Софийски университет "Св. Климент Охридски"

Факултет по Математика и Информатика

Катедра "Информационни технологии"

ДИПЛОМНА РАБОТА

Тема: Разпределена Web система за управление на човешките ресурси

Дипломант: Благовеста Йорданова Йорданова

Специалност: Информатика

Факултетен номер: М21030

Научен ръководител: н. с. С. Вървбанов

София, 2005

Съдържание

1 Увод, описание на задачата, практическа полза от реализацията

1.1 Кратко въведение

В настоящата дипломна работа са използвани специфични термини и понятия в областта на управлението на персонала и временното назначаване. Те са обяснени в края на първа глава. Тук е представено кратко описание на предметната област и целите които си поставя тя.

Агенциите за временно назначаване (Temporary Staffing Agency -TSA) са форма на посредници на пазара на труда, които задоволяват нуждите на компаниите-клиенти от персонал с различна квалификация за определен период от време.. Докато в началото на 70-те години отрасъла за временно назначаване (Temporary Staffing Industry - TSI) е само едва забележим в САЩ и европейските пазари на Великобритания, Франция и Холандия, сега той наистина може да бъде наречен глобален. Утвърждаването на TSI в неговите основни пазари в северна Америка и Западна Европа е съпроводено със съществото му разрастване в Източна Европа и напоследък в новопоявилите се пазари в Латинска Америка и източна Азия.

За да бъдат намалени административните разходи и да се увеличат продуктивността и конкурентноспособността, съвременните компании въвеждат Web базирани средства, вместо да увеличават персонала си. Основната задача е да се създаде стабилна основа, която може да продължи на бъде надграждана. И тази основа трябва да бъде Web базирана и възможно най-много да предлага самообслужване.

Продуктивността на персонала в наши дни зависи от достъпността до информацията. Web базираните фирмени системи осигуряват на персонала динамична информация, от която се нуждаят, за да взимат по-добри и своевременни решения. Те позволяват голям избор от дейности - възможности за персонала да управлява своето израстване, както и възможности за ръководителите да развиват своите отдели.

Успехът на електронният бизнес на една компания зависи от наличието на стабилна Web основа. Възможните пречки за една ефективна инфраструктура се свързват както с управлението, така и с използваните технологии. Стандартизираната и гъвкава архитектура, добре управляваните ресурси и усъвършенстваните системи за мениджмънт, които помагат (вместо да създават пречки) на бизнес процесите, са сред основните компоненти на успешната Web стратегия.

1.2 Цел на дипломната работа

Целта на дипломната работа се свежда до проектирането и създаването на пълноценно Web приложение. Разработката си поставя следните задачи:

- Приложението трябва да е способно да обслужва голям брой потребители и групи (фирми и техните филиали).
- Трябва да предостави инструмент на управленските структури за достъп до разнообразни по предназначение справки в реално време;
- Трябва да дава възможност за консолидиране и натрупване на най-подробна информация за персонала, работната заплата, осигурителните вноски и други данни;
- Приложението трябва да може да даде възможност за контрол върху работата на лицата обработващи информацията за персонала;
- Трябва да може да поддържат едновременно както стандартните Web браузъри.[1] (browser) за PC, така и новите мобилни устройства, които използват различни от HTML[2] Markup езици (WML[3], XHTML[4]) (т.е. едно приложение за всички видове клиенти). Това трябва да става, като информацията се представя с XML[5] документ и XSL[6] стилово

[1] Browser – програма-клиент, която може да изисква и изобразява документ получен от Web сервера, като интерпретира форматиращите инструкции на медийния език, с който е представен документа

[2]HTML – Hyper Text Markup Language (хипертекстови маркъп език)

[3]WML – Wireless Markup Language (маркъп език за безжични устройства)

[4] XHTML – eXtensible Hyper Text Markup Language (разширяем хипертекстови маркъп език)

[5]XML – eXtensible Markup Language (разширяем маркъп език)

множество (stylesheet), с което първата да се изобразява в разбираем за клиента медиен формат (WML, XHTML). По подразбиране трансформацията ще се извърши от страна на сървъра;

- Да улесни управлението на процеса на разширяването на приложението, като раздели представянето (оформлението на изгледа) на данните от самите данни с помощта на XML и XSL.
- Приложението трябва да може да осигури информационни услуги, като използва XML за описване на структурата, представянето и пренасянето на информацията.

Отделянето на данните от тяхното представяне би помогнало за по-лесното управление на проекта, най-вече като дава възможност програмистите да не се интересуват от дизайна на Web страниците, а само от логиката и данните.

Дизайнерите няма да се занимават с логиката на приложението, а само с неговия изглед. Те ще имат възможност да започнат изграждането на оформлението на по-ранен етап паралелно с разработката на самото приложение, като изпробват резултата от създадените от тях XSL трансформации върху статични XML документи.

Промяната на дизайна на Web сайта няма да променя логиката на приложението, което ще намали риска от грешки.

Благодарение на факта, че естественият език на документите в системата е XML, клиентите могат да използват Web за автоматично обновяване на информацията на други системи (счетоводен, финансов и друг софтуер).

1.3 Структура

Дипломната работа се разделя на следните глави:

[6]XSLT – XML Stylesheet Language for Transformations (XML стилов език за трансформации)

Глава I започва с кратка история и въведение в областта. Тя описва и основните тенденции в развитието на отрасъла за временно назначаване.

В глава “Използвани технологии” се дава кратко описание на последните. Прави се сравнение с други съществуващи технологии.

В глава “Проектиране и реализация на системата” се проследяват проблемите свързани с реализацията на една многослойна Web базирана система и се описва подхода възприет за решението им.

Глава „Примерни сценарии за използване на системата” съдържа кратко описание на системата от потребителска гледна точка. Даден е примерен сценарии за работа с нея.

Глава “Заключение” описва постигнатите резултати, дава насоки за бъдещото развитие на системата и формира заключение.

В глава “Използвана литература” се посочват основните източници използвани при реализирането на дипломната работа.

2 Особености, тенденции и глобализация на дейността на фирмите за временно назначение

2.1 Тенденции и развитие

През последните няколко години, поради различни причини, много компании започват да предпочитат смесица от постоянно и временно назначен персонал. Повече от 70% от временно наетите служители работят в областта на офис администрирането и индустрията, но процента на специалистите от различни области нараства всеки ден. Например: ИТ фирмите от Силиконовата долина временно наемат служители за специфичен проект или фармацевтичните компании наемат хора за тестовата фаза за ново лекарство, което са разработили. Днес служителят по договор за временно назначаване може да е както счетоводител и системен анализатор така и медицинска сестра или адвокат.

В резултат на гореизброените TSI се оказва един от най-бързо развиващите се отрасли в Съединените щати през последните няколко години.

С двуцифрени проценти печалби след 1991. той става все по-важен аспект на икономиката им.

Причините за разрастването на отрасъла са много и обхващат различи аспекти от развитието на стопанската дейност в страната и промяната в изискванията и нагласата на работодателя. Някои от изследователите поставят в основата на развитието на пазара рецесията настъпила през 2001 година (Jamie Peck, 2004), според други причините се коренят в последвалата я безработица или особеностите в развитието на бизнеса. Макар да изглежда просто, мениджъри и работодатели обикновено късно осъзнават нуждите си от допълнителна работна сила, като в този случай единствения изход, за да се спазят изискванията по договорите и отношенията е наемането на временен персонал.

Въпреки, че глобализацията се превръща във често срещана дума в наши дни, много аспекти от тази комплексна съвкупност от процеси остават недобре разбрани. Докато литературата по въпроса за икономическата глобализация продължава да расте, множеството от отраслите, използвани, за да се демонстрира този процес – разбиран като увеличаване на функционалната интеграция на международната дейност (Dicken, 2003) – остават удивително ограничени. Неедноктатно се използват примери в областта на електрониката, автомобилите и облеклото, за да се покаже взаимозависимост, която може да охарактеризира съвременната международна производствена система. От една страна, това е може би обяснимо, защото в тези производствени отрасли международните функционална интеграция е най-очебийна. От друга страна този подход дава изопачена представа за естеството на икономическата глобализация: а именно - като основана на чужди директни инвестиции и растяща международна икономическа дейност.

Отрасълът за временно назначаване предоставя изключителна възможност за изследване на тези явления в дълбочина.

Въпреки изследванията за това, кои типове хора предпочитат временните договори (Parker 1994), за законодателната основа(Rogers, 2000), за причините, поради което клиентските фирми използват агенции за временно наемане (Vosko, 2000) и за значението на пола в тези случаи , много малко се знае за географската организация и стратегии на международните корпорации.

По същият начин – докато изследванията на индустрията за временно назначение на двата основни пазара в САЩ и Великобритания нарастват значително през последните години (Theodore and Peck, 2002; Ward, 2003) , информацията за другите подобни пазари в останалите страни и ролята, която водещите международни агенции играят в тях, остава недостатъчна.

В световен мащаб този отрасъл е силно концентриран, като доминират международните инвестиции на група елитни северноамерикански и в по-малка степен западноевропейски агенции. Увеличението на тези инвестиции продължило до средата на 90-те години от всички водещи агенции поради значителен спад на дела им придобит в собствената им страна. През 1997 година, промени в законодателството, управлявани от Европейския съюз, премахнаха в повечето страни-членки забраната за временно назначаване. Като последствие от това, например, Adecco, най-голямата агенция в световен мащаб с годишен приход над $12 милиарда, увеличава своите доходи от чужбина с от 18% до 80% през периода от 1995 до 2000г.

Според статия на Ройтерс ("Merger pace picks up for U.S. temporary help firms", 12/25/2003), в отрасъла за временно наемане се извършват нови обединения. С повече от 7 000 агенции в САЩ, според статията, индустрията е готова за обединение. Причината за това е бумът на пазара на акции, който дава на някои от по-големите фирми достъп до повече капитали.

В световен мащаб, TSI продължава да се уедрява – като четирите най-големи компании притежават 32% от пазара. Голямата четворка предлагат услугите си предимно на големи, международни компании, често в професионалната и управленската област. Паралелно с нарастването на TSI в международен аспект корпорациите продължават да търсят начин да намалят постоянния си състав. Показателен пример в това отношение е Германия, която през последната година либерализира своето законодателство свързано с временното назначаване. Разхлабването на нейната строга политика за защита на труда е знак за появата на новото работно място на бъдещето. Нововъведенията в законодателната уредба на от 1997 година на Европейския съюз, като резултат от повишеното търсене, доведоха до значително нарастване на услугите от този вид в Европа и доминация на европейските компании.

Според неотдавнашно изследване, годишния приход на отрасъла в глобален мащаб е около $120 милиарда (Staffing Industry Analysts, Inc, 2003).

В Таблица 2.1-1 повечето от изброените фирми предлагат като услуги както консултации и подбор на персонал, така и временно назначаване.

Компания	**Държава**	**2002 приход (милиард)**	**Пазарен дял**
Adecco	Швейцария	$18,1	15%
Manpower	САЩ	$10.6	9%
Vedior	Холандия	$6.4	5%
Randstad	Холандия	$5.7	5%
Kelly Services	САЩ	$4.3	4%
Spherion	САЩ	$2.1	2%
Robert Half	САЩ	$1.9	2%
CDI	САЩ	$1.2	1%
MPS	САЩ	$1.1	1%
Venturi Partners	САЩ	$0.6	<1%

Таблица 2.1-1- Водещи фирми за временно назначение

За нарасналото значение на отрасъла говорят и следните цифри:

В щата Вашингтон отрасълът за временно наемане работи с над 40, 000 души и плаща годишни заплати за над $1 милиард. Отрасълът е създал над 1 милион нови работни места за последните 7 години.

В Калифорния броят на временно наетите през последните 15 години почти се е удвоил и днес броят им съставлява около 3% от работещите на трудов договор.

В световен мащаб за 2004 година агенциите са генерирали повече от $70 милиарда годишен доход, $63 милиарда от които са от услугите за временна заетост и $7 милиарда и услугите с постоянно заплащане. За същата година процента на временно назначените се е увеличил с 12.4, като това е втората година с двуцифрен процент на растеж, през която индустрията достига нива последно постигнати през 2000 година (данните са от изследване на American Staffing Association).

2.2 Особености

Като цяло TSI обхваща назначаването на човешки ресурси за непостоянни нужди на работното място. Временното назначаване е услуга предлагана на компании, които изнасят процеса по наемане на персонал във външна фирма, за да покрият нуждите си от такъв за определен, конкретен период от време.

Продължителността на назначението може да има специфични параметри или да бъде неопределено. Обикновено то варира в зависимост от нуждите на компанията – клиент. Договорите може да са краткосрочни или дългосрочни. Договор за наемане се сключва, от една страна, между фирмата-клиент и агенцията и от друга – между агенцията и хората, които търсят подобен тип посредничество. За разлика от агенциите за подбор на персонал, агенцията за временно наемане поема грижата за социалното и здравно осигуряване, застраховки и др. за хората сключили договор с нея.

Заплащането на служителите по временни договори се извършва през предварително договорен, кратък период – обикновено седмица или две. Срока на заплащане може да бъде съобразен и със политиката на компанията - клиент. Типично за този тип дейност е заплащането на изработен човекочас.

Бумът в развитието на отрасла доведе до съществен проблем, а именно: в момента липсват програмни решения, които да предлагат гореизброената функционалност. Съществуващият софтуер за управление на човешките ресурси не отговаря в пълна степен на изискванията на TSI. Усеща се липсата на интегриран продукт, обхващащ изцяло процеса по управление на персонала в една фирма за временно назначение и с лесен за използване интерфейс.

За щастие Интернет допринесе за наличието на голям избор от средства за създаването на продукт за оптимизация на управлението на работната сила, които могат да бъдат използвани, за да се подобри продуктивността в една TSA. Чрез въвеждането на подходящ web базиран продукт TSA могат по-ефективно от управляват бизнеса си .

Едно такова решение за оптимизация на работния процес би позволило да се фокусира върху основните детайли в работата чрез дейности с добавена стойност. Например може да се наблюдават и модифицират дейностите, свързани с управлението на човешките ресурси, да се правят отчети за извършените разходи - и всичко това да се извършва онлайн. Това решение също така би допринесло за ефективността на управлението на човешки ресурси, чрез автоматизиране на много от времеемките административни дейности като наемане на персонал, мениджмънт и комуникация между членовете на персонала. Това увеличава удовлетвореността на служителите и позволява на компанията да се концентрира върху по-важните за нея предизвикателства.

3 Използвани технологии

3.1 Основи за избор на технология

С навлизането на Интернет и взаимната свързаност между различни системи и бази от данни във все повече приложения днес се налага нуждата от общ интерфейс, възможност за мащабиране (или нарастване на производителността с нарастване на броя потребители, за които географските ограничения отпадат), преносимост и възможност за разпределяне на работата между 2 или повече машини.

Адекватен отговор на горните изисквания се явява т.нар. трислоен модел на системна архитектура. В основата на този модел лежи едно разграничение, което позволява лесни, бързи и с по-малко грешки преносимост, мащабиране и разпределение. Това разграничение отделя в самостоятелни архитектурни слоеве съхранението на данни, правилата за тяхната обработка и начина за представянето на информацията, явяваща се резултат от предните две. Най-просто могат да бъдат наречени данни, бизнес логика и интерфейс.

Така пред всеки системен архитект на макро ниво стои въпросът как да подреди елементите на трислойния модел, а на микро ниво – как да подбере технологиите в рамките на съответния слой.

3.2 Web приложение – архитектура и принципи

Интернет WWW архитектурата предоставя лесно приспособим и мощен програмен модел. Ресурсите са представени в стандартен формат и се разглеждат от приложения известни като Web браузъри. Web браузърите са мрежови приложения, те изпращат заявки за именувани обекти намиращи се на някой сървър, а той от своя страна отговаря предоставяйки данните в някой от стандартните формати. WWW стандартите определят много от механизмите необходими за изграждането на подходяща приложна среда, включвайки:

- Стандартен модел за именуване – Всички сървъри и ресурси са именувани чрез Интернет стандарта *Uniform Resource Locator (URL)*
- Типизиране на съдържанието – Съдържанието принадлежи на определен тип, позволявайки на web браузърите правилно да го обработват, базирайки се на неговия тип.
- Стандартен формат на съдържанието – Всички web браузъри. поддържат множество от стандартни формати. Те включват HiperText Markup Language (HTML), JavaScript скриптов език и множество от други.
- Стандартни протоколи – Стандартните мрежови протоколи позволяват на всеки web браузър да общува с всеки web сървър. Най-използваният протокол във WWW модела е *HiperText Transfer Protocol (HTTP).*

Първоначално Web се е състоял от сървъри, всеки от които поддържа известен брой статични хипертекстови документи. Клиентът е избирал определен документ от даден сървър и с помощта на хипервръзките в него е можел да изисква и други документи (ресурси) от същия или от друг сървър. Желаният документ е получаван с помощта на HTTP протокола.

В последствие се появява идеята за динамично съставяне на поискания от клиента документ (документът не е присъствал готов на сървъра, а се е създавал всеки път при неговото изискване), а по-късно и до изграждането на пълноценни приложения.

Web приложенията работят на сървъра и използват хипертекстовите документи, за да изградят своя интерфейс. Това става по следния начин:

1) Клиентът изпраща заявка за документ към сървъра (следствие на избиране на хипервръзка от друг документ, попълване на форма или чрез директно избиране на документа с неговия идентификатор (URL)
2) Сървърът активира подходящото приложение, което да обработи заявката.
3) Приложението извършва необходимите действия (то може да е получило и параметри в заявката, следствие на попълване на форма и др., а действията, които трябва да извърши, обикновено се определят от идентификатора на документа, който се изисква) и връща като отговор хипертекстови документ, който да покаже резултата от операцията и евентуално да окаже следващите възможни действия (с помощта на хипервръзки в новия документ например).

Common Gateway Interface (CGI) позволяват на Web сървъра да пусне отделен процес, базиран върху входа на потребителя, да работи върху тази информация и да връща динамично създадените Web страници към клиента. Една CGI програма би могла да извърши всякакъв тип обработка на данни, от която се нуждае програмиста и би могла да върне всякакъв тип страници, които HTML би позволил.

Обаче CGI приложенията имат някои неудобства. Всяка заявка трябва да пусне нейния собствен процес върху сървъра, така че множеството от заявки могат лесно да се объркат, дори и при средно натоварен сървър. Задачата за създаване на файл, пускане на отделен процес, изпълняване на процес и след това записване и връщане на друг файл е относително бавна.

Фигура 3.2-1 – Основен принцип на работа при CGI

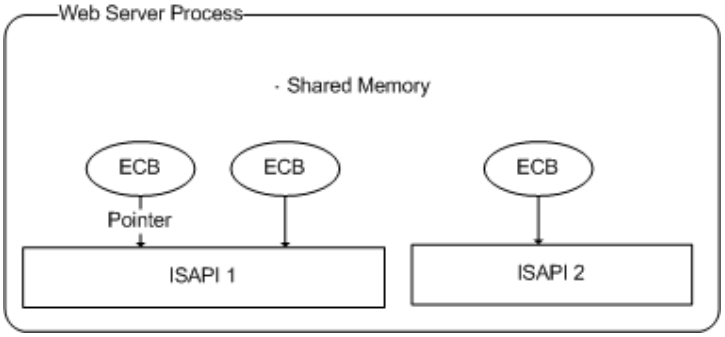
3.3 ISAPI

Основните web сървър доставчици – Microsoft и Netscape, видяха присъщата слабост в CGI програмирането. Вследствие на това, вместо създаване на отделни процеси при всяка заявка всяка компания написа API (Application Program Interface) за нейните web сървъри, които позволяваха на разширенията на web сървъри да бъдат стартирани като dynamic link libraries (динамично свързвани библиотеки).

ISAPI предоставя много по-широк достъп до Web сървъра, отколкото простите функции на традиционните интерпретаторни машини, като Perl, PHP и много други технологии.

Интерфейсът съдържа динамична свързана библиотека (Dynamic-Linkable Library-DLL), която се състои от изпълними рутини, изисквани от взаимодействието с различни наследствено информационни системи. DLL подрежда функциите по същия начин както Gateway програмите, но за разлика от външните програми, те се стартират като част от процеса на web сървъра, изпълняващи техния код в същото пространство на паметта, в което е и самия web сървър.

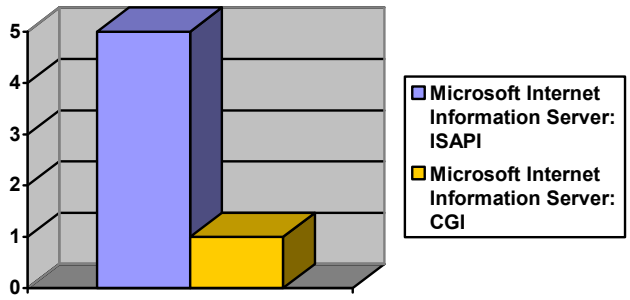
При всяка заявка от клиент web сървърът се обръща към HttpExtensionProc и подава указател на ISAPI Extension Control Block (ECB), който съдържа информация за заявката. ISAPI DLL може да използва обратни извиквания към сървъра за да получи информация за променливите на средата. ECB дава достъп на програмиста до някои поддържащи функции, като например: пренасочване на URL, управление на сесиите и хедърите на отговора, които не се поддържат от CGI приложенията.



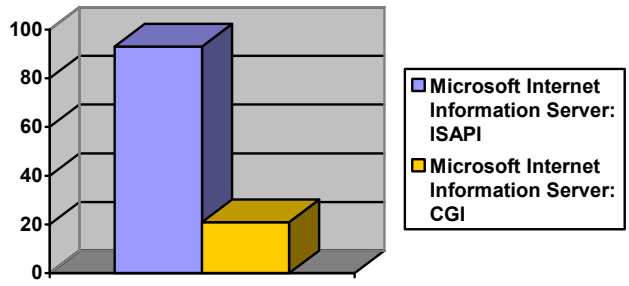
Фигура 3.3-1 – Основен принцип на работа при ISAPI

Вместо да трябва да подават информация като файлове, разширенията на web сървъра могат да подават информацията вътре в самото пространство на паметта. Ето защо създаването на процес за обработка на всяка заявка не изисква допълнителни ресурси. Тоест, в паметта остават само активните функции, докато сървъра динамично променя неактивните такива, за да се освободят ресурси на системата. Като резултат на това е многократно подобрено сървърното представяне. Това позволява по-бързи, по – ефективни и по-малко консумиращи ресурси web приложения..

Диаграма 3.3-1 и Диаграма 3.3-2 по - долу сравняват резултатите на 64 клиента за 100% CGI и ISAPI. Графиките показват броя на свързванията за една секунда и произведените мегабайтове за секунда:



Диаграма 3.3-1 Мегабайти за секунда



Диаграма 3.3-2 – Конекции за секунда

От тези графики е лесно да се види че при директно сравнение ISAPI е близо 5 пъти по–бърз от CGI при изпълнението на някои задачи на Microsoft

Internet Information Server. Това означава, че адаптирането на CGI базирано приложение към ISAPI интерфейса ще доведе до неколкократно намаляване на времето, през което потребителя очаква отговор от web сървъра и че сървъра ще има възможност да отговори на повече заявки.

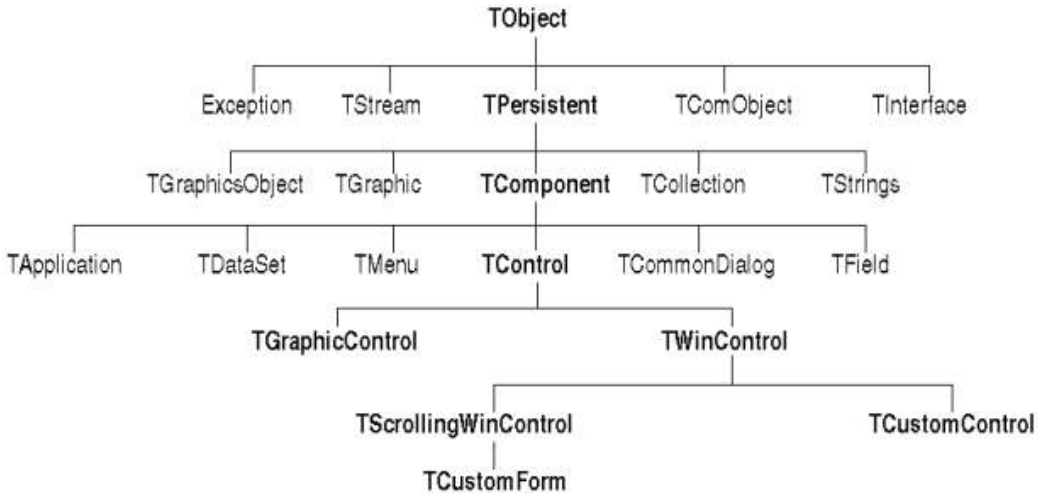
3.4 Делфи

3.4.1 Технология на Делфи, VCL компоненти

Продуктът Delphi е едно от най-популярните средства за бърза разработка на приложения под Windows. Поради своята изключителна леснота за използване, добра надеждност и големи възможности, не случайно Delphi се е превърнало в любимото средство за разработка на много програмисти. Продуктът е подходящ подходящ както за създаване на малки и прости програми, така и за разработване на големи и сложни програмни продукти. Той има прекрасни възможности за разработка на клиент-сървър и Internet ориентирани приложения, а също и сериозна поддръжката на бази от данни, което го прави мощна съвременна система за разработка на приложения. Все пак най-голямото предимство на продукта си остава ненадминатата леснота и удобство за работа.

Архитектурата на Delphi се базира на обектната технология - основава се на модули от визуални класове, наречени компоненти и стандартната библиотека VCL (Visual Component Library). Компонентите са основната градивна единица на всяко приложение. Потребителите имат възможност да използват не само стандартните за продукта компоненти, но и такива, предлагани от други разработчици, както и да разработват свои собствни. И понеже в Интернет могат да се намерят компоненти за почти всичко, това прави средата за разработка Delphi с на практика неограничени възможности.

VCL съдържа компоненти, които са видими по време на изпълнение – като контроли за редактиране, бутони и други елементи на потребителския интерфейс, както и други скрити контроли като таймери и структури от данни. Диаграма 3.4-1 показва някои от основните типове, от които е съставена VCL.



Диаграма 3.4-1 – Йерархия на компонентите във VCL

Обектите, наследени от TComponent имат свойства и методи, които им позволяват да бъдат инсталирани на палитрата с компоненти и добавени към формите в Делфи.

3.4.2 Делфи приложение за web – web модул, web диспечер

Всяка операция, която може да бъде извършена по принцип с Делфи, може да бъде включена в Web сървър приложение. С няколкото вградени Делфи web компоненти могат да бъдат създадени няколко типа приложения за web. Всеки тип използва съответните наследници на *TWebApplication*, *TWebRequest*, and *TWebResponse*:

Application Type	**Application Object**	**Request Object**	**Response Object**
Microsoft Server DLL (ISAPI)	*TISAPIApplication*	*TISAPIRequest*	*TISAPIResponse*
Netscape Server DLL (NSAPI)	*TISAPIApplication*	*TISAPIRequest*	*TISAPIResponse*
Console CGI application	*TCGIApplication*	*TCGIRequest*	*TCGIResponse*
Windows CGI application	*TCGIApplication*	*TWinCGIRequest*	*TWinCGIResponse*

Таблица 3.4-1 Компоненти на web сървър приложение.

Динамично свързаната библиотека на едно ISAPI web приложение се зарежда от web сървъра Съдържанието на заявката на web клиента се подава до DLL-а като структура и се преработва от TISAPIApplication, което от своя страна създава TISAPIRequest и TISAPIResponse обектите. Всяко съобщение-заявка се прихваща автоматично в отделно нишка на изпълнение.

Компонентата TWebModule е наследник на TDataModule, главния компонент в йерархията на Делфи, и може да бъде използван по същият начин – да осигурява централизиран контрол на бизнес правилата и компонентите на web приложението.

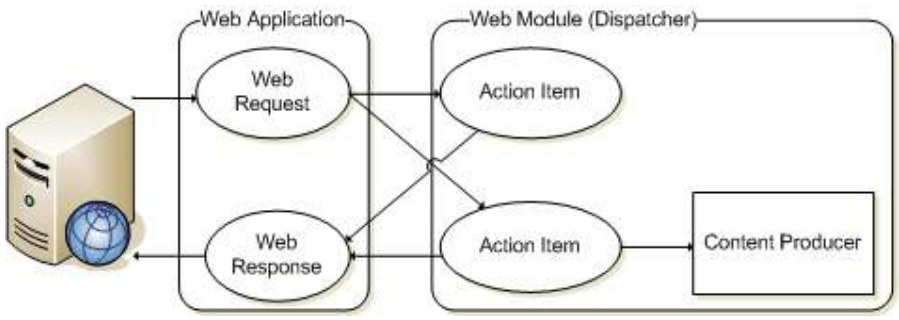
Когато едно приложение получи HTTP заявка то създава TWebRequest и TWebResponse обект. TWebRequest и TWebResponse са абстрактни класове, които реализират HTTP протокола. TWebRequest поддържа достъп до цялата информация, подадена от сървъра към клиента, а TWebResponse съдържа свойствата и методите, които позволяват да се отговори по който и да е от многото начини, които позволява HTTP протокола. ISAPI базираните web приложения всъщност използват TWebRequest и TWebResponse, които са наследници на абстрактните класове и са декларирани в ISAPIAPP.pas.

След създаването на TWebRequest и TWebResponse приложението подава тези два обекта към web диспечера (Web Module или TWebDispatcher компонент).

Web диспечера контролира потока във web сървър приложението. Един диспечер интерпретира HTTP заявката и извиква съответния обработчик на събитието, композира отговор и го връща към извикващия HTTP сървър.

Диспечерът съдържа колекция от action items (TWebActionItem), които знаят как да прихванат известните типове на HTTP заявките и на всеки от които съответства предварително дефиниран обработчик. “Action” е тип, който описва действие. Това включва уникално име на действието в рамките на обработчика - контролер и описание на параметрите му. Диспечера идентифицира съответния action item и предава създадените request и response обекти на неговия

обработчик, така че той да може да изпълни всички необходими действия или да композира съобщение-отговор.



Фигура 3.4-1 – принцип на работа на Delphi Web Module и Web Dispatcher

Всяко web приложение може да съдържа един единствен диспечер. Той може да бъде както като част от web модул (генерира се автоматично при създаването на проекта) или като отделен TWebDispatcher компонент добавен към проекта.

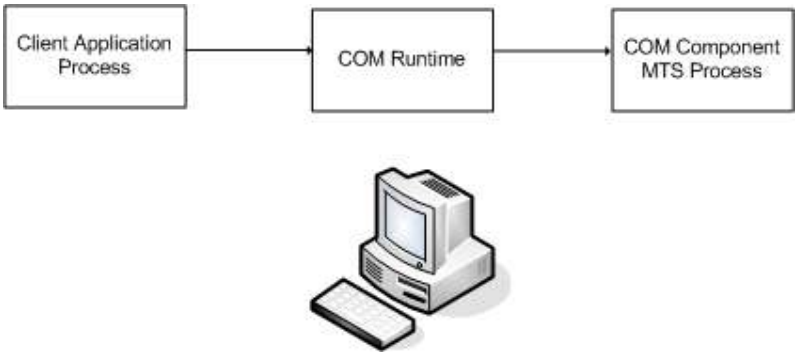
3.5 DCOM

Технологията клиент-сървър и многослойната архитектура винаги са играли ключова роля при разработката на големи бизнес приложения. Най-съществения елемент при този тип архитектура е средния слой. Той има различни имена: сървър на транзакциите, на приложенията, на компонентите или бизнес правилата. Това е слоя, който дава възможност за мащабируемост, адаптивност, възстановяване и управляемост.

Към настоящия момент, основните технологии за многослойно web базирано приложение са: Разпределеният Компонентен Обектен Модел (DCOM) и комбинацията от Java заедно с Common Object Request Broker Architecture (CORBA). DCOM е от Майкрософт – пазарния лидер в програмирането.

COM: COM (Component Object Model) е обектно базиран програмен модел, създаден за да позволи създаването на софтуерни компоненти, чрез използването на множество езици, инструменти и платформи. Той позволява вътрешно–процесорна комуникация между компоненти, стартирани на една и съща машина. Някои от ключовите концепции при COM са:

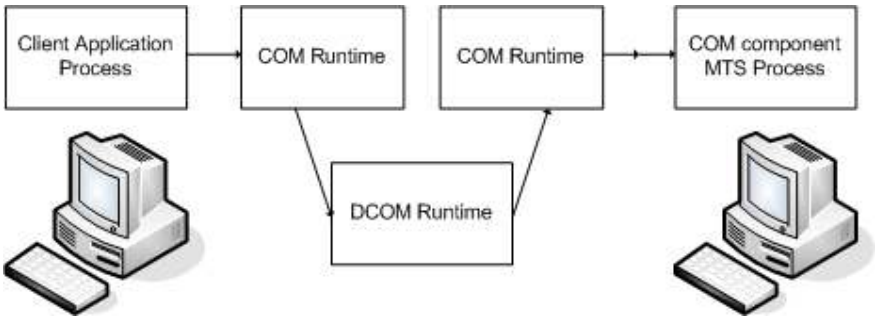
- Интерфейси: Всеки COM компонент има интерфейс, който е дефиниран като методи и свойства. Всеки COM компонент трябва да поддържа поне един интерфейс.
- Класове и сървъри: COM клас е част от код, който се идентифицира чрез уникален идентификатор: CLSID. COM класовете се пакетират в сървъри, които може да са вътрешно процесорни сървъри: (DLL) или извън процесорни сървъри. Вътрешно-процесорните COM сървъри се стартират като част от процесите на клиента, а извън процесорните работят самостоятелна на същата машина като клиента или на отдалечена такава.
- Прозрачност на местоположението: Всеки пък, когато клиента трябва да се свърже с сървърния компонент, не е необходимо той да знае точното му местоположение.



Фигура 3.5-1 - COM

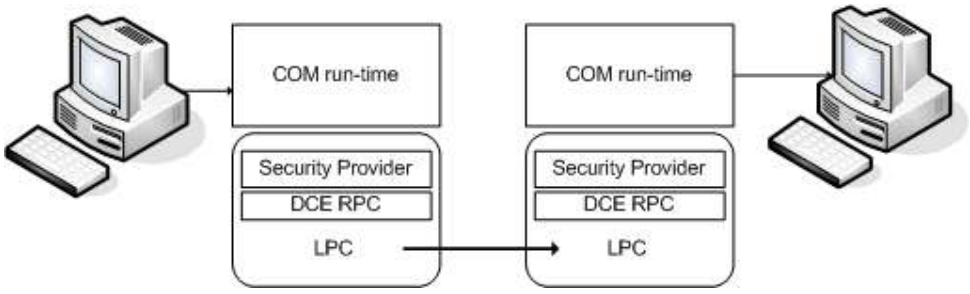
DCOM: (Distributed Component Object Model) е разширение на COM, при което, приложение или част от приложение *(exe или dll)*, е достъпно по мрежата от клиент, изпълнява се на сървър, а резултата се предава на заявителя. При DCOM технологията сървъра може да бъде разположен на всеки един компютър в мрежата и е достъпен от клиента чрез RPC (Remote Procedure Call) . За клиента не е необходимо да знае точно къде в мрежата се намира сървъра. Кода, необходим за работа с отдалечен обект е аналогичен с кода при работа с

локален. За разпределената обработка DCOM предоставя комуникация между компоненти стартирани на различни машини.



Фигура 3.5-2 - Вътрешнопроцесна комуникация между различни машини чрез използване на DCOM

В днешните операционни системи процесите са защитени един от друг. Клиент, който трябва да комуникира с компонент в друг процес не може да се обърне директно към него, а трябва да използва някаква форма на вътрешнопроцесна комуникация поддържана от операционната система. При COM технологията тази комуникация е реализирана по съвършено прозрачен начин: улавят се повикванията от клиента и се изпращат към компонента в друг процес. Фигурата по долу илюстрира начина, по който COM/DCOM библиотеките осигуряват връзка между клиента и компонента.



Фигура 3.5-3 – COM компоненти в различни процеси

Независимост от местоположението

При създаването на разпределени приложения в реалния свят няколко дизайнерски ограничения излизат наяве:

- Компонентите, които взаимодействат повече, трябва да са „по-близо” един до друг;
- Някои компоненти могат да бъдат задействани само на специфични машини и в специфични направления;
- Малките компоненти увеличават гъвкавостта на разработката, но увеличават мрежовия трафик;
- Големите компоненти намаляват мрежовия трафик, но намаляват гъвкавостта на решението.

С DCOM тези критични дизайнерски ограничения са наистина лесни за преодоляване, защото детайлите по работата на DCOM не се специфицират в изходния код. DCOM напълно скрива местоположението на компонента, независимо от това дали той е в същия процес като клиента или е на машина в другия край на света. При всеки от случаите, начина по който клиента се свързва с компонента и извиква неговите методи е идентичен. Не само защото DCOM не изисква промени в сорс кода, той дори не изисква програмата да бъде прекомпилирана. Прости промени в конфигурацията променят начина, по който компонентите се свързват един с друг.

Независимостта от местоположението опростява задачата на разпределените приложния за оптимизиране на цялостното изпълнение. Ако приложението има множество малки компоненти, натоварването на мрежата може да бъде намалено, ако те бъдат разположени в един и същ сегмент от локалната мрежа, на една и съща машина или дори в един и същ процес. Ако приложението е съставено от малък брой големи компоненти, натоварването на мрежата е най-малкия проблем, така че решението е те да бъдат инсталирани на най-мощните машини в мрежата, независимо къде се намират те.

С независимостта на местоположението на DCOM, приложението може да комбинира свързаните компоненти в машини, които са „близо” една до друга, във единствена машина или дори в същия процес. Дори ако голям брой от малки компоненти реализират функционалността на голям логически модул те продължават да комуникират ефективно един с друг. Компонентите могат да

бъдат разположени на машините в мрежата, по начин, следващ логиката на решението. Например, компонентите, които реализират потребителския интерфейс и валидацията, да са разположени близо до клиента, а бизнес правилата, зависещи от базата с данни – на сървър близо до самата база.

Фигура 3.5-4 показва как „валидиращият компонент” може да бъде разположен на клиентската машина, когато мрежовия капацитет между клиента и машината на средния слой е задоволителен и на сървъра, когато приложението е достъпно от клиента чрез бавна мрежова връзка.

Фигура 3.5-4 – DCOM - независимост от местоположението

Езикова неутралност

Една от задачите при разработката на разпределено приложение е избора на езика на който ще се програмира. Изборът на език обикновено е компромис между цена, налични експерти и производителност. Като разширение на COM, DCOM е езиково независим. Всеки един език може да бъде използван за създаването на COM компоненти и тези компоненти може да бъдат използвани от други езици или инструменти. Java, Microsoft Visual C++®, Microsoft Visual Basic®, Delphi, PowerBuilder и Micro Focus COBOL – всички те поддържат DCOM технологията.

С езиковата независимост на DCOM, разработчиците на приложения могат да изберат инструментите и езиците, които най-добре познават. Тя позволява създаването на бързи прототипи: компонентите могат да бъдат създадени най-напред на език от високо ниво, като Microsoft Visual Basic и след това пренаписани на друг езици, като например C++ или Java, който може по-

добре да използва предимствата на специалните функции на DCOM, като многонишковост, free threading и пулинг на нишките.

Управление на връзките

Мрежовите връзки са по-крехки от връзките в една машина. Компонентите в разпределените приложения трябва да бъдат уведомени ако клиента вече не е активен, при хардуерен проблем в мрежата, например. DCOM управлява връзките към компонентите, които се използват от един клиент, както и компонентите, които са разпределени между множество клиенти, чрез поддържане на указателен номер за всеки компонент. Когато клиент установи връзка с компонент DCOM увеличава номера на броя на референциите към компонента. Когато клиентът освободи конекцията си, DCOM намалява броя на референциите. Когато броя достигне нула, компонента може да се освободи сам.

DCOM използва ефективен ping протокол за да установи дали клиента е все още активен. Клиентските машини изпращат периодично съобщение. DCOM маркира връзката като разрушена, ако при повече от три пинг подавания компонента не получи обратно съобщение. Ако връзката е прекъсната DCOM намалява броя на активните си референции. От гледна точка на компонента, както случая на прекъсване на връзката с един клиент, така и тоталното разпадане на връзките в мрежата се прихващат от един и същ механизъм на броене на активните референции.

В много случаи потока от информация между компонента и неговите клиенти не е еднопосочен: компонента трябва да има възможност да инициализира някои операции от страна на клиента, като например: уведомяване, че е приключил продължителен процес, че данните, които потребителят разглежда са претърпели промяна (новини или цена на борсови акции) или да предаде съобщение в разпределена среда, като телеконференция или многопотребителска мрежова игра. При много протоколи е трудно да се постигне симетрична комуникация от този тип. При DCOM всеки компонент може да бъде едновременно както доставчик, така и потребителна функционалност. Един и същи механизъм управлява комуникацията в двете

посоки, улеснявайки създаването както на клиент/сървър комуникация така и на равнопоставено взаимодействие.

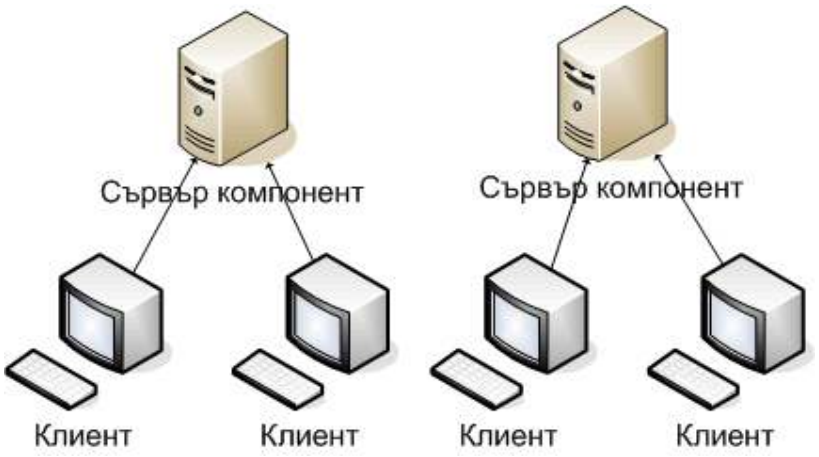
DCOM предоставя мощен разпределен механизъм за управление на паметта, който е напълно прозрачен за приложението. DCOM е наследен симетричен мрежов протокол и програмен модел. Той не само предлага традиционното еднопосочно взаимодействие между клиент и сървър, но също и предоставя богата и интерактивна комуникация между клиенти и сървъри или равнопоставените машини в една мрежа.

Мащабируемост

Критичен фактор за разпределените приложения е способността им да растат заедно с броя на потребителите, количеството данни и изисквана функционалност. Приложението трябва да бъде малко и бързо, когато изискванията са минимални, но то трябва да е способно да поеме допълнителни изисквания без да жертва изпълнението или надеждността си. DCOM разполага с множество качества, които повишават мащабируемостта на приложението:

Пример

Една организация с офиси на различни места: Ню Йорк, Лондон, Сан Франциско и Сидни, може да инсталира компоненти на нейните сървъри. При стандартно натоварване на един сървър той може да обслужи едновременно двеста потребителя, които имат достъп до 50 компонента разположени на него. При инсталирането на ново приложение, което използва някои от съществуващите компоненти с бизнес правила и някои нови, натоварването на сървъра нараства до 600 потребителя, а броя на бизнес компонентите става 70. С допълнителните потребители и приложения времето за отговор става неприемливо през пиковите часове. Администраторът добавя нов, втори сървър и прехвърля 30 от компонентите на новата машина. Двадесет от тях остават на стария сървър, докато останалите 20 са стартирани на двете машини.



Фигура 3.5-5

3.6 XML и XSL

XML е текстов markup език наследник на SGML (Standard Generalized Markup Language). Съществуват два вида markup езици – процедурни и описващи. В процедурните езици маркерите (тагове) се използват, за да укажат как да се обработи текста на документа (например как да се изобрази). HTML е пример за такъв език. При дескриптивните (описващи) markup езици маркерите се използват за да опишат части от структурата на документа, т.е. те са езици за определяне на структурата на документа. XML от своя страна е типичен дескриптивен (descriptive) markup език. Документ представен чрез XML, може да се разглежда като дърво от елементи. Всеки елемент може да включва:

- набор от атрибути,
- текст
- други елементи

Границите на елементите в документа се определят от два маркера:

- отварящ – определя началото на елемента;
- затварящ - определя края на елемента.

XML има следните свойства:

- Разширяем – типовете елементи в XML не са строго определени (както е например в HTML) и езикът може да се допълва с толкова типове,

колкото е необходимо. Това свойство му позволява да бъде мета език, с помощта на който да се създават нови езици.

- Строг в синтаксиса - за разлика от HTML, XML синтаксисът е прост и се спазва много строго. Това го прави лесен за обработка от приложения.
- Проверяем и дефинируем – XML специфицира механизъм за валидиране (validate - проверка) и определяне на структурата на документа. С помощта на DTD (Document Type Definition – спецификация за дефиниране на типа на документ) може да се опише структурата на даден тип документи – какви типове елементи може да притежава документа, колко пъти и къде може да се срещат в документа, могат ли да се пропускат, какви атрибути могат да имат. Този механизъм може да се използва за определянето на нови езици и за автоматична проверка на валидността на документ от даден тип (WML например е дефиниран точно по този начин).

XSLT (XSL Transformations) е език за описване на трансформации, с които даден XML документ може да се преобразува в друг XML документ. Преобразуването се извършва чрез прилагането на правила, които да трансформират входно дърво от XML елементи до резултатно дърво от XML елементи.

Правилата се представят с шаблони, които се прилагат върху елементите на входното дърво. В резултат от прилагането на даден шаблон се генерират елементи в изходното дърво. Основният принцип е шаблоните да се прилагат чрез рекурсивно обхождане на дървото-източник.

XSL трансформацията се описва във файл с помощта на XML и се пази отделно от XML документа.

3.7 Jabber протокол

Началото на проекта Jabber е било поставено от Джеръми Милър в началото на 1998 г. Със създаването на сървъра jabberd. Скоро след това към проекта се присъединяват няколко души, които продължили работата над

сървъра и клиенти на jabber за Windows и Linux, а също и шлюзове към основните системи: ICQ, MSN, AIM и Yahoo).

Jabber е отворен протокол, използващ XML, за бърз обмен на съобщения и информация между потребителите на интернет. Макар че по своята същност той е сходен с основните комерсиални продукти като AIM, MSN, ICQ и Yahoo, той има много предимства.

- Отвореност: протоколът jabber е освободен от лицензи, общодостъпен и въпреки това – лесен за разбиране. Съществуват много реализации на сървъри и клиенти а също и библиотеки с отворен код.
- Разширяемост: с помощта на именованите пространства в XML протоколът може да бъде разширен за изпълнението на трудоемки задачи и за обезпечаването поддръжката на взаимодействието между различни системи.
- Де-централизираност: всеки има право да пусне свой собствен jabber сървър.
- Безопасност: Достъпът до използваният jabber сървър може да бъде ограничен, много от реализациите използват Secure Sockets Layer при обмяна между клиенти и сървър.

4 Проектиране и реализация на системата

Съществуват много начини за реализиране на Web базирано приложение. Сложността на приложението зависи от различните нужди и изисквания. Има и различни ограничения за разработката на едно приложение като възможностите на разработчика (или екипа), очаквания жизнен цикъл на програмата и динамиката на промяната на съдържанието на приложението. Често се случва едно приложение да бъде създадено с цел да обслужва тесен кръг от хора и задачи, а след време, с нарастване на изискванията, програмата да се разрасне и стане трудно модифицируема. Това най-често се случва, ако при самото и проектиране не са взети под внимание възможностите приложението да бъде мащабируемо, лесно преносимо между различни среди, както и многослойно, в случай че се наложи промяна в предметната област която обслужва.

4.1 MVC парадигма

Първите приложения за Web са представлявали програми, които динамично да генерират HTML. При тях характерно е, че се смесва обработващ код, бизнес операции и код, който генерира съдържанието на документа (обикновено HTML страница). Това смесване води след себе си много проблеми – неструктурираност, трудно управление. Едно добро решение се явява в лицето на класическия MVC (Модел-Изглед-Контролер) модел, познат от SmallTalk

MVC се стреми да раздели приложението на три съставни части:

- модел – състоянието на системата, данните и бизнес логиката, на която те се подчиняват. Обикновено от модела се изискват данни за изграждане на изгледа и се изпълняват инструкции за промяна на данните по инициатива на контролера;
- изглед – управлява представянето на информацията – интерфейса с потребителя;
- контролер – компонентите на контролера трябва да определят логиката на приложението – неговото поведение. Те поемат входа на програмата и на базата на този вход активират бизнес действията, които да променят състоянието на модела (данните) и съответните компоненти от изгледа, които да осигурят изхода на приложението.

Фигура 4.1-1 – Принцип на работа при MVC

Ползите от едно такова разделение са много. То би улеснило:

- разпределението на задачите;
- откриването на грешки и отстраняването им;
- ефективното използване на отделни модули;
- внасянето на промени и намаляването на риска от регрес (поява на грешки вследствие промяната).

Затова системата трябва да осигури възможности за прилагане на тази парадигма и да стимулира използването и.

4.2 Дефиниция на единица в системата, характеристики и методи– Object Manager

4.2.1 Общо описание

Проектирането на системата е основано на концепцията, че всяка единица в нея може да се опише в предварително зададен тип. Т.е, независимо дали става въпрос за потребители, проекти, клиенти, работни часове или

фактури, те могат да бъдат отнесени към типа единица-обект на системата. Той се характеризира със следните свойства:

- Име (напр.: Проект);
- Описание на обекта ;
- Път до него (напр.: project);
- Име на XML документите, които описват обекта;
- Име на XSL документите, свързани с него;
- Таблицата в базата данни, която съдържа информацията;
- Видим – дали може да бъде достъпен директно от менюто;
- Документ с помощна информация;
- Полета: (Например за типа проект, всеки от проектите е описан със име, описание, клиент, активен).

Обикновено полетата на един обект съответстват на структурата на съответната таблица в базата от данни.

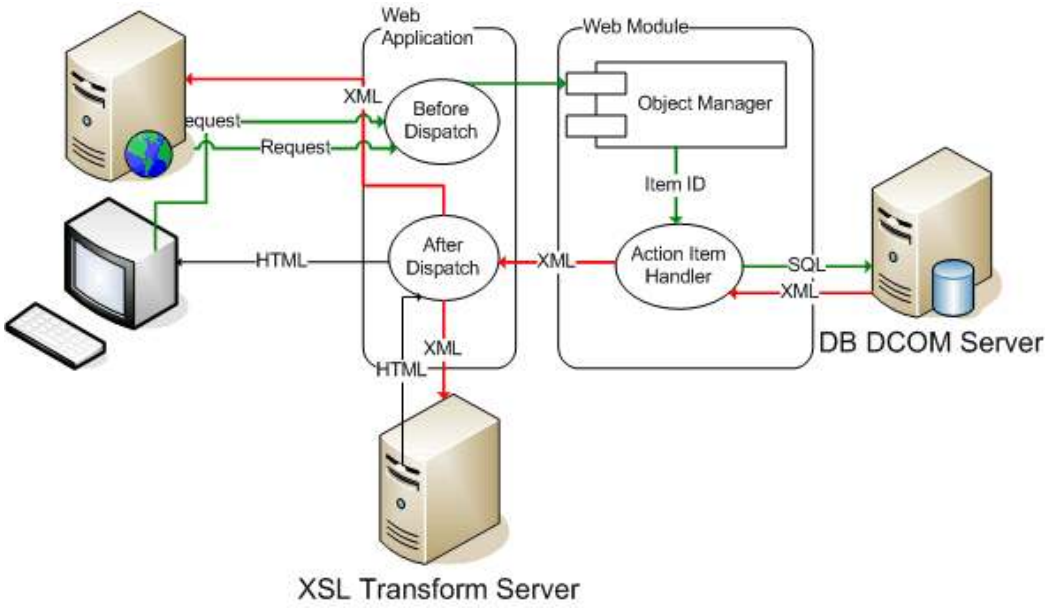
За улесненото представяне на всички типове в системата е създадена VCL компонент – Object Manager. Той описва информацията необходима за изпълняване на всеки един от методите на обекта, както и някои негови свойства. Например, при показването на списъка проектите – дали трябва да бъде използвано страниране, ако да – колко проекта трябва да се показват на отделната страница, кои полета на проекта да се показват във списъка и т.н.

Всеки метод на системата е представен чрез собствен Action Item в Web модула на ISAPI приложението. Клиентската заявка за всеки един от тях се подава на съответния на Action Item обработчик.

При подадена заявка всеки от обработчиците изисква определени входни параметри, за да извърши операцията, като някои от основните са:

- Името на обекта в системата;
- Характеристики на съответния обект – описани в ObjectManager компонента;
- Формат на изходния документ – той може да бъде изходен XML файл, или резултат от трансформация на XML-а със съответен XSL document.

Всеки един от обработчиците, въз основа на подадените му параметри, извършва определена операция върху базата от данни. Обобщеният процес по обработка на заявка от клиент е представен на Фигура 4.2-1:



Фигура 4.2-1 – Основен принцип на обработка на заявка от клиент

4.2.2 Видове методи

Методите – обработчици, в зависимост от резултата, който връщат и входната информация, която изискват, формално могат да бъдат разделени на две групи

4.2.2.1 Методи-изгледи

Този тип методи връщат като резултат данни, извлечени от базата. Данните са структурирани в XML документ с определен формат. Резултатът може да бъде генерирането на определен екран/страница в приложението. Те не водят до промяна на съхранената в базата данни информация Такива са:

Търсене (search): връща форма с полета, които дават възможност за търсене на обекти по въведените критерии.

Списък (list) - обикновено е резултат на търсене. Връща списък със всички единици, които отговарят на зададените в търсенето критерии.

Детайл (detail): Дава детайлна информация за избраната единица от списъка или за нова такава и дава възможност за редактирането и в зависимост от правата на текущия потребител.

Трите метода работят по аналогичен начин. Изискват един и същи параметри на заявката и винаги връщат XML като резултат.

Например, ако потребителя иска да види списък със всички проекти, той трябва да изпрати заявка с път list, и параметър – “o”, който има стойност „project”: *www.domain.com/list?o=project*.

Резултатът от изпълнението на list обработчика ще е XML със следната структура:

```
<?xml version="1.0"?>
<d>
  <cc>
    <c h="PROJECT_ID" t=”INT”> PROJECT _ID</c>
    <c h="Project Name" t="STRING">PROJECT_NAME</c>
    <c h="Project Description" t=”STRING”>PROJECT_DESC</c>
  </cc>
  <rr mx="50">
    <r a="78278" b=”Project 1 Name”  c=“Project 1 Description”/>
    <r a="88150" b=” Project 2 Name” c=” Project 3 Description”/>
    <r a="88149" b=" Project 2 Name " c=” Project 3 Description” />
  </rr>
  <i>
    <s>IxKz8Ay1kwqv6BUUElZnhZNufnI</s>
    <o>project</o>
    <k>PROJECT_ID</k>
    <t>Project List</t>
    <ps>3</ps>
    <pn>4</pn>
  </i>
</d>
```

В „<cc>” тага се намира описанието на всички върнати полета като и техния тип, чрез които определя типа форматиране на стойността. “<rr>” съдържа резултата от изпълнението на заявката до базата данни. В „<i>” тага се

пази допълнителна информация, необходима за връщането на коректен отговор към клиента, например:

<s> - като идентификационен код на потребителската сесия;

<o> - името на обекта;

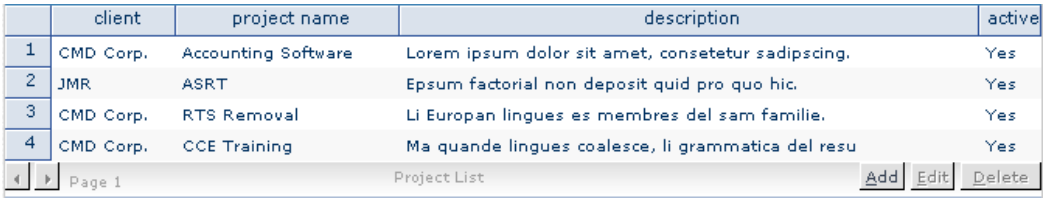
<k> - името на първичния ключ на съответната му таблица в базата от данни;

<t> - заглавие на върнатата информация;

<ps> - брой записи на страница;

<pn> - номер на текущата страница и др.

След съответната трансформация се генерира HTML, който в браузер изглежда по следния начин:



	client	project name	description	active
1	CMD Corp.	Accounting Software	Lorem ipsum dolor sit amet, consetetur sadipscing.	Yes
2	JMR	ASRT	Epsum factorial non deposit quid pro quo hic.	Yes
3	CMD Corp.	RTS Removal	Li Europan lingues es membres del sam familie.	Yes
4	CMD Corp.	CCE Training	Ma quande lingues coalesce, li grammatica del resu	Yes

Page 1 Project List Add Edit Delete

Фигура 4.2-2 – Пример за списък в приложението

Така генерираният списък позволява сортиране по колони и навигация между страниците му с контролините бутони, разположени в долния му край.

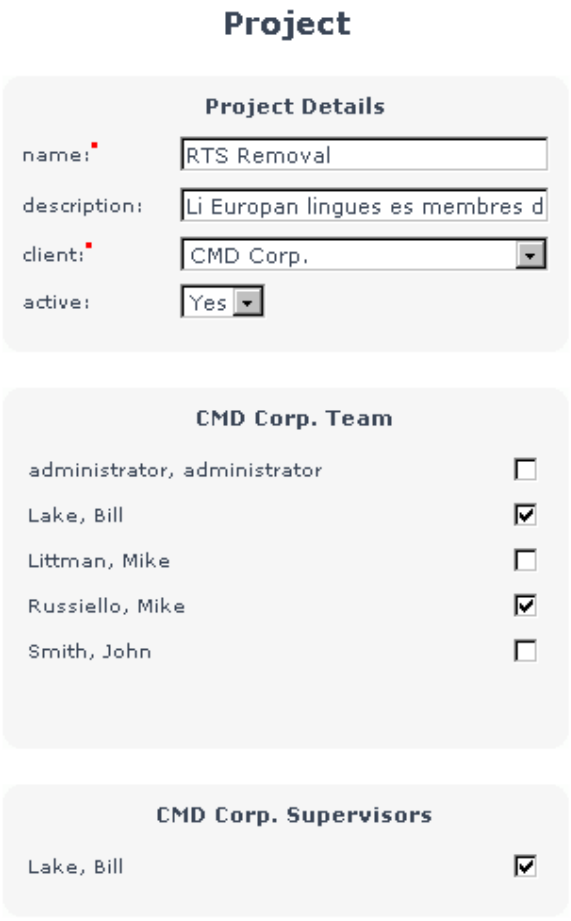
Аналогично, за да се получи информацията за конкретен проект се изпраща същата заявка с един единствен допълнителен параметър: „oid” - идентификационния номер на проекта в базата от данни.

Резултатът от изпълнението на *detail* обработчика ще е следният XML:

```
<?xml version="1.0"?>
<d>
  <cc>
    <c h="PROJECT_ID" t=”INT”> PROJECT _ID</c>
    <c h="Project Name" t="STRING">PROJECT_NAME</c>
    <c h="Project Description" t=”STRING”>PROJECT_DESC</c>
    <c h="Client" t=”INT”>CLIENT_ID</c>
    <c h="Active" t=”String”>ACTIVE</c>
</cc>
  <rr mx="50">
    <r a="78278" b=”Project 1 Name”  c=“Project 1 Description” d=”454” e=”Y”/>
  </rr>
<LOGIN_USER>
  <cc>
```

```
      <c h="LOGIN_USER_ID" t=”INT”> LOGIN_USER_ID</c>
      <c h="User Name" t="STRING">USER_NAME</c>
      <c h="Team" t=”STRING”>TEAM</c>
    </cc>
  <rr mx=”500”>
<r a="4312" b=”John Smith” c=”y” />
<r a="4123" b=” Mary Preston” c=”y”/>
<r a="4532" b="Kevin John" c=” y” />
  </rr>
</LOGIN_USER>
<CLIENT_ID >
   <cc>
      <c h="CLIENT_ID" t=”INT”>CLIENT_ID</c>
      <c h=" Name" t="STRING">NAME</c>
     </cc>
  <rr mx=”500”>
<r a="454" b=”Client 1” />
<r a="455" b=”Client 2”/>
<r a="456" b="Client 3" />
  </rr>
</LOGIN_USER>
   <i>
      <s>IxKz8Ay1kwqv6BUUElZnhZNufnI</s>
      <o>project</o>
      <k>PROJECT_ID</k>
      <t>Project List</t>
      <ps>3</ps>
      <pn>4</pn>
   </i>
</d>
```

Резултатния XML след съответната трансформация връща HTML документ, представен на Фигура 4.2-3



Фигура 4.2-3 – Пример за изглед на екран с детайли

Тъй като системата може да представя отговора на заявка под формата на XML, е възможно приложения-клиенти да използват сървъра като източник на информация. Нещо повече – възможно е сървърът да послужи за основа за разработване на business-to-business приложения.

В някои случаи е възможно преобразуването на първичния XML документ да не се извършва от сървъра. Съществуват браузъри, като Microsoft Internet Explorer версия 5.5 и версия 6.0, които могат да извършат локално желаната трансформация на оригиналния XML документ. Също е възможно да има и клиенти, които да се интересуват само от информативната част на документа и поради което да не желаят форматиране. Затова е необходимо да има механизъм за разпознаване на тези клиенти и за пропускане на процедурата на XSL трансформация.

4.2.2.2 Методи – контролери

Втората група методи водят до извършване на определена операция в базата данни, сървъра на съобщенията или друга, която не изисква връщане на резултат на клиента (web браузера), с изключение на случаите на възникнало изключение. Такива са:

- **Добавяне** (insert) – добавяне на нова единица от този тип в системата;
- **Редактиране** (update) – редакция на вече съществуващ единица;
- **Изтриване** (delete) – изтриване;
- **Изпращане** (send) – изпращането е метод, който е типичен само за някои от обектите в системата, например за съобщение, електронно писмо или шаблон на електронно писмо.

Тези методи приемат като параметър XML документ с определена структура, който съдържа детайлите, необходими за извършване на операцията. Например, за извършване на операцията update, е необходимо XML-а да съдържа новите стойности на полетата в базата данни, името на таблицата и др. Част от XML документа се генерира на клиента и се подава на update обрабочика, който добавя останалата необходима информация, като име на таблицата в базата данни, идентификационен номер на групата и др. Обикновено тя се дефинира от ObjectManager-а за всеки един тип в приложението.

```
<d>
 <i>
   <ss>TTS</ss> <!—Added by ObjectManager→
  <gb>B</gb><!—Added by ObjectManager→
 </i>
 <tt><t gb="">TTS.project</t></tt><!—Added by ObjectManager: ObjectManager.DBTable→
 <ww>
  <w c="PROJECT_ID"  k="">78</w>
  <w c="PROJECT_DESC" >CCE Training Description</w>
  <w c="LAST_UPDATED_BY">2</w></ww><!—Added by ObjectManager→
</d>
```

Генерирането не XML документа се извършва от страна на web клиента (web browser) чрез библиотека от JavaScript функции. За всеки един от методите контролери съществува съответната функция на Jscript, която събира

необходимата информация от клиента (форми или cookies) и генерира XML в подходящ за обработка от методите контролери вид.

4.3 Управление на достъпа до базата от данни. Database pooling and Query Queues

Описания дотук процес на комуникация между web клиента и сървъра изисква събиране на нужната информация от БД за всяка отделна заявка, което предполага повишен мрежов трафик на данни между обработчиците и БД.

Основното правило при увеличаване на мащабируемостта на едно приложение в web е, че намаляването на времето за отговор на една заявка увеличава производителността на приложението. Производителността на приложението не може да бъде увеличавана до безкрай, защото всяка добавена допълнително секунда, то използва за да отговори на заявка, но най-осезаемо тя може да бъде подобрена при оптимизирането на връзката му с базата данни.

Връзката с базата данни може да бъде една от най-времеемките задачи в програмирането за Web. Свързването с базата е операция, която отнема време, дотолкова доколкото базата трябва да задели комуникационни ресурси и ресурси от памет, както и да автентицира потребителя и да се настрои съответното ниво на сигурност. Времето за свързване може да е различно, разбира се, но не са редки случаите, при които процеса може да отнеме секунда или две. Еднократното установяване на връзка и след това многократното и използване може драматично да увеличи бързината на свързаните с база данни web приложения.

Намаляването на времето за създаване на конекция към базата може да бъде реализирано чрез процеса на пулинг на конекциите към базата.

Пул на конекциите към базата данни е съхранено множество от отворени конекции, които могат да бъдат използвани многократно; това спестява времето за създаване и разрушаване им. Това е метод за складиране на оскъдни ресурси на базата от данни, чрез разпределението на множество от конекции между няколко активни клиенти.

Множеството от конекции дава възможност за лесен начин за кеширане на отворени конекции към базата от данни. Това е полезен метод при клиент-

сървърните бази от данни, при които отварянето на нова връзка може да е бавно.

При разработването на системата целия процес за свързване с базата данни и пулинга е изнесен върху отделен DCOM сървър. Той приема като параметър SQL за извършването на стандартните, за работа с база данни, операции, и връща данните, получени в резултат на изпълнение на операцията във валиден XML (Фигура 4.3-1).

Предимствата на това решение са няколко:

- Сървъра може да бъде използван едновременно от няколко приложения, без това да повлияе на производителността му.
- Чрез пулинг-а на заявките се подобрява бързодействието;

Основното, което трябва да бъде направено е, в случая, да се извика функцията GrabConnection в DCOM сървъра на базата, за да се получи връзка и да се извика ReleaseConnection, за да се върне обратно връзката в множеството, при което тя се запазва отворена.

За да бъде пулинга към базата наистина успешен, при първоначалното си зареждане DCOM сървъра отваря определен, предварително конфигуриран, брой връзки към базата от данни.Тези връзки се поддържат непрекъснато отворени, докато DCOM сървъра не бъде окончателно спрян и освободи паметта.

Множеството от връзките към базата се проверява всеки път, когато се създава нова нишка и една връзка към базата бъде отделена за тази нишка. Нишката използва конекцията, за да свърши цялата необходима работа и накрая, когато приключи освобождава конекцията обратно.в множеството. Там връзката изчаква следващата нишка.

В зависимост от типа на базата данни, отварянето на хиляди едновременни конекции от една машина, по принцип е възможно, но е особено препоръчително и ефективно. Ето защо най-добрият вариант е да се изчисли оптималното отношение на милисекундите за един опит за достъп до приложението върху броя на отворените конекции (x). Резултата за x може да е

както 50 конекции, така и 5 или дори повече от 100, в зависимост от хардуерните възможности и възможностите на локалната мрежа. Тоест: максималния брой на отворените конекции е функция на ограничението на броя на едновременните опити за достъп до web приложението.

Нека предположим, че оптималното ограничение при предварително зададена хардуерна среда е 15 кеширани конекции. В този случай - 16-ия конкурентен опит за достъп ще върне на потребителя грешка. Тя може да бъде просто съобщение, че системата в момента е претоварена и той трябва да опита отново по-късно. Често в такава ситуация изпадат някои сайтове за онлайн. банкиране или резервация на полети, например. Но в идеалният случай, е желателно да се даде възможност за едновременен достъп на стотици конкурентни потребители до един хардуерен сървър.

В случая цифрата 15 означава, че са позволени само 15 конекции имат право на достъп до базата от данни в едно и също време. Като се има предвид, че това са едновременно опити за достъп до базата, би било основателно да се предположи, че по-краткото време за използване на конекцията ще допусне по-голям брой потребители до услугата. По-дългото време в което всяка една от тези 15 конекции е заета, води до намаляване на възможността останалите потребителя да я използват. Това може да бъде изразено със следната приблизителна алгебрична зависимост:

$s = 1/(t^4)*n$

където:

- s е коефициент на машабируемост;
- n е максимален брой на конекции към базата в множеството;
- t е времето за което се използва всяка от конекциите.

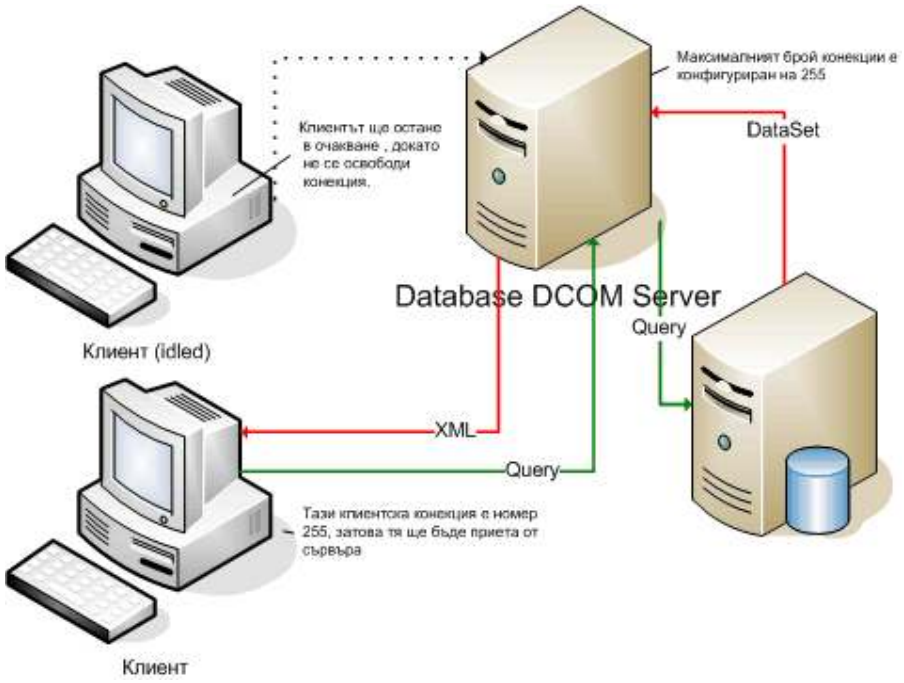
От това уравнение следва, че ако се намали броя на конекциите в множеството или се увеличи времето, за което се използва една конекция – това би довело до намаляване на мащабируемостта на приложението – т.е. намаляване на броя на потребителите, които могат да го използват

едновременно. Също би могло да се отбележи, че най голямо влияние върху мащабируемостта на приложението има времето t, за което то отговаря на една всяка една конекция. Ако това време може да се задържи под една секунда, това може да доведе до чувствително подобрение на изпълнението.

Кеширането на отворени конекции към базата е голямо подобрение на изпълнението и е препоръчително за повечето web приложения. В идеалния случай, приложението би могло да има стотици (дори хиляди) конкурентни потребители, без да се налага добавянето на допълнителни хардуерни сървъри.

Друг елемент, който подобрява изпълнението на DCOM сървъра за достъп до базата от данни и допълва концепцията за съхранено множество от отворени конекции е опашката от заявки.

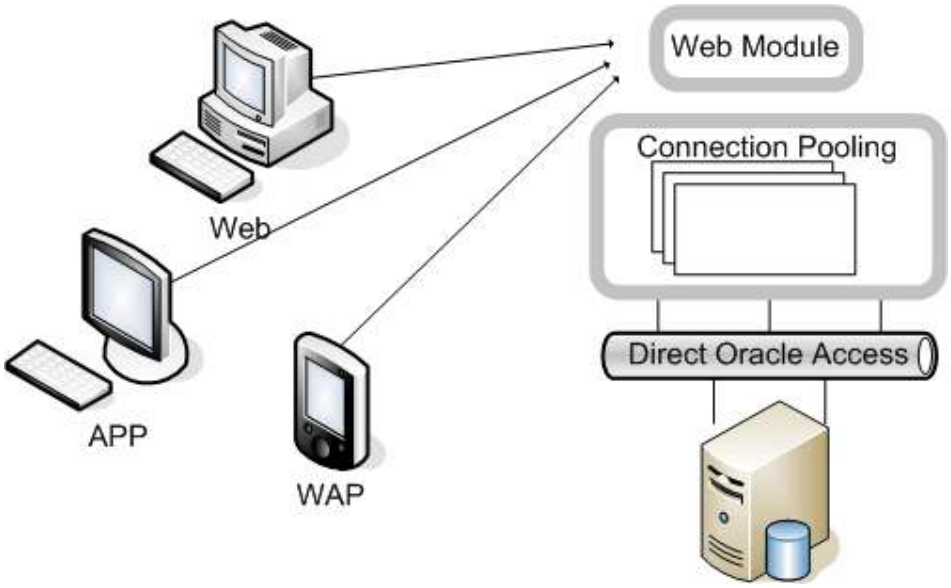
Опашката от заявки към базата от данни, не е нищо повече от запазено място за заявки, които не могат да бъдат изпълнени докато не се освободи отворена конекция към базата от пулинга. При освобождаването на конекция заявките се изпълняват по реда на влизането им и съответно това води до изпразване на опашката. Това е добър начин да се увеличи броя на потенциалните конекции и да се поеме грижата заявките да бъдат изпълнени, при случайни претоварвания на сървъра.



Фигура 4.3-1 – DB DCOM сървър и пулинг на конекциите към базата от данни

Технологията за съхранение на отворени конекции към базата работи оптимално добре, ако времето за изпълнението на SQL-а се поддържа максимално кратко. В този смисъл добре е заявките да са максимално оптимизирани - няма значение колко е спестено от времето за свързване с базата от данни, ако за изпълнението на всяка заявка се губи време от порядъка на 10 секунди. Едно от задължителните условия, е предварително тестване на всеки от SQL-ите с подходящи за целта инструменти.

Друга използвана оптимизация е параметризирането на всеки SQL. Тоест, динамичната част от SQL-а да се подава не като част от целия стринг, а като отделен обект от тип параметър. Това дава възможност на сървъра на базата данни да съхрани в кеша си структурата на всеки един от SQL заявките, което спестява един етап от протичането на процеса на изпълнение за заявката.



Фигура 4.3-2 – Архитектура на взаимодействието с базата от данни

Данните се съхраняват на Oracle 9i сървър. За достъп до него се използват Direct Oracle Access компонентите. Тези компоненти, както би могло да се предположи от името, дават директен достъп до интерфейса на Oracle. Това гарантира оптимална производителност при изпълнението на стандартните функции за достъп до базата.

4.4 Проектиране на XSL трансформациите

В приложението XSL трансформации се извършват на две места от обработката на клиентската заявка.

1. На входа на DCOM сървъра на базата данни, се подава XML от методите обработчици, които съдържа данните, които трябва да бъдат обновени или извлечени от базата. Входният XML се трансформира със съответната XSL трансформация (generate_insert_sql.xsl, generate_update_sql.xsl или друг) до SQL заявка, която съответно се подава за изпълнение на базата от данни чрез DCOM сървъра за достъп до базата данни.

2. Резултат от методите - изгледи е отново XML документ, който се трансформира до html или wml, в удобен за разглеждане вид. За тази цел на

всеки метод - изглед съответства XSLT трансформация, която е статична и е записана във файл на диска. Името на файла се явява низ от вида:

/ “име на метод изглед” “име на обект”.xsl”

Например: listtravel_profile.xsl

Всички XSL документи, които се отнасят за един и същи метод, имат сходна структура и логика. Разликите между тях са свързани предимно с разположението на елементите (напр: бутони, текстови полета) върху html документа- резултат от трансформацията.

Трансформациите се извършват от отделен DCOM сървър – XSLTransfom Server. За реализацията им се използва MSXML 4.0. Подобно на логиката, по която работи DCOM сървъра за връзка с базата от данни, предварително в паметта на сървъра се създава множество от MSXML2 DOMDocument OLE обекти , необходими за извършване на XSL трансформациите. По този начин се спестява времето за създаване и инициализиране на нов OLE обект и се оптимизира цялостния процес по трансформиране на XML документа.

4.5 Управление на сесиите

Управлението на сесиите съдържа техниките, чрез които web приложението може прозрачно да оторизира потребителя за всяка HTTP заявка без да се налага той да дава всеки път своето потребителска име, име на група и парола.

Управлението на сесиите трябва да бъде извършено на приложението защото лежащият отдолу HTTP протокол не разполага с такава функционалност. Това изисква приложението да изпраща на клиента сесиен маркер след успешно удостоверяване на неговата автентичност. Този маркер трябва да бъде връщан от клиента на сървъра при всяка HTTP заявка за да се идентифицира. Тогава приложението може да установи дали клиента има права за изисканата от него информация.

По принцип, механизмите за управление на сесиите могат да бъдат разделени на такива от страна на клиента и на сървърни механизми, в

зависимост от съдържанието на маркера, предаван между клиента и web приложението.

Процесът по управление на сесиите в приложението за управление на персонала е изнесено на сървъра. Създаден е сървър за управление на сесиите с разширена функционалност. След успешен логин на потребителя той създава уникален идентификатор на потребителската сесия, която връща обратно на киента. За да се идентифицира, клиента трябва да го добави в стринга на всяка от заявките си до сървъра.

Сървърът на сесиите съхранява данни за всички активни потребители и дава достъп до детайлна информация за всеки от тях. Чрез публичния му интерфейс, приложението има достъп до следната функционалност:

- инвалидиране на потребителска сесия (*InvalidateWebSession(SessionString,SessionExists)*);
- създаване на нова потребителска сесия (*NewWebSession(SessionLookupKey, SessionData, SessionPrivileges, SessionDataFilters,)*. При създаването на нова сесия, част от информацията която може да бъде съхранена в нея е: уникално име за сесията (може да бъде име напотребител), уникални данни за потребителя, които правят по-удобно или по бързо работата с приложението и привилегиите на потребителя, представени в XML формат,
- обновяване на потребителска сесия (*UpdateWebSession(const SessionString, UserPrivileges: WideString; out SessionExists)*), налага се в случаите, когато например потребителя е получено ново офлайн съобщение или е започнал да работи върху нова задача.
- Търсене на сесия по зададено име *WebSessionLookup(const SessionLookupKey: WideString; out SessionString)*

4.6 Комуникации – електронна поща, системи за съобщения

При реализиране на web приложение от подобен тип, при който потребителите се намират на различно географско местоположение сигурната и многофункционална система за връзка между тях е особено важна.

Възможностите за комуникация между потребилите, които предлага приложението са няколко:

Система за обмен на офлайн съобщения

Създадена е система, която позволява изпращането на кратки текстови съобщения от един потребител на друг или на група от потребители. Всеки потребител има достъп до кутия за входящи и изходящи съобщения. Информацията за новополучени съобщения се пази на ниво потребителска сесия, което позволява получателя да бъде информиран в момента на изпращане на съобщението. В горният десен ъгъл на клиентския интерфейс се появява мигащ индикатор за ново (нови) съобщения. Всяко съобщение се състои от тема, текст, списък на получатели и индикатор за важност.

Email broadcasting (разпространяване на електронна поща)

За улеснение на разпространението на електронната поща е създаден специален обект на системата – шаблон за електронна поща. Той позволява създаването на различни шаблони и асоцииране им с отделни роли или отделни потребители. Всеки шаблон съдържа основните компоненти на един стандартен емайл: подател, тема, съдържание.

Има възможност за добавянето на динамична информация към шаблона, като например: име на потребителя, паролата му и т. н., чрез вмъкването в стандартния текст на тагове със специален формат.

Освен стандартните шаблони се използват и така наречените системни шаблони. Това са съобщения, които се изпращат на предварително избрани потребители при случване на определено събитие. Например, когато служител направи промяна информацията за своята социална осигуровка, служителят от отдел „Човешки ресурси” бива своевременно уведомен.

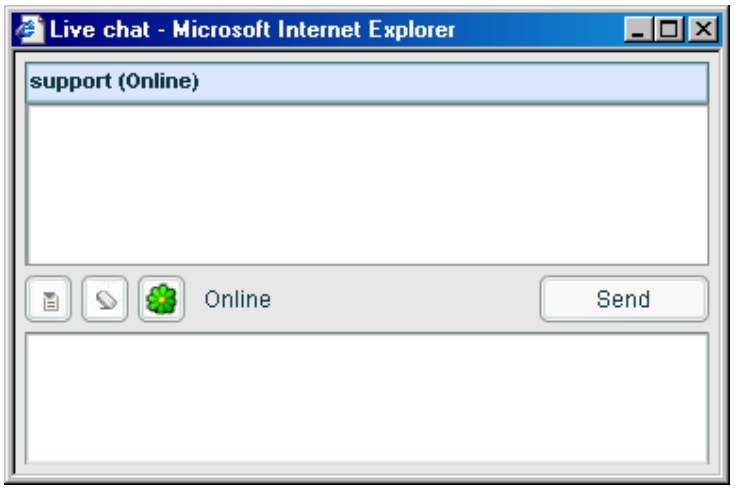
Функцията по изпращане на електронна поща отново е реализирана чрез самостоятелен DCOM сървър с цел – подобряване на продуктивността и спестяването на ресурси.

Система за обмен на съобщения в реално време.

Системата за обмен на съобщения в реално време се използва основно за поддръжка и помощ на потребителите на системата, които имат проблеми с

използването и. Реализирана е изцяло на базата на Jabber протокол. За тази цел са създадени flash базиран клиент и Jabber DCOM сървър.

Клиентът е реализиран чрез технологията на Macromedia: flash и action script (Фигура 4.6-1).



Фигура 4.6-1 – обмен на съобщения в реално време

Причините за избора и са няколко:

- Macromedia Flash, макар и отскоро позволява включването на активни елементи – бутони, текстови полета и др.
- Тя има относително добра поддръжка на XML формата, който стои в основата на приложението, и позволява изпращането на XML без презареждане на HTML документа. Последното е от ключово значение при реализиране на приложения за обмен на съобщения в реално време.

Технологията има един единствен недостатък за целите, за които е избрана: тя изисква инсталиране на Flash Player Plug-in за браузера. Но проблемът не е фатален тъй като основните браузъри в масовите си версии го инсталират по подразбиране. Възможно е обаче да се наложи надграждане (upgrade) на инсталираната версия, което пък, обаче, става безболезнено и лесно – от потребителя се изисква единствено да даде съгласие.

5 Примерен сценарий за използване на системата

Сценариите демонстрират някои ключови способи, по които може да бъде използвана програмата. Те представляват описание на взаимодействието на потребителите със системата от тяхна гледна точка.

5.1 Видове потребители на системата, дефиниция на роля и група. Базови роли.

Едно от ключовите понятия при използването на системата е ролята на потребителите в нея. Всеки потребител на системата може да има една единствена роля и оттам той притежава дефинираните от нея права за четене и редактиране на данните. Между различните видове роли съществува йерархична зависимост – т. е. всяка една от тях има правата на ролята под нея в йерахията, както и допълнителни, специфични само за нея. В системата съществуват няколко базови роли, като съществува възможност за създаване на нови роли, които наследяват правата от някоя от базовите.

Основните роли на потребителите в системата са:

Служител (Employee) – базова роля в системата – служители са всички хора, които имат сключен договор с агенцията за временно наемане. Дейностите и правата на всички останали роли са свързани с данните въведени от / за всеки служител. По принцип служителя има право за въвеждане на данни, свързани изключително с него – часове, разходи, задачи. След приключване на въвеждането той може да изпрати данните към следващата роля в йерархията;

Супервайзор (supervisor) – супервайзора обикновено е човек, служител на клиентската фирма, който носи отговорност за коректността на всички въведени от служителя данни. Негово задължение например, периодично, през предварително дефиниран в системата период от време (payroll period), да отбележи като одобрени или не въведените от служителя часове по проекти или задачи, или направените от него разходи. Супервайзора има и ограничени права за управление на ролите: той може да създава нови потребители, които наследяват неговите права: т. нар. суб-супервайзори.

Тоест – той може да делегира права за одобрението на определен служител или служители работещи под определен проект на свой подчинен.

Суб–супервайзор (sub-supervisor): има подмножество от правата на главния супервайзор за клиента Одобрява извършени разходи/часове за служител или проект.

Наблюдател (Observer): наблюдателят също е служител на фирмата клиент. Той няма пряко отношение към финансовата част от взаимоотношенията между трите субекта. Обикновено е ръководител на проект/проекти във фирмата клиент, и следи вложените човекочасове по проекти и задачи за период – обикновено месец.

Служител отдел Човешки ресурси (HR) задълженията му са свързани предимно с финансовата част – осигуровки, заплащане, покриване на разходите, фактуриране и пр. Обикновено има достъп до цялата социално-осигурителна информация на служителя по текущия му договор.

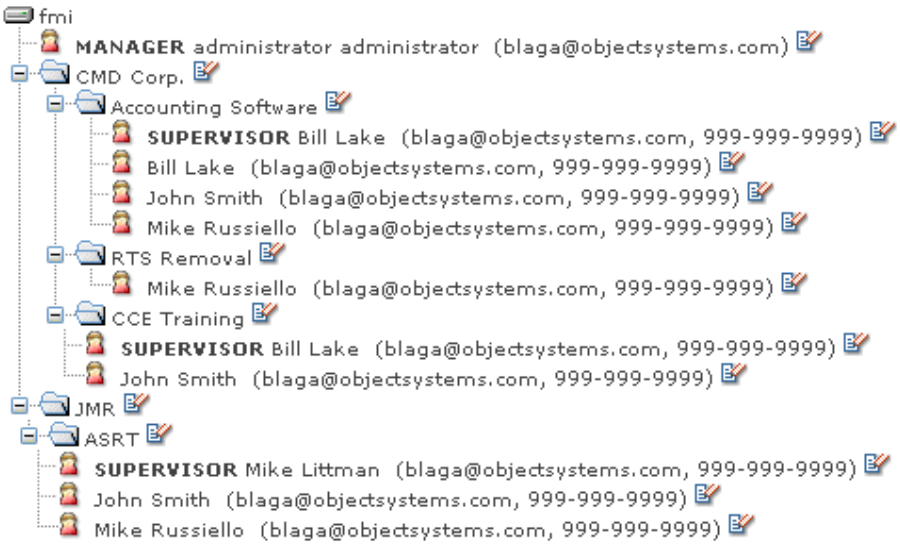
Администратор (Manager): Администраторът, както се подразбира, има права правата на всички гореизброени роли , включително права които му позволяват да управлява потребителите, ролите, да създаде базова конфигурация на системата и др.

Основният момент, който трябва да се отбележи е, че, правата на всички дотук изброени роли се отнасят само до собствената им група. Системата поддържа използването и от няколко независими една от друга групи, всяка от които може да се конфигурира различно, в зависимост от специфичните и особености. Това, логично, води до съществуването на още една роля:

Администратор на групите:

Основните права на потребителите с тази роля са свързани с създаването на нови групи, модифициране на съществуващите и т.н.

Йерархията на потребителите в една група, групирани по клиенти и проекти, може да бъде разгледана в детайли на „Administration/Relations” опцията на главното меню (Фигура 5.1-1).



Фигура 5.1-1 – Изглед на йерархията на потребителите, групирани по клиенти, проекти и роли.

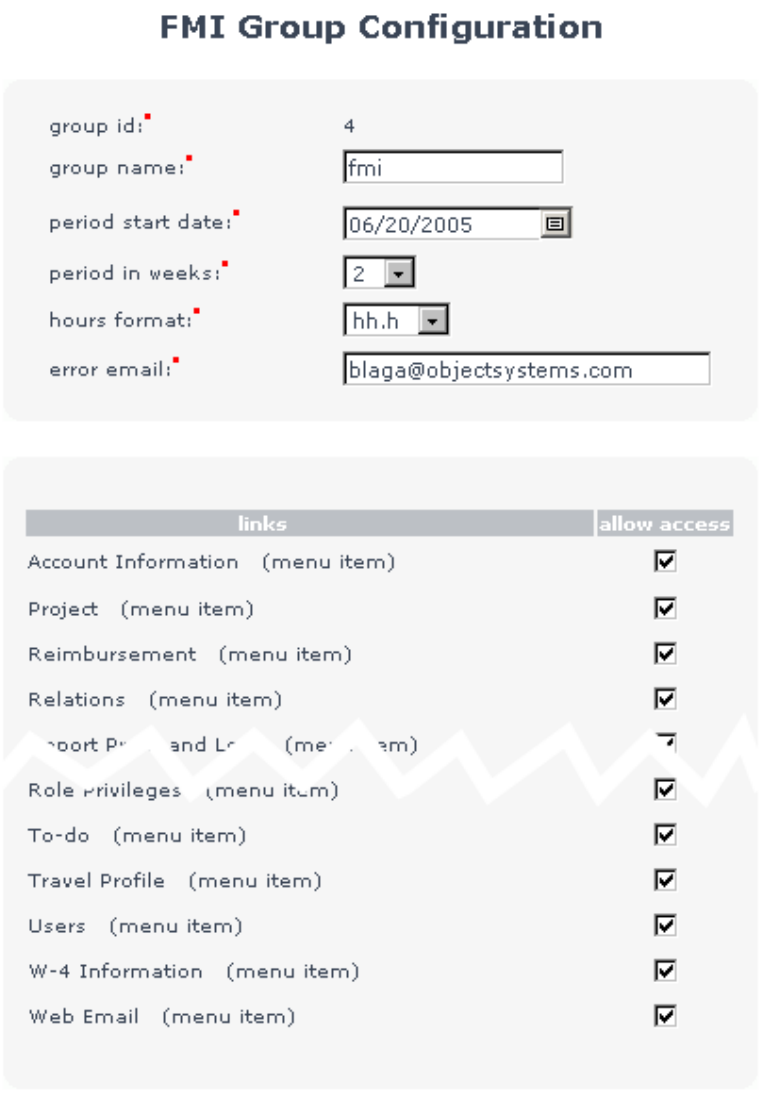
5.2 Конфигуриране на група

Конфигурирането на група преминава през няколко основни стъпки:

Създаване на нова група. За тази цел потребителят трябва да има права на администратор на групи. Става чрез избор от менюто на Administration/Groups и бутона „Add New Group”, попълват се основните конфигурационни полета:

- Име на групата;
- Стартова дата - въз основа на стартовата дата се преизчисляват периодите за осчетоводяване на данните;
- Дължина на периода за осчетоводяване в брой седмици.

От зададен списък се избират обектите, до които потребителите на групата имат достъп, например: expense statement, invoice, to-do, IRA contribution, reimbursement и др., в зависимост от целите за създаването и (Фигура 5.2-1).



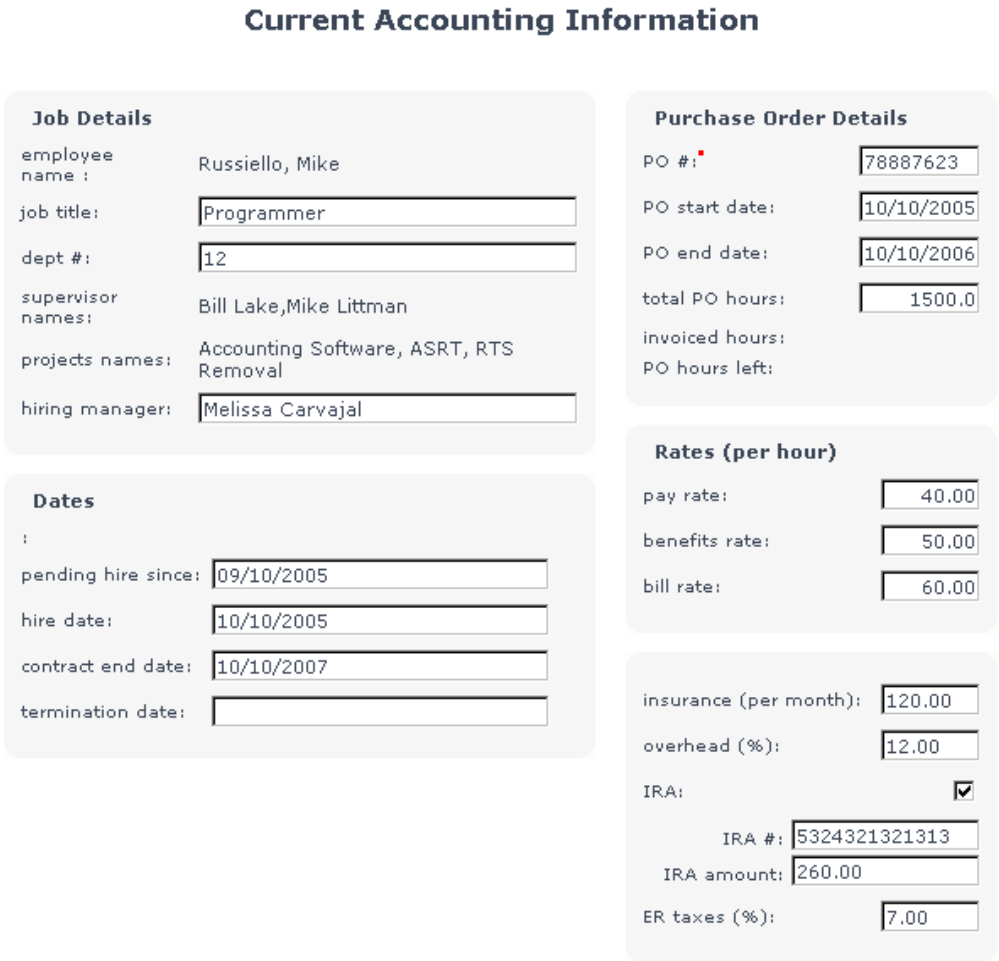
Фигура 5.2-1 – Конфигуриране на група

Конфигуриране на базовите данни се извършва на ниво администратор на група. То включва: въвеждане на потребители от всички нива: служители, супервайзори, наблюдатели. Системата не предлага възможност за публична регистрация на потребители. При създаването на нов потребител той получава емейл с данните, които може да използва, за да влезе в системата и произволно генерирана парола. Следва въвеждането на всички клиенти и съответните им проекти, с които работи фирмата или отдела. За всеки проект се създава екип, които работи върху него.

Отделна стъпка е създаването на счетоводен профил (Accounting Info) за всеки един от въведените служители. Той представлява набор от задължителни показатели, по-важните от които са:

- длъжност на служителя (Job Title)
- номер на отдел (Department Num)
- пряк ръководител на служителя (Hiring manager)
- данни, свързани с текущия договор за назначение, по който служителят работи: стартова дата, дата на приключване, брой часове по договор, договорените суми за един изработен час, застраховки, данни за социалното осигуряване (IRA Num, IRA Amount, ER Taxes) и др.

Всяка промяна на данните от счетоводния профил да служителя записва копие на старата информация в архива на системата.



Фигура 5.2-2 – Счетоводен профил на служител

Освен счетоводен профил, всеки от служителите регистрирани в системата има и пътнически профил (Traveling Profile) и профил (W-4 Profile), в който се съхраняват данните, свързани със социалната му осигуровка. При промяна на някой от тях, в зависимост от конфигурацията на групата, администраторът или друго оторизирано лице, получва електронна поща, известяваща за промяната в информацията.

Съществува възможност за конфигуриране на правата за всяка една от базовите роли на системата. Извършва се от меню Adminisration/Accounting Privileges. Там детайлно за всяка една от ролите може да бъде определен типа на достъп (Редактиране, Изглед или без достъп), за всяко поле от обектите. Ако същността на групата го изисква могат да бъдат създадени нови роли, които наследяват правата на някоя от съществуващите, например, могат да бъдат създадени супервайзори с права да създават нови задачи към назначен им проект.

По желание на администратора допълнително могат да бъдат създадени шаблони за електронна поща или конфигурирани съществуващите системни такива. Като пример, може да бъде създаден шаблон, който съдържа напомняне на служителите да изпратят часовете си за одобрение в края на разплащателния период или на супервайзорите – да одобрят изпратените им часове.

5.3 Запис на нови часове

Основният обект на системата е работен час. Права да въвеждат работното си време имат потребителите от почти всички роли. Работните часове представляват времето, през което един потребител е работил върху възложен му проект на определена дата. Въвеждането на работното време може да бъде извършено по няколко начина:

- Чрез Hours Worked опцията на менюто. Излиза списък с дните от текущия разплащателен период по проекти и възможност за въвеждане на продължителността на работното време и типа му. Типа може да бъде стандартен, болничен, отпуска или празник. За улесняване на навигация между периодите в горната част на екрана има календар с последните три месеца. След успешно въвеждане на работното време за целия

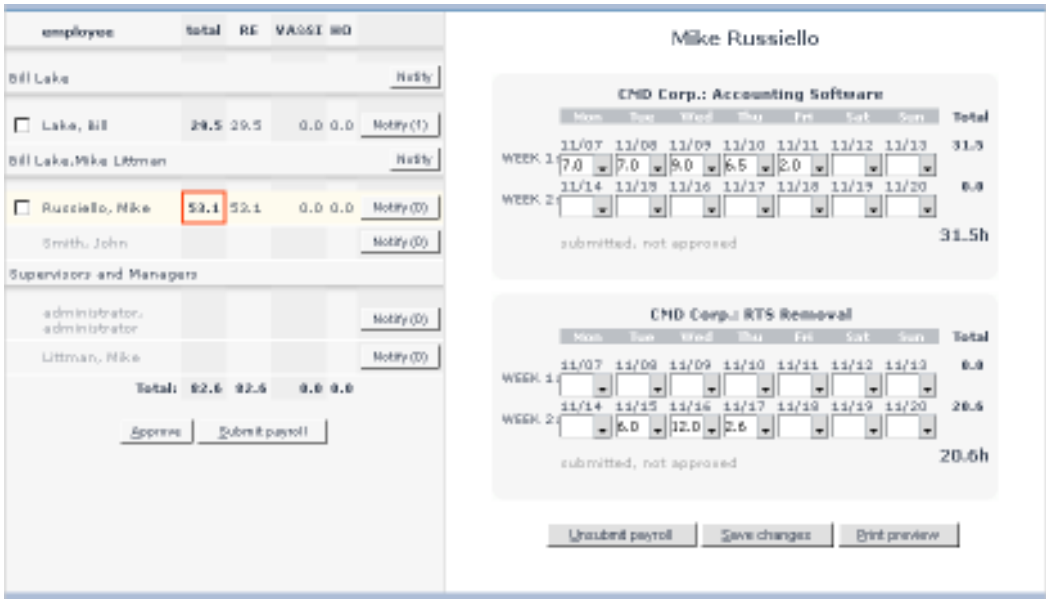
разплащателен период той трябва да бъде изпратен за одобрение към супервайзорите на проектите чрез натискане на бутона: „Send for approval”.

- Друга възможност за въвеждане е формата “Quick Entry Bar”, разположена най-отгоре на същата страница. Тя позволява директно въвеждане на датата, работното време и проекта.
- Ако потребителят има възложени задачи, системата може автоматично да запише времето, за което той работи по тях. На списъка със възложените му задачи (To-do/Search To-dos/Search) вдясно той може да открие бутон „Check out”, след натискането който се сменя на „Check in”. Системата автоматично записва времето до натискане на „Check in” бутона за съответния проект.

Приложението дава възможност потребителя да настрои екрана за въвеждане на часове в зависимост от предпочитанията си и текущите му задължения. Той може да посочи кои от проектите би искал да се скрият или дали да се показва формата за бързо въвеждане на работното време или не. Тази функционалност е достъпна от опцията на менюто „Display Preferences”.

5.4 Одобрение и фактуриране на часовете

Одобрението на часовете може да бъде извършено от супервайзорите на съответния проект, оторизираните от него суб-супервайзори и мениджъра на групата. На екранът за въвеждане на часовете („Hours Worked”) всички потребители с права за одобрение имат списък със служителите, които са въвели часовете си за периода, групирани по супервайзори. Всеки от тях има поле за отметка пред името си. Името на всеки потребител е и линк за детайлна информация за въведените от него часове. Пред името на всеки от служителите има бутон за изпращане за напомнящ емейл („Notify”), в случай, че не са изпратили часовете си на супервайзора. С натискането на бутона „Approve” часовете са одобрени и вече могат да бъдат фактурирани от мениджъра (Фигура 5.4-1).



Фигура 5.4-1 – Екран за одобряване на въведеното работно време

Единствени потребителите с права на мениджър във агенцията за временно назначение имат права за създаване на фактури върху въведените и одобрени работни часове или авансови отчети. Фактурирането е извършва от опцията на менюто: „Accounting/Invoicing”. След избирането и се зарежда списък със всички потребители, които имат одобрени часове за избрания разплащателен период. По подразбиране се зареждат данните за последния завършен период.

Всяка фактура зарежда по подразбиране данните от счетоводния профил на потребителя. По-важни от тях са: дата на фактурата, период, номер на договор, брой платени часове, цена за един час и др. Системата предлага възможност за едновременно фактуриране на часовете на няколко потребителя.

Обобщената информация за издадените от фирмата фактури и изработените часове по потребители и проекти може да се види в структуриран вид в отчетите, които предлага системата. Те могат да бъдат открити в „Reports” опцията на главното меню.

По аналогичен на гореописания начин протича и процеса на работа с авансовите отчети в приложението, отчитането на дължимите и изплатени социални осигуровки и др. Освен това съществува възможност за създаване на

пътнически профил на служителя и изпращането му към web приложението на пътническата агенция, с която фирмата има договор.

6 Заключение

В съвременните условия на бърз растеж, променящи се технологии и ограничени ресурси решенията за оптимизация могат да дадат изключителни резултати при повишаването на удовлетвореността и продуктивността на персонала, спестяване на разходи и ефективност на операциите. Новите технологии и стандарти също така дават възможност за усвояване на повече физически ресурси с цел обслужване на повече потребители едновременно, без това да намали драстично производителността на системата.

Средствата за оптимизация на работната сила предоставят тактически и стратегически предимства както за работодателите, така и за персонала:

- Намалени разходи: решенията за оптимизация на работната сила опростяват и автоматизират рутинните административни задачи. Преместването на ръчните процеси онлайн. и въвеждането на средствата за извършване на сделките, така че да бъдат обработвани от хората, които са ги инициирали, намалява нуждата от междинен персонал. Също така става възможно да се поддържа по-нисък брой на персонала въпреки растежа на бизнеса и компанията.
- Помощ за персонала: при използването на Web базирани решения за управление на човешките ресурси във TSI, може да се получи своевременна и точна информация независимо от местоположението на потребителя. Специалистите, които разполагат с информация, указания и възможности за вземане на решения, могат да реагират динамично. Така може значително да се подобри взаимодействието с клиентите, като проблемите се решават интелигентно и своевременно, вместо да се търси помощта на хора на по-високи позиции.
- Повишена продуктивност: средствата за самообслужване помагат на персонала да набира информация и позволяват по-лесно да се обработват сделките в Web базираната инфраструктура. Тези средства могат да се използват, за да се решават проблеми, свързани с човешките ресурси, или да се извършват плащания, без да се налагат специални срещи или прекалено много

работа в счетоводния отдел. При въвеждането на технологии (вместо на хора), за да се контролират тези процеси, могат да се забележат предимствата, които те предлагат - под формата на намалени административни разходи, повишено удовлетворение на персонала и по-бързи и по-успешни сделки.

В съвременната конкурентна икономика компаниите трябва да осигурят средствата, от които се нуждаят нейните специалисти, за да бъдат продуктивни, ефективни и информирани. За да оптимизират възможностите на компанията и персонала, предприемчивите мениджъри инвестират в Web базирани решения за оптимизация на работната сила. Тези решения включват обширен комплект от приложения и процедури, които улесняват персонала при извършването на рутинни административни. Също така те улесняват достъпа до информация и подобряват. Всичко това става възможно чрез една интегрирана, сигурна, глобална мрежа.

7 Използвана литература

1. Flexible recession: the temporary staffing industry and mediated work in the United States - Jamie Peck, Nik Theodore, Department of Geography, University of Wisconsin-Madison, USA & Center for Urban Economic Development, University of Illinois at Chicago, USA, November 24, 2004
2. Staffing Industry Analysts 2002b. Top 16 firms capture 29% of U.S. temp and perm market, Staffing Industry Report, vol. 13, 8
3. Theodore, N. and Peck, J. 2002. The temporary staffing industry: growth imperatives and limits to contingency, Economic Geography, vol. 78, 463-493
4. Vosko, L. F. 2000. Temporary Work: The Gendered Rise of a Precarious Employment Relationship, Toronto, University of Toronto Press
5. Wall Street Transcript 2000. Special Focus: Staffing Industry, February 14,http://www.twst.com
6. "Extensible Markup Language (XML) 1.0 (Second Edition)", W3C Recommendation, 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006
7. "XSL Transformations (XSLT), Version 1.0", W3C Recommendation, 16 November 1999, http://www.w3.org/TR/1999/REC-xslt-19991116

8. Burbeck S., Ph.D., “Applications Programming in Smalltalk-80(TM):How to use Model-View-Controller (MVC)”, 1987, 1992

9. HTML 4.01 Specification - W3C Recommendation 24 December 1999

8 Приложения

Приложение А: Основни термини и понятия

Агенция за временно назначаване (Temporary Staffing Agency) - Агенцията за временно назначаване е фирма, с която са сключили договор хора, търсещи временна работа. Тя има определен брой клиенти – фирми, които ползват услугите и при наемането на временни служители. Договор за наемане се сключва, от една страна, между фирмата-клиент и агенцията и от друга – между агенцията и хората, които търсят работа. За разлика от агенциите за подбор на персонал, агенцията за временно наемане поема грижата за социалното и здравно осигуряване, застраховки и др. за хората сключили договор с нея.

Timesheet - Timesheet е документ, който съдържа записите на работното време на служителите, той е колекция е от индивидуални разплащателни транзакции. Това е един от основните документи при временните трудови договори. Предназначението на системата е да записва действителното работно време за един ордер или проект. Timesheet може да съдържа данни за операцията, местоположението или типа на изпълняваната задача.

Разплащателна ведомост (Payroll) - Разплащателната ведомост е колекция от timesheets, със една и съща стартова дата или за едно за и също множество от дати. Представлява финансов запис за заплатата на служителя или неговите бонуси, удръжки или надници. Тя съдържа записи, детайлизирани относно заплатите, надбавките, удръжките за всеки един служител за специфичен период. Разплащателната ведомост съдържа комисионните, бонусите, заплащането за извънработно време, за празници или болнични.

Разплащателната ведомост включва индивидуално изчислението на сумите за всеки служител –почасови ставки, заплати, базирана на определен период през календарния месец , комисионни, изчисляеми на база на

извършената работа, както и възстановяваните суми по направените от служителя служебни разходи – за пътувания и други.

Приложение Б: JabberChat DCOM Server: Component Type Library

```
unit JabberChatServer_TLB;
{$TYPEDADDRESS OFF} // Unit must be compiled without type-checked pointers.
interface
uses Windows, ActiveX, Classes, Graphics, OleServer, OleCtrls, StdVCL;
const
 JabberChatServerMajorVersion = 1;
 JabberChatServerMinorVersion = 0;
 LIBID_JabberChatServer: TGUID = '{411461ED-516F-48DF-ABCE-3CD7EC371803}';
 IID_IJabberServer: TGUID = '{44963E6A-3251-4B36-B950-48CFAA22FFDD}';
 CLASS_JabberServer: TGUID = '{A48F1BD7-7867-4AE0-A349-1E47D75748B3}';
type

// *********************************************************************//
// Forward declaration of types defined in TypeLibrary
// *********************************************************************//
 IJabberServer = interface;
 JabberServer = IJabberServer;
 IJabberServer = interface(IUnknown)
  ['{44963E6A-3251-4B36-B950-48CFAA22FFDD}']
  function  Send(MaxWait: Integer; const FromUser: WideString; const ToUser: WideString;
            var MessageText: WideString; const MessageType: WideString;
            const GroupChatDisplayName: WideString): HResult; stdcall;
  function  Connect(MaxWait: Integer; const UserName: WideString; const Password: WideString;
              const Resource: WideString; out OutMessage: WideString): HResult; stdcall;
  function  Disconnect(const UserName: WideString; out OutValue: WideString): HResult; stdcall;
  function  GetLastMessages(MaxWait: Integer; const UserName: WideString;
                    const StatusForUser: WideString; out MessagesXML: WideString): HResult;
stdcall;
  function   GetRosterData(MaxWait: Integer; const UserName: WideString; out RosterXML:
WideString): HResult; stdcall;
 end;
 CoJabberServer = class
  class function Create: IJabberServer;
```

```
    class function CreateRemote(const MachineName: string): IJabberServer;
  end;
implementation
uses ComObj;
class function CoJabberServer.Create: IJabberServer;
begin
  Result := CreateComObject(CLASS_JabberServer) as IJabberServer;
end;
class function CoJabberServer.CreateRemote(const MachineName: string): IJabberServer;
begin
  Result := CreateRemoteComObject(MachineName, CLASS_JabberServer) as IJabberServer;
end;
end.
```

Приложение B: WebModule.pas (извадка)

```
unit WebModule;
interface
uses
  Windows, SysUtils, Classes, HTTPApp, Math, LogWriter_osi, ObjectManager, DzURL,
  IdHTTP, IdIntercept, IdSSLIntercept, IdSSLOpenSSL, IdTCPClient, IdBaseComponent, IdComponent,
IdTCPConnection;
type
     TWM = class(TWebModule)
    IdHTTPS: TIdHTTP;
    IdConnectionInterceptOpenSSL: TIdConnectionInterceptOpenSSL;
    MainPageProducer: TPageProducer;
    ObjectManager: TObjectManager;
    procedure WebModuleBeforeDispatch(Sender: TObject;
      Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
    procedure WebModuleCreate(Sender: TObject);
    procedure WMWebActionItemDetailAction(Sender: TObject;
      Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
……..
var
  WM: TWM;
implementation
{$R *.DFM}
uses
  ComObj, DataBaseServer_TLB, MailServer_TLB, WebSesServer_TLB,
```

```
  XSLTServer_TLB, zServer_TLB, JabberChatServer_TLB,
  MsMultipartParser;
var
  { COM interface pointers }
  ARS:IDataBaseServer;
  EMS:IEmailServer;
  WSS:IWebSessionsServer;
  XSLS:IXSLTransformServer;
  JS:IJabberServer;
{================================================}
{                    WebModule Event Handlers                    }
{================================================}
……..
procedure TWM.WMWebActionItemDetailAction(Sender: TObject;
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean);
var
  XML, XSL, Query, ObjectID, ObjectName: String;
  I, PageNumber :Integer;
  UseDefaultSQL: Boolean;
begin { /detail }
  try
    XML:= EmptyStr;
    OutValue:= EmptyStr;
    ObjectName:= GetRequestValue('o');
    ObjectID:= GetRequestValue('oid');
    UseDefaultSQL:= False;
     Query:=Trim(PO.DetailSQL.SQL.Text);
     case PO.DetailSQL.Param of
       pObjectID: Query:=Query+#127 + 'oid=' + ObjectID;
       pGrID: Query:=Query+#127 + 'grid=' + GrID;
       pUserID: Query:=Query+#127 + 'usid=' + UsID;
     end;
     try
       COMResult:=ARS.Detail(MAX_WAIT, StrToInt(GrID),  -1, Query, OutValue);
       CheckComError(COMResult, OutValue);
       XML:=OutValue;
      except
       on E: ENoDataFound do UseDefaultSQL:=True;
     end;
```

```
for I:=0 to PO.OptionLists.Count-1 do
with PO.OptionLists[I] do
begin
 if (PO.OptionLists[I].Name <> 'default') or UseDefaultSQL then
 begin
  Query:=Trim(FQueryItem.SQL.Text);
  case FQueryItem.Param of
   pObjectID: Query:=Query+#127 + 'oid=' + ObjectID;
   pGrID: Query:=Query+#127 + 'grid=' + GrID;
   pUserID: Query:=Query+#127 + 'usid=' + UsID;
  end;
  COMResult:=ARS.List(MAX_WAIT,StrToInt(GrID),-1,Query,0,OutValue);
  CheckComError(COMResult, OutValue);
  if PO.OptionLists[I].Name='default' then XML:=XML+OutValue
  else XML:=XML+'<'+PO.OptionLists[I].Name+'>'+ OutValue
+'</'+PO.OptionLists[I].Name+'>';
 end;
end;
    XSL:=LoadFile('detail'+ ObjectName+DOCEXT_XSL);
if PO.SecurityOn and (GrID<>'-1') then
begin
 COMResult:=ARS.Privileges(MAX_WAIT,StrToInt(GrID), PrID, PO. SecurityName
,Query,0,OutValue);
 CheckComError(COMResult, OutValue);
 XML:=XML+'<privileges>'+ OutValue +'</privileges>';
end;
  GetPageNumber(PageNumber);
 XML:='<?xml version="1.0"?><d><i><s>'+SessionID+'</s><oa>' +PO.Alias + '</oa>'+
  '<sti>'+Request.QueryFields.Values['sti']+'</sti>'+
  '<gd>'+FormatDateTime('mm/dd/yyyy',GroupStartDate)+'</gd>'+
  '<gp>'+IntToStr(GroupPeriod)+'</gp><gn>' + GroupName + '</gn>'+
  '<ghf>'+GroupHourFormat+'</ghf>'+
    '<ro>'+RoID+'</ro>'+
    '<pn>'+IntToStr(PageNumber)+'</pn><ps>'+IntToStr(LIST_PAGE_SIZE)+'</ps>'+
 </i>'+XML;
  XML:=XML+'</d>';
COMResult:=XSLS.Transform(MAX_WAIT,XML,XSL,OutValue);
CheckCOMError(COMResult,OutValue);
OutHTML:=OutValue;
```

```
  except
    on E:EObjectManager do
      OutHTML := ProcessError('detail', 'Unable to load a Privilege Object. ' + E.Message);
    on E: Exception do
      OutHTML := ProcessError ('detail'+ ObjectName, E.Message);
  end;
  Handled:=True;
end; { /detail }
```