VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH A TVORBA INFORMAČNÍHO SYSTÉMU PRO OPTIKY

DESIGN AND CREATION OF INFORMATION SYSTEM FOR OPTICIANS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Vít Komínek
AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Dydowicz, Ph.D.
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání diplomové práce

Ústav:	Ústav informatiky
Student:	**Bc. Vít Komínek**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Informační management
Vedoucí práce:	**Ing. Petr Dydowicz, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů a se Studijním a zkušebním řádem VUT v Brně zadává diplomovou práci s názvem:

Návrh a tvorba informačního systému pro optiky

Charakteristika problematiky úkolu:

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

Cíle, kterých má být dosaženo:

Hlavním cílem této práce je návrh a vytvoření zcela nového informačního systému, který bude směřovat svojí funkcionalitou zejména na oční optiky. Vytvářený informační systém bude jednoduchý, intuitivní a zajistí veškerou funkcionalitu pro podporu práce zaměstnanců očních optik. Požadovaná funkcionalita informačního systému bude analyzována ve spolupráci s majiteli a provozovateli očních optik, jimž dosavadní informační systém buď nevyhovuje či jej doposud nevyužívají. Tyto oční optiky se mohou stát budoucími uživateli vytvořeného systému. Informační systém má být navržen obecně pro všechny menší a středně velké oční optiky, tudíž při návrhu funkcionality a designu se budu inspirovat i ze stávajících řešení. Informační systém bude umožňovat propojení s dalšími CRM či ERP systémy pomocí můstků na úrovni Microsoft SQL databází.

Základní literární prameny:

BASL, J. a R. BLAŽÍČEK. Podnikové informační systémy. Podnik v informační společnosti. Praha: Grada, 2008. 283 s. ISBN 978-80-247-2279-5.

MOLNÁR, Z. Automatizované informační systémy. Praha: Strojní fakulta ČVUT, 2000. 126 s. ISBN 80-01-02269-2.

MOLNÁR, Z. Efektivnost informačních systémů. Praha: Grada Publishing, 2000. 142 s. ISBN 80-716--410-X.

ŘEPA, V. Analýza a návrh informačních systémů. Praha: Ekopress, 1999. 403 s. ISBN 80-86119--3-0.

SODOMKA, P. a H. KLČOVÁ. Informační systémy v podnikové praxi. Brno: Computer Press, 2010. 501 s. ISBN 978-80-251-2878-7.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně dne 29.2.2020

L. S.

doc. RNDr. Bedřich Půža, CSc. ředitel	doc. Ing. et Ing. Stanislav Škapa, Ph.D. děkan

Abstrakt

Diplomová práce je zaměřena na analýzu, návrh a tvorbu vlastního informačního systému vytvořeného pro oční optiky. Informační systém bude provázet uživatele v převážné většině procesů spojených s provozem pobočky oční optiky.

Klíčová slova

návrh, informační systém, optik, SQL, databáze, proces, Delphi, analýza

Abstract

This master’s thesis is focused on the analysis, design and creation of an information system created for ophthalmic opticians. This information system will accompany users in the vast majority of processes associated with the operation of the optician's branch.

Key words

design, information system, optician, SQL, database, process, Delphi, analysis

Bibliografická citace

KOMÍNEK, Vít. Návrh a tvorba informačního systému pro optiky [online]. Brno, 2020 [cit. 2020-05-17]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/127516. Diplomová práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Petr Dydowicz.

Čestné prohlášení

Prohlašuji, že předložená diplomová 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 17. května 2020

……..

podpis studenta

Poděkování

Moje poděkování patří především panu Ing. Petru Dydowiczovi, Ph.D. za vedení mé diplomové práce, poskytnuté odborné rady a pomoc při řešení problémů.

OBSAH

ÚVOD

V současném světě neustálého vývoje a posunu technologií vpřed, je téměř nepředstavitelné, aby cokoli fungovalo bez současných technologií. Pro firmy toto platí mnohonásobně více. Pokud firma nevyužívá digitálních technologií, zejména informačního systému, pak lze konstatovat, že nepodniká efektivně. Informační systémy totiž velmi urychlují a zefektivňují chod firmy. Stává se tak nedílnou součástí firem.

Na trhu se vyskytuje velké množství informačních systémů, a tedy i jejich široký výběr pro většinu odvětví podnikání. Spousta stávajících informačních systémů je však zastaralá, a pokud podnikatel uvažuje o jeho pořízení, výběr se mu zcela zúží. Většina podniků pak potřebuje informační systém na míru přizpůsobený svému oboru podnikání.

Ve svém okolí jsem poznal již řadu poboček očních optik, které doposud nevyužívaly žádného digitálního informačního systému nebo byl zkrátka velmi zastaralý bez aktualizací a údržby. Rozhodl jsem se pro to navrhnout a vytvořit zcela nový informační systém, který bude zcela přizpůsobený podnikům očních optik.

Informační systém je velmi komplexní a složité řešení, které pro správnou funkcionalitu musí být tvořeno a konzultováno společně se zaměstnanci a vedením poboček očních optik, kteří nejlépe vědí, co od něj potřebují.

Smyslem této práce je přinést zcela nové a moderní řešení informačního systému, který pokryje potřeby zaměstnanců a vedení očních optik.

CÍLE PRÁCE

Hlavním cílem této práce je návrh a vytvoření zcela nového informačního systému, který bude směřovat svojí funkcionalitou zejména na oční optiky.

Vytvářený informační systém bude jednoduchý, intuitivní a zajistí veškerou funkcionalitu pro podporu práce zaměstnanců očních optik.

Požadovaná funkcionalita informačního systému bude analyzována ve spolupráci s majiteli a provozovateli očních optik, jímž dosavadní informační systém buď nevyhovuje či jej doposud nevyužívají. Tyto oční optiky se mohou stát budoucími uživateli vytvořeného systému.

Informační systém má být navržen obecně pro všechny menší a středně velké oční optiky, tudíž při návrhu funkcionality a designu se budu inspirovat i ze stávajících řešení.

Informační systém bude umožňovat propojení s dalšími CRM či ERP systémy pomocí můstků na úrovni Microsoft SQL databází.

1 TEORETICKÁ VÝCHODISKA PRÁCE

V této kapitole se budu zabývat teoretickými východisky, která mi pomohou realizovat téma diplomové práce. V první řadě je třeba vymezit základní informace o informačních systémech a jejich budování. Rovněž bude potřeba seznámit se s teorií databází a s jazykem SQL. V poslední řadě je třeba objasnit, co je to vývojové prostředí Delphi a jaké jsou jeho výhody pro tvorbu informačních systémů a komunikaci s databázemi.

1.1 Informační systémy

Na začátku úvodu do informačních systémů je třeba vymezit základní pojmy informace, data a znalosti, které tvoří jejich základ. Dále je třeba vymezit principy a metodiky budování informačních systémů s požadovanými vlastnostmi, které se od IS očekávají.

1.1.1 Informace, data a znalosti

Existuje vzájemná souvislost mezi daty, informacemi a znalostmi. Data představují objektivní fakta o událostech, jsou však bez významu a kontextu, proto člověku příliš mnoho neřeknou. Mají jedinečný význam (10).

K tomu, abychom pochopili význam dat, slouží informace. Informace jsou tvořeny surovými daty a jejich popiskem, který již udává jejich význam. Mají jistý způsob interpretace a dokáží sdělit člověku data v pochopitelné formě. Životní cyklus informace počíná v jejím získání, zpracování a uchování. Druhá fáze jejího životního cyklu se týká distribuce a zpřístupnění informace s tím, že v konečné fázi informace přichází její vyhodnocení, použití, rozšíření a aplikace (10).

Znalostí lze označit stav, kdy je informace pochopena, její zpracování probíhá v lidské mysli. Znalost je tvořena daty, jejich významem a pochopením těchto prvků v nějakém kontextu (10).

Obr. 1: Informace, data a znalosti (10)

1.1.2 Co je to informační systém

Obecně systém lze chápat jako uspořádanou množinu prvků a vazeb mezi nimi s určitou strukturou a účelovým chováním. Informační systém je pak definován jako soubor prostředků složený z hardware, software, lidí, organizačních opatření a jiné, s účelovou formou využití informačních technologií v sociálních a ekonomických systémech. Tyto prostředky jsou dále použity pro sběr, zpracování a poskytování informací (11, 12).

Informační systém pomáhá při rozhodování a řízení firemních aktivit a stává se tak nezbytnou součástí každé společnosti (11, 12).

1.1.3 Informační a komunikační technologie, software

Informační a komunikační technologie si lze představit jako hardwarové a softwarové prostředky, které zajišťují sběr dat a práci s nimi. Umožňují vzájemnou komunikaci lidí s technologiemi (11).

Softwarový produkt je tvořen pomocí programových jednotek, jako jsou moduly, objekty, komponenty a služby. Mezi těmito jednotkami existují vazby a spolu dohromady utváří informační systém pro automatizované a neautomatizované činnosti (11).

Při tvorbě softwarových produktů lze proces rozdělit do dvou hlavních oblastí. Proces vývoje software nejlépe vystihuje model „programuj a opravuj“ (11).

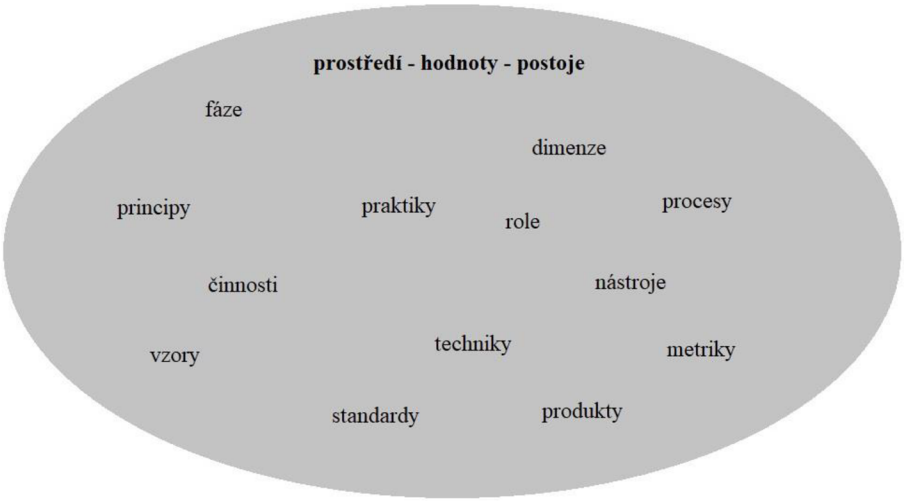
Obr. 2: Model „programuj a opravuj“ (11)

Na začátku vývoje je však třeba specifikovat požadavky, které jsou od softwarového produktu očekávány. Tyto požadavky jsou zpravidla konzultovány s klíčovou osobou a organizací, která tento produkt bude využívat. Cyklus opakování programování a opravy chyb pak tvoří základní část, na které je tento model založen. Každý programátor ví, že bez chyb se nikdy nic nevytvoří. Po úspěšném dokončení zdánlivě nekonečného cyklu přichází etapa implementace a předání softwarového produktu do reálného provozu (11).

1.1.4 Životní cyklus vývoje IS

Informační systém (IS) má svůj životní cyklus, který představuje časový úsek. Tento časový úsek je ohraničen na počátku tvorbou informačního systému a končí v momentě, kdy se informační systém přestane používat (11, 12).

Při budování informačních systémů je tento proces rozdělen do dvou hlavních oblastí vývoje a provozu. Cílem vybudování informačního systému je propojení podnikových procesů s procesy informatiky. Používají se různé metodiky, které však mívají zpravidla stejné, či velmi podobné prvky, u kterých se velmi těžko určuje, které jsou povinné, a které nejsou (11, 12).



Obr. 3: Prvky metodiky budování IS (11)

Existuje velké množství metodik budování IS, které popisují metody a postupy pro realizaci určitého softwarového projektu. V rámci těchto metodik se definují principy, procesy, praktiky, role, techniky, nástroje a produkty, jež se používají při vývoji, údržbě a provozu IS (11, 12).

1.1.5 Požadavky na informační systém

Podnikový informační systém (IS) lze chápat jako podpůrný nástroj při řízení podniku. Mezi hlavní požadavky patří:

- podpora automatizace každodenní rutiny,
- přehled a dostupnost informací podporujících rozhodování,
- jednotná verze pravdy přítomná ve všech výstupech systému,
- zajištění efektivního fungování podniku (13).

Na základě těchto požadavků se předpokládá, že IS zajistí podpůrné procesy ekonomiky a lidských zdrojů a podpoří manažerská rozhodnutí zejména statickými reportingy (13).

Takové formy IS však nepřináší nejvýhodnější poměr cena/kvalita/přidaná hodnota, za to však podpoří výkonnost a konkurenceschopnost podniku. Kvalita IS je definována mnoha parametry, mezi něž patří funkčnost, spolehlivost, udržitelnost, komfort pro uživatele,

dodatečná modifikace se schopností dalšího rozšíření. Neméně důležitou vlastností IS je jeho zabezpečení (13).

Poměr cena/kvalita/přidaná hodnota IS je podporovaná oblastmi, které nemusí přímo souviset s jeho vlastnostmi, ale zohlední navíc:

- změny vzniklé v organizační struktuře společnosti a jejím řízení,
- sdílení nejlepších praktik s experty v oboru,
- sjednocení a standardizaci podnikových procesů a návyků,
- celistvý pohled na fungování podniku,
- podpora manažerského rozhodování i ve strategické úrovni,
- růst výkonnosti a konkurenceschopnosti podniku či jednotlivých součástí organizace (13).

Pro úspěšnou integraci systému k podnikovým procesům je třeba znát vztahy ke strategickým cílům a hlavním činnostem podniku. Dále je třeba znát vlastnosti dodavatelů a zákazníků se vztahy k dané organizaci (13).

Na základě těchto zásad je možné řešit tyto dílčí části budování IS:

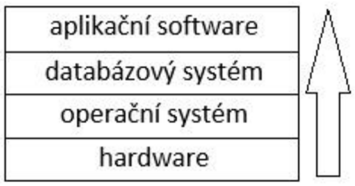
- analýza požadavků a stávajícího stavu organizace,
- určit problémy k řešení,
- určit cíle, kterých má být dosaženo s naplánováním realizace tvorby IS,
- analýza využití pokročilých metod pro návrh a tvorbu informačního systému,
- návrh nové koncepce řízení IS/ICT v návaznosti na hodnocení stávající koncepce (13).

1.1.6 Základní vrstvy IS

Důležitou součástí architektury informačního systému je jeho datový pohled. Ke sjednocení dat a vzdálenému přístupu k nim pomohly relační databáze. Tímto krokem se snížily náklady na materiálové zásoby a zkrátily časy potřebné k dokončení většiny firemních procesů (14).

Dnes je běžné pro většinu IS použití relační databáze s vhodným operačním systémem a databázovým systémem podporujícím nástroje dotazovacího jazyka SQL. Informační systémy se pak rozpadají do několika technologických prvků, které lze znázornit formou vrstev, které na sebe navazují (14).

Základ všeho tvoří hardware, který však nebude sám o sobě fungovat bez optimálního informačního systému. Spolu tak tvoří základnu IS. Pro sběr a sjednocení všech podnikových dat slouží databázový systém, který tvoří další vrstvu konceptu. Poslední vrstvou je aplikační software, jež zastřešuje aplikační řešení, včetně uživatelského rozhraní fungujícím na databázovém prostředí. K této vrstvě patří i dokumentace vytvořená k danému řešení (14).



Obr. 4: technologický model IS (14)

1.2 Databázový systém

Databázový systém je software se zaměřením na práci s databázemi. Zprostředkovává zpracování dat jako ukládání, modifikaci či mazání dat a jejich zobrazení. Tvoří rozhraní pro komunikaci mezi aplikační vrstvou, jež tvoří programy a prostředí komunikující přímo s uživateli a daty uloženými v databázi (1,2).

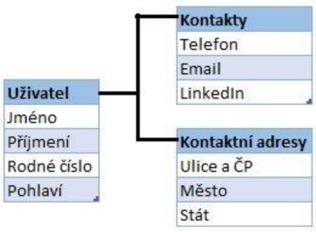
Databázový systém tvoří čtyři základní komponenty: uživatelé, databázová aplikace, systém řízení databáze a samotná databáze. Databáze a systém řízení databáze jsou uloženy na specializovaných počítačích nazvaných jako databázové servery (2).

1.2.1 Databáze

Pojem databáze je možné definovat jako soubor dat a informací uložených na datovém uložišti. V širším pojetí databáze se jedná o spojení pojmů **vlastní popis** a **související tabulky** (2).

Pojmem vlastní popis se myslí popis struktury databáze. Jedná se o informace o datech, které jsou ukládány do strukturovaných tabulek v databázi. Popis struktury je rovněž uložen v databázi v tzv. **metadatech** (2).

Datovou strukturu databáze tvoří strukturované tabulky, ve kterých jsou uložena data na základě popisu tabulky. Z důvodů konzistentnosti, přehlednosti a snadné manipulovatelnosti jsou data týkající se jednoho případu rozdělena do několika tabulek dle tématu, kterého se data týkají. V praxi lze rozdělit data o jednom člověku do několika tabulek, které se týkají základních informací, kontaktů a bydliště, kterých může mít více. Tabulek, které popisují jedinou osobu, je více a jedná se tedy o tzv. **související tabulky** (2).



Obr. 5: Související tabulky (vlastní zpracování, dle 2)

Metadata obsahují informace o názvech tabulek a jejich sloupců, jejich vlastností, datových typů atd. Zjednodušeně lze využít tyto metadata jako technickou dokumentaci k databázi. Pomocí metadat lze v databázových systémech nechat sestavit a vymodelovat strukturu celé databáze, aniž by ji kdokoli musel znát (2, 3).

Samotná databáze je složena z:

- uživatelských dat,
- metadat,
- indexů (odkazů) a dalších režijních informací,

- aplikačních metadat (není vždy podmínkou) (2).

1.2.2 Systém řízení databáze

Systém řízení databáze neboli DBMS (Database management system) je postavený nad samotnou databází. Zatímco databáze je pouze soubor či několik souborů s uloženými informacemi, DBMS je prostředek, který s těmito soubory pracuje. Jeho hlavním úkolem je efektivní práce s daty počínaje tvorbou, zpracováním a spravováním jedné či více databází. Mezi systémy řízení databáze patří Microsoft SQL Server (2).

Základní funkce, které by měl databázový systém obsahovat:

- tvorba databáze a tabulek,
- tvorba indexů a jiných podpůrných struktur,
- čtení a transformace velkých množství dat,
- úprava, vkládání nebo odstranění dat v databázi,
- správa integrity údržba databáze,
- zajištění bezpečnosti a kontrola přístupu,
- programovací prostředí pro jazyk SQL,
- řízení souběžnosti činnosti vícero uživatelů,
- záloha a obnovení databáze (2, 4).

Aplikace třetích stran, které využívají uživatelé, předávají informace databázovému systému, jak mají být data zpracována formou příkazů v jazyce SQL. Systém pro řízení databází pak tyto požadavky zpracovává a provádí konkrétní operace s daty (2).

1.2.3 Databázové aplikace

Databázová aplikace není nic jiného než program, který běžně používá uživatel při komunikaci s databází. Aplikační program má uživatelské rozhraní, které usnadňuje práci s daty v databázi. Obsahuje formuláře, prostředí pro vizualizaci dat či logiku pro

rozhodování při práci s daty. V pozadí této aplikace probíhá komunikace na základě SQL dotazů s databázovým systémem (2).

Typické funkce databázových aplikací:

- tvorba a zpracování uživatelských formulářů bez ohledu na přesnou strukturu tabulek,
- zpracování dotazů a komunikace s databázovým systémem,
- řízení aplikace,
- vytváření a zpracování sestav kombinací různých informací (2).

1.3 Relace

Nejběžněji využívané databáze jsou relační databáze. Relační databáze je založená na relacích, jež představují dvourozměrné tabulky. Relace neboli tabulky, se skládají ze sloupců a řádků se specifickými vlastnostmi:

- každý řádek je tvořen daty o určité entitě,
- každý sloupce představuje jeden druh informace entity – atribut,
- řádky a sloupce tvoří buňky, které musí obsahovat vždy pouze jedinou informaci ve vztahu ke sloupci,
- sloupce musí mít unikátní název, přičemž na jejich pořadí nezáleží,
- každý řádek musí obsahovat unikátní sadu dat, tj. nesmí existovat dva naprosto shodné řádky,
- na pořadí řádků nezáleží (2).

Každá tabulka obsahuje pouze specifická data. Při vytváření databáze se rozdělují informace na témata, podle kterých jsou dále vytvořeny tabulky (2).

1.3.1 Datové typy atributů relací

Jak již bylo zmíněno, tabulky (relace) jsou složeny z řádků a sloupců. Sloupce představují atributy tabulky, které mají definovaný datový typ. Datový typ určuje, v jakém formátu jsou data v daném sloupci uložena (2, 7).

Existuje spousta druhů datových typů pro ukládání řetězce textu, celých čísel, čísel s desetinou tečkou, binárních souborů, či datový typ s hodnotou datum nebo hodnotami pravda a nepravda (2, 7).

Mezi nejpoužívanější datové typy pro ukládání řetězce textu a souborů patří:

- **CHAR** – má přesně stanovenou délku řetězce, která je do něj vložená,
- **VARCHAR** – délka řetězce je dynamicky nastavitelná, definuje se pouze maximální délka,
- **BINARY** a **VARBINARY** – podobný jako (VAR)CHAR, používá se pro ukládání souborů,
- **BLOB** – podobný jako (VAR)BINARY, používá se pro ukládání velkých souborů,
- **TEXT** – pro ukládání velkých řetězců textu, podobný jako VARCHAR (7).

K datovým typům TEXT a BLOB existují variace TINYTEXT, MEDIUMTEXT, LONGTEXT, TINYBLOB, MEDIUMBLOB, LONGBLOB, které fungují stejně, nicméně jsou omezeny jejich maximální délky daným datovým typem (7).

K nejpoužívanějším datovým typům pro ukládání číselných hodnot patří:

- **BIT** – dvouhodnotový datový typ plněný číslem 1 nebo 0,
- **INT** – slouží k ukládání čísel v celočíselném tvaru,
- **NUMERIC** – podporuje ukládání čísel s desetinným místem,
- **MONEY** – podobný jako NUMERIC, slouží k ukládání finančních hodnot, při matematických operacích je přesnější (7).

Datové typy INT a MONEY mají variace TINYINT, SMALLINT, BIGINT, SMALLMONEY, které fungují stejně jako jejich nadřazený datový typ, mají však omezené maximální délky datovým typem (7).

Nejpoužívanější datové typy pro ukládání data a času jsou:

- **DATE** – umožňuje uložit pouze datum v korektním formátu,
- **TIME** – umožňuje uložit pouze čas v korektním formátu,
- **DATETIME** – kombinace dvou výše zmíněných datových typů (7).

1.3.2 Hodnota NULL

Ve sloupcích tabulek (relací) jsou ukládány informace, zadaných na vstupu. Při vkládání nových řádků bychom měli nadefinovat hodnoty pro všechny sloupce tabulky. To však není vždy pravidlem, neboť tabulky relací umožňují vyplnění pouze těch sloupců, které nás u daného záznamu zajímají. Atributy, kterým nebyla při vstupu přidělena žádná hodnota, zůstanou prázdné a v databázi je tento stav definován hodnotou **NULL** (2, 7).

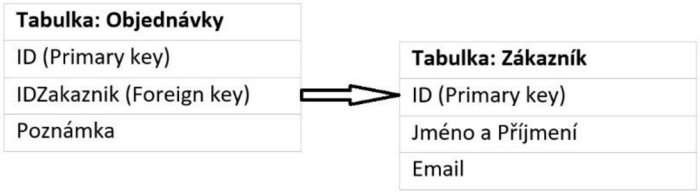
Tento stav nepředstavuje žádnou hodnotu a lze ji vystihnout slovem nic. Tato hodnota ve sloupci nezabírá žádné místo v databázi (2, 7).

V případě, že je hodnota NULL nežádoucí v určitém atributu, je možné nadefinovat atribut tak, aby databázový server donutil uživatele při vkládání dat atribut naplnit (**NOTNULL**). Druhou možností je nastavení na daném atributu defaultní hodnotu, kterou má být vyplněn v případě, že není plněn žádnou hodnotou na vstupu (2, 7).

1.3.3 Klíče relací

Každá tabulka (relace) by měla obsahovat klíč, který identifikuje konkrétní řádek. Jak je již z popisku zřejmé, tak klíč musí být pro každý řádek unikátní. Nesmí vzniknout stav, kdy zvolený klíč tabulky měl ve dvou řádcích stejné hodnoty. Tento klíč se nazývá „Primary key“ neboli **primární klíč tabulky**. Ideální primární klíč je krátký a složen pouze z čísel a jeho hodnota ke konkrétnímu řádku zůstává stejná, tj. neměnný (5).

Mimo primární klíč existuje tzv. „Foreign key“ neboli **cizí klíč**. Cizí klíč slouží k propojení dvou tabulek. Jedná se o jeden či více sloupců v jedné tabulce, které tvoří odkaz na primární klíč cizí tabulky a tím tvoří jednoznačnou vazbu mezi nimi. Hodnota cizího klíče musí odpovídat hodnotě primárního klíče v cizí tabulce, nelze tedy vkládat do atributu sloužící jako cizí klíč hodnoty, jež nemají odpovídající hodnotu na protistraně. Tomuto omezení se říká omezení referenční integrity, které nedovolí vložit neznámou hodnotu do pole cizího klíče (2, 5).



Obr. 6: Propojení tabulek pomocí klíčů (vlastní zpracování, dle 2)

Klíče mohou být složeny z jednoho či více sloupců (atributů). Ne vždy tabulka obsahuje sloupec, který v sobě nese unikátní hodnoty, pak je třeba vytvořit unikátní odkaz spojením několika atributů do jediného **složeného klíče**. Atribut či kombinace atributů identifikující řádky se nazývá **kandidátní klíč** (2).

1.3.4 Omezení relací

Při vytváření tabulek (relací) se volí název tabulky, její atributy a k nim přiřazené datové typy a omezení (CHECK). V předešlých podkapitolách bylo zmíněno, že lze omezit vkládání hodnoty NULL do atributů v případě jejich nezadání na vstupu a také definování klíčů tabulky (2, 6).

Při definování tabulky, ať jejím založením či dodatečné úpravy lze přidávat i další omezení **UNIQUE** a **CHECK**. Obě zmíněné omezení lze aplikovat na jednotlivé atributy (6).

Klauzule **CHECK** definuje uživatelsky nastavitelné omezení daného atributu, například pokud bychom chtěli, aby hodnota do číselného sloupce nebyla uložena menší než 0 či

jiná hodnota. Dále lze omezit textové sloupce tak, aby byl vždy vyplněn text se shodným počtem znaků, či text začínal na specifický řetězec písmen (6).

Klauzule **UNIQUE** definuje unikátnost hodnot v daném sloupci. To znamená, že do vybraného sloupce nelze vložit dvě stejné hodnoty, a tím pak vzniká sloupec kandidátním klíčem (6).

1.3.5 Indexy relací

Index je objekt, který je pevně spojen s tabulkou. Slouží k optimalizaci a zrychlení vyhledávání a dotazů prováděných nad tabulkou, ale také pro zajištění integrity dat (unikátnost, odkazy). Lze jej chápat jako rejstřík na konci knihy, pomocí kterého lze v knize rychleji vyhledávat (2, 6).

Index se definuje nad tabulkou a jejím konkrétním sloupcem či několika sloupci. V databázi je pak vytvořena struktura tohoto indexu, která se průběžně aktualizuje, což může zpomalovat procesy, které přidávají či upravují data. Použitím indexů je kladen větší nárok na kapacitu uložišť a operačních pamětí, neboť jsou hodnoty indexovaného sloupce uloženy v databázi duplicitně (2, 6).

1.4 Jazyk SQL

Jazyk SQL začala vyvíjet společnost IBM na konci osmdesátých let 20 století. Zkratka SQL představuje název strukturovaný dotazovací jazyk, z čehož je zřejmé, že se jedná o textově orientovaný jazyk, který lze psát v pouhém textovém editoru. V předešlých podkapitolách byla uvedena problematika databází, které slouží zejména pro uchování a distribuci dat. Pro přístup k těmto datům, jejich transformacím a jiným operacím s daty byl vytvořen právě jazyk SQL, pomocí kterého komunikuje systém řízení databáze se samotnou databází (6).

Jazyk SQL je speciální datový jazyk, který lze používat pomocí příkazů, rozdělených do kategorií. Nejtypičtější, a také nejčastěji používané jsou příkazy kategorie DDL (data definition language) a příkazy kategorie DML (data manipulation language) (2, 6).

1.4.1 Příkazy DDL jazyka SQL

Příkazy DDL jazyka SQL slouží k tvorbě či úpravě objektů databáze. Strukturu databáze definují tabulky (relace), podmíněné spouště (trigger), definované procedury a funkce, pohledy či indexy (2).

Mezi DDL příkazy patří:

- CREATE,
- ALTER,
- DROP,
- TRUNCATE TABLE (2).

Příkaz **CREATE** slouží pro vytvoření nového objektu. V závislosti na tom, jaký objekt vytváříme, se mění syntaxe tohoto příkazu. Příkaz **ALTER** slouží k modifikaci již existujícího objektu a mění se syntaxe příkazu v závislosti na typu modifikovaného objektu. Příkaz **DROP** má jednotnou syntaxi a slouží k odstranění objektu (2, 6).

Pomocí příkazu ALTER TABLE, lze upravovat strukturu tabulky či přidávat omezení a indexy (2, 6).

Příkazy lze použít i na vytvoření, modifikaci či odstranění celé databáze (2, 6).

1.4.2 Příkazy DML jazyka SQL

Jsou to příkazy, které umožňují odesílat databázové dotazy a provádět manipulace s daty či je upravovat. Existují tři operace, které můžeme s daty provádět. Vložení dat, jejich úprava (aktualizace) či jejich odstranění (2).

Mezi běžné DML příkazy patří:

- INSERT,
- UPDATE,
- DELETE,
- SELECT,
- EXPLAIN,
- SHOW,
- TRUNCATE (2, 6).

Příkaz **INSERT** slouží pro vkládání nových dat do tabulky (relace). Příkaz **UPDATE** umožní již existující data upravit a příkaz **DELETE** pak data smaže. Příkaz **SELECT** umožňuje data z tabulek či přehledů načíst a zobrazit. Jedná se tedy o příkazy zaměřené na práci s daty v tabulkách (2, 6).

Syntaxe příkazů INSERT, UPDATE, DELETE a SELECT je velmi podobná. Vždy se zapisuje jeden z těchto příkazů, vybere se jedna či více tabulek, atributy a na závěr je potřeba zvolit filtraci záznamů pomocí klauzule **WHERE** (6).

Doplňující příkaz **EXPLAIN**, který je používán v souvislosti s příkazem SELECT, vypisuje postup zpracování dat tímto příkazem. Pomáhá pak programátorům pochopit odezvu příkazu SELECT, pro jeho případnou optimalizaci. Poslední příkaz **SHOW** zobrazí databáze či jejich struktury a definice (2, 6).

Příkaz **TRUNCATE TABLE** s názvem tabulky slouží pro kompletní vymazání všech dat z tabulky, bez porušení její struktury (6).

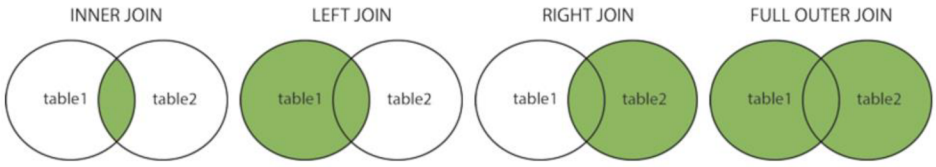
1.4.3 Propojení více tabulek pomocí JOIN

V podkapitole o relacích bylo zmíněno, že lze definovat u tabulek primární klíče a cizí klíče, které definují vazbu na primární klíč jiné tabulky. V dotazovacích příkazech lze těchto propojení využít a dotazovat se na hodnoty z vícero tabulek zároveň. Pro tyto dotazy je důležité správně nadefinovat vazby mezi tabulkami, a k tomu je potřeba znát strukturu databáze (2).

Běžný dotazovací příkaz SELECT hledá záznamy v tabulce nadefinované za klauzulí FROM. Nadefinovat lze však více tabulek, které lze na tuto hlavní tabulku či vrstveně na sebe navazovat. K navázání další tabulky slouží klauzule JOIN s tabulkou, kterou chceme navázat a podmínkou, kterou se vázaná tabulka s nosnou spojí (2, 6).

Existují tyto čtyři variace klauzule JOIN:

- INNER JOIN (JOIN) – vrátí záznamy se shodou v obou vázaných tabulkách,
- LEFT OUTER JOIN – vrátí záznamy z nosné (levé) tabulky a k nim přiřazené záznamy z pravé tabulky, pokud existují,
- RIGHT OUTER JOIN – vrátí záznamy z vázané (pravé) tabulky a k nim odpovídající záznamy z levé tabulky, pokud existují,
- FULL OUTER JOIN – vrátí záznamy se shodou v alespoň jedné tabulce (6).



Obr. 7: Propojení relací pomocí JOIN (6)

Propojením více tabulek (relací) vzniká zcela nová relace, jejíž struktura odpovídá zobrazovaným atributům z dotazu (2).

1.4.4 Integrované funkce jazyka SQL

V jazyce SQL se vyskytuje řada integrovaných funkcí, které umožňují početní či agregační operace s daty. Použití těchto funkcí je limitováno datovými typy sloupců, na které jsou použity (2).

Mezi algebraické a distributivní integrované funkce patří:

- COUNT – vrátí počet řádků odpovídající vybraným kritériím,
- SUM – sečte hodnoty ve vybraným sloupci,
- AVG – spočte průměrnou hodnotu vybraného sloupce,
- MAX – najde nejvyšší hodnotu z vybraného sloupce,

- MIN – najde nejnižší hodnotu z vybraného sloupce (2, 6).

Dále existuje agregační funkce **GROUP BY**, která seskupuje řádky podle společných hodnot z vybraného sloupce. S touto funkcí jsou nejčastěji používané výše zmíněné početní funkce (2, 6).

K omezení zobrazených výsledků s funkcí GROUP BY se používá funkce **HAVING**, která filtruje na základě zvolené podmínky zobrazované řádky (2, 6).

1.4.5 Transakce SQL

Databázové transakce obsahují množinu příkazů, které byly provedeny od jejího spuštění pomocí příkazů **BEGIN TRAN**. Pokud jsou nad databází prováděny změny typu INSERT, UPDATE či DELETE a souběžně s nimi běžela transakce, je možné všechny tyto změny vrátit do původního stavu příkazem **ROLLBACK TRAN**, či je potvrdit a zároveň ukončit příkazem **COMMIT TRAN** (2, 7).

Použití transakcí je vhodné tehdy, když provádíme změny v datech na několika různých místech s tím, že pokud nastane neočekávaný stav nebo vedou změny k nechtěnému výsledku, pak je možné všechny tyto změny navrátit do původních hodnot (2, 7).

Transakce se mohou hodit i pro ošetření skriptu, u kterého si jeho tvůrce není jistý, zda proběhne korektně. Každou započatou transakci je důležité řádně ukončit, neboť v rámci daného SQL připojení jsou všechny příkazy provedené zaznamenány v běžící transakci. Pokud nastane kritická chyba kdykoli během otevřené transakce, pak jsou veškeré provedené změny vráceny do původních hodnot a mohou tak být ztraceny i změny, které byly správné. V transakci pak může teoreticky běžet i celá databáze po celou dobu její existence (2, 7).

1.5 Programovatelné objekty SQL databázového systému

Databázové systémy SQL jako jsou Microsoft SQL Server, obsahují programovatelné objekty, pomocí kterých mohou programátoři vytvářet moduly zpracovávající logické a

databázové akce. Patří mezi ně uložené procedury a funkce, spouště (trigger) a pohledy (view) (2, 7).

1.5.1 Spoušť – trigger

Trigger si lze představit jako spoušť, která reaguje na určité události, které v databázi nastanou. Existují tři typy spouští, které dokážou reagovat na zcela jiné události. Patří mezi ně:

- DML Trigger – reaguje na události INSERT, UPDATE a DELETE,
- DDL – reaguje na události CREATE, ALTER a DROP,
- LOGON trigger – reaguje na nově vzniklé uživatelské připojení (2, 7).

DML Trigger je postaven nad jedinou tabulkou a je spouštěn na základě určité události s daty v tabulce. Trigger dokáže reagovat na tři události INSERT, UPDATE a DELETE. Tyto tři stavy představují vložení záznamů do tabulky, změnu záznamů v tabulce, či mazání dat v tabulce (7).

V těle spouště jsou definovány příkazy, které mají být provedeny v případě, že některá z těchto událostí nastane. Dále lze definovat moment, ve kterém má spoušť reagovat přidáním klauzulí BEFORE, AFTER nebo INSTEAD OF. V případě prostředí Microsoft SQL Server jsou spouště následovány dvěma tabulkami „inserted“ a „deleted“. Jejich struktura odpovídá struktuře nosné tabulky (2, 7).

Do tabulky „inserted“ se vloží jeden, či vícero záznamů, které odpovídají vkládaným datům do tabulky nosné, a to i v případě příkazu UPDATE, tedy aktualizaci dat. Do tabulky „deleted“ se vkládá jeden, či vícero záznamů, které odpovídají záznamům, jež byly z nosné tabulky smazány. Stejně tak se zde ukládají původní záznamy s původními hodnotami v případě aktualizace dat příkazem UPDATE (7).

Spouště pak dokážou zcela automaticky reagovat na změny dat v tabulkách, provádět dodatečné kroky, které je třeba s vložením nových záznamů podniknout, či zamezit uložení dat na základě stanovených podmínek (2, 7).

1.5.2 Uložené procedury – stored procedure

Uložené procedury představují databázové objekty, které neobsahují žádná data, ale plní funkci podprogramu. Procedury jsou umístěné v databázi a v jejich těle je podobně jako u spouští uložena řada příkazů, které se mají při jejím spuštění (zavolání) provést. Takovéto procedury je potřeba spouštět ručně, lze je volat i v těle spouště či jiných procedurách (2, 7).

Uložené procedury je možné definovat se vstupními parametry, díky kterým jim lze předat potřebná data k provedení příkazů, které jsou v nich uloženy (2, 7).

1.5.3 Definované funkce – function

Mezi další databázové objekty patří funkce. Databázové systémy obsahují standardně mnoho různých funkcí, Microsoft SQL Server však umožňuje vytvořit nové, programátorem definované funkce. Funkce se od procedury moc neliší, má však své výhody. Funkce je zpravidla soubor instrukcí, které zpracují informace poskytnuté na vstupu a výstupem je vrátí (2, 7).

Procedury lze volat pouze příkazem **EXEC** (execute) a lze je tedy použít jen omezeně. Definované funkce lze vyvolat uvnitř příkazu SELECT a tím za běhu měnit zobrazovaný výsledek (2, 7).

1.5.4 Pohledy – view

Definované pohledy jsou databázové objekty, které poskytují data podobně jako tabulky. V přehledu data uložena nejsou, pohled pouze definuje způsob, jak získat požadovaná data. V pohledech lze zobrazit agregovaná data z několika datových zdrojů (tabulek), ale také upravené funkcemi či výpočty. Pohled je definován funkcí SELECT (2, 7).

Hlavní výhodou pohledu je schopnost zobrazit data kombinovaná z několika tabulek, a přitom nezabírá téměř žádné místo v databázi. V případě, že se změní data v tabulce, změní se také zobrazované informace v přehledu, které z dané tabulky čerpá (2, 7).

1.6 Entity-relationship model

Datový E-R model definuje entity a vztahy, jak je z jeho názvu zřejmé. Datový model ER je nejoblíbenější technika pro návrh kvalitní databáze. V ER diagramech jsou uvedeny jednotlivé entity a vztahy mezi nimi (2, 9).

ER diagramy jsou vytvářeny zejména při návrhu databáze. Když programátor vytváří tabulky a vazby mezi nimi, ER diagram mu umožní pochopit strukturu, kterou má vytvořit. ER diagramy lze však využít i při ladění či opravě databáze. Při návrhu ER diagramů je potřeba popsat entity jejími atributy, které pomáhají pochopit vlastnosti dané entity (2, 9).

Entita je objekt, který chceme sledovat, jsou to například zákazníci, objednávky, zboží a jiné. Entita představuje v databázi tabulku s atributy (2, 9).

Vztah pak představuje, jakým způsobem jsou dvě entity spolu spojeny. V databázi vztah představuje vazbu mezi dvěma tabulkami pomocí klíčů (2, 9).

1.6.1 Kardinality

Kardinalita definuje počet výskytů jedné entity v entitě druhé, jako příklad poslouží zákazník a objednávky. Jeden zákazník může mít mnoho objednávek (2, 9).

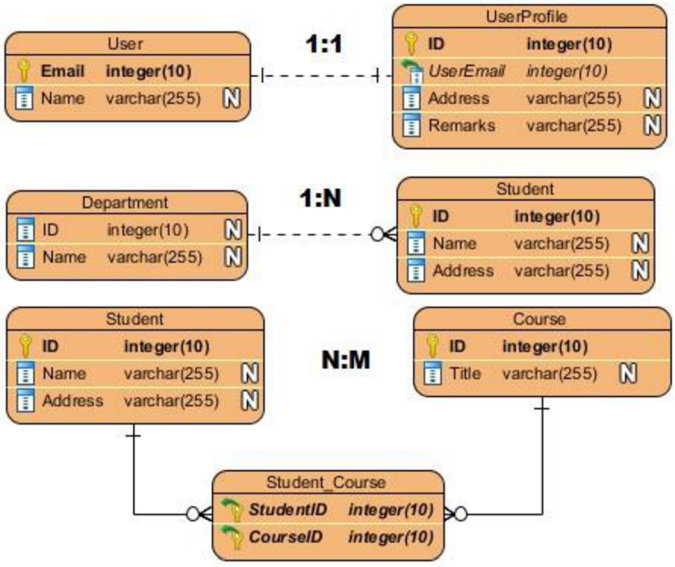
Existují tři typy vztahů mezi entitami:

- kardinalita 1:1,
- kardinalita 1: N / N:1,
- kardinalita N:M (2, 9).

Vztah 1:1 představuje stav, kdy jedna instance první entity je vzájemně propojena přesně s jednou instancí entity druhé. Například jeden uživatel má právě jeden uživatelský profil (2, 9).

Ve vztahu 1: N (N:1) jedna instance první entity může mít N (několik) instancí druhé entity a naopak. Příkladem je objednávka, která může mít navázaných několik položek (2, 9).

Vztah N:M přestavuje stav, kdy několik instancí první entity může mít několik instancí entity druhé. V praxi to pak znamená, že mezi těmito dvěma entitami je vytvořena další entita, která slouží pouze jako vazební. Příkladem jsou studenti a kurzy, na které se mohou přihlásit. Student může být přihlášený na několika kurzech a zároveň na kurzu může být přihlášeno několik studentů (2, 9).



Obr. 8: Typy vztahů mezi entitami (9)

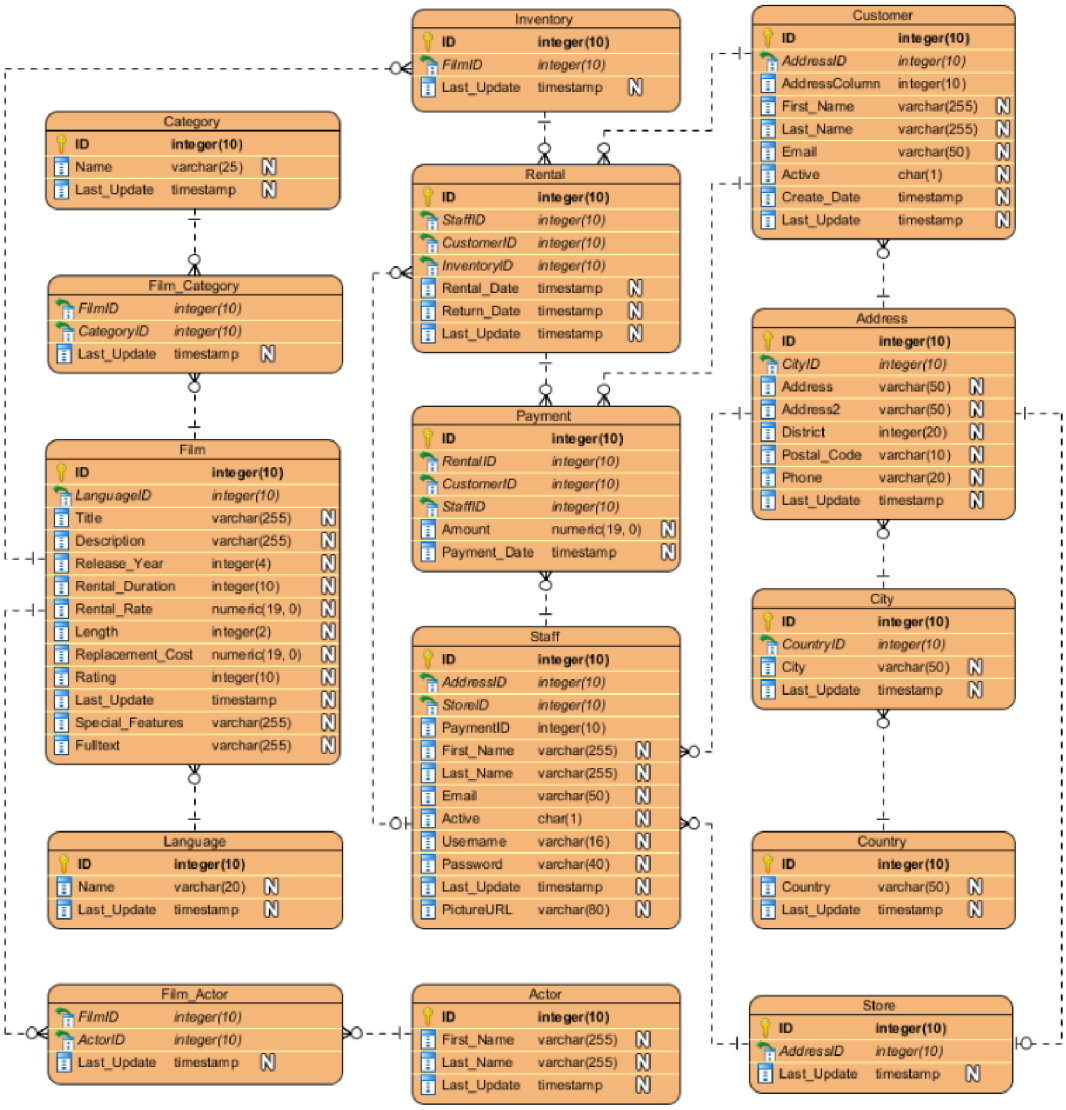
1.6.2 Fyzický datový model

Existují tři způsoby kreslení datových modelů: koncepční, logický a fyzický. Liší se od sebe v informacích, které jsou v nich uvedeny, a tím i cílová skupina lidí, kterým jsou prezentovány (2, 9).

Tab. 1: Vlastnosti ER diagramů (9)

Vlastnosti ER diagramu	Koncepční	Logický	Fyzický
Subjekt	Ano	Ano	Ano
Vztah	Ano	Ano	Ano
Sloupce (atributy)		Ano	Ano
Datové typy sloupců		Volitelné	Ano
Primární klíče			Ano
Cizí klíče			Ano

Pro vývojáře je nejvhodnější fyzický datový model, neboť je ze tří zmíněných komplexní a obsahuje většinu informací potřebných k vytvoření relační databáze. Jediné, co zde chybí, jsou informace o omezeních (CHECK) a defaultních hodnotách sloupců (2, 9).



Obr. 9: Příklad ER diagramu (9)

1.7 Vývojové prostředí Delphi

Delphi je grafické vývojové prostředí jazyka Pascal, respektive jeho nadstavby Object Pascal. Delphi je produktem firmy Borland se zaměřením pro vývoj na platformě Microsoft Windows. Programování v Delphi je pohodlné a rychlé, zejména díky uživatelskému prostředí, které umožňuje návrh grafického rozhraní s komponentami, přičemž automaticky generuje kostru zdrojového kódu (15, 17).

Knihovny komponent VCL, CLX, Indy a FMX jsou největší předností Delphi, a díky tomu se stále vyjímá nad některými konkurenčními produkty. K další přednosti Delphi patří možnost vytváření vlastních komponent (15, 17).

Pro Delphi je charakteristické zejména to, že je založené na programovacím jazyce Pascal obohaceném o objekty. Umožňuje vytvářet vlastní komponenty, či stáhnout již existující z webu a pracovat s nimi (15, 17).

Výhody Delphi:

- podpora systému RAD (vývojový nástroj pro Object Pascal, C++, ...),
- vyšší programovací jazyk,
- jednoduché propojení s databázemi,
- možnost kompilace do jediného spustitelného souboru,
- knihovny VCL, CLX, Indy, FMS, a jiné,
- kompatibilita vývojového kódu napříč verzemi Delphi,
- objektově orientovaný programovací jazyk,
- možnost tvorby multiplatformních aplikací (15, 17).

Naopak k **nevýhodám** Delphi a programovacího jazyka Object Pascal patří:

- špatná optimalizace výsledného kódu,
- špatná podpora 3D aplikací (15, 17).

1.7.1 Programování v Delphi

Vývojářské prostředí Delphi při založení nového projektu vytvoří **prázdný formulář**, na který je možné vkládat požadované **komponenty** a tím vytvářet vizuální stránku vyvíjené aplikace. Po přidání komponenty na formulář, vývojové prostředí automaticky vytvoří zdrojový kód k této komponentě (15).

Jednotlivé komponenty mají mnoho vlastností, které jsou v prostředí velmi jednoduše měnitelné. Patří mezi ně například název, velikost, barva, systémové jméno, viditelnost a další vlastnosti v závislosti na tom, o jakou komponentu se jedná (15).

Komponenty a jiné objekty prostředí Delphi umožňují reagovat na stisknutí tlačítek, pohyb myši či jiné vzniklé události (15).

Ve vývojovém prostředí lze přepínat mezi vizuální stránkou formuláře a zdrojovým kódem. **Zdrojový kód** lze zařadit do jednotlivých tříd (15).

1.7.2 Základní prvky prostředí Delphi

Při spuštění prostředí Delphi se zobrazí **hlavní okno**, které obsahuje hlavní nabídku, panely nástrojů, lištu s komponentami rozdělených do skupin po záložkách (15).

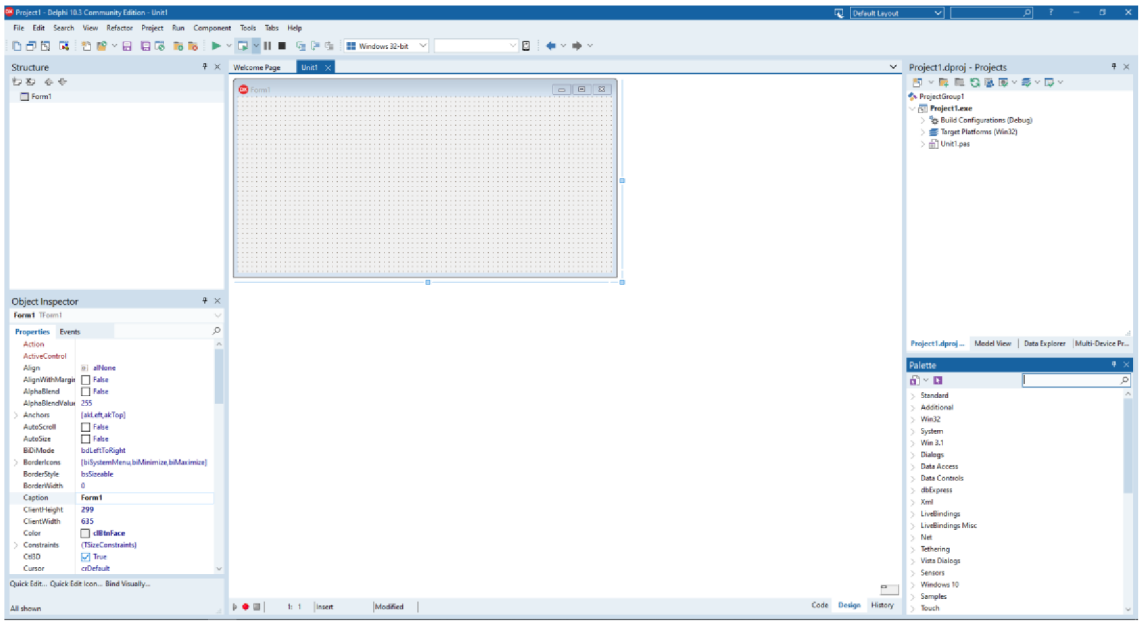
Při vyvíjení vizuální stránky aplikace se pracuje s formuláři, kterých může mít aplikace mnoho. Tyto formuláře se stávají součástí dané aplikace, a každý z nich má svoji vlastní **programovou jednotku** (unit) (15).

Pro zobrazení hierarchie komponent vybraného formuláře slouží objekt **TreeView**, ve které jsou graficky zobrazeny vztahy mezi komponentami **vlastníka** (owner) a současně **rodiče** (parent) (15).

Jednotlivé komponenty mají mnoho vlastností, které lze upravovat ručně ve zdrojovém kódu, či využitím **inspektora objektů** (Object Inspector), ve kterém jsou tyto vlastnosti zobrazeny. Inspektor objektů je rozdělen do dvou záložek, na záložce **Properties** se nachází vlastnosti dané komponenty a na druhé záložce **Events** jsou zobrazeny funkce, procedury a jiné reakce na vzniklé události, například kliknutí myší na komponentu (15).

Nejdůležitějším prvkem Delphi je **pak editor zdrojového kódu**, který slouží ke vkládání vlastního programového kódu. Editor lze vyvolat přepnutím z formulářového pohledu, tím je zajištěné propojení programové jednotky s vizuálním formulářem (15).

Při přepnutí do editoru zdrojového kódu se zobrazí **prohlížeč kódu** (Code Explorer), který shrnuje metody, deklarované proměnné a konstanty uvnitř zvolené komponenty. Slouží zejména k rychlému zobrazení proměnných, konstant a kritických či výstražných chyb ve zdrojovém textu (15).



Obr. 10: Vývojové prostředí Delphi 10.3.2 (vlastní zpracování, dle 17)

2 ANALÝZA PROBLÉMU A SOUČASNÉ SITUACE

Tato kapitola bude zaměřena na analýzu požadavků a očekávání od informačního systému pro podniky se zaměřením na Oční optiku. K tomu mi pomůže konzultace s vedením a zaměstnanci oční optiky, která doposud nevyužívá žádného informačního systému. S její pomocí analyzuji požadovanou funkcionalitu. Struktura databáze bude navržena dle analyzovaných funkcí a také procesů, které v pobočce běžně probíhají.

V poslední části kapitoly se budu zabývat analýzou s hodnocením stávajících informačních systémů zaměřených na zmíněný obor podnikání. Soustředím se i na výběr vývojových prostředí, které pro tvorbu informačního systému využiji.

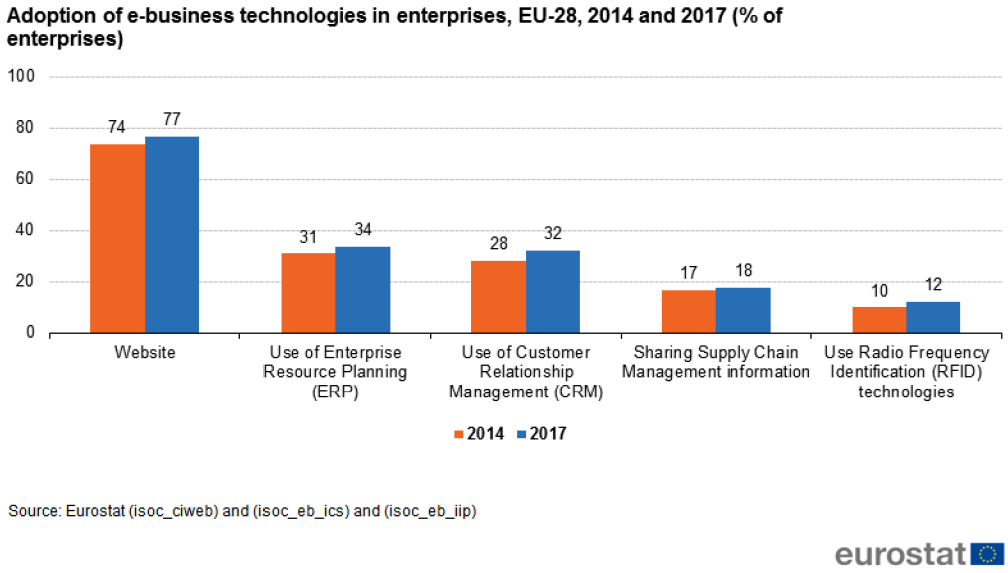
Výsledek analýzy bude sloužit jako důležitý podklad pro tvorbu vlastního řešení.

2.1 Využití informačních systémů v podnicích

Informační systémy, dále jen IS, se stávají nedílnou součástí úspěšných podniků. Informační systém funguje bok po boku s každodenní práci zaměstnanců, managerů či administrátorů, představuje vše, čemu se říká digitální podnikání – e-economy (18).

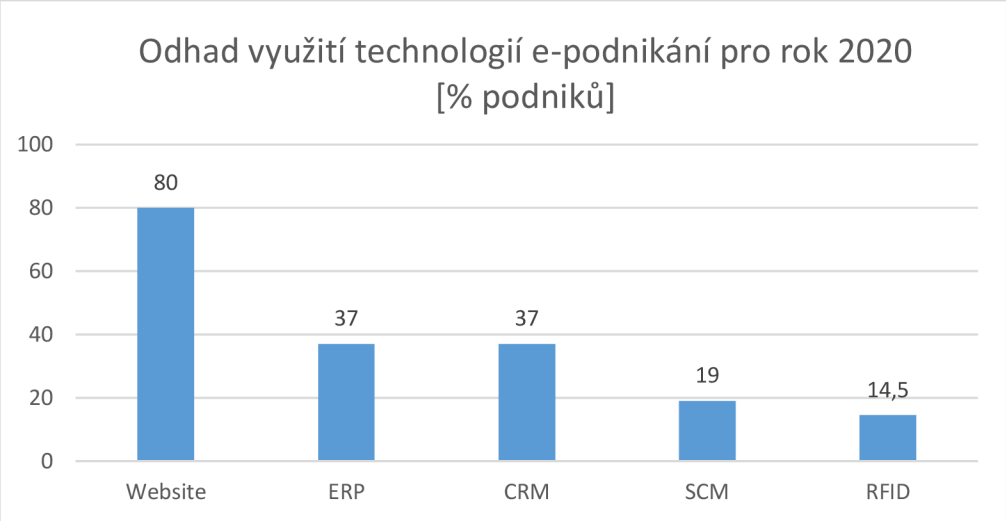
Dle statistik získaných z Eurostatu je stále vysoké procento malých a středních podniků v Evropské unii, které nevyužívají žádných technologií e-podnikání, jež zahrnuje různé formy IS. Největší rozdíl v zastoupení těchto technologií se vyskytuje mezi skupinou malých a středních podniků a skupinou velkých podniků. Ve velkých podnicích je využití mnohem vyšší, neboť se pro ně stávají IS nezbytnou součástí (18).

Procento zastoupení technologií e-podnikání, a tedy IS v podnicích však každým rokem roste. V roce 2014 využívalo celkem 31 % oslovených podniků ERP systém s posunem na 34 % za tři roky. Stejně tak jsou více využívány internetové stránky podniků a digitální technologie podporující činnosti podnikání (18).



Graf 1: Využití technologií digitálního podnikání v podnicích EU za rok 2017 (18)

Informace o využití technologií digitálního podnikání za rok 2017 byly aktualizovány v druhé polovině roku 2018. Bohužel novější statistiky prozatím nejsou dostupné. Na základě aktualizací z předchozích let lze očekávat, že budou vydány až v polovině roku 2021. Vytvořil jsem tedy odhad využití těchto technologií na základě procentuálního nárůstu využití z minulých let.



Graf 2: Odhad využití technologií e-podnikání pro rok 2020 (vlastní zpracování, dle 18)

2.2 Popis firemních procesů

Informační systém by měl fungovat bok po boku se všemi firemními procesy. Je tedy třeba se zaměřit na každodenní a běžně prováděné činnosti jednotlivými zaměstnanci oční optiky.

2.2.1 Přímá obsluha zákazníka

Nejprve se zaměřím na běžnou činnost, kterou představuje obsluha zákazníka. Zákazník přijde do oční optiky a vybírá si z nabízeného sortimentu zboží. Obsluha by v případě zájmu o konkrétní zboží měla být schopna zjistit stav, zda se zboží, které zákazník požaduje, nachází na skladě a je dostupné, tedy není zarezervované. Zároveň je třeba, aby měla obsluha z jednoho místa dostupné informace o všem zboží, které může zákazníkovi nabídnout a informace o jeho skladové dostupnosti. S tím nadále souvisí odhady, kdy skladově nedostupné zboží je schopna oční optika doplnit. Pro zajištění odhadů bude třeba vést informace o dodavatelích s vazbou ke konkrétnímu zboží.

Oční optiky často nabízí svým zákazníkům i služby, které nemají skladové množství. Nejběžněji nabízenou službou je například změření zraku pro výběr dioptrických skel. Služba může být nabídnuta zákazníkovi zvlášť bez prodeje zboží. Služby je třeba vést v systému podobným způsobem jako zboží, aby bylo možné shromažďovat požadavky zákazníka na jednom místě, například za účelem vytvoření objednávky.

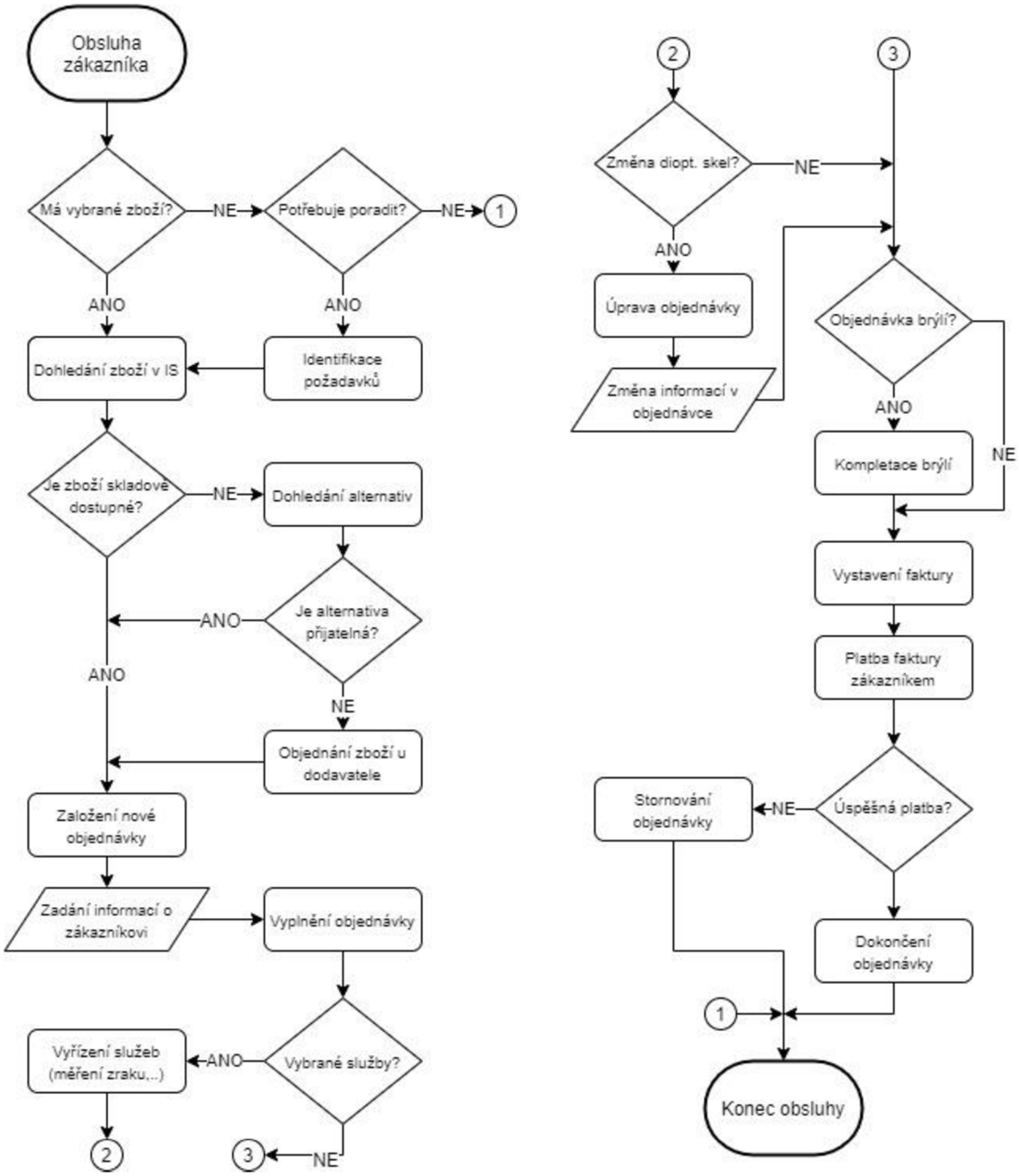
Další krok při obsluze zákazníka je vytvoření nové zakázky, jež představuje vytvoření objednávky. Při zakládání nové zakázky je třeba vést základní informace zákazníkovi s kontakty. Zákazníka je třeba dále vést v informačním systému pro zajištění kontaktních údajů, neboť si zboží může pouze zarezervovat a nevyzvednout hned. Pro zjednodušení zadávání nových zakázek mohou být o zákazníkovi vedeny i rozšiřující informace, například z měření zraku, či informace o nakupovaném zboží. V zakázce je třeba specifikovat, co zákazník objednává. Měly by se tu vyskytovat možnosti pro zadávání brýlí, obrouček, dioptrických skel, čoček a dalších položek.

Následuje vyřizování objednávky, kde se vyskytuje rozcestník. Buď má pobočka zboží dostupné na skladě a může se zákazníkem pokračovat v objednávce, či zboží skladově dostupné není a je třeba jej objednat od dodavatele. V případě nedostupnosti by bylo vhodné automaticky generovat vydanou objednávku dodavateli chybějícího zboží.

Oční optiky běžně zhotovují brýle na počkání na pobočce. Zákazník si vybere obroučky a dioptrická skla na základě měření zraku ať už konkrétní oční optikou či jiným očním lékařem. Na trhu jsou obroučky různých tvarů a velikostí. Dioptrická skla jsou dodávána převážně v univerzálních velikostech a je třeba je upravit, tzv. zabrousit. Proces zabroušení dioptrických skel probíhá již převážně na automatických frézách. Tento proces bývá zcela individuální pro každé obroučky a není třeba pro něj informační systém upravovat. Bylo by však vhodné, plnit v informačním systému informace o rozměrech obrouček, které nadále pomohou obsluze při nastavování frézy. Zabroušení dioptrických skel a zhotovení obrouček může být služba nabízená zcela zvlášť bez prodeje jiného zboží a je třeba mít možnost ji evidovat.

Jakmile jsou veškeré součásti objednávky skladově dostupné a případně zhotovené brýle, nadchází proces dokončení objednávky. K dokončení objednávky je třeba ze strany obsluhy vystavit fakturu, která vychází z předešlého objednávkového dokladu a dále v případě potřeby vytvořit záruční list na zboží. Ze strany zákazníka se očekává zaplacení faktury, tedy zboží. Zákazníci mívají několik možností, jak provést platbu. Mezi běžné možnosti platby patří platba v hotovosti, kartou, šekem, fakturou, či bývá zákazníkovi nabízena i možnost postupného splácení.

Po úspěšném dokončení platby a převzetí zboží zákazníkem je objednávka uzavřena a proces obsluhy zákazníka dokončen.



Obr. 11: Proces obsluhy zákazníka – vývojový diagram (vlastní zpracování)

2.2.2 Nepřímá obsluha zákazníka – vzdálená

Objednávky mohou být vyřizovány i telefonicky, či za pomocí emailové komunikace, tedy bez přítomnosti zákazníka. Tento proces je velmi shodný s procesem obsluhy zákazníka za jeho přítomnosti na pobočce. Rozdíl je pouze v tom, že obsluha vytvoří objednávku, která nemusí v daný moment být ihned vyřízena až do konce, zboží není v

průběhu procesu zaplaceno a převzato zákazníkem. Takovéto objednávky jsou označeny jako „vyřizující se“ a obsluha zapisuje předpokládaný termín, kdy si zákazník zboží vyzvedne osobně. Druhou možností je po zhotovení objednávky využít doručení pomocí přepravců. Nejčastěji se využívá České pošty, PPL či DPD logistických služeb. Pro dokončení takovéto objednávky je třeba zadat příkaz pro expedici, při kterém se vystavuje expediční list a objednává přepravní služba.

V budoucnu se očekává, že by objednávky mohly vznikat i přes internetový obchod neboli e-shop. Ten by pak měl být uzpůsoben pro zadávání objednávek se zajištěním potřebných informací k založení objednávky v informačním systému.

2.2.3 Reklamace

Mezi procesy, které běžně zaměstnanci očních optik řeší, jsou případná reklamace prodaného zboží či prodané služby. Obsluha musí identifikovat problém, se kterým zákazník přichází a vyřešit jej. Obsluha vytvoří reklamační protokol, do kterého uvede reklamovanou závadu a v závislosti na jejím charakteru se dále rozhoduje, jak bude spor řešen. Pokud je vzniklá závada na produktu reklamovaná v blízké době od nákupu zboží, které není zjevně fyzicky poškozeno, bývá zpravidla řešena výměnou zboží či vrácením peněz. V těchto případech vzniká příjem na sklad vadného zboží, které je třeba následně odeslat dodavateli k reklamaci. Následuje výměna zboží za nový kus, pokud je zboží skladově dostupné a vzniká nový výdej zboží ze skladu, který je třeba evidovat. V případě nedostupného zboží následuje domluva se zákazníkem, kdy zboží může být objednáno u dodavatele s dodací lhůtou, či vráceny peníze. Při dohodě o vrácení peněz se vadné zboží taktéž přijímá na sklad a vydává se peněžní částka z pokladny anebo se odesílá formou bankovního převodu. Proces reklamace je poměrně složitý, při kterém vzniká mnoho skladových a finančních operací. Očekává se podrobná evidence reklamací.

2.2.4 Hlídání skladové dostupnosti

Obchody očních optik mívají vystavené a skladově dostupné zboží, které je třeba hlídat. Zaměstnanci pobočky sledují aktuální trendy brýlí či sledují jejich vlastní prodeje. Na

základě prodejů určitého sortimentu zboží zjišťují jeho oblíbenost a skladovou dostupnost. Oblíbené zboží by mělo být dostatečně skladově dostupné. K tomu je potřeba vést statistiky prodeje zboží, zmíněného oblíbeného zboží, či „trendy“ zboží průběžně doplňovat, tedy vystavovat objednávky dodavatelům.

2.2.5 Hlídání příležitostí a rozesílání newsletter

Zaměstnanci oční optiky sledují v tabulkách nadcházející narozeniny či výročí kontaktu zákazníka s obchodem a rozesílají přání, či gratulace, často v doprovodu se slevovými kupony.

Produktový management sleduje prodeje zboží, aktuální trendy a dostupnost u dodavatelů. Pokud zboží leží na skladě příliš dlouho, zlevní jej a vystaví do akčních letáků – newsletter, které zasílají na evidované emailové adresy zákazníků.

2.2.6 Poskytování a sledování splátek

Splátky jsou poskytovány často ze strany oční optiky bez využití bankovních či nebankovních institucí, např. Home Credit. Pokud zákazník při vyřizování objednávky chce zvolit možnost nákupu na splátky, musí daná obsluha zkontrolovat jeho stav, což spočívá ve sledování obratu zákazníka u konkrétní oční optiky či přehled o tom, zda zákazník již u prodejce nesplácí nějakou z objednávek. Rovněž se vedou indexy spolehlivosti zákazníka, které si dle určitých kritérií stanovuje obsluha. Pokud zákazník splní požadavky, má nárok na poskytnutí prodeje na splátky. Poté společnost na vlastní riziko vystaví smlouvu se zákazníkem o splácení a obchod se uzavře. Prodeje na splátky se splátkovými kalendáři a zákazníky využívající této možnosti platby je potřeba vést a monitorovat pro sledování a případné rozesílání upozornění zákazníkům.

2.2.7 Inventura

V určitých intervalech jsou zaměstnanci s doprovodem vedení nuceni provádět inventury a sledovat stav pokladen. Inventura probíhá tak, že se v nepracovní den sepisuje stav skladu, respektive zboží, které se ve skladě nachází. Po kompletním soupisu zboží je aktuální stav skladu porovnán s digitálně vedeným stavem skladu. V případě rozdílů, tedy když zboží fyzicky na skladě chybí, se jej zaměstnanci snaží dohledat. V případě neúspěchu se generuje rozdílový výdej ze skladu, na kterém jsou uvedeny chybějící položky a jejich cena.

Prodeje zboží a počáteční stav pokladen jsou zapisovány do tabulek, ze kterých je pak jednoduché na konci dne pozorovat peněžní rozdíl v pokladně.

2.2.8 Evidence práce zaměstnanců

V rámci tvorby mezd a evidence zaměstnanců je třeba sledovat docházku jednotlivých zaměstnanců. Zaměstnanci si evidují své příchody do práce, pauzy v pracovní době ze soukromých účelů, pauzy na oběd a odchody. Snímky pracovního dne zaměstnanců dále slouží jako podklad pro tvorbu mezd a pro detailní sledování docházky, případně i činnosti.

2.3 Analýza požadovaných funkcí a přehledů

Funkce a přehledy informačního systému by měly být zcela přizpůsobeny k zadávání informací a práci se systémem v podnicích s oborem podnikání v oční optice. Stejně tak je třeba danému oboru přizpůsobit i struktury přehledů.

2.3.1 Zakázky

Na základě analýzy obsluhy zákazníka je třeba přizpůsobit prostředí k zadávání informací posbíraných v daném procesu. Je třeba mít přehled o všech zakázkách na jednom místě.

Zakázky je třeba evidovat přijaté a vydané s unikátním číslem pro snadnou identifikaci a dohledávání. V případě zakázek jsou požadovány evidence:

- zadavatele – zákazník, který však může zadat více zakázek,
- způsob vytvoření zakázky – např.: „Na pobočce, telefonicky, mailem“,
- zodpovědný zaměstnanec – zaměstnanec, který zakázku zadal do systému,
- telefonní a emailový kontakt na zákazníka,
- stav zakázky – např.: „vyřizuje se, čeká se na dodavatele, nevyzvednuto, vyzvednuto“,
- časový údaj o vzniku a splnění zakázky,
- zákazníkem zadaný požadovaný termínu vyřízení zakázky,
- očekávané vyzvednutí zákazníkem.

2.3.2 Objednávky

V návaznosti na zakázky je očekáván kompletní přehled všech objednávek. Bude se jednat o objednávky přijaté, které jsou přijímány na základě požadavků od zákazníků a objednávky vydané, které naopak oční optika vystavuje svým dodavatelům na základě potřeby doplnění zboží na sklad. Tyto objednávky je třeba rozeznávat pro jednoduchou orientaci a další s nimi prováděné operace.

Objednávky přijaté vychází zejména ze zakázek, a proto je mezi nimi požadována vzájemná vazba. Objednávku je možné vytvořit také jako tzv. rychlý prodej, při kterém není třeba zadávat detailní informace. Tento druh objednávky nebude vyžadovat vazbu na zakázku.

V objednávkách je třeba evidovat:

- příjemce objednávky,
- výdejce objednávky,
- kontaktní informace – kontakt na dodavatele či zákazníka převzatý ze zakázky,
- dodací údaje – upřesňující adresa pro doručení v případě využití přepravní služby, a pokud se budou dodací údaje lišit od fakturačních údajů,

- položky objednávky,
- autor objednávky,
- zaměstnanec, který měřil zrak zákazníka,
- datum vzniku objednávky,
- požadovaný způsob doručení – osobní převzetí, pošta či jiné přepravní služby,
- požadovaný způsob platby – hotovost, platba kartou, fakturou, šekem, na splátky,
- případná záloha a její výše.

2.3.3 Zákazníci

Při obsluze zákazníka je třeba pracovat se seznamem zákazníků a se základními údaji o něm. Evidence zákazníků dopomůže k urychlení vytváření opakovaných objednávek, či ke zjištění již proběhlých objednávek a informací o něm. Podstatné informace o zákazníkovi jsou jeho jméno a příjmení, bydliště, údaje o dioptrických sklech z měření zraku.

Udržování informací o zákaznících, zakázkách a půjčkách umožní vést různé statistiky a přehledy pro pozdější práci s nimi, či pro usnadnění rozhodování.

2.3.4 Evidence pokladny

Vzhledem k práci se zakázkami a evidenci objednávek probíhají peněžní pohyby, které musí být evidovány, a proto je třeba vést seznam příjmů. Požaduje se kontrola tržeb a počtu zákazníků v průběhu celého dne ke zjištění okamžitého stavu. Tyto statistiky poslouží k ekonomickému hodnocení a optimalizace chodu oční optiky. Očekává se vyhodnocení pohybů a zisku za libovolné období. Zároveň se počítá i s evidencí příkazů k úhradě dodavatelům, jež představují odvody z pokladny. V pokladně je potřeba evidovat všechny výdaje a příjmy dané pobočky.

Očekává se funkce uzávěrky pokladny, včetně možnosti zobrazení předcházejících denních uzávěrek.

2.3.5 Recepty a dávky pojišťovnám

Zákazníci mohou pro nákup dioptrických brýlí využívat receptů vydaných očními lékaři. V těchto receptech jsou uvedeny diagnózy, dále kódy PTZ, zboží a pojišťoven, IČZ lékaře a základní informace o zákazníkovi.

Očekává se aktualizace a evidence ceníků PTZ, evidence IČZ lékařů se základními informacemi. S recepty je potřeba pracovat v zakázkách, ve kterých ponižují výslednou cenu. Suma cen receptů, o kterou byly sníženy ceny zakázek, se vydává k úhradě pojišťovnám. Z tohoto důvodu je potřeba vytvořit přehled přijatých receptů na základě, kterých budou generovat dávky pojišťovnám k proplacení. Recepty se vyplňují a mohou být tedy upravovány.

2.3.6 Přehledy zboží a ceníky

Vzhledem k vystavování a prodeji zboží je potřeba vést seznam zboží a udržovat informace o jejich skladovém množství. Každá položka potřebuje mít svůj unikátní identifikátor, název a krátký popis. Dioptrická skla mají mnoho vlastností, které popisují atributy AX (osa skla), PD (vzdálenost očí od sebe), ADD, výšku nad nosem, Prs (prisma), Bas (báze), průměr skla. Zároveň kontaktní čočky mají navíc vlastnost Bc, udávající informaci o zakřivení čočky. Veškeré tyto informace je potřeba evidovat k danému zboží. Vyplnění veškerých atributů však není nutné, neboť optiky evidují různé druhy zboží.

Zboží umístěné na skladě nese informaci o skladu, kde se nachází. Optimálně by mělo být doplněné i konkrétní umístění na skladě v případech, kdy je sklad podle umístění rozdělen. Nabízí se evidence informace o aktuálním skladovém množství, předpokládaném budoucím stavu ovlivněném budoucími výdeji a příjmy. Dále bylo zmíněno, že se hlídá minimální skladové množství, které by mělo být součástí přehledu pro možné budoucí generování reportů. Každé zboží má svoji finanční hodnotu neboli cenu. Cen může být mnoho, nákupní a prodejní, vedené historicky. Bude potřeba vytvořit přehled ceníku k jednotlivým položkám zboží.

2.3.7 Emailová komunikace

Vzhledem k evidenci zakázek, pokud se jejich vyřizovací doba protáhne na několik dní, starají se zaměstnanci optik o rozesílání informací zákazníkům k dané zakázce. Zároveň odesílají upomínky, reklamace, akční nabídky, případně přání k narozeninám. Očekává se automatizace těchto procesů pomocí automatického generování a odesílání emailů.

2.4 Průzkum trhu

Na trhu již existuje řada informačních systémů, liší se v mnoha faktorech, jako je funkcionalita, cena či segment podniků, na které cílí. V této části práce se budu zabývat analýzou a zhodnocením stávajících informačních systémů, ke kterým vytvářím alternativu.

Navrhovaný informační systém směřuji funkcionalitou a strukturou na segment očních optiků. Zaměření na oční optiky jsem si vybral zejména z důvodu nízkého výskytu konkurenčních informačních systémů s touto problematikou. Společnosti, zaměřující se na oční optiky, mají v tuto chvíli možnost vybrat si z relativně malého počtu informačních systémů, které jsou zaměřeny na jejich obor podnikání, nebo zvolit některý z obecných informačních systémů. Obecné informační systémy pak pro ně nemusí být zcela vyhovující z důvodu jejich universálnosti. Tyto společnosti v případě obecných informačních systémů platí za funkcionality, kterých nevyužívají a naopak jim některé funkce mohou chybět.

Mezi nejrozsáhlejší české informační systémy se zaměřením zejména na oční optiky patří Newton od společnosti Maxsoft, s.r.o., Tiss od společnosti TISS Optic CZ, s.r.o., dále pak případně rozhraní softwaru OPTIK implementované v informačním systému ABRA od společnosti ABRA Software, a.s.

Universálních podnikových informačních systémů je mnoho. K analýze využiji český informační systém, který je nejrozsáhlejší v menších a středních podnicích Helios Orange od společnosti Asseco Solutions, a.s.

2.4.1 Newton

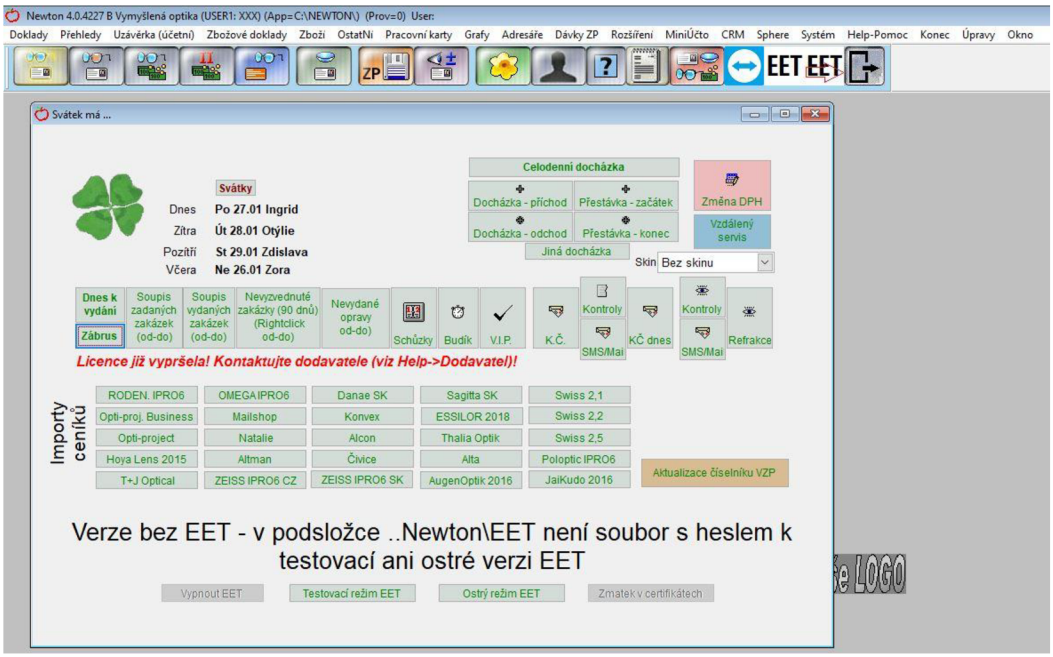
Newton od společnosti Maxsoft, s.r.o. je specializovaný informační systém pro oční optiky. Jedná se o první a zároveň nejrozšířenější produkt společnosti. Software je vyvíjen již velmi dlouho a obsahuje většinu potřebných funkcí pro oční optiky (19).

V informační systému lze evidovat:

- zakázky brýlí s podporou zábrusu a zakázkové karty na opravu,
- vyšetření očí pro brýle i kontaktní čočky,
- uživatele tvrdých a měkkých kontaktních čoček,
- optické a zdravotnické pomůcky,
- zákazníky, dodavatele,
- sklady – vlastní majetek, skladové doklady,
- ceníky a dávky pro zdravotní pojišťovny,
- evidence pracovní doby zaměstnanců,
- vedení pokladny a finanční operace (19).

Systém umožňuje práci se zákazníky, jež zahrnuje správu zakázek a posílání hromadných mailů či SMS. V systému lze generovat tiskové formuláře k přehledům a pro doklady, evidovat pokladny a vytvářet pohledy či grafy s měsíčními přehledy. V posledních verzích přibývá i podpora elektronické evidence tržeb – EET (19).

Prostředí i zadávací formuláře jsou plně přizpůsobeny očním optikům. Editor v režimu zakázek obsahuje informace o zákazníkovi, brýlích, dioptrických sklech a čočkách, ceny, slevy a mnohé jiné atributy, se kterými optici pracují při vytváření nových zakázek.

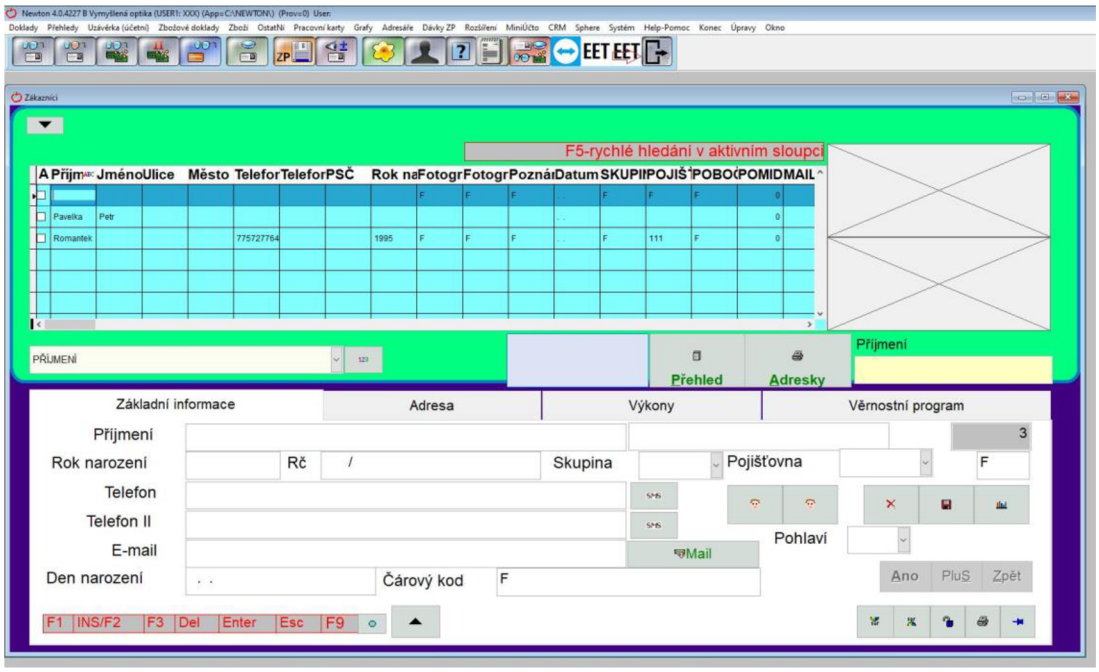


Obr. 12: IS Newton – úvodní obrazovka (vlastní zpracování)

Informační systém je rozhodně dobře vybaven, bohužel po funkční a vizuální stránce je systém velmi zastaralý. Použité komponenty, nejčastěji tlačítka, nejsou výrazné a splývají mezi sebou. Uživatelé tohoto systému jsou pravděpodobně na vzhled zvyklí, pro nové zákazníky však může být nevyhovující. Vzhled a struktura rozmístění komponent není zcela intuitivní, rozmístění nejen tlačítek, pak může působit chaoticky.

V přehledech jsou použité dnes již nevhodné zobrazovací komponenty, ve kterých zaniká text a většina zadávacích formulářů je zbytečně malá z důvodu umístění pod přehledy. Přehledy jsou z důvodu rozmístění rovněž zmenšené a zobrazují pouze několik jednotek záznamů naráz.

Seznam sloupců, představující atributy z editoru, není zcela kompletní, například v přehledu zákazníků chybí RČ – rodné číslo a přehledy nejsou uživatelsky přizpůsobitelné, nelze přidat či odebrat zobrazované atributy.

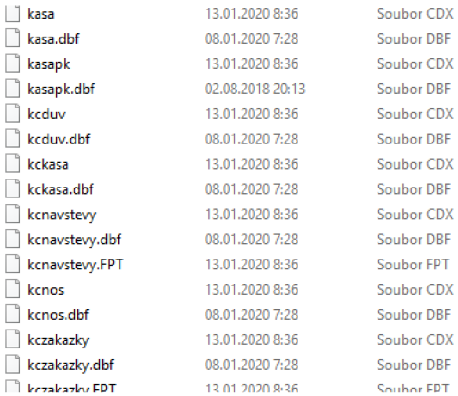


Obr. 13: IS Newton – přehled zákazníků s editorem (vlastní zpracování)

V editorech chybí potvrzovací hláška při změně záznamu. Rovněž chybí i upozornění na rozpracované záznamy v případě zavření editoru s přehledem. Při ukládání záznamů je použito pouze minimum kontrolních omezení, například duplicitně použité rodné číslo, duplicitní čísla dokladů a jiné. Filtrovat přehledy lze pouze pomocí nadefinovaných atributů z rozevíracího seznamu. Vždy lze filtrovat pouze na základě jediného atributu, nelze určit více filtrovacích podmínek zároveň.

Řazení neboli třídění v přehledech lze aplikovat pouze na sloupce, u kterých je nadefinováno.

Informační systém je naprogramován ve vývojářském prostředí Microsoft Visual ForPro. Veškeré uživatelsky vkládané informace a data z přehledů jsou uloženy na lokálním disku v databázových souborech DBF a CDX. Vývojáři do systému implementovali možnost cloudového zálohování, s možností propojení několika počítačů, tedy instancí s přístupem ke stejným datům. Celková funkcionalita systému běží na koncovém zařízení, a proto jsou na tyto zařízení – počítače kladeny vyšší nároky na výkon a kapacitu uložišť.



Obr. 14: IS Newton – způsob lokálního ukládání dat (vlastní zpracování)

Vhodnější by bylo využití databáze SQL, která pracuje na společném serverovém počítači a veškeré dotazy a operace s daty probíhají zde. Pomocí databáze SQL by byl kladen menší nárok na výkon koncových zařízení, kterých může být k centralizované databázi připojeno hned několik.

Cena licence informačního systému Newton se všemi moduly je 14 500 Kč bez DPH na jednu provozovnu. Informace o systémové podpoře a její ceně není na stránkách výrobce uvedena (19).

2.4.2 TISS Optic

TISS Optic je online informační systém zaměřený na oční optiky pro, které má přizpůsobený design a funkcionalitu. Zaměřuje se na malé i více pobočkové optiky, e-shopy se zbožím optiky nebo pro oční lékaře (20).

Pro malé a více pobočkové optiky software nabízí:

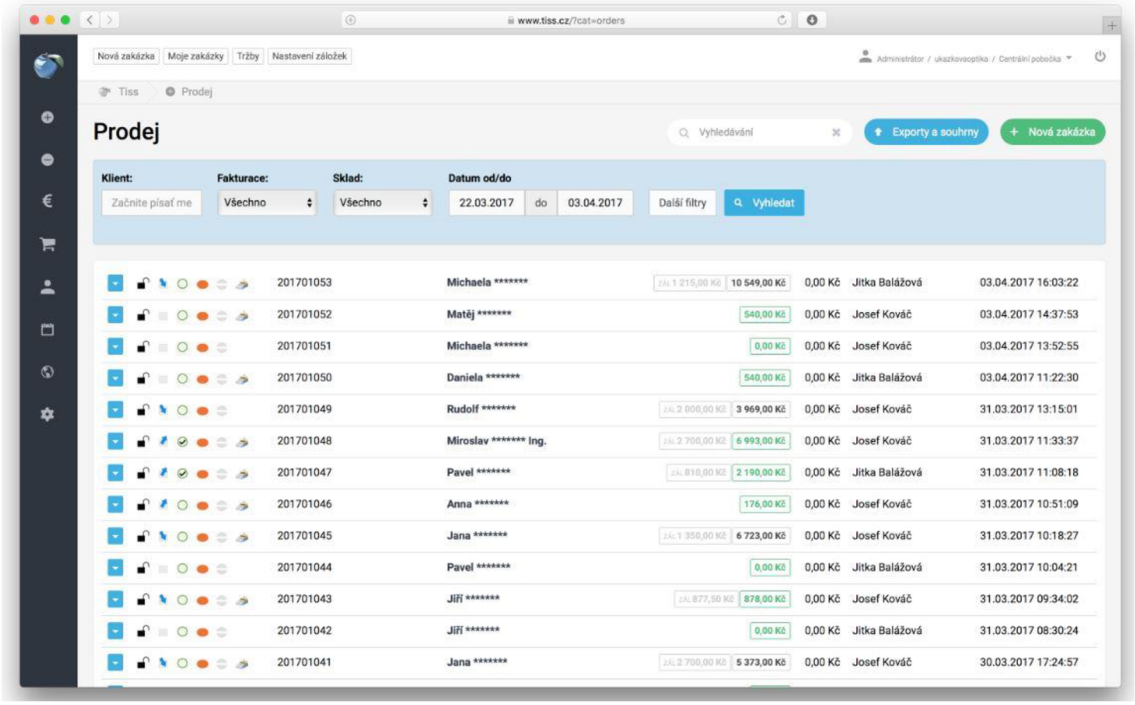
- evidenci a operace se zakázkami,
- maloobchodní prodej a komunikaci s pokladnami,
- přehled a řízení reklamací,
- tvorby a úpravy ceníků,
- generování dávek pro pojišťovny,
- evidence dokladů – došlých a vydaných faktur, dodací listy, a jiné
- skladové operace,

- evidence zákazníků a dodavatelů,
- statistiky prodejů a výkonů prodejen,
- online – cloudový přístup k systému,
- práce z jakéhokoli zařízení – mobil, tablet, počítač a jiné,
- vedení pokladny a finanční operace,
- evidenci vyšetření očí,
- propojení s webovými stránkami (20).

TISS Optic je velmi propracovaný moderní informační systém, který existuje na trhu po dobu více jak deseti let. Pracuje v online prostředí skrze internetový prohlížeč, všechna data jsou uložena na online cloudovém uložišti. Mezi nesporné výhody patří možnost použití na jakémkoli zařízení s přístupem k internetu. V posledních verzích je informační systém optimalizován pro běžné použití i na tabletech. V jedné z posledních verzí přibyla funkce umožňující sbírat digitální podpisy z tabletu.

Bohužel není dostupná testovací verze či demoverze systému TISS Optic, k analýze využiji ukázkového videa s doprovodným textem.

Prostředí systému je velmi dobře zpracované, přehledné a intuitivní. Při vytváření zakázek je možné přidat do jedné zakázky více položek, brýlí, kontaktních čoček a jiné. V rámci jedné zakázky lze definovat dioptrické údaje zákazníka v sekci pracovní karty. Výběr již existujících položek či zákazníků je formou přenosových přehledů.



Obr. 15: IS TISS Optic – prodej (vlastní zpracování)

Systém podporuje čárové kódy, které umí generovat či podle nich dohledávat zboží. O stav zakázky, či jiné události lze zákazníky informovat vestavěného modulu na odesílání SMS zpráv. Každá odeslaná SMS zpráva ze systému je však zpoplatněna.

Uživatelé informačního systému lze rozdělit do skupin, s předem nadefinovanými právy. U jednotlivých uživatelů je možné sledovat informace o jejich činnosti. Jedná se zejména o přihlášení a odhlášení ze systému či záznam o nových záznamech, které uživatel vytvořil.

V systému lze pozorovat různé statistické informace o uživatelích, nelze v něm však evidovat pracovní dobu a snímky pracovního dne.

Online informační systém má své výhody v možnostech téměř neomezeného přístupu k systému odkudkoli. Nevýhodou bych viděl nutnost neustálého internetového připojení. V případě výpadku či jiných problémech internetového připojení není možné systém používat vůbec či s překážkami. Zákazník využívající informační systém má přístup ke svým datům, které však nemá umístěny u sebe na svém počítači či vlastním serveru, ale na cloudovém uložišti třetí strany. Nabízí se otázky: „Jak je toto řešení

bezpečné? Je zákazník ochotný podstoupit riziko, že jsou jeho data umístěna na cizím uložišti? “.

Cena informačního systému je tvořena implementačním poplatkem a dále pravidelnými měsíčními poplatky za používání informačního systému.

Poplatek	Cena
Implementační poplatek pro 1. pobočku	14 990 Kč
Implementační poplatek pro každou další pobočku	11 990 Kč

Poplatek	1 uživatel	2-3 uživatelé	4-7 uživatelé	8 a více uživatelů
Firemní uživatelský poplatek bez ohledu na počet poboček	1 799 Kč	2 190 Kč	2 490 Kč	dohodou

Obr. 16: IS TISS Optic – ceník (20)

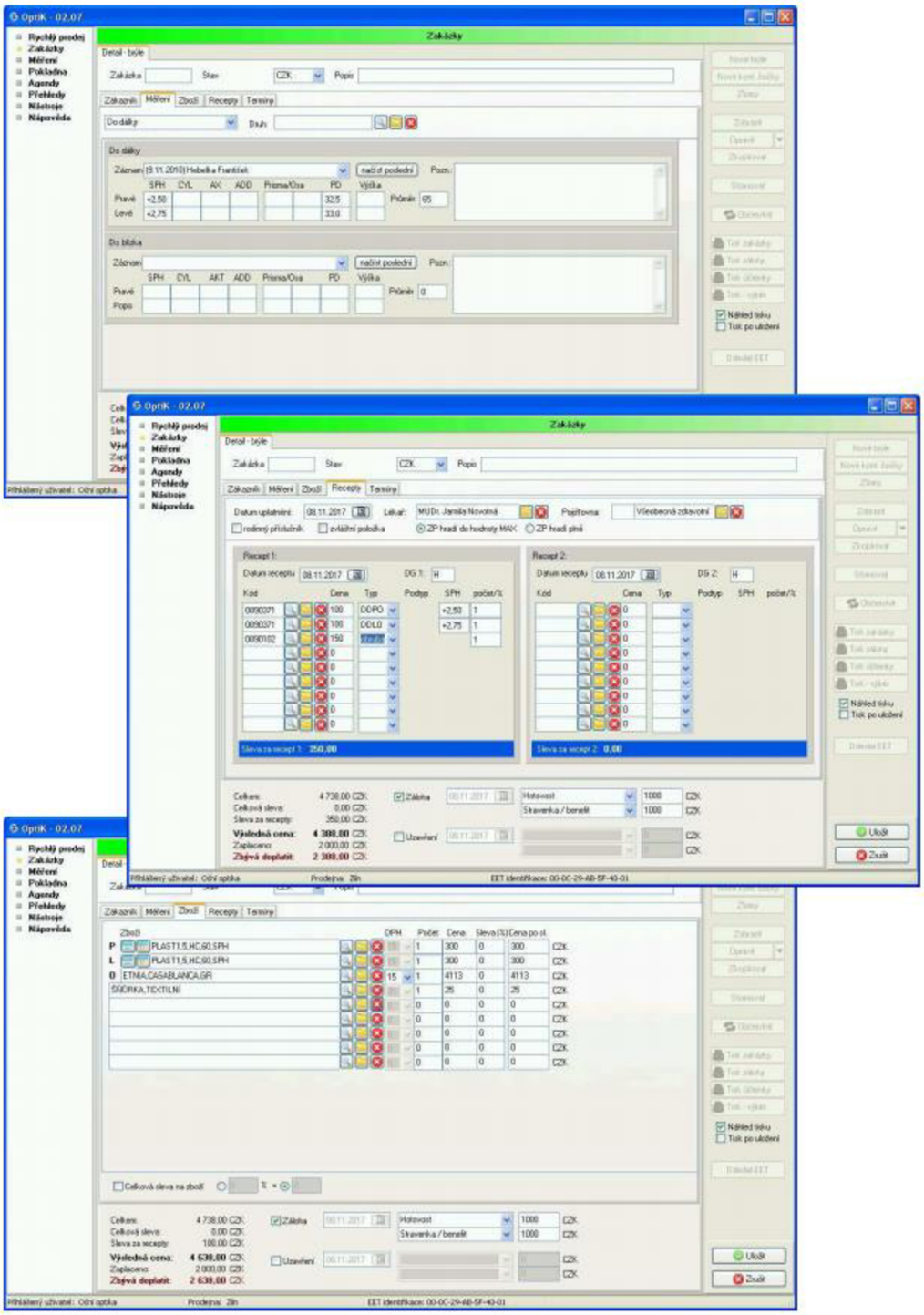
2.4.3 OPTIK

Informační systém OPTIK vznikl spojením systému ABRA s nadstavbou OPTIK. Vznikl tak kompletní ERP systém pro řízení mnoha prodejen očních optik a řízení firemních procesů (21).

Řešení OPTIK přináší řešení:

- zakázek brýlí a dioptrických čoček,
- přímého prodeje hotových výrobků,
- měření zraku a ošetřujících lékařů,
- zákazníků a jejich aktivit,
- ceníků a dávek pro zdravotní pojišťovny,
- pokladny, banky, účetnictví a EET,
- skladových operací, reklamací a majetku,
- docházky, zaměstnanců, mezd a personalistiky,
- nákupu a marketingu,
- statistik tržeb, pohledávek či závazků (21).

Informační systém OPTIK nelze pořídit v testovací či demo verzi programu. Při analýze budu vycházet z veřejně dostupné brožury, kterou společnost ADMIS CZ, s.r.o. dodává. V brožuře se nachází reálné snímky z programu i s popiskem, jak jej lze využít (21).



Obr. 17: IS OPTIK – ukázka přehledů a formulářů (21)

Základní prostředí informačního systému je velmi jednoduché a snadno pochopitelné. Skládá se z postranní lišty, kde se nachází seznam funkcí a přehledů a zbytek obrazovky tvoří pracovní prostředí (21).

Při zakládání zakázek se eviduje zákazník, měření zraku, zboží, výběr receptů a očekávané termíny. Editory vypadají velmi stručně, a ne příliš přehledně, chybí zde například přenosové editory pro výběr zákazníků či položek. Místo nich se v systému používají rozevírací seznamy (21).

Jak již bylo zmíněno, OPTIK je softwarová nadstavba plnohodnotného ERP systému ABRA. Využívá tedy stejné datové základny s možnostmi využití v celé firmě (21).

Mezi další funkce patří číselníky zdravotních potřeb, měření zraku, zákazníků, dodavatelů a jiné. V modulu skladového hospodářství je možné řídit veškeré skladové operace s dostupnými ceníky (21).

Jádro systému podporuje práci s čárovými kódy typu EAN a lze přidat doplňkové moduly propojující informační systém se čtečkami čárových kódů (21).

Prostředí informačního systému na mě působí zastarale a velmi jednoduše. Jednotlivá okna se navzájem překrývají a nevyužívá se vyskakovacích oken (21).

Společnost ADMIS CZ, s.r.o. bohužel neposkytuje o svém informačním systému více informací veřejně, pouze potencionálním zákazníkům. Cenu tohoto informačního systému rovněž nelze dohledat a bude individuální na základě každé implementace a použitých modulů (21).

2.4.4 Helios Orange

Informační systém Helios Orange patří mezi nejrozšířenější kompletní podnikové informační systémy se zaměřením na malé a střední firmy (22).

Helios Orange obsahuje řadu modulů pro práci:

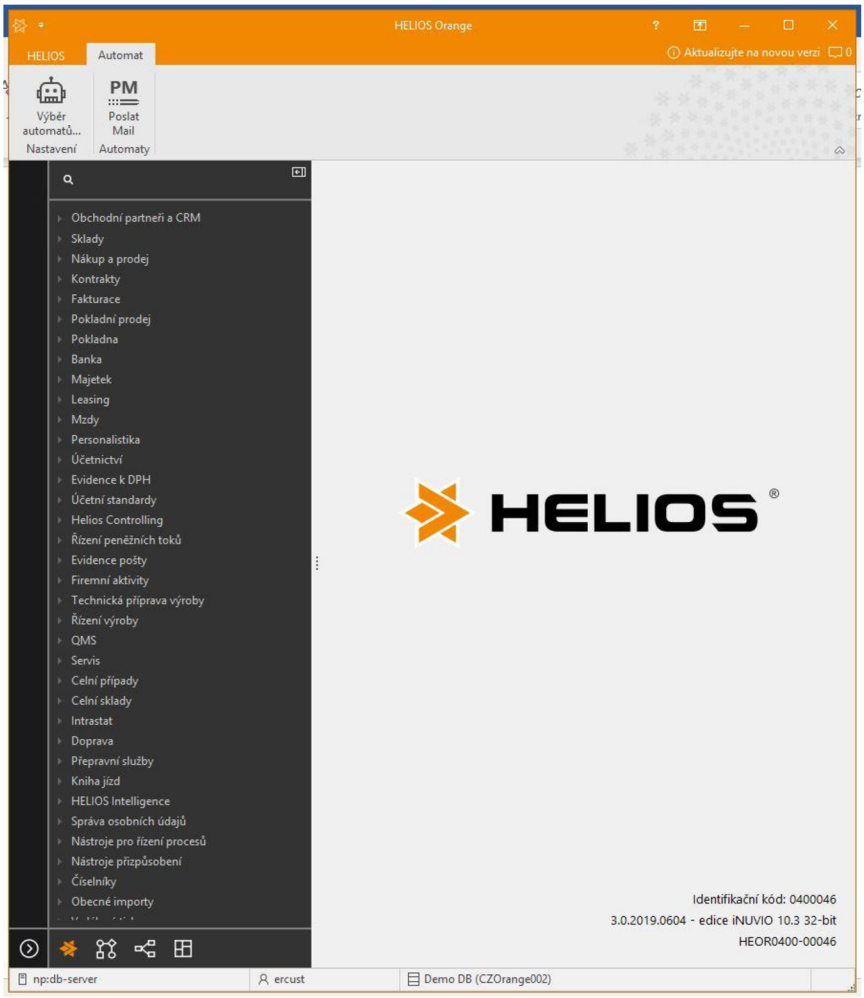
- s obchodními partnery a CRM,
- se sklady, skladovými položkami a inventury,
- s pohyby zboží v rámci příjemek, výdejek, objednávek a jiné,
- s pokladnou, pokladním prodejem, platebními příkazy či bankovními výpisy,
- s kontrakty a odvolávkami,
- s fakturací a řízením peněžních toků,
- s evidencí majetku včetně jeho pohybů,
- s leasingy, opakovanými platbami či splátkami,
- se zaměstnanci, mzdovými údaji a personalistikou,
- s kompletním účetnictvím, účetními standarty a evidencemi k DPH,
- s evidencí firemních aktivit a pošty,
- s technickou přípravou výroby a řízením výroby,
- se servisními úkony a reklamacemi,
- s celními sklady a případy,
- s dopravou, přepravními službami a knihou jízd,
- se statistickými informacemi o vedení podniku a business inteligence,
- s řízením procesů a workflow,
- a mnohé další.

Informační systém je velmi mohutný a obsahuje nespočet funkcí, které lze pořídit ve formě dodatečných modulů či pluginů, které lze za pochodu pořídit. O informační systém se mimo jeho zakladatelskou společnost Asseco Solutions, a.s. stará mnoho jiných společností v partnerské síti, které systém implementují a vytváří téměř jakékoli zakázkové úpravy na základě požadavků zákazníků.

Helios Orange podporuje automaty či časově organizované úkoly, které pak provádí automaticky na základě definicí uživatele. Umožňuje tak například odesílat i emaily s různými upozorněními.

Podporované jsou i čtečky čárových kódů, či pokročilé scannery s operačním systémem Windows či Android.

Informační systém využívá pro ukládání dat databázi pracující v prostředí Microsoft SQL. Lze jej propojit i s jinými informačními systémy či webovými portály.

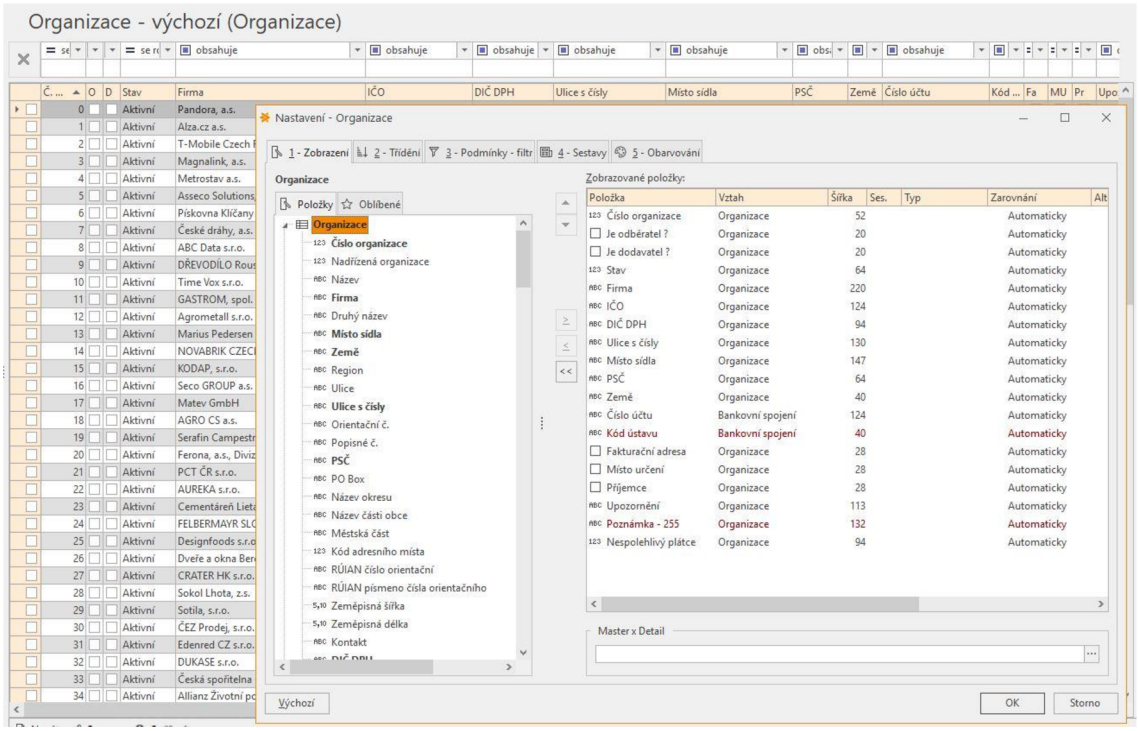


Obr. 18: ERP Helios Orange – úvodní obrazovka (vlastní zpracování)

Jak jsem již zmínil, jedná se o obecný informační systém, který není napřímo přizpůsoben procesům očních optik. Chybí zde moduly upravující formuláře pro zadávání zakázek, zápisu údajů z měření zraku, zadávání lékařských receptů či evidenci pojišťoven. Stejně tak nejsou těmto procesům přizpůsobeny struktury tabulek.

Prostředí je však velmi moderní a přehledné. Nově jsou v Helios Orange vydání iNUVIO implementovány pásy karet – RIBBON.

Inspirující je způsob filtrování v přehledech, kdy lze filtrovat na základě více sloupců zároveň. Stejně tak zobrazování sloupců přehledů, jejichž přidávání či odebírání je plně uživatelsky konfigurovatelné.



Obr. 19: ERP Helios Orange – číselník organizací a vlastnosti přehledu (vlastní zpracování)

Cena informačního systému Helios Orange je velmi individuální na základě požadavků od zákazníka, velikosti firmy, rozsahu podnikání a společnosti, která systém implementuje. Lze však konstatovat, že základní cena bude mnohem vyšší než u informačních systémů zaměřených pouze na oční optiky (22).

2.5 Shrnutí analýzy současného stavu

Na začátku analýzy současného stavu jsem zjistil, že v podnicích České republiky a obecně podnicích států Evropské unie je prozatím velmi malé procento využití technologií digitálního podnikání, ke kterým patří i informační systémy. Zejména v malých a středních podnicích je využití jakýchkoli informačních systémů minimální, a právě tato část trhu se stává pro mě zajímavou.

V druhé části analýz jsem důkladněji rozebral firemní procesy, kterými běžně prochází zaměstnanci a vedení očních optik každým pracovním dnem. Na základě těchto analýz budu v následující kapitole navrhovat strukturu databáze a vytvářet jak grafické, tak i funkční rozhraní nového informačního systému pro optiky.

Oční optiky, které nevyužívají pro svoji činnost žádného digitálního informačního systému, nemohou být efektivní. Důvody neefektivnosti řešení bez digitálního informačního systému vystihuje SWOT analýza tohoto řešení v tabulce níže.

Tab. 2: SWOT analýza IS tužka papír (vlastní zpracování)

SILNÉ STRÁNKY	SLABÉ STRÁNKY
• Minimální pořizovací náklady • Funguje i v případě výpadku elektriky a internetu • Naprostá ochrana proti kybernetickým útokům	• Papírová evidence • Zabezpečení • Vyhledávání • Efektivita • Časová náročnost • Negativní vliv na životní prostředí (spotřeba papíru) • Neexistence sdílení dat pomocí společné databáze
PŘÍLEŽITOSTI	**HROZBY**
• Začínající podniky	• Ztráta dokumentů • Poškození dokumentů vlivem klimatických podmínek • Nemožnost dohledat dokumenty pro případné předložení státní správě

Při průzkumu trhu jsem zjistil, že na trhu sice existuje množina konkurenčních řešení, které mohou oční optici využít, nicméně vyšly najevo i jejich výrazné nedostatky, které mi otevírají možnosti ke vstupu s novým IS. Obecné IS se pro tuto činnost zkrátka nehodí, neboť nemají optimalizované prostředí pro analyzované procesy. Největším problémem současných IS pro oční optiky je jejich stáří, a tedy využití zastaralých metod a zastaralých vývojových prostředí, která již nemají budoucnost s tím, že ve většině případech již nejsou výrobci podporovány. Právě z těchto důvodů jsem se rozhodl použít

vývojové prostředí Delphi, které je stále udržované aktuálním, a jeho komponenty jsou rovněž inovovány s možností tvorby vlastních. Díky novému vývojovému prostředí spolu s databázovým systémem Microsoft SQL Sever mám mnoho nových možností, které není možné realizovat ve většině stávajících IS pod zastaralým vývojovým prostředím.

3 VLASTNÍ NÁVRHY ŘEŠENÍ

V této kapitole se budu zabývat vlastními návrhy nového informačního systému, založenými na analýze z předešlé kapitoly. Hlavními úkoly v této kapitole budou návrh a vytvoření databáze se strukturami, se kterými bude IS pracovat. A poté design a tvorba samotného informačního systému. Pro informační systém je třeba navrhnout grafické rozhraní, funkcionalitu a komunikaci s databází SQL. Databázový produkt bude použit Microsoft SQL Server s rozhraním Management studio. Vývoj designu a funkcionality informačního systému bude probíhat v programovacím prostředí Delphi.

3.1 Příprava

V této podkapitole se budu zabývat přípravnou fází, kterou tvoří instalace potřebného software pro tvorbu informačního systému. K tomu využiji databázový produkt Microsoft SQL Server 2019 v edici Express, komunikační rozhraní Microsoft SQL Server Management Studio ve verzi 18 a Microsoft SQL Server Profiler ve verzi 18. Dále využiji vývojového prostředí Delphi 10.3 Community Edition.

3.1.1 Instalace a registrace Delphi

Společnost Embarcadero, která je vývojářem prostředí Delphi nabízí začínajícím programátorům zdarma komunitní edici. Tato edice Delphi je plnohodnotným vývojářským nástrojem s drobnými omezeními. Edici lze využít i pro komerční účely za předpokladu, že tržby z prodeje vytvořených aplikací nesmí přesáhnout částku 5000 amerických dolarů za rok.

Na webovém portálu Embarcadero pak lze jednoduše zaregistrovat nový účet, pro který je nutné vyplnit základní informace jako jméno a příjmení, heslo, email a telefonní číslo. Na e-mailovou adresu poté přijde potvrzovací zpráva s vygenerovaným licenčním číslem a odkazem ke stažení produktu Delphi ve verzi Community Edition. Instalace je velmi jednoduchá a intuitivní, stačí vybrat produkt, který požadujeme, nainstalovat a vybrat

k němu moduly. Na závěr je potřeba zadat licenční číslo, a poté je vývojové prostředí připravené k použití.

3.1.2 Instalace Microsoft SQL Serveru

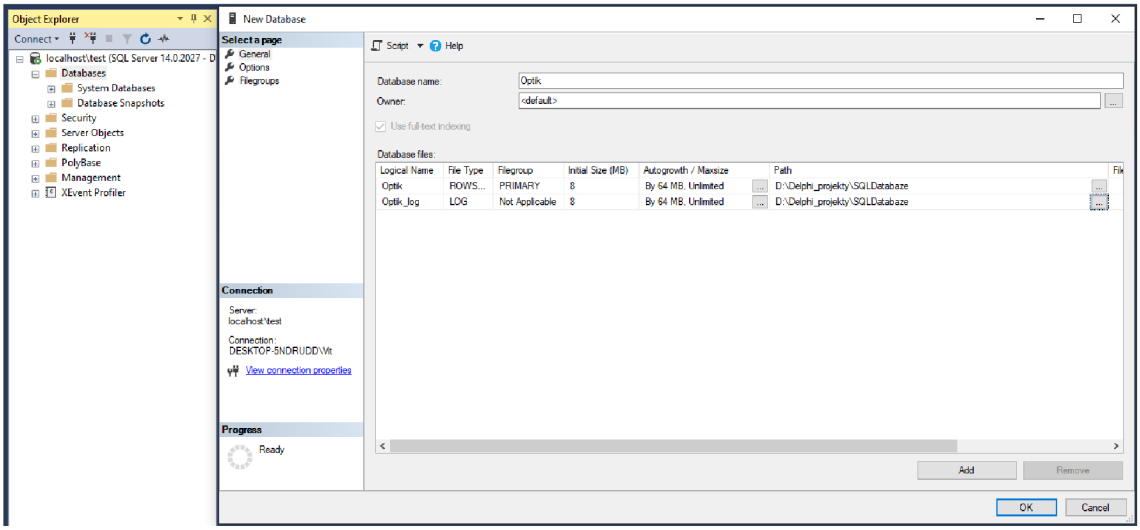
Microsoft SQL Server je druhý softwarový produkt, který je stěžejní pro tvorbu mého informačního systému. Microsoft nabízí Express edici SQL Serveru, která je zcela zdarma i pro komerční využití. Má však řadu omezení, mezi která patři:

- Instance SQL Serveru umí využít maximálně jeden procesor (max. 4 jádra)
- Omezené využití operační paměti (max. 1410 MB pro instanci)
- Omezená velikost databáze na 10 GB.

Pro začínající podniky a začínající informační systémy jsou tato omezení dostačující. Na webovém portálu Microsoft, lze v sekci ke stažení stáhnout instalační soubor k produktu Microsoft SQL Server v aktuální verzi. Instalace je opět velmi intuitivní a rychlá. V dalším kroku je třeba zvlášť nainstalovat komunikační rozhraní Management Studio, které je rovněž volně dostupné ke stažení na webovém portálu Microsoft. Instalace již sama detekuje instanci SQL Serveru a spáruje se s ní.

3.2 Vytvoření databáze a jejích struktur

Na začátku si vytvořím novou databázi na SQL serveru za pomocí průvodce v Microsoft SQL Server Management Studio. Prozatím stačí zvolit pouze jméno databáze a její umístění. Ostatní nastavení ponechám ve výchozích hodnotách.



Obr. 20: Založení nové databáze na SQL Serveru (vlastní zpracování)

3.2.1 Vytvoření relačních tabulek databáze

Ve stejném rozhraní Microsoft SQL Server Management Studio vytvořím relační tabulky na základě druhé kapitoly analýzy současného stavu. V předešlém kroku jsem vytvořil novou databázi Optik, ve které se budou nacházet veškeré relační tabulky a data, se kterými bude informační systém pracovat. V prostředí Management Studio lze využít průvodců pro vytváření nových relací, kterých však nevyužiji a upřednostním ruční psaní definice tabulek, jejich omezení a vazeb.

Definice tabulek probíhá pojmenováním tabulky, a dále definicí jejích atributů s datovými typy omezením na hodnotu NULL či omezením atributu. V následujících obrázcích je možné vidět například definici atributu ID, který je číselného datového typu Integer a má nastavený parametr IDENTITY, pomocí kterého získá každý nově založený řádek v tabulce novou hodnotu ID inkrementovanou o jedničku. Některé atributy tabulek mají nastavené defaultní hodnoty, které budou vyplněny v případě, že nebudou nadefinovány při vstupu dat. Každá vytvořená tabulka má atribut ID, který je jejím primárním klíčem a je zajištěno jeho automatické plnění. Dále definuji cizí klíče tabulky, a tím propojení jednotlivých tabulek mezi sebou za předpokladu, že spolu souvisí.

```
CREATE TABLE dbo.TabZbozi
(
ID INT IDENTITY (1, 1),
KOD NVARCHAR(30) UNIQUE NOT NULL,
Nazev NVARCHAR(100) NULL,
Skupina NVARCHAR(3) NULL,
SPH NVARCHAR(5) NULL,
CYL NVARCHAR(5) NULL,
AX INT NULL,
PD NUMERIC(7,2) NULL,
ADD1 NUMERIC(7,2) NULL,
Vyska NUMERIC(7,2) NULL,
Prs NUMERIC(7,2) NULL,
Bas INT NULL,
Bc NUMERIC(7,2) NULL,
Hmotnost NUMERIC(19,6) NULL,
MJ NVARCHAR(10) NULL,
Poznamka NVARCHAR(4000) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID)
)
```

Obr. 21: Definice tabulky TabZbozi (vlastní zpracování)

Navrhl jsem tabulku pro evidenci zboží **TabZbozi**. Do této tabulky budou zadávány veškeré fyzické položky, představující zejména zboží, které je možné evidovat pod specifickým kódem, jež může představovat i čárový kód, a s vlastním názvem pro pojmenování každé položky. Pokud položka představuje dioptrické sklo, pak je možné zadat doplňující informace, které dané sklo blíže identifikují.

```
CREATE TABLE dbo.TabSkladZbozi
(
ID INT IDENTITY(1, 1),
IDZbo INT NOT NULL,
Mnozstvi NUMERIC(19,6) NULL,
BudMnoz NUMERIC(19,6) NULL,
MinMnoz NUMERIC(19,6) NULL,
BlokMnoz NUMERIC(19,6) NULL,
Sklad NVARCHAR(20) NULL,
Popis NVARCHAR(100) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID),
CONSTRAINT FK_SZ_IDZbo FOREIGN KEY (IDZbo)
  REFERENCES TabZbozi(ID)
)
```

Obr. 22: Definice tabulky TabSkladZbozi (vlastní zpracování)

Tabulka **TabSkladZbozi** slouží pro specifikaci, na jakém skladě se určitá položka nachází a v jakém množství. Díky rozdělení informací o položce a skladovém množství je možné vést informace o umístění zboží na více skladech, či skladových pozicích zároveň. V této tabulce se také nachází atribut **budoucí množství**, který bude zohledňovat budoucí stav skladu po příjmu či výdeji zboží. Atribut **minimální množství**, bude sloužit pro stanovení minima, na které bude upozorněno v případě poklesu množství pod něj.

V poslední řadě se zde nachází atribut, představující počet blokovaného zboží, který dá informaci o situaci, kdy se zboží na skladě nachází, ale je zablokované v určitém počtu pro zakázky.

```
CREATE TABLE dbo.TabCenik
(
ID INT IDENTITY(1, 1),
IDZbo INT NOT NULL,
Cena NUMERIC(19,6) NULL,
CenaNakup NUMERIC(19,6) NULL,
PlatnostOD DATETIME NULL,
PlatnostDO DATETIME NULL,
SazbaDPH numeric(19,6) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID),
CONSTRAINT FK_Cenik_IDZbo FOREIGN KEY (IDZbo)
  REFERENCES TabZbozi(ID)
)
```

Obr. 23: Definice tabulky TabCenik (vlastní zpracování)

Tabulka **TabCenik** slouží pro evidenci nákupní a prodejní ceny určité položky.

```
CREATE TABLE dbo.TabPojistovny
(
ID INT IDENTITY(1, 1),
Cislo nvarchar(5) NOT NULL UNIQUE,
Nazev nvarchar(100) NOT NULL,
UliceCP nvarchar(50) NULL,
Mesto nvarchar(30) NULL,
PSC nvarchar(6) NULL,
Telefon nvarchar(13) NULL,
ICO nvarchar(20) NULL,
VarSymbol nvarchar(20) NULL,
PRIMARY KEY(ID)
)
```

Obr. 24: Definice tabulky TabPojistovny (vlastní zpracování)

Tabulka **TabPojistovny** shromažďuje informace o pojišťovnách, které jsou klíčovou informací v případě receptů, pohybových dokladů a v zakázkách.

```
CREATE TABLE dbo.TabOrganizace
(
ID INT IDENTITY(1, 1),
Firma nvarchar(45) NULL,
Jmeno nvarchar(15) NULL,
Prijmeni nvarchar(30) NULL,
RodneCislo nvarchar(11) NULL,
TitulPred nvarchar(10) NULL,
TitulZa nvarchar(15) NULL,
UliceCP nvarchar(50) NULL,
Mesto nvarchar(30) NULL,
PSC nvarchar(6) NULL,
Telefon nvarchar(13) NULL,
Email nvarchar(100) NULL,
VIP bit NULL,
Stat nvarchar(5) NULL,
Stav int NULL,
ICO nvarchar(20) NULL,
DIC nvarchar(15) NULL,
Sleva numeric(7,2) NULL,
CisloPojist nvarchar(5) NULL,
Poznamka nvarchar(4000) NULL,
Dodavatel BIT NULL,
Odberatel BIT NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID),
CONSTRAINT FK_Org_CisloPojist FOREIGN KEY (CisloPojist)
  REFERENCES TabPojistovny(Cislo)
)
```

Obr. 25: Definice tabulky TabOrganizace (vlastní zpracování)

Tabulka **TabOrganizace** slouží k evidenci veškerých informací, včetně kontaktů o zákaznících a dodavatelích. Umožňuje evidovat poskytované slevy, či stav, ve kterém se daná organizace nachází ve vztahu k oční optice.

```
CREATE TABLE dbo.TabLekar
(
ID INT IDENTITY(1, 1),
ICZ nvarchar(30) NULL UNIQUE,
Jmeno nvarchar(15) NULL,
Prijmeni nvarchar(30) NULL,
Ordinace nvarchar(45) NULL,
Telefon nvarchar(13) NULL,
Email nvarchar(100) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
Poznamka nvarchar(4000) NULL,
PRIMARY KEY(ID),
)
```

Obr. 26: Definice tabulky TabLekar (vlastní zpracování)

Tabulka **TabLekar** slouží k evidenci stěžejních informací o očních lékařích a jejich ordinacích. Oční lékaři jsou přiřazováni k vydaným receptům na objednání dioptrických skel.

```
CREATE TABLE dbo.TabRecepty
(
ID INT IDENTITY(1, 1),
Davka int not null,
Cislo int NOT NULL UNIQUE,
IDOrg INT NOT NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
Diag nvarchar(5) NOT NULL,
PTZ nvarchar(30) NOT NULL,
ICZLekar nvarchar(30) NULL,
Sleva numeric(19,6) NULL,
KodSklaPojist int NOT NULL,
CisloPojist nvarchar(5) NULL,
Poznamka nvarchar(4000) NULL,
PRIMARY KEY(ID),
CONSTRAINT FK_Recepty_IDOrg FOREIGN KEY (IDOrg)
  REFERENCES TabOrganizace(ID),
CONSTRAINT FK_Recepty_ICZLekar FOREIGN KEY (ICZLekar)
  REFERENCES TabLekar(ICZ),
CONSTRAINT FK_Recepty_CisloPojist FOREIGN KEY (CisloPojist)
  REFERENCES TabPojistovny(Cislo)
)
```

Obr. 27: Definice tabulky TabRecepty (vlastní zpracování)

Tabulka **TabRecepty** slouží k evidenci vystavených receptů, které mohou být párovány k pohybovým dokladům a jsou nezbytně nutné při proplácení brýlí a dioptrických sklech pojišťovnou. Recept se vztahuje k zákazníkovi, lékaři, který jej vydal a pojišťovně, která jej proplácí.

```
CREATE TABLE dbo.TabZakazka
(
ID INT IDENTITY(1, 1),
CisZak INT UNIQUE NOT NULL,
Stav INT NULL,
DatSplneni DATETIME NOT NULL,
ZpusobVzniku int,
Autor nvarchar(15) NULL,
Telefon nvarchar(13) NULL,
Email nvarchar(100) NULL,
PozadTerm datetime NULL,
Meril nvarchar(15) NULL,
OcekavanyTerm datetime NULL,
Zaloha numeric(19,6) NULL,
ReceptCis int NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
b_pSPH NVARCHAR(5) NULL,
b_pCYL NVARCHAR(5) NULL,
b_pAX INT NULL,
b_pPD NUMERIC(7,2) NULL,
b_pADD1 NUMERIC(7,2) NULL,
b_pVyska NUMERIC(7,2) NULL,
b_pPrs NUMERIC(7,2) NULL,
b_pBas INT NULL,
b_lSPH NVARCHAR(5) NULL,
b_lCYL NVARCHAR(5) NULL,
b_lAX INT NULL,
b_lPD NUMERIC(7,2) NULL,
b_lADD1 NUMERIC(7,2) NULL,
b_lVyska NUMERIC(7,2) NULL,
b_lPrs NUMERIC(7,2) NULL,
b_lBas INT NULL,
d_pSPH NVARCHAR(5) NULL,
d_pCYL NVARCHAR(5) NULL,
d_pAX INT NULL,
d_pPD NUMERIC(7,2) NULL,
d_pVyska NUMERIC(7,2) NULL,
d_lSPH NVARCHAR(5) NULL,
d_lCYL NVARCHAR(5) NULL,
d_lAX INT NULL,
d_lPD NUMERIC(7,2) NULL,
d_lVyska NUMERIC(7,2) NULL,
PRIMARY KEY(ID),
CONSTRAINT FK_Zakazka_ReceptCis FOREIGN KEY (ReceptCis)
  REFERENCES TabRecepty(Cislo)
)
```

Obr. 28: Definice tabulky TabZakazka (vlastní zpracování)

Tabulka **TabZakazka** je určena pro evidenci vzniklých zakázek. V zakázce je možné evidovat její stav, způsob, jakým byla zakázka pořízena, a termíny, ke kterým má být vyřízena, či je očekáváno její vyřízení. V zakázce je možné evidovat informace z měření zraku očními lékaři. Na základě vyplnění požadovaných hodnot na dioptrické čočky bude možné automaticky vygenerovat objednávkový doklad s dohledaným zbožím dle zadaných údajů z měření. Na zakázku bude možné navázat recept.

```
CREATE TABLE dbo.TabDoklad
(
ID INT IDENTITY(1, 1),
Autor nvarchar(15) NULL,
IDZak INT NULL,
IDOrg int NULL,
Dod_Firma nvarchar(45) NULL,
Dod_UliceCP nvarchar(50) NULL,
Dod_Mesto nvarchar(30) NULL,
Dod_PSC nvarchar(6) NULL,
Dod_Telefon nvarchar(13) NULL,
Mena nvarchar(10) NULL,
CCbezDPH numeric(19,6) NULL,
CCsDPH numeric(19,6) NULL,
DPH numeric(19,6) NULL,
SazbaDPH numeric(19,6) NULL,
Sleva numeric(19,6) NULL,
CCsDPHpoSleve numeric(19,6) NULL,
Druh int NOT NULL,
ZpusobPlatby int NOT NULL,
ZpusobDoruceni int NOT NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
Poznamka nvarchar(4000) NULL,
PRIMARY KEY(ID),
CONSTRAINT FK_Doklad_IDOrg FOREIGN KEY (IDOrg)
  REFERENCES TabOrganizace(ID),
CONSTRAINT FK_Doklad_IDZak FOREIGN KEY (IDZak)
  REFERENCES TabZakazka(ID)
)
```

Obr. 29: Definice tabulky TabDoklad (vlastní zpracování)

V tabulce **TabDoklad** budou evidovány hlavičky pohybový dokladů, rozlišené pomocí atributu Druh. Tento atribut rozliší, zda se jedná například o doklady typu příjem, výdej, faktura a jiné. V této tabulce je evidována celková cena za zboží umístěné na dokladu, informace o způsobu platby a doručení zároveň s místem, kam má zboží být doručeno, pokud se nejedná o osobní předání. Doklad je možné propojit s organizací dodavatelskou, tak i odběratelskou (zákazníkem).

```
CREATE TABLE dbo.TabPolDok
(
ID INT IDENTITY(1, 1),
Autor nvarchar(15) NULL,
IDDoklad INT NOT NULL,
IDSkladZbo INT NOT NULL,
Mnozstvi numeric(19,6) NULL,
CenaBezDPH numeric(19,6) NULL,
CenaSDPH numeric(19,6) NULL,
DPH numeric(19,6) NULL,
SazbaDPH numeric(19,6) NULL,
Poznamka nvarchar(4000) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID),
CONSTRAINT FK_PolDok_IDDoklad FOREIGN KEY (IDDoklad)
  REFERENCES TabDoklad(ID),
CONSTRAINT FK_PolDok_IDSkladZbo FOREIGN KEY (IDSkladZbo)
  REFERENCES TabSkladZbozi(ID)
)
```

Obr. 30: Definice tabulky TabPolDok (vlastní zpracování)

Tabulka **TabPolDok,** slouží jako vazební tabulka mezi pohybovým dokladem a zbožím. Položka dokladu nese informaci o zboží a dokladum na který patří. Bude zde dotažená aktuální cena, prodejní cena z ceníku k dané položce zboží a množství, se kterým je přijímáno nebo vydáváno.

```
CREATE TABLE dbo.TabPokladna
(
ID INT IDENTITY(1, 1),
Castka numeric(19,6) NOT NULL,
Poznamka nvarchar(4000) NULL,
Typ int NOT NULL,
Autor nvarchar(15) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID)
)
```

Obr. 31: Definice tabulky TabPokladna (vlastní zpracování)

Tabulka **TabPokladna** slouží pro evidenci finančního stavu pokladny. Na začátku dne zde bude zaevidován peněžní obnos, který se v pokladně nachází, a vyřízené pohybové doklady zde budou vytvářet záznamy, které finanční stav pokladny poníží či navýší za účelem kontroly finančního stavu pokladny na konci určitého období. V budoucnu se očekává rozšíření využití této tabulky.

```
CREATE TABLE dbo.TabSplatky
(
ID INT IDENTITY(1, 1),
IDOrg int NOT NULL,
PujcenaCastka numeric(19,6) NOT NULL,
SplatitDO datetime NULL,
Poznamka nvarchar(4000) NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID),
CONSTRAINT FK_TabSplatky_IDOrg FOREIGN KEY (IDOrg)
  REFERENCES TabOrganizace(ID)
)
```

Obr. 32: Definice tabulky TabSplatky (vlastní zpracování)

Tabulka **TabSplatky** bude využita pro evidenci poskytnutých půjček zákazníkům.

```
CREATE TABLE dbo.TabUziv
(
ID INT IDENTITY(1, 1),
LoginID nvarchar(10) UNIQUE,
Heslo nvarchar(55),
Jmeno nvarchar(15) NULL,
Prijmeni nvarchar(30) NULL,
RodneCislo nvarchar(11) NULL,
UliceCP nvarchar(50) NULL,
Mesto nvarchar(30) NULL,
PSC nvarchar(6) NULL,
Telefon nvarchar(13) NULL,
Email nvarchar(100) NULL,
PRIMARY KEY(ID)
)
```

Obr. 33: Definice tabulky TabUziv (vlastní zpracování)

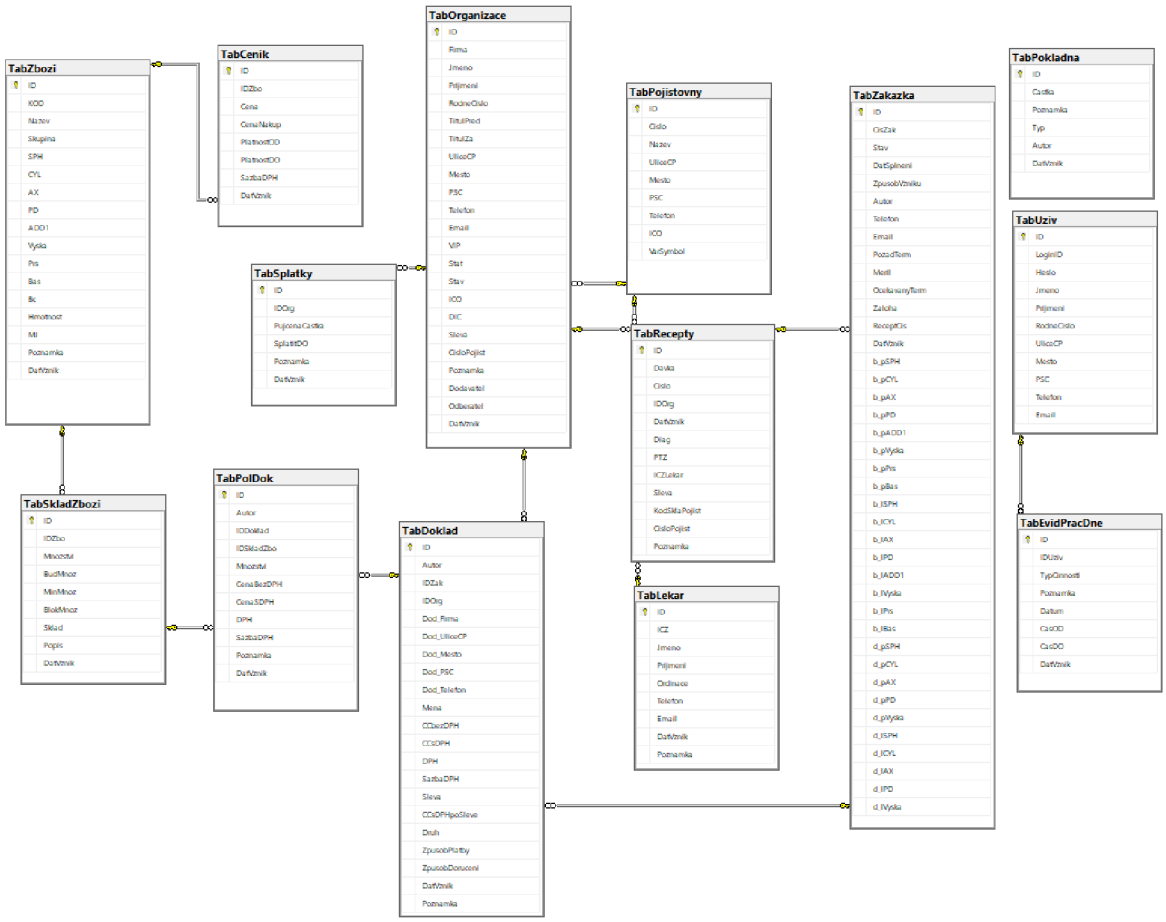
Tabulka **TabUziv** nachází vícenásobné využití v informačním systému. V prvotním kroku spouštění IS slouží pro ověření a přihlášení zaměstnance do systému. Ukládá přihlašovací údaje pro zabezpečení práce zaměstnanců a neoprávněného vniku do informačního systému. Využití však najde i v případě evidence zakladatelů záznamů v tabulkách pokladny a zakázek.

```
CREATE TABLE dbo.TabEvidPracDne
(
ID INT IDENTITY(1, 1),
IDUziv int NOT NULL,
TypCinnosti nvarchar(15) NULL,
Poznamka nvarchar(255) NULL,
Datum date NOT NULL,
CasOD time NULL,
CasDO time NULL,
DatVznik DATETIME NOT NULL DEFAULT GETDATE(),
PRIMARY KEY(ID),
CONSTRAINT FK_EvidPracDne_IDUziv FOREIGN KEY (IDUziv)
  REFERENCES TabUziv(ID)
)
```

Obr. 34: Definice tabulky TabEvidPracDne (vlastní zpracování)

Tabulka **TabEvidPracDne** umožní zaměstnancům evidovat svůj pracovní den, kde si mohou vykazovat odvedenou práci. Bude sloužit ke kontrole docházky, a k usnadnění tvorby zaměstnaneckých mezd.

V následujícím obrázku je vyjádřena kompletní struktura relačních tabulek databáze Optik s jejich vzájemnými vazbami.



Obr. 35: Kompletní schéma tabulek databáze Optik (vlastní zpracování)

Některé z tabulek nemají vazbu na jiné tabulky. Je to dané tím, že tabulky jsou v danou chvíli považovány jako informativní pro pouhé rozšíření funkcionality. V budoucnu se očekává rozšiřování informačního systému o další funkce a struktury, ve kterých tyto tabulky mohou hrát důležitou roli.

S procedurami typu „trigger“ nad tabulkami se prozatím nepočítá, všechna omezení tabulek jsou již nadefinována a zbylá kontrola a úprava dat bude řešena na úrovni předzpracování dat v IS. Zároveň nebudou vytvářeny definované pohledy typu view.

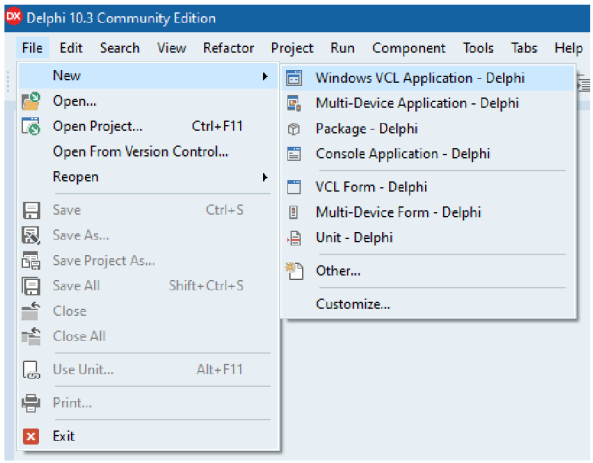
Logika zobrazování dat, propojování tabulek a vybírání souvisejících atributů bude řešena ze strany aplikace IS.

3.3 Tvorba načítání a úvodní obrazovky IS

V této podkapitole se zaměřím na vytvoření nového projektu a úvodní obrazovky s načítáním informačního systému, pro který jsem zvolil jméno: *„Optician Solution“*.

3.3.1 Založení nového projektu

Nadchází vytvoření nového projektu v prostředí Delphi. Nový projekt představuje návrh informačního systému. V Delphi vyberu novou Windows VCL aplikaci, pro které mi Delphi automaticky založí projekt a vygeneruje soubory, které s ním souvisí. Dále je vygenerován nový formulář s jednotkou pro umístění vlastního zdrojového kódu.



Obr. 36: Delphi založení nového projektu (vlastní zpracování)

3.3.2 Návrh spouštění a přihlašování do IS

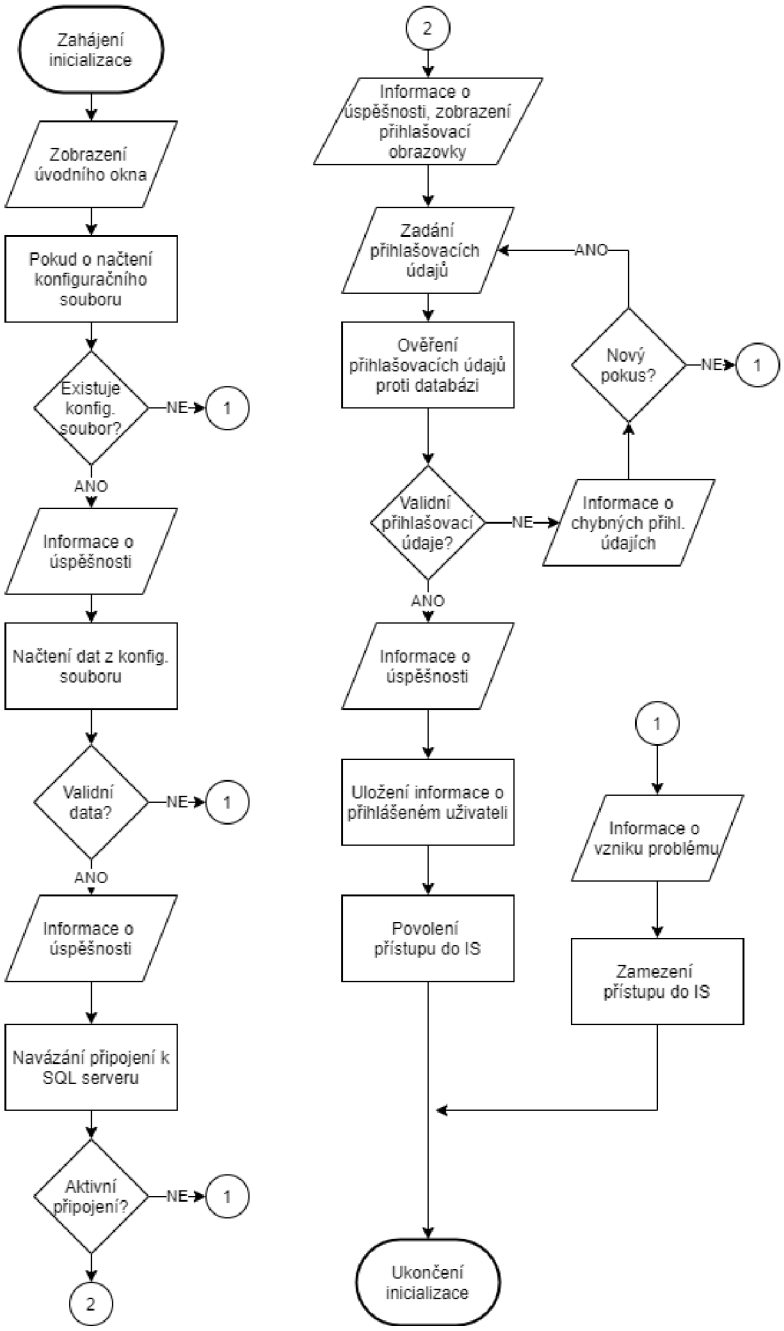
Na samém začátku tvorby aplikace je potřeba stanovit, co se má stát po jejím spuštění, jaké kroky mají být provedené, co je třeba načíst a jak takové načítání má vypadat.

Proces spouštění informačního systému může být zdlouhavý a uživatel musí být informován o stavu jeho spouštění.

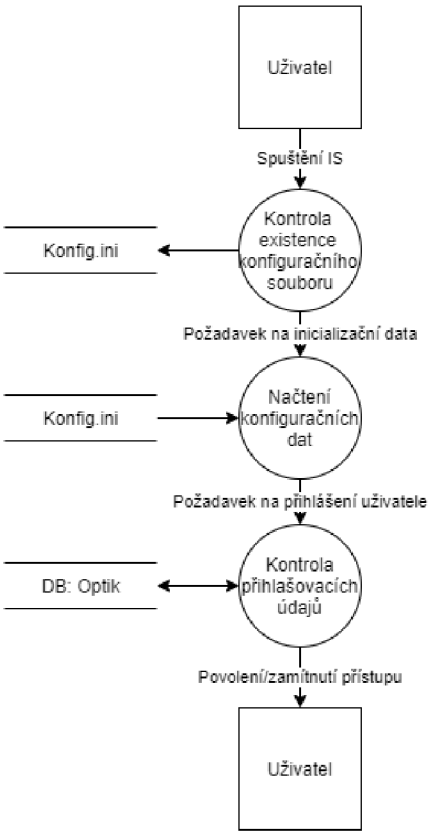
Při spuštění informačního systému je třeba načíst konfigurační soubor, ve kterém budou uloženy informace o adrese na SQL server, ke kterému se má připojit, o databázi, ze které budou informace čerpány a jiné hodnoty, které je třeba nadefinovat pro správný chod informačního systému. Po načtení konfiguračního souboru je potřeba navázat připojení k SQL serveru a zkontrolovat, jestli je připojení aktivní.

Na základě analýzy jsem navrhl tabulku uživatelů, kteří budou mít přístup do informačního systému. Pro přístup k informačnímu systému se budou muset při jeho spouštění identifikovat pomocí přístupového jména a hesla, jež budou prozatím uloženy v databázové tabulce. Bez správné kombinace přístupových údajů bude uživateli přístup do IS zamítnut.

Celý proces spouštění a přihlašování do IS je vyjádřen jako proces inicializace ve vývojovém diagramu a diagramu toku dat na obrázcích níže.



Obr. 37: Proces inicializace IS – vývojový diagram (vlastní zpracování)



Obr. 38: Proces inicializace IS – diagram toku dat (vlastní zpracování)

3.3.3 Načítání konfiguračního souboru

Při spuštění aplikace je potřeba načíst konfigurační soubor. Pro zachování zpětně kompatibility a jednoduchosti úprav či načítání souboru, jsem zvolil práci s textovým INI souborem, který má udanou vlastní strukturu.

Pro vyhledávání a načítání informací ze souboru jsem si vytvořil funkci se vstupními parametry, které udávají, v jaké sekci inicializačního souboru chci hledat, a jaký parametr zde chci najít. Dalšími vstupy funkce je cesta k souboru a jeho název. Funkce vrací textovou hodnotu příslušného parametru.

```
function TINIItem.NactiINI(Kde,Co: String;aFI: TStrIniFile;N_Aplikace: String):Str
var
  NH  : String;
  SLB,
  SLE : TStringList;
  PP12: Pole2;
  i,j : integer;
Begin
  NH := 'NIC';
  SLB := TStringList.Create;
  SLE := TStringList.Create;
  try
    aFI.LoadFromFile(ExtractFilePath(ParamStr(0))+N_Aplikace+'.ini');
    if aFI.SectionExists(Kde) then
      Begin
        aFI.GetStrings(SLB);
        j:=0;
        for I := 0 to SLB.Count-1 do
          begin
            if (('['+Kde+']')= SLB.Strings[I])or(j>0) then
              begin
                PP12 := RozdelZaznam(SLB.Strings[I]);
                if Co=PP12.S1 then
                  NH := PP12.S2;
                Inc(j);
              end;
            if J>6 then Exit;
          end;
      End;
  finally
    SLB.Free;
    SLE.Free;
    Result := NH;
  end;
End;
```

Lister - [d:\Delphi_projekty\Optik\Win32\Debug\Data\Konfig.INI]

Soubor Upravit Možnosti Kódování Nápověda

```
[Nastaveni]
Server=localhost\test
Databaze=Optik
```

Obr. 39: Načítání z konfiguračního souboru – funkce (vlastní zpracování)

3.3.4 Navázání připojení k SQL serveru

Pro připojení k Microsoft SQL Serveru jsem využil standartních komponent ADO z knihoven Data.Win.ADODB a Data.DB. Vytvořil jsem vlastní programovou jednotku s funkcemi pro komunikaci mezi aplikací a SQL serverem. Vytvořená funkce **ConnectToSQL** slouží k navázání připojení k SQL serveru. Adresa SQL serveru a jméno databáze tvoří její vstupní proměnné na základě, kterých se funkce pokusí připojit k SQL serveru. V případě úspěchu naplní nadefinovanou globální proměnou připojením. V případě neúspěchu vrátí chybu.

```
procedure ConnectToSQL(server,dbname:string);
var ADOConn  : TADOConnection;
    myFile : TextFile;
    text   : string;
begin
  { Create an ADO connection. }
  ADOConn := TADOConnection.Create(nil);
  { Set up the connection string. }
  ADOConn.ConnectionString:=FORMAT('Provider=SQLOLEDB.1;Integrated Security=SSPI;'#13+
                                   'Persist Security Info=False;Initial Catalog=%0:s;Data Source=%1:s',[dbname,server]);
  { Disable login prompt. }
  ADOConn.LoginPrompt := False;

  try
    ADOConn.Connected := True;
  except
    on e: EADOError do
    begin
      MessageDlg('Error while connecting', mtError,
                 [mbOK], 0);

      Exit;
    end;
  end;
  Con:= ADOConn;
end;
```

Obr. 40: Připojení k SQL serveru – procedura (vlastní zpracování)

3.3.5 Funkce pro vyřizování SQL dotazů

Při spouštění informačního systému probíhá ověření přihlašovacích údajů zadaných v aplikaci s hodnotami uloženými v tabulce databáze SQL. Zároveň je potřeba, aby IS uměl posílat dotazy na SQL server a pracoval s nimi. Vytvořil jsem proto funkci OpenSQL a proceduru ExecSQL.

Procedura ExecSQL slouží k odeslání SQL dotazu neboli skriptu na SQL Server bez požadavku na návratovou hodnotu. Vstupní proměnnou této procedury je textový řetězec s SQL dotazem, který má být vykonán. Procedura využívá stejné knihovny jako funkce navázání připojení.

```
procedure ExecSQL(SQL:string);
var ADOQuery : TADOQuery;
begin
  ADOQuery := TADOQuery.Create(nil);

  ADOQuery.Connection := Con;
  ADOQuery.SQL.Text:=SQL;
  Try
    ADOQuery.ExecSQL;
  except on E: EADOError do
    begin
      MessageDlg('Error while doing query',mtError,[mbOK],0);
      exit;
    end;
  End;

end;
```

Obr. 41: Odesílání SQL dotazu bez očekávání výsledku – procedura (vlastní zpracování)

Funkce OpenSQL slouží velmi obdobně jako zmíněná procedura ExecSQL s výjimkou toho, že očekává návratovou hodnotu ve formě výběru hodnot z SQL pomocí příkazu SELECT. Funkce vrací pole hodnot převzatých vyhodnocených SQL serverem.

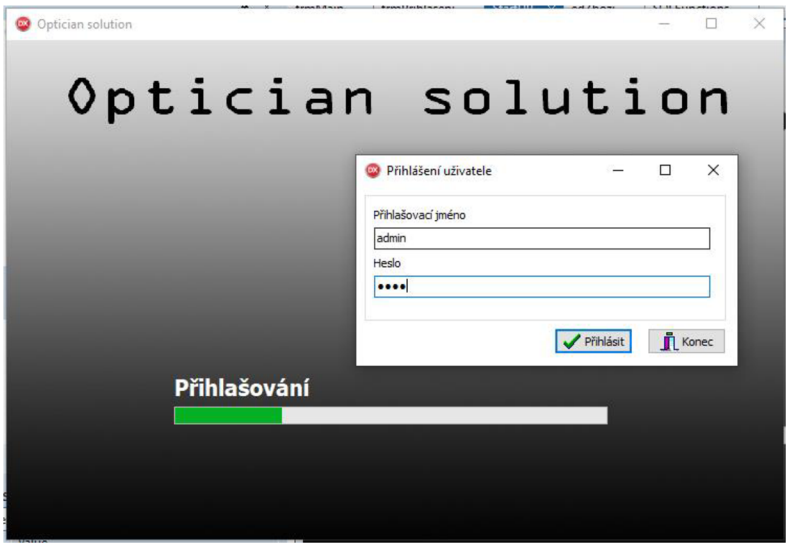
```
function openSQL(SQL:string):TADOQuery;
var ADOQuery : TADOQuery;
begin
  ADOQuery := TADOQuery.Create(nil);
  ADOQuery.Connection := Con;
  ADOQuery.SQL.Add(SQL);
  try
    ADOQuery.Active := True;
  except
    on e: EADOError do
    begin
      MessageDlg('Error while doing query', mtError,
                 [mbOK], 0);
        Exit;
    end;
  end;

  ADOQuery.Open;
  ADOQuery.First;

  result:=ADOQuery;
end;
```

Obr. 42: Odesílání SQL dotazu návratovou hodnotou – funkce (vlastní zpracování)

3.3.6 Design spouštěcí obrazovky IS

Již jsem zmínil, že spouštění informačního systému musí být doprovázeno úvodní obrazovkou, aby byl uživatel v povědomí, že se IS spouští. Zároveň bylo potřeba zajistit přihlášení do IS během spouštění.

Pro úvodní obrazovku jsem využil jednoduchého formuláře s načítací lištou, která v doprovodu s textovým polem informuje o stavu načítání informačního systému. Zároveň jsem vytvořil malý formulář pro zadání přihlašovacích údajů.



Obr. 43: Design úvodního okna s přihlašovacím formulářem (vlastní zpracování)

Pole **heslo** je pro zajištění bezpečnosti překryto maskou znaků, aby nebyl vidět zadávaný text.

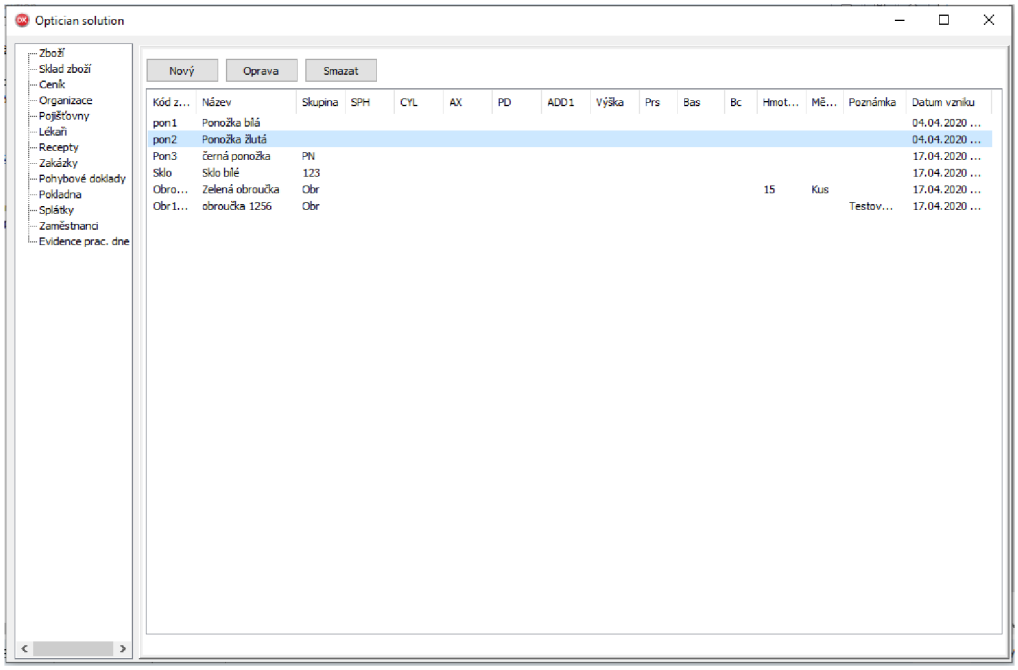
3.4 Tvorba rozhraní a funkcionality IS

V této podkapitole se budu zabývat tvorbou nejdůležitější části této práce, kterou je logika zobrazování a funkcionalita informačního systému, včetně jeho designové stránky.

3.4.1 Rozhraní IS

Po úspěšném přihlášení do IS se zobrazí uživatelské rozhraní. Toto rozhraní by mělo být velmi jednoduché a uživatelsky přívětivé. Vzhled a rozmístění komponent musí být vytvořen tak, aby vše důležité bylo snadno dostupné a se systémem se dalo snadno, rychle, a efektivně pracovat. V tomto ohledu jsem se inspiroval z informačního systému Helios Orange, jež jsem ve své práci analyzoval. Umístění stromu přehledů v levé části informačního systému se zobrazovanými přehledy ve zbytku obrazovky je velmi vhodné. Uživatel tak jednoduše může přepínat mezi přehledy v průběhu jejich prohlížení. Toto řešení mi přijde mnohem přívětivější než řešení ostatních analyzovaných systémů, kde se využívalo převážně formy prokliků tlačítky.

Navrhl jsem tedy uživatelské rozhraní tak, že na levé straně IS se nachází strom s dostupnými přehledy, jež představují zobrazované tabulky. Ve zbylé části rozhraní se nachází přehled s tlačítky, jež zajišťují operace s daty v tabulkách přehledů.



Obr. 44: Uživatelské rozhraní hlavního okna IS (vlastní zpracování)

3.4.2 Zobrazení dat v přehledech

Při výběru položky ze stromu se zobrazí přehled, který zvolená položka představuje. Jako zobrazovací komponentu jsem využil ListView, a to z důvodu velmi přehledného a jednoduchého vzhledu. Tato komponenta umožňuje označení jednoho či více řádků. Díky tomu lze nad označenými řádky provádět operace, které nadefinuji. Zároveň lze nadefinovat akci, která se má stát při stisknutí hlavičky jednotlivých sloupců přehledu. Díky tomu lze jednoduše nadefinovat řazení přehledu pomocí zvoleného sloupce. Sloupce mají uživatelsky nastavitelnou šířku, kterou lze nastavit i programově. Je možné tak ukládat šířky sloupců a znovu je nastavovat dle uživatelských preferencí. Komponenta rovněž umožňuje uživatelsky seřadit, respektive měnit pořadí sloupců.

Pro naplnění přehledů daty z jednotlivých tabulek jsem vytvořil proceduru, které je ve formě vstupních parametrů předán název komponenty (listview), definice zobrazovaných

dat pomocí příkazu SQL SELECT, filtrovací podmínku pro omezení zobrazovaných dat, pořadové číslo sloupce, dle kterého má být přehled řazený (standardně se řadí dle hodnoty ve sloupci ID) a posledním parametrem je počet zobrazovaných sloupců.

```
procedure TfrMain.naplnPrehled(lv: TListView; select,where: string; OrderBy, pocSl: integer);
var sql:string;
    li:TlistItem;
    i:integer;
begin
  sql:=select;
  IF where <>'' then sql:=sql+' WHERE '+where;
  IF OrderBy > 0 then sql:=sql+' ORDER BY '+IntToStr(OrderBy);

  PoslSelect:=sql;
  lv.Clear;
  with SQLFunctions.OpenSQL(sql) do
  while NOT(EOF) do
  begin
    li := lv.Items.Add;
    li.Caption := Fields[0].AsString;
    for I:=2 to pocSl do
      li.SubItems.Add(Fields[i-1].AsString);
    next;
  end;
end;
```

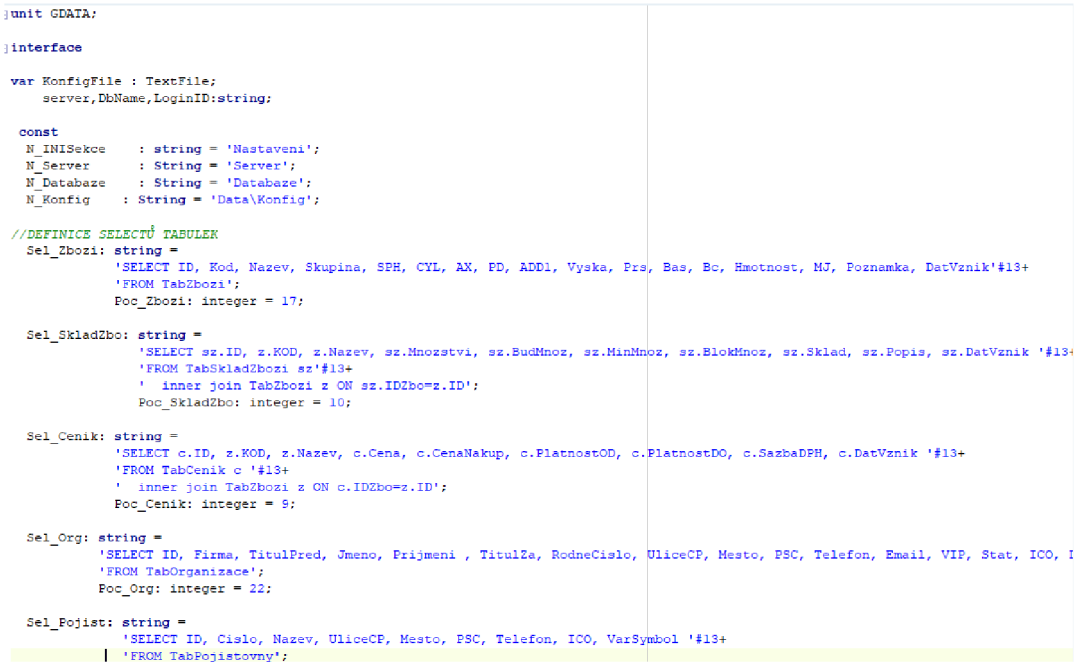
Obr. 45: Naplnění přehledů daty z tabulek – procedura (vlastní zpracování)

Procedura pro naplnění přehledu je volána v případě vynuceného obnovení přehledu, které nastává po dokončení akcí přehledu, a také volbou přehledu ve stromu přehledů.

```
procedure TfrMain.Refresh;
var i:integer;
begin
  for i := 0 to TreeView1.Items.count-1 do
    if (TreeView1.Items.Item[i].Selected) then
      case i of
      0:begin
          PageControl1.ActivePageIndex := 1;
          naplnPrehled(LvTabZbozi, GDATA.Sel_Zbozi,'', 0, GDATA.Poc_Zbozi);
        end;
      1:
        begin
          PageControl1.ActivePageIndex := 2;
          naplnPrehled(LvTabSkladZbo, GDATA.Sel_SkladZbo,'', 0, GDATA.Poc_SkladZbo);
        end;
      2:
        begin
          PageControl1.ActivePageIndex := 3;
          naplnPrehled(LvTabCenik, GDATA.Sel_Cenik,'', 0, GDATA.Poc_Cenik);
        end;
```

Obr. 46: Přepínání přehledů a výběr datových zdrojů k naplnění – procedura (vlastní zpracování)

Definice příkazů SELECT, jež definují atributy z tabulek, které mají být zobrazeny, jsou umístěny jako konstanty ve zvláštní programové jednotce GDATA (globální data), která slouží primárně pro uchování hodnot v proměnných a jejich přenos napříč mezi formuláři.

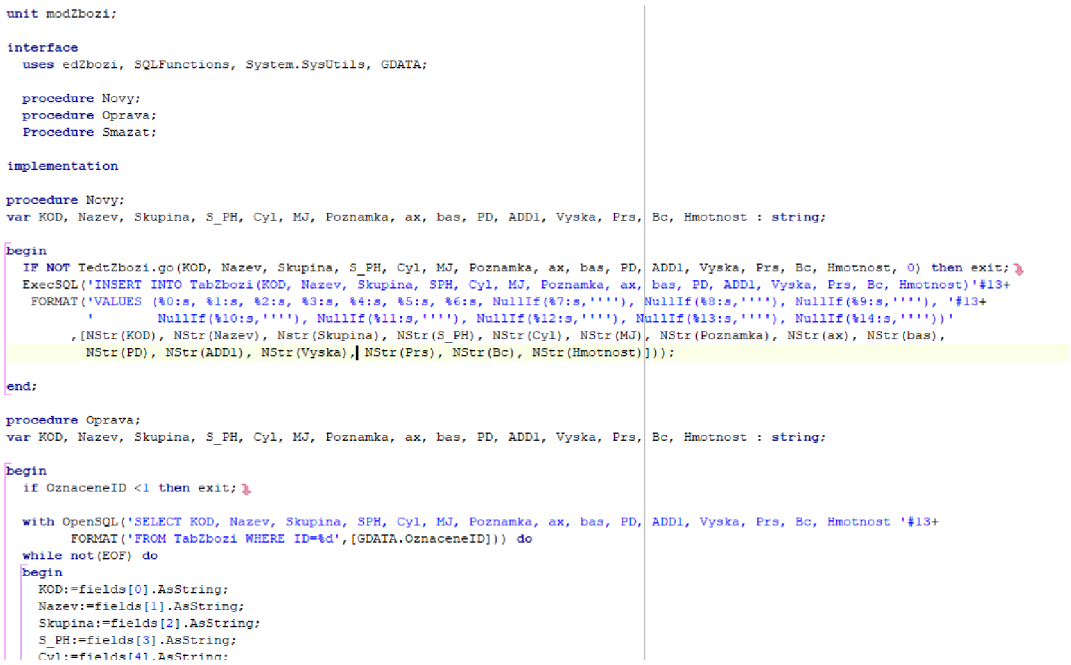


```
unit GDATA;

interface

var KonfigFile : TextFile;
    server,DbName,LoginID:string;

 const
  N_INISekce    : string = 'Nastaveni';
  N_Server      : String = 'Server';
  N_Databaze    : String = 'Databaze';
  N_Konfig    : String = 'Data\Konfig';

//DEFINICE SELECTŮ TABULEK
  Sel_Zbozi: string =
             'SELECT ID, Kod, Nazev, Skupina, SPH, CYL, AX, PD, ADD1, Vyska, Prs, Bas, Bc, Hmotnost, MJ, Poznamka, DatVznik'#13+
             'FROM TabZbozi';
             Poc_Zbozi: integer = 17;

  Sel_SkladZbo: string =
                'SELECT sz.ID, z.KOD, z.Nazev, sz.Mnozstvi, sz.BudMnoz, sz.MinMnoz, sz.BlokMnoz, sz.Sklad, sz.Popis, sz.DatVznik '#13+
                'FROM TabSkladZbozi sz'#13+
                '  inner join TabZbozi z ON sz.IDZbo=z.ID';
                Poc_SkladZbo: integer = 10;

  Sel_Cenik: string =
             'SELECT c.ID, z.KOD, z.Nazev, c.Cena, c.CenaNakup, c.PlatnostOD, c.PlatnostDO, c.SazbaDPH, c.DatVznik '#13+
             'FROM TabCenik c '#13+
             '  inner join TabZbozi z ON c.IDZbo=z.ID';
             Poc_Cenik: integer = 9;

  Sel_Org: string =
           'SELECT ID, Firma, TitulPred, Jmeno, Prijmeni , TitulZa, RodneCislo, UliceCP, Mesto, PSC, Telefon, Email, VIP, Stat, ICO, I
           'FROM TabOrganizace';
           Poc_Org: integer = 22;

  Sel_Pojist: string =
              'SELECT ID, Cislo, Nazev, UliceCP, Mesto, PSC, Telefon, ICO, VarSymbol '#13+
              'FROM TabPojistovny';
```

Obr. 47: Globální programová jednotka s definicí konstant a proměnných (vlastní zpracování)

3.4.3 Datové moduly

Ke každému přehledu je vytvořen datový modul, který slouží pro definici akcí přehledu. V takových datových modulech je definováno, co se má vykonat v případě stisku tlačítek Nový, Oprava, Smazat a jiné v závislosti na funkcionalitě daného přehledu.



```
unit modZbozi;

interface
  uses edZbozi, SQLFunctions, System.SysUtils, GDATA;

  procedure Novy;
  procedure Oprava;
  Procedure Smazat;

implementation

procedure Novy;
var KOD, Nazev, Skupina, S_PH, Cyl, MJ, Poznamka, ax, bas, PD, ADD1, Vyska, Prs, Bc, Hmotnost : string;

begin
  IF NOT TedtZbozi.go(KOD, Nazev, Skupina, S_PH, Cyl, MJ, Poznamka, ax, bas, PD, ADD1, Vyska, Prs, Bc, Hmotnost, 0) then exit;
  ExecSQL('INSERT INTO TabZbozi(KOD, Nazev, Skupina, SPH, Cyl, MJ, Poznamka, ax, bas, PD, ADD1, Vyska, Prs, Bc, Hmotnost)'#13+
   FORMAT('VALUES (%0:s, %1:s, %2:s, %3:s, %4:s, %5:s, %6:s, NullIf(%7:s,''''), NullIf(%8:s,''''), NullIf(%9:s,''''), '#13+
          '       NullIf(%10:s,''''), NullIf(%11:s,''''), NullIf(%12:s,''''), NullIf(%13:s,''''), NullIf(%14:s,''''))'
        ,[NStr(KOD), NStr(Nazev), Nstr(Skupina), NStr(S_PH), NStr(Cyl), NStr(MJ), NStr(Poznamka), NStr(ax), NStr(bas),
          NStr(PD), NStr(ADD1), NStr(Vyska), NStr(Prs), NStr(Bc), NStr(Hmotnost)]));

end;

procedure Oprava;
var KOD, Nazev, Skupina, S_PH, Cyl, MJ, Poznamka, ax, bas, PD, ADD1, Vyska, Prs, Bc, Hmotnost : string;

begin
  if OznaceneID <1 then exit;

  with OpenSQL('SELECT KOD, Nazev, Skupina, SPH, Cyl, MJ, Poznamka, ax, bas, PD, ADD1, Vyska, Prs, Bc, Hmotnost '#13+
        FORMAT('FROM TabZbozi WHERE ID=%d',[GDATA.OznaceneID])) do
  while not(EOF) do
  begin
    KOD:=fields[0].AsString;
    Nazev:=fields[1].AsString;
    Skupina:=fields[2].AsString;
    S_PH:=fields[3].AsString;
    Cyl:=fields[4].AsString;
```

Obr. 48: Programová jednotka datového modulu přehledu (vlastní zpracování)

Tlačítka však stále zůstávají nadefinovány v hlavním uživatelském rozhraní s přehledy. Mají však přiřazené funkce ze souvisejících datových modulů. Toto rozdělení obsluhy akcí přehledů do vlastních datových modulů usnadňuje budoucí rozšiřování funkcionality, kdy programová jednotka hlavní obrazovky s uživatelským rozhraním zůstává stále poměrně jednoduchá.

Veškeré přehledy mají tři shodně fungující akce Nový, Oprava a Smazat. Jednotlivé akce volají uložené procedury, ve kterých je využito vytvořených funkcí OpenSQL a ExecSQL pro odesílání dotazů na SQL server.

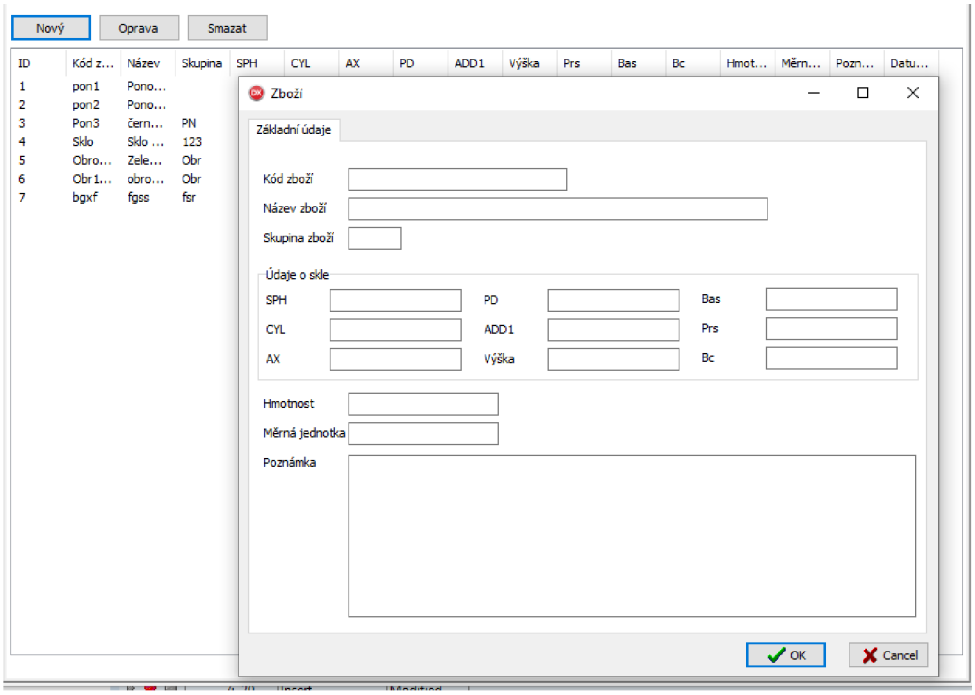
Pro práci se stávajícími záznamy, jež jsou nejčastěji funkce pro opravu či smazání stávajícího záznamu, jsem vytvořil funkci **GetSelectedID**. Vstupním parametrem této funkce je přehled, pro který má být zjištěné ID záznamu, který je v danou chvíli označen. Každá komponenta přehledu ListView má zobrazené ID záznamu v prvním sloupci, který je pro uživatele skrytý. Funkce zjistí tuto hodnotu a uloží ji do celočíselné globální proměnné **OznaceneID**.

```
function TfrMain.GetSelectedID(lv:TlistView): integer;
var i:integer;
begin
  for i:= 0 to lv.Items.Count-1 do
    if lv.Items[i].Selected then
    begin
      OznaceneID:=StrToInt(lv.Items[i].Caption);
      break;
    end;
    result:=OznaceneID;
end;
```

Obr. 49: Vyhledání ID označeného záznamu – funkce (vlastní zpracování)

3.4.4 Editory přehledů

Každý z přehledů má vlastní formulář pro zadávání hodnot, které jsou propisovány do tabulek SQL serveru. Formuláře neboli editory jsou volány jednotlivými procedurami datových modulů přiřazených danému přehledu.



Obr. 50: Design zadávacího formuláře přehledu (vlastní zpracování)

Zobrazení zadávacího formuláře je formou vyskakovacího okna. Formulář lze posouvat po celé obrazovce, a tím je možné nahlédnout do přehledu, který se nachází v pozadí v průběhu zadávání hodnot do formuláře. Nacházejí se v něm editační pole, které odpovídají atributům tabulky zobrazeného přehledu. Některé zadávací formuláře však spojují data z více tabulek zároveň, a tedy editují data hned v několika tabulkách.

Všechny formuláře mají na tlačítko OK před tím, než je provedené potvrzení změn, jde o validační funkci. Do této funkce je možné zadat kontrolní omezení, tím jsem zajistil rychlejší a vhodnější způsob ověření zadaných informací, než v případě použití SQL spouští – trigger.

Zadávací pole ve formuláři umožňují zadat informace v datových typech, jimiž jsou definovány atributy tabulek.

3.5 SWOT analýza navrženého řešení

V kapitole analýzy současného stavu jsem vytvořil SWOT analýzu na IS typu tužka a papír. Tedy způsob řízení podnikových procesů metodou ručního skladování dokumentů a zapisování všech informací na stránky papíru.

Nyní provedu SWOT analýzu nad řešením, které jsem vytvořil. Při porovnání těchto dvou SWOT analýz vyplynou výhody a nevýhody jednotlivých řešení.

Tab. 3: SWOT analýza navrženého IS Optician Solution (vlastní zpracování)

SILNÉ STRÁNKY	SLABÉ STRÁNKY
• Moderní vývojové prostředí • Rychlý a intuitivní nástroj pro řízení firemních procesů • Snadné a přehledné zpracování informací a dokumentů • Široké možnosti pro rozšiřování funkcionality • Propojení s dalšími IS • Více souběžných instancí IS • Podniková data na jednom uložišti	• Nutnost patřičného HW vybavení • Nemožnost uživatelského rozšíření tabulek • Jednoduchá funkcionalita
PŘÍLEŽITOSTI	**HROZBY**
• Začínající oční optiky • Optiky bez digit. IS	• Rozšíření funkcionality konkurence • Vstup nového specializovaného IS na trh • Substituční řešení v podobě outsourcingových řešení

Mezi silné stránky jsem záměrně zařadil využité vývojové prostředí, které je moderní a stále se vyvíjí. Nabízí širokou škálu funkcí a udržuje se aktuální. Tyto aspekty jsou rozhodující při rozhodování do budoucna. Moderní a udržované vývojové prostředí umožní stálé rozvíjení informačního systému v moderním designu, rychlostí vyřizování požadavků, a v neposlední řadě funkcím, které umožní dalšího rozvoje funkcionality. Silnou stránkou navrženého IS je i jeho jednoduchost a rychlost. Jednoduchý a přehledný

design IS nebude výraznou bariérou ani pro uživatele, které nikdy předtím s digitálními IS nepracovali. Využití SQL serveru k ukládání informací umožňuje velmi rychlý, strukturovaný a snadný přístup k uloženým datům. Rovněž je toto řešení velmi universální, na úrovni SQL serveru umožní propojení databází s jinými IS. Značnou výhodou zvoleného řešení je bezpečnost, kdy jsou veškerá data uložena na jednom centrálním uložišti, které bude silně doporučené k dennímu zálohování.

Naopak slabou stránkou tohoto řešení je nutnost pořízení patřičného HW vybavení. Pro zajištění funkčnosti IS. Informační systém pro svoji funkčnost potřebuje vlastní SQL server a další počítače, na kterých poběží aplikace IS. Tato slabá stránka je zároveň i silnou stránkou, neboť umožňuje souběžné používání více instancí aplikace IS. Jako drobný nedostatek vnímám fixní struktury tabulek bez možnosti uživatelského rozšíření. Veškeré úpravy musí být realizovány programově. Poslední mnou identifikovanou slabou stránkou je prozatímní funkcionalita informačního systému, která je velmi zjednodušená proti konkurenčním řešení. Očekává se však její postupné rozšiřování ve spolupráci s očními optikami.

Příležitosti pro toto řešení jsou nově začínající podniky, které se právě rozhodují nad pořízením informačního systému. Díky modernímu rozhraní a velikému potenciálu pro rozšíření se jeví toto řešení jako vhodné. Navržené řešení se stává ideálním kandidátem pro stávající oční optiky, které doposud nevyužívají digitálních IS, zejména díky své jednoduchosti a intuitivnosti.

Naopak hrozby, které mohou zmařit úspěch daného řešení, jsou konkurenční IS, které se mohou i nadále rozšiřovat, či vstup zcela nového IS specializovaného na oční optiky.

3.6 Ekonomické zhodnocení práce

Finanční náklady na tvorbu informačního systému z této práce tvoří z počátku pouze náklady na čas jednoho či více programátorů. Vývojové prostředí Delphi je zdarma pro komerční účely do chvíle, než přesáhnou roční tržby z prodeje aplikací 5000 amerických dolarů. Zároveň produkt Microsoft SQL Server v omezené verzi Express je také zdarma s řadou omezení. Produkty jsou tak vhodné pro začínající programátory i začínající

organizace, které v počátcích svého působení negenerují velké objemy dat. Časová náročnost tvorby informačního systému je však vysoká. Na tvorbě tohoto informačního systému jsem strávil kolem 300 hodin. Vzhledem k tomu, že informační systém není řešení, které se jednou vytvoří a poté už pouze prodává, lze tak očekávat další investice času do dalšího rozvoje daného řešení, jež představuje další čas programátora. Odhadovaná cena za hodinu programátorské práce se může vyšplhat na částky i několika tisíc Kč. Ve své práci budu počítat s finančním ohodnocením programátora zaměstnaného na pracovní poměr. Pokud se jedná o programátora s běžným programátorským platem, pohybuje se jeho finanční ohodnocení za hodinu práce kolem **350 Kč**.

V tabulce níže se pokusím rozvést finanční náročnost pro vytvoření daného řešení.

Tab. 4: Finanční náročnost daného řešení (vlastní zpracování)

Položka	Časová náročnost [v hod.]	Náklad [v Kč]
Licence Microsoft SQL Server v edici Express	0	0
Licence Delphi - Community edition	0	0
Programátorské práce	300	105 000

Při uvážení, že tržby z prodeje informačního systému přesáhnou hodnotu 5000 amerických dolarů, pak bude programátor nucen zakoupit placenou licenci vývojového nástroje Delphi. Z licencí Delphi, které jsou dostupné, zcela postačí edice Professional, jejíž cena se pohybuje kolem 35 000 Kč s DPH.

Informační systém bude nabízen očním optikám za **jednorázovou cenu licence 20 000 Kč** s DPH, včetně **roční systémové podpory**, jejíž cena bude z počátku činit **13 000 Kč s DPH za rok**. Očekává se do budoucna další rozšiřování IS o nové moduly, jež budou rovněž zpoplatněny a nabízeny jako volitelně k zakoupení. Jejich cena bude individuální

v závislosti na složitosti a přínosu daného modulu. Moduly budou mít rovněž zpoplatněnu roční systémovou podporu.

S rostoucím počtem dat, a tím i velikosti databáze se dostanou oční optiky do stavu, kdy budou nuceny pořídit placenou licenci k produktu Microsoft SQL server bez omezení. Vznikne jim tak jednorázový náklad na pořízení standartní licence Microsoft SQL Server, jejíž cena se pohybuje kolem 24 500 Kč s DPH. Tato licence je trvalá.

3.6.1 Přínosy práce

Vytvářený informační systém Optician Solution je určen primárně pro pobočky očních optik. Pro tento segment firem je zcela uzpůsobena jeho struktura, funkcionalita i design.

Toto řešení přinese očním optikám:

- jednoduchý způsob obsluhy zákazníka
- vyřizování zákaznických požadavků
- evidenci zákazníků a dodavatelů
- evidenci pojišťoven
- práci s recepty
- skladové operace a evidenci zboží
- evidenci pohybových dokladů
- evidenci pokladny
- vytváření dávek pro pojišťovny
- evidenci pracovního dne zaměstnanců
- vedení ceníků (i historicky)
- evidenci lékařů
- evidenci splátek
- možnost propojení s jinými IS

Největším přínosem této práce pro oční optiky je evidence všech činností a aktivit na jednotném místě. Díky tomu lze snadno a rychle přistupovat k firemním datům z jediného místa, snadno se v nich orientovat a vyhledávat i historická data.

3.7 Výhledy do budoucna

Tvorba informačního systému je nikdy nekončící proces. Neustále bude potřeba rozšiřovat funkcionalitu, přizpůsobovat jej, aby zůstal využitelný a konkurenceschopný vůči stávajícím a novým řešením. Stejně tak bude pokračovat i má práce na tomto IS. Většina výhledů do budoucna vzešla při jeho samotné tvorbě, kdy v analýze s nimi nebylo počítáno.

K základním úpravám, jež chci v budoucnu dodělat, patří uživatelská možnost úpravy zobrazovaných dat z přehledů. Uživatelé by mohli vybírat, jaké sloupce a atributy ze zobrazované tabulky a tabulek souvisejících, mají být zobrazeny v přehledu.

Ve vztahu k úpravám přehledů vznikl další výhled do budoucna, kdy by měla být vytvořena možnost ukládání pořadí a šířek zobrazovaných sloupců dle uživatelských preferencí.

Přehledy podporují filtrování pomocí jednoho atributu, který si uživatel zvolí. Do budoucna bych chtěl přidat možnost rozšířeného filtrování přehledu, dle více atributů současně.

V navrženém IS jsou vytvářeny a uchovávány různé doklady. Mám v plánu rozšířit funkcionalitu IS o generování tiskových formulářů, pomocí kterých bude možné tisknout například objednávky, faktury a jiné doklady.

Plánuji rozšíření evidence pokladny a vytvoření modulu účetnictví. Vystavování faktur by pak mohlo podporovat EET.

Dalším výhledem do budoucna je vytvoření souhrnných přehledů, které budou složit pro reportování dosažených výsledků, například souhrn cen z objednávek zákazníků.

Další rozšíření a úpravy, které bude potřeba dodělat, vzejdou z požadavku případných zákazníků, jež si tento IS pořídí.

ZÁVĚR

Cílem této diplomové práce bylo vytvořit informační systém pro oční optiky, který má těmto uživatelům má výrazně usnadnit práci a snížit rizika ztráty přehledu o pohybech zboží a dokumentech. V první fázi jsem se věnoval teoretické problematice IS, databází, databázového jazyka SQL a programovacího jazyka Delphi. V druhé fázi práce jsem analyzoval trh digitálních technologií e-podnikání, firemní procesy, požadavky na IS a současná řešení.

Pomocí analýzy jsem zjistil, že informačních systémů zaměřených na tento obor podnikání není mnoho a většina jich je zastaralých a v dnešních podmínkách již těžko použitelných. V ČR se nachází mnoho poboček očních optik, které nevyužívají žádného digitálního IS. Právě proto se tento segment pro mě stal zajímavým. S pomocí zaměstnanců očních optik jsem analyzoval požadavky na IS a způsob, jak má být vytvořeno grafické rozhraní, aby optikům bylo co nejvíce přizpůsobeno.

V návrhové části jsem se věnoval vlastním návrhům daného řešení. Na základě provedených analýz jsem vytvořil databázi a její struktury. Propojil jsem databázi SQL s aplikačním rozhraním, vytvořeným v Delphi, které po designové stránce bylo vytvořeno na míru oboru očních optik. V dalších fázích návrhové části jsem vytvořil přehledy navazující na požadovanou funkcionalitu a vytvořil k nim jednoduché, a hlavně přehledné zadávací formuláře.

Celé řešení bylo vytvořeno tak, aby bylo modulární a snadno rozšiřitelné. Tvorba IS je nikdy nekončící proces, kdy se neustále rozšiřuje funkcionalita a optimalizují stávající procesy.

Navrhovaný IS systém se mi podařil dovést do funkčního stavu s vyřešením téměř všech analyzovaných požadavků. Nyní je IS nasazen v testovacím provozu jedné oční optiky, která se podílela na analýze požadovaných funkcí.

SEZNAM POUŽITÝCH ZDROJŮ

1. Co je to databázový systém? - Správa.sítě.eu. Správa sítě - slovník pojmů: správa sítě, zabezpečení sítě, outsourcing IT [online]. Praha: Aira GROUP, c2016 [cit. 2020-05-05]. Dostupné z: https://www.sprava-site.eu/databazovy-system/

2. KROENKE, David a David J. AUER. Databáze. Brno: Computer Press, 2015. ISBN 978-80-251-4352-0

3. POLÁŠEK, Mgr. Marek. Metadata a datové sklady. SystemOnLine.cz - ekonomické a informační systémy v praxi [online]. Brno: CCB spol. s r.o., c2001-2020 [cit. 2020-05-05]. Dostupné z: https://www.systemonline.cz/clanky/metadata-a-datove-sklady.htm

4. THIRUPATAIAH, Ch. Functions of Database Management System (DBMS). Myreadingroom - One stop for Jobs & Career Guidance for Students [online]. Vidžajavada: Myreadingroom.co.in., c2016 [cit. 2020-05-05]. Dostupné z: http://www.myreadingroom.co.in/notes-and-studymaterial/65-dbms/465-functions-of-dbms.html

5. ROUSE, Margaret. What is a Primary Key? - Definition from WhatIs.com. SQL Server: Covering today‘s SQL Server topics [online]. Newton: TechTarget, c2005-2020 [cit. 2020-05-05]. Dostupné z: https://searchsqlserver.techtarget.com/definition/primary-key

6. STEPHENS, Ryan K., Ronald R. PLEW a Arie JONES. Naučte se SQL za 28 dní: [stačí hodina denně]. Brno: Computer Press, 2010. ISBN 978-80-251-2700-1.

7. SQL Data Types for MySQL, SQL Server, and MS Access. W3Schools Online Web Tutorials [online]. Sandnes: Refsnes Data, c1999-2020 [cit. 2020-05-05]. Dostupné z: https://www.w3schools.com/sql/sql_datatypes.asp

8. BRUST, Andrew J., Stephen FORTE a Arie JONES. Mistrovství v programování SQL Serveru 2005. Brno: Computer Press, 2007. Mistrovství. ISBN 978-80-251-1607-4.

9. What is Entity Relationship Diagram (ERD)? Ideal Modeling & Diagramming Tool for Agile Team Collaboration [online]. Hongkong: Visual Paradigm, c2020

[cit. 2020-05-05]. Dostupné z: https://www.visual-paradigm.com/guide/data-modeling/what-is-entity-relationship-diagram/

10. Hierarchie Data → Informace → Znalost – Wikisofia. Wikisofia [online]. Mountain View: Creative Commons, c2013 [cit. 2020-05-05]. Dostupné z: https://wikisofia.cz/wiki/Hierarchie_Data_→_Informace_→_Znalost

11. BUCHALCEVOVÁ, Alena, Stephen FORTE a Arie JONES. Metodiky budování informačních systémů. Praha: Oeconomica, 2009. Mistrovství. ISBN 978-80-245-1540-3.

12. BRUCKNER, Tomáš. Tvorba informačních systémů: principy, metodiky, architektury. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-4153-6.

13. SODOMKA, Petr a Hana KLČOVÁ. Informační systémy v podnikové praxi. 2., aktualiz. a rozš. vyd. Brno: Computer Press, 2010. ISBN 978-80-251-2878-7.

14. BASL, Josef a Roman BLAŽÍČEK. Podnikové informační systémy: podnik v informační společnosti. 2., výrazně přeprac. a rozš. vyd. Praha: Grada, 2008. Management v informační společnosti. ISBN 978-80-247-2279-5.

15. Stránky k výuce informatiky: Delphi od začátku [online]. Vlašim: Marta Bechyňová, c2020 [cit. 2020-05-05]. Dostupné z: http://www.ivt.mzf.cz/algoritmizace-a-programovani/delphi-od-zacatku/

16. Delphi.cz | Komunitní portál Delphi [online]. Doubravy: Ing. Radek Červinka, c2009-2018 [cit. 2020-05-05]. Dostupné z: https://delphi.cz

17. Fast Cross-Platform App Development Software - Embarcadero [online]. Austin: EMBARCADERO, c2020 [cit. 2020-05-05]. Dostupné z: https://www.embarcadero.com

18. SODOMKA, Petr a Hana KLČOVÁ. Trendy na českém ERP trhu a jeho aktuální vývoj. SystemOnLine.cz - ekonomické a informační systémy v praxi [online]. Brno: CCB spol. s r.o., c2001-2020 [cit. 2020-05-05]. Dostupné z:

https://www.systemonline.cz/erp/trendy-na-ceskem-erp-trhu-a-jeho-aktualni-vyvoj.htm

19. Newton - spec. software - Software pro různé obory. [online]. Kolín: Maxsoft, 2000 [cit. 2020-05-05]. Dostupné z: https://www.maxoft.cz

20. Tiss Optic - online informační systém pro optiky [online]. Zlín: TISS Optic CZ, 2020 [cit. 2020-05-05]. Dostupné z: http://www.tissoptic.cz/optic/kontakt/

21. Optik | oční optika | řešení pro prodejny očních optik | ADMIS CZ s.r.o. - IT partner [online]. Zlín: ABRA Publisher, c2014 [cit. 2020-05-05]. Dostupné z: http://www.admis.cz/optik

22. HELIOS – podnikový informační systém, ekonomický a účetní software, systém pro veřejnou správu [online]. Praha: Asseco Solutions, c2020 [cit. 2020-05-05]. Dostupné z: https://www.helios.eu

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

IS – informační systém (Information System)

ERP – plánování podnikových zdrojů (Enterprise Resource Planing)

SQL – strukturovaný dotazovací jazyk (Structured Query Language)

ER – entitně vztahový (Entity Relationship)

SEZNAM GRAFŮ

SEZNAM OBRÁZKŮ

SEZNAM TABULEK