VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
ÚSTAV INFORMATIKY

FACULTY OF BUSINESS AND MANAGEMENT
INSTITUTE OF INFORMATICS

NÁVRH A IMPLEMENTACE DOCHÁZKOVÉHO SYSTÉMU V MALÉM PODNIKU POMOCÍ VBA

DESIGN AND IMPLEMENTATION OF THE TIME AND ATTENDANCE SYSTEM IN A SMALL COMPANY USING VBA

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — RADEK PETRUŠKA
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR DYDOWICZ, Ph.D.
SUPERVISOR

BRNO 2014

Vysoké učení technické v Brně
Fakulta podnikatelská

Akademický rok: 2013/2014
Ústav informatiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Petruška Radek

Manažerská informatika (6209R021)

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách, Studijním a zkušebním řádem VUT v Brně a Směrnicí děkana pro realizaci bakalářských a magisterských studijních programů zadává bakalářskou práci s názvem:

Návrh a implementace docházkového systému v malém podniku pomocí VBA

v anglickém jazyce:

Design and Implementation of The Time and Attendance System in a Small Company Using VBA

Pokyny pro vypracování:

Úvod
Vymezení problému a cíle práce
Teoretická východiska práce
Analýza problému a současné situace
Vlastní návrh řešení, přínos práce
Závěr
Seznam použité literatury

Seznam odborné literatury:

BRADEN, Melanie a Michael SCHWIMMER. Excel 2007 VBA. Velká kniha řešení. Brno: Computer Press, a.s., 2009. 685 s. ISBN 978-80-251-2698-1.
ČIHAŘ, Jiří. 1001 tipů a triků pro Microsoft Excel 2007/2010. Brno: Computer Press, a.s., 2011. 488 s. ISBN 978-80-251-2587-8.
KRÁL, Martin. Excel VBA. Výukový kurz. Brno: Computer Press, a.s., 2010. 504 s. ISBN 978-80-251-2358-4.
KRÁL, Mojmír. Excel 2010 – snadno a rychle. Praha: Grada Publishing a.s., 2010. 143 s. ISBN 80-2473-495-8.
LAURENČÍK, Marek. Programování v Excelu 2007 a 2010. Praha: Grada Publishing a.s., 2011. 192 s. ISBN 978-80-247-3448-4.

Vedoucí bakalářské práce: Ing. Petr Dydowicz, Ph.D.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2013/2014.

L.S.

doc. RNDr. Bedřich Půža, CSc. Ředitel ústavu	doc. Ing. et Ing. Stanislav Škapa, Ph.D. Děkan fakulty

V Brně, dne 27.05.2014

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací docházkového systému v malém podniku. Součástí návrhu je i ekonomická výhodnost řešení pro daný podnik.

Abstract

Main objective of this bachelor thesis is to design and implement time and attendance system in a small company. Economic profitability of this solution is included as a part of the design document.

Klíčová slova

VBA, SWOT, Microsoft Office, Excel, Access, Visual Basic for Applications, Docházkový systém

Key words

VBA, SWOT, Microsoft Office, Excel, Access, Visual Basic for Applications, Attendance and time system

Bibliografická citace bakalářské práce

PETRUŠKA, R. *Návrh a implementace docházkového systému v malém podniku pomocí VBA*. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 80 s. Vedoucí bakalářské práce Ing. Petr Dydowicz, Ph.D.

Čestné prohlášení

Prohlašuji, že předložená bakalářská práce je původní a zpracoval jsem ji samostatně. Prohlašuji, že citace použitých pramenů je úplná, že jsem ve své práci neporušil autorská práva (ve smyslu Zákona č. 121/2000 Sb., o právu autorském a o právech souvisejících s právem autorským).

V Brně dne 25. května 2014 .....

Podpis

Poděkování

Rád bych tímto poděkoval panu Ing. Petru Dydowiczovi za ochotu vést mou bakalářskou práci a za poskytování cenných rad a zpětné vazby po celou dobu naší spolupráce. Dále bych chtěl poděkovat všem zaměstnancům společnosti GISoft, Johnu Pembertonovi a sourozencům Corrovým, bez nichž bych tuto práci nikdy nedokončil.

Obsah

ÚVOD

Docházkové systémy jsou velmi podstatnou součástí informačních systémů podniku. Společnost GISoft v.o.s. však žádné z nabízených řešení dodnes nezavedla do praxe, přestože se jedná o velmi důležitou pomůcku, jak pro vedení, tak pro samotné zaměstnance. S jeho pomocí totiž mohou pracovníci evidovat svou docházku, na jejímž základě jsou následně vypočítávány mzdy. Tento výpočet lze navíc do značné míry automatizovat, takže dochází k celkovému zefektivnění procesů uvnitř organizace.

Proti implementaci takového řešení stojí relativně malá velikost společnosti, která nezaměstnává příliš velký počet zaměstnanců. Kontrola údajů o docházce tedy není prozatím tak náročná, aby se vyplatilo pořizovat drahé řešení, jehož možnosti by nebyly naplno využity.

Existuje zde však možnost interně vytvořit vlastní implementaci tohoto systému s využitím maxima již existujících technologií, čímž by se náklady výrazně snížily. Tato práce si klade za cíl analyzovat reálnost tohoto kroku tím, že kromě samotné implementace rovněž posoudí náklady s realizací spojené.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Cílem práce je navrhnout a implementovat docházkový systém tak, aby splňoval veškeré běžně kladené požadavky na podobné systémy. Během této implementace budou zhodnoceny možnosti využití použitých technologií v podnikovém prostředí. Toto hodnocení bude vytvořeno na základě vlastní implementace docházkového systému v reálném podniku, malé společnosti zabývající se vývojem softwaru.

Před započetím samotné implementace dojde ke zhodnocení technických možností technologií, kterými podnik disponuje a na jejichž základě bude celý systém vybudován. Toto vyhodnocení je nutné provést z toho důvodu, aby se již v průběhu realizace systému neobjevil neřešitelný, nebo jen obtížně řešitelný problém, kterého se šlo vyvarovat detailnější analýzou platformy. Součástí této analýzy bude i výběr vhodné mateřské aplikace, která bude samotný docházkový systém provozovat. Vybrána bude ta aplikace, která se bude nejvíce hodit pro tento typ úkolu, tj. ukládání většího počtu záznamů do předem daných struktur.

Docházkový systém se bude řídit zákonnými normami a z toho důvodu budou související zákony prostudovány v analýze současného stavu, aby pak získané znalosti mohly být přeneseny do návrhu funkcí celého systému. V této části dojde rovněž ke zhodnocení přínosů a rizik zvolené platformy. Základním požadavkem totiž je vytvoření co možná nejlépe udržovatelného systému, jehož provoz si nevyžádá dodatečné náklady. Ke splnění všech požadavků mají pomoci pečlivě stanovená kritéria, jimiž se má aplikace řídit, čímž se hrozba možných rizik výrazně sníží.

Kromě implementace systému, který bude splňovat požadavky, které na něj společnost GISoft klade, bude součástí praktické části této práce i ekonomické zhodnocení celého procesu tvorby. Do tohoto zhodnocení se promítnou veškeré náklady, které bude nutné vynaložit na vytvoření výkonného kódu a všechny potřebné optimalizace, včetně všech přípravných prací na zvolené platformě. Tyto přípravné práce totiž zřejmě zaberou nejvíce času a jsou pro úspěšnou implementaci naprosto klíčové. Nakonec dojde na zhodnocení potenciálního přínosu pro podnik, během kterého bude zjištěno, zda došlo k naplnění očekávání majitelů společnosti.

1 TEORETICKÁ VÝCHODISKA PRÁCE

V teoretických východiscích práce jsou blíže vysvětleny pojmy jako je VBA, včetně přiblížení jeho technických možností a nastínění jeho budoucího použití. Podstatnou částí teoretických východisek je popsání návrhového vzoru SOLID, který bude sloužit jako základ pro jednoduchou udržovatelnost systému.

Vysvětlen bude rovněž pojem docházkový systém jako takový. Dále v práci bude využívána analýza SWOT, proto i ta je v teoretických východiscích objasněna.

1.1 VBA

VBA je programovací jazyk určený k vytváření předpřipravených akcí, tzv. maker, sloužících k automatizaci často opakovaných úkonů prováděných v mateřské aplikaci. Z tohoto důvodu nemůže běžet samostatně, ale vyžaduje, aby byl spuštěn v rámci jiné aplikace, která mu bude poskytovat rozhraní (Lomax, 1998).

1.1.1 Historie VBA

VBA vzniklo jako odnož programovacího jazyka Visual Basic, aby nahradil několik současně používaných makro jazyků. Poprvé se na trhu objevil v roce 1993, kdy byl integrován do aplikace Microsoft Excel (Lomax, 1998), kde měl nahradit do té doby používaný XLM (Green et. al., 2007).

Dalším velkým milníkem v procesu uvedení VBA na trh učinil Microsoft o dva roky později. V té době totiž jazykem VBA nahradil původní makro jazyk Access Basic integrovaný v databázové aplikaci Access, čímž nastartoval masivní nárůst popularity VBA v kancelářských aplikacích (Lomax, 1998).

V následujících letech VBA nahradilo Word Basic v aplikaci Microsoft Word a bylo rovněž integrováno do prezentačního nástroje PowerPoint. Tímto krokem se tak VBA dostalo do všech aplikací z balíku Microsoft Office 97, s výjimkou Outlooku, ve kterém se nadále využíval konkurenční VBScript (Lomax, 1998). Ve stejném roce navíc Microsoft začal poskytovat licence na použití VBA také výrobcům softwaru třetích stran. VBA se tak mohlo snadno rozšířit do všech typů aplikací a objevuje se například i v populárním projektovacím software AutoCAD (HyperPics, 2010).

1.1.2 Technické pozadí VBA

VBA, jakožto jazyk pro tvorbu maker, vyžaduje ke svému spuštění hostitelskou aplikaci. Typickým příkladem takové aplikace je Microsoft Excel a Microsoft Access. Z tohoto důvodu není možné program napsaný ve VBA zkompilovat do samostatně spustitelného souboru, na rozdíl od běžného Visual Basic. VBA je tedy vždy interpretovaný za běhu, kdežto aplikace napsané ve Visual Basic mohou být jak kompilované, tak interpretované (Lomax, 1998).

Přestože VBA vzniklo jako náhrada za klasické makro jazyky, které se zaměřovaly na pouhé automatizování často se opakujících činností, nabízí daleko více možností a volnosti. Jazyk se výhradně neomezuje na použití programových rozhraní nabízených hostitelskou aplikací, ale umožňuje plnou interakci s vnějším okolím pomocí práce s OLE objekty, práci s daty z externích úložišť přes rozhraní ODBC a schopnost ovládat ostatní VBA programy (Lomax, 1998).

1.1.3 Tok programu VBA

Ve VBA existují dva základní typy programových bloků. Prvním je procedura, která je definována svým názvem a nepovinně také parametry. Procedura slouží ke spouštění na sebe logicky navazujících částí kódu.

Na rozdíl od procedury může funkce, která je rovněž definovaná názvem a nepovinně parametry, vracet vypočítanou hodnotu. Obecně lze říci, že se program skládá z jedné hlavní procedury, která následně volá ostatní podprocedury a funkce. (Lomax, 1998)

Pro definování toku programu využíváme ve VBA několik standardních příkazů (Green et. al., 2007):

- If – definuje jednoduchou podmínku.
- Select Case – provádí jednu z definovaných větví kódu na základě testovaného výrazu.
- Do…Loop a For…Next – opakuje část kódu, dokud není splněna ukončovací podmínka (Green et. al., 2007). For…Next je vhodné použít, pokud známe počet opakování dopředu (Liberty, 2007).
- For Each…Next – prochází určenou kolekcí od začátku do konce (Green et. al., 2007).

1.2 Návrhové principy SOLID

Špatně navržený software trpí mnoha problémy, kterým šlo zabránit již při vzniku jeho designu, tedy před samotným započetím programování. Vážnými problémy trpí software především, pokud (Jonáš, 2012):

- Změna programu je velmi obtížná, ne-li dokonce nemožná, a znamená spoustu potenciálních chyb na mnoha místech.
- Změna v kódu ovlivňuje i jiné, naprosto nesouvisející kusy programu.
- Zavedení znovupoužitelného kódu je dražší, ať už časově nebo finančně, než vytvořit kód se stejnou funkčností znovu (Jonáš, 2012).

Tyto problémy lze do značné míry potlačit zavedením a dodržením určitých pravidel, návrhových principů. Jedním z přístupů řešení návrhu softwaru jsou i návrhové principy SOLID. Název se skládá z počátečních písmen pěti základních principů: Single Responsibility Principle, Open-Closed Principle, Liskov Substitution Principle, Interface Segregation Principle a Dependency Inversion Principle (Jonáš, 2012).

1.2.1 Single Responsibility Principle

Single Responsibility Principle (dále jen SRP) je sada návrhových principů, pomáhajících vývojářům softwaru s vytvořením robustních aplikací. Základem SRP je přesvědčení, že každá třída a každý modul v programu by měl odpovídat pouze za jedinou činnost (Jonáš, 2012).

Odpovědností se rozumí důvod ke změně. Pokud existuje více než jeden důvod ke změně třídy a její funkce, je třída odpovědná za více než jednu činnost (Object Mentor, 2006).

Důvod dodržování tohoto principu je nasnadě: v případě, že je třída odpovědná za více činností a dojde ke změně požadavků na funkčnost v jedné z nich, mohou se změny nechtěně projevit také v ostatních činnostech a způsobit nefunkčnost programu (Object Mentor, 2006).

Protože je odpovědnost třídy předem definována, a protože má odpovědnost za jednu činnost, má být dle SRP třída i vhodně pojmenována tak, aby bylo již z názvu na první pohled patrné, k čemu slouží, čímž se do značné míry zlepšuje jak orientace v kódu, tak bezpečnost vytvořeného softwaru, z důvodů menší chybovosti (Jonáš, 2012).

Aplikací principu SRP dochází k rozdělení programu do samostatně funkčních bloků složených z většího množství jednoduchých tříd. Takový program se následně daleko snáze opravuje, případně rozšiřuje o další funkce, protože je snazší najít místo, kde je potřeba provést změnu (Jonáš, 2012).

1.2.2 Open-Closed Principle

Princip otevřenosti a uzavřenosti (dále jen OCP) nabádá k tomu, aby byly všechny třídy v programu rozšiřitelné o novou funkcionalitu, bez nutnosti modifikovat stávající kód. Přidání nové funkce do třídy by tedy nemělo vyžadovat úpravy již existující metody (Jonáš, 2012).

Dodržením OCP jsou minimalizovány hrozby neúmyslného zanesení chyb do hotového kódu. Již existující části aplikace, které danou třídu využívají, prostě dodatečné změny nevyužívají a tím nedochází k ohrožení jejich stability (Jonáš, 2012).

OCP je nejčastěji spojováno s abstrakcí, kdy je funkční kostra modulu abstrahována do základní třídy. V případě potřeby je z ní vytvořen potomek, který již může funkcionalitu rozšířit bez toho, aby jakkoliv ovlivnil jak ostatní potomky, tak samotnou základní třídu (Martin, 2000).

1.2.3 Liskov Substitution Principle

Liskovové princip zaměnitelnosti (dále jen LSP) nám říká, že každá odvozená třída musí být zaměnitelná za svou bázovou třídu. Tento fakt lze ověřit pomocí kódu, který pracuje pouze s bázovou třídou. Pokud je LSP bezezbytku dodržen, bude takový kód pracovat naprosto stejně a bez chyb, i pokud mu bude podstrčena jiná třída, která je z té bázové odvozena. Tento princip se sice může v objektově orientovaném programování jevit jako samozřejmý, ale nemusí tomu tak vždy být. Třídy lze sice technicky zaměnit, ale nikde není zaručena správná funkce kódu v případě, že k této záměně dojde (Martin, 2000).

Aby byly tyto nedostatky odstraněny, zavádí LSP systém podmínek, které musí být splněny před voláním každé metody a po zavolání každé metody odvozené třídy. Mluvíme o tzv. preconditions a postconditions. Každá odvozená třída musí zajistit, aby nevyžadovala více preconditions a nesplňovala méně postconditions než bázová třída. V praxi to znamená, že odvozená třída toho nesmí vyžadovat více a poskytovat méně než její rodič (Martin, 2000).

1.2.4 Interface Segregation Principle

Pokud jedna třída obsluhuje více klientů a každý klient využívá pouze část rozhraní, při každé změně metody využívané jedním klientem je možné, že dané změny nechtěně ovlivní i ostatní klienty. Tento problém řeší Princip odděleného rozhraní (dále jen ISP). Ten říká, že místo vytváření třídy, která sama o sobě obsahuje velké množství vlastních metod, je lepší vytvořit pro každého klienta vlastní rozhraní, které se následně pomocí vícenásobné dědičnosti v hlavní třídě implementuje (Martin, 2000).

Pokud je ISP dodržen a každý klient pracuje pouze se svým rozhraním, jakákoliv změna v tomto rozhraní nijak neovlivňuje ostatní klienty, kteří zase pracují pouze se svým rozhraním. ISP ovšem nepřikazuje, aby bylo pro každého klienta vytvořeno speciální rozhraní. V takovém případě by totiž třída závisela na úplně všech klientech, i když je to v daném případě kontraproduktivní (Martin, 2000). Místo toho se jednotlivá rozhraní mají vytvářet s rozumem tak, aby logicky seskupovala ty metody tříd, které se obvykle používají současně (Jonáš, 2012).

1.2.5 Dependency Inversion Principle

Základním smyslem Principu obrácených závislostí (dále jen DIP) je odstranění závislostí na něčem, co se často mění (Jonáš, 2012).

DIP lze realizovat tak, že každá závislost bude směřovat buď na rozhraní, nebo na abstraktní třídu. V žádném případě by se neměla vytvářet závislost na konkrétní implementaci, neboť hrozí, že se bude implementace v čase měnit a mohla by ovlivnit kód, který ji využívá. V případě rozhraní a abstraktních tříd však časté změny nejsou předpokládány (Martin, 2000).

Přestože je doporučováno využívat DIP v maximální možné míře, nejlépe vždy, existují i výjimky, kdy je možné tento princip ignorovat. Jedná se zejména o situace, ve kterých jsou využívány standardizované moduly, u kterých lze předpokládat, že jsou neměnné. V takovém případě je ale nutné brát v potaz, že je sice zachována neměnnost, ale to samé už nemusí platit o nezaměnitelnosti, což může výrazně ztížit úpravu programu ve chvíli, kdy se změní požadavky na jeho funkci (Martin, 2000).

1.3 Jazyk SQL

Pro potřeby manipulace s daty uloženými v relační databázi vznikl jazyk SQL. Původní návrh na nový jazyk vzešel od tvůrce myšlenky relačních databází Edgara Codda, tehdy ještě pod názvem DSL/Alpha. Později společnost IBM vytvořila pracovní skupinu, která přišla se zjednodušenou verzí původního jazyka a pojmenovala jej SEQUEL, jež byl následně přejmenován na kratší SQL (Beaulieu, 2009).

Americký standardizační institut ANSI publikoval v roce 1986 první standard jazyka SQL, který v následujících letech dále rozvíjel. Dnes už SQL poskytuje možnosti, jak pro práci s daty využívat objektově orientovaný přístup, nebo jak provádět dotazy do XML dokumentů uložených v databázi (Beaulieu, 2009).

Jazyk SQL patří mezi tzv. neprocedurální jazyky, tzn., že při jeho použití definujeme pouze požadované výsledky a způsob, jak jich dosáhnout přenecháváme databázovému stroji. Jazyk SQL je rozdělen do několika tříd příkazů, mezi které patří (Beaulieu, 2009):

- Příkazy pro práci se schématem databáze.
- Příkazy pro manipulaci s daty.
- Příkazy pro práci s transakcemi (Beaulieu, 2009).

1.4 Automatické testy

Pro potřeby zefektivnění vývoje softwaru bývají často využívány tzv. automatické testy, jednoduché kusy kódu kontrolující výstupy programu. Jednotkové neboli unit testy jsou jedním z druhů automatických testů, jejichž smyslem je testovat správné chování nejmenší programové jednotky (unit), která není závislá na žádné jiné (Hammil, 2004).

O správu sad unit testů se stará unit testovací framework. Existuje celá řada takových frameworků, které se mohou v některých aspektech mírně lišit, v zásadě jsou to ale softwarové nástroje umožňující vytváření a spouštění unit testů s následným zobrazením výsledků (Hammil, 2004).

Jednotkové testy jsou často vytvářeny během nebo dokonce ještě před započetím vlastního vývoje, ale obvykle stojí mimo samotný produkční kód. Testy tedy sice využívají a testují funkce programu, ale protože nejsou jeho součástí, zůstává produkční kód přehledný a výsledná velikost je menší, než kdyby tomu tak nebylo (Hammil, 2004).

1.5 Docházkový systém

Součástí informačního systému společnosti bývá zpravidla i docházkový systém. Ten slouží k evidenci a kontrole docházky zaměstnanců. Díky tomu nadřízení v podniku přesně vidí, který z pracovníků nedosahuje smluvené výše odpracovaných hodin a mohou tento problém efektivně řešit (OR-CZ, 2013).

Oproti původním papírovým docházkovým systémům se ty elektronické liší především ve větších možnostech dodatečných úprav a vylepšení. Tyto systémy jsou často napojeny přímo na systém mezd, čímž lze výpočet a vyplácení mezd zaměstnancům částečně automatizovat a zefektivnit. Díky této modulárnosti lze ale systém rozšířit i o další možnosti. Docházkové systémy tak mohou kontrolovat i návštěvy, evidovat přidělená firemní vozidla, případně sloužit též jako přístupový systém. V takovém případě jsou pak zaměstnanci vybaveni malým, přenosným zařízením, typicky kartou, s jejíž pomocí se mohou dostat do firemních prostor přes zabezpečené terminály (OR-CZ, 2013).

1.5.1 Základní funkce docházkového systému

Primární funkcí docházkového systému je zaznamenávat pracovní dobu každého zaměstnance. Kromě této funkce je po těchto systémech ovšem dále požadováno (OR-CZ, 2013):

- Úplná kompatibilita s platnými zákony a normami, které jakkoliv ovlivňují danou problematiku.
- Možnost exportovat zaznamenaná data pro následné zpracování mezd.
- Modularita řešení s možností dodatečných úprav podle potřeb uživatele.
- Zpřístupnění vlastní docházky každému zaměstnanci.
- Editace údajů nadřízenými pracovníky a s tím spojené nastavení oprávnění pro jednotlivé uživatele.
- Kontrola docházky zaměstnanců nadřízenými pracovníky prostřednictvím vyexportovaných sestav.
- Automatické opravy v případě neúplných údajů s následným upozorněním odpovědných osob (OR-CZ, 2013).

1.5.2 Licencování

Oblíbeným platebním modelem výrobců docházkových systémů je poskytování licencí na určitý počet uživatelů. V tomto případě jsou počty uživatelů odstupňovány do balíčků pro určitý počet uživatelů, přičemž cena za každou licenci s rostoucím počtem objednaných licencí klesá. Některé společnosti, které docházkové systémy hostují na svých serverech, pak mohou vyžadovat pravidelné platby za každého uživatele, výměnou za poskytování bezúdržbové služby (ANeT-Advanced Network Technology, 2011).

Tabulka č. 1: Ukázka způsobu licencování hostovaného docházkového systému

Popis	Měsíční platba	Čtvrtletní platba
Služba A – do 10 osob	480,-	1200,-
Služba B – do 25 osob	680,-	1700,-
Služba C – do 50 osob	880,-	2200,-
Služba D – každých 10 osob nad službu C	280,-	700,-

(Zdroj: ANeT-Advanced Network Technology, 2011)

Ať už je systém hostovaný u poskytovatele služby nebo lokálně, do ceny pořízení se velkou měrou promítají náklady na pořízení potřebného hardwaru v případě, že se podnik rozhodne nerealizovat svůj docházkový systém jako čistě softwarový (ANeT-Advanced Network Technology, 2011).

1.5.3 Způsob realizace docházkového systému

K poskytnutí základních funkcí není potřeba žádného dodatečného hardwaru, ovšem v případě, že jsou požadovány nadstandardní služby, je vytvoření odpovídající infrastruktury nezbytností.

Čistě softwarové docházkové systémy

Pro menší podniky, které nevyžadují napojení na přístupový systém, a které chtějí na pořízení evidence docházky vynaložit minimální finanční prostředky, se jako ideální volba jeví řešení postavena jako čistě softwarová. Tyto produkty musí podporovat úplnou funkčnost bez hardwarových terminálů (BM Software, n.d.).

Často lze takové systémy ovládat přes uživatelské rozhraní ve webovém prohlížeči. Zaměstnanci v něm zaznamenávají příchody, odchody a všechny ostatní události

ovlivňující jejich pobyt na pracovišti. Jejich nadřízení pak odsud mohou provádět dodatečné úpravy, tisknout sestavy a pracovat se mzdami (BM Software, 2013).

Docházkové systémy s využitím terminálů

Pro větší pohodlí zaměstnanců jsou docházkové systémy napojeny na hardwarové terminály, které se typicky umisťují přímo ke vchodům na pracoviště. Tyto terminály tak pomáhají zabránit opomenutí pracovníků zaznamenat si příchod či odchod z práce, protože jsou dostatečně výrazné. Nutností se terminály stanou ve chvíli, kdy nejsou pracovníci vybaveni počítačem nebo vůbec nejsou v jeho blízkosti. V takovém případě jsou terminály jedinou možností, jak evidenci docházky realizovat (BM Software, n.d.).

S použitím terminálů roste i pořizovací cena celého systému, kvůli zvýšeným nákladům na potřebný hardware, který čítá kromě samotných terminálů rovněž potřebnou kabeláž a přístupové karty (RON Software, 2013). V případě, že podnik nechce zaměstnance vybavit vlastními kartami, je možné vytvořit systém, pracující na principu snímání otisku prstů (OR-CZ, 2013) nebo rozpoznávání obličejů. Tyto varianty ale cenu pořízení systému ještě zvyšují (RON Software, 2013).

1.6 Microsoft Access

Jedním z mnoha řešení systému pro správu báze dat (zkráceně DBMS) je i nástroj Access vyvíjený společností Microsoft, jenž se stal součástí balíku kancelářských aplikací Microsoft Office. Access ale neslouží pouze ke správě dat a struktur, které tato data uchovávají, ale také jako jejich editor a designer. Jako takový nabízí Access možnost vytvářet a upravovat (Microsoft Corporation, 2007):

- Tabulky a data v nich uložená.
- Formuláře sloužící k interakci s uživateli.
- Sestavy shrnující data do požadovaných celků.
- Dotazy přímo manipulujícími s daty v tabulkách.
- Makra a moduly, které mohou pomocí jazyka VBA rozšířit aplikační logiku databáze (Microsoft Corporation, 2007).

1.6.1 Historie Microsoft Access

DMBS Access je pojmenován po zastaralém terminálovém emulátoru Microsoft Access. Microsoft byl často soudně napadán kvůli porušení registrovaných značek, a proto raději zvolil možnost znovuoživit svou, již dávno nepoužívanou, značku, u které nebezpečí soudního sporu nehrozilo (Microsoft Corporation, 2006).

Prapůvod aplikace Access můžeme nalézt v databázovém systému Omega. Omega byl experimentální systém, který měl v budoucnu nahradit Microsoftem do té doby využívanou databázi Rbase. Z projektu Omega nakonec vzešel Jet engine, který je v Accessu využíván jako výchozí databázový stroj a byl uveden již v první vydané verzi Access 1.1 (Goodhew, 1996).

Access byl veřejnosti představen na výstavě COMDEX v roce 1992 v době, kdy 85% trhu s databázemi pro osobní počítače ovládala společnost Borland, zatímco Microsoft na trhu nepůsobil vůbec. Již o dva roky později byl vydán Access 2.0 (Chung, 2002) a vývoj pokračuje až do dneška, kdy se na trhu objevila poslední verze Access 2013 (Microsoft Corporation, 2012).

1.6.2 Systém uložení dat

Nejnovější Access 2013 umožňuje ukládat data hned dvěma způsoby. Data mohou být uložena klasicky přímo do souboru s příponou accdb, kde se o správu dat stará databázový stroj Microsoft Jet, nebo mohou být součástí Microsoft SharePoint 2013 serveru, kde jsou uložena přímo v Microsoft SQL Server 2012 (Microsoft Corporation, 2012).

Pokud je zvolena první možnost uložení dat, je při vzniku nové databáze vytvořen jediný soubor na disku, ve kterém jsou obsaženy všechny objekty, jako jsou tabulky, sestavy a formuláře, společně s daty (Microsoft Corporation, 2007). Tento způsob uložení dat představuje architekturu File-Server, kdy na serveru je uložen pouze soubor s daty a samotnou manipulaci s nimi a řešení konfliktů při vícenásobném přístupu musí řešit aplikace Access na každém klientovi zvlášť, čímž dochází k rapidnímu nárůstu požadavků a odpovědí mezi klientem a serverem (Microsoft Corporation, 2000).

Na druhou stranu, využitím možnosti uložit data přímo v Microsoft SharePoint 2013 je zaručeno, že všechny operace budou vždy řízeny serverem, tzn., že bude využita architektura Klient-Server. Ukládání dat na SharePoint serveru má i další výhody. Tou

největší je automatické zpřístupnění databáze přes webový prohlížeč, ve kterém mohou uživatelé pracovat jak se sestavami, tak s formuláři. Navíc, protože jsou data uložena přímo v Microsoft SQL Server 2012, mohou být na základě této databáze jednoduše postaveny další aplikace, které nebudou závislé na Accessu (Microsoft Corporation, 2012).

1.7 Analýza SWOT

Pro posouzení plánu na implementaci docházkového systému bude využita analýza SWOT. Název je odvozen od způsobu, jakým k analýze dochází: hodnotí se totiž silné stránky (strengths), slabé stránky (weaknesses), příležitosti (opportunities) a hrozby (threats). SWOT analýza je velmi oblíbená v oblasti marketingu, ale lze ji s úspěchem použít na téměř jakoukoliv činnost spojenou s podnikáním (Rothwell, 2010).

1.7.1 Faktory vnitřního prostředí

Silné a slabé stránky jsou vnitřní záležitostí podniku. Do této kategorie spadá analýza všech významných oblastí, kterými jsou např. systémy řízení a informační systémy, organizační struktura, zaměstnanci, dostupné zdroje a finance a ekonomika (Grasseová, 2007).

1.7.2 Faktory vnějšího prostředí

Z vnějšího prostředí působí na podnik příležitosti a hrozby. Na podnik působící ve veřejném sektoru má vliv prostředí politicko-ekonomické, legislativní, ekonomické, demografické, technicko-ekonomické a ekologicko-ekonomické (Grasseová, 2007).

1.7.3 Provedení SWOT analýzy

Pro realizaci SWOT analýzy neexistuje přesně stanovený postup. Obvyklými kroky při její sestavení jsou (Grasseová, 2007):

1. Zjištění a vyhodnocení silných a slabých stránek.
2. Zjištění a vyhodnocení příležitostí a hrozeb.
3. Grafické znázornění SWOT do matice (Grasseová, 2007).

Jakmile je SWOT analýza dokončena, je potřeba vyhledat nejslabší stránky a největší hrozby a přiřadit jejich řešení největší prioritu. Zároveň s tímto krokem se rovněž

identifikují nejsilnější stránky spolu s největšími příležitostmi. Právě tyto silné stránky a příležitosti totiž mohou sloužit k eliminaci slabých stránek a ke snížení hrozeb (Rothwell, 2010).

1.7.4 Nevýhody využití modelu SWOT

Při interpretaci SWOT analýzy může docházet k dezinterpretaci výsledků, v případě, že se porovnávají pouhé součty silných a slabých stránek, případně příležitostí a hrozeb. Správně by se měly obě strany porovnávat podle závažnosti, což je ovšem často velice obtížné (Rothwell, 2010).

2 ANALÝZA PROBLÉMU A SOUČASNÉ SITUACE

Součástí analýzy současného stavu bude představení společnosti GISoft v.o.s. a jejího dosavadního systému pro správu docházky. Následně budou popsány veškeré požadavky, které tato společnost klade na nový docházkový systém a legislativa, která implementaci systému ovlivní.

Vybrané technologie budou poté zhodnoceny a podrobeny detailnější analýze, která bude následně využita v návrhu řešení, tak, aby byly splněny všechny požadavky a eliminovány nejvážnější hrozby, které z použití těchto technologií vyplynou.

2.1 GISoft v.o.s.

Společnost GISoft v.o.s. byla založena v roce 1995 a sídlí ve městě Opava. Hlavní činností tohoto podniku je tvorba informačních systémů, jejichž účelem je získávání, ukládání a další zpracování geografických dat, pro něž se vžila zkratka GIS (geographic information system), a systémů pro správu technické dokumentace (GISoft, n.d.).

Vizí společnosti je snaha poskytnout svým zákazníkům co možná nejintuitivnější aplikace, které mohou obsluhovat uživatelé pohybující se v oblasti geografie, kteří však často nebývají počítačovými odborníky. Svými produkty proto pomáhají uživatelům zvyšovat efektivitu práce a potenciálně snižovat zákazníkovy náklady (GISoft, n.d.).

2.1.1 Činnost podniku

Gisoft se zabývá nasazováním GIS systémů u svých zákazníků, kterým nabízí kompletní řešení, od implementace geografických a projekčních systémů, implementace technologických linek CAD a vlastních modulů, až po poskytování softwarové podpory a případných konzultací v oblasti výběru GIS a CAD systémů (GISoft, n.d.).

Své produkty postavil GISoft na základech aplikací celosvětově známé společnosti Bentley Systems, jejímž obchodním partnerem se stal již v roce svého založení a v současné době je certifikovaným Bentley Product Sales Partner a Bentley Technology Partner. GISoft je tak autorizován k prodeji jejich produktů, které může zároveň jako technologický partner upravovat a doplňovat o nové funkce (GISoft, n.d.):

- Foundation – klientské i serverové aplikace z produktových řad MicroStation, Bentley Publisher, atd., které tvoří jádro jednotlivých řešení.

- Geospatial – klientské i serverové aplikace z řady Geospatial.
- Academic – výukové verze produktů určené pro školy.
- SELECT – program rozšířené podpory (GISoft, n.d.).

2.1.2 Stav informačního systému

V současné době nemá GISoft na svůj informační systém napojen žádný software pro evidenci docházky. Předpokládá se tedy, že všichni zaměstnanci tráví každý den v práci přesně 8 hodin a případné výjimky se řeší ústní dohodou pro každý konkrétní případ. Vzhledem k velikosti společnosti je tento způsob sice stále udržitelný, nicméně nepohodlný a značně neefektivní.

Evidence a kontrola docházky je přitom snadno realizovatelná, protože všichni zaměstnanci mají svůj osobní počítač napojený do podnikové sítě. Díky tomu by se mohla nezbytná administrativa značně urychlit a zefektivnit. To je také důvod, proč chce GISoft docházkový systém zavést.

2.1.3 Požadavky na docházkový systém

GISoft na nový docházkový systém klade několik požadavků, které musí být splněny, aby jej bylo možné v podnikovém prostředí používat. Tyto požadavky lze rozdělit do 3 skupin.

Požadavky na funkce docházkového systému

Nejkritičtější podmínkou je úplná kompatibilita s platnými českými zákony, především se zákoníkem práce. Převážně se však bude docházkový systém orientovat na tyto oblasti:

- Evidence denní docházky do práce a z práce, včetně zaznamenávání přestávek, s následným výpočtem odpracované doby.
- Evidence pracovních cest.
- Evidence dovolených.
- Práce přesčas nejsou u zaměstnanců povoleny.
- Systém musí upozorňovat jak konkrétního uživatele, tak administrátory na nesrovnalosti v docházce.
- Možnost exportu získaných dat do účetního softwaru.

Požadavky na technické řešení

Protože podnik až do současnosti žádnou softwarově vedenou evidenci docházky neprovozoval, panují obavy především z nákladů na provoz a z obtížnosti dodatečných úprav systému pro případ, kdy bude nutné systém modifikovat. Aby se předešlo problémům, požaduje vedení GISoftu následující:

- Minimální náklady spojené s pořízením i provozováním samotného systému, nejlépe s využitím stávajících systémů.
- Snadná modifikovatelnost pro případné změny v legislativě.
- Rozšiřitelnost o další funkce související s evidencí docházky.
- Současný přístup více uživatelů k datům.

Požadavky na uživatelskou přívětivost

Nový způsob zaznamenávání docházky vyžaduje kooperaci systému s uživateli, což na ně klade dodatečné povinnosti. Z tohoto důvodu je nutné vytvořit systém tak, aby zaměstnance nezdržoval a aby v jeho zavedení viděli smysl. Konkrétní možnosti systému v oblasti práce s uživateli jsou definovány takto:

- Zabezpečení dat proti neoprávněnému čtení.
- Rozdělení uživatelů do rolí, určující jejich práva tak, aby řadoví zaměstnanci byli schopni vidět pouze své záznamy a administrátoři mohli záznamy v případě nutnosti editovat.
- Jednoduché a intuitivní ovládání, které nebude na zaměstnance klást přehnané nároky.

2.2 Zákoník práce

Zákoník práce je hlavním zákonem České republiky, který upravuje vztahy zaměstnance a zaměstnavatele a definuje pro ně pravidla, kterými se musí řídit. Parlament České republiky tento zákon přijal v roce 2006, ale od té doby byl již několikrát novelizován (ČESKO, Zák. č. 262/2006 Sb.).

Tato kapitola popisuje jednotlivé části zákoníku práce, které mohou ovlivnit implementaci docházkového systému a nastíní požadavky vedení společnosti GISoft, jak a zdali vůbec mají být tyto části implementovány.

2.2.1 Pracovní poměr a dohody o pracích mimo pracovní poměr

Zákon umožňuje, kromě standardního pracovního poměru, mezi zaměstnancem a zaměstnavatelem uzavřít také tzv. dohody o pracích konaných mimo pracovní poměr. Jedná se o (ČESKO, Zák. č. 262/2006 Sb.):

- Dohodu o provedení práce
- Dohodu o pracovní činnosti (ČESKO, Zák. č. 262/2006 Sb.)

Všechny tři typy pracovních úvazků se vyznačují rozdílnými parametry, se kterými je potřeba v rámci docházkového systému počítat, aby byla zajištěna jeho stoprocentní kompatibilita se zákonem.

V současné době nemá se společností GISoft žádná osoba uzavřenu dohodu o provedení práce nebo dohodu o pracovní činnosti. Z tohoto důvodu vedení podniku rozhodlo, že systém nemusí být schopen jednotlivé rozdílnosti zpracovat a bude se tedy ke všem zaměstnancům automaticky chovat, jako k zaměstnancům ve standardním pracovním poměru. Na druhou stranu ale vedení podniku požaduje, aby byl systém od začátku připraven na možnost změny tohoto rozhodnutí. V návrhu se proto s různými typy pracovních poměrů bude muset počítat.

2.2.2 Pracovní doba

Zákon říká, že pracovní dobou je myšlena doba, po kterou je zaměstnanec povinen vykonávat práci pro zaměstnavatele, a doba, po kterou je zaměstnanec na pracovišti připraven k výkonu práce podle pokynů zaměstnavatele (ČESKO, Zák. č. 262/2006 Sb.).

Zaměstnanci v pracovním poměru mají stanovenou týdenní pracovní dobu, která činí 40 hodin týdně. V zákoně jsou navíc uvedeny výjimky pro pracující v třísměnném nebo dvousměnném pracovním režimu a pro pracující v podzemí při těžbě uhlí, rud a nerostných surovin, v důlní zástavbě a na báňských pracovištích geologického průzkumu, které jsou však pro společnost GISoft irelevantní a v návrhu se s nimi proto nebude počítat (ČESKO, Zák. č. 262/2006 Sb.).

Zákon rovněž umožňuje sjednat kratší pracovní dobu pro konkrétního zaměstnance (ČESKO, Zák. č. 262/2006 Sb.). Tuto výjimku nebude potřeba v rámci implementace systému nutné řešit, protože do chodu systému nijak nezasáhne.

Ve společnosti GISoft je využíváno pružné rozvržení pracovní doby. Pro toto rozvržení platí, že zaměstnavatel určí základní časové úseky pracovní doby, během kterých musí být zaměstnanec přítomen na pracovišti, a úseky volitelné pracovní doby, ze kterých si může zaměstnanec libovolně vybírat. Celková délka směny však nesmí překročit 12 hodin nepřetržité práce a pro mladistvé do 18 let nesmí překročit 8 hodin nepřetržité práce (ČESKO, Zák. č. 262/2006 Sb.).

V případě, že se nejedná o pracovní poměr, ale o dohodu o provedení práce nebo dohodu o pracovní činnosti, je uplatňována pouze jediná podmínka, a tou je omezení výkonu práce na maximálně 12 hodin během 24 hodin po sobě jdoucích. Obě dohody pak mají následující omezení (ČESKO, Zák. č. 262/2006 Sb.):

- Dohoda o provedení práce – rozsah práce nesmí překročit 300 hodin v témže kalendářním roce.
- Dohoda o pracovní činnosti – rozsah práce nesmí překročit polovinu týdenní pracovní doby, tj. 20 hodin týdně (ČESKO, Zák. č. 262/2006 Sb.).

2.2.3 Přestávky v práci

Zákon přikazuje zaměstnavatelům poskytnout svým zaměstnancům nejdéle po 6 hodinách nepřetržité práce přestávku na jídlo a oddech. Tato přestávka musí mít trvání nejméně 30 minut. V případě, že je zaměstnanec mladiství, zkracuje se prodleva mezi přestávkami na 4,5 hodiny. Tyto přestávky se nezapočítávají do pracovní doby (ČESKO, Zák. č. 262/2006 Sb.).

Legislativa umožňuje přestávku v práci na jídlo a oddech rozdělit do několika částí. V takovém případě je však nutné zajistit, že jedna z těchto částí bude trvat alespoň 15 minut. Dále je stanoveno, že přestávku nelze poskytnout na začátku a konci pracovní směny (ČESKO, Zák. č. 262/2006 Sb.).

Přestávky v práci musejí být poskytnuty zaměstnancům v pracovním poměru. V případě dohody o provedení práce nebo dohody o pracovní činnosti není zaměstnavatel povinen zaměstnancům přestávky poskytovat, musí pouze zajistit, aby výkon práce nepřesáhl 12 hodin během 24 hodin (ČESKO, Zák. č. 262/2006 Sb.).

Požadavkem společnosti GISoft je zajistit automatické přidělování přestávek v práci všem zaměstnancům tak, aby bylo vyhověno platným zákonům. Zaměstnanci sami nejsou nijak omezováni v tom, kdy si přestávku v práci vyberou, ale systém bude muset zajistit, aby byla zaměstnancům, v případě nutnosti, automaticky připočtena přestávka po šesti odpracovaných hodinách, pokud si ji sami nevyberou.

Vedení podniku chce speciálně řešit možnost rozdělení přestávek do více částí. Systém tedy bude muset být schopen kontrolovat, zda některá z částí dosáhla délky 15 minut a v opačném případě zbývající čas přičíst.

2.2.4 Odpočinek a nepřetržitý odpočinek

Povinností zaměstnavatele je rozvrhnout pracovní dobu tak, aby měl zaměstnanec mezi směnami odpočinek po dobu alespoň 11 hodin v případě dospělých a alespoň 12 hodin v případě mladistvých zaměstnanců (ČESKO, Zák. č. 262/2006 Sb.).

Doba odpočinku může být zkrácena až na 8 hodin během 24 hodin, pokud je zaměstnanec starší 18 let a následující doba odpočinku mu bude prodloužena o dobu zkrácení tohoto odpočinku (ČESKO, Zák. č. 262/2006 Sb.). Podmínky, které toto zkrácení doby odpočinku umožňují, a které jsou aplikovatelné ve společnosti GISoft jsou následující:

- Jedná se o nepřetržitý provoz, nerovnoměrnou pracovní dobu nebo práci přesčas.
- Jsou zapotřebí naléhavé opravné práce.
- Jde-li o živelní událost nebo jiný mimořádný případ (ČESKO, Zák. č. 262/2006 Sb.).

Kromě povinnosti zaměstnavatele poskytnout zaměstnancům odpočinek mezi směnami v délce 11, resp. 12 hodin, je zaměstnavatel povinen rozvrhnout pracovní dobu tak, aby měl zaměstnanec v rámci jednoho týdne nepřetržitý odpočinek po dobu alespoň 35 hodin, resp. alespoň 48 hodin v případě, že je zaměstnanec mladší 18 let. Protože to situace v GISoftu umožňuje, stanovuje se všem zaměstnancům podle zákona nepřetržitý odpočinek na stejný den tak, aby do něj spadala neděle (ČESKO, Zák. č. 262/2006 Sb.). V tomto konkrétním případě je nepřetržité volno naplánováno vždy na sobotu a neděli.

Ustanovení o odpočinku a nepřetržitém odpočinku se nevztahují na dohody o provedení práce a dohody o pracovní činnosti, s výjimkou zákazu přesáhnutí výkonu práce 12 hodin během 24 hodin (ČESKO, Zák. č. 262/2006 Sb.).

Vedení společnosti GISoft nepožaduje po docházkovém systému, aby jakkoliv ošetřoval dobu odpočinku a nepřetržitého odpočinku, protože díky vnitřním předpisům není možné, aby bylo zaměstnancům upřeno jejich právo na odpočinek.

2.2.5 Dny pracovního klidu a noční práce

Do dnů pracovního klidu jsou zahrnuty dny, na které připadá nepřetržitý odpočinek zaměstnance v týdnu a dny, na které připadají svátky podle zákona č. 245/2000 Sb., o státních svátcích, o významných dnech a o dnech pracovního klidu (ČESKO, Zák. č. 262/2006 Sb.).

Během dnů pracovního klidu smí zaměstnavatel nařídit zaměstnancům práci jen ve výjimečných případech definovaných zákonem. Mezi tyto výjimečné případy, které se týkají i společnosti GISoft patří následující (ČESKO, Zák. č. 262/2006 Sb.):

- Naléhavé opravné práce.
- Práce při živelních událostech a v jiných obdobných mimořádných případech (ČESKO, Zák. č. 262/2006 Sb.).

Zaměstnavatel může po zaměstnancích vyžadovat práci v noci. V takovém případě však nesmí délka směny v rámci 24 hodin překročit 8 hodin (ČESKO, Zák. č. 262/2006 Sb.). Na docházkový systém nejsou z hlediska dnů pracovního klidu a noční práce kladeny žádné požadavky, protože noční práce, ani mimořádné práce během dnů pracovního klidu nejsou v podniku povoleny.

2.2.6 Práce přesčas

Zaměstnavatel může svým zaměstnancům nařídit práci přesčas pouze ve výjimečných případech z vážných provozních důvodů. Práce přesčas může být nařízena i na dobu nepřetržitého odpočinku mezi dvěma směnami a za určitých podmínek rovněž na dny pracovního klidu (ČESKO, Zák. č. 262/2006 Sb.).

Nařízené přesčasy však nesmějí překročit 8 hodin v jednotlivých týdnech a celkově 150 hodin za kalendářní rok. Toto pravidlo může být porušeno jedině na základě dohody mezi zaměstnavatelem a zaměstnancem (ČESKO, Zák. č. 262/2006 Sb.).

Zaměstnanci ve společnosti GISoft mají zakázáno pracovat přesčas a vedení žádné přesčasy nařizovat nechce. Docházkový systém sice dle zadání má umět s přesčasy pracovat, ale nesmí zaměstnancům umožnit je zaznamenávat.

2.2.7 Pracovní cesty

Zaměstnanci společnosti GISoft musejí často podnikat pracovní cesty. Pracovní cesta je časově omezené vyslání zaměstnance za účelem výkonu práce na místo jiné než to, které bylo původně sjednáno jako místo výkonu práce (ČESKO, Zák. č. 262/2006 Sb.).

Zákon stanoví, že pro zaměstnance na pracovní cestě se neuplatňuje pružně rozvržená pracovní doba, která je jinak v GISoftu standardem, a zaměstnavatel v takovém případě musí zaměstnanci předem určit týdenní pracovní směny (ČESKO, Zák. č. 262/2006 Sb.).

Docházkový systém musí být dle zadání schopen zaznamenávat začátek a konec pracovní cesty jako speciální snadno rozlišitelnou akci. To má pomoci zaměstnancům odpovědným za vyplácení cestovní náhrad v jejich práci.

2.2.8 Dovolená

Každému zaměstnanci, který vykonává zaměstnání v pracovním poměru, vzniká právo čerpat dovolenou. Minimální délka dovolené je stanovena zákonem na 4 týdny v kalendářním roce, nicméně zaměstnavatel může svým zaměstnancům dovolenou prodloužit jako zaměstnanecký benefit. Zaměstnanci vykonávající práci na základě dohody o provedení práce nebo dohody o pracovní činnosti nemají za zákona nárok na dovolenou (ČESKO, Zák. č. 262/2006 Sb.).

Výjimku na délku dovolené zákon uděluje zaměstnancům, jejichž zaměstnavatelem je stát, neziskové organizace, apod., kterým přiznává dovolenou o délce 5 týdnů a pedagogům, kteří mají nárok na 8 týdnů dovolené (ČESKO, Zák. č. 262/2006 Sb.).

Čerpání dovolené se přerušuje v případě (ČESKO, Zák. č. 262/2006 Sb.):

- Nástupu zaměstnance na vojenské cvičení.
- Uznání zaměstnance dočasně práce neschopného.
- Ošetřuje-li nemocného člena rodiny.
- Nástupem zaměstnankyně na mateřskou a rodičovskou dovolenou, případně nástupem zaměstnance na rodičovskou dovolenou (ČESKO, Zák. č. 262/2006 Sb.).

Dle zadání má být systém schopný zaznamenávat začátek a konec čerpání dovolené zaměstnancem. Technicky by nebyl problém zajistit každému zaměstnanci jinou délku dovolené, např. její prodloužení jako zaměstnanecký benefit pro seniorní pozice, ale vedení podniku toto odmítlo. Všem zaměstnancům bude tedy systém měřit délku dovolené po zákonem stanovené minimum 4 týdnů.

Jedním z důvodů, proč je zaznamenávání čerpání dovolené zaměstnanci žádoucí, je snadný přístup k informaci o tom, kolik dnů dovolené ještě zaměstnanci zbývá. Při implementaci systému se proto bude muset vzít na tento fakt ohled a získání informace o zbývající dovolené maximálně usnadnit.

2.3 Platforma pro docházkový systém

Základní podmínkou, kterou si společnost GISoft kladla, bylo maximální využití stávajících technologií. To je rozumné rozhodnutí hned z několika důvodů:

- Ušetření nákladů, které by se musely vynaložit na pořízení nové technologie.
- Podmínkou by byla distribuce této technologie na všechny klientské stanice ať už formou instalace, případně zpřístupnění ze serveru.
- Zaměstnanci by museli být vyškoleni tak, aby s novými technologiemi uměli pracovat.
- Zavedením nové technologie by se zvýšila zátěž na administraci celého systému.

Protože je docházkový systém z principu databázová aplikace, je rovněž nutné zajistit, aby byla platforma schopna komunikovat s databází, do které bude ukládat data. Vzhledem k tomu, že naprosto všechna data budou pocházet z databáze, je nanejvýš vhodné, aby byla tato spolupráce co nejsnazší a nejefektivnější.

Při bližším zkoumání situace ve společnosti GISoft se objevilo hned několik možností, jak by se mohl docházkový systém realizovat, z nichž dvě varianty byly vyhodnoceny jako proveditelné. Protože společnost disponuje vlastním databázovým a webovým serverem, mohla by být vytvořena vlastní webová aplikace, která by plnila funkci docházkového systému. Pořízení takové aplikace by se ale prodražilo zejména kvůli větší časové náročnosti na implementaci, proto bylo rozhodnuto vydat se druhou cestou, použitím existující aplikace s následnými úpravami.

Jako platforma pro vznik docházkového systému byl tedy zvolen databázový program Access z balíku kancelářských aplikací Microsoft Office. Tento program vyhovuje podmínce dostupnosti, protože již v době, provádění analýzy současného stavu informačního systém, byl nainstalován na všech klientských počítačích.

2.3.1 Dostupné verze platformy

Na počítačích v kancelářích GISoftu je nainstalováno několik různých verzí softwaru Access, které nejsou stoprocentně dopředně kompatibilní, tzn., funkce novější verze nemusejí být podporovány verzí starší. Ve výběru konečné verze se mohou objevit Access 2007, Access 2010 nebo Access 2013.

Nejideálnějším kandidátem se pro jádro aplikace jeví Access 2013 napojený na server SharePoint 2013. Použitím této varianty by se obešla nutnost mít nainstalovaný Access 2013 na všech klientských stanicích, protože celá aplikace by sice byla vytvořena v Accessu, dostupná by ale byla pomocí SharePointu ve webovém prohlížeči.

Toto řešení však bylo zavrhnuto, protože ačkoliv má GISoft licenci i na nejnovější SharePoint server, nikdy jej nezprovoznil. Tento server by se musel nainstalovat pouze z důvodu jeho využití pro docházkový systém, což nepřipadá v úvahu. Proto budou v následující části podrobněji vysvětleny nové funkce a vlastnosti každé verze Accessu, aby bylo na konci možné vybrat tu nejvhodnější.

Access 2007

Od verze 2007 umožňuje Access vytvářet formuláře sloužící k interakci uživatele s daty v novém rozložení ovládacích prvků. Zatímco ve starších verzích bylo možné uspořádávat ovládací prvky do mřížky, Access 2007 nabízí možnost seskupit související prvky do jednoho celku tak, aby s nimi bylo možné manipulovat jako s jediným prvkem. Druhou možností, kterou novější verze Accessu poskytují, je vytvoření tabulkového rozložení, kdy se k formuláři připojí také záhlaví. Tento režim je vhodné použít při zobrazování a manipulování s větším počtem záznamů najednou (Microsoft Corporation, 2010).

Formuláře v Accessu 2007 umožňují používat tzv. vložená makra. Takové makro je uloženo přímo ve vlastnosti objektu, ke kterému náleží. Makro je vytvářeno za pomoci průvodce, čímž odstraňuje nutnost psát vlastní kód makra, protože ten je generován automaticky. Tato makra jsou systémem kontrolována tak, aby potenciálně nemohla způsobit žádnou škodu, a proto jsou označována jako důvěryhodná (Microsoft Corporation, 2010).

Další novinkou pro vývojáře je nový datový typ sloužící k ukládání binárních dat, např. obrázků, textových dokumentů apod. V rámci zjednodušení práce s datem a časem Access 2007 poskytuje vývojářům předpřipravenou komponentu kalendáře, která automaticky zajistí převzetí vstupu od uživatele a jeho validaci (Microsoft Corporation, 2010).

Access 2010

Nový Microsoft Access 2010 oproti předchozí verzi nepřinesl příliš mnoho změn. Užitečnou novinkou mohou být tzv. vypočítaná pole. Tato pole, která jsou běžnou součástí databázové tabulky, jsou dynamicky vypočítávána z ostatních polí. Použití této funkce je omezeno výhradně na pole ze stejné tabulky (Microsoft Corporation, 2010).

Access 2010 rovněž do značné míry vylepšil formátovací možnosti pro prezentaci dat. Pole obsahující číselnou informaci lze nastavit tak, aby pomocí lehkého podbarvení znázorňovala vloženou hodnotu graficky. To pomáhá uživatelům najít potřebnou informaci rychleji (Microsoft Corporation, 2010).

Access 2013

Zatím poslední vydaná verze Access 2013, razantně mění způsob práce, na který byli jeho uživatelé dosud zvyklí. Přináší s sebou koncept tzv. webových aplikací. Aplikace je nejprve vytvořena v Accessu, následně nahrána na server SharePoint a spouštěna ve webovém prohlížeči (Microsoft Corporation, 2013).

Jak již bylo zmíněno, tento nový vývojový model byl již zpočátku zavrhnut pro nedostatek volných zdrojů, převážně času, ale také lidí. Pro podporu vývoje původních databázových aplikací nepřidal Microsoft v této verzi Accessu žádnou novou funkci a lze předpokládat, že v tomto trendu bude i nadále pokračovat, neboť je v jeho vlastním zájmu prosadit nově vzniklý koncept webových aplikací, na nichž může stavět další zpoplatněné služby, např. hostování v cloudovém uložišti.

2.3.2 Výběr verze platformy

Při výběru použité verze platformy byl kladen důraz jednak na dostupnost konkrétní verze na pracovních stanicích v rámci celé firmy, hlavně pak ale na technickou způsobilost dané verze softwaru provozovat aplikaci typu docházkového systému.

Z klíčových prvků, které musí platforma umět, byly vybrány následující:

- Schopnost ukládat a získávat data z relační databáze.
- Možnost uzamknout všechny části aplikace proti neoprávněnému čtení a editaci.
- Plná kontrola nad spravovanými daty a uživatelským rozhraním pomocí skriptů.
- Podpora pro vytváření uživatelského rozhraní ve formě formulářů s možností seskupovat a opakovaně využívat existující komponenty.
- Široká nabídka předpřipravených komponent použitelných vzhledem k předmětu funkce docházkového systému, např. komponenty pro práci s datem a časem.

Jako ideální kandidát na platformu pro docházkový systém byl nakonec zvolen Microsoft Access 2007. Tato verze totiž splňuje všechny základní požadavky, které jsou na ní kladeny. Díky zpětné kompatibilitě novějších verzí je navíc zajištěno, že i zaměstnanci s modernějším softwarem budou schopni s docházkovým systémem bez problémů pracovat. Použitím Accessu 2007 se tedy odstranila nutnost instalace jakéhokoliv nového softwaru jak na pracovních stanicích, tak na firemních serverech.

2.4 Analýza vlastností platformy

Aplikace Access umožňuje vývojářům vytvořit jednak návrh databáze, včetně tabulek a vazeb mezi nimi, jednak databázových sestav, které uživatelům přehledně zobrazí požadovaná data, a konečně formuláře sloužící k interakci s daty.

Primárně jsou jako styčné body mezi databází a uživatelem určeny právě sestavy a formuláře se svými standardními funkcemi, založenými na dotazech v jazyce SQL. Jejich funkce lze ale rozšířit pomocí makro jazyka VBA.

Access standardně obsahuje editor kódu napsaného v jazyce VBA. Ten je však již poměrně zastaralý a nevyhovuje dnešním standardům pro vývojová prostředí. To bude mít za následek menší efektivitu práce a tím i větší časové nároky na implementaci. Protože se předpokládá, že bude jazyk VBA využíván ve velké míře po celém řešení, budou jeho vlastnosti v této kapitole přiblíženy.

2.4.1 Princip dědičnost objektů ve VBA

V jazyce VBA jsou programové části, běžně známé jako třídy, nazývány modulovými třídami. Na rozdíl od většiny ostatních jazyků, VBA nepodporuje klasickou dědičnost, kdy základní třída definuje obecné chování a její potomek definuje funkce úžeji zaměřené na konkrétní způsob využití, případně modifikují chování svého rodiče (Lomax, 1998).

Jediným způsobem jakým lze spojit dvě třídy do vztahu nadřízenosti a podřízenosti je implementovat v potomkovi rodičovskou třídu jako rozhraní. Implementace rozhraní zajistí, že každý potomek musí bezpodmínečně obsahovat implementaci všech veřejných funkcí, procedur a vlastností svého rodiče, jinak se nepodaří programový kód ani zkompilovat. Touto metodou lze zajistit, že se všechny modulové třídy, které pojí dohromady nějaká společná vlastnost, budou chovat stejným způsobem (Lomax, 1998).

Výraznou nevýhodou je, že s tímto způsobem implementace dědičnosti nelze dosáhnout stavu, kdy potomek využívá obecných funkcí svého rodiče. Každá třída tedy musí vytvořit vlastní implementaci, třebaže je úplně stejná, jako u všech ostatních tříd se stejným rodičem.

Důsledkem tohoto implicitního chování jazyka VBA je psaní redundantního kódu na mnoha různých místech a s tím ruku v ruce jdoucí horší udržovatelnost výsledného kódu,

vyžadujícího při každé změně úpravu ve všech výskytech. Neexistence dědičnosti tříd částečně znemožňuje aplikaci návrhového vzoru SOLID, neboť přímo porušuje pravidlo Open-Closed Principle. Při analýze bylo toto chování jazyka identifikováno jako výrazný nedostatek, na který se bude muset brát při implementaci docházkového systému ohled.

2.4.2 Práce s databází

Microsoft Access umožňuje jazyku VBA plný přístup k datům v databázi. Data získaná z databáze jsou vždy uložena v objektu typu Recordset. Datový typ Recordset představuje kurzor ukazující na aktuální záznam v databázi, přičemž svými metodami umožňuje aktuální záznam přepínat na jiný. S daty aktuálního záznamu může následně programátor dále pracovat (Harkins et. al., 2004).

Při získávání i úpravě dat v databázi se přímo v programu používají příkazy jazyka SQL uložené jako textové řetězce (Harkins et. al., 2004). Protože docházkový systém z principu vyžaduje neustálou součinnost s databází, hrozí, že se stane takový způsob zadávání příkazů nepřehledným a špatně udržovatelným. S předchozím problémem úzce souvisí hrozba výskytu chyb a překlepů v samotných SQL příkazech, které by mohl vývojář do těchto příkazů nevědomky zanést.

Ideálním řešením, které by mělo být v rámci implementace docházkového systému vytvořeno, by byl jednotný systém získávání a úpravy dat, který by programátora úplně oprostil od nutnosti psát SQL příkazy v kódu jazyka VBA.

Dalším podstatným problémem při práci s datovým typem Recordset je nemožnost provádět statickou typovou kontrolu již v době kompilace. Absence tohoto typu kontroly by při práci s databází mohla způsobovat dva typy chyb:

- Při pokusu o uložení dat jiného datového typu, než jakého je ovlivňovaný sloupec v databázi.
- Při pokusu o přiřazení hodnoty z databáze do proměnné nevyhovujícího datového typu jazyka VBA.

Oba dva druhy chyb by mohly mít za následek zhroucení programu, a proto je důležité se na jejich eliminaci zaměřit již ze začátku vývoje. Při implementaci docházkového

systému bude jednou z hlavních priorit vyřešit způsob práce s daty v databázi tak, aby umožňovala statickou typovou kontrolu v době kompilace.

2.4.3 Nekompatibilita datových typů

Jelikož je spolupráce mezi jazykem VBA a databází nezbytností, je důležité zajistit vzájemnou bezproblémovou výměnu dat. Při práci se složitějšími datovými typy však vývojář brzy narazí na jejich vzájemnou nekompatibilitu v rámci obou systémů.

Jedním z hlavních představitelů těchto vzájemně nekompatibilních implementací je datový typ umožňující uchování informace o datu a času. Právě tento datový typ bude v rámci docházkového systému využíván ve velké míře, proto se jedná o závažný problém.

Jednotlivá pole v databázi navíc umožňují data nezadávat vůbec. V takovém případě jsou nastavena na hodnotu NULL. Jazyk VBA ovšem podobný mechanizmus nenabízí pro všechny datové typy. Ty se ve VBA dělí do dvou kategorií (Lomax, 1998):

- Hodnotové datové typy.
- Referenční datové typy (Lomax, 1998).

Obdobu hodnoty NULL poskytuje VBA pouze pro referenční datové typy. Hodnotové datové typy musejí mít vždy nějakou hodnotu nastavenu (Lomax, 1998). Předpokládá se přitom, že právě hodnotové datové typy bude nutné ukládat do databáze nejčastěji.

V rámci analýzy bylo tedy zjištěno, že do systému zajišťujícího spolupráci programu s databází budou muset být implementovány konvertory mezi datovými typy, a že tento systém bude muset být schopen zpracovat a uchovat informaci o tom, že data nejsou v databázi vůbec zadána.

2.5 Analýza SWOT

V rámci SWOT analýzy se bude hodnotit samotný projekt implementace docházkového systému.

2.5.1 Shrnutí silných a slabých stránek

Výhodou zvoleného řešení jsou bezesporu nulové náklady na dodatečný hardware nebo programové vybavení. Nejde navíc pouze o fakt, že není potřeba platit za samotný statek,

ale rovněž i za lidské zdroje, které by bylo nutné potenciálně vynaložit při údržbě a správě nového vybavení.

V případě, že by se v podniku ukázalo, že stávající docházkový systém již nevyhovuje požadavkům na něj kladených, je možné celý systém relativně snadno převést na robustnější řešení ať již v podobě webové aplikace napsané v Accessu 2013 nebo samostatné serverové aplikace pracující na principu klient-server.

Výrazným handicapem použité technologie je však její stáří. Jazyk VBA se již několik let prakticky nevyvíjí a neodráží proto moderní trendy v oblasti softwarového vývoje. Největším neduhem VBA je úplné ignorování principu dědičnosti tříd a několik dalších svou důležitostí již menších problémů, které jsou však již, na rozdíl od zmíněné dědičnosti, snadno řešitelné.

Samotný Access 2007 bude na hranicích svých možností z pohledu schopnosti vyřizovat požadavky více uživatelů v jeden okamžik a v případě, že se požadavky na něj uvnitř společnosti zvýší, přestane být dostačujícím řešením. Zastaralost Accessu se také promítá do délky vývoje aplikace, především nedostačujícími funkcemi, které poskytuje editor kódu.

2.5.2 Shrnutí příležitostí a hrozeb

Důvodem, proč chce společnost GISoft implementovat docházkový systém je zlepšení znalostí o docházce svých zaměstnanců. Přestože se svými zaměstnanci nemá GISoft výraznější problémy, vedení by rádo tento systém zavedlo, aby zvýšilo jejich odpovědnost za svou přítomnost na pracovišti.

Zavedením softwarového vedení docházky lze navíc automatizovat některé úkony spojené s účetnictvím a vyplácením mezd. Tímto krokem by se tedy celkově zefektivnil každodenní chod v podniku.

Zvolené platformě, na které bude docházkový systém vystaven, však hrozí, že přestane být časem podporována. Již před vydáním nejnovější verze balíku kancelářských aplikaci Microsoft Office se spekulovalo, že bude jazyk VBA nahrazen výrazně modernějším jazykem Javascript (Foley, 2011). Nakonec se ukázalo, že Microsoft opravdu chce změnit

způsob, jakým se v Accessu vyvíjejí aplikace. Bohužel tento způsob se ukázal být nevhodným pro použití v GISoftu.

Bezproblémové a hladké fungování systému mohou také ohrozit legislativní změny v zákonech zabývajících se úpravou vztahu mezi zaměstnanci a zaměstnavatelem, pracovní dobou, přestávkami mezi prací, apod.

2.5.3 Výsledky SWOT analýzy

Po uvážení všech vstupů do analýzy byla vytvořena následující tabulka, přehledně znázorňující jak silné a slabé stránky, tak i příležitost a hrozby spojené s nasazením docházkového systému:

Tabulka č. 2: Grafické znázornění výsledků SWOT analýzy

Silné stránky	**Slabé stránky**
• Podnik již vlastní vyhovující infrastrukturu. • Není potřeba pořizovat nový software. • Žádné dodatečné náklady spojené s běžným fungováním systému. • Možnost migrace na jiné systémy.	• Zastaralá technologie jako součást použité platformy. • Jazyk VBA nebyl po mnoho let modernizován. • Nepříliš pohodlné vývojářské nástroje.
Příležitosti	**Hrozby**
• Snížení nákladů spojených s počítáním mezd. • Zvýšení angažovanosti zaměstnanců ve svém zaměstnání.	• Nejasná budoucí podpora platformy ze strany jejích tvůrců. • Potřeba měnit chování systému při změně legislativy.

(Zdroj: vlastní zpracování)

2.5.4 Celkové shrnutí analýzy

Společnost GISoft poměrně striktně omezila možnosti při implementaci docházkového systému na technologie, kterými již disponuje. Nejsnazším řešením se ukázalo být využití stávajícího databázového software Microsoft Access, který spoustu požadovaných funkcí nabízí již ze své podstaty.

I Access však s sebou přináší výzvy, které je nutno efektivně vyřešit, zejména v oblasti aplikačního programového rozhraní. Přestože bylo objeveno několik nedostatků, kterými trpí Access a na něj navázaný jazyk VBA, byla navržena opatření tak, aby bylo možné všechny problémy vyřešit na dostatečně obecné úrovni tak, aby byla řešení znovupoužitelná, čímž by se navíc značně urychlila implementace celého docházkového systému.

V neposlední řadě se jako velká výhoda vývoje v Accessu jeví minimální náklady na pořízení a následnou údržbu. Navíc, pokud by k tomu snad někdy mělo dojít, bude systém již od počátku navrhován tak, aby byl případný přechod na jinou technologii maximálně zjednodušený. Proto byl projekt implementace docházkového systému vyhodnocen jako životaschopný a dojde k jeho realizaci.

Následující část práce bude z výše jmenovaných důvodů z velké části věnována návrhům řešení jednotlivých omezení použité platformy, které nakonec umožní docházkový systém vytvořit ve znatelně kratším čase. Po odstranění všech překážek bude nakonec vytvořena implementace aplikace na základě definovaných požadavků.

3 VLASTNÍ NÁVRH ŘEŠENÍ, PŘÍNOS PRÁCE

V rámci návrhu řešení nebyla vytvořena pouze implementace docházkového systému, ale byly provedeny i rozsáhlé úpravy na úrovni platformy tak, aby bylo možné následný vývoj maximálně urychlit a hlavně, aby se umožnila migrace na jinou technologii v případě, kdy by to bylo nutné.

3.1 Návrh databázového schématu

Podstatnou součásti docházkového systému je databáze, která ukládá veškerá data. Využit je systém pro správu báze dat zabudovaný přímo v Accessu. Databáze je vytvořena z několika vzájemně propojených tabulek, tzv. relací. Pro každou tabulku, resp. záznam v tabulce platí, že musí být označen jedinečným číselným identifikátorem. Toto pravidlo neplatí obecně, ale bylo zavedeno pro účely docházkového systému.

3.1.1 Uživatelé a role

Velmi podstatnou součástí docházkového systému jsou jeho uživatelé, neboli zaměstnanci, kteří zaznamenávají svou docházku a jejich vedoucí, kteří ji kontrolují a vypočítávají na jejím základě mzdy.

Pro ukládání dat uživatele je rezervována tabulka Users. Ta umožňuje uchovávat kromě celého jména daného člověka i unikátní přihlašovací jméno, na jehož základě je uživatel v systému autentizován. Tabulka Users vede záznamy rovněž o tom, zda byl uživatel již aktivován a může se do systému přihlásit, nebo zda byl zrušen.

Tabulka č. 3: Atributy tabulky Users

Jméno atributu	Popis atributu	Datový typ a délka	NULL
UserID	Identifikátor uživatele	Automatické číslo	Ne
UserLogin	Přihlašovací jméno	Text (50 znaků)	Ne
UserName	Jméno	Text (30 znaků)	Ano
UserLastName	Příjmení	Text (30 znaků)	Ano
UserIsDisabled	Uživatel je zablokován	Boolean	Ne
UserIsActivated	Uživatel byl aktivován	Boolean	Ne
SystemRolesRoleID	Zařazení do role	Celé číslo	Ano

(Zdroj: vlastní zpracování)

Společnost GISoft požaduje, aby bylo možno rozdělit uživatele do rolí, na jejichž základě pak bude moci jednotlivým zaměstnancům přidělovat různý stupeň oprávnění. Údaje o těchto rolích jsou uloženy v tabulce SystemRoles. Tabulka SystemRoles je velmi jednoduchá a kromě povinného číselného identifikátoru obsahuje pouze slovní název každé systémové role. I přes to, že není tato tabulka příliš rozsáhlá, je extrémně důležitá, protože na jejím základě funguje přidělování práv k rozličným operacím a vůbec možnost zobrazovat konkrétní součásti uživatelského rozhraní.

Tabulka č. 4: Atributy tabulky SystemRoles

Jméno atributu	Popis atributu	Datový typ a délka	NULL
RoleID	Identifikátor role	Automatické číslo	Ne
RoleName	Jméno role	Text (50 znaků)	Ne

(Zdroj: vlastní zpracování)

3.1.2 Údaje o docházce zaměstnanců

Záznamy, které mají něco společného se zaznamenáváním docházky zaměstnanců, se ukládají do tabulek, v jejichž názvu se vyskytuje výraz „CheckIn“. Předně je potřeba rozlišovat mezi několika různými typy zapsání docházky, protože zaměstnanec může do systému poznačit nejen příchod do práce, ale i odjezd na služební cestu apod.

Pro definici všech typů těchto záznamů slouží tabulky CheckInTypes a CheckInInteruptionTypes. První tabulka obsahuje informace o všech globálních typech aktivit, které může zaměstnanec ve svém zaměstnání vykonávat. V této tabulce tedy jsou definovány pracovní směny, mateřské a rodičovské dovolené, pracovní cesty apod.

Tabulka č. 5: Atributy tabulky CheckInTypes

Jméno atributu	Popis atributu	Datový typ a délka	NULL
TypeID	Identifikátor typu záznamu	Automatické číslo	Ne
TypeName	Název typu záznamu	Text (50 znaků)	Ne
TypeCodeName	Kódové označení typu	Text (30 znaků)	Ne

(Zdroj: vlastní zpracování)

Ve druhé zmíněné tabulce jsou vedeny definice aktivit, které mohou globální aktivity přerušit. Nejčastějším případem takové aktivity je přestávka v práci, ale je možné rozlišovat mezi více typy, např. odchod k lékaři.

Tabulka č. 6: Atributy tabulky CheckInInteruptionTypes

Jméno atributu	Popis atributu	Datový typ a délka	NULL
TypeID	Identifikátor typu přerušení	Automatické číslo	Ne
TypeName	Název typu přerušení	Text (50 znaků)	Ne
TypeCodeName	Kódové typu přerušení	Text (30 znaků)	Ne

(Zdroj: vlastní zpracování)

Tabulky CheckInTypes a CheckInInteruptionTypes slouží k ukládání stejných informací, ovšem na jiné úrovni, a proto jsou rozděleny do vlastních tabulek. Jejich struktura je ale jinak naprosto stejná. Obě dvě tabulky obsahují jednak název aktivity a následně kódové označení této aktivity. Kódové označení se využívá při práci v jazyce VBA.

Samotné informace o tom, kdy se zaměstnanec zapsal do docházkového systému a kolik času strávil v práci, jsou zapsány v tabulce CheckIns. Tento údaj je uchován ve dvou sloupcích obsahujících čas příchodu a čas odchodu. Protože se systémem pracují lidé, kteří mohou dělat chyby, uchovává tabulka informaci o tom, zda aktuální záznam již byl úspěšně uzavřen, nebo zdali ještě čeká na zásah správce, aby opravil vzniklé chyby. Kromě těchto sloupců tabulka navíc obsahuje cizí klíče na uživatele a typ aktivity, který záznam představuje.

Tabulka č. 7: Atributy tabulky CheckIns

Jméno atributu	Popis atributu	Datový typ a délka	NULL
CheckInID	Identifikátor záznamu	Automatické číslo	Ne
CheckInTime	Čas vytvoření záznamu	Datum a čas	Ne
CheckOutTime	Čas uzavření záznamu	Datum a čas	Ano
UsersUserID	Přiřazení k uživateli	Celé číslo	Ne
CheckInTypesTypeID	Typ záznamu	Celé číslo	Ne
CheckInIsCompleted	Úspěšně uzavřený záznam	Boolean	Ne

(Zdroj: vlastní zpracování)

Na záznamy v tabulce CheckIns jsou přímo navázány údaje o přerušení aktivit. Tato přerušení jsou uchovávána v tabulce CheckInInteruptions. Záznamy se poněkud liší od záznamů v CheckIns. Shodně obsahují informaci o přesném datu a času vzniku záznamu, ale místo doby ukončení obsahují číselný údaj o době trvání. Tento způsob záznamu byl

zvolen vzhledem k potřebě přesně vypočítávat odpracovanou dobu. Návaznost na konkrétní aktivitu je zajištěna cizím klíčem, stejně tak určení typu přerušení aktivity.

Tabulka č. 8: Atributy tabulky CheckInInteruptions

Jméno atributu	Popis atributu	Datový typ a délka	NULL
InteruptionID	Identifikátor přerušení	Automatické číslo	Ne
InteruptionCreationTime	Čas přerušení	Datum a čas	Ne
InteruptionDuration	Délka přerušení	Celé číslo	Ano
CheckInsCheckInID	Přerušený záznam	Celé číslo	Ne
CheckInInteruptionTypesTypeID	Typ přerušení	Celé číslo	Ne

(Zdroj: vlastní zpracování)

3.1.3 Textové řetězce a nastavení

Kromě tabulek zajišťujících ukládání informací přímo souvisejících s docházkou jsou v databázi navíc vytvořeny i servisní tabulky, které slouží výhradně k běhu docházkového systému jako takového.

První servisní tabulkou je Settings. Ta je schopna pod unikátním klíčem uchovávat libovolné informace. Data z této tabulky lze účinně využívat v algoritmech počítajících odpracovanou dobu, protože lze díky nim omezit potřebu zásahu do programového kódu. Typickým příkladem pro takový záznam může být doba, po které musí zaměstnanec nastoupit na přestávku v práci. Pokud platná legislativa tento údaj v budoucnu změní, nebude potřeba upravovat algoritmus pro výpočet odpracované doby, ale pouze jeden záznam v databázi.

Tabulka č. 9: Atributy tabulky Settings

Jméno atributu	Popis atributu	Datový typ a délka	NULL
SettingKey	Název klíče	Text (100 znaků)	Ne
SettingValue	Hodnota klíče	Text (proměnlivá délka)	Ne

(Zdroj: vlastní zpracování)

Další dvě servisní tabulky, které se v databázi nacházejí, uchovávají informace o statických textových řetězcích použitých v uživatelském prostředí. Tyto textové řetězce je výhodné používat, protože je možné je kdykoliv změnit bez nutnosti editovat

programový kód. Namísto toho lze jejich znění upravit v databázi a změny se automaticky projeví na všech místech, kde je tento řetězec použit.

Tabulka ResStringCategories samotná neobsahuje žádné takové řetězce, ale místo toho pomáhá kategorizovat řetězce do logických celků. Jedním z příkladů takového logického celku je jeden formulář. U každého formuláře se předpokládá, že bude mít svůj specifický set textových řetězců, a proto se hodí je kategorizovat. Tato tabulka obsahuje pouze identifikátor a název kategorie.

Tabulka č. 10: Atributy tabulky ResStringCategories

Jméno atributu	Popis atributu	Datový typ a délka	NULL
CategoryID	Identifikátor kategorie	Automatické číslo	Ne
CategoryName	Název kategorie	Text (50 znaků)	Ne

(Zdroj: vlastní zpracování)

Samotné textové řetězce jsou uloženy v tabulce ResStrings pod kódovým označením, podle kterého jsou vyhledávány v kódu VBA a vlastním textovým řetězcem, který má být při překladu použit. Tabulka navíc obsahuje cizí klíč do ResStringCategories, aby byla zachována informace o kategorii, do které konkrétní textový řetězec spadá.

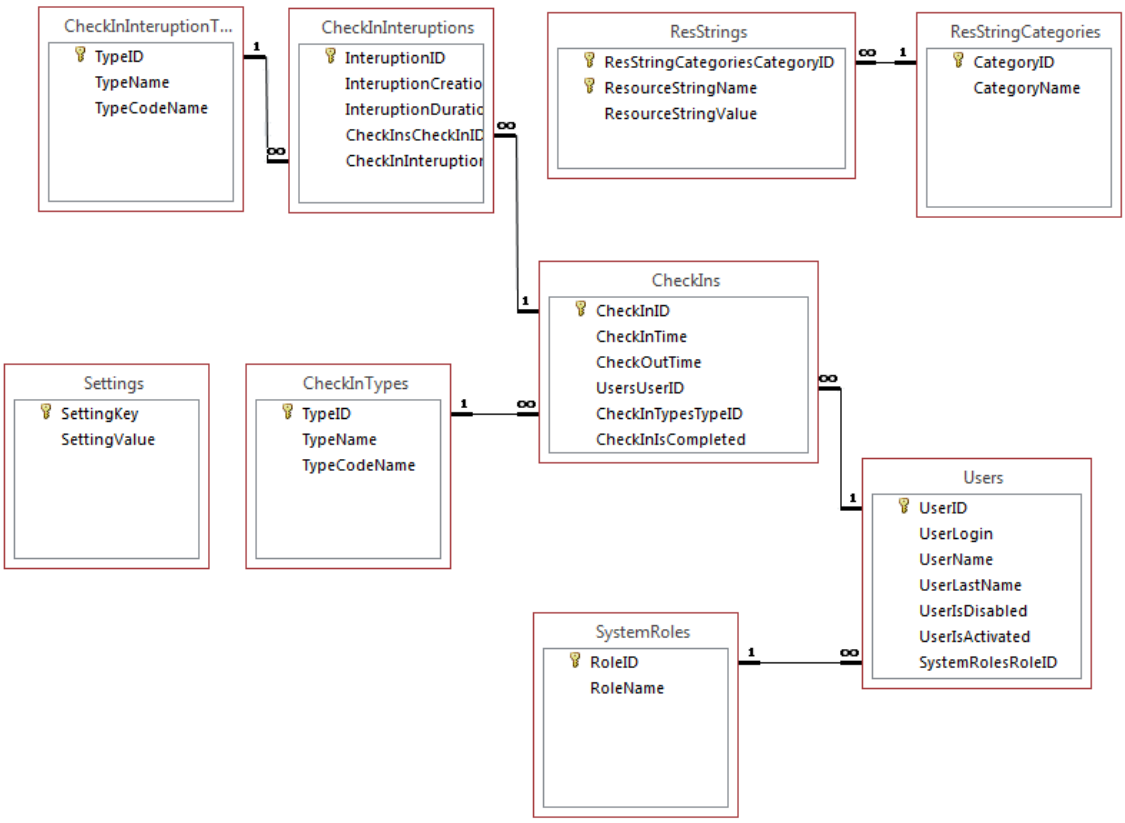
Tabulka č. 11: Atributy tabulky ResStrings

Jméno atributu	Popis atributu	Datový typ a délka	NULL
ResStringCategoriesCategoryID	Přiřazení do kategorie	Celé číslo	Ne
ResourceStringName	Název řetězce	Text (50 znaků)	Ne
ResourceStringValue	Hodnota řetězce	Text (255 znaků)	Ne

(Zdroj: vlastní zpracování)

3.1.4 Vztahy mezi relacemi

Zmíněné tabulky dohromady tvoří strukturu, která zajišťuje integritu celého docházkového systému. Vztahy mezi relacemi zachycuje následující diagram.



Obrázek č. 1: ER diagram docházkového systému
(Zdroj: vlastní zpracování)

3.2 Framework pro automatické testy

Protože jedním z hlavních požadavků na docházkový systém je jeho bezproblémová údržba spojená s minimalizací nákladů, bylo rozhodnuto, že jako jeden z hlavních pilířů, zajišťujících neustálou funkčnost systému, budou automatické testy. Pomocí automatických testů vývojář okamžitě zjistí, že při úpravě stávajícího kódu udělal chybu a může ji napravit ještě před tím, než stihne napáchat další škody.

Protože program Access, ani jazyk VBA sám o sobě neobsahuje sám o sobě vestavěnou podporu pro psaní automatických testů, byla tato podpora vytvořena vlastními silami v rámci implementace tohoto projektu.

3.2.1 Jednotkový test

Na začátku bylo definováno, jak má takový test vypadat. Kromě obecných pravidel, která např. zakazují umísťovat do testu funkční logiku, musí testy splňovat dvě podmínky:

- Musí jít o veřejnou funkci vracející hodnotu typu Integer.
- Hodnotu, kterou funkce vrací, musí pro správné vyhodnocení testu určit samotný testovací framework.

Obecně test vypadá tak, že nejprve provede inicializaci všech potřebných proměnných, zavolá testovanou metodu a na konci testu nechá testovací framework rozhodnout, zda byla očekávaná vrácená hodnota shodná s hodnotou skutečně vrácenou.

3.2.2 Správce testů

Samotný správce testů byl vytvořen v rámci modulu TestFramework. Ten obsahuje proceduru, která zajistí spuštění všech automatických testů v modulu, jehož název je proceduře předán jako parametr. Po tomto modulu je vyžadováno, aby obsahoval výhradně testy a žádný jiný kód.

Spuštění testů je rozděleno do několika oddělených fází. To je nutné hlavně proto, že se během spouštění testů dynamicky za běhu mění programový kód a Access nezvládá tolik změn najednou kompilovat. Dokud nebylo testování rozděleno, docházelo k okamžitému pádu aplikace bez zjevné příčiny nebo chybového hlášení.

3.2.3 Fáze zpřístupnění testovaných metod

Velice častým jevem je snaha testovat funkce, jejichž přístupnost je omezena výhradně pro interní účely, tzn., jsou to soukromí členové modulu. Jak již bylo zmíněno, testovací framework vyžaduje, aby byly testy umístěny do svého vlastního speciálního modulu. To ale způsobuje, že testy nemohou přistupovat k privátním členům jiného modulu a nemohou je tedy testovat.

Aby se testům umožnilo testovat i soukromé členy modulů a tříd, byla zavedena podpora pro dočasné zviditelňování testovaných funkcí. Ve správci testů jsou zaregistrovány ty funkce, které je potřeba zpřístupnit před spuštěním samotných testů a tyto funkce jsou pak v rámci první fáze dynamicky přímo v kódu zpřístupněny, čímž se stanou testovatelnými.

Protože jsou ale testy napsány ještě ve chvíli, kdy funkce nejsou viditelné, došlo by při pokusu o kompilaci řešení k chybě kvůli snaze přistoupit na neveřejné členy jiného modulu. Z toho důvodu musí být testy po celou dobu umístěny v komentářích, tak aby nezpůsobovali havárii při kompilaci, a teprve po zpřístupnění testovaných metod zajistí správce testů odstranění komentářů. V tu chvíli už je řešení úplně kompilovatelné a správce testů může přejít k další fázi.

3.2.4 Fáze přípravy spuštění testů

Až v této fázi začíná správce testů ve skutečnosti procházet modul s testy a registrovat jednotlivé testy tak, aby byly spustitelné. Po nalezení všech testů musí správce zajistit, aby byly testy kompatibilní s testovacím frameworkem. Ten totiž kvůli omezením platformy musí spouštět testy ve speciálním režimu a připravuje se tak o některé jinak standardní vlastnosti jazyka VBA.

Zajištění kompatibility probíhá tak, že se na začátek každého testu za běhu programu umístí speciální obslužný kód zajišťující odchytávání chyb, které se v testu mohou objevit, a na úplný konec testu volání pomocné procedury zajišťující předání popisu chyby zpět do testovacího frameworku.

Tato fáze zajišťuje, že přestože se v testovaném kódu objeví neodchycená výjimka, test jako takový nezpůsobí havárii programu a přerušení provádění ostatních testů. Místo toho se prostě zaznamená, k jaké chybě v testu došlo a pustí se následující test.

3.2.5 Fáze průběhu testů

Když je kompatibilita testů s frameworkem zajištěna, může být spuštěna předposlední a nejdůležitější fáze, spuštění samotných testů. Protože testy jsou již zaregistrovány, může se jeden po druhém projít a spustit.

Spuštění testu ale neprobíhá standardní cestou. Protože se počet testů může měnit dynamicky např. vytvořením nového testu nebo testováním jiné sady testů umístěné v jiném modulu, není možné testy spouštět prostým zavoláním. Namísto toho se název testu použije jako výraz do standardní vestavěné funkce pro vyhodnocování výrazů Eval, která se o jeho spuštění postará.

Spouštění testů pomocí funkce Eval s sebou nese jeden problém: kontext, ve kterém je daný test spuštěn, je totiž jiný, než ten, ve kterém běží zbytek programu. To například znamená, že standardní odchytávání výjimek nebude pro testovaný výraz fungovat a chyba v testu by způsobila pád celého programu. Právě z tohoto důvodu se musí v předchozí fázi všechny testy projít a upravit tak, aby bylo odchytávání výjimek zajištěno uvnitř každého testu zvlášť.

Na základě toho, jakou hodnotu daný test vrátil, se test vyhodnotí jako úspěšně vykonaný nebo selhaný. Za selhaný test se považuje i takový, který během svého provádění způsobil výjimku. Výsledky testů, tedy počty úspěšných a neúspěšných testů, včetně jejich názvů a popisu chyby, jsou následně vypsány do vývojářské konzole v prostředí jazyka VBA.

3.2.6 Fáze obnovení

Poté, co je testování dokončeno, musí správce testů opětovně projít všechny testy a odstranit z nich automaticky vygenerovaný kód, který zajišťuje kompatibilitu s testovacím frameworkem. Bylo rozhodnuto, že se tento kód bude z testů odstraňovat, aby testy zůstaly přehlednější a hlavně proto, aby se dodatečnými úpravami nemuseli zabývat programátoři, kteří testy píší.

Po úpravě testů do původního stavu se navíc celý kód modulu s testy opětovně umístí do komentářů, aby testy nezpůsobovaly problémy při pozdější kompilaci, spojené zejména s přístupem k soukromým členům jiných modulů.

Úplně posledním krokem, který správce testů provede, je opětovné znepřístupnění zaregistrovaných testovaných metod tak, aby jejich použití nebylo možné mimo vlastní modul. Tento krok musí následovat až poté, co jsou všechny testy umístěny do komentářů, jinak hrozí, že selže kompilace. Ukončením této fáze je kód ve stejném stavu, v jakém byl na začátku, ale zároveň mohlo dojít k otestování požadované funkcionality.

3.3 Abstrakce od databáze

Již od začátku projektu bylo jasné, že platforma pro docházkový systém v podobě Microsoft Access je sice pro aktuální účely naprosto dostačující, nicméně pro případné dodatečné požadavky v budoucnu již nevyhovující. Z tohoto důvodu se celým procesem implementace táhne snaha v maximální možné míře abstrahovat veškeré funkce

specifické pro Access tak, aby bylo možné použitou technologii co možná nejsnáze nahradit.

První a nejvýznamnější prvek, kterým se Access vyznačuje a který je zároveň stavebním kamenem docházkového systému, je vestavěná databáze. Již při analýze současného stavu bylo zjištěno, že při případných změnách v architektuře by musela být datová vrstva vyměněna jako první.

Aby bylo možné datovou vrstvu v podstatě kdykoliv nahradit, vznikl modul DataEngine, jež působí jako rozhraní mezi zbytkem programu a perzistentním úložištěm dat. Tento modul poskytuje základní metody pro získávání, vkládání a úpravu dat. Pro zajištění oddělitelnosti programu od datového úložiště musí veškerá komunikace související s uloženými daty probíhat právě přes tento modul.

DataEngine předpokládá, že jsou data vždy uložena v logických strukturách identifikovatelných svým unikátním názvem. V kontextu relačních databází je tímto unikátním názvem myšlen název tabulky. Tento unikátní název a samotná data jsou to jediné, co modul potřebuje k tomu, aby byl plně funkční. Metody, pro které to dává smysl, navíc volitelně poskytují možnost ovlivnit, s jakými daty budou pracovat na základě podmínky.

3.3.1 Univerzální datové úložiště

DataEngine představuje prostředníka pro komunikaci s jakýmkoliv vyhovujícím typem datového úložiště, nicméně sám o sobě není schopen s žádnými daty v úložišti manipulovat. K tomu využívá speciální třídu univerzálního datového úložiště implementující rozhraní IDataConnector. Tato třída může pracovat s daty v prakticky libovolném formátu, od tabulek v relační databázi, po soubory typu XML.

Výhodou univerzálního datového úložiště je, že se musí programátor úplně oprostit od použití dotazů v jazyce SQL, protože ty by pro některá úložiště nedávaly žádný smysl. Namísto těchto dotazů musí využívat třech obecných metod pro získávání, vkládání a aktualizaci dat – GetData, InsertData a UpdateData.

3.3.2 Konvertory datových typů

Aby se smazaly rozdíly mezi datovými typy a prostředím Accessu, vzniklo rozhraní IDataConverter. To musí povinně implementovat každá implementace IDataConnector a zároveň jej musí využívat při práci s daty, ať už směrem z programu do perzistentního úložiště nebo naopak.

K manipulaci s daty poskytuje IDataConverter dvě metody: Convert a ConvertBack. Metoda Convert slouží ke konverzi datového typu jazyka VBA na odpovídající datový typ kompatibilní s perzistentním úložištěm. Naopak ConvertBack zajišťuje převedení datového typu použitého v perzistentním úložišti zpět do jazyka VBA.

Pomocí konvertorů datových typů bylo možno úplně odstínit perzistentní úložiště dat od výkonného kódu a zjednodušit celý vývoj odstraněním starostí o způsob uložení konkrétní informace.

3.3.3 Napojení na Microsoft Access

Konkrétní implementací rozhraní IDataConnector vytvořenou přímo pro projekt docházkového systému je třída JetEngineConnector zajišťující komunikaci se systémem pro řízení báze dat zabudovaném v Microsoft Access, který se nazývá Jet engine.

Jet engine, resp. celý Access staví systém ukládání dat na relační databázi, se kterou lze komunikovat za pomoci SQL jazyka. Proto se již v této konkrétní implementaci musí jazyk SQL využívat. Pro zbytek aplikace ale stále platí, že o příkazech v SQL nemusí nic vědět a k datům se přesto dostane.

V analýze současného stavu vyšlo najevo, že je nutné přímo v kódu jazyka VBA dynamicky vytvářet dotazy v jazyce SQL a že je tento způsob velmi náchylný na neúmyslné chyby v podobě překlepů, případně syntaktických nedostatků. Aby se tomuto nebezpečí předešlo, vznikla speciálně pro tento případ třída Query, která pomáhá se sestavením SQL dotazu tak, aby byl syntakticky v pořádku. I když je třída původně určena pro použití v JetEngineConnector, lze ji, díky podobnosti jazyka SQL napříč systémy, využít i v případě migrace na jiné databázové systémy.

Základní myšlenkou třídy Query je snaha o objektově sestavený dotaz, který je sice velice podobný syntaxi jazyka SQL a dobře čitelný pro člověka, zároveň ale bezchybný a

připravený pro zpracování databázovým strojem. Třída Query definuje několik funkcí svým názvem velice podobným svým ekvivalentům v SQL. Jsou to funkce:

- Selection a SelectionDistinct – slouží pro určení konkrétních vybraných sloupců.
- From, InsertInto, Update – definují tabulku, ve které se nacházejí nebo budou nacházet právě zpracovávaná data.
- Values – definuje právě ukládané hodnoty.
- Where – omezující podmínka.
- OrderBy – způsob, jakým se mají získaná data seřadit.

Všechny tyto funkce vrací novou instanci třídy Query nastavenou tak, aby obsahovala stejná data jako předtím, navíc doplněná o aktuální příkaz, což umožňuje řetězení všech funkcí tak, aby byla výsledná podoba opravdu podobná syntaxi jazyka SQL, jak ukazuje následující příklad:

```
q.Selection("*").From("Users").OrderBy("UserName ASC")
```

Díky tomu, že programátor pouze zadává požadovaná data a nestará se o způsob, jakým se následně vytvoří výsledný SQL dotaz, se předchází spoustě zbytečných problémů, které by jinak mohly nastat.

3.4 Dědičnost a rozhraní IInheritable

V jazyce VBA může i běžné rozhraní obsahovat implementaci všech svých členů a tím pádem může být vytvořena jeho instance. Současným problémem ovšem je, že každý potomek, který je na tomto rozhraní založen, musí bezpodmínečně implementaci těchto členů přepsat svou vlastní, čímž se vzdává šance pracovat s původním kódem v základní třídě.

Jako řešení problému s neexistencí dědičnosti mezi základní třídou a jejím potomkem vzniklo rozhraní IInheritable, které je založeno právě na výše zmíněné vlastnosti jazyka VBA. Toto rozhraní je velice jednoduché a obsahuje pouze dvě vlastnosti:

- Base – odkazuje na rodičovský objekt, k němuž se přistupuje ve zděděných třídách.
- Current – odkazuje na aktuální instanci objektu. Je nastavitelná z dědících tříd a využívá se převážně v základní třídě pro přístup k datům.

Každá třída, u níž se předpokládá, že by měla v budoucnu sloužit jako základní třída svým potomkům, musí bezpodmínečně toto rozhraní implementovat, čímž dává jasně najevo, že je připravena sloužit jako rodič.

Součástí implementace rozhraní IInheritable je i úprava veškerého kódu tak, aby tohoto rozhraní využíval. To znamená, že třída musí ke svým veřejně přístupným datům nově přistupovat výhradně přes vlastnost Current. Změna v kódu vypadá takto:

Původní kód: `idColumn = Me.ObjectIDColumn`
Nový kód: `idColumn = Current.ObjectIDColumn`

Tato úprava zajistí, že se bude vždy využívat správné implementace ať už je umístěna v základní třídě nebo v jejím potomku.

Zděděná třída se musí explicitně přihlásit ke vztahu rodič-potomek, aby mohla využívat funkcí implementovaných v předkovi. Přihlášení probíhá tak, že potomek vytvoří instanci svého rodiče a zaregistruje se u ní nastavením vlastnosti Current. Od té chvíle lze přes vlastnost Base přistupovat k metodám rodiče a vše ostatní se řeší automaticky. Zjednodušená úprava ve zděděné třídě tak, aby využívala funkcí svého rodiče, vypadá takto:

```
Dim parent As IInheritable: Set parent = New ParentClass
parent.Current = Me
Dim id as Integer
id = parent.Base.RunMethodImplementedInParentClass()
```

Podmínkou pro správnou funkčnost tohoto řešení je, že rozhraní musí veřejně zpřístupnit všechny vlastnosti a metody, které nějakým způsobem pracují s daty a vnitřním stavem objektu, aby dal svým potomkům šanci tyto členy sám implementovat.

Zároveň je nezbytně nutné, aby potomek implementoval rodiče jako rozhraní, tzn., musí vytvořit vlastní implementaci všech veřejných členů a v nich volat odpovídající protějšky v základní třídě přes vlastnost Base.

I když je toto řešení poměrně omezené, pomáhá alespoň částečně obejít omezení daná použitou platformou a zpřístupňuje vývojáři možnost dodržovat Open-Closed Principle definovaný v návrhovém vzoru SOLID, který jinak není možné v jazyce VBA dodržet.

3.5 Manipulace s daty na straně VBA

Během implementace docházkového systému bylo potřeba vyřešit několik problémů zjištěných v analýze současného stavu. Předně se jednalo o různý způsob uložení dat v datovém úložišti oproti jazyku VBA. Neméně velkou výzvou pak bylo zajištění bezproblémové komunikace mezi oběma vrstvami, navíc doplněnou o nutnost určovat podmínky, za jakých se mají konkrétní data zpracovat.

3.5.1 ExtendedProperty a ExtendedPropertyCollection

Jako nástroj, který má pomoci vývojářům systému odstranit rozdíly mezi způsobem uložení dat v paměti počítače a perzistentním úložištěm vznikla třída ExtendedProperty. Jedná se o tenkou vrstvu, která odděluje samotná data a zbytek programu, který s nimi pracuje. Výhodou ExtendedProperty oproti standardní proměnné je schopnost ukládat vedle požadované hodnoty také dodatečné popisné informace, tzv. metadata.

V případě potřeby lze informace uložené v metadatech kdykoliv rozšířit. Standardní kolekci metadat nicméně reprezentují tyto vlastnosti:

- DisplayName – název vlastnosti tak, jak je použit v kódu VBA.
- Name – název vlastnosti v perzistentním úložišti.
- PropertyType – datový typ vlastnosti.
- Value – hodnota, kterou má vlastnost ukládat.
- DefaultValue – implicitní hodnota použitá ve chvíli, kdy není nastavena vlastnost Value.
- IsIdentifier – označuje danou vlastnost jako unikátní identifikátor.
- UseNullIfDefaultValue – říká, že se má místo implicitní hodnoty uložit NULL.
- IsOrderKey – označuje, zda se má vlastnost použít k seřazení dat.

Vlastnost uložená v ExtendedProperty je reprezentována jednak svou hodnotou, ale také jménem a datovým typem. Pro potřeby uložení dat v databázi je nutné datový typ uvádět explicitně pro každou vlastnost zvlášť.

Protože hodnotové datové typy musí být v jazyce VBA vždy nastaveny, obsahuje ExtendedProperty kromě běžné hodnoty i hodnotu implicitní, která je vrácena v případě, kdy není vlastnost nastavena vůbec, čímž simuluje hodnotu NULL. Zároveň ale není způsobena výjimka, pokud se program pokusí tuto hodnotu přiřadit do proměnné. Vhodným kandidátem pro implicitní hodnotu, může být prázdný řetězec pro textová pole, případně číslo -1 pro identifikátory objektů, která normálně mohou nabývat pouze kladných hodnot.

Pro usnadnění práce v perzistentním úložišti obsahuje ExtendedProperty přepínače, kterými označuje danou vlastnost jako unikátní identifikátor nebo jako pole, které může obsahovat hodnotu NULL. Podle nastavení těchto přepínačů se pak logika ukládání dat rozhoduje, zda tyto vlastnosti v úložišti nastavit, či nikoliv.

Objekty, které chceme ukládat v perzistentním úložišti, zpravidla obsahují více než jednu vlastnost. Z tohoto důvodu vznikla rozšiřující kolekce ExtendedPropertyCollection, jejímž úkolem je seskupovat libovolný počet souvisejících vlastností, ke kterým následně poskytuje jednoduchý přístup. Pro zajištění většího pohodlí vývojáři si kolekce vytváří instance jednotlivých vlastností zcela automaticky. Vývojář se tedy nemusí vůbec zabývat tím, jak jsou data vnitřně uložena, stačí k nim přistoupit přes název vlastnosti a nastavit hodnotu. Další nastavení, jako například definici datového typu dané vlastnosti lze provést později bez omezení.

3.5.2 Info objekt

Každý objekt, který má být uložen v databázi a u něhož požadujeme, aby byl programově editovatelný, je reprezentován Info objektem. V aktuálním řešení má svůj vlastní Info objekt každá databázová tabulka. Základním stavebním prvkem je rozhraní GeneralizedInfo, které definuje pouze dvě vlastnosti:

- ObjectID – vlastnost vracející unikátní číselný identifikátor objektu.
- Properties – kolekce ExtendedPropertyCollection všech vlastností, které mají být v rámci každého objektu uchovávány v perzistentním úložišti.

Vlastnost ObjectID je součástí Info objektu z historických důvodů a v případě nutnosti by mohla být odstraněna, protože samotný identifikátor dat musí být zároveň uložen ve

vlastnosti Properties. Vzhledem k četnosti použití této vlastnosti v kódu však byla zachována, protože její přítomnost ničemu nevadí.

Jednotlivé typy objektů musí toto rozhraní povinně implementovat, aby byla zajištěna jejich automatická synchronizace mezi pamětí počítače a perzistentním úložištěm. V rámci implementace je nutné správně nastavit kolekci Properties podle toho, jaká data má být schopna uchovávat. Následně může třída implementující GeneralizedInfo zpřístupnit tato data přes bezpečně typované vlastnosti a tím zvýšit bezpečnost při běhu programu.

3.5.3 Info provider

Info objekt působí jako pouhá schránka na data bez vlastní aplikační logiky, která by sama o sobě zařizovala ukládání nebo zpětné získávání dat z perzistentního úložiště. K tomu slouží třídy typu Info provider. Každý Info objekt má i svůj protějšek v podobě Info provideru, který dokáže komunikovat s perzistentním úložištěm a vyhledávat a upravovat v něm odpovídající záznamy. Funkcionalita byla rozdělena do dvou tříd, aby bylo vyhověno principu SRP.

Každý Info provider musí povinně implementovat rozhraní GeneralizedInfoProvider, který definuje následující veřejné metody a funkce, které navíc s výjimkou první jmenované i sám implementuje:

- InstantiateInfo – vytvoří novou instanci Info objektu, jehož data je schopen sám uložit.
- GetInfo – na základě unikátního identifikátoru získá z perzistentního úložiště odpovídající záznam a vrátí jej.
- GetInfoByName – podobně jako GetInfo vrátí Info objekt, tentokrát ale na základě jeho jména.
- GetInfos – vrátí pole Info objektů, volitelně omezené výběrovou podmínkou.
- SetInfo – uloží nebo upraví Info objekt v perzistentním úložišti.

Kromě těchto metod obsahuje rozhraní ještě definici třech veřejných vlastností. Vlastnost TableName slouží k identifikaci názvu entity v perzistentním úložišti, která slouží k ukládání Info objektů. Vlastnosti ObjectIDColumn a ObjectNameColumn obsahují

informaci o tom, v jakém sloupci se nachází unikátní identifikátor a unikátní jméno každého uloženého Info objektu.

GeneralizedInfoProvider implementuje rozhraní IInheritable, takže je schopen, opět s výjimkou funkce InstantiateInfo, poskytnout všem dědícím třídám obecné metody pro práci s perzistentním úložištěm. Ty se tak nemusejí starat o vlastní implementaci. V aktuálním řešení docházkového systému tuto možnost využívají úplně všechny objekty typu Info provider.

3.5.4 Statický Info provider

V celém řešení se neustále získávají, upravují a následně opět ukládají data v databázi. Tyto operace vždy zajišťují objekty typu Info provider, pro které by v běžném případě musela být vždy vytvořena nová instance, která zabere další místo v operační paměti počítače.

Z principu věci ale není potřeba, aby ve většině případů existovala více než jedna instance konkrétního Info provideru, protože jeho vnitřní stav nelze normálně změnit a tedy i při současném přístupu z více míst současně bude vracet vždy správné a očekávané výsledky. Jedinou výjimku v tomto případě tvoří instance vytvořené ve zděděných třídách za účelem zachování dědičnosti, jejichž vnitřní stav měnit lze.

Pro usnadnění práce při prostém získávání dat, což je nejčastější případ použití Info providera, byl vytvořen statický modul Provider. Ten vytváří a zpřístupňuje všechny Info providery jako tzv. singleton. Žádný z Info providerů není inicializován do chvíle, než si jej výkonný programový kód poprvé vyžádá. V tu chvíli se vytvoří jeho instance, která se uloží a od tohoto okamžiku se bude vždy používat právě ona. Použitím Info providerů z modulu Provider se jednak šetří paměť a další hardwarové prostředky počítače, ale hlavně se do značné míry zpřehledňuje a zkracuje výsledný kód aplikace.

3.5.5 Podmíněná manipulace s daty

Kdekoliv v systému je často potřeba pracovat jen s konkrétní sadou záznamů v databázi. Aby bylo vůbec možné definovat, které záznamy mají být ovlivňovány, vznikla obecná třída WhereCondition. Ta umožňuje v kódu VBA vytvořit podmínku, za jaké se mají data buď z úložiště získat, nebo v úložišti upravit.

Při návrhu se vycházelo ze syntaxe pro zápis podmínek v jazyce SQL, který je již programátorům blízký a je pro ně intuitivní. Podmínku lze rozdělit do několika sekcí, přičemž řešení každé sekce má nejvyšší prioritu. V každé sekci lze definovat výraz, sestávající z levého a volitelně pravého operandu, spojeného jedním z nabízených operátorů. Výrazy je možno propojovat pomocí logických operátorů Ano, Ne a Negovat.

Stejně jako třída Query zmíněná výše, je i WhereCondition postavena tak, aby bylo možné její modifikační funkce řetězit a tím dosáhnout člověkem čitelného zápisu, kterému bude snadné porozumět. Čitelnost je v tomto případě nezbytnou nutností, protože, na rozdíl od třídy Query, která se využívá pouze v jediném modulu, je WhereCondition použita ve velké míře po celém systému a v rámci snahy o dobrou udržovatelnost kódu vyžaduje perfektní čitelnost. Metody, které se používají pro budování podmínky a vracejí opětovně instanci třídy WhereCondition jsou následující:

- Where – definuje základní podmínku, kdy operand na levé straně musí odpovídat operandu na pravé straně na základě určeného vztahu.
- StartSection a EndSection – tyto metody zajistí vytvoření sekcí a umožňují tak určit prioritu vyhodnocování výrazů.
- AndThen, OrElse, Negate – logické operátory které ve stejném pořadí představují logický součin, logický součet a negaci.

Za pomoci těchto metod lze sestavit obecnou podmínku využitelnou při podmíněné selekci ovlivněných dat, jak ukazuje následující příklad:

```
c.Where("UsersUserID", Equals, 23) _
.AndThen() _
.Where("CheckInTime", GreaterEquals, #03/02/1992 08:00#)
```

Podmínka sama o sobě neposkytuje funkce, jejichž výstupem by byla její vlastní využitelná interpretace. Každý modul, který s podmínkou potřebuje pracovat ale má přístup k obecnému tvaru podmínky a může si ji ve správném formátu sestavit sám. V případě současné implementace docházkového systému se toto týká pouze tříd implementujících rozhraní IDataConnector.

3.6 Systém pro grafické rozhraní

Docházkový systém je aplikace založená na platformě Microsoft Access, a jako takový využívá jeho možností pro tvorbu grafického rozhraní. Tyto možnosti spočívají v tvorbě jednoduchých formulářů, s jejichž pomocí uživatelé manipulují s daty.

Protože grafické rozhraní samotné je silně závislé na použité technologii, není ho možné navrhnout s ohledem na snadnou přenositelnost do jiného prostředí. V tomto případě to však není na škodu, neboť slouží k pouhé grafické prezentaci dat a na jiné platformě by na něj pravděpodobně byly kladeny jiné požadavky.

Pro zjednodušení a sjednocení práce s grafickým rozhraním vznikl modul UIManager, který má na starosti zajištění správného vykreslení každého formuláře. Jeho hlavním úkolem je zpřístupňovat konkrétní části formuláře pouze těm uživatelům, kteří na to mají právo. Možnosti UIManageru jsou probrány dále v této kapitole.

3.6.1 Autentizace uživatele

Při vstupu do administračního rozhraní je důležité uživatele autentizovat a na základě přidělených práv zpřístupnit jen ty části systému, do kterých by měl mít přístup. Uživatel, který není zaměstnancem, nesmí být do administračního rozhraní vpuštěn.

Problém, který vyvstal během implementace, bylo zjištění, že všechny počítače ve společnosti GISoft jsou sice fyzicky zapojeny do jedné sítě, ale ne všichni uživatelé jsou napojeni do firemní domény. Z tohoto důvodu není možné využít standardní autentizaci operačního systému Windows.

Druhé řešení, jak autentizaci provést, je osobní uživatelské jméno a heslo. Toto řešení ale nebylo přijato, protože by po uživatelích při každém přihlášení do systému vyžadovalo zadávat přihlašovací údaje, což bylo identifikováno jako nepřekonatelná překážka v bezproblémovém fungování systému.

Nakonec bylo domluveno, že se uživatelé budou bez hesla přihlašovat pod jménem složeným z domény a uživatelského jména, pod kterým se přihlašují do své pracovní stanice. Využívá se zde faktu, že jakýkoliv účet, byť není napojen na doménu Active Directory, má přidělenu nějakou doménu, nejčastěji shodnou s názvem počítače. Jedná se o nejjednodušší, ale zároveň nebezpečné řešení, neboť kdokoliv, kdo zná název

domény a uživatele, se může vydávat za cizí osobu. Vedení GISoftu s daným řešením nicméně souhlasí, neboť ze strany zaměstnanců neočekává podvodné jednání a opatření proti neoprávněnému přístupu osoby zvenčí má podnik zajištěn dostatečně.

Autentizační modul se kvůli výše zmíněnému problému stará o pouhou identifikaci uživatele, aby mohl zbytek systému uživateli poskytovat relevantní data. Při prvotním přístupu je každému uživateli automaticky vytvořen profil, který ovšem není aktivovaný, takže uživatel nevidí na úvodní obrazovce nic jiného než informační hlášku a nemůže provádět žádné akce. Jakmile mu profil jeden z administrátorů aktivuje a přidělí odpovídající roli oprávnění, považuje se uživatel jako autentizovaný a může možnosti docházkového systému využívat naplno.

3.6.2 Metadata pro komponenty

Při vytváření formuláře se často objeví nutnost nastavit konkrétnímu ovládacímu prvku speciální příznak, který je pak možno použít v kódu. V navrhovacím režimu programu Access lze každému ovládacímu prvku nastavit širokou škálu vlastností ovlivňující jejich vzhled a chování. Access ale neumožňuje vytvářet takové vlastnosti dodatečně.

Přestože lze speciální vlastnosti nastavovat a uchovávat v kódu, bylo by daleko přehlednější a méně náchylné na chyby, kdyby se mohly jednoduše nastavit přímo u komponenty v designeru formuláře. Access naštěstí pro každý ovládací prvek poskytuje jednu univerzální vlastnost nazvanou Tag, do které lze uložit jakoukoliv informaci.

Vlastnost Tag je v docházkovém systému využita pro zápis řídících příkazů ve velmi jednoduchém makro jazyku. Řídící příkaz může být buď jednoduchý, pak je definován pouze svým názvem a hovoříme o něm jako o příznaku či přepínači, nebo složený. Složený příkaz je definován svým názvem, za nímž následuje znak pro rovnost a data, která má příkaz použít jako parametr. Pokud je potřeba uložit pro jeden ovládací prvek více řídících příkazů, je možné je oddělit středníkem.

Tento textový řetězec: „DoNotTranslate;AllowedRoles=Developer“, nastavený na tlačítku např. zajistí, aby bylo tlačítko dostupné pouze uživatelům v roli Developer a zároveň, aby se textový popisek v tlačítku nenahrazoval souvisejícím řetězcem z tabulky ResStrings.

Tímto způsobem lze nastavit spoustu vlastností formuláře již v době návrhu bez nutnosti upravovat kód běžící na pozadí formuláře. Navíc je makro jazyk odolný vůči překlepům v příkazech, protože ty, které nepozná, přeskakuje.

3.6.3 Standardní prvky formuláře

Předpokladem pro správnou funkci UIManageru a s tím spojeného zajištění kontroly práv uživatele, je přítomnost dvou speciálních prvků na každém formuláři. Těmito prvky jsou:

- Popisek pro zobrazení libovolného chybového hlášení s příznakem ErrorMessage.
- Tlačítko umožňující formulář i v případě chyby opustit s příznakem ErrorEscaper.

Popisek pro zobrazení chybového hlášení musí být na formuláři přítomný zejména z hlediska uživatelské přívětivosti, aby uživatel při nastalé chybě věděl o tom, co se děje.

Naopak tlačítko pro odchod z formuláře musí být ve formuláři umístěno z technických důvodů. V případě, kdy např. není uživatel aktivován a UIManager musí schovat všechny ovládací prvky, totiž dochází k nezachytitelné výjimce při pokusu o schování jakéhokoliv prvku, který je právě aktivní. Proto musí na formuláři zůstat alespoň jeden prvek, který může být aktivován a tím je právě toto tlačítko. UIManager jej v případě potřeby aktivuje, tak aby zbytek ovládacích prvků mohl bez problému schovat.

3.6.4 Zabezpečení a přidělení práv uživateli

Formulář, který chce využívat kontrolu práv uživatele, musí minimálně při zavedení formuláře zavolat metodu v modulu UIManager s názvem HideUnavailableUI. UIManager v ní projde celý formulář a identifikuje všechny ovládací prvky. Pro každý prvek následně testuje, zda je jeho přístupnost omezena na konkrétní role a v případě, že uživatel v této roli není, prvek znepřístupní.

Právě kvůli znepřístupňování ovládacích prvků v této metodě je vyžadováno, aby formulář obsahoval tlačítko pro odchod z formuláře, které nelze schovat a je automaticky aktivováno při schovávání zbytku rozhraní.

Samotné nastavení ovládacího prvku tak, aby byla jeho přístupnost omezena, probíhá přímo v designeru formuláře použitím příkazu AllowedRoles. Ten je schopen jako parametr převzít i více než jednu povolenou roli. V takovém případě musí být role odděleny čárkou.

3.6.5 Automatické překlady grafického rozhraní

Díky databázovým tabulkám ResStringCategories a ResStrings je možné uchovávat všechny textové řetězce na jednom místě v logických celcích. Jejich využitím v grafickém rozhraní se usnadňuje editace formulářů, protože pro úpravu textů není potřeba zasahovat do kódu nebo do vlastností ovládacích prvků v designeru formulářů.

Vlastnostem ovládacího prvku, které mají obsahovat uložený řetězec, stačí v designeru nastavit kódový název požadovaného řetězce a následně při inicializaci formuláře zavolat metodu UIManageru TranslateForm. Tato metoda projde všechny podporované prvky ve formuláři, získá z nich kódové označení řetězce a na jeho místo dosadí novou hodnotu získanou z databáze.

Metoda TranslateForm předpokládá, že všechny prvky ve formuláři jsou přeložitelné a v případě, že v databázi nenalezne odpovídající záznam, způsobí výjimku. Protože existují případy, kdy není potřeba, aby byl ovládací prvek přeložen, existuje řídící příkaz DoNotTranslate, který, pokud je na prvku nastaven, způsobí, že jej metoda přeskočí a výjimku nezpůsobí.

Získávání uložených řetězců z databáze je poměrně náročné, zvlášť u obecných řetězců, které se využívají na více místech v systému, proto jsou všechny již získané řetězce cachovány a primárně se vybírají právě z cache. Tím se do značné míry uleví databázi od zbytečné zátěže a i samotný překlad probíhá rychleji, obzvlášť při druhém otevření téhož formuláře.

3.7 Administrace uživatelských účtů

Důležitým modulem v rámci docházkového systému je administrace jeho uživatelů. Díky administraci je možné nastavit práva každému uživateli. Uživatelé také mají přístup ke svým datům, která mohou kdykoli kontrolovat a nahlašovat případné chyby.

3.7.1 Uživatelské role

Každý uživatel vedený v docházkovém systému je přiřazen do jedné z dostupných rolí, které určují jeho práva. Aktuálně jsou do systému zavedeny tři role pravomocí – Reader, Administrator a Developer, přičemž jen první dvě jsou určeny k běžnému použití. Role Developer je v systému přístupná pouze pro servisní účely.

Standardně je uživatel po svém vytvoření zařazen do role Reader. V této roli může pouze prohlížet své vlastní záznamy, k záznamům ostatních nemá přístup. Pouze ve výjimečných případech je mu umožněno manipulovat s daty, ale i to probíhá takovým způsobem, aby si tento fakt vůbec neuvědomoval, např. pomocí tlačítek zaznamenává svůj příchod a odchod. Role Reader je určena pro řadové zaměstnance.

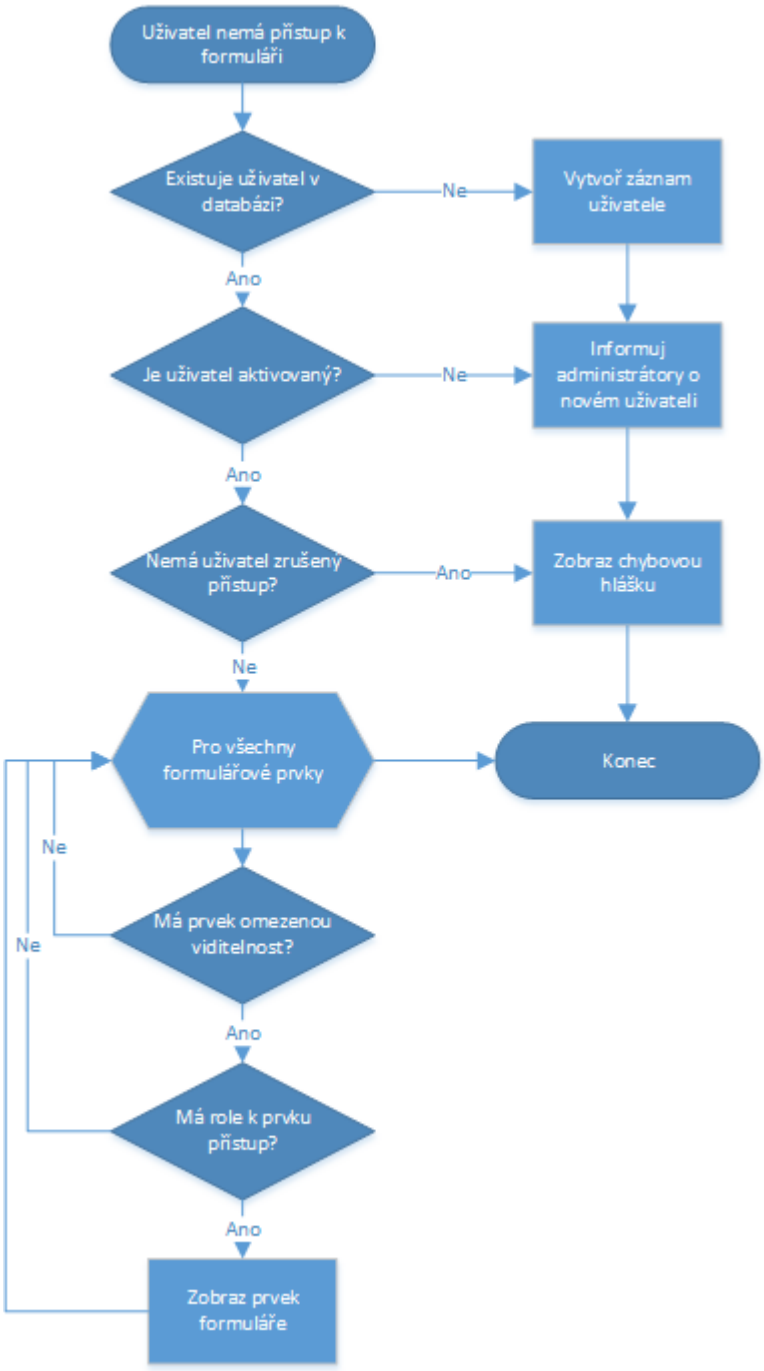
Vedoucí pracovníci, kteří jsou odpovědni za své podřízené a mají mít možnost editovat jejich záznamy, např. z důvodu oprav, jsou přiřazeni do role Administrator. Administrátoři mají podstatně bohatší možnosti v tom, co mohou v systému dělat. Protože mají pravomoc editovat data všech uživatelů a tedy i svých, je nanejvýš vhodné, aby takových pracovníků nebylo příliš mnoho, jinak by hrozily podvody ve vedení docházky.

Servisní role Developer by měla mít oprávnění na úplně všechny funkce systému, protože je určena k vývoji a údržbě systému, a potřebuje všechny části systému testovat. Developer má jako jediný přístup do vývojářské konzole, ze které smí spouštět automatické testy.

3.7.2 Zavedení uživatele do systému

Ve chvíli, kdy uživatel poprvé otevře aplikaci docházkového systému, je pro něj automaticky vytvořen uživatelský účet. Přestože má uživatel svůj účet již vytvořen a má dokonce přidělenu roli Reader, není schopen vidět žádný ovládací prvek, protože ještě nebyl aktivován.

Uživatele může aktivovat jedině administrátor systému. Ten je při přihlášení do systému okamžitě informován o tom, že se v systému objevil nový uživatel, který čeká na aktivaci. Jediným kliknutím se pak může přesunout do administračního rozhraní, ve kterém uvidí karty všech neaktivovaných uživatelů, kde je může postupně povolit. Od té chvíli se může nový uživatel bez problému přihlašovat, pracovat s prvky formuláře příslušející jeho roli oprávnění a číst svá data.



Obrázek č. 2: Vývojový diagram autentizace uživatele a zpřístupnění UI
(Zdroj: vlastní zpracování)

3.7.3 Administrační rozhraní

Administrační rozhraní pro správu uživatelů tvoří jeden formulář. Pokud je uživatel běžným zaměstnancem, má přístup pouze ke svému profilu, ve kterém navíc vidí pouze některá pole. Tato pole nemůže ani editovat. Formulář mu ale například poskytuje

informaci o zbývající dovolené, což byl jeden z požadavků na docházkový systém kladený společností GISoft.

Naproti tomu uživatel v roli administrátora má právo přistupovat k libovolnému profilu, který může rovněž editovat a v případě, že je uživatel úplně nový, i aktivovat. Administrátor má oprávnění uživatele zablokovat. Zablokovaný uživatel se nemůže přihlásit do systému a číst data v něm uložená. Tato možnost je do systému přidána pro případ, že některý zaměstnanec v GISoftu skončí. Takto se uživatel nemůže do systému přihlásit, ale zároveň nejsou ztracena jeho uložená data.

3.8 Zadávání docházky

Hned poté, co uživatel vstoupí do docházkového systému, je mu zobrazena úvodní obrazovka, na které může zaznamenávat svou docházku nebo je informován o chybách v jeho docházce, např. z důvodu neukončení předchozí pracovní směny.

3.8.1 Zadávací formulář

Úvodní formulář obsahuje několik tlačítek, každé s vlastní funkcí. Pokud tedy zaměstnanec přijde do práce, může hned po otevření systému zaznačit svůj příchod. V průběhu dne pak nahlašuje odchody a návraty z přestávek a na konci dne zaznamená odchod z práce.

Formulář obsahuje i možnost nahlásit odjezdy a návraty z pracovních cest, odchod na dovolenou apod. přesně podle zadání vedení společnosti GISoft. Na tomto formuláři se také objevují varování v případě, že nemá uživatel správně uzavřen záznam o docházce.

3.8.2 Výpisy a editace údajů o docházce

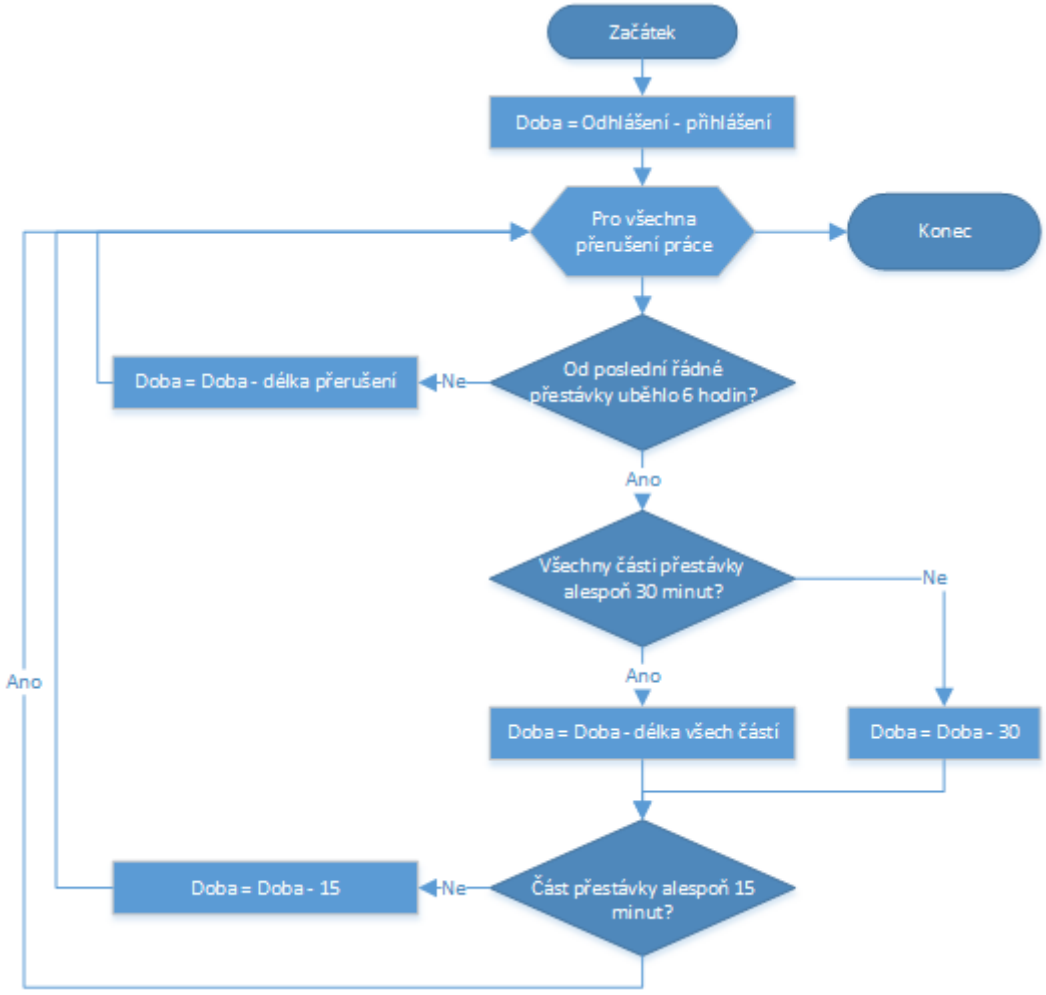
Své údaje o docházce mohou zaměstnanci sledovat v měsíčních výpisech, ve kterých jsou zaznamenány jednak časy příchodů a odchodů, ale také celková vypočítaná odpracovaná doba.

Uživatelé v roli Reader mají právo číst pouze své záznamy, kdežto administrátoři mají přístup k záznamům všech uživatelů, navíc s možností editace. Na tomto formuláři mohou administrátoři provádět korekce docházky v případech nekonzistence záznamů, a proto se jedná o velmi důležitou součást celého systému.

3.8.3 Algoritmus pro výpočet odpracované doby

Algoritmus počítající počet odpracovaných hodin pracuje s přesností na minuty. Jako výchozí hodnotu pro výpočet bere rozdíl mezi příchodem a odchodem z práce. Tuto hodnotu následně iterativně upravuje o délky všech přestávek, které si zaměstnanec během dne vybral.

Algoritmus je schopen si poradit s rozdělením jedné přestávky do více části, přičemž v takovém případě hlídá, aby alespoň jedna část dosáhla zákonem požadované délky 15 minut. Pokud jí nedosáhne, systém zaměstnanci přestávku automaticky prodlouží. Systém rovněž automaticky zahajuje zaměstnanci přestávku v případě, že ji sám nenastoupí do šesti hodin od zahájení pracovní směny.



Obrázek č. 3: Vývojový diagram výpočtu odpracované doby
(Zdroj: vlastní zpracování)

3.9 Vývojářské nástroje

Prostředkem pro usnadnění vývoje se stal formulář DevelopmentTools, který je dostupný pouze vývojářům v roli Developer. Protože se nejedná o veřejně přístupný formulář, je úplně vytržen z kontextu zbytku aplikace a zobrazování není řízeno UIManagerem.

Nejdůležitější funkcí tohoto nástroje je usnadnění spouštění automatických testů tak, aby bylo zachováno správné pořadí prováděných akcí a testy mohly proběhnout bez problému a bez chyb.

Formulář DevelopmentTools dále umožňuje mazat cache textových řetězců, aby bylo možné předcházet chybám spojených s úpravou nacachovaných záznamů. Bez smazání cache by totiž až do restartu aplikace byly používány neaktualizované řetězce.

3.10 Zabezpečení a nasazení systému

Aby se předešlo nechtěným nebo neoprávněným změnám, zajišťuje aplikace uzamknutí veškerého ovládací rozhraní Microsoft Access ihned po startu. Jedinou výjimku tvoří uživatelé v roli Developer, jimž se rozhraní neuzamkne nikdy. Pojistkou proti situaci, kdy v systému není veden žádný vývojář, je zajištění povýšení prvního přihlášeného uživatele do role Developer. K systému proto nemůže společnost nikdy ztratit přístup.

Ve společnosti GISoft byla aplikace docházkového systému v podobě jediného souboru formátu *.accdb nahrána na centrální firemní server. Tento server není přístupný z vnějšího prostředí a nehrozí proto odcizení uložených údajů. Zaměstnanci se do systému přihlašují prostým spuštěním zástupce na aplikaci, který může být umístěn např. na ploše operačního systému jejich pracovní stanice.

3.11 Ekonomické zhodnocení

Podmínkou společnosti GISoft, kromě požadavků na funkce docházkového systému, byla minimální pořizovací cena celého řešení, včetně dodatečných nákladů spojených s jeho údržbou a administrací.

Na trhu existují i produkty s otevřeným programovým kódem, které lze již z principu využívat zcela zdarma. Jak se ale později ukázalo, všechny tyto systémy jsou náročnější na instalaci i administraci. Vedení podniku zamítlo možnost nasazení takového softwaru,

neboť by vyžadoval složitější zásahy do infrastruktury a vyhrazení IT odborníka, který by na systém neustále dohlížel.

Výše zmíněné omezilo možnosti pro nasazení docházkového systému na jednoduchou vlastní implementaci vytvořenou z již dostupných technologií. Touto cestou lze sice snížit náklady za nákup softwaru na minimum, na druhou stranu ale zvyšuje cenu řešení vinou zvýšené doby implementace.

Po celou dobu praktické realizace systému se vývoji věnoval jeden člověk. Pro výpočet skutečných nákladů na pořízení docházkového systému jsou využity odhady doby implementace jeho jednotlivých částí. Jako referenční mzda, která by byla vyplácena člověku odpovědnému za implementaci systému, byla zvolena částka 200 Kč za hodinu. Tato částka zhruba odpovídá požadavkům méně zkušeným vývojářům v Moravskoslezském kraji (Český statistický úřad, 2012).

Jednotlivé části docházkového systému byly časově ohodnoceny takto:

- Teoretická příprava a analýza – 10 hodin.
- Podpora pro automatické testy – 27 hodin.
- Struktura databáze a komunikace s databází – 43 hodin.
- Zavedení principů objektově orientovaného programování do systému – 18 hodin.
- Algoritmy pro výpočty docházky – 10 hodin.
- Manažer grafického rozhraní – 8 hodin.
- Formuláře pro interakci s uživatelem – 17 hodin.
- Dodatečné práce – 4 hodiny.

Celkový odpracovaný čas na docházkovém systému činí 137 hodin. Z této hodnoty vyplývá, že za podpory stávajících technologií byla jedna osoba schopna docházkový systém implementovat za necelé tři týdny práce.

Výsledná odpracovaná doba znamená, že celkové náklady spojené s pořízením docházkového systému by činily 27 400 Kč. Do nákladů by se totiž promítla pouze odměna pro vývojáře, protože společnost GISoft již disponovala veškerým potřebným softwarovým vybavením a měla vybudovánu postačující infrastrukturu.

Přínos pro podnik v současné době spočívá především ve zvýšení informovanosti zaměstnanců o své docházce, což by je mělo motivovat k svědomitějšímu přístupu k zaměstnání. Zároveň umožňuje výrazně zrychlit některé administrativní úkony, jako výpočet mezd, zjišťování zbývajících dnů dovolené, apod. Pro podnik je nesmírně výhodné, že byl celý systém postaven tak, aby byl jednoduše modifikovatelný, takže v případě dodatečných úprav, nebo jen změn v důsledku novelizace příslušných zákonů, není nutné vyhledávat specializovaného správce, ale může je provést kterýkoliv zkušenější vývojář za velmi krátkou dobu, tedy velmi levně.

ZÁVĚR

Projekt implementace docházkového systému byl již od začátku limitován podmínkami, které mohl GISoft nabídnout. Protože vývoj probíhal ve specifických podmínkách, kdy představitelé podniku na jednu stranu chtěli být informováni o docházce svých zaměstnanců, na druhou stranu ale tuto informaci vyloženě nutně nepotřebovali, bylo již od počátku jasné, že hlavním úkolem při implementaci bude vyjít ze stávajících řešení a pokud možno je využít na sto procent.

Velké obavy ze strany vedení rovněž směřovaly na potřebu správy a údržby takového systému. Proto požadovalo maximální jednoduchost obsluhy systému, který se o sebe bude schopen do značné míry starat sám.

Z nemnoha možností, které se od počátku nabízely, byla jako hlavní technologie, na které bude systém později vybudován, uvažována databázová aplikace Microsoft Access. Ta se jevila jako nejvhodnější kandidát, především díky svému zaměření, které se do značné míry shodovalo s potřebami docházkového systému. Přestože se při detailnějším rozboru objevilo několik překážek, které by mohly později způsobovat problémy, bylo rozhodnuto, že právě Access bude skutečně sloužit jako základ pro vybudování docházkového systému. Z ekonomického hlediska se jednalo o logickou volbu, neboť tato aplikace byla dostupná na všech klientských počítačích a nebyly tak zapotřebí žádné další investice do softwaru nebo síťové infrastruktury.

Technické řešení systému je v současné době silně závislé na jazyku VBA, který je přímou součástí aplikace Access. Veškeré obavy, které vzešly z výběru platformy, byly zažehnány implementací vlastních, promyšlených řešení.

Velice důležitou součástí systému se staly automatické testy a platforma, která umožňuje jejich spouštění. Přínos automatických testů nebývá často dostatečně oceněn, ale při implementaci docházkového systému několikrát napomohly odhalit chyby ještě předtím, než mohly napáchat vážnější škody někdy v budoucnu.

Automatickými testy byla pokryta většina výkonného kódu, z největší části to však byly moduly zajišťující komunikaci s perzistentním úložištěm a to z toho důvodu, že práce s uloženými daty je klíčovou, ne-li nejpodstatnější funkcí docházkového systému. Manipulace s daty byla pro vývojáře do značné míry zjednodušena tak, aby se nemuseli

vůbec starat o způsob, jakým jsou data získávána a ukládána, ale aby pouze definovali, jaká data požadují, a o zbytek se postaral systém sám. Na toto zobecnění přístupu k datům přímo navazuje myšlenka univerzálního perzistentního úložiště, které je v současné době reprezentováno databází v Microsoft Access, ale řešení je připraveno kdykoliv přejít na jiný systém ukládání dat, aniž by byly ovlivněny funkce docházkového systému.

Uživatelská přívětivost systému je zajištěna komponentami, které uživatelům poskytují pouze ta data, která jsou v současné chvíli důležitá, a ke kterým má mít uživatel přístup. Grafické rozhraní bylo navrženo tak, aby byly uživateli zobrazeny smysluplné chybové hlášky, které jej nasměrují na řešení problému v nejkratším možném čase tak, aby se v práci nemusel zabývat ničím, co není hlavní náplní jeho zaměstnání. I přes závislost na jazyku VBA byl systém navržen tak, aby byla migrace na jinou technologii ať už pro uchovávání, tak i pro prezentaci dat, umožněna v podstatě kdykoliv.

Docházkový systém se při výpočtu odpracovaných směn výhradně drží platných zákonů České republiky. Systém tedy může být v podniku nasazen a data exportovaná z databáze mohou být využita v účetním softwaru pro výpočty mezd všech zaměstnanců.

Kromě obtížně řešitelných problémů s autentizací uživatelů byl celý docházkový systém naimplementován za necelé čtyři týdny s celkovými náklady nedosahujícími ani 30 000 Kč, do kterých nemusely být započítávány částky na pořízení nového hardwaru ani softwaru, protože bylo použito výhradně stávajícího vybavení.

SEZNAM POUŽITÉ LITERATURY

ANET-ADVANCED NETWORK TECHNOLOGY, 2011. Ceník docházkového systému aTime. *aTime* [online] [cit. 2013-14-12]. Dostupné z: http://www.atime.eu/atimeportal/jak-se-stat-uzivatelem/cenik.aspx

BEAULIEU, A., 2009. *Learning SQL*. Second Edition. Sebastopol: O'Reilly Media. ISBN 978-0-596-52083-0.

BM SOFTWARE, 2013. Docházkový systém Docházka 3000 - Software. *Dochazka.eu* [online] [cit. 2013-12-14]. Dostupné z: http://www.dochazka.eu/dochazka3000/shop/doc_d3000sw.pdf

BM SOFTWARE. Docházkové terminály. *Dochazka.eu* [online]. [cit. 2013-12-14]. Dostupné z: http://dochazka.eu/dochazka3000/terminaly/index.html

ČESKO. Zák. č. 262/2006 Sb. ze dne 21. dubna 2006 zákoník práce.

ČESKÝ STATISTICKÝ ÚŘAD, 2012. Mzdy IT odborníků v České republice. *ČSÚ* [online] [cit. 2014-05-21]. Dostupné z: http://www.czso.cz/csu/redakce.nsf/i/mzdy_it_odborniku_v_ceske_republice/$File/it_platy_2012.pdf

FOLEY, M. J., 2011. Microsoft to focus on HTML5 and JavaScript for Office 15 extensions. *ZDNet* [online] [cit. 2014-03-22]. Dostupné z: http://www.zdnet.com/blog/microsoft/microsoft-to-focus-on-html5-and-javascript-for-office-15-extensions/10266

GISOFT. Profil firmy GISoft. *gisoft.cz* [online]. [cit. 2014-02-22]. Dostupné z: http://www.gisoft.cz/GISoft/GISoft

GOODHEW, T., 1996. Jet Engine: History. *Australian Visual Developers Forum* [online] [cit. 2013-12-30]. Dostupné z: http://www.avdf.com/nov96/acc_jet.html

GRASSEOVÁ, M., 2007. Využití SWOT analýzy pro dlouhodobé plánování. In: *Obrana a strategie* [online]. 20. 3. 2007 [cit. 2013-11-19]. Dostupné z: http://www.defenceandstrategy.eu/cs/archiv/rocnik-2006/2-2006/vyuziti-swot-analyzy-pro-dlouhodobe-planovani.html

GREEN, J. et al., 2007. *Excel 2007 VBA Programmer's Reference*. Indianapolis: Wiley Publishing. ISBN 978-0-470-04643-2.

HAMMIL, P., 2004. *Unit Test Frameworks: Tools for High-Quality Software Development*. First Edition. Sebastopol: O'Reilly Media. ISBN 0-596-00689-6.

HARKINS, S. S. a M. GUNDERLOY, 2004. *Automating Microsoft Access with VBA*. Indianapolis: Que Publishing. ISBN 0-7897-3244-0.

HYPERPICS, 2010. HyperPics. In: *Book Center* [online].©2000-2010 [cit. 2013-11-17]. Dostupné z: Dostupné z: http://www.hyperpics.com/eBooks/Intro_to_VBA_for_AutoCAD/Introduction_to_VBA_for_AutoCAD_(Mini_Guide).pdf

CHUNG, L., 2002. Microsoft Access History: Whitnessing the Launch of Access and its Success Over Borland in 1992 by …. *FMS* [online] [cit. 2013-12-30]. Dostupné z: http://www.fmsinc.com/MicrosoftAccess/history/index.html

JONÁŠ, M., 2012. Návrhové principy: SOLID. *Zdroják* [online] [cit. 2013-12-18]. Dostupné z: http://www.zdrojak.cz/clanky/navrhove-principy-solid/

LIBERTY, J., 2007. *Naučte se C++ za 21 dní*. Brno: Computer Press. ISBN 978-80-251-1583-1.

LOMAX, P., 1998. *VB & VBA in a Nutshell: The Language*. Sebastopol: O'Reilly & Associates. ISBN 1-56592-358-8.

MARTIN, R., 2000. Design Principles and Design Patterns. *Object Mentor* [online] [cit. 2013-18-12]. Dostupné z: http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf

MICROSOFT CORPORATION, 2000. File-Server vs. Client/Server. *Office development* [online] [cit. 2013-12-30]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/aa165278(v=office.10).aspx

MICROSOFT CORPORATION, 2006. Where did the name for Microsoft Access come from? *MSDN Blogs* [online] [cit. 2013-30-12]. Dostupné z: http://blogs.msdn.com/b/oldnewthing/archive/2006/04/13/575739.aspx

MICROSOFT CORPORATION, 2007. Database basics - Access. *Office.com* [online] [cit. 2013-30-12]. Dostupné z: http://office.microsoft.com/en-us/access-help/database-basics-HA010064450.aspx

MICROSOFT CORPORATION, 2010. What's new in Microsoft Access. *Office.com* [online] [cit. 2014-03-20]. Dostupné z: http://office.microsoft.com/en-us/access-help/what-s-new-in-microsoft-access-HA010342117.aspx

MICROSOFT CORPORATION, 2012. Introducing Access 2013. *Access Blog* [online] [cit. 2013-12-30]. Dostupné z: http://blogs.office.com/b/microsoft-access/archive/2012/07/20/introducing-access-2013-.aspx

MICROSOFT CORPORATION, 2013. What's new for Access 2013 developers. *Microsoft Developer Network* [online] [cit. 2014-20-03]. Dostupné z: http://msdn.microsoft.com/en-us/library/office/jj250134(v=office.15).aspx

OBJECT MENTOR, 2006. SRP: The Single Responsibility Principle. *Object Mentor* [online] [cit. 2013-12-18]. Dostupné z: http://www.objectmentor.com/resources/articles/srp.pdf

OR-CZ, 2013. DOCHÁZKOVÝ SYSTÉM. *Skupina OR* [online] [cit. 2013-12-14]. Dostupné z: http://www.orcz.cz/www/www-new.nsf/DOCHAZKOVY_SYSTEM_2010.pdf

RON SOFTWARE, 2013. Ceník. *RON Software* [online] [cit. 2013-14-12]. Dostupné z: http://cms.ron.cz/files/attachments/610/9007-dochazka_hw_cz.pdf

ROTHWELL, P., 2010. What Is A SWOT Analysis? In: *Freshbussinessthinking.com* [online]. 25. 5. 2010 [cit. 2013-11-19]. Dostupné z: http:// www.freshbusinessthinking.com/business_advice.php?AID=5743

SEZNAM TABULEK

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

Výsledný program se nachází na přiloženém CD.