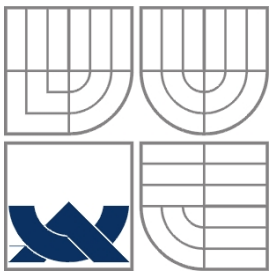
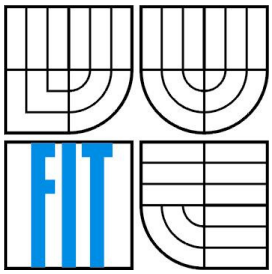


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PORTÁL PRO FIREMNÍ OBJEDNÁVÁNÍ OBĚDŮ

LUNCH BOOKING SYSTEM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — JIŘÍ LANG
AUTHOR

VEDOUCÍ PRÁCE — ING. LADISLAV RUTTKAY
SUPERVISOR

BRNO 2011

Abstrakt

V této práci je popsán návrh a implementace webového portálu pro zajištění objednávání obědů firemním zaměstnancům. Portál je určen pro dodavatele zajišťujícího firemní stravování jako nástroj pro jednoduché vystavení jídelníčku, správu firemních zaměstnanců a jednotlivých jídelen. Dále poskytuje dodavateli přehled objednávek a celkovou cenu. Jednotlivým strávníkům pak portál umožňuje online objednávání obědů a evidenci historie objednávek. Portál je implementován pomocí objektově orientovaného přístupu na třívrstvé architektuře s použitím nástrojů ASP.NET Framework, MS SQL Server a AJAX Control Toolkit.

Abstract

This thesis describes design and implementation of web portal for providing lunch booking for company employees. Portal is intended to lunch provider as a tool for publishing the menu, management of company employees and restaurants. It provides overview of orders and total cost to provider. Employees can online booking lunches and show the history of orders. Portal is implemented using object oriented programming on three-tier architecture using tools like ASP.NET Framework, MS SQL Server and AJAX Control Toolkit.

Klíčová slova

Objednávání obědů, webový portál, ASP.NET, MS SQL.

Keywords

Lunch booking, web portal, ASP.NET, MS SQL.

Citace

Lang Jiří: Portál pro firemní objednávání obědů, bakalářská práce, Brno, FIT VUT v Brně, 2011

Portál pro firemní objednávání obědů

Prohlášení

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

……
Jiří Lang
24. května 2011

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce Ing. Ladislavu Ruttkayovi za odborné vedení a cenné připomínky při tvorbě této práce.

Obsah

1 Úvod

Cílem této práce je implementace webového portálu pro firemní objednávání obědů. Existuje mnoho jídelen a dodavatelů zajišťujících stravování firmám a jejich zaměstnancům. Zdaleka ne všechny, ale používají skutečně efektivní nástroj pro evidenci objednávek obědů. Stále na trhu existují jídelny, kde se vyřizují objednávky papírovou formou, telefonicky nebo elektronickou poštou. Exitující webové objednávkové systémy jsou zaměřeny nejčastěji na stravování ve školních jídelnách. Většina firem řeší objednávky obědů implementací vlastního portálu umístěného na firemní síti, kam mají přístup pouze zaměstnanci. Výhodou je efektivní objednávání obědů z pohledu zaměstnance a evidence firemních objednávek. Nevýhodou je ale složitá komunikace s dodavatelem, kdy je přehled objednávek dodavateli zasílán elektronickou poštou nebo sdělován telefonicky. Hlavní myšlenka této práce je vytvořit systém, do kterého budou mít přístup všichni zaměstnanci z různých firem a zároveň dodavatel obědů, který získá jednoduše přehled o celkovém počtu objednávek na daný den.

Portál bude implementován za použití objektově orientovaného přístupu, který umožňuje provést abstrakci reálných částí systému na objekty v programu. Díky objektovému přístupu bude portál v budoucnu možné snáze rozšířit o další funkcionalitu. Pro návrh systému budou použity diagramy modelovacího jazyka UML. Struktura uložených dat v databázi bude navržena pomocí Entity-Relationship diagramu.

Práce je tematicky rozdělena do několika kapitol. Ve druhé kapitole je uvedeno srovnání existujících řešení firemního stravování. Je zde nastíněn princip jednotlivých způsobů zajištění stravování a zmíněny výhody a nevýhody.

Třetí kapitola se zabývá současnou situací ve firemním stravování, zejména z pohledu zaměstnavatele. Jsou zde popsány příspěvky na stravování z hlediska legislativy.

Čtvrtá kapitola popisuje technologie použité k návrhu a implementaci portálu. Jedná se zejména o jazyk UML, platformu Microsoft .NET Framework, XML, XSLT a základní koncepty objektově orientovaného přístupu.

Pátá kapitola se zabývá konkrétním návrhem webového portálu. Jsou zde uvedeny požadavky na aplikaci a návrhové modely jazyka UML a E-R diagram.

Šestá kapitole pak popisuje samotnou implementaci webového portálu. Je zde popsána architektura systému a její jednotlivé vrstvy. Dále jsou pak zmíněny nejvýznamnější operace v systému.

2 Srovnání existujících systémů

2.1 Neelektronické systémy

Mnoho stravovacích zařízení zajišťujících závodní stravování, řeší objednávky obědů neelektronickou formou. Příkladem je systém papírových lístků. Tento systém funguje tím způsobem, že strávník si zakoupí objednávkové lístky, přečte si na nástěnce jídelní lístek na následující dny a vybere si z nabídky obědů. Svůj výběr pak razítkem označí na dvojici lístků, z nichž jeden si ponechá a druhý vhodí do připraveného boxu. Boxy jsou číslovány podle jednotlivých obědů. Každý den v určitou hodinu zaměstnanci jídelny vyberou jednotlivé boxy a spočítají lístky. Podle množství lístků poté na druhý den připraví potřebný počet porcí.

Tento systém má řadu nevýhod. Největší nevýhodou jsou časové nároky na spočítání lístků a stanovení počtu porcí. Dále také nevratnost objednávky, jestliže strávník jednou vhodí lístek s objednávkou do boxu, nemůže svou objednávku změnit nebo zrušit.

Za výhody by se dala považovat přehlednost objednávání a snadné pochopení systému ze strany strávníka.

2.2 Objednávkové terminály s čipovými kartami

Pokročilejší metodou pro objednávání obědů je zavedení systému čipových karet a terminálů. Každý strávník má vlastní čipovou kartu, která reprezentuje jeho elektronické konto. Strávník si na konto ukládá peníze, které poté čerpá při placení obědů. Dle typu stravovacího zařízení si strávník buď objednává obědy předem přes objednávkový terminál, nebo si každý den vybírá z aktuální nabídky.

Výhodou tohoto systému je jednodušší zjištění množství obědů ze strany dodavatele. Strávník si může zobrazit pohyb peněz na svém účtu a díky tomu má přehled kolik stály jednotlivé obědy. Také má díky čipové kartě možnost do určité hodiny změnit nebo zrušit svoji objednávku.

Mezi nevýhody patří nutnost objednávání obědů přes terminál umístěný zpravidla v blízkosti jídelny a propadnutí oběda v případě nevyzvednutí.

2.3 Webový portál

Objednávání obědů přes webové rozhraní je nejrychlejším a nejefektivnějším způsobem. Strávník si vybere a objedná oběd přímo z kanceláře a svou objednávku může také s dostatečným předstihem změnit či zrušit. Dodavatel má jednoduše přehled o počtu a druhu objednaných obědů.

2.3.1 Přihlašování z firemní sítě

Příklad z praxe je obědový systém jedné brněnské firmy. Zaměstnanci mají přiděleno přihlašovací jméno a heslo do systému, kde si jednoduše mohou objednat oběd. Uzávěrka objednávek je každý den v 13.00. Do té doby lze oběd na další den odhlásit. Pokud zaměstnanec chce zrušit objednávku po této hodině, má možnost umístit oběd do tzv. burzy obědů. Burza obědů je společná pro všechny uživatele systému. Ti si tak mohou objednat oběd, který někdo jiný zrušil, i po ukončení objednávek. Systém dále umožňuje uživateli zjistit stav konta a historii jednotlivých transakcí. Komunikace s dodavatelem

probíhá telefonicky případně elektronickou poštou. Dodavatel jednou týdně pošle jídelníček na aktuální týden a administrátor ho zadá do systému. Po skončení objednávek administrátor vytiskne statistiku počtu objednávek a tu poté zašle dodavateli.

Výše zmiňovaný systém obsahuje dobrou koncepci z hlediska firemního objednávání obědů. Z pohledu zaměstnance se jedná o nejvíce přívětivou a efektivní metodu. Jedinou nevýhodou pro zaměstnance je nemožnost přihlášení do systému z počítače mimo firemní síť. Další nevýhodou je komunikace mezi firmou a dodavatelem, která zde není vyřešena zcela ideálně. Dodavatel nemá přístup přímo do systému a musí tak jídelničky zasílat emailem správci systému, přehled objednávek je dodavateli sdělován telefonicky.

Navrhovaný webový portál řeší tuto komunikaci přidáním role dodavatele do systému. Dodavatel tak jednoduše zadá jídelní lístek přímo do systému a nechá si vypsat seznam objednávek. Toto řešení je úspornější z hlediska využití času a pracovní síly.

2.3.2 Portál Strava.cz

Tento portál umožňuje registrovaným uživatelům online objednávat obědy v partnerských jídelnách. Při přihlášení si uživatel zvolí stravovací zařízení, kde je registrován, zadá své přihlašovací údaje a email, na který mu budou zasílány zprávy ze systému. Přihlášený uživatel může prohlížet jídelníček vybrané jídelny a přihlašovat a odhlašovat obědy. Dále má možnost zobrazit přehled vydaných obědů, přehled provedených plateb a historii odeslaných zpráv.

Jak již bylo zmíněno, portál poskytuje zasílání zpráv o provedených akcích uživateli na email. V nastavení svého účtu si uživatel může zvolit, jaké zprávy si přeje zasílat. Na výběr jsou tyto druhy zpráv: potvrzení přijaté objednávky, upozornění na nedostatečnou výši konta, měsíční přehled stravování a upozornění na neodebranou stravu. Tyto zprávy tak poskytují uživateli dodatečný přehled nad jeho účtem.

Zajímavým rozšířením tohoto objednávkového portálu je možnost objednávání obědů pomocí SMS. Uživatel napíše strukturovanou SMS obsahující číslo jídelny, jeho přihlašovací údaje, datum oběda a požadovanou operaci (přihlásit nebo odhlásit oběd) a po zpracování SMS je uživateli zaslána potvrzovací zpráva na email.

Pokud porovnáme tento webový portál s navrhovanou aplikací, najdeme hlavní rozdíl v podobě cílového uplatnění. Portál Strava.cz není primárně určen pro zajištění firemního stravování, je užíván spíše školními jídelnami a studenty základních a středních škol. [4]

2.3.3 Portál E-strava

Tento portál byl vyvinut firmou Ulrich software a slouží pro online přihlašování a odhlašování obědů vydávaných stravovacími zařízeními ve školách, domovech důchodců a jiných sociálních zařízeních. Pro objednávání musí mít uživatel magnetickou nebo čipovou kartu do příslušné jídelny a musí mít zakoupeno stravné na dané dny. Po přihlášení a výběru jídelny se uživateli zobrazí jídelníček na aktuální měsíc a může zde přihlašovat nebo odhlašovat své obědy. Informace o provedených změnách mohou být zasílány uživateli také elektronickou poštou.

Firma Ulrich software se zabývá komplexním řešením školního stravování. Zajišťuje jak softwarové řešení zahrnující informační systémy pro evidenci objednávek a skladovou evidenci, tak hardwarové řešení v podobě čtečky magnetických a čipových karet přímo v jídelnách a objednávkových terminálů. Internetový portál pro objednávání obědů E-strava je tak pouze nadstavbou nad již existujícími firemními systémy a sám o sobě tedy není příliš komplexní. [5]

3 Firemní stravování

Velké podniky s vlastními závodními jídelnami už dnes nejsou zdaleka tak časté jako dříve. V současné době se skládá firemní sféra z většího počtu spíše menších subjektů. Tyto firmy pak volí způsob stravování zaměstnanců nejčastěji zajištěním obědů v závodních jídelnách třetích stran nebo přímým příspěvkem na oběd zaměstnanci v podobě stravenek. Ze zákona přitom vyplývá, že firma není povinna zajistit zaměstnanci stravování, je pouze povinna mu poskytnout přestávku na oběd. V rámci konkurenčního boje na trhu práce však většina firem nabízí svým zaměstnancům zajištění stravování nebo stravenky v podobě zaměstnaneckých benefitů. Zaměstnanci si na tyto benefity zvykli a proto lze předpokládat, že je firmy budou nadálé poskytovat i po eventuální změně zákona, která by zrušila daňové zvýhodnění příspěvků na oběd.

3.1 Daňové zvýhodnění příspěvku na oběd

V současnosti jsou příspěvky na oběd zaměstnancům daňově zvýhodněné, což znamená, že zákon o daních z příjmu umožňuje zaměstnavateli zaúčtovat příspěvek na oběd jako daňově uznatelný výdaj. Pro zaměstnavatele je tak výhodnější přispět určitou částkou zaměstnanci na stravování, než mu o stejnou částku navýšit jeho hrubou mzdu. Při navýšení hrubě mzdy by musel zaměstnavatel tuto částku ještě zdanit, ale z příspěvků na stravování daň odvádět nemusí.

Zákony platné k 1. 1. 2011 stanovují daňové zvýhodnění příspěvku na oběd takto: §24 odst. 2 písm. j) bod 4 zákona č. 586/1992 o daních z příjmů, stanovuje jako daňově uznatelné: výdaje vynaložené na provoz vlastního stravovacího zařízení nebo příspěvky na stravování zajišťované prostřednictvím jiných subjektů. Tento příspěvek je uznatelný jako výdaj až do výše 55% ceny jednoho jídla za jednu směnu, maximálně však do výše 70% stravného vymezeného pro zaměstnance v rámci pracovní cesty. [6]

Výše stravného je každoročně upravována vyhláškou vydávanou Ministerstvem práce a sociálních věcí. Od 1. 1. 2011 nabývá platnost vyhláška č. 377/2010 Sb., která stanovuje sazbu základních náhrad zaměstnanců za stravné ve výši 63 Kč, trvá-li pracovní cesta 5 až 12 hodin. [7]

Z výše uvedeného vyplývá, že v roce 2011, může zaměstnavatel účtovat příspěvek na stravování jako daňově uznatelný výdaj, pokud výše příspěvku nepřesáhne 44 Kč a zároveň 55% z ceny jednoho jídla. V navrhované aplikaci se to projeví při výpočtu příspěvku na oběd. Pokud příspěvek přesáhne 44 Kč, aplikace jej zaokrouhlí na 44 Kč, aby zaměstnavatel mohl uplatnit daňové zvýhodnění příspěvku. Vzniklý rozdíl v ceně pak uhradí zaměstnanec.

4 Použité technologie

4.1 UML

UML je modelovací jazyk, který slouží k popsání návrhových modelů aplikace pomocí jednotné syntaxe. UML slouží ke grafickému zobrazení modelovaného systému tak, aby byl model dostatečně srozumitelný pro zákazníka, ale zároveň užitečný pro programátora. Modely pomáhají určit požadavky na systém již na počátku jeho tvorby.

4.1.1 Diagram případů užití (Use case diagram)

Diagram případů užití umožňuje stanovit přesné požadavky uživatelů na systém a vymezit rozsah implementovaných funkcí. Každý případ užití popisuje jeden způsob použití systému, tedy jednu jeho funkci. Uživatel vystupuje v diagramu jako aktér. Je to vlastně role, kterou zaujímá uživatel vůči systému. Pro každou roli je dána množina operací, kterou smí aktér nad systémem provádět. Jeden aktér může provádět více případů užití, stejně jako jeden případ užití může být prováděn více aktéry.

4.1.2 Diagram časování (Time diagram)

Diagram časování je specifickým diagramem interakcí, který se používá pro modelování systémů pracujících v reálném čase. Diagram popisuje stav objektu v závislosti na čase, přičemž může popisovat jeden nebo více objektů. Tento diagram má dvě formy stavovou a hodnotovou.

V práci je použita hodnotová forma diagramu časování, která je tvořena časovou osou, životními čárami objektu a jeho stavy. Každý stav je definován horizontální životní čarou, na níž může být vyznačena konkrétní doba trvání stavu objektu. Změna z jednoho stavu do druhého je vyjádřena vertikálním posunutím čáry života. [14]

4.2 Diagram entit a vztahů (Entity Relationship Diagram)

Diagram entit a vztahů (ERD) slouží k modelování datové části aplikace. Popisuje způsob uložení perzistentních dat a vztahy mezi nimi pomocí entit a vztahů. Entita zastupuje skutečnou věc či osobu z reálného světa. Příklad entity je uživatel systému, objednávka v systému atd. Každá entita je v rámci systému jedinečná. Vztah definuje interakce mezi entitami, například uživatel “má“ objednávku, objednávka “obsahuje“ oběd.

4.3 Microsoft .NET Framework

.NET Framework je rámec pro vývoj a správu aplikací vyvinutý společností Microsoft. Jednou z jeho předností je nezávislost na zvoleném programovacím jazyku, díky CLR (Common Language Runtime – společný jazykový běhový modul). CLR vytváří virtuální vrstvu mezi operačním systémem a aplikací. Další výhodou tohoto rámce je komponenta knihovna tříd rámce .NET (Framework Class Library ─ FCL). FCL dodává objektově orientované API, které poskytuje množství funkcí a objektů užitečných pro vývoj aplikace.

Součástí rámce jsou webové služby XML a ASP.NET (ASP – Active Server Pages). ASP.NET přináší serverové ovládací prvky, které spouštějí události zpracovávané na straně serveru pomocí skriptů a HTML omezují pouze na klienta. [3]

4.4 XML

XML (Extensible Markup Language) je součástí rodiny značkovacích jazyků SGML. SGML je jazyk, který vznikl pro deklaraci různých typů dokumentů. Nejrozšířenější aplikace SGML je jazyk HTML sloužící pro popis hypertextových dokumentů. Ze SGML se poté vyvinul upravený metajazyk pro deklaraci různých typů strukturovaných dat XML. [8]

Vidíme tedy, že jazyk XML je příbuzný s jazykem HTML. Na rozdíl od HTML, který slouží převážně pro zobrazení dat, XML je používán zejména k jejich uložení. Oba jazyky jsou značkovací, tzn., že používají značky k vyjádření struktury a informací o dokumentu. Ale zatímco HTML má množinu značek a jejich smysl pevně dán, XML nedefinuje sémantiku značek, jejich interpretace se tak v různých dokumentech liší. XML je tedy jazyk sloužící k popisu a strukturalizaci dat. Jeho výhodami jsou:

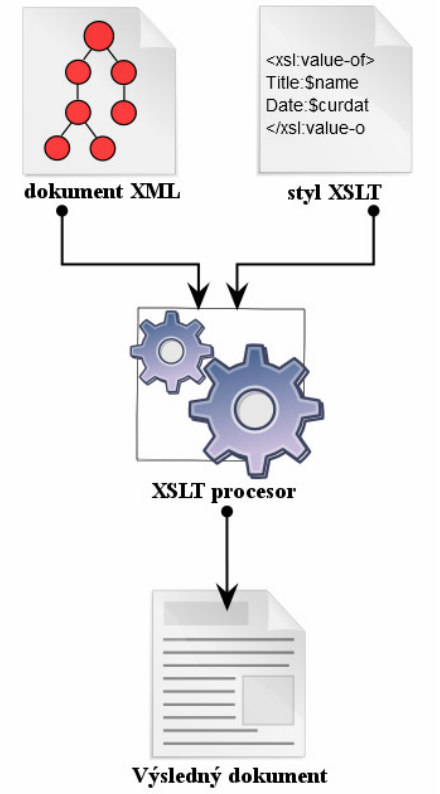
- nezávislost na platformě,
- široké rozšíření,
- velký počet nástrojů pro práci s XML dokumenty a jejich snadná modifikovatelnost
- snadná čitelnost i pro laika.

Příklady použití XML jsou například zajištění společného formátu pro sdílení dat mezi firmami, přenos dat mezi webovými servery nebo ukládání strukturovaných dat.

4.5 XSLT

„Standard XSLT (Extensible Stylesheet Language Transformation) slouží k transformaci zdrojových dokumentů XML (případně pouze jejich částí) na výstupní dokumenty XML prostřednictvím transformačních pravidel, která jsou uvedena v dokumentu XML zvaném styl XSLT (XSLT stylesheet). Výstupní dokument nemusí nutně splňovat požadavky standardu XML; může jím být i dokument HTML, případně textový dokument.“ (cit. Hruška, Kroulík, Techet, 2007, 42) Schéma transformace je na obr. 1.

Použití XSLT je výhodné zejména při sdílení strukturovaných dat ve formátu XML mezi firmami, kdy každá firma může mít svůj vlastní formát XML souboru. Při výměně souborů pak firma jednoduše použije šablonu XSLT pro transformaci předaného XML do svého firemního formátu. Pomocí XSLT můžeme snadno transformovat XML také na HTML. Toho se využívá při zobrazení strukturovaných dat ve formátu XML na webové stránce. Vzhled XML na webu lze upravit také podle kaskádových stylů CSS, ale XSLT je univerzálnější a zajišťuje větší kontrolu nad výstupem. [3]



Obr. 1: Schéma XSLT transformace. (upraveno podle [10])

4.6 ASP.NET AJAX Control Toolkit

Jedná se o open-source projekt založený na technologii rámce Microsoft ASP.NET AJAX. Je to výsledek společného úsilí komunity Microsoft a komunity ASP.NET AJAX, který poskytuje výkonnou infrastrukturu pro psaní znovupoužitelných, nastavitelných a rozšiřitelných ovládacích prvků. V současnosti projekt nabízí bohatou škálu hotových ovládacích prvků pro vytváření interaktivních webových stránek. (převzato z angličtiny [11])

4.7 Objektově orientované programování

Základní koncepty objektově orientovaného programování (OOP) dle [15]

- **Objekt** je základní jednotkou OOP, která v sobě spojuje data a funkcionalitu. Objekty umožňují modelovaný problém intuitivně rozdělit na podčásti přímo odpovídající realitě a pomocí jejich vzájemné komunikace tento problém řešit.
- **Abstrakce** je pohled na vybraný problém reálného světa a návrh jeho řešení použitím programu. Abstrakce určuje schopnost programu zjednodušit nebo zanedbat některé informace nebo vlastnosti objektů reálného systému.
- **Zapouzdření** zajišťuje, že interní stav objektu nelze měnit libovolným způsobem, ale pouze přes rozhraní poskytované objektem. Komunikace mezi objekty tak probíhá přes externí rozhraní a vlastní implementace metod uvnitř objektů je skryta. Zapouzdření je hlavní vlastností, která zajišťuje snadnou modifikovatelnost programu, protože při provádění změn stačí zachovat kompatibilitu s rozhraními objektů a nikoliv s jejich vnitřní implementací.

- **Polymorfismus** využívá toho, že v OOP se místo volání podprogramů používá mechanismus zasílání zpráv. Jakou metodu pak zpráva konkrétně vyvolá, závisí na konkrétním objektu, kterému je zpráva zaslána. Jedna zpráva tak může vyvolat více druhů chování.
- **Dědičnost** umožňuje implementovat sdílené chování. To znamená, že nové objekty mohou sdílet a rozšiřovat chování již existujících objektů, čímž odpadá nutnost opakovaného psaní stejných metod pro nové objekty. Dědičnost má klíčovou roli z hlediska rozšiřitelnosti objektově orientovaných systémů.

5 Návrh webového portálu

5.1 Požadavky na webový portál

Webový portál pro firemní stravování musí zajišťovat jednoduché, rychlé a efektivní objednávání obědů. Portál spravuje databázi všech strávníků, kteří mají zřízen uživatelský účet a mohou tak objednávat obědy. Ovládání musí být intuitivní a přehledné, aby strávník jednoduše věděl jak objednat či zrušit oběd a byl schopen jasně pochopit co očekávat od jednotlivých jídel. Strávník by měl mít možnost zobrazit složení jednotlivých jídel a sledovat historii svých objednávek. Portál musí kontrolovat aktuální čas a starat se o uzávěrku objednávek. Dodavateli musí poskytovat jednoduchý nástroj pro zadání jídelního lístku na příští dny a také přehled o počtu objednávek na jednotlivá jídla a informace o potřebných surovinách. O správu webového portálu se stará administrátor, který spravuje uživatele systému, firemní zákazníky a jednotlivé jídelny.

5.2 Návrh webového portálu

Návrh systému předpokládá existenci jednoho dodavatele, který vlastní několik jídelen do kterých dodává obědy. Dodavatel uzavírá smlouvu s celými firmami, následně pak zajišťuje stravování pro jejich zaměstnance. Správu jídelen, firem a zaměstnanců v systému provádí pro dodavatele administrátor systému. Dodavatel má na starost tvorbu jídelního lístku. Nejprve zadává do aplikace jednotlivá jídla a jejich složení. Poté vytváří týdenní jídelníček zadáním jídel k určitému datu. Oběd je tvořen jednou polévkou a nejméně třemi hlavními jídly.

Dodavatel může vytvořit jednotný jídelní lístek pro všechny jídelny, nebo také do každé jídelny dodávat jiné obědy. Aplikace umožňuje dodavateli zobrazit všechny objednávky na daný den, rozdělené mezi jídelny. V souvislosti s počtem objednaných porcí jednotlivých obědů, aplikace vypočítá celkové množství surovin nutných k nákupu pro uvaření obědů.

Zaměstnanec smluvní firmy má možnost objednat si oběd z libovolné jídelny. Cena oběda pro zaměstnance závisí na výši příspěvku na obědy jeho firmy. Výše příspěvku na oběd se může u různých firem lišit. Zaměstnanci se tak v systému zobrazí pouze cena, kterou bude on doplácet. Systém předpokládá, že dodavatel bude účtovat firmě plnou cenu objednaného oběda za každého jejího zaměstnance a firma si potom v rámci svého účetnictví odečte z ceny výši příspěvku a zbytek strhne zaměstnanci ze mzdy. Zaměstnanec má možnost zobrazit historii svých objednávek a celkovou cenu, získá tak přehled o očekávané měsíční srážce ze mzdy.

5.3 Funkce webového portálu dle diagramu případů užití

Diagram případů užití webového portálu je na obr. 2. S webovým portálem pracují následující aktéři:

- **Zaměstnanec** je běžný strávník, který objednává obědy. Může si objednat oběd, zobrazit složení obědového jídla, zobrazit uskutečněné objednávky a zobrazit týdenní jídelníček ve formátu k tisku.
- **Dodavatel** zajišťuje přísun obědů. Zadává do systému nová jídla spolu s jednotlivými ingrediencemi, vytváří týdenní menu a zjišťuje ze systému přehled objednávek. Přehled

objednávek je možné filtrovat podle data, firmy nebo jídelny. Na každý den si může vypsat počet objednaných porcí jednotlivých jídel a souhrnné množství potřebných surovin.

- **Administrátor** je privilegovaný uživatel a také správce portálu. Může provádět stejné operace jako Dodavatel, ale navíc se stará o správu uživatelů systému, firemních zákazníků, zaměstnanců a jídelen.

Obr. 2: Diagram případů užití

5.3.1 Popis jednotlivých případů užití

Zaměstnanec

- Objednat oběd - zaměstnanec si může objednat oběd, pokud není po uzávěrce obědů.
- Zrušit objednávku - zaměstnanec může v určitém termínu zrušit objednávku již objednaného obědu.
- Zobrazit své objednávky – zaměstnanec si zobrazí historii svých objednávek.
- Zobrazit jídelníček – zaměstnanec si zobrazí týdenní přehled obědů, pro zvolenou jídelnu.

Dodavatel

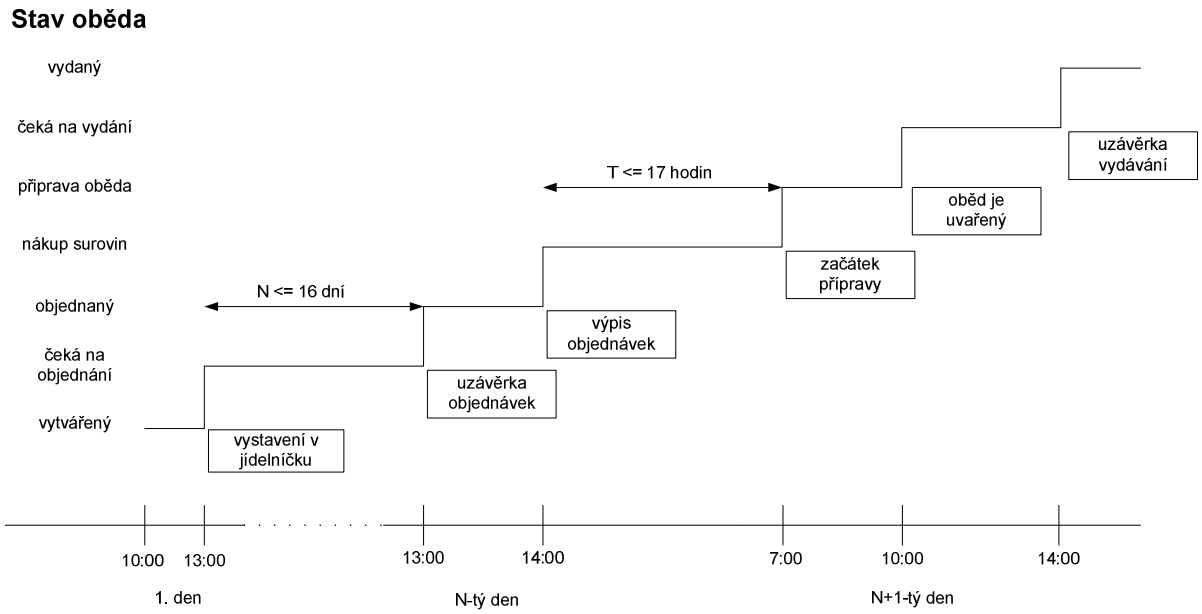
- Zobrazit report - dodavatel získá přehled objednaných porcí jednotlivých obědů, včetně množství surovin nutných k uvaření jídel.
- Vytvořit jídelníček - dodavatel vytvoří týdenní jídelníček.
- Zobrazit objednávky - dodavatel získá přehled všech uskutečněných objednávek.
- Správa jídel - dodavatel může přidat jídlo do systému a přiřadit k němu jednotlivé ingredience, může smazat již existující jídlo, které ještě nebylo vystaveno v jídelníčku.

Administrátor

- Správa uživatelů - administrátor přidává a odebírá účty uživatelů systému.
- Správa firem - administrátor přidává do systému nové firemní zákazníky a může upravit či odstranit již existující zákazníky.
- Správa zaměstnanců – administrátor zadává a upravuje údaje o firemních zaměstnancích.
- Správa jídelen - administrátor přidává do systému nové jídelny, upravuje a odstraňuje již existující jídelny.

5.4 Diagram časování

Diagram časování ukazuje životní cyklus jednoho oběda. Dodavatel nejprve vytvoří jídelníček s obědem a vystaví ho v systému. Jídelníček je vytvářen každou středu a to až na dva týdny dopředu. Oběd si může do uzávěrky objednávek strávník objednat nebo zrušit. Po uzávěrce objednávek je oběd již nevratně objednán. Nastává fáze přípravy oběda, nejprve je nutné nakoupit suroviny potřebné na uvaření jídla, další den ráno začíná samotné vaření oběda, tak aby byl oběd od 10:00 nachystán k vydání. Ve 14 hodin končí vydávání obědů a oběd je pokládán za vydaný.

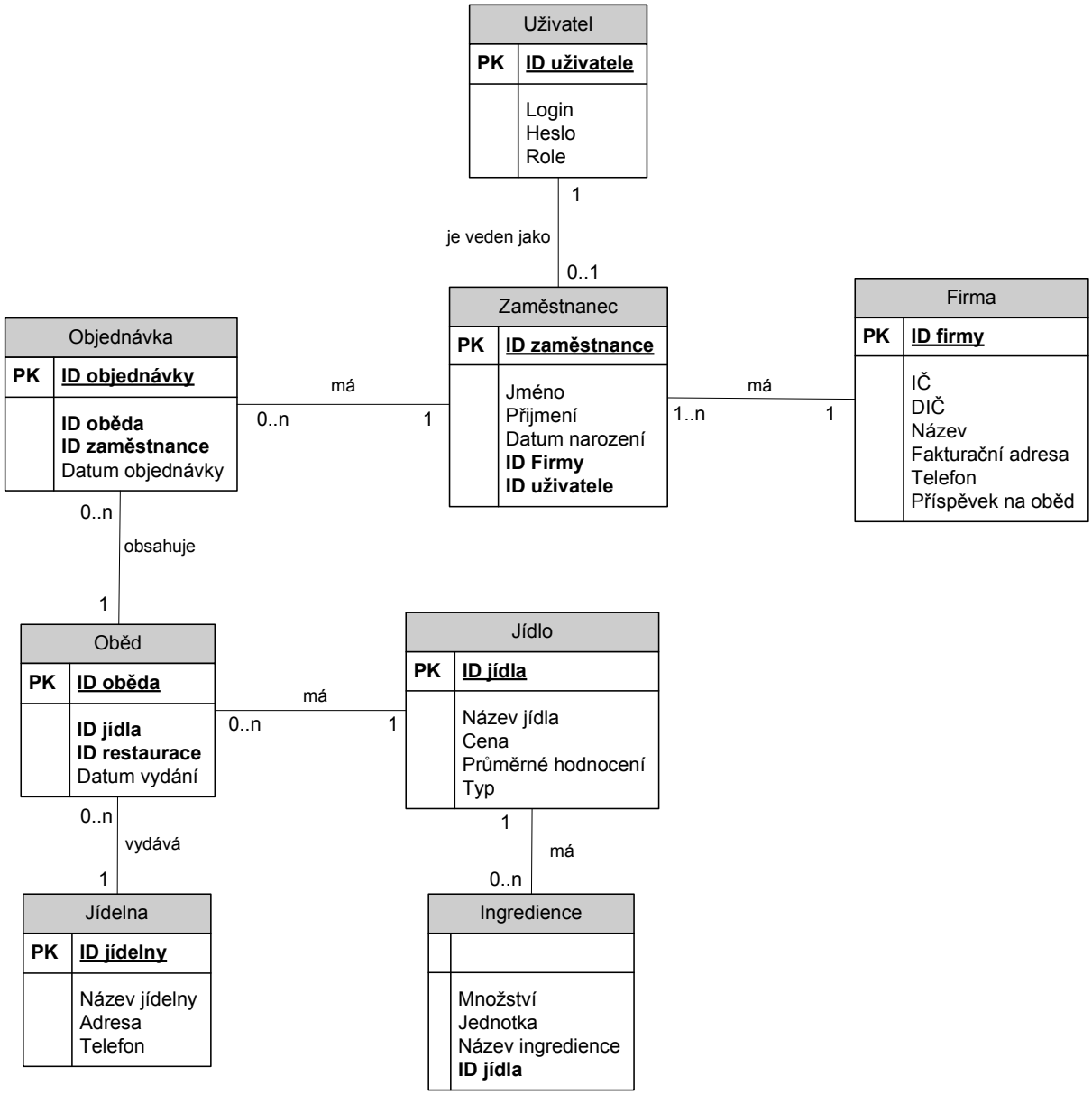


Obr. 3: Diagram časování pro životní cyklus oběda.

5.5 ER diagram

Návrh databáze webového portálu je znázorněn na obr. 4 pomocí ER diagramu. Diagram obsahuje tyto entity:

- **Oběd** - obsahuje informace o vytvořeném obědu, je identifikována unikátním identifikačním číslem. Entita dále uchovává informace o datu, kdy je oběd vydáván, jídelně, která oběd vydává a odkaz na entitu jídlo, které tvoří oběd.
- **Objednávka** - představuje konkrétní objednávku jednoho strávníka na jeden oběd. Entita obsahuje informace o identifikační číslo uživatele, který ji uskutečnil, identifikační číslo oběda, který je objednán a datum vytvoření objednávky.
- **Uživatel** - eviduje uživatele systému a jeho přihlašovací údaje, obsahuje položky identifikační číslo uživatele, login, heslo, a role.
- **Zaměstnanec** - představuje firemního zaměstnance, který si objednává obědy. O zaměstnanci jsou uchovávány tyto údaje: identifikační číslo zaměstnance, jméno, příjmení, datum narození, identifikační číslo zaměstnancovi firmy a identifikační číslo uživatele, kterým je zaměstnanec reprezentován v systému.
- **Firma** - představuje firmu, která má smlouvu s dodavatelem o poskytování obědů svým zaměstnancům. Uchovává tyto údaje: identifikační číslo firmy, název firmy, IČ, DIČ, adresa firmy, telefonní číslo a výše příspěvku na obědy.
- **Jídlo** - představuje jídlo uložené v databáze, přiřazením jídla ke konkrétnímu datu a jídelně vzniká oběd. Jídlo obsahuje informace: identifikační číslo jídla, název jídla, cena jídla, průměrné hodnocení jídla a typ jídla. Typ jídla rozlišuje, zda se jedná o polévku či hlavní jídlo.
- **Ingredience** - uchovává informace o jednotlivých ingrediencích potřebných pro přípravu jednotlivých jídel. Obsahuje položky množství, jednotka, ingredience a identifikační číslo jídla, do kterého se ingredience přidává.
- **Jídelna** - představuje jednu z jídelen, do kterých dodavatel dodává obědy. O jídelně jsou uchovávány tyto informace: identifikační číslo jídelny, název jídelny, adresa jídelny a telefonní číslo.



Obr. 4: ER diagram

6 Implementace

6.1 Objektově orientovaný přístup

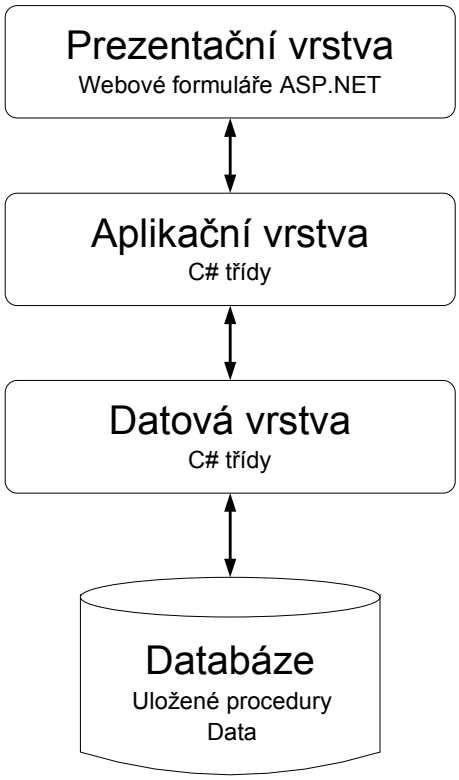
Pro implementaci aplikace byl zvolen objektově orientovaný přístup, který je vhodný pro komplexní aplikace tohoto typu, protože přináší možnost abstrakce reálného systému na softwarový model. Například v našem objednávkovém systému jsou evidování zaměstnanci, v aplikaci se tedy vyskytuje objekt Zaměstnanec (Employee). Reálný zaměstnanec má jméno, příjmení, datum narození a zaměstnavatele, stejně tak objekt Zaměstnanec má odpovídající atributy. Tímto způsobem je provedena abstrakce nad všemi součástmi řešeného systému. Objektový přístup přináší několik výhod, například funkce přebírají jako parametr celý objekt, pokud bychom tedy chtěli změnit uchovávané informace o zaměstnanci, stačí pouze přidat nebo odebrat atributy objektu a funkce pracující s objektem jako s celkem zůstanou nezměněny. Taktéž má práce s objekty tu výhodu, že v jakémkoliv okamžiku máme k dispozici všechny atributy objektu a všechny metody pro práci s objektem.

6.2 Architektura aplikace

Aplikace pracuje na třívrstvé architektuře, kde je od sebe oddělena vrstva pro práci s daty, aplikační vrstva a prezentační vrstva. Schéma architektury je na obr.5. Důvody pro použití této architektury jsou následující.

Výhody třívrstvé architektury:

- oddělení databázové logiky od aplikační
- snadná modifikovatelnost - při změně databáze se změny promítnou pouze do datové vrstvy, ostatní vrstvy zůstanou nezměněny
- nezávisí na vnitřní implementaci vrstvy pouze na rozhraní přes které vrstvy komunikují - můžeme tak měnit implementaci libovolné vrstvy při zachování společného rozhraní
- znovu použitelnost aplikace
- rozložení výpočetního výkonu např. na více serverů



Obr. 5: Schéma architektury aplikace

6.2 Databáze

Databáze aplikace je uložena na Microsoft SQL Serveru. Obsahuje tabulky dle návrhu databáze v kapitole 5.5 a uložené procedury.

6.2.1 Uložené procedury

Uložené procedury představují SQL příkazy uložené přímo v databázi. Použití uložených procedur je výhodné pro každý příkaz vykonávaný na databázi opakovaně. Oproti dynamickým SQL příkazům se uložené procedury vykonávají rychleji, protože jsou již zkompilované. Použití uložených procedur tak může podstatně zvýšit výkon datové aplikace. [3]

Výhodou uložených procedur je také to, že pracují s typovanými parametry. To snižuje riziko útoku na aplikaci pomocí SQL injection. Pokud bychom měli neparametrizovaný dynamický SQL příkaz např.

```
"SELECT * FROM USER WHERE userId = ' " + userID + "'"
```

Může potenciální útočník snadno napadnout naší databázi podvrhnutím hodnoty userId, např.

```
string userId = "'; DROP DATABSE nazev --";
```

příkaz pak bude vypadat takto

```
"SELECT * FROM USER WHERE userId = ''; DROP DATABASE nazev--'
```

V tomto případě se nejprve vyhodnotí klasický `SELECT` s nespecifikovaným `userId`, a poté díky středníku, který ukončuje příkaz `SELECT` začne vyhodnocení dalšího příkazu v tomto případě `DROP DATABASE nazev`. Dvojité pomlčky způsobí, že se nevyhodnocuje zbytek příkazu, tím pádem apostrof nezpůsobí chybu překladu, a příkaz se úspěšně provede. Pokud používáme uložené procedury tento útok by se nezdařil, protože procedura očekává pouze hodnotu parametru a nikdy nemůže dojít k vyhodnocení parametru jako dalšího příkazu. [12]

Na obr.6. je uveden příklad uložené procedury, která získá z databáze seznam všech objednávek filtrovaný podle zadaných parametrů. Nejprve je v proceduře uveden seznam parametrů, které procedura očekává spolu s jejich typy. Pokud například místo celočíselného identifikačního čísla jídelny bude zadán řetězec (podvržený příkaz), procedura skončí s chybou a příkaz se neprovede. Procedura pracuje s těmito parametry: rozsah data od, rozsah data do, identifikační číslo jídelny a identifikační číslo firmy. Pokud je některý z parametrů nulový (dodavatel filtruje objednávky pouze podle některých kritérií), funkce `ISNULL()` zajistí, že se daný sloupec neporovnává s nulovým parametrem, ale sám se sebou, tzn. výsledek se podle tohoto sloupce nebude filtrovat.

```
CREATE PROCEDURE GetOrdersFiltered
    @fromDate datetime,
    @toDate datetime,
    @restaurantId smallint,
    @companyId smallint
AS
BEGIN
    SET NOCOUNT ON;
    SELECT o.id, o.lunchId, o.employeeId, o.date
    FROM "Order" o, "Employee" e, "Lunch" l
    WHERE l.date >= ISNULL(@fromDate,l.date)
    AND l.date <= ISNULL (@toDate,l.date)
    AND o.employeeId = e.id
    AND e.companyId = ISNULL(@companyId,e.companyId)
    AND o.lunchId = l.id
    AND l.restaurantId = ISNULL(@restaurantId,l.restaurantId)
    ORDER BY l.date
END
```

Obr. 6: Příklad uložené procedury aplikace

6.3 Datová vrstva - Data Access Layer

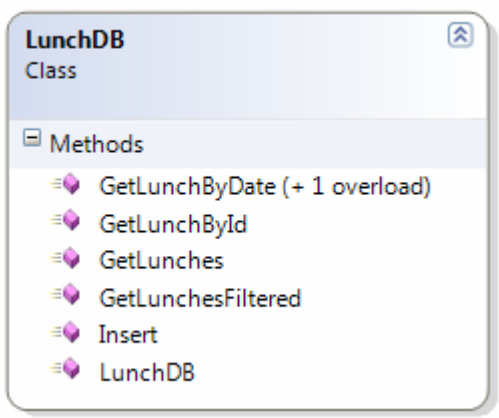
Datová vrstva slouží pro přístup k databázi a práci s daty. V aplikaci je datová vrstva implementována jako samostatný projekt s názvem `DataAccessLayer`. Tento projekt obsahuje jednotlivé třídy, které odpovídají tabulkám v databázi. V každé třídě jsou metody pro volání uložených procedur provádějících SQL dotazy nad příslušnou databázovou tabulkou. Při volání uložené procedury jsou jí předány parametry uložené v kolekci `SqlParameterCollection`, používání parametrů je výhodné zejména s ohledem na riziko SQL injection. Uložená procedura očekává typované parametry, při předání špatných parametrů, například jiného datového typu, procedura vrátí chybu. Data vrácená uloženou procedurou jsou uložena do objektu typu `DataReader`. Ten se poté předává dále do aplikační vrstvy.

6.3.1 DataReader

Je objekt zapouzdřující výsledky vrácené databázovým dotazem. Má vlastní metody sloužící pro procházení výsledků, například vrácení hodnoty podle názvu sloupce nebo pořadí sloupce, načtení následujícího řádku výsledku atd. Výhodou objektu `DataReader` je jeho výkonnost, protože přebírá pouze ta data, která jsou opravdu požadována a ne celou tabulku jako je tomu například u objektu `DataSet` (ten ukládá všechny výsledky do mezipaměti a umožňuje nad nimi dále provádět pokročilé operace). `DataReader` je tak ideálním nástrojem pro velké množství často se opakujících databázových dotazů, u nichž předpokládáme pouze čtení výsledků.

6.3.2 Třída LunchDB

Na příkladu této třídy si popíšeme princip práce se třídami na datové vrstvě. Schéma třídy je vidět na obr.5. Třída obsahuje metody, které volají uložené procedury provádějící dotazy nad databází, zejména nad tabulkou `Lunch`, výsledky těchto dotazů se předávají třídě `Lunch` v aplikační vrstvě.



Obr. 7: Schéma třídy LunchDB

Příkladem ilustrující práci s uloženými procedurami v datové vrstvě může být přetížená metoda `GetLunchByDate()` viz obr.8. Tato metoda má argumenty: datum, pro které chceme vypsat obědy, identifikační číslo jídelny vydávající oběd a typ jídla (indikuje zda se jedná o polévku nebo hlavní jídlo). Metoda vytvoří nový objekt typu `Hashtable` (jedná se o kolekci párových záznamů klíč-hodnota, záznamy jsou indexovány podle klíče) a do něj uloží parametry pro uloženou proceduru. Poté volá metodu `executeProcedure()` z třídy `Connection`, která namapuje parametry z `Hashtable` do kolekce `SqlParameterCollection` a zavolá uloženou proceduru. Příkazem `ExecuteReader()` se vykoná uložená procedura a výsledek se uloží do objektu `DataReader`, který je dále předán aplikační vrstvě.

```
public SqlDataReader getLunchByDate(DateTime date, int restaurantID, int
                                                                    mealType)
{
    Hashtable parameters = new Hashtable();
    parameters["@date"] = Convert.ToDateTime(date);
    parameters["@restaurantId"] = restaurantID;
    parameters["@mealType"] = mealType;
    return Connection.executeProcedure("GetLunchesByDateAndRestaurant",
                                        parameters);
}
```

Obr. 8: Příklad metody pro volání uložené procedury

6.4 Aplikační vrstva - Business Layer

Aplikační vrstva tvoří spojovací článek mezi datovou a prezentační vrstvou. V aplikační vrstvě se pracuje s objekty, které jsou plněny daty poskytnutými datovou vrstvou, a dále poskytovány vrstvě prezentační. V aplikaci je aplikační vrstva implementována jako samostatný projekt s názvem `BusinessLayer`. V tomto projektu jsou třídy odpovídající třídám v datové vrstvě. Všechny třídy jsou typu `Serializable`, aby bylo možné je ukládat do kolekcí.

6.4.1 Kolekce objektů

V aplikační vrstvě jsou jednotlivé části systému plynoucí ze struktury databáze implementovány jako objekty např. objekt `Lunch`. Každý objekt má atributy odpovídající informacím, které o něm potřebujeme uchovávat, a metody popisující množinu operací s objektem. Protože se však objekty nevyskytují pouze samostatně, potřebujeme nějak uchovávat množinu objektů. Nejlepším úložištěm pro skupinu objektů je silně typová kolekce objektů např. `LunchCollection`, která může obsahovat pouze objekty typu `Lunch` a přejímá odkazy na tento typ bez nutnosti dalšího přetypování. Z toho důvodu je ke každé třídě přiřazena příslušná třída `Collection`, která slouží pro vytváření kolekce objektů dané třídy. Do prezentační vrstvy je tak předána kolekce objektů, která má tu výhodu, že každý prvek kolekce je samostatný objekt, což nám umožňuje v kterémkoliv okamžiku přistupovat k jeho atributům a metodám.

6.4.2 Třída Lunch

Příkladem třídy v aplikační vrstvě je třída `Lunch` (obr.9), která obsahuje tyto atributy:

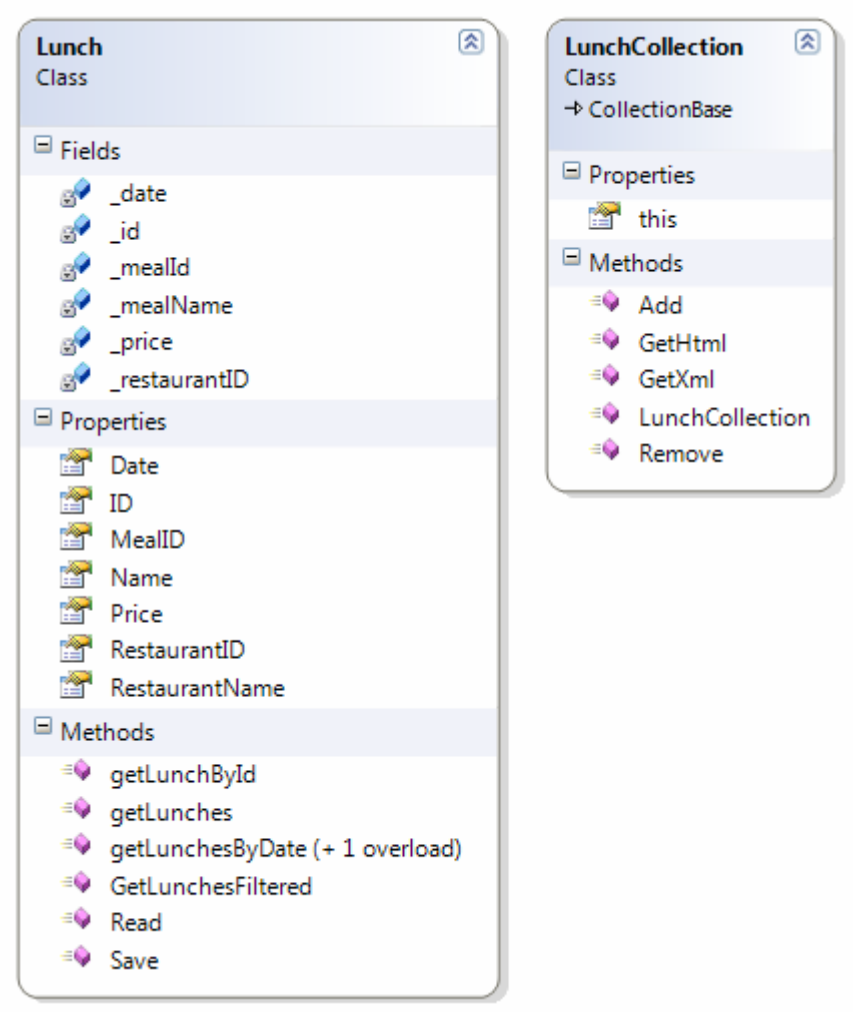
- `_id` - identifikační číslo oběda,
- `_mealId` - identifikační číslo jídla tvořícího oběd,
- `_date` - datum oběda,
- `_price` - cena oběda, vypočítává se podle přihlášeného uživatele a výše příspěvku jeho zaměstnavatele
- `_restaurantID` - identifikační číslo jídelny, která vydává oběd,
- `_mealName` - název obědového jídla.

a tyto metody:

- `GetLunchById` - vrací oběd podle zadaného identifikačního čísla
- `GetLunches` - vrací všechny obědy z databáze jako kolekci `LunchCollection`
- `GetLunchesByDate` - přetížená metoda vrací obědy podle více parametrů, metoda je rozebrána níže
- `Read` - vytvoří nový objekt a jeho jednotlivé atributy naplní hodnotami odpovídajících polí s objektu `DataReader`
- `Save` - slouží pro uložení nového objektu třídy `Lunch`.

6.4.3 Třída LunchCollection

Třída `LunchCollection` (viz obr.9) dědí z abstraktní třídy `CollectionBase`, která jí poskytuje metody pro práci se silně typovými kolekcemi. Tato třída obsahuje metody `Add`, `Remove`, které slouží pro přidávání a odebírání objektů z kolekce. Dalšími metodami této třídy jsou metody `GetXml()`, která provádí transformaci kolekce objektů do formátu XML, a `GetHtml()`, která transformuje vstupní data ve formátu XML na HTML užitím pravidel uvedených ve stylu XSLT.



Obr. 9: Schéma tříd Lunch a LunchCollection

Příkladem ilustrujícím práci s objekty v aplikační vrstvě může být metoda `GetLunchesByDate()` (viz ob.10), která vrací všechny obědy vypsané na zadaný den, vydávané zvolenou jídelnou. Metoda má argumenty: datum oběda, identifikační číslo jídelny a typ jídla (polévka nebo hlavní jídlo). Tato metoda zavolá příslušnou metodu z databázové vrstvy, od které převezme naplněný `DataReader` a ten pak předá vlastní metodě `Read()`, kde jsou jednotlivé atributy třídy `Lunch` naplněny daty z objektu `DataReader`. Naplněný objekt třídy `Lunch` přidá do kolekce `LunchCollection`. Tento postup se opakuje tak dlouho dokud nejsou přečteny všechny záznamy z objektu `DataReader`. Metoda vrací kolekci typu `LunchCollection`, obsahující všechny obědy na zadaný den v zadané jídelně.

```
public static LunchCollection getLunchesByDate(DateTime date, int
                                                restaurantID, int mealType)
{
    LunchCollection lunches = new LunchCollection();
    LunchDB lunchDB = new LunchDB();
    SqlDataReader reader = lunchDB.getLunchByDate(date,
                                        restaurantID, mealType);
    while (reader.Read())
    {
        lunches.Add(Read(reader));
    }
    reader.Close();
    return lunches;
}
```

Obr. 10: Příklad metody pro naplnění kolekce objektů

6.5 Prezentační vrstva - Presentation Layer

Prezentační vrstva je implementována jako projekt s názvem `PresentationLayer`. Obsahuje jednotlivé webové formuláře pro komunikaci systému s uživatelem. Prezentační vrstva zobrazuje data poskytnutá aplikační vrstvou. Nejčastěji se jedná o kolekce objektů, které jsou zobrazovány pomocí ovládacího prvku `DataGrid`.

Celkový vzhled systému je definován na stránce `MasterPage`, ze které ostatní stránky dědí. Prezentační vrstva je rozdělena do tří částí za účelem řízení přístupu pomocí uživatelských rolí. V aplikace jsou tři uživatelské role, které odpovídají jednotlivým aktérům v Use case diagramu: dodavatel, zaměstnanec a administrátor. Každá role má v projektu svoji složku, kde jsou uloženy stránky, na které má role přístup.

Pokud se například přihlásí uživatel s rolí zaměstnanec, uvidí pouze stránky ze složky Zaměstnanec. Na jiné stránky nemá přístup. Řízení přístupu je implementováno pomocí Formulářového ověřování.

6.5.1 DataGrid

DataGrid je ovládací prvek sloužící ke komplexnímu vázání dat. Komplexním vázáním dat se rozumí navázání ovládacího prvku na jeden či více sloupců dat. DataGrid patří mezi iterativní ovládací prvky a je charakteristický vykreslováním vícesloupcové, datově vázané mřížky složené z dat. DataGrid podporuje definice různých typů sloupců a pomocí šablon řídí rozvržení výsledných buněk. Tento prvek dále umožňuje stránkování výsledných dat a také přidání funkčnosti například pro editaci položek. Ovládacímu prvku DataGrid můžeme přidružit zdroj dat v podobě kolekce a on pak projde všechny položky zdroje dat a na jednotlivé řádky aplikuje šablony HTML. [16]

Příkladem pro zobrazení dat na prezentační vrstvě je stránka pro objednávání obědů. Na stránce najdeme jednotlivé dny v týdnu a pro každý den je připraven ovládací prvek `DataGrid`. Zavoláním metody `GetLunchesByDate()` z aplikační vrstvy získáme kolekci typu `LunchCollection`, obsahující jednotlivé obědy, na zadaný den vydávané ve zvolené jídelně. Tuto kolekci pak, jednoduše spojíme s prvkem `DataGrid`, který zajistí její vykreslení na stránce. Sloupce prvku `DataGrid` lze jednoduše přiřadit jednotlivým atributům objektu třídy `Lunch`, což poskytuje kontrolu nad tím, které informace o obědu chceme zobrazit.

6.5.2 Formulářové ověřování

Jedná se o ověřování uživatelů na základě údajů zadaných uživatelem do webového formuláře. V souboru `Web.config` se definují podmínky pro formulářové ověřování, jako přihlašovací stránka nebo rozsah chráněných stránek. Při prvním pokusu o přístup na chráněnou stránku je uživatel přesměrován na zadanou přihlašovací stránku, kde je formulář pro zadání přihlašovacích údajů. Po úspěšném přihlášení je uživateli přidělena autentikační cookie a je přesměrován zpátky na původní stránku. Nyní se může uživatel pohybovat volně po chráněných stránkách až do vypršení platnosti cookie.[3]

Použití formulářového ověřování ušetří programátorovi velké množství kódu na každé stránce nutného pro manuální ověřování zda je uživatel přihlášen a přesměrování na přihlašovací stránku a zpět. Na druhou stranu má formulářové ověřování i nevýhody, tou nejvýznamnější je, že chrání pouze soubory ASP.NET. Pokud jsou na serveru i jiné soubory, například s příponou .html musí se chránit jinými nástroji. [3]

Implementace formulářového ověřování v aplikaci.

Všechny stránky webového portálu jsou chráněny pomocí formulářového ověřování. Nepřihlášený uživatel se tak dostane pouze na stránku s přihlašovacími údaji. Konkrétní implementace je v souboru `Web.config`, který obsahuje položku `authentication` a ta udává typ ověřování. Ukázka souboru `Web.config` tohoto portálu s implementací formulářového ověřování je na obr.11.

```
<authentication mode="Forms">
      <forms name="loginCookie" loginUrl="LoginForm.aspx" protection="All"
    timeout="30" />
</authentication>
```

Obr. 11: Ukázka ze souboru Web.config aplikace

Element `forms` může mít následující atributy:

- `name` - uvádí jméno vytvořené autentikační cookie
- `loginUrl` - obsahuje url adresu přihlašovací stránky
- `protection` - určuje úroveň ochrany cookie, tato položka může nabývat hodnot:
    - `None` - neprovádí se šifrování ani ověřování pravosti cookie
    - `Validation` - provádí se pouze ověřování pravosti cookie
    - `Encryption` - provádí se pouze šifrování cookie
    - `All` - provádí se jak šifrování tak ověřování pravosti cookie
- `timeout` - platnost cookie v minutách, v tomto případě systém uživatele odhlásí po 30 minutách nečinnosti

Díky formulářovému ověřování se do systému nemůže přihlásit uživatel bez platných přihlašovacích údajů. Protože však se systémem pracuje více uživatelů s různými oprávněními, je formulářové ověřování doplněno o řízení přístupu pomocí rolí. To je implementováno opět v souboru Web.config, kde jsou uvedeny jednotlivé složky prezentační vrstvy a seznam rolí, které mají povolený přístup. Příklad řízení přístupu na základě rolí do složky Dodavatel je na obr.12.

```
<location path="Dodavatel">
    <system.web>
      <authorization>
        <allow roles="dodavatel, admin"/>
       <deny users="*" />
      </authorization>
    </system.web>
  </location>
```

Obr. 12: Příklad řízení přístupu pomocí rolí

6.5.3 Ověřování formulářových polí

Webový portál používá pro ověření správných vstupů ve formulářích ověřovací ovládací prvky, které obsahuje ASP.NET tzv. validátory. Tyto prvky ověřují správnost přiřazených formulářových polí a to jak na klientovi, tak i na serveru. Dvojí kontrola má výhodu v tom, že data není tak snadné podvrhnout a ověřování funguje i když na klientovi není podporován JavaScript. [3]

Existuje několik různých typů ověřovacích prvků:

- `RequiredFieldValidator` - kontroluje, zda není vstupní pole prázdné
- `RangeValidator` - kontroluje, zda je vstupní hodnota v požadovaném rozsahu
- `RegularExpressionValidator` - kontroluje vstupní řetězec podle zadaného regulárního výrazu..
- `CompareValidator` - porovnává vstupní hodnotu se zadanou hodnotou nebo s jiným vstupním polem
- `CustomValidator` - ověřuje vstupní hodnotu podle algoritmu, který si programátor sám naprogramuje [3]

Použití ověřovacích prvků v aplikaci

Pro ověřování povinných formulářových polí je použit `RequiredFieldValidator`, příkladem je formulář pro zadání nového firemního zákazníka do systému. V tomto formuláři jsou všechna pole povinná, každé pole má tedy přiřazeno `RequiredFieldValidator`, který v případě, že pole není vyplněno vypisuje příslušné upozornění pro uživatele.

`RangeValidator` je použit pro ověřování zadaného rozsahu například u položky výše příspěvku ve formuláři pro zadání nového firemního zákazníka. Toto číslo značí výši příspěvku na oběd v procentech, má tedy smysl pouze v rozmezí 0 až 100. (V praxi zaměstnavatelé přispívají maximálně 55% viz kapitola 3.1).

Použití `RegularExpressionValidator` je mnohem širší, protože tento prvek lze do velké míry přizpůsobit konkrétnímu zadání. V aplikaci se pomocí něj ověřuje například zadávání telefonního čísla, které může obsahovat pouze devět čísel a volitelnou předvolbu +420, naopak nesmí obsahovat písmena a jiné speciální znaky.

Ověřování platného IČ firmy

Při přidávání nového firemního zákazníka se ověřuje platné IČ pomocí CustomValidator. Při hledání algoritmu pro ověření IČ jsem narazil na oficiální algoritmus uvedený na stránkách Ministerstva vnitra ČR [18] viz (1). Tento algoritmus uvádí, že kontrolní součet pro prvních sedm čísel IČ dostaneme následujícím způsobem: číslo na pozici `i` vynásob číslem `9−i` a součiny sečti. K tomuto součtu přičteme poslední osmou číslici, přičemž pokud je poslední číslice rovna 0 nahradíme ji konstantou 10. Aby IČ bylo platné musí být výsledný součet beze zbytku dělitelný číslem 11.

$$zbytek = ((\sum_{i=1}^{7} H_i * (9 - i) + H_8) \bmod 11$$
$$\text{if } H_8 = 0 \text{ then } H_8 = 10 \tag{1}$$

Příklady z praxe však ukázali, že tomuto algoritmu nevyhovují všechna existující IČ, například jedna brněnská firma s IČ 25596641 má podle tohoto algoritmu neplatné IČ. Proto je v aplikaci použit mírně modifikovaný algoritmus převzatý ze [17], kterému vyhoví i výjimky z oficiálního algoritmu. Tento algoritmus stejně jako oficiální počítá kontrolní součet prvních sedmi číslic sečtením součinů pozice čísla `i` s číslem `9-i`, k tomuto součtu se však nepřičítá poslední osmá číslice, ale rovnou se dělí 11. Podle výsledného zbytku po dělení pak kontrolujeme hodnotu poslední osmé číslice H_8 přičemž musí platit:

- je-li zbytek 0 nebo 10, pak $H_8 = 1$
- je-li zbytek 1, pak $H_8 = 0$
- v ostatních případech musí být $H_8 = 11$ - zbytek

Pokud poslední číslice nesplňuje ani jednu z těchto podmínek je IČ neplatné.

6.7 Důležité části systému

V této kapitole si popíšeme nejvýznamnější části portálu, které poskytují nejdůležitější služby uživateli.

6.7.1 Objednávání obědů

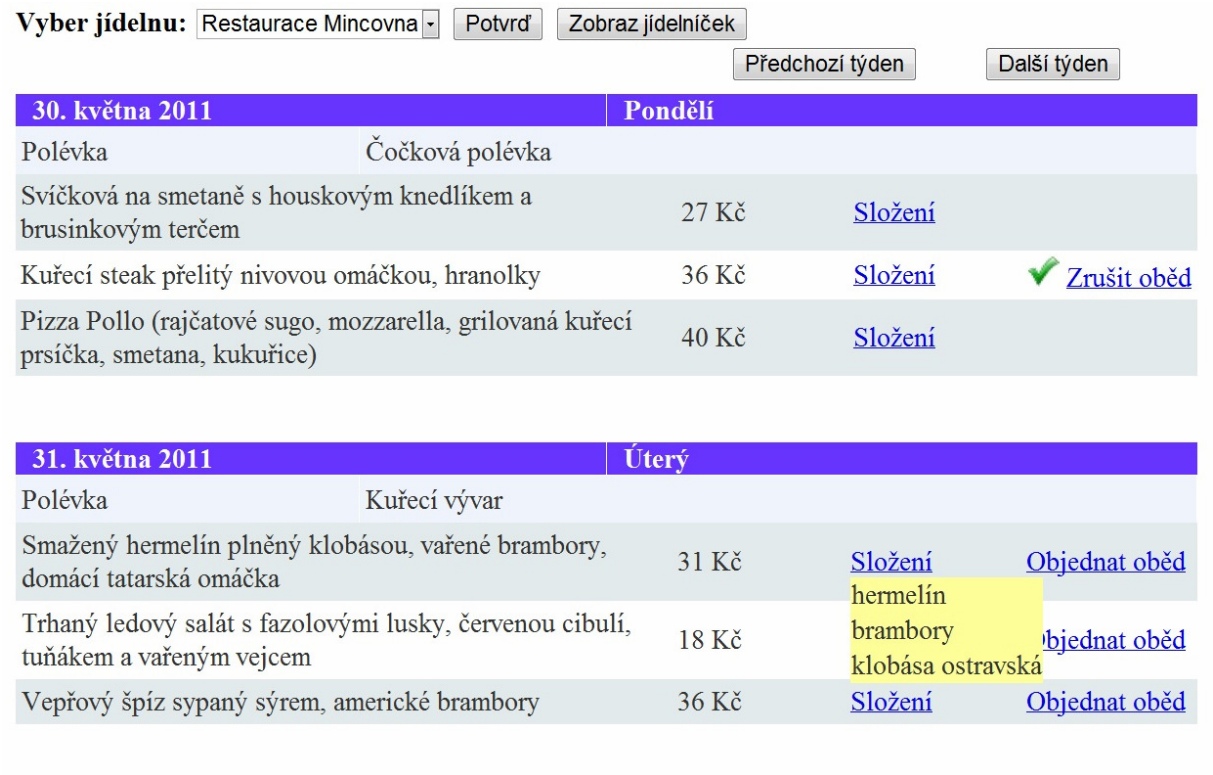
Z pohledu návrhu

Koncept systému v podobě firemního objednávání obědů, předpokládá objednávání obědů pouze u zaměstnanců smluvní firmy. Objednávkový formulář se tak zobrazí pouze uživatelům s rolí zaměstnanec.

Zaměstnanec si může na každý den objednat maximálně jeden oběd. Ke každé objednávce dostane zdarma také polévku dle denní nabídky. Objednávku oběda lze realizovat nejpozději do jedné hodiny odpolední předchozího dne. Nabídka obědů je vypisována na dva týdny dopředu, obědy je tak možno objednávat s předstihem. Jakmile si zaměstnanec oběd objedná, má možnost jej do uzávěrky objednávek zrušit a objednat si jiný. U každého jídla je také tlačítko složení, které zobrazí seznam všech použitých ingrediencí.

Z pohledu implementace

Na stránce `NewOrder.aspx` si zaměstnanec vybere jídelnu a poté se mu zobrazí nabídka obědů pro vybranou jídelnu na aktuální týden. Pomocí tlačítek předchozí a další týden je možno zobrazit jídelníček na minulý týden a na dva nadcházející týdny. Na každý den je vypsáno několik obědů, po objednání oběda se vybrané jídlo zvýrazní a je nabídnuta možnost jej zrušit, která platí až do uzávěrky objednávek. U každého jídla je také tlačítko pro zobrazení složení jídla, které je rozšířeno o ovládací prvek `PopupControlExtender`. Tento prvek zajišťuje dynamické zobrazení seznamu ingrediencí.



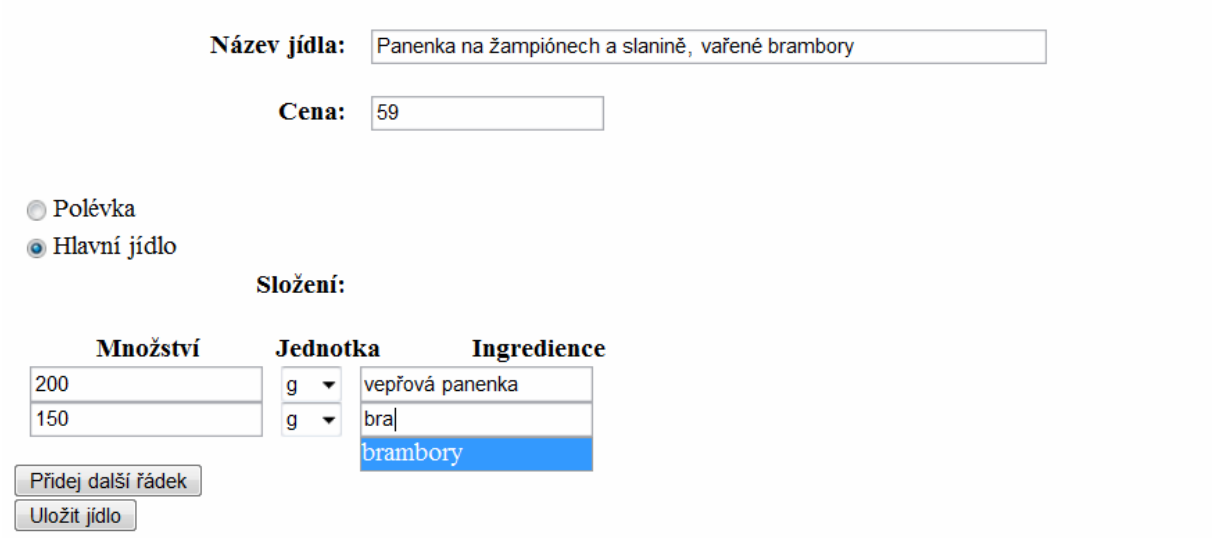
Obr. 13: Stránka pro objednání oběda

6.7.2 Vložení nového jídla

Na této stránce dodavatel zadává do systému nová jídla, která bude přiřazovat k jednotlivým dnům jako obědy. Nejprve zadá název jídla a cenu, poté zvolí zda se jedná o hlavní jídlo nebo o polévku a nakonec zadá složení. Složení se zadává jako seznam ingrediencí ve tvaru: množství, jednotka, ingredience. Protože každé jídlo vyžaduje na přípravu různé množství surovin, obsahuje formulář pro přidání nového jídla tlačítko pro přidání nového řádku, které dynamicky připojí k formuláři další řádek se vstupními poli pro zadání ingredience.

Webová služba AutoComplete.asmx

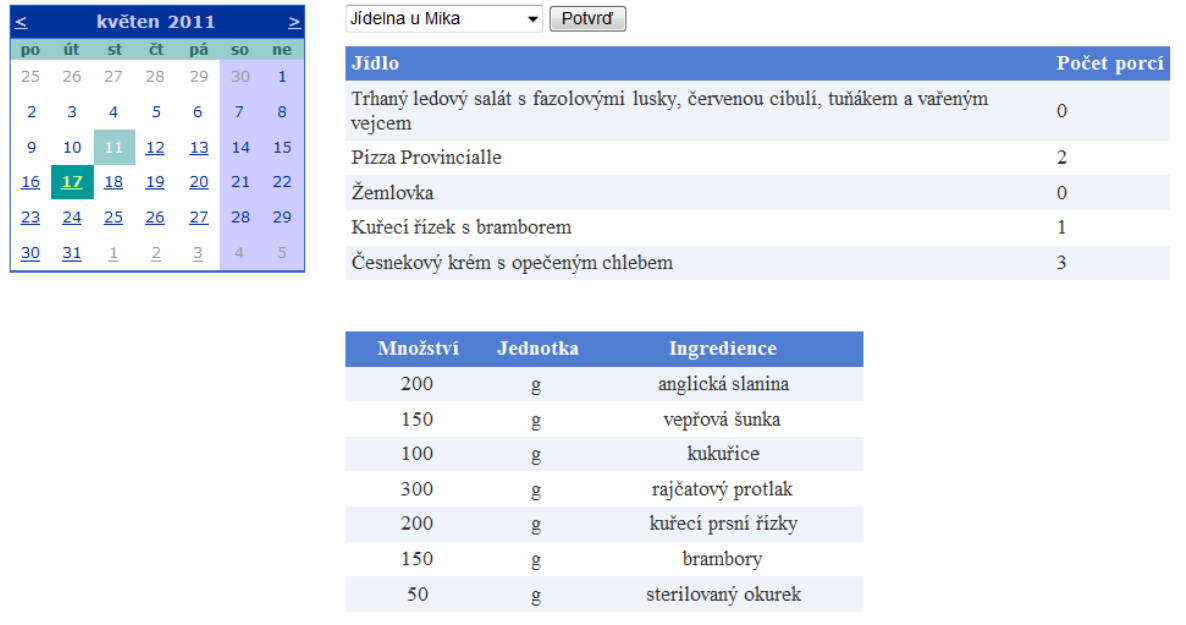
Z důvodu usnadnění zadávání ingrediencí je pole pro zadání názvu ingredience rozšířeno o webový ovládací prvek `AutoCompleteExtender`, ze sady prvků Ajax Control Toolkit. Tento ovládací prvek je propojen s webovou službou `AutoComplete.asmx`, která načítá názvy již existujících ingrediencí z databáze. V aplikaci se to projeví tak, že dodavatel napíše první dvě písmena ingredience a webová služba `AutoComplete` vyhledá v databázi podobné názvy a ty pak nabídne dodavateli jako nápovědu v podobě dropdownlistu. Tato funkce přináší nejen výhodu pohodlnějšího zapisování ingrediencí pro dodavatele, ale zároveň snižuje riziko vložení jedné ingredience se dvěma různými názvy do databáze.



Obr. 14: Stránka pro přidání nového jídla

6.7.4 Zobrazení ingrediencí k nákupu

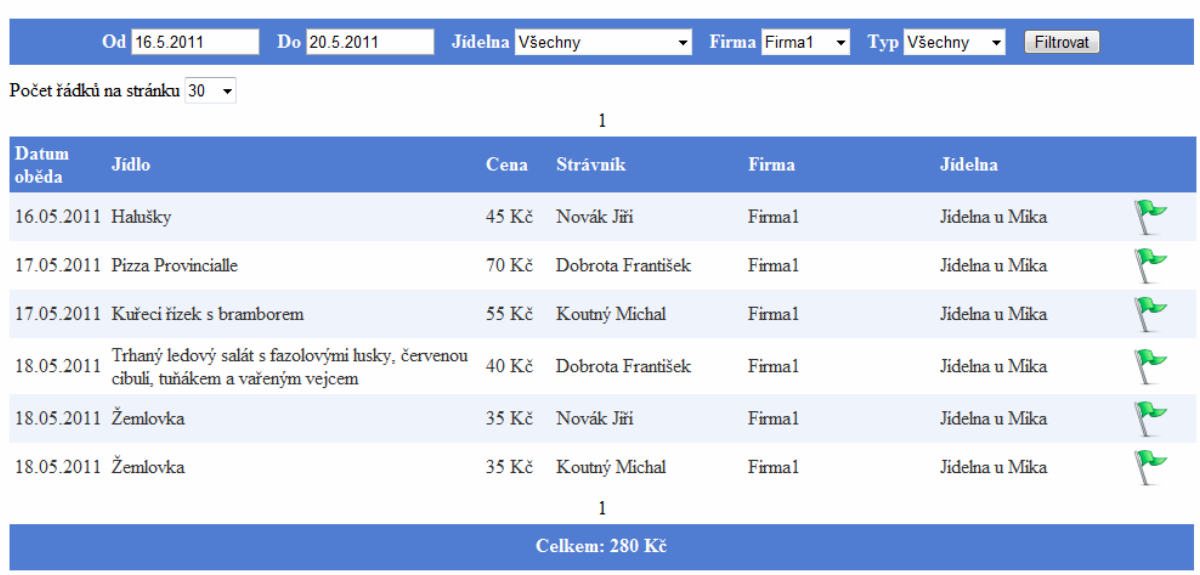
Tato stránka poskytuje dodavateli efektivní způsob pro plánování nákupu potravin. Na této stránce si dodavatel zvolí den, na který bude nakupovat suroviny a jídelnu, do které je bude navážet. Poté se mu objeví seznam obědů vypsaných na zvolený den ve zvolené jídelně a počet objednaných porcí jednotlivých jídel. Ve spodní části stránky se vypíše souhrnný seznam ingrediencí nutných k navaření daného množství porcí.



Obr. 15: Zobrazení ingrediencí k nákupu

6.7.5 Přehled objednávek

Dodavatel má možnost nechat si vypsat přehled všech objednávek a celkovou cenu. Přehled může filtrovat podle data oběda a vypsat tak pouze objednávky za určité období. Dále je možno přehled filtrovat na základě jídelny, která oběd vydává. Lze také vypsat objednávky všech zaměstnanců konkrétní firmy.



Obr. 16: Přehled objednávek

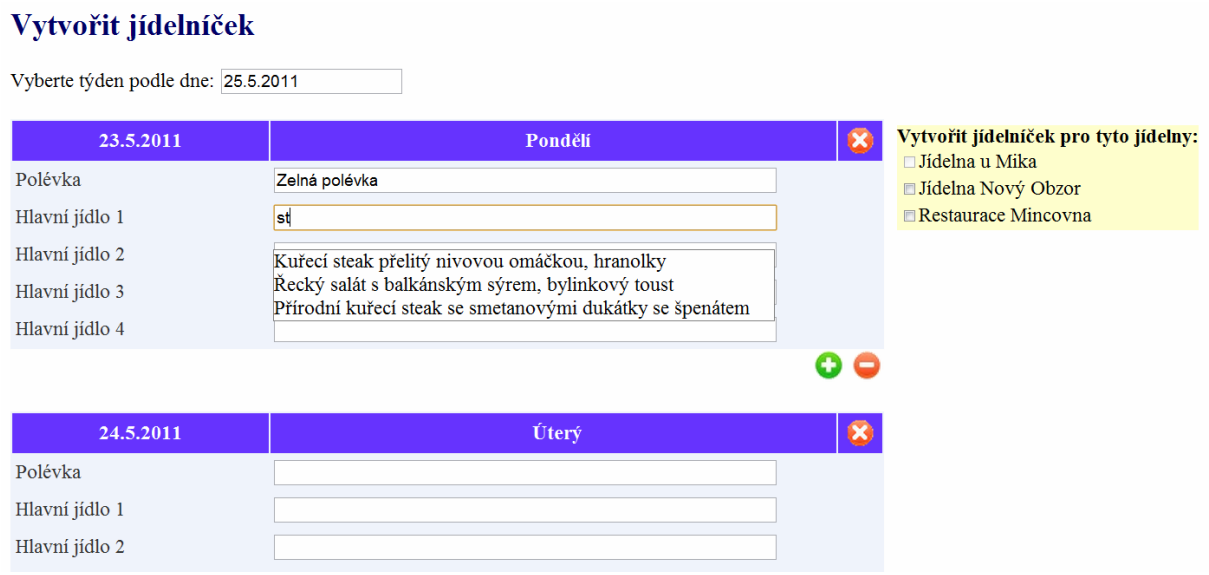
6.7.6 Vytvoření jídelníčku

Z pohledu návrhu

V jídelnách zajišťujících firemní stravování se obědy vypisují v podobě jídelníčku na celý týden, toto je promítnuto do systému jako stránka pro vytvoření týdenního jídelníčku. Dodavatel si nejprve zvolí týden na který chce vypsat jídelníček a poté zvolí pro které jídelny bude tento jídelníček platit. Minimální počet jídel na jeden den je tři maximální počet není omezen. Pokud určitý den bude mít jídelna zavřeno, například z důvodu státního svátku, má dodavatel možnost tento den odstranit z jídelníčku a nevypisovat na něj obědy. Při vytváření jídelníčku potřebuje mít dodavatel přehled o cenách jednotlivých jídel z důvodu plánování cenové politiky.

Z pohledu implementace

Stránka `NewMenu.aspx` obsahuje pět tabulek, každá představuje jeden den v týdnu. Každá tabulka má tlačítka plus a mínus pro dynamické přidávání či odebírání řádku, tak aby bylo možně zadat 3 až n obědů, kde n není nijak omezeno. Každá tabulka má dále v pravém horním rohu křížek pro její skrytí v případě, že dodavatel na tento den nebude vypisovat obědy. Zadávání jídel funguje na principu nápovědy na základě vepsaných písmen pomocí webové služby `AutoComplete.asmx`. Toto řešení je efektivnější oproti použití rozbalovacího seznamu obsahujícího všechna jídla, který by byl neúměrně dlouhý. Po zadání jídla do vstupního pole se ověří zda jídlo v databázi existuje a pokud ne vypíše se chybová hláška. V případě, že je jídlo platné objeví se jeho cena. Na pravé straně stránky se nachází seznam všech jídelen, do kterých dodavatel dodává obědy. Při ukládáni obědů si tak může zvolit jednu či více jídelen pro nichž tento jídelní lístek platí. Odpadá tak nutnost vyplňovat stejný jídelníček pro každou jídelnu zvlášť. Při ukládání jídelníčku se ověřuje zda jsou všechna pole vyplněná, jestli zadaná jídla existují v databázi a zda byla zvolena alespoň jedna jídelna.



Obr.17: Stránka pro vytvoření jídelníčku

6.8 Transformace XML pomocí XSLT

Zobrazení týdenního menu je implementováno pomocí XML a XSLT. Tento postup je výhodný v tom, že pracuje přímo s kolekcí objektů, není tedy nutné implementovat webovou stránku pro zobrazení a řešit, zobrazení informací o objektu. Transformace kolekce obědů na XML umožňuje sdílení jídelníčku ve strukturovaném formátu mezi více jídelnami. Každá jídelna pak má svou vlastní XSLT šablonu pro vytvoření vlastního vzhledu jídelníčku.

Konkrétní postup zobrazení jídelníčku vypadá následovně. Uživatel si vybere týden a jídelnu, pro kterou chce jídelníček zobrazit. Aplikace naplní objekt třídy `LunchCollection` vypsanými obědy. Metoda `GetXml()` poté transformuje kolekci objektů do řetězce ve formátu XML (viz obr.18).

```
<?xml version="1.0" encoding="utf-16"?>
<ArrayOfLunch xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <Lunch>
    <ID>25</ID>
    <MealID>10</MealID>
    <Date>2011-05-23T00:00:00</Date>
    <RestaurantID>1</RestaurantID>
    <Price>0</Price>
    <Name>Zelná polévka</Name>
  </Lunch>
  <Lunch>
    <ID>26</ID>
    <MealID>14</MealID>
    <Date>2011-05-23T00:00:00</Date>
    <RestaurantID>1</RestaurantID>
    <Price>69</Price>
    <Name>Smažený hermelín plněný klobásou, vařené brambory, domácí
tatarská omáčka</Name>
  </Lunch>
...
</ArrayOfLunch>
```

Obr. 18: Ukázka struktury uložení týdenní nabídky obědů do formátu XML.

Tento XML řetězec se poté předá funkci `GetHtml()`, která načte vstupní dokument ve formátu XML spolu se šablonou stylu XSLT a provede transformaci XML na HTML podle pravidel zadaných v šabloně stylu (viz obr.19).

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:msxsl="urn:schemas-microsoft-com:xslt" exclude-result-
    prefixes="msxsl"
    xmlns:ms="urn:schemas-microsoft-com:xslt"
>
    <xsl:template match="/">
    <html>
      <body style="text-align:center">
        <h2>Jídelna U Mika</h2>
        <h3>
          Týdenní menu
          <xsl:value-of select="ms:format-date(ArrayOfLunch/Lunch/Date,
           'dd.MM.yyyy')"/>
          -
          <xsl:value-of select="ms:format-
               date(ArrayOfLunch/Lunch[last()]/Date, 'dd.MM.yyyy')"/>
        </h3>
        <table border="0" width="100%">
          <tr style="font-weight: bold;">
            <td>Datum</td>
            <td>Jídlo</td>
            <td>Cena</td>
          </tr>
          <xsl:for-each select="ArrayOfLunch/Lunch">
            <xsl:sort select="Date" order="ascending"/>
            <xsl:variable name="lastPosition"
             select="position()"></xsl:variable>
            <xsl:if test="Price = 0">
              <tr>
                <td width="10%">
                  <xsl:value-of select="ms:format-date(Date,
                    'dd.MM.yyyy')"/></td>
                <td width="60%" style="font-style: italic">
                  <xsl:value-of select="Name"/>
                </td>
              </tr>
            </xsl:if>
            <xsl:if test="Price &gt; 0">
              <tr>
                <td></td>
                <td width="60%">
                  <xsl:value-of select="Name"/>
                </td>
                <td width="10%">
                  <xsl:value-of select="Price"/> Kč
                </td>
              </tr>
            </xsl:if>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Obr. 19: Ukázka šablony XSLT, která slouží pro formátování XML dokumentu:

Výsledný HTML text obsahující týdenní jídelníček se poté zobrazí na stránce ve formátu vhodném pro tisk a umístění do areálu jídelny v papírové formě. Po transformaci z XML pomocí XSLT vypadá výsledná HTML stránka dle obr.20.

Jídelna U Mika

Týdenní menu 23.05.2011 - 27.05.2011

Datum	Jídlo	Cena
23.05.2011	*Zelná polévka*	
	Smažený hermelín plněný klobásou, vařené brambory, domácí tatarská omáčka	69 Kč
	Kuřecí steak přelitý nivovou omáčkou, hranolky	79 Kč
	Milánské špagety sypané sýrem	69 Kč
24.05.2011	*Čočková polévka*	
	Pizza Pollo (rajčatové sugo, mozzarella, grilovaná kuřecí prsíčka, smetana, kukuřice)	89 Kč
	Kuřecí kapsa plněná špenátem a slaninou, hranolky	79 Kč
	Řecký salát s balkánským sýrem, bylinkový toust	69 Kč
25.05.2011	*Kuřecí vývar*	
	Vepřový špíz sypaný sýrem, americké brambory	79 Kč
	Kuřecí nudličky na kari se zeleninou, rýže	69 Kč
	Svratecký guláš (z hovězí roštěné), bramboráčky	79 Kč
26.05.2011	*Kmínová polévka s vejci*	
	Přírodní kuřecí steak se smetanovými dukátky se špenátem	79 Kč
	Kuřecí steak přelitý nivovou omáčkou, hranolky	79 Kč
	Kuřecí kapsa plněná špenátem a slaninou, hranolky	79 Kč
27.05.2011	*Domácí bramboračka*	
	Vepřový špíz sypaný sýrem, americké brambory	79 Kč
	Smažený hermelín plněný klobásou, vařené brambory, domácí tatarská omáčka	69 Kč
	Řecký salát s balkánským sýrem, bylinkový toust	69 Kč

Všechny ceny jsou uvedeny včetně DPH.
Změna jídelního lístku vyhrazena.

Obr. 20: Zobrazení týdenního jídelníčku

8 Závěr

V této práci je popsán návrh a implementace webového portálu pro zajištění stravování firemním zaměstnancům. Systém je určen pro dodavatele zajišťujícího firemní stravování konkrétně jako nástroj pro jednoduché vystavení jídelníčku na internetu a správu firemních zákazníků a zákazníkových zaměstnanců. Dále poskytuje dodavateli přehled všech objednávek a celkovou cenu, stejně jako přehled objednávek zaměstnanců pouze konkrétní firmy. Jednotlivým strávníkům pak portál poskytuje pohodlné objednávání obědů přímo z kanceláře spolu s evidencí historie objednávek.

Portál splňuje požadavky vytyčené v zadání bakalářské práce. Aplikaci funkčně rozšiřuje evidence ingrediencí potřebných pro přípravu jednotlivých jídel. Seznam ingrediencí se zobrazuje při objednávání obědů, díky čemuž má strávník lepší představu o objednávaném jídle. Dodavateli se pak při výpisu denních objednávek zobrazí celkový seznam všech ingrediencí nutných na konkrétní den. Dále portál umožňuje dodavateli vytvářet jídelníčky pro více jídelen a evidovat objednávky v každé jídelně zvlášť.

Jedním z možných rozšíření je přidání možnosti hodnotit jídlo. Uživatel by zpětně mohl zadat hodnocení jídla na nějaké stupnici a toto hodnocení by se dále uchovávalo v databázi u každého jídla. Za tímto účelem obsahuje databázová tabulka Jídlo sloupec průměrné hodnocení, kde je možné ukládat průměr všech hodnocení daného jídla.

Při návrhu portálu jsem použil objektově orientované metody návrhu zejména jazyk UML. Pro návrh jednotlivých uživatelů systému a jejich akcí byl použit Use Case diagram, pro modelování životního cyklu objektů byl použit Diagram časování a navrhovaná struktura databáze byla vyjádřena pomocí E-R diagramu. Portál byl implementován pomocí objektově orientovaného přístupu na třívrstvé architektuře s použitím nástrojů ASP.NET Framework, MS SQL Server a AJAX Control Toolkit. Při řešení bakalářské práce jsem se s těmito nástroji blíže seznámil a pochopil jejich výhody při programování dynamických webových stránek propojených s databází. Vypracování práce pro mě bylo velkým přínosem, ať už z hlediska pochopení výhod plynoucích z použití návrhových modelů při modelování komplexního informačního systému, nebo díky možnosti detailního seznámení s rámcem ASP.NET, který poskytuje možnost psát strukturované webové projekty zaměřené na modularitu a znovu použitelnost.

Literatura

[1] KANISOVÁ, Hana; MÜLLER, Miroslav. UML srozumitelně. Brno : Computer Press, 2006. 176 s. ISBN 80-251-1083-4.

[2] KOČÍ, Radek; KŘENA, Bohuslav. Úvod do softwarového inženýrství : Studijní opora. Brno : FIT VUT, 2006. 100 s.

[3] PROSISE, Jeff. Programování v Microsoft .NET : Webové aplikace v C#, ASP.NET a .NET Framework. Brno : Computer Press, 2003. 712 s. ISBN 80-7226-879-1.

[4] Strava.cz [online]. [cit. 2011-05-11]. Dostupné z WWW: <http://www.strava.cz/istravne/>.

[5] E-Strava [online]. 2003 [cit. 2011-05-11]. Dostupné z WWW: <https://secure.ulrichsw.cz/estrava/>.

[6] ČR. Úplné znění zákona č 586/1992 Sb., o daních z příjmů. Úplné znění Daně z příjmů 2004. 2004, s. 48-49. Dostupný také z WWW: <http://www.businessinfo.cz/cz/clanek/dan-z-prijmu/c3-p24-zakon-o-danich-z-prijmu/1000652/13565/?fornewsid=13565>.

[7] BusinessInfo.cz [online]. 27.12.2010 [cit. 2011-05-11]. Sazby náhrad za používání vozidel, stravné a průměrné ceny pohonných hmot platné od 1. 1. 2011. Dostupné z WWW: <http://www.businessinfo.cz/cz/clanek/dane-ucetnictvi/nahrady-pouziti-vozidel-prum-cen-hmot-11/1000465/59282/>.

[8] HRUŠKA, Tomáš; BURGET, Radek. Internetové aplikace (WAP) II. část SGML, HTML, CSS, DOM : Studijní opora. Brno : FIT VUT, 2007. 204 s.

[9] HRUŠKA, Tomáš; KROULÍK, Jan; TECHET, Jiří. Internetové aplikace (WAP) III. část XML, XML schémata, XPath, XSLT : Studijní opora. Brno : FIT VUT, 2007. 122 s.

[10] Wikipedia [online]. 2011 [cit. 2011-05-11]. XSLT. Dostupné z WWW: <http://en.wikipedia.org/wiki/XSLT>.

[11] ASP.NET [online]. 2011 [cit. 2011-05-11]. ASP.NET AJAX Control Toolkit. Dostupné z WWW: <http://www.asp.net/ajax/ajaxcontroltoolkit/samples/>.

[12] MSDN Library [online]. 2005 [cit. 2011-05-11]. How To: Protect From SQL Injection in ASP.NET. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/ff648339.aspx>.

[13] MSDN Library [online]. [cit. 2011-05-11]. MD5 Class. Dostupné z WWW: <http://msdn.microsoft.com/en-us/library/system.security.cryptography.md5.aspx>.

[14] Příklady použití diagramů UML 2.0 [online]. 2009 [cit. 2011-05-15]. Diagram časování. Dostupné z WWW: <http://uml.czweb.org/diagram_casovani.htm>.

[15] KŘIVKA, Zbyněk; KOLÁŘ, Dušan. Principy programovacích jazyků a objektově orientovaného programování IPP – II : Studijní opora. Brno : FIT VUT, 2008. 99 s.

[16] ESPOSITO, Dino. ASP.NET a ADO.NET : tvorba dynamických webových stránek. Praha : Grada, 2003. 352 s. ISBN 80-247-0474-9.

[17] GRUDL, David. La Trine [online]. 14. 9. 2007 [cit. 2011-05-22]. Jak ověřit platné IČ a rodné číslo?. Dostupné z WWW: <http://latrine.dgx.cz/jak-overit-platne-ic-a-rodne-cislo>.

[18] RAK, Roman. Ministerstvo vnitra České republiky [online]. 1999 [cit. 2011-05-22]. Unikátní objektové identifikátory. Dostupné z WWW: <http://aplikace.mvcr.cz/archiv2008/casopisy/kriminalistika/1999/9903/rak.html>.