Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Relační databáze v MS SQL Serveru a jejich využití při tvorbě e-shopů

Bakalářská práce

Autor: Patrik Pahulák
Studijní obor: Informační management

Vedoucí práce: doc. RNDr. Petra Poulová, Ph.D.

Hradec Králové Říjen 2021

Prohlášení:

Prohlašuji, že jsem bakalářskou práci zpracoval samostatně a s použitím uvedené literatury.

vlastnoruční podpis

V Hradci Králové dne 15.11.2021 Patrik Pahulák

Poděkování:
Děkuji vedoucímu bakalářské práce doc. RNDr. Petra Poulová, Ph.D. za cenné rady, možnost vytvářet práci pod Jejím vedením a věnovaný čas.

Anotace

Práce je věnována problematice relačních databází a jejich využití pro tvorbu databáze e-shopu. Cílem práce je seznámit se s teoretickými základy relačních databází, syntaxí a příkazy v MSSQL 2012. Dále se práce soustředí na představení základů programovacího jazyka PHP. Tyto teoretické základy poté v praktické části využít pro návrh a tvorbu jednoduché ilustrační e-shopové databáze a jejího propojení s PayPal platební bránou. Tento návrh i samotná tvorba je v práci zaznamenána. Zároveň je implementace otestována pomocí testovacího prostředí PayPal Sandbox.

Annotation

Title: MS SQL Server relational databases and their use in development of e-shop

This thesis is focused on the issue of relational databases and their usage in the process of creating an e-shop database. The goal of this thesis is to get acquainted with theoretical basics of relational databases, syntax and commands in MSSQL 2012. This thesis is also focused to present an introduction to PHP programming language. These theoretical basics are later used to model and implement an e-shop database and its connection with PayPal. The process of modeling this database along with the implementation is written down in this thesis. The implementation is also tested via PayPal Sandbox.

Obsah

1 Úvod

Vývoj informačních technologií za poslední roky vedl k rozšíření internetu do všech oblastí života. Oblast internetového obchodování se rozrostla natolik, že ji lze považovat za již neodmyslitelnou část lidského života. S ohledem na tento fakt je důležité si uvědomit důležitost databázových systémů, které za tímto pokrokem stojí.

V úvodu je nutné zdůraznit, že celá práce je psána v duchu pomůcky při studiu databázových systémů. Tato práce je rozdělena na dvě části. První část, teoretická, je zaměřena na vysvětlení základů relačních databází s ohledem na SQL Server 2012 a základů programovacího jazyka PHP. Teoretická část je rozdělena na dvě hlavní kategorie. První část se zaobírá problematikou relačních databází. V této části jsou rozebrány a popsány základní pojmy týkajících se dané problematiky. Pojmy jsou dále doplněny o ukázkový příklad jejich funkce. Druhá část teoretické slouží jako jednoduchý popis základů programovacího jazyka PHP. Druhá část práce, praktická, se z dílčí části zaobírá problematikou nejvíce využívaných databázových systémů. Zbytek praktické části, která tvoří majoritu obsahu celé praktické části, je zaměřena na ilustrační vývoj velmi jednoduchého webového obchodu

2 Cíl práce

Práce má dva cíle. Prvním cílem, s menší prioritou, je prozkoumat oblast relačních databázových systémů a nalézt ty systémy, které jsou nejpopulárnější. Z těchto systémů poté vybrat čtyři systémy a porovnat je.

Druhý cíl, jež tvoří majoritní část praktické části práce, je implementace jednoduchého malého e-shopu. V rámci této části je navržen relační model databáze daného e-shopu. Model je poté implementován za pomoci Microsoft SQL Management Studio. Tato implementace je využita pro ukládání dat o uživatelích a jejich objednávkách. V rámci této části je dále naimplementována dynamická funkcionalita webového rozhraní e-shopu. Konkrétně se jedná o základní funkcionalitu registrace uživatelského účtu, přihlášení do daného uživatelského účtu. Funkcionalita je dále rozšířena o možnost provést nákupy za pomocí platební brány PayPal. Funkcionalita je naimplementována pomocí programovacího jazyka PHP. Tato implementace je v praktické části práce zaznamenána.

3 Teoretická část

3.1 Definice databáze

Na začátek je dobré si definovat slovo databáze. Databázi můžeme zjednodušeně definovat jako množinu logicky souvisejících dat. Jiná definice dle Šimonové [1] je “databáze je souhrn dat, který se nějak týká určitého tématu nebo účelu”. Kroenke David [2] definoval databáze jako “kolekci souvisejících záznamů, které obsahují vlastní popis”. Jako poslední definice je uvedena definice: “Databáze je soubor dat, který se používá pro modelování některých typů organizačních struktur nebo organizačních procesů, nezáleží přitom jestli je pro záznam a ukládání dat využito papíru, nebo počítače. Obecně pokud shromažďujeme data nějakým organizovaným způsobem, máme databázi.”.

Poslední definice poukazuje na spojitost počítačů a databází. V této práci se samozřejmě budeme věnovat databázím počítačovým za využití databázových systémů. Nejdřív je však nutné uvést alespoň absolutní minimum o historii databází z 20. století.

3.2 Historie databází

Systém řízení bází dat

Systém řízení bází dat, dále již pouze “SŘBD”, je počítačový nástroj, který umožňuje správu databáze. Mezi obecné úkoly SŘBD patří tvorba, zpracování a správa databáze. Konkrétnějšími funkcemi SŘBD je tvorba tabulek, přístup k datům databáze, neboli čtení, úprava-aktualizace dat, zálohování, zajištění bezpečnosti. Příkladem SŘBD je Microsoft SQL Server, který je využit při řešení praktické části této práce. [1][2][3]

První SŘBD se objevují v 60. letech 19. století. Tyto SŘBD vycházely ze dvou přístupů. Prvním přístupem je vzájemné propojování souboru dat, jejichž představitelem jsou síťové a hierarchické databázové systémy. Druhým přístupem je přístup fyzicky nezávislých souborů dat [3], jejichž představitelem jsou databáze relační, kterým se budu věnovat do detailu.

3.3 Relační databáze

Základním pojmem relačních databází je relace. Obecná definice slova relace je "obecná vlastnost konkrétního objektu, která se váže k jinému objektu" [4]. Z matematického hlediska se jedná o libovolnou podmnožinu kartézského součinu $R \subseteq A_1 \times \ldots \times A_n$, kde n € N. Jelikož je relace sama o sobě množinou, můžeme nad ní provádět množinové operace. Relaci v našem případě dále můžeme jednoduše definovat jako tabulku, která se skládá z řádků a sloupců. Tato tabulka se vyznačuje následujícími vlastnostmi.

Relační databáze jsou založené na relačním modelu, o jehož první aplikaci se pokusil E. F. Codd na konci 60. let 20. st. Relační model obecně definuje způsob strukturace dat, způsoby ochrany dat, a operace, která nad daty můžeme provádět. [1] O relačních databázích lze obecně říci, že platí následující:

- Veškerá data jsou reprezentovatelná pomocí uspořádaných struktur řádků a sloupců, relací - tabulek
- V každé konkrétní pozici řádku a sloupce dané tabulky je právě jedna hodnota
- Operace v databázi se provádí vždy nad celou relací a výsledkem těchto operací je vždy opět celá relace

[1][2]

Další charakteristikou je fakt, že dosažení vazby je dosaženo pomocí vztahů, které jsou zastoupeny svými primárními klíči. Pokud se jedná o vazby 1:1, 1:N, pak není použita vazební tabulka. V případě vztahu 1:N je straně N tabulka obohacena o další atribut, sloupec, neboli cizí klíč.

Primární klíč je atribut, sloupec, jehož hodnota vždy jednoznačně identifikuje záznam v tabulce. V předchozí větě je nutno poukázat na slovo "identifikuje", jelikož s primárními klíče se většinou potkáme ve formě nějakého identifikátoru. Dále je nutno poukázat na slovo "jednoznačně", tato vlastnost primárního klíče poukazuje na fakt, že každý řádek v tabulce má svůj jedinečný primární klíč, který nemá žádný jiný řádek v této tabulce. Primární klíče mohou být reprezentovány jako množina

několika atributů najednou. Těmto primárním klíčům říkáme složené primární klíče. Je to tedy primární klíč složen z několika sloupců tabulky, přičemž se nemůže objevit jedna kombinace těchto všech klíčů najednou více než jednou.[1][2]

Cizí klíč je atribut, sloupec, který v tabulce zprostředkovává spojení s další tabulkou, tabulkami. Hodnota tohoto cizího klíče pak odpovídá hodnotě primárního klíče v tabulce jiné. Jinak řečeno cizí klíč v jedné tabulce je primárním klíčem tabulky druhé.[1]

Pochopení principu obou klíčů je nutné pro pochopení relací, vztahů, mezi jednotlivými tabulkami.

3.3.1 Vztahy mezi tabulkami

V předchozí kapitole byly zmíněny vztahy mezi tabulkami při rozboru primárních a cizích klíčů. Těmto vztahům můžeme mimo jiné říkat vazby. Vztah mezi tabulkami je vyjádřením nějaké spojitosti, vazby, tabulek. U tabulek rozlišujeme tři možnosti vztahů, které jsou označeny pomocí tzv. kardinality. Kardinalita označuje poměr, který představuje kolik záznamů jedné tabulky může navázat spojení se záznamy tabulky druhé. V předchozí větě je nutné poukázat na možnost, ne nutnost, spojení. Vztah v relačních databází pouze dává možnost nějakým záznamům propojení navázat, je tedy naprosto možné, že tabulka bude mít záznam, který s okolními tabulkami vztahu nenavázal. Existují tři možné vztahy:

- 1:1
- 1:N
- N:N

3.3.1.1 Vztah 1:1

Vztah 1:1 představuje situaci, kdy záznam jedné tabulky může navázat spojení s jedním záznamem tabulky druhé a zároveň jeden záznam tabulky druhé může navázat spojení s jedním záznamem tabulky první. Jako příklad uvedu situaci, kdy máme tabulku Lidé, jež reprezentuje lidi, kteří mohou mít automobil, a tabulku

Automobil, která reprezentuje automobily. V této situaci budeme uvažovat, že jeden člověk může mít jeden automobil. Tedy jedná se o situaci, kdy tabulky propojíme vztahem 1:1. Na obrázku č.1 vidíme dva lidi v tabulce Lidé. Primárním klíčem tabulky Lidé je id_clovek, dále evidujeme atributy jméno a příjmení a cizí klíč id_auto. Tabulka Automobil má v sobě dva záznamy. Jak je vidět máme dva originální záznamy tabulky Lidé, přičemž pouze jeden z nich má nějaké auto. Tato situace poukazuje na fakt, že při zavedení vztahu 1:1 má záznam pouze možnost propojit se s nějakým jiným záznamem. Tedy doopravdy může nastat situace, že člověk auto nemá. Tato situace není namodelovaná ideálně a slouží pouze pro příklad.

	id_clovek	Jmeno	Prijmeni	Automobil
1	1	Marcel	Ondřej	Škoda Octavia
2	2	Jiří	Ondrejka	NULL

Obrázek 1 - Vztah 1:1
Zdroj – Vlastní tvorba

3.3.1.2 Vztah 1:N

Tento vztah reprezentuje situaci, kdy jeden záznam tabulky A může spojení s jedním nebo vícero záznamy tabulky B. V opačné situaci to pak vyjadřuje fakt, že jeden záznam tabulky B může navázat jedno spojení. Pro příklad uvedu vylepšenou situaci z popisu předchozího vztahu 1:1. V této situaci uvažujeme znovu tabulku Lidi a tabulku Automobily. V této situaci však jeden člověk může mít automobilů více. Na níže uvedeném obrázku je vidět, že v tabulce Lidé již není cizí klíč, sloupec id_automobil, referující na tabulku Automobily. Tato situace je namodelovaná tak, že každý automobil ví, ke kterému člověku patří. Opět zde máme dva záznamy v tabulce Lidé. V tabulce Automobily máme záznamy tři, přičemž dva z těchto záznamů mají stejný cizí klíč, referující na primární klíč tabulky Lidé. Tato situace tedy říká, že záznamy s id_auto 1 a 2 v tabulce Automobil navazují spojení se záznamem z tabulky Lidé s id_clovek s hodnotou 1. Třetí záznam tabulky Automobil nemá žádné spojení a zároveň spojení nemá ani druhý záznam tabulky Lidé, což znovu poukazuje na fakt, že ke spojení nemusí dojít. Tento vztah se v praxi objevuje často.

Lidé

	id_clovek	Jmeno	Prijmeni
1	1	Marcel	Ondřej
2	2	Jiří	Ondrejka

Automobily

	id_auto	Nazev	cena	id_clovek
1	1	Škoda Octavia	50000	1
2	2	Škoda Fabia	65000	1

Obrázek 2 - Tabulky se vztahem 1:N
Zdroj - Vlastní tvorba

3.3.1.3 Vztah N:N

Vztah N:N řeší situací, kdy záznam tabulky A může mít spojení s několika záznamy tabulky B a zároveň platí, že jeden záznam z tabulky B může mít spojení s několika záznamy tabulky A. V relačních databázích však tento vztah nelze implementovat přímo. Místo přímého propojení tabulek se využívá tabulky spojovací. Konkrétně to vypadá tak, že se mezi tabulky ve vztahu N:N, přidá jedna tabulka navíc, a tuto tabulku poté spojíme s jednotlivými tabulkami ve vztahu 1:N. Na příkladu se znovu ukáže, proč tak děláme. Znovu uvažujme situaci, kdy máme tabulky Automobily a Lidé. V této situaci, ale budeme uvažovat, že jeden člověk může mít spojení s vícero záznamy tabulky Automobily, ale zároveň jeden automobil může být spojen s vícero lidmi. Tedy jeden člověk může vlastnit vícero typů aut a zároveň jeden konkrétní typ auta může být vlastněn vícero lidmi. Jak je zde vidět máme znovu dva záznamy v tabulce Lidé a dva záznamy v tabulce Automobily. Dále zde přibyla propojovací tabulka LidéAutomobily, která má vlastní primární klíč a dva cizí klíče - id_clovek a id_auto. V této situaci se rozhodneme, že první člověk vlastní oboje auta, čemuž odpovídají první dva záznamy v propojovací tabulce. Dále uvažujme, že druhý člověk vlastní pouze první auto, čemuž odpovídá třetí záznam propojovací tabulky. Pomocí spojovací tabulky jsme docílili toho, že nedostáváme duplicitní data v našich hlavních tabulkách.

Lidé

	id_clovek	jmeno	prijmeni
1	1	Marcel	Ondřej
2	2	Jiří	Ondrejka

Automobily

	id_auto	nazev	cena
1	1	Škoda Octavia	50000
2	2	Škoda Fabia	65000

LidéAutomobily

	id_propojovaci	id_clovek	id_auto
1	1	1	1
2	2	1	2

Obrázek 3 - Tabulky ve vztahu N:N
Zdroj - Vlastní tvorba

3.3.2 Normalizace

Proces normalizace je proces, při kterém dochází k zjednodušování a optimalizaci databázových struktur – tabulek. Cílem tohoto procesu je odstranit nežádoucí jevy. Pro tyto účely jsou tu tzv. normalizační pravidla. Obecně platí, že by správně navržená tabulka měla splňovat tato pravidla.[3]

3.3.2.1 První normální forma (1NF)

Tabulka je v 1NF za předpokladu, že její atributy jsou již dále nedělitelné, tzv. atomické. [3][1] Tato normální forma tedy říká, že atributy tabulek by měly mít ideálně jednu hodnotu. Jednou hodnotou se však nerozumí nutně jedno slovo, či jedno písmeno, samozřejmě to možností je, ale není to ideální ve všech případech. Například při evidenci filmů bychom chtěli evidovat název filmu, což by neznamenalo, že pro každé slovo v názvu vytvoříme vlastní atribut.

3.3.2.2 Druhá normální forma (2NF)

Tabulka je v 2NF, pokud je v 1NF a zároveň musí být každý neklíčový atribut závislý na celém primárním klíči. Toto pravidlo se však musí řešit pouze v případech, že tabulka obsahuje primární klíč složený z vícero atributů. Tabulky s jednoduchým primárním klíčem, by toto pravidlo měly splňovat automaticky.[3][2]

3.3.2.3 Třetí normální forma (3NF)

Tabulka je v 3NF, jestliže je v 2NF a zároveň jsou všechny neklíčové atributy navzájem nezávislé. Pokud se objeví atribut, který je závislý na jiném atributu než na primárním klíči, pak je žádoucí, aby byl z dané tabulky tento atribut odstraněn. Dalším krokem je tvorba tabulky nové. Primárním klíčem této tabulky je atribut, na kterém byl již odstraněný atribut závislý. Dále přidáme samotný atribut, který jsme z původní tabulky odstranili. Jako příklad zde uvádím situaci, kdy evidujeme lidi a jejich automobily. Primárním klíčem této tabulky je id_clovek. Zjišťujeme, zda jsou všechny atributy tabulky závislé jedině na primárním klíči. Když dojdeme k atributu

rychlost_v_km_za_h, který vyjadřuje rychlost automobilu, pak uvidíme, že tento atribut je závislý na atributu id_auto, tedy není závislý na primárním klíči. V této situaci je žádoucí, abychom atribut z tabulky odstranili a přidali ho do tabulky, která odpovídá automobilům.

	id_clovek	jmeno	prijmeni	id_auto	rychlost_v_km_za_h
1	1	Jiří	Ondrejka	1	150
2	2	Aneta	Čechová	2	155

Obrázek 4 - tabulky před 3NF
Zdroj - Vlastní tvorba

Lidé

	id_clovek	jmeno	prijmeni	id_auto
1	1	Jiří	Ondrejka	1
2	2	Aneta	Čechová	2

Automobily

	id_auto	nazev_auta	rychlost_v_km_za_h
1	1	Škoda Felicia	150
2	2	Škoda Fabia	155

Obrázek 5 - Tabulky po 3NF
Zdroj - Vlastní tvorba

3.4 SQL

SQL, Structured Query Language - Strukturovaný dotazovací jazyk, je neúplný programovací jazyk, některá literatura ho nazývá "podjazykem".[5][6] SQL je standardizovaný jazyk, který je využíván při komunikaci s databázemi. Dotaz, Query, je odesílaný požadavek na databázi, na který databáze žadateli posílá zpět určitou množinu dat. SQL slouží pro práci s relačními databázemi, je však dobré vědět, že databáze, která rozumí SQL jazyku, nemusí být automaticky relační a zároveň platí, že může být relační databáze, která SQL jazyku rozumět nebude.[5][2] SQL je jazyk, který má mnoho variant. Varianty mají společné základní principy, ale jsou do nějaké míry modifikovány. Tento fakt se potom projevuje hlavně tak, že SQL jedné varianty poté nemusí fungovat v programu, který využívá varianty jiné. Příkladem takových variant je např. MySQL a Microsoft SQL. SQL napsané pod standardem MySQL pak nemusí fungovat v programu, který očekává standard typu Microsoft SQL.

3.4.1 SQL příkazy

3.4.1.1 SELECT

Příkaz SELECT je příkaz, který umožňuje načítat data z databáze. Je to příkaz, který data nijak nemění, pouze je zobrazuje. Nyní uvedu základní syntaxi příkazu SELECT.

```
SELECT Názvy sloupců oddělenné čárkami FROM Název tabulky
```

Při psaní tohoto příkazu, je vždy na začátku slovo SELECT. Následují názvy sloupců, které chceme zobrazit. Pokud chceme zobrazit všechny sloupce tabulky, máme možnost místo vypisování všech sloupců napsat pouze " * ". Jako další část příkazu je klauzule FROM, po které následuje název tabulky, ze které chceme data zobrazit. Za názvem tabulky máme možnost ještě přesněji definovat o jaká data máme zájem. K tom slouží např. klauzule WHERE, HAVING a jiné. SELECT nám dále umožňuje data srovnat pomocí klauzule ORDER BY. Níže lze vidět konkrétní příklad tohoto příkazu.

```
SELECT * FROM dbo.Automobily WHERE dbo.Automobily.cena<100000 ORDER BY
dbo.Automobily.cena DESC;
```

Příkaz SELECT nabízí mnohem více možností, jak data zobrazovat. Do detailu o nich však hovořit nebudu.

3.4.1.2 INSERT, UPDATE, DELETE

Všechny tyto příkazy jsem se rozhodl popsat v jedné podkapitole. Důvodem tohoto rozhodnutí je využití těchto příkazů. Všechny totiž umožňují nějakou formu editace již existujících tabulek.

a) INSERT

Příkaz INSERT umožňuje přidat do tabulky nový záznam. Základní syntax příkazu INSERT je uveden níže.

```
INSERT INTO Název tabulky (Názvy sloupců) VALUES (Vkládané hodnoty)
```

Tento příkaz umožňuje i variantu, kdy neudáváme jednotlivé názvy sloupců. Nutné je podotknout pořadí vkládaných hodnot za slovem „VALUES". Toto pořadí musí

odpovídat pořadí, ve kterém jsme napsali názvy sloupců. Jestli jsme použili variantu, kdy názvy sloupců vynecháváme, pak musí pořadí hodnot odpovídat pořadí sloupců v tabulce. Při vkládání je dále nutné myslet na zadávání originálních hodnot pro hodnoty odpovídající primárnímu klíči. Pokud tabulka, do které vkládáme data, obsahuje nějaký cizí klíč, pak musíme dbát na to, abychom jako hodnotu cizího klíče uváděli hodnotu, která odpovídá nějakému záznamu v tabulce odkazované.

b) UPDATE

Jak již napovídá název daného příkazu, příkaz UPDATE slouží pro aktualizaci dat v tabulkách. Syntax příkazu UPDATE je uvedena níže.

```
UPDATE Název tabulky
SET Název sloupce hodnoty=Nová hodnota
WHERE Podmínky;
```

Tento příkaz nastavuje novou hodnotu řádkům, záznamům, které splňují uvedenou podmínku. Existuje i možnost podmínku neuvádět, což by znamenalo, že se změní hodnota všech záznamů.

c) DELETE

Příkaz DELETE slouží pro mazání dat. Syntax příkazu DELETE je znovu uvedena níže.

```
DELETE FROM Název tabulky
WHERE Podmínky;
```

Jak je vidět z výše uvedeného SQL příkazu, příkaz DELETE a UPDATE se podobají. Oba příkazy fungují na základě podmínek napsané za klíčovým slovem WHERE.

3.4.1.3 JOIN

JOIN neboli operace spojování tabulek, je příkaz, který je založen na principu kartézského součinu první tabulky s tabulkou druhou. Výsledkem tohoto kartézského součinu je potom množina všech kombinací první a druhé tabulky. Toto spojení můžeme získat jednoduchou selekcí, kdy za slovo FROM napíšeme název

tabulky první a název tabulky druhé oddělené čárkou.[4] Toto spojení však není úplně ideální, jelikož pro každý řádek v první tabulce dojde ke spojení s každým řádkem v tabulce druhé, což při velkém počtu řádků není nikterak žádoucí. Pro tyto případy musíme jasně definovat jakým způsobem tabulky spojovat. Existují následující varianty:

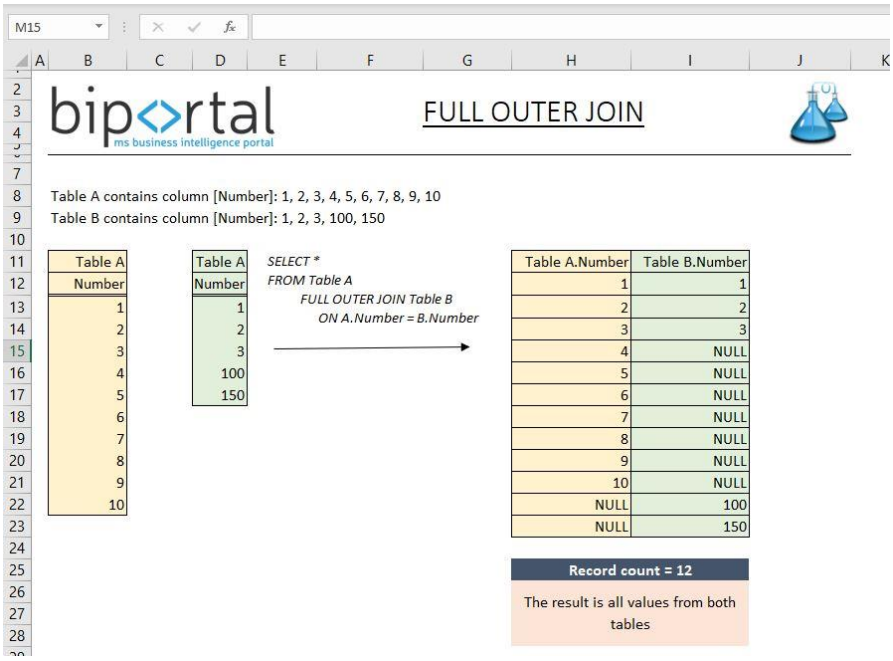
- JOIN (INNER JOIN)
- OUTER JOIN, LEFT OUTER JOIN, RIGHT OUTER JOIN

3.4.1.4 INNER JOIN

Výsledkem příkazu INNER JOIN tabulky A a tabulky B je množina řádků, které mají stejné hodnoty ve společných sloupcích tabulek A a B. Tedy výsledkem tohoto příkazu jsou všechny kombinace řádků z tabulky A, které navázaly spojení s řádky tabulky B.

3.4.1.5 OUTER JOIN

Tento příkaz nazýváme vnějším spojením. Vnější spojení funguje tak, že nám vrátí každý řádek z první tabulky spojený s 0 až více řádky z tabulky druhé, zároveň vrací i každý řádek z druhé tabulky spojený s 0 až více řádky z tabulky první. [6] Z předchozí věty je nutné si všimnout spojení 0 a více. Tato věta poukazuje na fakt, že pokud má daný řádek první tabulky 0 spojení s řádky tabulky druhé, pak jsou ve výsledku příkazu vyjádřeny hodnoty druhé tabulky vyjádřeny pro tento konkrétní řádek pomocí NULL. Příkaz má dvě varianty – LEFT OUTER JOIN a RIGHT OUTER JOIN. LEFT OUTER JOIN je stejné jako klasické vnější spojení, ale jsou v něm pouze spojené a nespojené řádky tabulky, která je nalevo od LEFT OUTER JOIN v SQL Query. RIGHT OUTER JOIN funguje stejně, ale pro tabulku napravo od RIGHT OUTER JOIN v SQL Query.



Obrázek 6 - Příklad funkce vnějšího spojení
Zdroj - [7]

3.5 PHP

PHP je programovací jazyk pro vývoj dynamických, interaktivních webových stránek. PHP programy běží na webovém serveru. PHP programy reagují na požadavky uživatelů webových stránek. Jednou ze základních vlastností PHP, je fakt, že PHP kód se dá vložit přímo do HTML webových stránek, což umožňuje rychlou tvorbu dynamického obsahu těchto stránek. Dynamické webové stránky jsou stránky, které jsou schopny automaticky měnit svůj obsah při každé návštěvě takové stránky. Interaktivní webové stránky lze definovat jako stránky, které reagují na vstupy od svých uživatelů. [9]

3.5.1 Historie PHP

PHP vzniklo jako PHP Tool v roce 1994. Autorem PHP je Rasmus Lerdorf, který PHP vytvořil jako náhradu některých skriptů jazyka Perl, které sám využíval na své osobní webové stránce. [10] Systém PHP byl původně určen pouze pro osobní použití autora, který ho později sdílel s jeho přáteli, což vedlo k přidávání nových funkcionalit. Lerdorf se poté rozhodl v roce 1995 rozhodl sdílet zdrojový kód s širokou veřejností pod názvem PHP version 2. V roce 1998 poté vyšla verze PHP 3

a následně PHP 4 v roce 2000. PHP 5 vydané v roce 2004 přineslo nový objektový model, který původně PHP chyběl. [9][10][11]

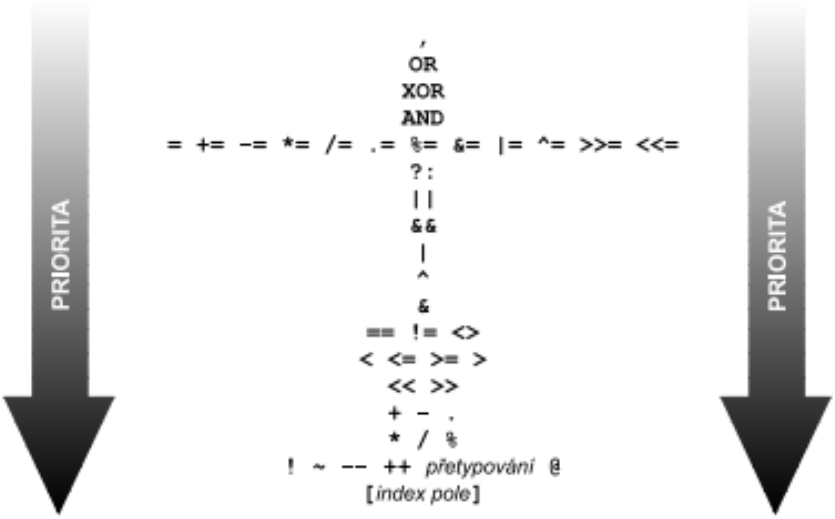
3.5.2 Základy PHP

PHP program – skript je stejně jako každý jiný programovací jazyk složen ze sekvence příkazů. PHP umožňuje dva různé způsoby, jak od sebe příkazy oddělovat. Prvním způsobem je oddělení pomocí středníku. Druhý způsob je vložit každý příkaz do následujících znaků: <? ?>. [10] Dále je potřeba zmínit označování proměnných. Proměnné jsou v jazyce PHP vyznačeny pomocí znaku „ $ “. Proměnné jazyka PHP jsou takzvaně case-sensitive, což znamená, že proměnná $promenna a proměnná $PROMENNA jsou dvě různé proměnné. U zápisů funkcí a příkazů toto pravidlo však neplatí. Dále o PHP proměnných platí, že musí začínat buď písmenem, nebo podtržítkem. Zajímavostí u proměnných PHP je automatické přiřazení datového typu proměnné, což se v praxi označuje jako „loose typing“.[9][13]

3.5.3 Operátory a výrazy jazyka PHP

Jako každý jiný programovací jazyk nabízí PHP operátory, které slouží pro manipulaci s proměnnými. Operátorem se obecně rozumí symbol spjatý s daným typem manipulace proměnných. [9] Operátory se dělí na aritmetické, přiřazovací, bitové, porovnávací, logické, String, Array, inkrementální a dekrementální, prováděcí a operátor řízení chyb a další.[9][11] Operátory mají v PHP jako v jiných jazycích prioritu. Priorita operátorů je zobrazena na obrázku č. 7. Pojmem výrazy se zjednodušeně rozumí cokoli, co ve svém procesu produkuje novou hodnotu. [9]

```
switch(výraz){
case hodnota1:příkaz1;
              příkaz2;
              break;
case hodnota2:příkaz1;
              příkaz2;
              break;
default:příkaz1;
        příkaz2;
        break;
}
```



Obrázek 7 - priorita PHP operátorů
Zdroj – [10]

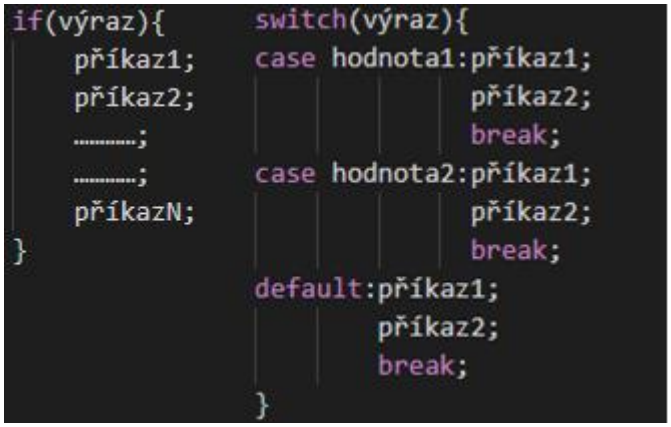
3.5.4 Větvení

V případě, že potřebujeme algoritmus, program, rozvětvit na vícero možností tu existují dva způsoby větvení neboli dva různé způsoby, jak ošetřovat podmínky.

if, if else, if elseif else – Všechny tyto příkazy jsou založeny na stejné logice. Do závorky se zapisuje logický výraz, který v případě, že je vyhodnocen na hodnotu True, logická 1, vykoná příkazy v závorkách „ { } “. Varianta if else pracuje na stejném principu, akorát v případě, kdy podmínka splněná není, jsou vykonány příkazy v závorkách „ { } “ zapsané za klíčovým slovem „else“. Třetí varianta, if elseif else, umožňuje do stejného větvení přidávat další možnosti. Tedy v případě, že první podmínka splněná není, se testuje, zda platí druhá podmínka, v případě že

podmínka splněna je, je spuštěna patřičná sekvence příkazů. V případě, že podmínka splněna není, probíhá další kontrola další možné elseif podmínky. V případě, že žádná definovaná podmínka není splněna, je proveden blok příkazů za slovem else.[9][11][12]

switch – Příkaz switch je založen na myšlence existence jednoho výrazu, který může nabývat velké množiny hodnot. V případě, že tyto hodnoty můžeme jasně definovat, je vhodné využít pro větvení příkazu switch. Výhodou příkazu switch je poskytnutí jednoduché změny definice výrazu – podmínky. Jednotlivé hodnoty výrazu se zapisují za slovo case. Každý case má poté zapsané příkazy, které se mají vykonat. Jednotlivé hodnoty se poté mohou oddělovat pomocí „break“, který ukončí plnění příkazů dané hodnoty. V případě, že break chybí můžou příkazy takzvaně „propadnout“, což znamená, že se vykonají další příkazy následující hodnoty. [9][11]



Obrázek 8 - ukázka if a switch v PHP
Zdroj – Vlastní tvorba

3.5.5 Cykly

Cykly jsou příkazy, které slouží k opakovanému provádění sekvence příkazů. Opakování trvá, dokud není splněna nějaká podmínka. Podmínkou se rozumí výraz. Existují tři různé cykly. Tyto cykly jsou for, while a do-while. Cykly for a while se mezi sebou dají zaměňovat. [11]

while – Cyklus while je nejjednodušší na pochopení. Je to z toho důvodu, že anglické slovo while je do češtiny přeloženo jako „dokud“. Cyklus funguje tak, že dokud platí podmínka, výraz, je prováděn příkaz, nebo sekvence příkazů. Podmínka

je zde kontrolována hned na začátku, což může znamenat, že se cyklus vůbec nemusí vykonat. [9][11] [13]

do-while – Obdobně jako cyklus while, je cyklus do-while vcelku jednoduchý na pochopení. Oproti cyklu while, však funguje tak, že sekvence příkazů v těle cyklu se provádí, dokud platí podmínka. Kontrolování podmínky cyklu se provádí až na konci cyklu, což znamená, že cyklus se vždy provede alespoň jednou. [9][11][13]

for – Cyklus for je podobný jako příkaz while. Podmínka cyklu je taktéž kontrolována hned na začátku, takže k provedení příkazů v těle cyklu nemusí vůbec nikdy dojít. Cyklus for se zpravidla využívá, pokud přesně víme, kolikrát se má tělo cyklu provést. [9][13]

```
while(výraz){
    příkaz1;
    příkaz2;
    .......;
    .......;
    příkazN;
}
```

```
do{
    příkaz1;
    příkaz2;
    .......;
    .......;
    příkazN;
}while(výraz)
```

```
for(výraz1;výraz2;výraz3){
    příkaz1;
    příkaz2;
    .......;
    .......;
    příkazN;
}
```

Obrázek 9 - ukázka cyklů v PHP
Zdroj – Vlastní tvorba

4 Praktická část

Prvním cílem praktické části je průzkum oblasti SŘBD. Průzkum je konkrétně směřován na nalezení několika nejvíce využívaných systémů a jejich následné porovnání. Druhým cílem je návrh a implementace databáze pro mnou vyvíjený webový obchod. Webová stránka využitá pro plnění tohoto úkolu byla vytvořena a schválena v předchozím studiu předmětu TNPW1.

4.1 Průzkum oblasti SŘBD

Tato kapitola se soustředí na nalezení několika využívaných SŘBD. Tyto SŘBD jsou poté porovnány. SŘBD jsou vyhledány na základě různých statistik a průzkumů sahajících maximálně do roku 2018. Je nutné zmínit, že průzkum se týká pouze relačních SŘBD využívajících jazyka SQL. NoSQL databáze mohou být ve využívaných zdrojích zmíněné, ale v tomto průzkumu se nebudou využívat. Mezi nejdůležitější zdroje patří dvojice Stack Overflow a DB-Engines. Důvodem pro jejich využití je největší konzistence v podobě sledování daného tématu.

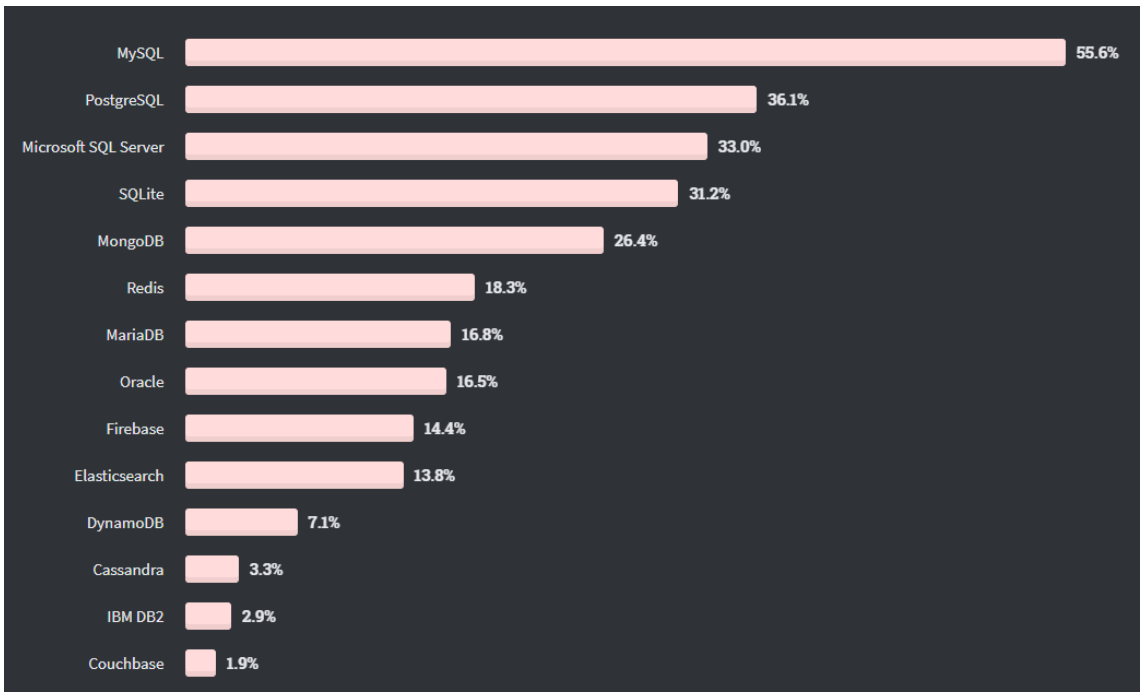
4.1.1 Rozbor statistik a dotazníků

Nejaktuálnějším zdrojem průzkumu je webový portál db-engines.com. Jedná se o tabulku obsahující názvy 10 nejvíce využívaných databázových systémů ke květnu roku 2021. Způsob získávání dat u průzkumů db-engines spočívá ve sčítávání počtu vyhledávání názvů systémů na Google.com a Bing.com, dále do výpočtu patří vývoj na Google Trends, frekvence dotazů na diskuzních fórech jako Stack Overflow, zmínky na sociálních médiích a mnoho dalších. Z tabulky je nutno si zapamatovat názvy Oracle, MySQL, Microsoft SQL Server (MSSQL), PostreSQL a MongoDB. Důvodem je fakt, že se jedná o nejčastěji zmiňované názvy napříč různými zdroji.

Rank	Name	Type	May 2021	Chart May 2021
1.	Oracle	Relational, Multi-model	1269.94	
2.	MySQL	Relational, Multi-model	1236.38	
3.	Microsoft SQL Server	Relational, Multi-model	992.66	
4.	PostgreSQL	Relational, Multi-model	559.25	
5.	MongoDB	Document, Multi-model	481.01	
6.	IBM Db2	Relational, Multi-model	166.66	
7.	Redis	Key-value, Multi-model	162.17	
8.	Elasticsearch	Search engine, Multi-model	155.35	
9.	SQLite	Relational	126.69	
10.	Microsoft Access	Relational	115.40	

Obrázek 10 - Nejvyhledávanější RDMBS
Zdroj - [18]

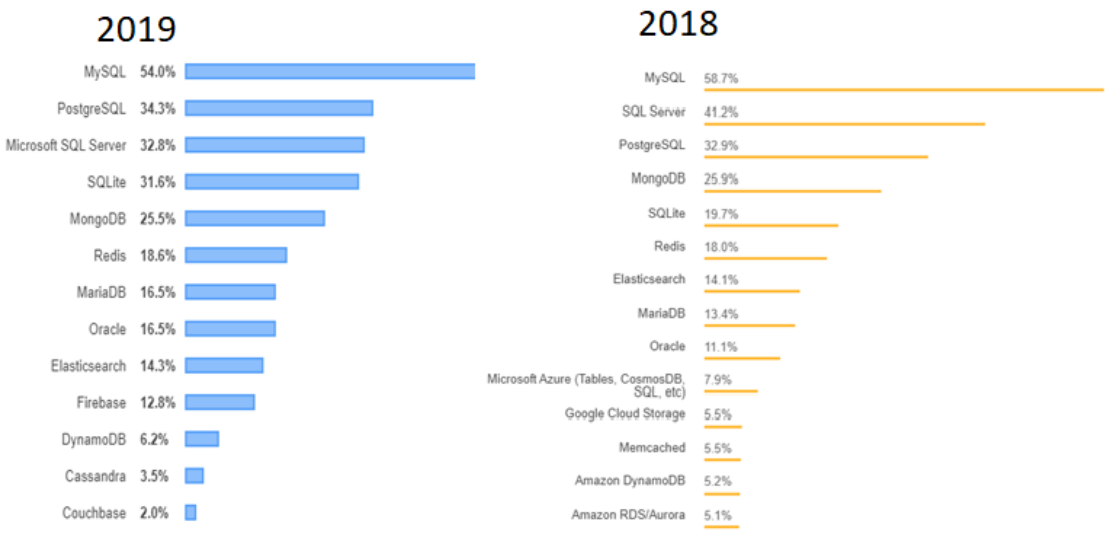
Druhým zdrojem je průzkum portálu Stack Overflow z roku 2020. Průzkum se koná každý rok a týká se velkého množství technologických oblastí. Průzkumu se daný rok zúčastnilo přibližně 65.000 respondentů. Důležitým poznatkem tohoto průzkumu, je fakt, že Oracle databáze se oproti předchozímu zdroji posunuly z prvního místa na místo osmé. Druhé místo zaujímá PostgreSQL a třetí místo zaujímá MSSQL stejně jako u předchozího zdroje.



Obrázek 11 - Nejoblíbenější DBMS 2020
Zdroj – [19]

Zdroj třetí je porovnání výsledků výše zmiňovaného průzkumu portálu Stack Overflow z roků 2019 a 2018. Důležitým poznatkem je absolutní dominance

MySQL za roky 2020, 2019 a 2018. Dále je zajímavý růst PostgreSQL a SQLite a zároveň pokles MSSQL.



Obrázek 12 - Nejoblíbenější DMBS 2019, 2018
Zdroj – [20][21]

4.1.2 Porovnání vybraných SŘBD

Tato část je věnována samotnému představení vybraných SŘBD z předchozí části. Pro výběr byly stěžejní hlavně průzkumy z portálu Stack Overflow. Vybrány byly top čtyři SŘBD z těchto průzkumů za poslední tři roky.

1. Microsoft SQL Server (MSSQL)
2. MySQL
3. SQLite
4. PostgreSQL

	MSSQL	MySQL	SQLite	PostgreSQL
Vývojář	Microsoft	Oracle	D. Richard Hipp	PostgreSQL Global Development Group
Typ licence	Free verze pro vývojáře a malé podniky, placená pro firmy	Open-source	Open-source	Open-source
Vývoj v jazyce	C, C++	C++	C	C
Podpora OS	Windows OS, SUSE, Red Hat, Ubuntu, Docker Engine	Windows OS, Linux, Solaris, HP-UX, Debian, macOS, SUSE, Ubuntu a další	Android, iOS, Linux, Mac, Windows, Solaris, VxWorks	Linux, Windows, macOS, Solaris, FreeBSD, OpenBSD, NetBSD, HP-UX, UnixWare a další
Architektura	Client/Server	Client/Server	File based	Client/Server
Podporované programovací jazyky	C#, .NET, Java, Node.js, Python, C++, PHP, Ruby	C, C++, Delphi, Perl, Java, Lua, .NET, Node.js	ActionScript, Ada, Basic, C, C#, C++, D, Delphi, Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MATLAB, PHP, PL/SQL	NET, C, C++, Delphi, Java, Perl, PHP, Python, Tcl
Populární využití	Velké podniky, Střední a malé podniky	Webové stránky, Webové aplikace	Nízkoprovozové webové stránky, IoT, Testing a Development,	Analytika, Data mining, Datové sklady, Business Inteligence

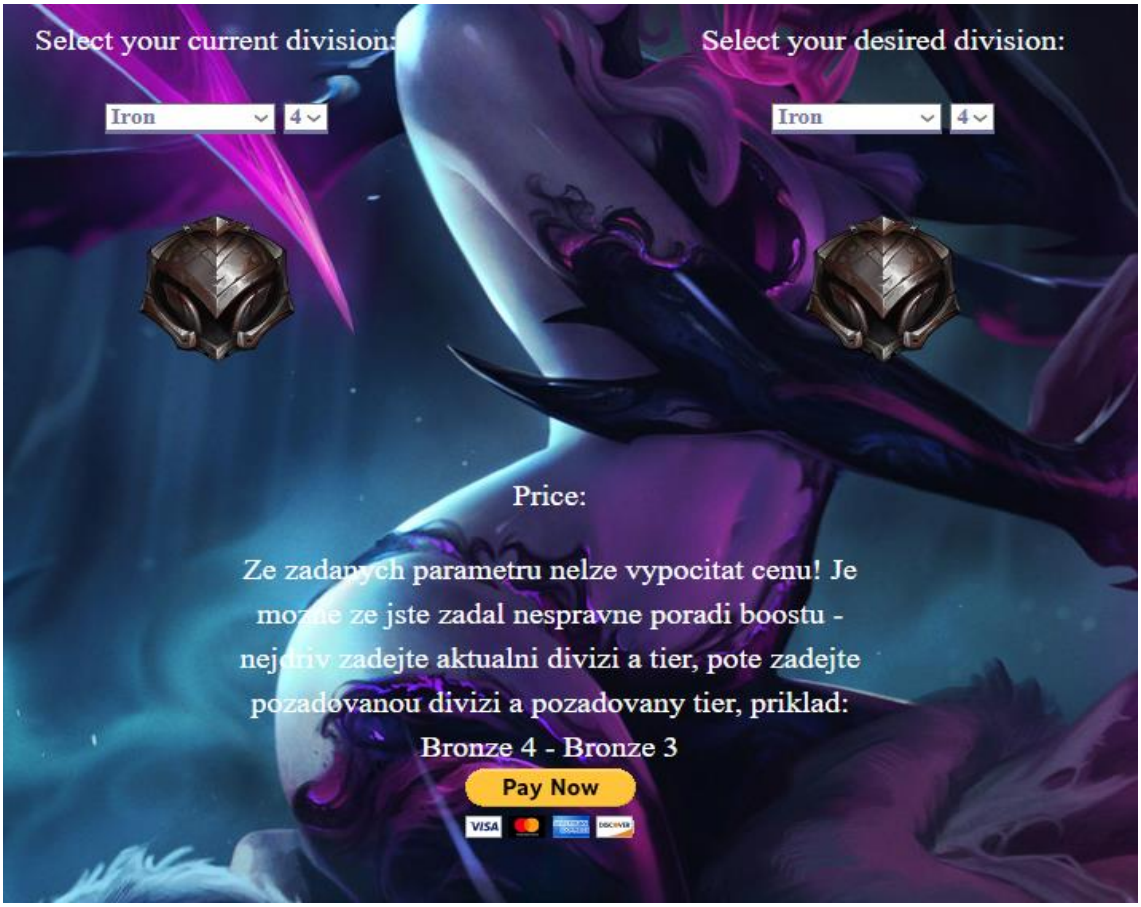
Tabulka 1 - Porovnání DBMS

4.2 Popis projektu a požadavky na databázi

4.2.1 Popis projektu

Webový obchod nabízí služby tzv. eloboostingu pro videohru League of Legends. Cílem je dorazit na web, zadat určité parametry pomocí 4 dropdown

tlačítek. Po vybrání adekvátních parametrů je posléze vypočítána cena tohoto úkonu. Po stisku Paypal tlačítka je klient přesměrován na platební bránu Paypal, kde je vyzván k platbě dané služby. Zaplacením se vyvolá navázání připojení k databázi a následného zápisu dat týkajících se daného klienta a daného úkonu.



Obrázek 13 - Ukázka vybírání parametrů
Zdroj - Vlastní tvorba

Požadavky na databázi činí v první části možnost ukládat data o zakázkách. Konkrétně se jedná o parametry začínající a končící divize, ze kterých se počítá výsledná cena úkonu, výsledná cena služby, datum provedené platby a email klienta. Dále databáze drží do určité míry data o samotných klientech. Tento projekt vyžaduje pouze klientův email, jméno a příjmení a pro registrované klienty jejich heslo. Databáze umožňuje evidovat zaměstnance a jejich přezdívku a přístupové heslo.

4.2.2 Použité technologie pro vývoj projektu

Tato podkapitola je zaměřena na představení a krátký popis všech využitých technologií a softwaru sloužících k implementaci projektu. Je nutno podotknout, že vzhledem k teoretické části nejsou v této podkapitole zmíněny a popsány technologie MSSQL a PHP.

HTML

Hyper Text Markup Language (HTML) je značkovací jazyk, který komunikuje s webovým prohlížečem a určuje strukturu webových stránek na World Wide Webu. [14][15]

CSS

Cascading Style Sheets (CSS) je jazyk, který slouží pro definici vzhledu dokumentů napsaných ve značkovacích jazycích jako HTML. Jeho nejčastějším využití je stylování webových stránek. [16]

JavaScript

JavaScript (JS) je skriptovací programovací jazyk, který slouží pro vývoj dynamických webových aplikací. Jeho původní využití bylo založené pouze pro stranu klienta, prohlížeče, v dnešních dnes se však používá i na straně serveru. Jedná se o jeden z nejvíce využívaných programovacích jazyků na světě.[17]

XAMPP – Apache

XAMPP je open-source balíček, který se využívá pro vývoj na webu. Hlavní funkcionalita, která je v projektu využita je Apache. Apache je cross-platformový webový server. V projektu je konkrétně využit pro hostování serveru, na kterém běží PHP kód.

SQL Server Management Studio

SQL Server Management Studio je software od společnosti Microsoft, který slouží pro správu a tvorbu databází v MSSQL. Jedná se o placenou službu, která však má developerskou edici, která je zcela zdarma a obsahuje veškerou funkcionalitu verze placené. V projektu je software využíván pro tvorbu a správu databáze projektu.

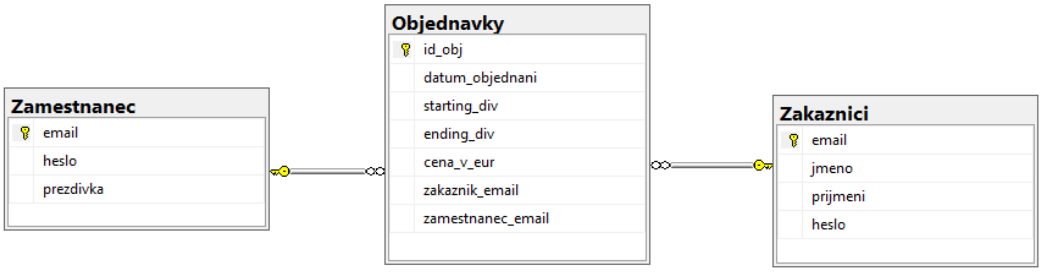
Visual Code

Visual Code je textový editor od společnosti Microsoft. Tento editor je zcela zdarma a v projektu je využit pro veškerou programovací činnost projektu.

Ngrok

Jedná se o cross-platformovou aplikaci, která je schopna zpřístupnit lokální serverové porty na internet. Konkrétní využití aplikace v projektu spočívá v možnosti navštívit lokální PayPal IPN listener v PHP, který běží na lokálním Apache serveru. Bez této aplikace by PayPal nemohl posílat zpětná data o vykonaných platbách.

4.3 Návrh tabulek databáze



Obrázek 14 - Diagram tabulek databáze
Zdroj - Vlastní tvorba

4.3.1 Tabulka Objednavky

Column Name	Data Type	Allow Nulls
id_obj	int	☐
datum_objednani	date	☐
starting_div	varchar(20)	☐
ending_div	varchar(20)	☐
cena_v_eur	float	☐
zakaznik_email	varchar(50)	☐
zamestnanec_email	varchar(50)	☑

Obrázek 15 - Tabulka Objednavky
Zdroj - Vlastní tvorba

Tabulka Objednavky odpovídá výše uvedeným parametrům. Je obohacena o další sloupec „zamestnanec_email“, který má nastaven Allow Nulls. Hodnota Null bude přidávána nově přidaným objednávkám do databáze a posléze bude této objednávce přiřazen zaměstnanec zodpovědný za plnění této objednávky. Tabulka

je propojena s tabulkami Zamestnanec a Zakaznici. V obou případech ve vztahu 1:N. Cizí klíče referující na tyto tabulky jsou sloupce „zakaznik_email“ a „zamestnanec_email“.

4.3.2 Tabulka Zamestnanec

Column Name	Data Type	Allow Nulls
email	varchar(50)	☐
heslo	varchar(50)	☐
prezdivka	varchar(50)	☐

Obrázek 16 - Tabulka Zamestnanec
Zdroj - Vlastní tvorba

Tabulka Zamestnanec je navržena v této jednoduché podobě, jelikož u tohoto konkrétního projektu není vyžadováno složitější struktury. Zaměstnanec vždy zůstává pouze zaměstnancem bez možnosti povýšení, tudíž není nutno tvořit případný sloupec pro pozici, případně tabulku pro ukládání pozic zaměstnance. Tabulka tedy obsahuje pouze nutné sloupce

4.3.3 Tabulka Zakaznici

Column Name	Data Type	Allow Nulls
email	varchar(50)	☐
jmeno	varchar(20)	☐
prijmeni	varchar(20)	☐
heslo	varchar(50)	☑

Obrázek 17 - Tabulka Zakaznici
Zdroj - Vlastní tvorba

Tabulka Zakaznici je obdobně jako tabulka Zamestnanec navržena v jednoduché podobě. Důvodem jednoduchosti návrhu je opět fakt, že data o zákaznících k ničemu nepotřebujeme. Jediné co potřebujeme je znát jejich email. Tabulka je navržena s možností držet hesla zákazníků. To je reprezentování sloupcem „heslo“, který dovoluje nulové hodnoty. Myšlenka je taková, že pokud se Zakaznik nezaregistruje, pak jeho konkrétní záznam bude mít heslo jako Null, v opačném případě heslo bude zaznamenáno. (TODO: salt + hash) Tabulka využívá faktu, že emailové adresy jsou unikátní, a tak je emailová adresa zvolená jako unikátní identifikátor zákazníků – tedy je primárním klíčem tabulky.

4.4 Tvorba tabulek databáze

```
CREATE TABLE [dbo].[Objednavky](
	[id_obj] [int] IDENTITY(1,1) NOT NULL,
	[datum_objednani] [date] NOT NULL,
	[starting_div] [varchar](20) NOT NULL,
	[ending_div] [varchar](20) NOT NULL,
	[cena_v_eur] [float] NOT NULL,
	[zakaznik_email] [varchar](50) NOT NULL,
	[zamestnanec_email] [varchar](50) NULL,
 CONSTRAINT [PK_Objednavky] PRIMARY KEY CLUSTERED
(
	[id_obj] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Objednavky]  WITH CHECK ADD  CONSTRAINT [FK_Objednavky_Zakaznici]
FOREIGN KEY([zakaznik_email])
REFERENCES [dbo].[Zakaznici] ([email])
GO

ALTER TABLE [dbo].[Objednavky] CHECK CONSTRAINT [FK_Objednavky_Zakaznici]
GO

ALTER TABLE [dbo].[Objednavky]  WITH CHECK ADD  CONSTRAINT [FK_Objednavky_Zamestnanec]
FOREIGN KEY([zamestnanec_email])
REFERENCES [dbo].[Zamestnanec] ([email])
GO

ALTER TABLE [dbo].[Objednavky] CHECK CONSTRAINT [FK_Objednavky_Zamestnanec]
GO


CREATE TABLE [dbo].[Zakaznici](
	[email] [varchar](50) NOT NULL,
	[jmeno] [varchar](20) NOT NULL,
	[prijmeni] [varchar](20) NOT NULL,
	[heslo] [varchar](50) NULL,
 CONSTRAINT [PK_Zakaznici] PRIMARY KEY CLUSTERED
(
	[email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

–

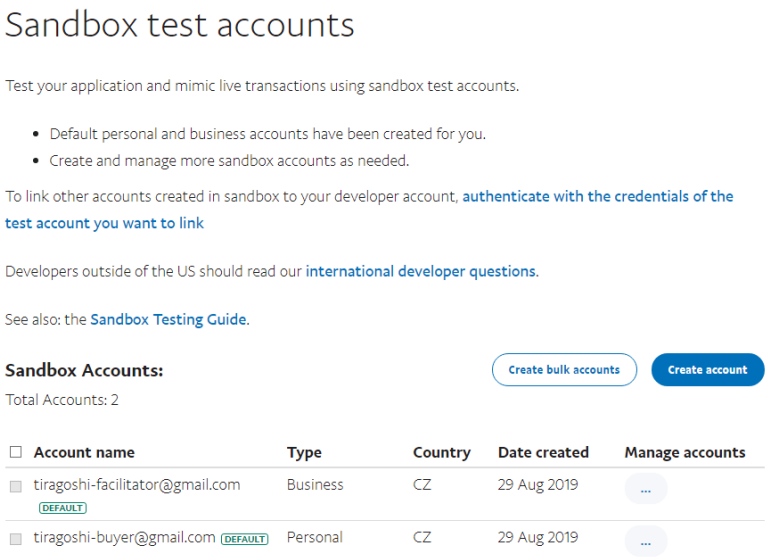
```
CREATE TABLE [dbo].[Zamestnanec](
	[email] [varchar](50) NOT NULL,
	[heslo] [varchar](50) NOT NULL,
	[prezdivka] [varchar](50) NOT NULL,
 CONSTRAINT [PK_Zamestnanec] PRIMARY KEY CLUSTERED
(
	[email] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON
[PRIMARY]
) ON [PRIMARY]
GO
```

4.5 Paypal

Platební brána projektu je zprostředkována přes platební bránu Paypal. Veškeré testování s platbami je taktéž vykonání za pomoci prostředků přímo od Paypalu. Konkrétně je využito služby IPN a Sandbox. IPN odešle při přijetí platby detaily platby, které jsou poté využity pro zaevidování dat do databáze.

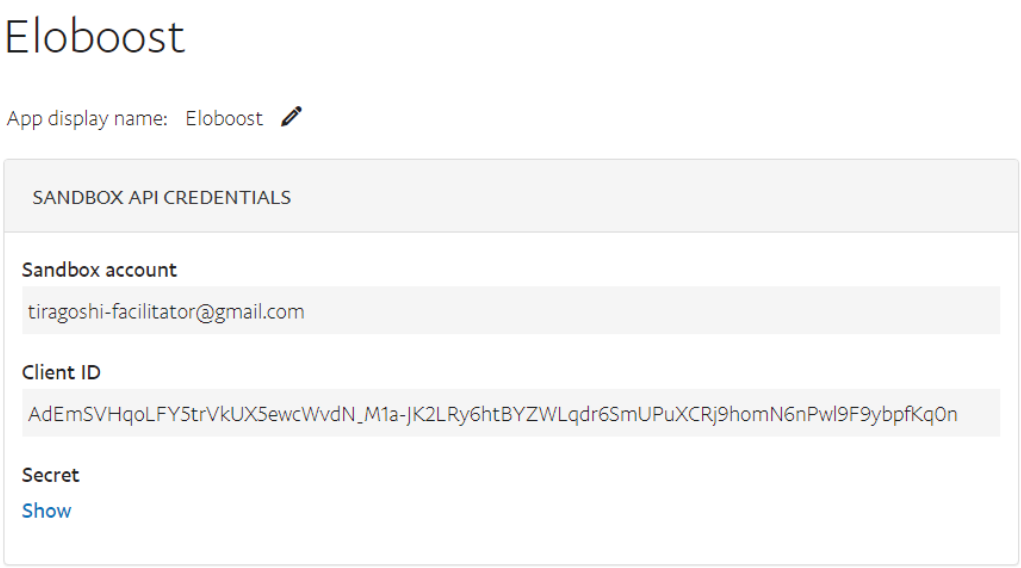
Sandbox platforma Paypalu umožňuje vytvořit testovací účty pro testování procesu plateb, získání dat a jejich následné uložení do databáze.

Pro vytvoření Sandbox účtů je potřeba mít základní Paypal účet. Poté navigovat do developer tools a pod kolonkou Sandbox je k nalezení odkaz na tvorbu účtů. Pro toto testování je potřeba mít účty dva – nakupujícího a účet, který slouží jako zprostředkovatel služby.



Obrázek 18 - Ukázka Sandbox účtů
Zdroj - Screenshot PayPal

Dále potřebujeme vytvořit Paypal aplikaci, která vygeneruje kód do zadání do kódu Paypal tlačítka.



Obrázek 19 - Sandbox Client ID
Zdroj – Vlastní tvorba, Sandbox [8]

4.6 Implementace Paypal IPN listeneru

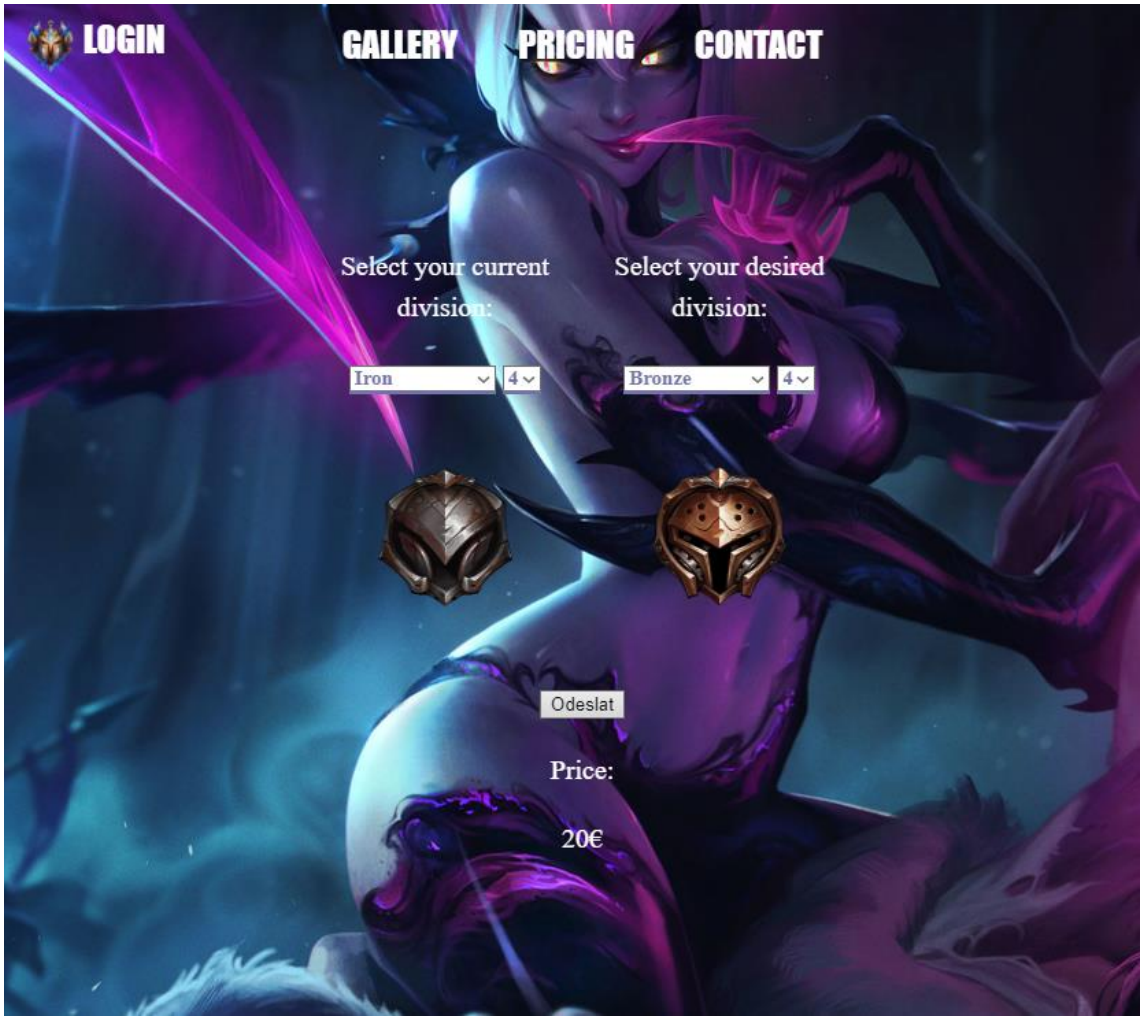
Po vygenerování potřebného kódu v předchozím bodu je potřeba vytvořit tzv. listener či handler. Tento PHP soubor je zodpovědný za přijímání dat z provedené transakce. V kódu se jedná o získání dat do jednoho textového řetězce, ze kterého je potřeba získat potřebná data pro pozdější zapisování do tabulek. Nejdříve je nutné připojit se pomocí níže uvedeného PHP kódu na server vytvořený v MS SQL Management Studio. Na serveru musí být vytvořený uživatelský účet, který slouží pro připojení do serveru a následnou manipulaci s daty tabulky. Při úspěšném připojení je poté možné posílat na databázi SQL dotazy.

```
$server_nm="DESKTOP-1GK65IO";
$connection = array("Database"=>"testxamp","UID"=>"xampxamp","PWD"=>"pass");
$conn=sqlsrv_connect($server_nm,$connection);
```

```php
<?php
$ch = curl_init();
curl_setopt($ch,CURLOPT_URL,'https://ipnpb.sandbox.paypal.com/cgi-bin/webscr');
curl_setopt($ch,CURLOPT_RETURNTRANSFER,1);
curl_setopt($ch,CURLOPT_SSL_VERIFYHOST,0);
curl_setopt($ch,CURLOPT_SSL_VERIFYPEER,0);
curl_setopt($ch,CURLOPT_POST,1);
curl_setopt($ch,CURLOPT_POSTFIELDS,"cmd=_notify-
validate&".http_build_query($_POST));
$response = curl_exec($ch);
curl_close($ch);
if($response == "VERIFIED" && $_POST['receiver_email']=="tiragoshi-
facilitator@gmail.com"){
    $date=$_POST['payment_date'];
    $mail=$_POST['payer_email'];
    $price=$_POST['mc_gross'];
    $item=$_POST['item_name'];
    $divs=explode("-",$item);
    $server_nm="DESKTOP-1GK65IO";
    $connection = array("Database"=>"testxamp","UID"=>"xampxamp","PWD"=>"pass");
    $conn=sqlsrv_connect($server_nm,$connection);
    if($conn){
        $zak = "INSERT INTO dbo.Zakaznici (email,jmeno,prijmeni) VALUES (?,?,?)";
        $zak_params = array($mail,"Jmeno","Prijmeni");
        $stmt=sqlsrv_query($conn,$zak,$zak_params);

        $ins="INSERT INTO dbo.Objednavky (datum_objednani,starting_div,ending_div,
cena_v_eur,zakaznik_email) VALUES (?,?,?,?,?)" ;
        $params=array(date("Y-m-d"),$divs[0],$divs[1],$price,$mail);
        $stm=sqlsrv_prepare($conn,$ins,$params);
        sqlsrv_execute( $stm );
        file_put_contents("db.txt",$params);
    }
    else{
        die(print_r(sqlsrv_errors(),true));
    }
    $handle = fopen("test.txt","w");
    foreach($_POST as $key => $value){
        fwrite($handle,"$key=>$value \r\n");
    }
    fclose($handle);
}
?>
```

4.7 Implementace platby pomocí PayPal

Platba v e-shopu je naimplementována v podobě výběru čtyři parametrů. Na stránce pricing.php jsou čtyři dropdown elementy. Po výběru validní kombinace těchto parametrů je uživateli umožněno kliknout na tlačítko „Odeslat“. Po kliknutí na toto tlačítko je uživatel přesměrován na stránku order-confirmation.php.

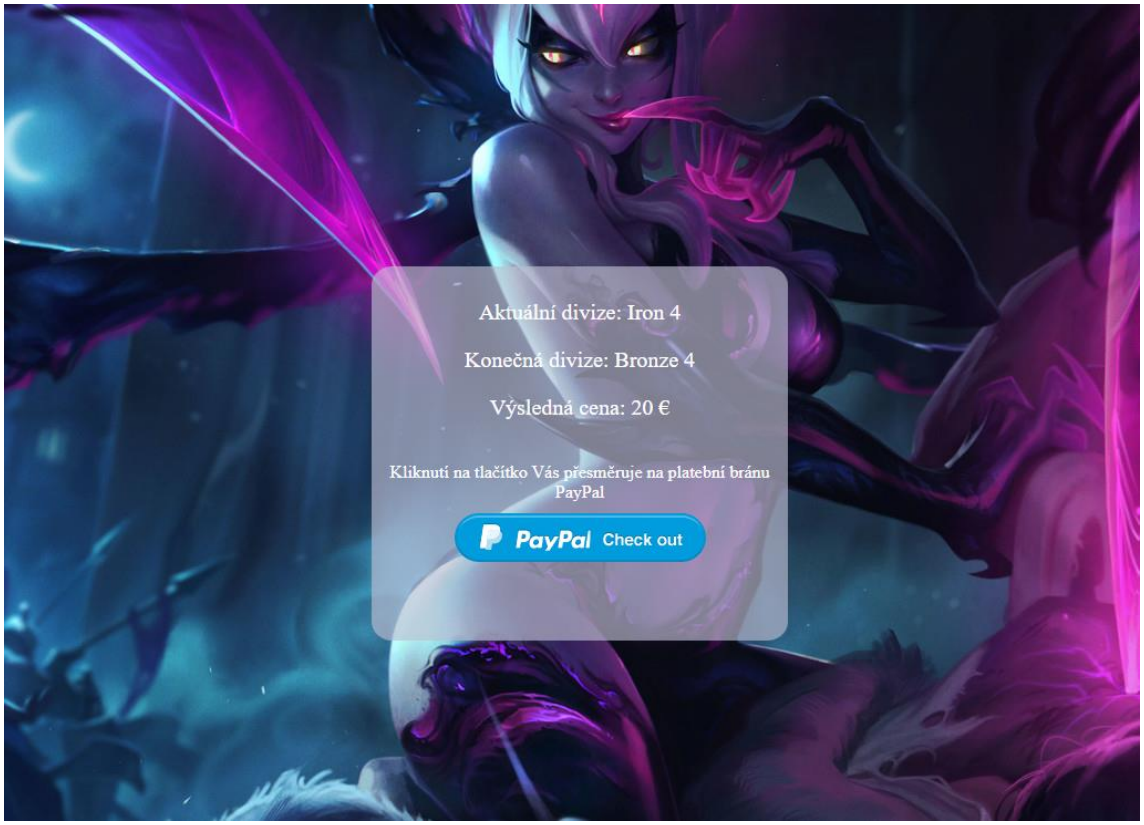


Obrázek 20 - Validní parametry, tlačítko Odeslat
Zdroj – Vlastní tvorba

Níže uvedený kód je zodpovědný za nastavení patřičných parametrů do pole, které je uloženo do $_SESSION proměnných. Za předpokladu že jsou zadány validní parametry je uživatel přesměrován na dříve zmíněnou stránku order-confirmation.php.

```php
 <?php
if (!isset($_SESSION)) {
session_start();
}
$d1 = 0;
$t1 = 0;
$d2 = 0;
$t2 = 0;
if (!isset($_SESSION["price"])) {
$_SESSION["price"] = null;
}
if (isset($_POST["divize1"])) {
$d1 = $_POST["divize1"];
}
if (isset($_POST["tier1"])) {
$t1 = $_POST["tier1"];
}
if (isset($_POST["divize2"])) {
$d2 = $_POST["divize2"];
}
if (isset($_POST["tier2"])) {
$t2 = $_POST["tier2"];
}

if (
isset($_POST["divize1"]) &&
isset($_POST["tier1"]) &&
isset($_POST["divize2"]) &&
isset($_POST["tier2"])
) {
  if ($d1 < $d2 || ($d1 == $d2 && $t1 < $t2)){
$_SESSION["division-data"]=array($d1, $t1,$d2 , $t2);
header("Location:order-confirmation.php");}
}

exit();
}
?>
```

Stránka order-confirmation.php obsahuje níže uvedené shrnutí o objednávce. V případě, že je uživatel připraven zaplatit, je mu umožněno kliknout na PayPal tlačítko.



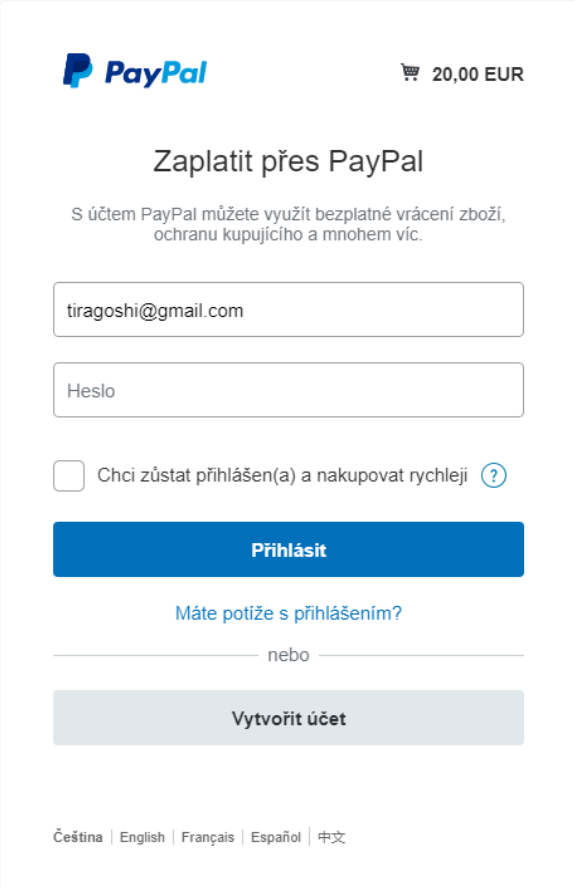
Obrázek 21 - Shrnutí objednávky
Zdroj – Vlastní tvorba

Níže uvedený kód je zodpovědný za vypočítaní správné ceny z hodnot v poli $_SESSION[“division-data“].

```
function calculatePrice($d1, $d2, $t1, $t2){
    $prices = [
        [5, 5, 5], [6, 6, 6], [7, 7, 7], [8, 8, 8],
        [9, 9, 9], [10, 10, 10], [11], [11], [11]];
    $price = 0;
    if ($d1 < $d2 || ($d1 == $d2 && $t1 < $t2)) {
        while ($d1 < $d2 && $d1 < 6) {
            while ($t1 < 3) {
                $price += $prices[$d1][$t1];
                $t1++;
            }
            $price += $prices[$d1][0];
            $d1++;
            $t1 = 0;
        }
        if ($d2 < 6) {
            while ($t1 < $t2) {
                $price += $prices[$d1][$t1];
                $t1++;
            }
        } elseif ($d2 > 6) {
            while ($d1 < $d2) {
                $price += $prices[$d1][0];
                $d1++;
            }
        }
        return $price;
    } else {
        return null;
    }
}
```

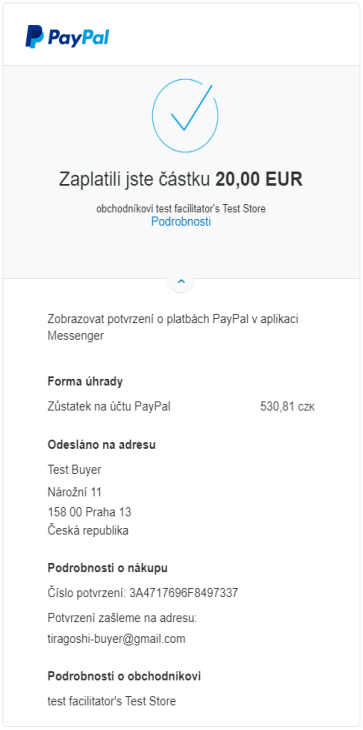
```php
<?php
if (!isset($_SESSION["division-data"])) {
    echo "<div><p>Nebyly zvoleny divize pro výpočet ceny </p><p>:(</p></div>";
} else {
    $divisions = array("Iron", "Bronze", "Silver", "Gold", "Platinum",
"Diamond", "Master", "Grandmaster", "Challenger");
    $div_data = $_SESSION["division-data"];
    $div_start = $divisions[$div_data[0]];
    $tier_start = 4 - $div_data[1];
    $div_end = $divisions[$div_data[2]];
    $tier_end = 4 - $div_data[3];
    $item_name = $div_start . " " . $tier_start . "-" . $div_end . " " .
$tier_end;
    $final_price = calculatePrice($div_data[0], $div_data[2], $div_data[1],
$div_data[3]);
    echo '<div id="price-confirmation">
        <p class="aligned-text">Aktuální divize: ' . $div_start . " " .
$tier_start . "</p>
        <p class='aligned-text'>Konečná divize: " . $div_end . " " . $tier_end
. "</p>
        <p class='aligned-text'>Výsledná cena: " . $final_price . " €</p>
        <br/>Kliknutí na tlačítko Vás přesměruje na platební bránu PayPal
        <br/><form method='post'><button name='paypal' id='pp-button'
type='submit'
src='https://www.paypalobjects.com/en_US/i/btn/btn_paynowCC_LG.gif'  alt='PayP
al . The safer, easier way to pay online.' ></button></form>
        </div>";
    $testmode = true;
    $paypalurl = $testmode ? "https://www.sandbox.paypal.com/cgi-bin/webscr"
    : "https://www.paypal.com/cgi-bin/webscr";
    if (isset($_POST["paypal"])) {
        $data = [
            "cmd" => "_xclick",
            "business" => "tiragoshi-facilitator@gmail.com",
            //'cancel_return'    =>'https://yourwebsite.com/index.php?page=car
t',
            'notify_url' => 'http://674c-78-102-141-188.ngrok.io/tnpw-
pahulak/listener.php',
            "currency_code" => "EUR",
            "item_name" => $item_name,
            "amount" => $final_price,
        ];
        header("location:" . $paypalurl . "?" . http_build_query($data));
        exit();
    }
}

?>
```

Výše uvedený kód zprostředkovává samotné vygenerování výsledného přehledu objednávky z obrázku 20. Dále je v tomto kódu zahrnuta samotná implementace platby pomocí PayPalu. Do pole $data jsou načtena potřebná data o objednávce, která jsou potom odeslaná na patřičnou PayPal platební bránu. V tomto případě se jedná o testovací rozhraní PayPal Sandbox. V tomto poli se objevuje hodnota 'notify_url'. Tato hodnota zodpovídá php souboru, listeneru, který byl naimplementován v předchozí kapitole 4.6. Jelikož se jedná o lokální soubor, je nutno využít softwaru ngrok, který umožní zpřístupnit lokální soubory na webu. Díky tomuto je možné, aby PayPal zaslal informace o provedené platbě do dříve zmiňovaného listeneru. Tento listener poté provádí funkci definovanou v kapitole 4.6.



Obrázek 22 - Přihlášení do PayPal
Zdroj - Vlastní tvorba, [8]

Obrázek 21 zobrazuje chování po kliknutí na platební tlačítko. Uživatel je přesměrován na přihlášení do svého PayPal účtu. Po přihlášení je uživateli zobrazen přehled a je mu umožněno zaplatit.

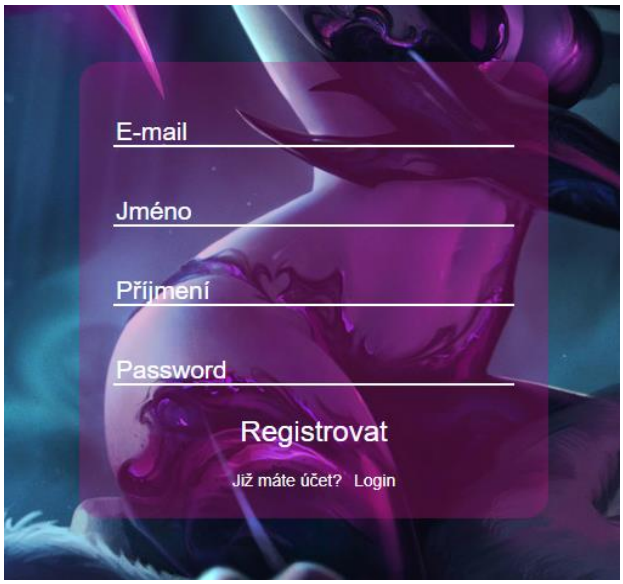


Obrázek 23 - detail po zaplacení
Zdroj - Vlastní tvorba, [8]

Obrázek 22 zobrazuje informace o úspěšně provedené platbě.

4.8 Registrace a přihlášení

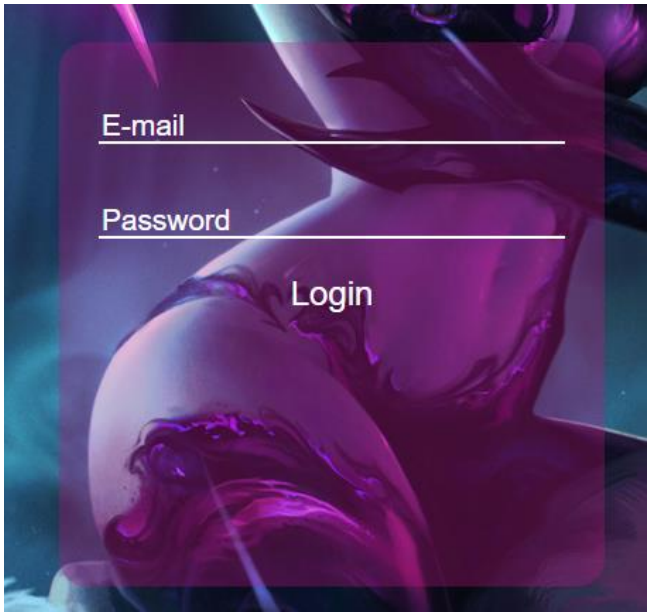
Registrace je implementována v níže uvedeném php kódu a vizualizována na obrázku 24.



Obrázek 24 - Registrační formulář
Zdroj - Vlastní tvorba

```php
<?php
session_start();
$server_nm = "DESKTOP-1GK65IO";
$connection = array("Database" => "testxamp", "UID" => "xampxamp", "PWD" =>
"pass");
$conn = sqlsrv_connect($server_nm, $connection);
if (isset($_POST["register"]) && $_POST["password"] != "" &&
$_POST["username"] != "" && $_POST["first_name"] != "" && $_POST["last_name"]
!= "") {
    $mail = $_POST["username"];
    $password = $_POST["password"];
    $first_name = $_POST["first_name"];
    $last_name = $_POST["last_name"];
    if ($conn) {
        $sql = "SELECT * FROM dbo.Zakaznici WHERE email='$mail';";
        $stmnt = sqlsrv_query($conn, $sql);
        if (!sqlsrv_has_rows($stmnt)) {
            $add = "INSERT INTO dbo.Zakaznici (email,jmeno,prijmeni,heslo)
VALUES (?,?,?,?);";
            $user_params = array($mail, $first_name, $last_name, $password);
            $stmt = sqlsrv_query($conn, $add, $user_params);
            if ($stmt) {
                $_SESSION["username"] = $mail;
                $_SESSION["pass"] = $password;
                $_SESSION["error"] = false;
                header("Location:account.php");
            }
        } else {
            $_SESSION["error"] = true;
            header("location:registration.php");
        }
    }
}
?>
```

Přihlášení je zprostředkováno v níže uvedeném kódu a vizualizováno na obrázku 25. Po zadání údajů je uživatel přesměrován na stránku account.php. V případě, že byla zadána validní data je uživatel přihlášen do svého účtu, jinak je přesměrován zpět na login stránku.



Obrázek 25 - Přihlašovací formulář
Zdroj - Vlastní tvorba

Historie vašich objednávek:

Datum	Startovací divize	Konečná divize	Celková cena
2021-01-26 00:00:00	Gold 4	Platinum 5	40
2021-01-26 00:00:00	Gold 5	Platinum 5	45

Logout

Obrázek 26 - Tabulka objednávek uživatele
Zdroj - Vlastní tvorba

```php
<?php
    session_start();
    if(isset($_SESSION["username"])&& isset($_SESSION["pass"])){
        if($_SESSION["username"] &&
$_SESSION["pass"])header("Location:account.php");
    }
    if(isset($_POST["login"])){
        $_SESSION["username"]=$_POST["username"];
        $_SESSION["pass"]=$_POST["password"];
        header("Location:account.php");
    }
?>
```

Níže uvedený kód má na starosti validace přihlašovacích údajů. Zároveň je zodpovědný za generování tabulky, která zobrazuje uživatelovi zakázky. Na úplném konci kódu je zaznamenáno odhlášení.

```php
<?php
session_start();
$username = $_SESSION["username"];
$pass = $_SESSION["pass"];
$server_nm = "DESKTOP-1GK65IO";
$connection = array("Database" => "testxamp", "UID" => "xampxamp", "PWD" =>
"pass");
$conn = sqlsrv_connect($server_nm, $connection);
if ($conn) {
    $sql = "SELECT * FROM dbo.Zakaznici WHERE email='$username' AND
heslo='$pass'";
    $stmnt = sqlsrv_query($conn, $sql);
    if ($stmnt) {
        if (sqlsrv_has_rows($stmnt)) {
            $row = sqlsrv_fetch_array($stmnt, SQLSRV_FETCH_ASSOC);
            echo "<p>" . "Vítejte na svém účtu, " . $row["email"] . "</p>";
            echo "<p>Historie vašich objednávek: </p>";
            $sql = "SELECT * FROM dbo.Objednavky WHERE
zakaznik_email='$username'";
            $stmnt = sqlsrv_query($conn, $sql);
            while ($row = sqlsrv_fetch_array($stmnt, SQLSRV_FETCH_ASSOC)) {
                $string = $row["datum_objednani"]->format('Y-m-d H:i:s');

                echo "<tr><td>" . $string . "</td><td>" . $row["starting_div"]
. "</td><td>" . $row["ending_div"] .
                    "</td><td>" . $row["cena_v_eur"] . "</td></tr>";
            }
        } else {
            session_unset();
            header("Location:login.php");

        }
    }
}
if (isset($_POST["logout"])) {
    session_unset();
    header("Location:login.php");
}

?>
```

5 Závěry a doporučení

Tato práce měla nastavené dva cíle. Prvním cílem bylo prozkoumat aktuálně nejvíce využívané relační databázové systémy. Hlavním zdrojem tohoto průzkumu se staly statistiky oblíbenosti developerskou komunitou a statistiky založené na vyhledávání určitého systému napříč webem. Z těchto zdrojů byly zvoleny čtyři relační databázové systémy, které byly navzájem porovnány.

Druhým a hlavním cílem práce byl kompletní návrh databáze malého webového e-shopu. Návrh byl poté implementován v Microsoft SQL Management Studiu. V práci byla poté vyvinuta dynamická webová stránka, která umožňuje uživateli zaregistrovat uživatelský účet, do uživatelského účtu se přihlašovat, provádět objednávky za pomocí platební brány PayPalu a jejich následné zobrazování v přehledu svého účtu.

Práci je možno využít jako podpůrný výukový materiál při studiu databázových systémů a základů jazyka PHP.

Oblast databázových systémů obecně je za posledních let s vývojem internetu pořád na vzrůstu a s dalším vývojem internetu bude oblast určitě nadále nabývat na důležitosti. Na základě této myšlenky je navrženo doporučení pro budoucí práce v podobě zkoumání alternativních databázových systémů.

6 Seznam použité literatury

[1] RIORDAN, Rebecca M. *Vytváříme relační databázové aplikace*. Praha: Computer Press, 2000. Databáze. ISBN 80-7226-360-9.

[2] ŠIMONOVÁ, Stanislava. *Databázové systémy I*. Pardubice: Univerzita Pardubice, 2013. ISBN 978-80-7395-702-5.

[3] LACKO, Ľuboslav. *Web a databáze*. Praha: Computer Press, 2001. Všechny cesty k informacím. ISBN 80-7226-555-5.

[4] OPPEL, Andrew J. *SQL bez předchozích znalostí: [průvodce pro samouky]*. Brno: Computer Press, 2008. ISBN 978-80-251-1707-1.

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

[6] KŘÍŽ, Jiří a Petr DOSTÁL. *Databázové systémy*. Brno: Akademické nakladatelství CERM, 2005. ISBN 80-214-3064-8.

[7] Funkce vnějšího spojení [online]. [cit. 2020-11-28]. Dostupné z: https://biportal.cz/en/sql-left-outer-join-joining-tables-in-sql-with-excel/

[8] Posílání peněz, placení online nebo založení účtu obchodníka – Paypal. [online]. Copyright 199 [cit. 31.01.2021]. Dostupné z https://:www.paypal.com/cz/home

[9] DOYLE, Matt. *Beginning PHP 5.3*. New York, USA: John Wiley & Sons, 2009. ISBN 0470413964.

[10] KOSEK, Jiří. *PHP - tvorba interaktivních internetových aplikací: podrobný průvodce*. Praha: Grada, 1998. Průvodce (Grada). ISBN 80-7169-373-1.

[11] *PHP - oficiální stránky* [online]. [cit. 2021-10-24]. Dostupné z: https://www.php.net/manual/en/history.php.php

[12] *W3Schools* [online]. [cit. 2021-10-29]. Dostupné z: https://www.w3schools.com/php/php_if_else.asp

[13] HUDSON, Paul. *PHP in a Nutshell*. USA: O'Reilly Media, 2005. ISBN 9780596100674.

[14] DUCKETT, Jon. *HTML & CSS: design and build websites*. Indianapolis, IN: Wiley, [2011]. ISBN 9781118008188.

[15] KOSEK, Jiří. *HTML: tvorba dokonalých WWW stránek : podrobný průvodce*. Praha: Grada, 1998. Průvodce (Grada). ISBN 80-7169-608-0.

[16] POUNCEY, Ian a Richard YORK. *Beginning CSS: Cascading Style Sheets for Web Design*. 3. Spojené království: Wrox Press, 2011. ISBN 978-0470891520.

[17] FLANAGAN, David. *JavaScript: the definitive guide*. Seventh edition. Sebastopol: O´Reilly, 2020. ISBN 978-1491952023.

[18] Db-engines. *Db-engines* [online]. [cit. 2021-10-31]. Dostupné z: https://db-engines.com/en/ranking_definition

[19] *Stack Overflow* [online]. [cit. 2021-10-31]. Dostupné z: https://insights.stackoverflow.com/survey/2020#technology-databases-all-respondents4

[20] *Stack Overflow* [online]. [cit. 2021-10-31]. Dostupné z: https://insights.stackoverflow.com/survey/2019

[21] *Stack Overflow* [online]. [cit. 2021-10-31]. Dostupné z: https://insights.stackoverflow.com/survey/2018

7 Seznam obrázků

8 Seznam tabulek