu*, které poskytne možnosti pro vytvoření nové události, změnu události, na kterou bylo „kliknuto“ a nebo odebrání této události.

Při požadavku na změnu nebo přidání událostí je zobrazeno nové dialogové okno, které obsahuje několik základních komponent, jako jsou textová pole, tlačítka, nebo kombo boxy. Součástí okna je i stavová lišta, na kterou jsou zobrazována chybová hlášení (např. „Musíte zadat název události!“). Okno obsahuje dva kombo boxy, a to pro výběr kategotrie a typ opakování události. Při potvrzení zadaných údajů jsou tato data ověřena a případně doplněna (při nezadání času je automaticky nastavena půlnoc). Pokud jsou všechny údaje správně vyplněny, je zaslán požadavek na aplikační logiku, která tuto událost přidá, popř. upraví v databázi. Okno je poté automaticky uzavřeno.

Modul kategorie

Tento modul se zabývá správou kategorií. Přístup ke správci kategorií je přes nabídku *Nástroje*, která se nachází v hlavním menu klienta. Po zvolení této položky je zobrazeno dialogové okno, které obsahuje seznam již vytvořených položek, tlačítek pro přidání, upravení a odebrání kategorie a také status bar, na který jsou případně vypisována chybová hlášení. Pro úpravu a odebrání určité kategorie je nutné tuto kategorii označit. Při stisknutí tlačítka pro přidání nebo úpravu kategorie je zobrazeno dialogové okno. V případě, že se jedná o úpravu kategorie, má okno již předvyplněné informace o kategorii (nastavenou barvu a vyplněné jméno). I toto dialogové okno obsahuje status bar, na který jsou případně vypisována chybová hlášení. Při kliknutí na tlačítko potvrdit je vznesen požadavek na přidání, popř. úpravu kategorie. Při úspěšné operaci je toto dialogové okno uzavřeno a při neúspěšné je na status bar vypsáno chybové hlášení. Výběr barvy je proveden prostřednictvím komponenty pro výběr barvy, která je přímo obsažena ve Swing API.

5.2.2 Implementace prezentační vrstvy

`Main`

Třída si vytváří reference na EJB komponenty pomocí anotace `@EJB`. Třída obsahuje jeden atribut, a to `RemotedUser`, který zahrnuje informace o uživateli, který je přihlášený (samozřejmě tyto informace obsahuje až po přihlášení). Třída obsahuje také dvě metody, a to `main` a `setUser`. Metoda `main` je spuštěna automaticky při startu aplikačního klienta a obsahuje nastavení grafického vzhledu a zobrazení framu `LoginFrame`.

`LoginFrame`

Tato třída rozšiřuje kontejner `JFrame`. Do tohoto kontejneru jsou poté pomocí nástroje NetBeans rozmístěny základní komponenty, které slouží pro zadání uživatelova loginu a hesla, zobrazení registračního formuláře, odeslání dat na aplikační logiku, uzavření okna a vypsání chybových hlášek. Při označení komponenty `JTextField` a stisku klávesy *enter*

je automaticky generována akce, jejíž součástí je vytvoření nového vlákna, které se stará o ověření zadaných údajů. Při správném vyplnění je zavolána metoda z třídy `Main`, jejímž parametrem jsou informace o uživateli (zjištěného při ověřování údajů), a okno je uzavřeno. Metoda nastaví parametr třídy `Main` na tohoto uživatele a zobrazí hlavní okno aplikačního klienta. Při chybném vyplnění údajů nebo při chybě při připojování k aplikačnímu serveru je na status bar vypsáno chybové hlášení.

`CzechCalendar`

Tato třída slouží pro práci s kalendářovými daty, pro kterou se obvykle využívá třídy `GregorianCalendar`. Tuto třídu sice vnitřně `CzechCalendar` také využívá, ale atributy jsou přepočítávány na pro nás obvyklejší hodnoty (např. rozsah 0-11 pro měsíce je přepočítáván na rozsah 1-12). Asi nejsložitější částí této třídy jsou konstruktory, které obsahují přepočty atributů z třídy `GregorianCalendar`.

Rozdělení konstruktorů podle atributů:

- Bez parametru,
- GregorianCalendar,
- CzechCalendar,
- long Date,
- int year, int day, int month.

Z předcházejícího přehledu je patrné, že byly vytvořeny konstruktory pro nejčastěji využívané možnosti. Třída obsahuje několik jednoduchých metod pro práci s daty, jako je například nastavení dne, měsíce, roku. Součástí třídy je i několik metod, které mají speciální využití. Jedná se například o metodu vracející informaci o dni v týdnu, kterým začíná daný měsíc. Tato metoda je využita v třídě `MonthPanel` (blíže o třídě viz 5.2.2).

`PopupMenu`

Třída obsahuje několik vnitřních tříd, které představují jednotlivé položky nabídky. Jelikož se tato třída využívá při různých příležitostech, bylo nutné vytvořit pro každou příležitost jinou inicializační metodu. Inicializace také probíhá rozdílně podle atributu, který je metodě předáván, díky tomu máme rozdílné okno při kliknutí na událost a jiné při kliknutí do místa, kde žádná událost není vyznačená. Vnitřní třídy obsahují konstruktor, který obsahuje volání metody `super()`. Atribut metody je ve formátu (`String`) a určuje název položky v *popup menu*. Při kliknutí na položku je volána metoda `actionPerformed()`, která provede požadovanou činnost. Třídu využíváme tak, že při určité události vytvoříme instanci této třídy, inicializujeme menu (v atributu předáváme informaci o tom, zda bylo kliknuto na událost), zjistíme bod, na kterém se má *popup menu* zobrazit, a poté tam toto menu zobrazíme.

Modul kalendář – zobrazení měsíce

Tato část modulu kalendář se skládá ze tří tříd a to `MonthIFrame`, `MonthPanel` a `MonthPane`.

Třída `MonthPane` reprezentuje jednu buňku (den) z tabulky dnů v měsíci. Třída obsahuje atributy, které určují barvu, hodnotu, popisek (využíváno u buněk, které reprezentují dny v týdnu, např. Po, Út, St, atd.) a barvu rámečku. Atribut, který je označený jako hodnota, označuje, jestli se jedná o den, popisek dne (např. Po), nebo o buňku, která představuje den, jenž nepatří do daného měsíce. Při inicializaci je také k buňkám představujícím den přidán *listener*, který reaguje na nastavení kurzoru myši na buňku. Při této události je provedena automatická změna barvy rámečku. Při opuštění buňky je barva rámečku nastavena zpět na černou.

Druhá třída v této části je `MonthPanel`. Tato třída rozšiřuje kontejner `JPanel` a pro rozmístění komponent používá `GridLayout` manažer. Tento manažer využívá tabulkové struktury. Při inicializaci panelu jsou jednotlivé buňky rozmístěny na panelu. Při rozmisťování buněk je využita metoda z třídy `CzechCalendar`, která vrací pole popisků pro dny v týdnu. Na začátku jsou vytvořeny buňky obsahující popisky ke dnům v týdnu a poté jsou vytvořeny ostatní buňky. Tyto buňky jsou nastaveny jako buňky, které nespadají do měsíce. Inicializace buněk probíhá až v metodě `refresh()`, která přepočítává a přenastavuje obsahy buněk, které mohou zastupovat den. Tato metoda je také volána při změně měsíce nebo roku, které se mají zobrazovat. Třída obsahuje také *listener*, který reaguje na kliknutí myši. Při kliknutí myši na určitou buňku je zavolána metoda, která náleží panelu pro výpis událostí dne, a jako parametr této metody je předáno datum odpovídající buňce, na kterou bylo „kliknuto“. Pokud je „kliknuto“ na buňku, která nereprezentuje žádné datum, není na tuto událost nijak reagováno.

Třída `MonthIFrame` rozšiřuje kontejner `JInternalFrame`. Kontejner obsahuje kromě panelu s dny v měsíci ještě čtyři další komponenty, a to dvě komponenty typu `JSpiner` a dvě komponenty typu `JLabel`. Komponenty `JSpiner` zastupují měsíc a rok. Třída obsahuje také *listener*, který reaguje na na změnu stavu *spinerů*. Při změně stavu je zjištěno, jestli se jedná o listener pro rok, nebo pro měsíc a je zavolána metoda `refresh()`, které je jako parametr předána informace, zda změna proběhla na spineru, který představuje rok. Metoda poté podle tohoto parametru přenastaví datum. Poté je volána metoda z třídy `MonthPanel`, která provede přepočítání a znovunastavení buněk představujících dny.

Modul kalendář – zobrazení událostí dne

Zbývající část modulu kalendář se skládá ze čtyř tříd, a to `EventManager`, `EventPanel`, `EventPane` a `EventForm`.

Třídu `EventManager` je možné s trochou nadsázky označit jako „správce“ událostí. Stará se o přidávání, odebírání a úpravu událostí a také o obnovu výpisu událostí. Z tohoto důvodu obsahuje metody, které přes třídu `Main` přistupují k aplikační logice pro události.

Třída `EventPanel` rozšiřuje kontejner `JLayeredPane`, který je schopný obsahovat komponenty na několika vrstvách. Díky této možnosti jsme byli schopni vykreslovat komponenty zastupující události nad mřížku. O rozmístění komponent se stará `GridBagLayout`. Tento manager spolupracuje s třídou `GridBagConstraints`, jejíž instance obsahují informace o jednotlivém rozmístění jednotlivých komponent. Mřížka dne je vytvořena pomocí uspořádaných komponent typu `JPanel`, které mají nastavený rámeček. Rámeček obsahuje čtyřicet osm komponent typu `JPanel` vyjadřujících půl hodiny a dvacet čtyři komponent typu `JPanel` vyjadřujících hodiny.

Jedna z metod třídy počítá rozmístění událostí dne. Na začátku tohoto procesu jsou odstraněny všechny komponenty vyjadřující úkoly z kontejneru a poté jsou vytvořena tři samostatná pole, která jsou postupně plněna informacemi o souřadnicích x a začátcích a

koncích událostí. Dalšími kroky metody jsou zaokrouhlení časů na půlhodiny a oříznutí intervalu události tak, aby nepřesahoval interval určeného den. Následuje seřazení událostí tak, aby byly počáteční časy událostí vzestupné. Pro toto seřazení byl využit algoritmus *Bubble sort*. Poté je vypočítán maximální počet kolizí, což je v podstatě maximální počet událostí zobrazených vedle sebe. Posledním krokem metody je výpočet umístění a velikosti události. Výpočet těchto hodnot je již triviální, pouze výpočet hodnoty souřadnice x je složitější a provádí se tak, že se pro každou událost projdou veškeré předcházející události a ověří se, jestli se nepřekrývají. Tyto informace jsou poté využity v druhém cyklu, který již „ví“, která místa jsou využita, a přiřadí události první volné místo. Následuje zavolání metody, která událost umístí na správné místo.

Třída obsahuje *listener*, který reaguje na kliknutí pravého tlačítka myši. Pokud je této možnosti využito, je vytvořeno *popup menu*, které obsahuje pouze jednu položku, po jejímž vybrání je zobrazeno dialogové okno pro přidání události.

Třída `EventPane` rozšiřuje kontejner `JPanel`. Instance této třídy reprezentují při výpisu událostí právě jednu událost. Atributy třídy nesou informace o události, barvě a pozici dané komponenty. Třída obsahuje také *listener*, který reaguje na kliknutí myši. V případě kliknutí levým tlačítkem myši na událost je zavolána metoda pro zobrazení okna s informacemi o této události. Konstruktoru tohoto okna je předán atribut, který obsahuje tuto událost. Tím je metodě řečeno, že se jedná o úpravu a ne o přidání události.

Posledné třídou, která patří do této části modulu kalendář, je třída `EventForm`. Tato třída rozšiřuje kontejner `JDialog`. Výsledné okno je využito jak pro vytvoření nové události, tak pro úpravu již existující události. Výběr varianty je pomocí atributu konstruktoru. Kontejner obsahuje několik komponent `JLabel` a `JTextField`, které reprezentují určité informace. Jsou zde také použity komponenty `JComboBox`, které se starají o zobrazení nabídky kategorií a nabídky opakování a o výběr z nich.

Modul kategorie

Poslední modul, který jsem měl za úkol vytvořit, se skládá ze tří tříd. Tyto třídy nesou názvy `CategoryManager`, `CategoryManagerForm` a `CategoryForm`.

Třída `CategoryManager` je opět jakýmsi „zprostředkovatelem“ kategorií. Metody této třídy využívají metod třídy `Main`, která tyto požadavky přeposílá dál na aplikační logiku. Metody třídy `CategoryManager` se starají například o odebírání, přidávání nebo úpravu kategorií.

Další třídou je `CategoryManagerForm`, který rozšiřuje kontejner `JDialog`. Po zobrazení obsahuje výsledné okno seznam existujících kategorií, které jsou zobrazeny formou komponenty `JList`. Okno dále obsahuje status bar, tlačítka pro práci s kategoriemi a tlačítko pro uzavření okna. Při kliknutí na tlačítko pro vytvoření nové kategorie, nebo pro úpravu již existující kategorie je otevřeno nové okno.

Poslední třídou tohoto modulu je `CategoryForm`. Kontejner této třídy obsahuje dvě tlačítka, dále jedno textové pole pro název kategorie, status bar vytvořený pomocí `JLabel` a komponentu `JColorChooser`, která se stará o výběr barvy.

Kapitola 6

Závěr

Tato práce vznikla jako zpráva k aplikaci Java EE Organizér. Návrh této aplikace prošel několika změnami, jako například rozhodnutí vytvořit aplikačního klienta místo webového rozhraní. K této změně došlo po skončení fáze analýzy stávajících řešení. Při této analýze bylo zjištěno, že naprostá většina již vytvořených aplikací využívá právě webového rozhraní. Z tohoto důvodu jsme se rozhodli pro vytvoření aplikačního klienta.

Tvorba aplikace byla týmová, a proto jsme museli řešit i některé problémy, které jsou s prací v týmu spojeny. Většina problémů byla pouze komunikačního rázu a díky vzájemné vstřícnosti byly tyto problémy minimalizovány. Práce v týmu přinášela i několik výhod, jako bylo například sdílení zkušeností a poznatků nebo přinášení nápadů, které ostatní nenapadly.

Jedním z největších přínosů, které mi tato práce poskytla bylo seznámení s technologiemi, které jsou obsaženy v Javě EE, a s těmi, které s touto platformou úzce spolupracují. Dle mého názoru je znalost těchto technologií v dnešní době velmi důležitá a užitečná a to především z důvodu velké oblíbenosti a použitelnosti této platformy. Tyto technologie také velmi urychlují a usnadňují práci programátora a poskytují mu nástroje pro vytváření komplexních a propracovaných aplikací. Programování aplikační vrstvy bylo velmi poučné a přínosné, protože se jedná o oblast, která bude, dle mého názoru, procházet velkým vývojem a využívanost vícevrstvých aplikací se také zvýší.

Velkým přínosem pro tvorbu aplikace byla vývojová platforma NetBeans, která poskytovala další nástroje pro usnadnění práce. Asi největší podporu práce poskytovala při tvorbě GUI, které by bylo velmi nesnadné vytvořit jiným způsobem. Velikou pomocí při práci byla i nápověda obsahující informace o třídách, metodách a konstantách. Tuto nápovědu je sice nutné do NetBeans přidat manuálně, ale výsledek se rozhodně vyplatí. Velkou podporou práce je i automatická nabídka tříd, metod a konstant, oznamování chyb v kódu a automatické dokončování příkazů. Z předchozího výčtu je patrné, že podpora práce je opravdu rozsáhlá, a to jsem zmínil pouze některé základní a na první pohled patrné nástroje.

V průběhu práce se objevilo několik problémů, přičemž jedním z nejsložitějších bylo vytvoření algoritmu pro správné vykreslení událostí spadajících do určitého dne. Tento problém se naštěstí podařilo vyřešit, a to díky pomoci ostatních členů týmu, kteří přispěli svými návrhy a zkušenostmi.

Jedním z cílů práce bylo vytvoření aplikace, která by ukazovala možnosti aplikací postavených na platformě Java EE, což se nám v mnoha oblastech podařilo. Možnosti Javy EE jsou nicméně velmi rozsáhlé a předvést všechny, které nabízí, by bylo přinejmenším velmi nelehké. Java EE Organizér je snadno pochopitelná a intuitivní aplikace umožňující zaznamenávání a práci s kontakty, kategoriemi, úkoly a událostmi. Zvažovaných možností

rozšíření aplikace je několik. Jedná se například o rozšíření umožňující sdílení informací mezi uživateli, přidání e – mailového klienta, nebo i webového rozhraní.

Literatura

[1] BRANICKÝ, M.: *Java Servlets – predstavenie technológie* [online]. 25. 4. 2003 [cit. 2009-05-13].
URL http://interval.cz/clanky/java-servlets-predstavenie-technologie/

[2] *Firebird – The RDBMS that's going where you're going* [online]. 2009-05-01 [cit. 2009-05-02].
URL http://www.firebirdsql.org/

[3] *GlassFish – Open Source Application Server* [online]. 2009-05-02 [cit. 2009-05-03].
URL https://glassfish.dev.java.net/

[4] JENDROCK, E.; BALL, J.; CARSON, D.; aj.: *The Java EE 5 Tutorial: For Sun Java System Application Server 9.1 [online]*. Sun Microsystems, Inc., October 2008 [cit. 2009-03-06].
URL http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf

[5] KAYL, K.: *Java Tech Review* [on-line]. April, 2002 [cit. 2009-05-12].
URL
http://java.sun.com/developer/technicalArticles/RoadMaps/techreview/

[6] NISZANSKÝ, P.: *JSTL 1.1 – CORE TAG LIBRARY* [online]. 10.12.2003 [cit. 2009-05-13].
URL http://nb.vse.cz/~zelenyj/it442/eseje/xnisp01/core.htm

[7] PALÁT, P.: *Java EE organizér – softwarová architektura*. Bakalářská práce, Brno, FIT VUT v Brně, 11.5.2009 [cit. 2009-05-12], cD – ROM, slajdy/j2ee zaklad.odp.

[8] RYCHLÝ, M.: *Java Platform, Enterprise Edition (Java EE)* [online]. 19. listopadu 2007 [cit. 2009-05-12].
URL http://www.fit.vutbr.cz/research/grants/index.php?file=%2Fproj%2F378%2Fslides-javaee.pdf&id=378

[9] *Třívrstvá architektura* [online]. April 19th, 2006 [cit. 2009-05-08].
URL http://www.itexpert.cz/trivrstva-architektura/

[10] Wikipedia: *SOAP with Attachments API for Java – Wikipedia, The Free Encyclopedia* [online]. 2008 [cit. 2009-05-14].
URL http://en.wikipedia.org/w/index.php?title=SOAP_with_Attachments_API_for_Java&oldid=231420545

[11] Wikipedia: *Java Architecture for XML Binding – Wikipedia, The Free Encyclopedia* [online]. 2009 [cit. 2009-05-14].

URL http://en.wikipedia.org/w/index.php?title=Java_Architecture_for_XML_Binding&oldid=288941164

[12] Wikipedia: *StAX – Wikipedia, The Free Encyclopedia* [online]. 2009 [cit. 2009-05-14]. URL http://en.wikipedia.org/w/index.php?title=StAX&oldid=286968277

[13] ZENDULKA, J.: *Architektura klient/server a třívrstvá architektura* [online]. 16. prosinec 2003 [cit. 2009-05-12]. URL http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/10_clsrv.pdf

Dodatek A

Obsah CD

Disk obsahuje následující strukturu

- *thesis* – Složka obsahuje zdrojové soubory k bakalářské práci a také výsledný dokument ve formátu PDF.
- *manual* – Složka obsahuje uživatelské manuálové stránky ve formátu HTML.
- *source* – Složka obsahuje kompletní zdrojové soubory k aplikaci Java EE Organizér(XML soubory, NetBeans projekty, ...).