VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA PODNIKATELSKÁ
FACULTY OF BUSINESS AND MANAGEMENT

ÚSTAV INFORMATIKY
INSTITUTE OF INFORMATICS

NÁVRH SOFTWAROVÉ APLIKACE PRO TVORBU AUTOMATICKÉ NABÍDKY
DESIGN OF SOFTWARE APPLICATION FOR CREATION OF PRICE QUOTE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Josef Tesarčík**
AUTHOR

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

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA PODNIKATELSKÁ

Zadání bakalářské práce

Ústav:	Ústav informatiky
Student:	**Josef Tesarčík**
Studijní program:	Systémové inženýrství a informatika
Studijní obor:	Manažerská informatika
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 bakalářskou práci s názvem:

Návrh softwarové aplikace pro tvorbu automatické nabídky

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:

Tato bakalářská práce se zabývá návrhem softwarové aplikace pro společnost, jejíž účelem bude zefektivnění nabídkového systému.
Práce analyzuje předchozí verzi systému automatické nabídky a následně navrhuje nové konkrétní řešení, které bude realizované ve společnosti.

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.

PECINOVSKÝ, R. Myslíme objektově v jazyku Java: kompletní učebnice pro začátečníky. Praha: Grada, 2009. 570 s. ISBN 978-80-247-2653-3.

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

Termín odevzdání bakalářské 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

Tato bakalářská práce se zabývá návrhem softwarové aplikace pro společnost, jejíž účelem bude zefektivnění nabídkového systému. Práce analyzuje přechozí verzi systému automatické nabídky a následně navrhuje nové konkrétní řešení, které bude realizované ve společnosti.

Abstract

Bachelor thesis focuses on the design of software application for specific company for increase effectivity of supply system.

Thesis analyze precending version of price quote system and propose new conrete solution, which will be realized in the company environment.

Klíčová slova

Webová aplikace, datový model, funkční model, návrh

Key words

Web application, data model, funcional model, design

Bibliografické citace

TESARČÍK, Josef. Návrh softwarové aplikace pro tvorbu automatické nabídky [online]. Brno, 2020 [cit. 2020-05-17]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/127502. Bakalářská práce. Vysoké učení technické v Brně, Fakulta podnikatelská, Ústav informatiky. Vedoucí práce Petr Dydowicz.

Čestné prohlášení

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

V Brně dne 17. května 2020 ……

podpis autora

Poděkování

Obsah

Úvod

Součástí procesu péče o zákazníka je bezpochyby také poskytování cenových nabídek k specifickým produktům, nebo službám, která daná společnost nabízí.

V oboru, kterým se daná společnost zabývá je nabídkový proces často zdlouhavý a složitý. Je nutné tedy vytvořit systém zjednodušující administraci a vytváření samotných nabídek pro efektivní práci a konkurence schopnost. Není možné využít již hotové produkty právě kvůli komplexnosti dané problematiky.

Aplikace by měla pomoci pro rychlejší nabídkový proces, v budoucnu pak pro zákazníky, aby si sami mohli provést vlastní konfiguraci podobně jako je tomu například u automobilových konfigurátorů.

VYMEZENÍ PROBLÉMU A CÍLE PRÁCE

Cílem této práce má datový a funkční návrh pro software automatické nabídky, která bude dostupná jako webová aplikace.

V analýze problému se seznámíme s dosavadním řešením, poskytujícím základní informace o společnosti pro následný návrh. Podklady vycházejí ze zkušeností zaměstnanců a z dosavadního použití.

Aplikace by měla být intuitivní a uživatelky přívětivá, aby zjednodušila práci a časovou náročnost.

Postupy a metody použité při zpracování jsou následující:

- SWOT analýza
- Analýza případů užití
- Sémantický popis pomocí UML jazyka
- ERD diagram
- Přínosy

1 ANALÝZA SOUČASNÉHO STAVU

Tato kapitola poslouží k posouzení současného stavu systému pro tvorbu automatické nabídky a zároveň také ke zběžnému představení společnosti, pro kterou bude tato nová aplikace určená.

Na základě vstupních dat a informací získaných při této analýze bude vytvořen návrh nového nabídkového systému se všemi funkcionalitami, které jsou uvedené v předběžném zadání vytvořeným zodpovědnými zaměstnanci dané společnosti.

V první části bude, jak již bylo uvedeno, představená daná společnost, její organizační struktura a předmět podnikání. V části druhé přistoupíme k samotné analýze aplikace. Pomocí nástroje SWOT a dekompozicí použitých technologií, jejich výhod a nevýhod.

1.1 Představení společnosti

Společnost Smarteon systems s.r.o se zabývá implementací smart home systémů 3. generace od firmy Loxone.

Jedná se o rakouský systém pro domácí automatizaci, který se v ČR i jiných státech Evropy těší velké oblibě a Loxone jako takový má spoustu instalačních partnerů. Mezi výhody partnerství patří také přidělení stupně spolupráce. V současnosti jefirma Smarteon Systems jediná na Moravě s tzv. Flagship partnerstvím. Tedy s nejvyšším stupněm spolupráce.

Obrázek 1: Logo společnosti (Zdroj smarteon.cz)

Mimo vlastní chytré řízení domu se specializuje také na kvalitní počítačovou a kybernetickou bezpečnost navržených systémů. Dále pak na rozšiřitelnost Loxone řešení- Programátoři a systémoví integrátoři ve Smarteonu se snaží o propojení systému Loxone s co možná největším počtem technologií třetích stran, a to ať už za použití vlastních vytvořených softwarových a hardwarových nástrojů, nebo pomocí možností komunikace a implementace samotného Loxone rozhraní. Společnost vznikla v roce 2017 v Brně. Vlastní také pobočku v UAE, a to konkrétně v Dubaji, kde nabízí taktéž řešení chytré domácnosti.

Název společnosti: Smarteon Systems s.r.o.
Právní forma: společnost s ručením omezeným
IČO: 03677427
Datum vzniku:
Sídlo: Borky 618, 664 52, Sokolnice

1.2 Struktura společnosti

Struktura společnosti je v celku jednoduchá vzhledem k tomu, že se jedná o menší střední firmu.Ve společnosti je struktura hierarchicky rozdělená. Jednotliví členové týmu zastávají následující funkce:

Managing Director

Chief Technology officer

Project manager

Consultant, Security officer

Smart home engineer

Electro design designer

Back office support

IT network specialist

Lead software developer

Electrician.

1.3 Analýza současného stavu aplikace

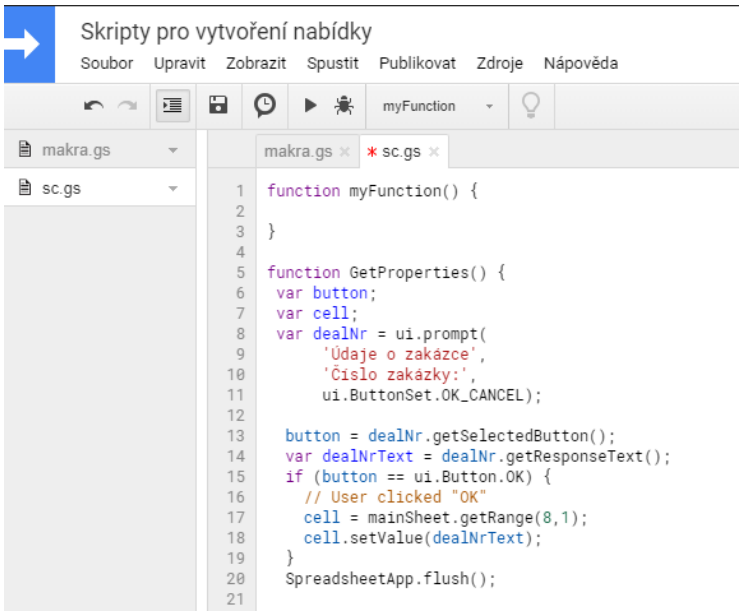
Předmětem této analýzy bude nabídkový systém, který slouží jako prvotní podklad pro rozhodovací proces klientů.

1.3.1 Prostředí systému

Nabídková platforma je vytvořena v aplikaci google documents, konkrétně tedy google sheets. Obsahuje část, kterou edituje uživatel – jedná se o dokument tabulek google a dále pak script v jazyce Google apps script (.gs)

Tato scriptovací platforma má základ v javascriptu a byla vyvinuta samotnými vývojaáři G Suite pro jednoduchý vývoj přidružených aplikací.

Editor google scriptu spolu s ladícím nástrojem se nachází přímo v aplikaci tabulky google a tudíž není třeba používat žádné IDE (Integration Development Enviroment) k jeho úpravě.



Obrázek. 2: Editor scriptu (Zdroj: Vlastní)

1.3.2 Architektura

Základ konkrétní nabídky tvoří vždy soubor google spreadsheet, s implementovanými makry, výpočtovými vzorci a samotným scriptem, který generuje finální nabídku ze zadaných dat.

1.3.3 SWOT analýza aplikace

Následující analýza vyplývá z dosavadních poznatků uživatelů aplikace a potřeb vzniklých v průběhu dosavadního používání při tvorbě nabídkových dokumentů ve společnosti Smarteon systems. Reflektuje výhody aplikace, ale také její slabiny, k jejichž řešení má sloužit právě tato práce. V neposlední řadě lze uvést příležitosti pro rozvoj a rozšíření funkčností.

SWOT analýza	
Strenghts • Jednoduchý přenositelný nástroj • Srozumitelný pro uživatele bez IT/elektro vzdělání	Opportunities • Rozvoj funkcionalit • Vývoj webové aplikace
Weaknesses • Doba generování nabídky • Omezená editace souboru	Threads • Ukončení programu G Suite

Tabulka 1: SWOT matice (vlastní tvorba)

Silné stránky:

Nabídkový formulář je snadno přenositelný, jedná se o google tabulku. Je také vázán s emailovou adresou, takže lze dohledat úpravy jednotlivých osob. Nespornou výhodou je intuitivnost práce i pro netechnicky vzdělané uživatele

Slabé stránky:

Generování samotné nabídky je poměrně zdlouhavé. Proces by šel urychlit optimalizací skriptu. Konkrétně použitím jiných datových typů a cyklických funkcí. Jedna z nevýhod samotného souboru je nemožnost rozsáhlejší editace buněk (např. posun kategorií v rámci listu) Script je vázán na konkrétní buňky a rozsahy tabulky. Opět je tento problém řešitelný optimalizací.

Příležitosti:

Příležitosti nabízející se pro vylepšení nástroje pro nabídky vyplynuly z dosavadního používání.

Je potřeba zakomponovat nové funkcionality, jako například generování podkladů pro projektanta na základě vytvořené nabídky. Dále můžeme uvažovat o propojení s dalšími firemními nástroji, jako je firemní CRM systém (Notion). Jednalo by se o validaci textových řetězců v předem daném formátu, představujících

Mezi další možnosti můžeme zařadit celkový redesign nabídkového systému, která by pak měla běžet jako webová aplikace s uvedenými funkčnostmi a postupným přidáváním nových vlastností a pokročilých funkcí. Výhodou bude nulové zatížení počítače uživatele, protože aplikace poběží na serveru.

Hrozby

Jediná potenciální hrozba, která by způsobila znemožnění použití již hotového řešení je ukončení podpory a provozování služeb G Suite společností Google. Vzhledem k množství uživatelů používajících tento nástroj jak na osobních počítačích, tak na mobilních zařízeních ovšem tento krok není pravděpodobný.

2 TEORETICKÁ VÝCHODISKA PRÁCE

V této kapitole bakalářské práce se zaměříme na teoretické pozadí pro jednotlivé použité služby a technologie, které jsou pro návrh webové aplikace klíčové. Slouží k vysvětlení souvisejících pojmů a také k základnímu popisu fungování daných služeb

Při tvorbě webových aplikací se musíme zamyslet nad tím jaké vývojové nástroje budeme používat pro jednotlivé části aplikace. Tyto části jsou tzv. Front-End, Back-End a také nástroje DevOps

2.1 Front-End technologie

Front End představuje část aplikace, která je viditelná uživateli. Tedy vizuální stránku, GUI (grafické uživatelské prostředí) a interaktivní prvky.

2.1.1 HTML

Termín HTML je zkratka pro Hypertext Markup Language. Jedná se o jazyk pro strukturování a tvorbu HTML dokumentů a webových stránek. Ty mohou obsahovat text, tabulky, obrázky atd. Dále je možné přistupovat k jiným dokumentům a stránkám pomocí hypertextových odkazů. [1]

Základní struktura

HTML dokument je tvořen elementy (tagy) a atributy, do kterých vkládáme hodnoty. Můžeme říct, že každý element obsahuje určité atributy. Struktura dokumentu je ohraničena tagy <html> na začátku a </html> na konci. Každý HTML dokument obsahuje tag <head> ten obsahuje titulek a další atributy, které prohlížeč používá při zobrazování dokumentu. Je to například použité kódování textu, odkaz na soubor se scriptem, který se na stránce spouští. V tagu <body> se nachází námi vložený obsah. Tyto elementy se nazývají strukturální tagy. [2]

```
<html>
<head>
<title>Barebones HTML Document</title>
</head>
<body>
This illustrates, in a very <i>simp</i>le way,
the basic structure of an HTML document.
</body>
</html>
```

Obrázek 3: Strukturállní tagy HTML dokumentu (Zdroj:[2])

Textové tagy

Pro to, aby dokument byl strukturovaný v rámci HTML dokumentu slouží množství nástrojů a tagů. Text se dá formátovat pomocí speciálních tagů

Division

Jak již z anglického názvu vyplývá, pomocí tohoto tagu lze stránku jednoduše rozdělit do několika sekcí. Postupem času s příchodem inovací v nových verzích HTML se tento tag začal využívat k označení individuálních sekcí dokumentu pomocí, u kterých jsou vyžadovány stejné vlastnosti. [2]

Odstavec <p>

Rozděluje text do odstavců. Opět lze pracovat s množstvím tributů jako je zarovnání, id, jazyk, atributy událostí nad daným odstavcem

Nadpis <h>

Nadpisy mohou být různých úrovní, a proto se k tagu vždy přidává číslo, znázorňující danou úroveň. Nadpisy dosahují až 6 úrovní <h1> – <h6>. Stejně jako ostatní elementy HTML jazyka mají několik atributů, které lze měnit.

Odkaz <a>

Tento element je vlastně klíčový, pomocí něj lze spojovat dokumenty napříč jedním, nebo vícero HTML dokumenty, popř. se lze spojit sestránkou/dokumentem kdekoliv na internetu. [2]

Odkazování mezi dokumenty je možné několika způsoby:

Absolutní URL adresa:

používá se, jestliže chceme odkazovat na konkrétní webovou stránku

Příklad: <a href =“https://seznam.cz/home.html“>

Relativní URL adresa:

při tvorbě webu určujeme cestu mezi adresáři, které jsou uložené na webovém serveru. Záleží tedy na tom, z jakého adresáře se na daný soubor odkazujeme. Na základě toho se relativní adresa mění. [2]

Příklad: <a href=“dir/home.html“>

Typů adres existuje více. Jedna z hojně používaných je adresa souboru:

Adresa souboru:

pomocí tohoto zápisu lze také přistupovat k souborům na lokálním zařízení, popř. i na libovolné webové adrese.

Příklad: file://home/john/document.html

Příklad: file://marketing.kumquat.com/monthly_sales.html [2]

Média

Webové stránky mohou samozřejmě obsahovat různé typy mediálních souborů, ať už se jedná o grafiku, zvuky, animace nebo videa. Webová stránka tak může tvořit rozhraní pro audio/video podcast, nebo fungovat jako interkativní obrazovka s navigačními ikonami. [3]

Video

Aby bylo možné integrovat video do stránky, je nutné jej poskytnout alespoň ve dvou formátech. Toto zajistí kompatibilitu s různými prohlížeči. HTML5 podporuje tyto 3 hlavní kodeky: **Og**g, **H.264**, **WebM**. Novější verze všech nejpoužívanějších prohlížečů taktéž. [3]

Příklad videostreamu, bez ovládacích prvků:

<video src="streamer.mp4"> </video>

Tag videa opět může obsahovat několik atributů – od ovládání, přes možnost opětovného přehrávání, až po rozlišení.

Audio

Stejně jako u videa, je potřebné zajistit, byl prohlížeč podporoval kodek audio souboru. Existuje 5 hlavních audio formátů: **Ogg**, **MP3**, **WAV**, **ACC**, **MP4**. Opět, je nutné, aby zdroj audia byl dostupný alespoň ve dvou formátech.

Příklad audio souboru, vloženého do webové stránky:

<audio src="myAudio.ogg"> </audio> [2]

Atributy audio tagu jsou obdobné jako u videa

Obrázky

Také při vkládání obrázků do webu musíme mít na paměti volbu správného formátu vzhledem k jeho kvalitě a velkosti. Chceme dosáhnout co největší kvality s co nejmenším objemem dat.

Běžné formáty jsou: **GIF**, **PNG** a **JPEG**. Právě JPEG je vhodný například pro fotografie. Oproti tom pro loga a plochy s menším množstvím barev jsou lepší formáty GIF a PNG, pro lepší kompresi větších jednobarevnách ploch.

Příklad tagu pro obrázek:

HTML nám nabízí velkou škálu možností editace textu, médií a tvorby strukturovaných webových stránek/dokumentů. Není možné v této práci obsáhnout Veškerou specifikaci jazyka. Proto se kapitola soustředila pouze na základní elementy.

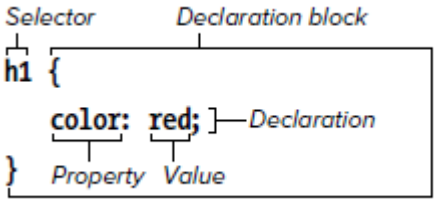
2.1.2 CSS (Kaskádové styly)

Kaskádové styly slouží k navržení celkového designu webové stránky. Poskytují tak větší kontrolu nad rozvržením celé stránky. První verze CSS1 se objevila v roce 1996 a hned o 2 roky později přišla verze CSS2. v roce 2005 byla uveřejněna zatím poslední verze CSS3 [4]

Kaskádový styl si můžeme představit zjednodušeně jako textový soubor, který obsahuje pravidla pro zobrazování jednotlivých elementů na webové stránce. [2]

Tvorba stylu

Každému stylu je v prvé řadě definovat dvě hlavní části. Těmi jsou: **Selektor**, ten určuje, které elementy budou ovlivněny daným kaskádovým stylem. Dále pak **Deklarace**, ve které je popsáno, jak bude výsledný styl vypadat. V bloku deklarace se vždy nachází určitá **vlastnost** (barva, zarovnání, tloušťka čáry) a k ní náležící hodnota. [2] Při volbě vhodného editoru html kódu jsou vám již hodnoty jednotlivých vlastností našeptávány a to buď textovou, hexadecimální nebo grafickou formou.



Obrázek 4: struktura CSS stylu (zdroj: [2])

Jednotlivé CSS pravidla je hodné okomentovat, popř. rozdělit podle toho k jaké části webové stránky se vztahují. Komentář také může sloužit k vypnutí některé vlastnosti, abychom si mohli ověřit, jak ovlivňuje vzhled stránky.

Načtení stylu do stránky lze provést několika způsoby:

1. Načíst styl z externího .css souboru ve stejném adresáři, kde je uložen .html soubor
2. Vložit konkrétní styl do hlavičky HTML dokumentu
3. Aplikovat styl přímo ve specifickém elementu [2]

Dědičnost

Výhoda CSS stylů je ta, že je lze vytvořit, jako tzv. třídu. Vlastnosti této třídy pak mohou html elementy dědit pouze tím, že je k dané třídě přiřadíme. Zde se projevuje vlastnost, podle které se styly také jmenují. O tom, jaký styl daný element dostane, rozhoduje kaskáda. Zde platí zákon specifičnosti. Čím více je daný selektror specifický, tím vyšší má prioritu při aplikaci daného stylu. Příklady si představíme na následujícím obrázku.

```
p {
    color: red;
}

p.group {
    color: blue;
}

p#last {
    color: green;
}

p#last {
    color: magenta;
}
```

Obrázek 5. tvoření tříd css stylů (zdroj: [2])

Nejnižší prioritu má element p, vyšší prioritu má element s uvedenou třídou tedy p.group. Pokud by například element obsahoval třídy dvě jeho priorita tím vzroste ještě více. Nejvyšší prioritu má element s atributem id: p#last. [2]

Na prioritu stylu má vliv také jeho samotné umístění v rámci HTML dokumentu. Pravidlo, které se vyskytuje v dokumentu později získává větší váhu.

Jestliže tedy máme pravidlo vyskytující se přímo v HTML elementu, mají větší váhu než styl uvedený v hlavičce stránky.

Poslední a nejvyšší prioritu má příkaz **inherit**, který spolu se znakem „!“ umístíme na konec daného pravidla.

Vlastnosti stylu a jejich hodnoty

Každá vlastnost CSS stylu má různá pravidla, pro hodnoty, kterých může nabývat. Jedná se např. o text, čísla, barvy v různých formátech, URL adresy a další.

Hodnotu vlastnosti lze z rodičovské třídy na potomka předat pomocí klíčového slova **inherit**.

2.1.3 Javascript

Tento jazyk je hlavním programovacím jazykem pro webové stránky. Využívají jej jak všechny prohlížeče, tak většina moderních webů. Slouží pro vytváření interakcí s uživatelem a definování chování webové stránky. Jedná se o vysokoúrovňový, objektově orientovaný jazyk, zaměřený na funkcionální programování.Prvotní syntaxe jazyka je odvozena z Javy. V současnosti existuje velké množství knihoven a frameworků pro možnost rozsáhlého webového vývoje. [6]

Javascript se vyznačuje tím, že se spouští přímo v prohlížeči klienta. V HTML dokumentu je definován tagy <script> </script>. [7]

```
<html>
  <head><title>Hello World</title></head>
  <body>
    <script type="text/javascript">
      document.write("Hello World")
    </script>
    <noscript>
      Your browser doesn't support or has disabled JavaScript
    </noscript>
  </body>
</html>
```

Obrázek 6. vnoření jednoduchého kódu v jazyce javascript do HTML dokumentu (zdroj [7])

Script lze také umístit přímo do hlavičky dokumentu. Což je ideální řešení pro případ, že chceme, aby se script spustil při načtení stránky. Pomocí této metod lze také zajistit zapisovaní tzv. meta tagů do hlavičky dokumentu. [7]

Další možností je k HTML dokumentu přidat soubor s příponou .js, ve kterém bude umístěn zdrojový kód. Ten můžeme načít i z libovolného zdroje kdekoliv na internetu

```
<script type="text/javascript" src="script.js"></script> ..


<script type="text/javascript" src="http://someserver.com/script.js">
</script>
```

Obrázek 7 příklad připojení místního a cizího javascrpit souboru (zdroj [7])

Frameworky Javascriptu

Framework neboli aplikační rámec je softwarová abstrakce, která slouží k jednoduššímu programování a řešení specifických problémů. Javascript má frameworků mnoho. Vyjmenujeme jen ty v současnosti nejznámější a nejpoužívanější.

Angular

Velmi výkonný framework, je open-source a používá se k vytváření webových aplikací na jednoduché HTML stránce (SPA) Single Page Application

React

Vytvořen firmou Facebook. Používá se k vývoji stránek s dynamickým uživatelským prostředím s velkou datovou režií.

Vue

Vhodný pro vytváření pokročilých SPA a také pro zavedení do projektů pracujících i s jinými .js knihovnami.

jQuery

Je hojně podporovaná velkým množstvím prohlížečů. Byla navržena pro zjednodušení HTML DOM stromového modelu. Je zdarma a open-source.

DOM

Jedná se o zkratku názvu Document Object Model (objektový model dokumentu). Slouží jako Aplikační rozhraní pro programování (API) a obstarává manipulaci a s obsahem HTML stránky. Reprezentace HTML dokumentu je v tomto případě stromová. Slouží jako prostředí mezi samotným dokumentem v objektové podobě a javascriptovým kódem.[7]

Javascriptový kód jako takový pak může:

- Přidat, změnit, nebo odstranit jakýkoli HTML element a atribut
- Změnit jakýkoli CSS styl
- Reagovat na vzniklé události a vytvářet nové

2.2 Backend Technologie

Tato část aplikace zajišťuje její skryté funckionality, komunikaci s databázovými servery např. pomocí jazyka SQL, dále její vlastní fungování, které navrhujeme v některém z dostupných jazyků vhodných pro webový vývoj. Patří mezi ně například: PHP, Java, Python, .NET (C#, VB) nebo Ruby. Mezi obory Backend vývoje můžeme zařadit také datové a funkční modelování

2.2.1 SQL

Jazyk SQL vznikl jako podpora pro nakládání a práci s daty v datových úložištích firem. Jeho úkolem bylo získat přístup k datům a dle zadaných dotazů a transakcí s daty manipulovat a vytvářet tak obchodní a ekonomický vhled do dat společnosti.

Jazyk SQL pracuje s tzv. relačním typem databáze, který stále patří hojně používanému modelu uspořádání dat při softwarovém vývoji. [8]

Databáze

Pod tím to termínem si můžeme představit systém shromažďující a organizující data. V souvislosti se softwarovým vývojem je ovšem myšlena relační databáze. Ta obsahuje zpravidla větší množství oddělených tabulek se vzájemnými vazbami s uloženými daty.

Dnes ovšem databáze využívá v podstatě každá aplikace a software pro ukládání jak aplikačních, tak uživatelských dat a práci s nimi. Tabulky jsou rozděleny z důvodu tzv. normalizace. Jedná se o proces rozdělení různých typů dat do oddělených tabulek, namísto shlukování dat do jedné tabulky, kde dochází k redundancím dat, nepřehlednosti a je tak těžké data měnit, aktualizovat. [8]

Pro správu databáze potřebujeme tzv. Database Management systém. Ten využíváme pro tvorbu dotazů v jazyku SQL a navrácení příslušných dat. Lze v něm také databáze přímo vytvářet. SQL používáme nejen k tvorbě dotazů nad databázemi jeho další důležité funkce jsou:

- **Definice dat:** Pomocí SQL definujeme strukturu a organizaci dat uvnitř databáze.
- **Načtení dat:** lze data uložená v databázi načíst a použít
- **Manipulace:** Data v databázi lze aktualizovat, mazat, přidávat, upravovat
- **Kontrola přístupu:** Uživatelům lze přidělit oprávnění ohledně nakládání s daty. Lze také kontrolovat neoprávněný přístup do databáze
- **Integrita dat:** SQL zaručuje integritu dat a chrání je tak proti poškození vlivem systémových chyb [9]

Výběr Databázového řešení

Databázových řešení existuje celá řada. Některá jsou úsporná, běží na osobním PC a ukládají data do jednoho souboru, jiná používají k funkčnosti databázový server a jsou rozsáhlá s možností obsloužení stovek uživatelů najednou.

Databázové systémy rozdělíme tedy do dvou kategorií: Úsporná a Centralizovaná.

Úsporné databáze:

Vhodné pro testování navržené aplikace, pro malý počet uživatelů. Samotná databáze je uložena v souboru, ke kterému mohou přistupovat ostatní uživatelé. Nevýhoda tohoto řešení se projevuje při paralelních úpravách souboru více uživateli.

Příklady: Microsoft Access, SQLite

Centralizované databáze:

Jestliže uvažujeme přístup k naší aplikaci od velkého počtu uživatelů/služeb zároveň, je vhodné nasadit tento typ databáze. Databázový systém zpravidla běží na serveru, ke kterému se připojují uživatelé pro získání dat.

Příklady: my SQL, Microsoft SQL server, Oracle, PostgreSQL

Základní operace nad databází

Uvedeme zde základní klíčová slova a pomocí kterých lze provádět operace se záznamy v databázi.

Select

Nejpoužívanější a stěžejní klauzule při tvorbě databázových dotazů a transakcí. Slouží k výběru dat z konkrétní tabulky.

Výběr tabulky provádíme pomocí klíčového slova From:

Select * from customer

Ve vráceném dotazu se nám objeví ty sloupce, které jsme uvedli v dotazu. Znak „* “ je ekvivalent pro veškeré sloupce tabulky. Chceme-li konkrétní sloupec, je nutné jej přidat ke klíčovému slovu select, např. takto:

Select customer_id, name from customer

V dotazu lze také provádět matematické operace nad jednotlivými sloupci.

Where

Slouží k filtrování v záznamu pomocí předem daného kritéria. Klíčové slovo přidáváme za klauzule SELECT a FROM například takto:

SELECT * FROM station_data WHERE year = 2010;

Klauzuli WHERE můžeme používat s různými datovými typy od BOLLEAN po text.

Agregační funkce

Vhodné pro vytváření celkových přehledů o sledovaných datech, součty, třídění podle specifikovaných požadavků. Mezi agregační funkce patří: Suma, Min, Max, Průměr, a Počet. Výsledky dotazu také můžeme seskupovat pomocí GROUP BY a seřazovat pomocí ORDER BY a to vzestupně nebo sestupně.

K agregačním funkcím se vztahuje také klíčové slovo Having. Slouží k filtrování záznamů právě agregovaných dat. V podstatě se jedná o náhradu Where, protože pomocí tohoto klíčového slova nemůžeme procházet agregovaná data, ale pouze jednotlivé záznamy v tabulkách

Join

Jedná se o klíčovou funkcionalitu definující použtí SQL. Slouží ke spojování tabulek a zobrazení dat mezi nimi.

Typy:
Inner Join, Lef Join, Right Join, Outer Join

2.2.2 Datové modelování a design databáze

Před samotným vytvářením databáze je důležité se zamyslet nad několika otázkami. Mezi ně patří nároky systému, pro který databázi navrhujeme, dále druhy a počty datových tabulek, návrh jejich normalizace a zamyšlení se na vztahu mezi nimi. Kromě těchto faktorů je důležité zvážit také datové nároky tabulek a uživatelů. Záleží na objemu dat vytvářeného v jednotlivých tabulkách, zdroji dat a nutnosti procesu automatizace úkonů vytváření tabulek a správy celé databáze. Další oblast, kterou je třeba se zaměřit je bezpečnost celé databáze. Musíme definovat typy uživatelů, jejich přístup k databázi a oprávnění, které můžeme rozdělit na zápis dat a čtení dat. V případě běžného uživatele bude mít pravděpodobně jen „read only“ přístup“. Tedy čtení dat. Důležité je také zabezpečení databáze pro weby a webové aplikace proti případným útokům.[7]

Tabulka

Každá tabulka musí obsahovat tzv. Primární klíč. Což je specifický řádek v tabulce, který každému záznamu přiřadí unikátní identitu. Pomocí primárního klíče můžeme propojovat tabulky relačními vztahy. Aby však bylo spojení korektní, musíme v propojovací tabulce uvést tzv. cizí klíč, který odkazuje na tabulku s primárním klíčem.

Vztahy mezi tabulkami

Vztahy mezi tabulkami jsou omezeny kardinalitou. Ta nám udává, kolik n-tic relací si vzájemně odpovídá. Vztahy známe celkem 4: [10]

Vztah 1-1:

Vztah 1-N:

Vztah N-M:

Nejběžnější reálný k bývá právě vztah N-M. Z logiky věci vyplývá, že nemůžeme vést vazbu mezi entitami a musíme tedy provést operaci přidání tzv. průnikové entity, která nám umožní použít vazbu 1-M. Tato entita obsahuje kandidátní klíče z obou předchozích tabulek.

Integrita modelu

Jedná se o stav, kdy data, která máme dostupná v modelu, odpovídají svými vlastnostmi běžných reálným objektům. Omezení jsou následující:

- Integritní omezení pro entity
- Integritní omezení pro vztahy [10]

Integritní omezení pro entity

Zde patří Doménová integrita, Entitní integrita a Referenční integrita.

Doménová integrita:

Hodnota atributů jednotlivých položek musí pocházet z množiny přípustných hodnot (domény). Je potřeba definovat samotnou doménu a pak její specifické atributy jako je: datový typ, povinnost zadání, jednoznačnost, rozsah a seznam přípustných hodnot.

Entitní integrita:

Každá relace musí obsahovat tzv. primární klíč, který je:

1. Jednoznačný: V relaci neexistuje entita se stejnými hodnotami.
2. Minimální: Není možné vynechat atribut, aniž bychom porušili první pravidlo.

Atributy primárního klíče musí vždy obsahovat hodnotu

Referenční integrita

Vztahuje se k cizímu klíči, který nám umožní vytvářet spojení mezi tabulkami. Cizí klíč musí být definován pod stejnou doménou jako primární klíč.

Normalizace

Tento proces je nutný pro úpravu datových struktur, aby splňovaly normalizační pravidla (formy). Ty jsou celkem 4 a vycházejí z požadavků na pro úsporné a efektivní ukládání dat. V tomto procesu dochází k odstranění redundancí dat, ale zároveň je zachována jejich integrita. Jestliže některé z normalizačních pravidel není dodržené, nelze považovat model za optimalizovaný. Prerekvizitou pro určitý stupeň normalizace je nutnost provedení normalizace na všech nižších úrovních. [10]

1. Normální forma: Všechny atributy musí být jednoduché, a nesmí být více hodnotové.

2. Normální forma: Všechny atributy tabulky musí plně záviset na primárním klíči a je splněna 1.NF.

3. Normální forma: Všechny atributy, které neobsahují klíč musí být vzájemně nezávislé a je splněna 1.NF a 2.NF

Boyce-Coddova Normální forma: Jedná se o obdobu 3.NF, je to speciální forma pro konkrétní složitější případy. Mezi kandidátními klíči nesmí být závislost a kandidátní klíče musí být v tabulce minimálně 2 a musí některé atributy sdílet. Jsou splněny předchozí formy.

4. Normální forma: V jedné relaci by nemělo dojít ke spojení nezávislých skupin, které se opakují. Jsou splněny všechny předchozí formy.

Speciální případ:

Existuje ještě 5. Normální forma pro cyklické závislosti a nimi související omezení. Jestliže máme tři relace a: R1 je spojena s R2, ta zase s R3, která je zpětně spojena s R1, je nutné, aby byly entity součástí stejného vektoru hodnot. Jestliže bychom ze spojení některou relaci vyřadili, mohlo b docházet k nesprávným výsledkům dotazů nad nimi. [10]

2.2.3 Funkční modelování

Problematika se zaobírá analýzou činností, které v systému probíhají. Jejich následnou algoritmizací a grafickým, popř. textovým znázorněním jejich průběhu. Jestliže popisujeme činnosti v sytému, je možné provést jejich hierarchický rozklad od obecných až po konkrétní. Při této dekompozici nejsme nijak omezeni počtem vnořování se na nižší úrovně. Úrovně se označují čísly od 1 výše, Nejnižší úroveň se nazývá elementární funkce. [10]

Pro funkční modelování procesů se používá jazyk UML - Unified Modelling Language. Jedná se o standart pro pro vizuální modelování procesů, analýzu, design a impelemetaci softwarových systémů. Je aplikovatelný v široké škále odvětví od bankovnictví, internet, zdravotnictví, letectví atd. [11]

Proces UML modelování je následující:

- Vedení a seřazení týmových aktivit v daném projektu
- Řízení úkolů jednotlivých vývojářů i celého týmu
- Měření a monitorování výsledků a aktivit projektu
- Vymezení vývoje klíčových částí

Pro popis činností používáme UML diagramy případů užití. Ty nám říkají, jak jednotliví uživatelé systému používají aplikaci nebo systém. Jedná se o strukturovaný grafický diagram používající několik typů grafických prvků pro odlišení objektů, aktorů asociací a vztahů.

Use case je zobrazován jako elipsa s názvem uvnitř. Osoba, která interaguje s jednotlivými případy užití se nazývá aktor a je zobrazována jako figurka. Účelem Use Case diagramu je popsat funckionality systému tak, jak to vidí uživatel. Měl by vypovídat o tom co bude budoucí systém umět, nikoli o tom, jak budou činnosti prováděny

Typy Use Case diagramů:

Bussines use case diagram: lze pomocí něj reprezentovat bussines funkce, procesy a aktivity v modelovaném prostředí.

System use case Diagram: Slouží ke specifikování externích požadavků a použití systému pro zachycení toho co by měl dělat, návrh funkcionalit systému.

2.3 Vývojové prostředí

Vývojové prostředí neboli IDE je potřebné pro tvorbu kódu aplikace a jejich funkcionalit. Poskytuje nástroje pro úpravu a ladění kódu a editaci souborů potřebných pro chod aplikace.

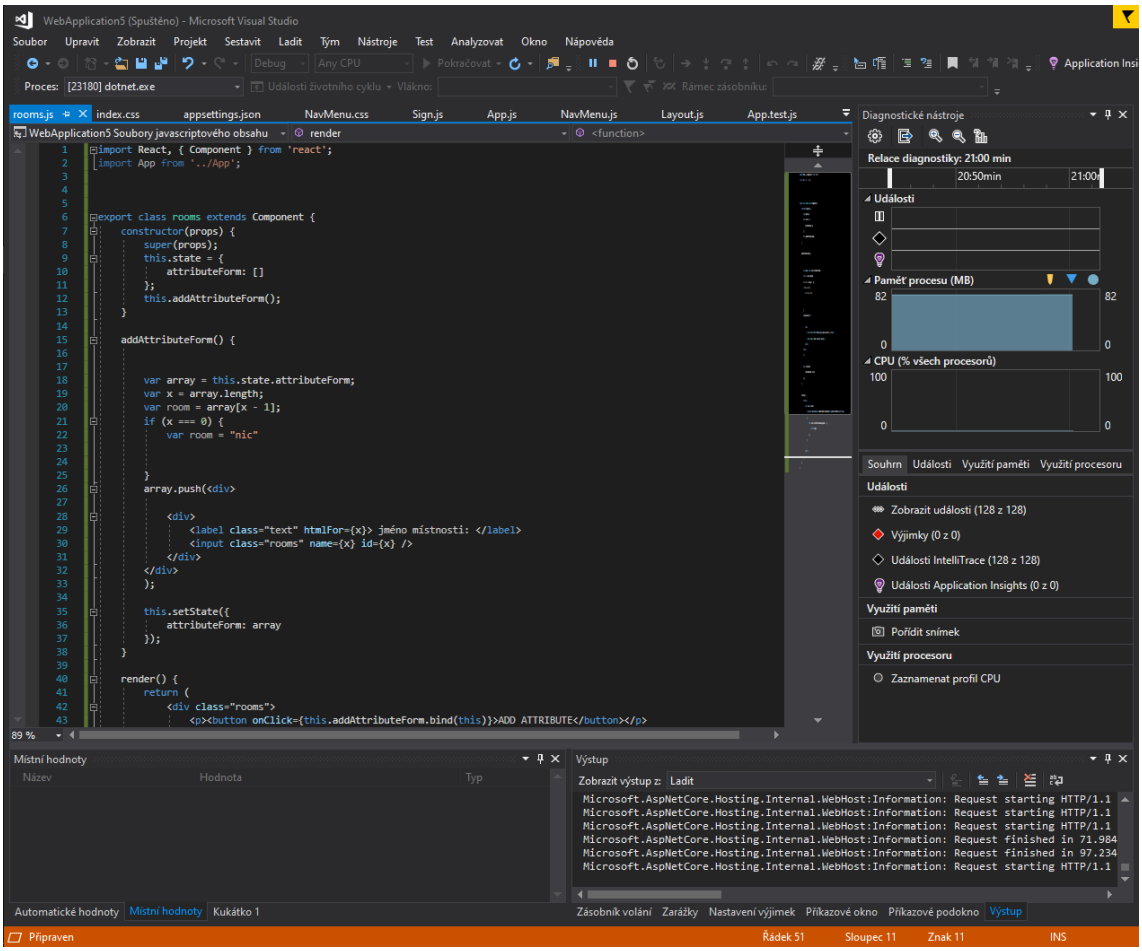
Visual Studio je vývojové prostředí od firmy Microsoft, které je velmi bohaté na možnosti použití různých technologií a programovacích jazyků pro vývoj software. Kromě editoru a ladícího progamu obsahuje tato sada také kompilátory, nástroje pro automatické dokončení kódu, grafický návrh a několik dalších funkcí, usnadňujících vývoj softwaru.

Visual studio je k dispozici jak pro windows tak i pro Mac poslední verze je z roku 2019 s ukončením podpory v roce 2028. Visual studio existuje v několika edicích rozlišených cenou a funkčností. Nazývají se Community, Professional a Enterprise. Verze Enterpise obsahuje pokročilé metody ladění a testovací nástroje.

Mezi základní podporované programovací jazky patří: C++, C#, framework .NET v několika verzích, Visual Basic. Sada je rozšiřitelná o nástroje pro vývoj mobilních aplikací, vývoj her, unixových aplikací, framweworky javascriptu, python aplikace a jeho webové frameworky. Lze také vyvíjet pro IoT. [12]

Mezi užitečné funkce mimo výše zmíněné patří: upozornění na chyby v kódu pomocí jeho podtržení červenou barvou a oknem s dodatečnou informací, související s možnou chybou. Další užitečná součást aplikace se jmenuje „vyčištění kódu“ a poskytuje možnosti opravy kódu, než dojde k jeho kontrole. Neméně důležitá funkcionalita zvaná refactoring, která nám pomůže inteligentně přejmenovávat proměnné, extrahovat kód mezi metodami a měnit parametry metod. (týká se jazyka C#). Sada funkcí IntelliSense zobrazuje informace o sadě kódu přímo v editoru a není nutné tedy dodatečné hledání informací jinde.

Základním prvkem při tvorbě je tzv. řešení. Obsahuje vše potřebné pro tvorbu kódu/ aplikace v daném programovacím jazyce, popř. frameworku jazyka. Jedná se v podstatě o kontejner k uspořádání jednoho nebo více projektů. [12]



Obrázek 7. Vývojové prostředí Visual Studio (vlastní tvorba)

2.4 DevOps

V této kapitole se zaměříme na nástroje pro správu a verzování kódu a souborů. Pro udržování kódu a orientaci v provedených změnách je nutné používat některý z verzovacích nástrojů. Nástroje existují v několika typech: S lokální správou, centralizovanou správou a distribuované systémy.

Lokální správa: V počítači uživatele existuje systém uchovávající v databázi veškeré změny souborů.

Centralizovaný systém správy: Obsahují serverovou část, kde jsou uchovávány všechny verzované soubory. Z tohoto úložiště si potom jednotlivé soubory uživatelé mohou stáhnout.

Distribuované systémy: Uživatel nestahuje jen nejnovější verze souborů, ale každý počítač obsahuje kompletní repozitář souborů. Při kolapsu serveru lze data obnovit z kteréhokoli stroje. Zde patří Git, jeden v současnosti nejpoužívanějších verzovacích systémů. [13]

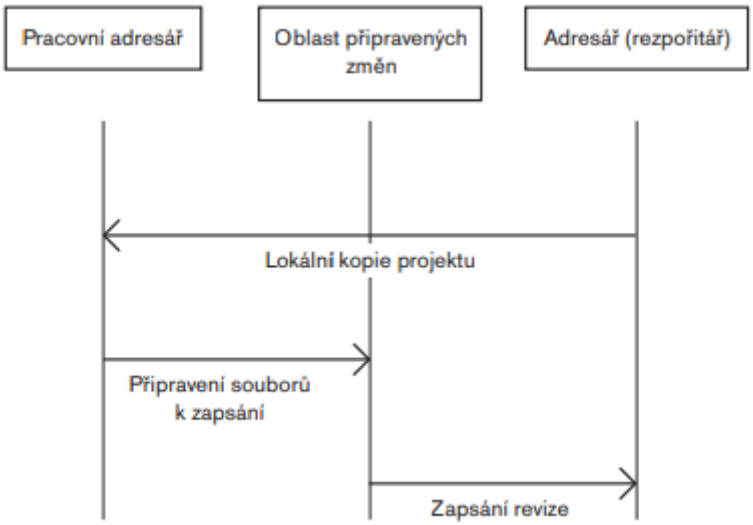
2.4.1 Git

Vznikl v roce 2005, za pomocí vývojářů Linuxových systémů na popud zpoplatnění dosud používaného systému bitKeeper. Požadované vlastnosti byly: rychlost, jednoduchý design, nelineární vývoj, distribuovatelnost, správa velkých projektů (Např. Linuxové jádro). [14]

Základy systému

Zpracování dat si můžeme představit jako jakési snímkování souborového systému. Při každém zápisu do souboru, dojde ke „snapshotu“. Než dojde k jakémukoliv uložení dat, je proveden kontrolní součet, který charakterizuje každou provedenou operaci. Využívá mechanismu SHA-1 (hashovací algoritmus) v podobě řetězce o 40 hexadecimálních znacích. Příklad: „24b9da6552252987aa493b52f8696cd6d3b00373“. [13]

Git používá tři základní stavy: zapsáno (commited), změněno (modified) a připraveno k zapsání (staged).



Obrázek 8. Stavy souborů v systému Git (zdroj [14])

V gitu se orientujeme pomocí příkazů v Linuxové příkazové řádce. V případě, že disponujeme operačním systémem Windows, je možné git provozovat v aplikaci Windows subsystem for Linux.

Pro serverovou část systému pak slouží velké množství hostingových služeb. Nejznámější je zřejmě GitHub, dále GitLab z freeware systémů, které mohou běžet na Vašem serveru je to například Gitea.

3 VLASTNÍ NÁVRH ŘEŠENÍ

V této části bakalářské práce nastíním možný návrh designu a také částečně fungování webové aplikace pro tvorbu nabídky

3.1 Architektura aplikace

Aplikace by měla používat SPA (Single Page Architecture) architekturu. Jedná se o moderní styl vytváření aplikace, kdy je důraz kladen na klientskou část. Chování aplikace je naprogramováno v některém z frameworků javascriptu. Pro tuto aplikaci např. navrhuji javascriptový framework React, který je open source a používá virtuální DOM, který je rychlejší než reálný. Komunikace se serverem bude probíhat pomocí web API a JSON.

3.1.1 GUI

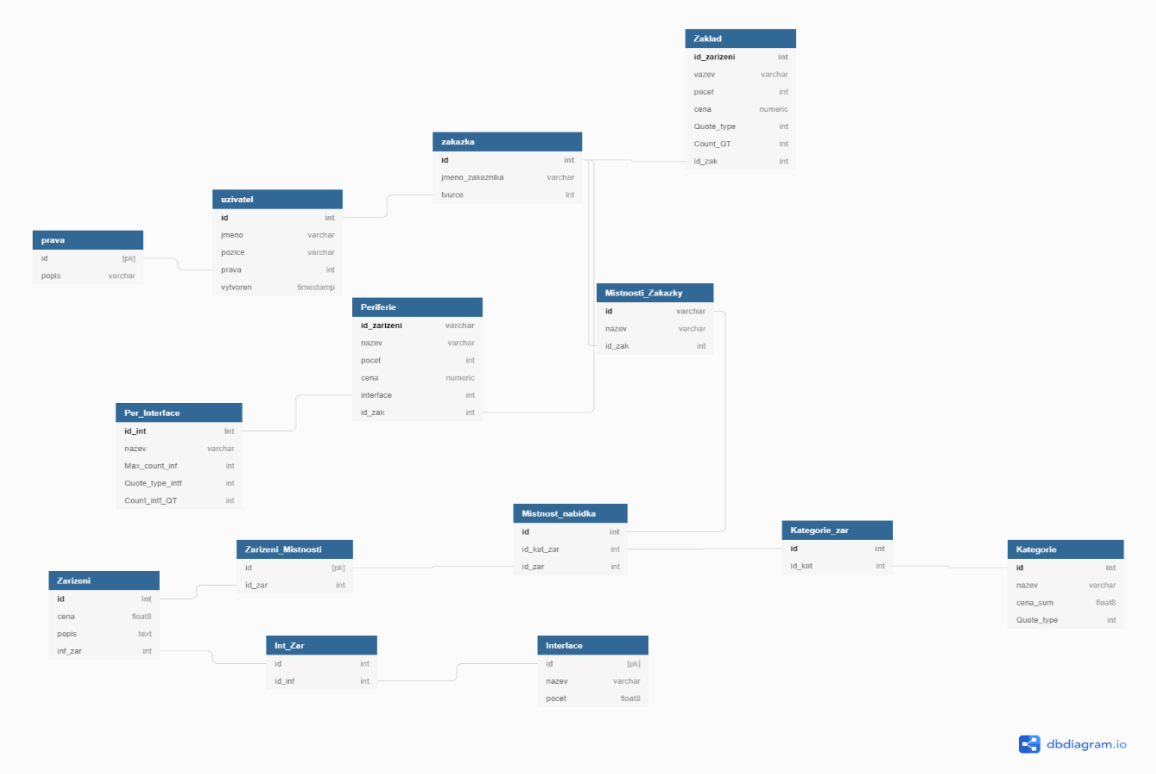
Návrh uživatelského rozhraní je důležitý pro dobrou orientaci a pohodlí uživatele představíme si tedy návrh jednotlivých sekcí budoucí aplikace.

Aplikace bude mít několik nezávislých částí. Některé budou navrženy pro interakci s uživatelem a jiné budou pouze zobrazovat data.

Přihlášení Bude probíhat pomocí SSO (Sinle Sign On) přes firemní google účet.
V případě rozšíření aplikace pro zákazníky bude dodán přihlašovací modul.

3.2 Návrh databáze

Pro fungování aplikace je potřeba navrhnout relační databázové schéma, které bude odpovídat potřebám aplikace. Jako typ databáze je zvolen MySQL vhodný pro webové aplikace vzhledem ke své rychlosti.



Obrázek 9. Databázový diagram (vlastní tvorba)

Při tvorbě databáze byly dodrženy normalizační pravidla pro její korektní funkčnost. K migraci dat z a do databáze budeme taktéž používat javascriptový framework react.

3.2.1 Tvorba datového modelu

Probíhala v nástroji dbdiagram.io . K návrhu můžeme využít také prostředí microsoft sql serveru, nebo nástroje pro tvorbu databází. Výhodou této platformy je to, že je online a lze z ní vyexportovat několik typů souborů, připravených pro nasazení databáze. formáty pro export jsou: postgreSQL, MySQL, a SQL Server

```
Table uzivatel as U {
  id int [pk, increment] // auto-increment
  jmeno varchar
  pozice varchar
  prava int [ref: > P.id]
  vytvoren timestamp

}


Table zakazka as Z{
  id int [pk, increment]
  jmeno_zakaznika varchar
  tvurce int [ref: > U.id]
 }

Table prava as P {
   id [pk]
   popis varchar
 }

Table Periferie as Pe {
   id_zarizeni varchar [pk]
   nazev varchar
   pocet int
   cena numeric
   interface int [ref:> PI.id_int] //rozhrani napr relay ext, DI ext,

   id_zak int [ref:> Z.id]
 }
Table Per_Interface as PI {
  id_int Int [pk]
  nazev varchar
  Max_count_inf int
  Quote_type_intf int // typ  nabidky
   Count_intf_QT int //pocet daneho rozhrani pro typ nabidky
}
```

Obrázek 10. Tvorba tabulek pomocí SQL editoru (vlastní tvorba)

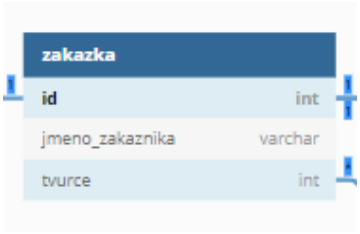
Náš datový model se skládá s několika druhů entit:

1.Základní entity:

2. Pomocné entity

3. Personalizační entity

Základní entita tohoto databázového modelu je tabulka Zakázka, její primární klíč je číslo zakázky, které je ve firemním procesu přiřazeno každé nové zakázce.



Pomocné entity jsou entity místností, periferií a zařízení, které si uživatel vybírá v nabídce.

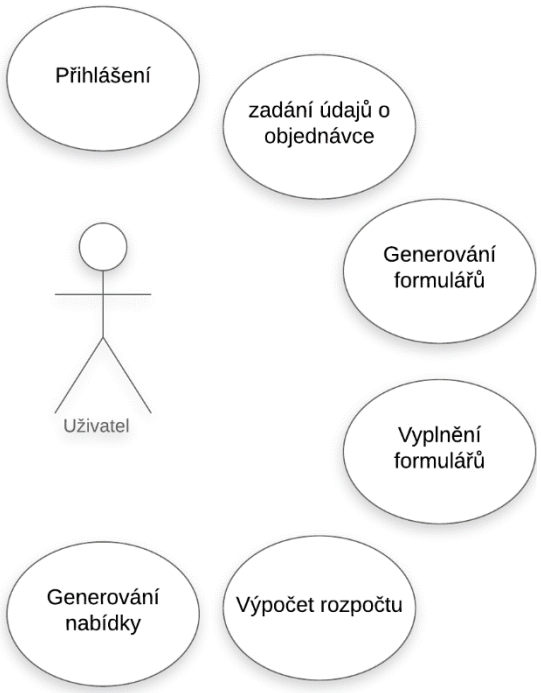
Personalizační entitou můžeme nazvat tabulku uživatel, která shromažďuje data o přihlášeném uživateli.

Všechny tabulky datového modelu jsou uvedeny v příloze ve vygenerovaném konfiguračním souboru.

3.3 Případy užití

V této podkapitole uvedeme případy užití aplikace. K jejich popisu slouží Use case Diagram. Popisuje činnosti, které budou vykonávány jednotlivými typy uživatelů V podstatě se v základní verzi bude jednat o dva typy uživatelů, a to bude tvůrce nabídky – zaměstnanec společnosti a administrátor aplikace.

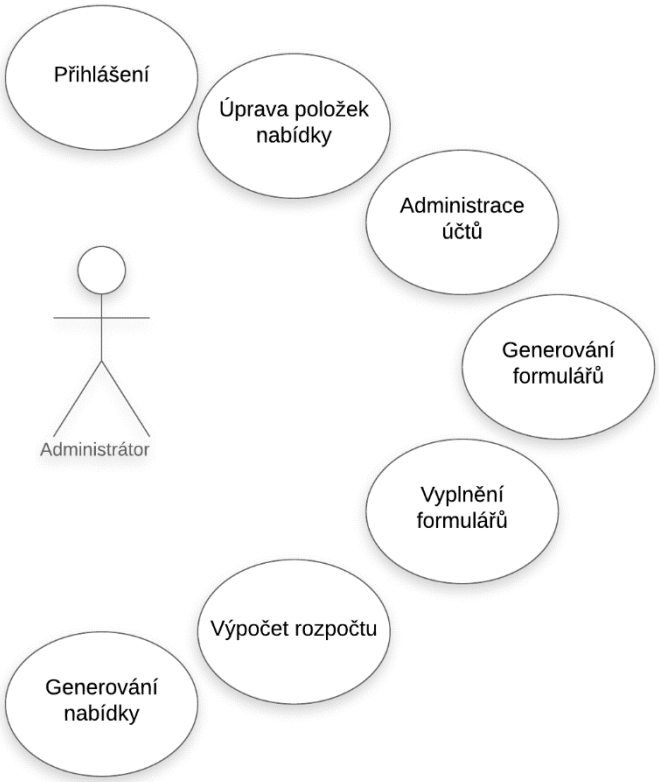
Zaměstnanec:



Obrázek 11. Use case diagram (vlastní tvorba)

Zaměstnanec má oproti adminovi nižší pravomoce jen v tom, že nemůže konfigurovat a upravovat nabídkové formuláře. Použití aplikace v jeho případě slouží opravdu pro tvorbu nabídek a rozpočtů.

Administrátor:



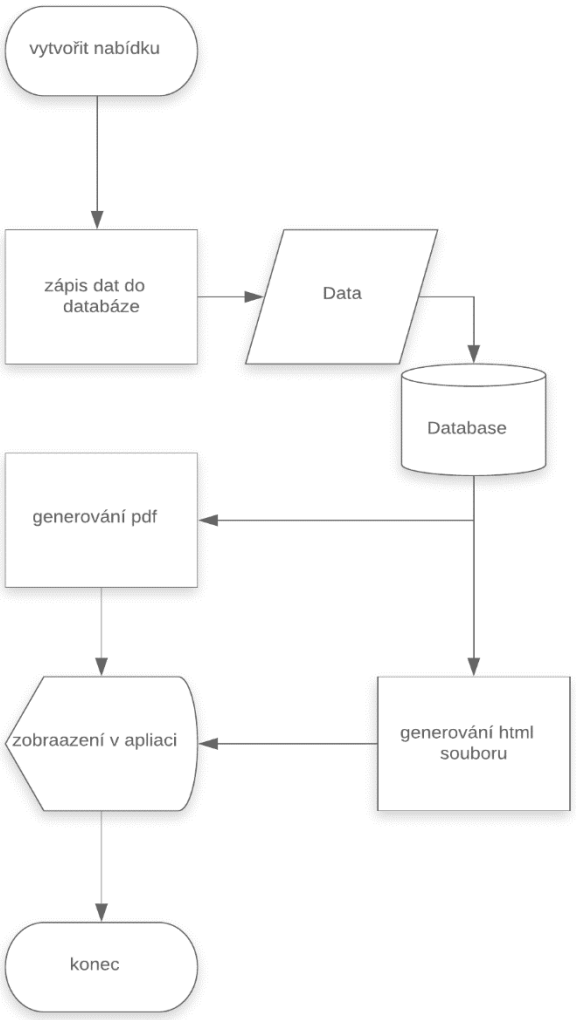
Obrázek 12. Use case diagram (vlastní tvorba)

Úkolem administrátora je kromě úkonů společných s uživatelem také administrovat uživatelské účty, upravovat samotnou nabídku a její databázi.

3.4 Funkční modelování: vývojové diagramy

Abychom mohli analyzovat chování aplikace a jejích součástí používáme vývojové diagramy pro grafické znázornění algoritmizace.

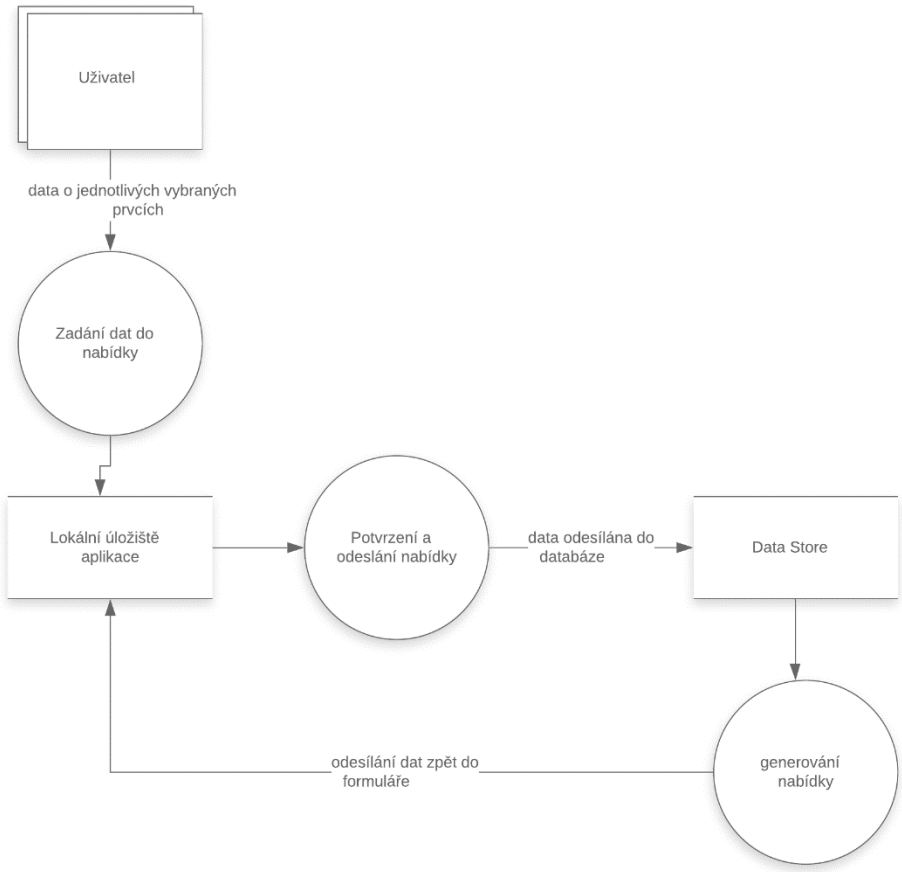
Vývojový diagram pro generování místností



Obrázek 13. Vývojový diagram části aplikace pro generování obsahu (vlastní tvorba)

DFD data flow diagram

Tento diagram slouží k analýze toku dat v aplikaci. Poskytuje informace u vstupních a výstupních datech jednotlivých modulů aplikace.



Obrázek 14. DFD diagram toku dat (vlastní tvorba)

Data v aplikaci využívají 2 typy úložiště. Pro práci s daty při samotném vytváření nabídky. Když uživatel listuje jednotlivými formuláři místností, jistě není žádoucí, aby se data mazala a proto zůstávají v tzv „local storage“ což je paměť prohlížeče.

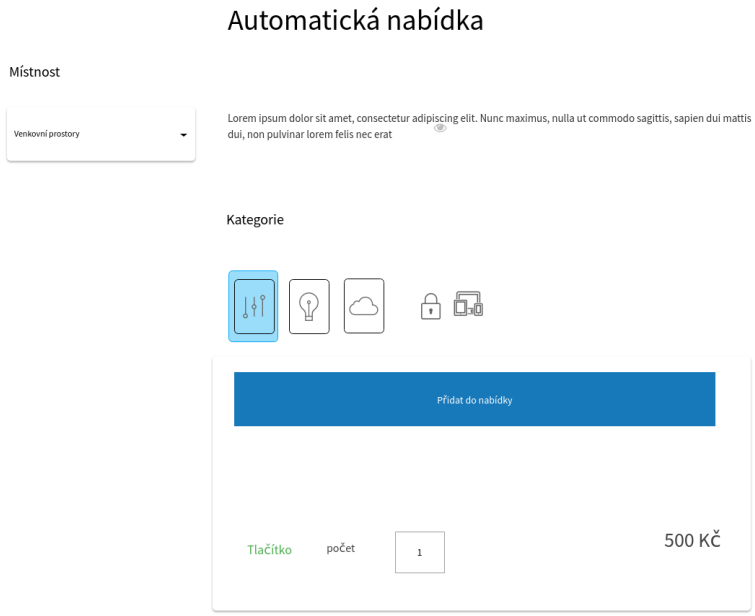
Prohlížeč takto může uložit opravdu velké množství dat. V závislosti na typu až 1 GB. Což je pro naše účely položka -> hodnota, dostačující.

Druhým typem úložiště je samotná databáze, která slouží k obnovení dat zakázky do aplikace, nebo ke generování nabídky v různých formátech

3.5 Design aplikace

Aplikace bude obsahovat několik samostatných modulů. Je třeba oddělit tvorbu nabídky od jejího zobrazení a také zobrazení rozpočtů bude mít vlastní modul.

Návrh nabídkového modulu:

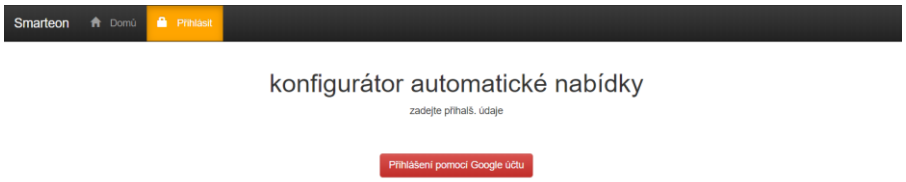


Obrázek 15. Wire frame návrh konfigurátoru (vlastní tvorba)

Návrh je inspirován automobilovými konfigurátory, kde uživatel kliká mezi jednotlivými kategoriemi. Po skončení konfigurace je mu zobrazen souhrn, ze kterého si pak může vygenerovat nabídku.

3.6 Ukázka části aplikace

Bude se jednat o webovou aplikaci, spouštěnou v libovolném prohlížeči a nezávislou na operačním systému.



Obrázek 16. Prvotní návrh GUI rozhraní apliakce (vlastní tvorba)

Design by měl být jednoduchý a intuitivní pro přehlednou práci. V aplikaci budou následující moduly:

- přihlášení
- zadání místností v domě pro tvorbu dílčí nabídky
- modul pro zobrazení finální nabídky
- rozpočtový modul

Navigace bude probíhat pomocí lišty v horní části aplikace. Díky integraci Bootstrapu, což je volně dostupná sada nástrojů pro tvorbu CSS stylů, bude aplikace plně responzivní i pro menší rozlišení.

3.7 Přínosy

Mezi přínosy nově navrženého systému bychom mohli uvést lepší výpočty pro jednotlivé položkové rozpočty. Dále přibudou funkce pro efektivní tvorbu a rozvržení a přiřazení prvků v rámci rozvaděče a jeho částí.

Další významný bod spatřuji v celkovém zrychlení běhu aplikace a generování nabídky. Momentálně dochází ke generování finální nabídky pomocí dat a hodnot z jednotlivých listů google sešitu. Kopírování nabídkového textu a hodnot zároveň je velmi časově náročné.

Je dobré se také zaměřit na designovou stránku budoucího software. Vize nabídky přístupné klientům na webu bude působit reprezentativně a může se tak jednat o konkurenční výhodu.

Správně navržené GUI pomůže k rychlému a efektivnímu procesu tvorby nabídky, který momentálně trvá i větší počet hodin. Zaměstnanec tak bude moci vytvořit více nabídek a oslovit více potenciálních zákazníků, kteří budou mít o smart home systém zájem.

ZÁVĚR

Cílem této práce bylo navrhnout datovou a funkční strukturu aplikace pro tvorbu automatických nabídek. Na základě analýz provedených v první části práce jsme mohli následně vybrat vhodný postup řešení problému a přistoupit k návrhu.

Teoretická část se věnuje jednak popisu tvorby webů a webových aplikací, dále také teorii datového a funkčního modelování a v neposlední řadě také popisuje nástroje pro údržbu a správu kódu ve vývojovém prostředí i mimo něj.

Hlavní cíl, a to návrh modelu nabídkové aplikace pro lepší a rychlejší nabídkový proces byl splněn. Díky analýze bylo možné vypracovat jednotlivé diagramy a návrh databázového modelu. Aplikace bude dále vyvíjena a následně implementována v systému.

SEZNAM POUŽITÝCH ZDROJŮ

[1] The world Wide Web Consortium [online]. W3C: ©2016 [cit.1.3.2020] Dostupné z: https://www.w3.org/standards/webdesign/htmlcss#moreinfo

[2] MUSCIANO, Chuck; KENNEDY, Bill. HTML & XHTML: The Definitive Guide: The Definitive Guide. " O'Reilly Media, Inc.", 2002.

[3] CASTRO, Elizabeth; HYSLOP, Bruce. HTML5 a CSS3. Albatros Media as, 2015.

[4] GRANT, Keith J. CSS in Depth. Manning Publications Co., 2018.

[6] FLANAGAN, D. JavaScript: The Definitive Guide, 6th Edition. 2011.

[7] NIXON, Robin. Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5. " O'Reilly Media, Inc.", 2014.

[8] NIELD, Thomas. Getting Started with SQL: A Hands-on Approach for Beginners. " O'Reilly Media, Inc.", 2016.

[9] OPPEL, Andrew J.; GROFF, James R.; WEINBERG, Paul N. SQL: The Complete Reference. McGraw-Hill, 2010.

[10] KOCH, Miloš a Vysoké učení technické v Brně. Datové a funkční modelování. Brno: Akademické nakladatelství CERM, 2006. s. 32. ISBN 80-214-3252-7. Dostupné také z: https://kramerius5.nkp.cz/uuid/uuid:56aeb190-7cdf-11e7-94b3-005056825209

[11] Unified Modeling Language, v2.5.1 [online]. Object Management Group, 2017

[cit. 20.4. 2020]. Dostupné z: https://www.omg.org/spec/UML/2.5.1/PDF

[12] Visual Studio [online]. Microsoft 2020 [Cit. 30.4. 2020]. Dostupné z: https://visualstudio.microsoft.com/cs/

[13] CHACON, Scott. Pro Git. Praha: CZ.NIC, c2009. CZ.NIC. ISBN 978-80-904248-1-4. Dostupné také z: http://www.digitalniknihovna.cz/mzk/uuid/uuid:08825630-78a3-11e5-99af-005056827e52

SEZNAWM POUŽITÝCH OBRÁZKŮ

SEZNAM POUŽITÝCH TABULEK

SEZNAM POUŽITÝCH GRAFŮ

SEZNAM PŘÍLOH

Kód pro generování databázového modelu:

Scheme.sql