UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Implementace informačního systému pro hotely

Tomáš Tvrdý

Diplomová práce
2018

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2017/2018**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Tomáš Tvrdý**
Osobní číslo:	**I16243**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Implementace informačního systému pro hotely**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

Obsahem práce je vytvoření vnitropodnikového webového servisního systému v Java EE pomocí frameworku Spring. Informační systém je určen pro zaměstnance malých a středních hotelů, kteří jeho prostřednictvím budou moci provádět rezervace, zobrazovat statistiky. Systém bude mít implementované RESP či SOAP rozhraní pro napojení na restaurační systém. Navázán bude také welness subsystém s jeho rezervačními rutinami. V teoretické části by se měl student zabývat problematikou Informačního systému (IS), členění IS a rozboru jeho jednotlivých typů. Dále je příhodné představit použité technologie a důsledně zdůvodnit jejich volbu. Aplikace bude obsahovat rozhraní pro napojení externích rezervačních systémů. Systém bude podporovat přímé generování faktur a bude mít rozhraní pro připojení EET.

Rozsah grafických prací: **10**

Rozsah pracovní zprávy: **60**

Forma zpracování diplomové práce: **tištěná**

Seznam odborné literatury:

FREEMAN, Adam. Pro ASP.Net MVC 5. New York: Apress, 2013. ISBN 978-1-4302- 6529-0.
LACKO, Luboslav. ASP.NET a ADO.NET 2.0 : Hotová řešení. Brno : Computer Press, 2006. 385 s. ISBN 80-251-1028-1
MACDONALD, Matthew, Adam FREEMAN a Mario SZPUSZTA. ASP.NET 4 a C# 2010: tvorba dynamických stránek profesionálně. Vyd. 1. Překlad Jan Pokorný. Brno: Zoner Press, 2011, 880 s. Encyklopedie Zoner Press. ISBN 978-80-7413-131-8.

Vedoucí diplomové práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání diplomové práce: **30. října 2017**

Termín odevzdání diplomové práce: **18. května 2018**

Ing. Zdeněk Němec, Ph.D.
děkan

L.S.

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2017

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 15. 08. 2018

Tomáš Tvrdý

Poděkování

Zde bych chtěl poděkovat vedoucímu práce panu Ing. Josefu Brožkovi za to, že mi umožnil zpracovat takto zajímavé téma s praktickým využitím a za jeho cenné rady při konzultacích, které mi pomohly při tvorbě této práce. Děkuji také své rodině a přátelům za poskytnutí výborných podmínek pro studium.

Anotace

Teoretická část práce se zabývá úvodem do teorie informačních systémů se zaměřením na hotelové informační systémy, architekturou webových aplikací a popisem technologií použité v praktické části. Praktická část práce se zaobírá návrhem a implementací interního hotelového systému.

Klíčová slova

Informační Systém, Hotel, Java, Jave EE, Spring, Hibernate, JPA, Webové služby, SPA, JavaScript, Angular

Title

Implementation information system for hotel

Annotation

The theoretical part deals with the introduction to the theory of information systems focusing on hotel information systems, web application architecture and description of technologies used in the practical part. The practical part deals with the design and implementation of the internal hotel system.

Keywords

Information System, Hotel, Java, Jave EE, Spring, Hibernate, JPA, web services, SPA, JavaScript, Angular

Obsah

Seznam zkratek

CBIS	Computer base information Systems
CLI	Common Language Infrastructure
CLR	Common language runtime
CRM	Customer Relationship Management
CSS	Cascading Style Sheets
ČR	Česká republika
DI	Dependency Injection
DB	Database
DCL	Data Control Language
DDL	Data Definition Language
DML	Data Manipulation Language
DWH	Data Warehouse
ERP	Enterprise Resource Planning
EU	Evropská unie
GDS	Global distribution system
HRM	Human Resource Management
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IoC	Inversion of Control
IS	Information Systém
JS	JavaScript
JSON	JavaScript Object Notation
LAN	Local area network
MIS	Management Information Systems
MES	Manufacturing Execution Systems
MPA	Multiple page application
MRP	Manufacturing Resources Planning
ODA	Online Travel Agency
PC	Personal computer
PMS	Property Management System
SPA	Single page application
SW	Software
TC	Transaction Control Language
TS	TypeScript
UCB	University of California, Berkeley
URL	Uniform Resource Locator
WWW	World Wide Web
XML	Extensible Markup Language

Seznam obrázků

Seznam zdrojových kódů

Seznam tabulek

1 Úvod

Cílem této práce je implementace informačního hotelového systému pomocí Java EE s využitím frameworku Spring. V teoretické části je nutné vytvořit úvod do informačních systémů včetně jejich členění. Bude vhodné také představit problematiku hotelových systémů. Dále rozebrat webové technologie, architekturu a popsat důvod jejich výběru. V praktické části bude kromě samotné implementace nutné vytvořit kvalitní návrh aplikace.

Od čtenáře této kvalifikační práce se předpokládá základní znalost v oblasti objektově orientovaného programování v libovolném vyšším programovacím jazyce, základní znalost tvorby webových aplikací a základy standardu SQL.

Před samotným návrhem aplikace bude vhodné zopakovat úvod do informačních systémů. Hlavní přidaná hodnota této části bude vysvětlení základních pojmů práce. Dále text bude zaměřen na popsání historie vývoje informačních systémů a jejich typů, kde hlavním cílem bude popsat typ, který bude implementován v rámci diplomové práce. Tato kapitola jako celek má za cíl ujasnit výhody společností používajících informační systém před konkurencí.

Pro jednodušší tvorbu funkčních požadavků pro aplikaci systému se bude následující část zabývat problematikou hotelnictví. Cílem kapitoly nebude popsat přesný chod všech agend, ale popsat základní teoretické poznatky o hotelnictví s důrazem na záležitosti, které bude potřeba zohlednit při tvorbě požadavků, jako jsou například povinnosti vyplívající ze zákona a podobně.

Jelikož součástí práce je také zdůvodnění výběru technologií, bude po části hotelnictví následovat kapitola, která se bude zabývat architekturou webových aplikací. Aby bylo možné rozebrat tuto kapitolu, je nutné nejdříve rozebrat technologie, na kterých je web stavěn. Hlavním cílem tohoto oddílu bude porovnání architektur s následným výběrem architektury pro hotelový informační systém a také zdůvodnění tohoto výběru.

Poté bude následovat tvorba funkčních požadavků, nefunkčních požadavků a dalších potřebných diagramů, podle kterých bude následně implementován systém. V této fázi vznikne také databázový a analytický model.

Dalším bodem práce je představení technologií, které byly použity. Tento bod je z důvodu rozsáhlosti velice problematický. Hlavním problémem je, že v systému bude použita řada různorodých technologií, kde je možné o každé z nich napsat několik stovek stran textu. Proto tyto programovací jazyky, frameworky a další budou popsány koncepčně a cílem zde nebude vytvoření programátorské příručky, ale pochopení konceptu těchto technologií s důrazem na prvky, které budou využity při vývoji. Ke konci každé z těchto kapitol budou popsány stěžejní části z implementace aplikace.

Výše popsanými body budou splněny veškeré požadavky na kvalifikační práci, včetně návrhu praktické části. V závěrečné kapitole bude čtenář seznámen s aplikací.

1.1 Informační systém

V úvodu kapitoly jsou popsány základní pojmy v oblasti informačních technologiích. Poté je definováno, co je informační systém a jak se vyvíjel v posledních desetiletích. V závěru kapitoly se čtenář dozví jeho klasifikaci a jeho základní typy.

1.2 Data

„Data chápeme jako rozpoznané signály (údaje), které vypovídají o situacích a stavech sledovaných a řízených objektů. Jsou podkladem pro další zpracování, během kterého se data mění na informace.“ [1]

1.3 Informace

Informace jsou data, kterým je porozuměno pomocí zkušeností, znalostí apod. neboli data už mají určitý smysl. Dá se také říci, že po přijetí informace se jejímu příjemci zmenší nebo dokonce úplně odstranění neurčitost v oblasti, které se informace týká. [1]

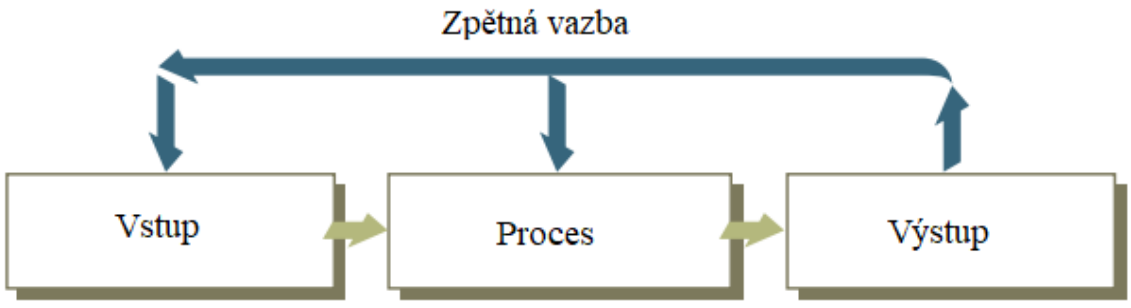
1.4 Systém

Systém je množina prvků, které jsou navzájem propojeny vazbami do určitého složení. Těmito prvky mohou být například lidé, datové nebo informační zdroje. Vazba určuje relaci mezi nimi. Propojení skupin může být jak jednosměrné, tak obousměrné. Tento objekt je spjat se svým okolím. Systém je možné zkoumat jako celek nebo je možné rozložit na podsystémy, které se následně stanou samy o sobě systémem. [1]

1.5 Informační systém

Celá podkapitola čerpá z [2].

Informační systém je množina vzájemně propojených prvků, které sbírají, manipulují, ukládají a šíří data a informace a také poskytují zpětnou vazbu. Mechanismus zpětné vazby pomáhá firmám k dosažení jejich cílů, jako je rozpoznání kritických bodů ve výrobě, vliv nastavení parametrů ve firmě na zisk, historická data pro reporting a podobně.



Obrázek 1 – Struktura IS [2]

Vstup

Vstup je aktivita, při které dochází ke shromažďování a získávání prvotních dat. Může se jednat například o počet hodin odpracovaných za měsíc, známky studentů, počet objednávek a podobně.

Proces

Proces je funkce, která zpracovává shromážděná data na požadovaný výstupy. Obvykle může zahrnovat výpočty, porovnání dat a jejich ukládání. Proces je možné provést dvěma způsoby, a to buď ručně, nebo za asistence počítače. Příklad procesu v hotelovém systému může být například výpočet celkové ceny ubytování při check-out, kdy jsou klientovi sečteny veškeré jeho placené služby (vstupy) v navštěvovaném zařízení. Při této funkci se generuje výsledná faktura do systému a je následně uložena a vytisknuta.

Výstup

Výstup je výsledkem procesu. Obvykle to mohou být různé reporty, dokumenty a další podobné texty. Bývají to např. platy zaměstnanců, hodnocení produkce, studentů a podobně.

Zpětná vazba

Zpětná vazba je informace ze systému, která je dále použita jako změna vstup do procesu nebo zpracování procesu. Příklad může být detekce a následná oprava chyby. Konkrétně, pokud se vytváří faktura a částky neodpovídají, je třeba zadat vstup znova. Také mohou být výstupy vstupem pro další proces, který bude určovat predikci dalšího vývoje a podobně.

Shrnutí

Celý podnadpis čerpá z [1], [3] a [4].

Existují i jiné definice, ale všeobecně se dá konstatovat, že IS má za úkol zpracovávat informace, aby určitým způsobem zefektivnil, zjednodušil a zrychlil procesy ve firmě.

Význam informačních systémů je v dnešní době obrovský a nadále roste. Střední a velké podniky jsou zcela závislé na užívání podobných aplikací, jelikož by bez jejich organizačních funkcí či databází nebyly schopny udržet pořádek v informacích tak velkého rozsahu.

1.6 Historie IS

Celá podkapitola čerpá z [5].

Pro lepší představu je zde popsán vývoj informačních technologií s důrazem na nejdůležitější milníky z hlediska informačních systémů.

Sálové počítače

Doba od konce padesátých let do šedesátých let minulého století byla obdobím nadvlády sálových počítačů. V době byly první obchodní počítače vnímány jako nástroje pro zefektivnění výpočtů. Primárním účelem těchto systémů bylo zpracovávat veliké objemy dat. Jejich problémem byla jejich velikost a cena. První počítače byly velké, zabíraly i několik místností a mohly si je dovolit pouze velké společnosti, univerzity a vláda. Dalším problémem byla jejich náročnost na obsluhu, kdy každý uživatel počítače musel být odborník. Na konci šedesátých let byly představeny MRP systémy. Tento typ SW umožňoval zefektivnit výrobní procesy.

Obrázek 2 – Sálový počítač [6]

Počítačová revoluce

V roce 1975 byl představen první mikropočítač Altair 8800. Ihned se stal velice populárním a desítky společností začaly tento počítač prodávat. Z důvodu, že byl dostupný a použitelný i pro „obyčejné“ lidi, stal se hitem a kupovalo si ho spoustu domácností. V této době vznikla například společnost Apple. Vývoj na sebe nenechal dlouho čekat a v roce 1981 vydala společnost IBM svou verzi osobního počítače a pojmenovala ho „PC“. Podniky pro své podnikové IS začaly přecházet ze sálových počítačů na IBM PC. Vznikaly také první operační systémy a z tohoto důvodu mohli začít pracovat s PC i „obyčejní lidé“. Typickou operací v této době představovalo zpracování textu, tabulek a databází.

Obrázek 3 – Altair 8800 [7]

Klient-server

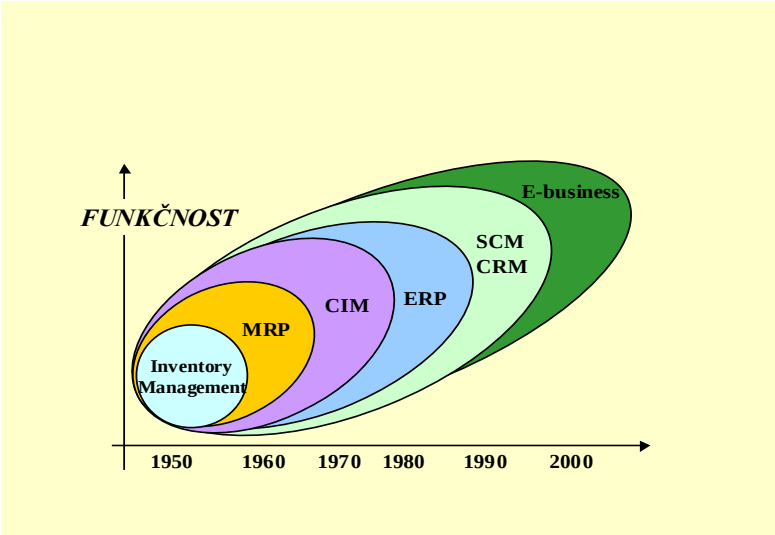
Uprostřed osmdesátých let dvacátého století se objevil problém, že všechny počítače byly „osamocené“ a podniky chtěly své počítače propojit, aby bylo možné data mezi počítači sdílet. Tato architektura se nazývá „klient-server“. Uživatelé se mohou přihlásit do LAN pro připojení k serveru, který poskytuje sdílené zdroje. Tato koncepce také umožňovala komunikaci mezi uživateli. V této době vznikly první systémy ERP.

WWW a E-Commerce

Na konci šedesátých byl vynalezen internet, který v počátku používala pouze vláda a univerzity. Firmám přestávalo stačit spojení pouze po LAN, bylo jim proto umožněno připojit se na internet.

V roce 1989 Tim Berners-Lee vyvinul jednoduchou metodu pro sdílení informací přes síť WWW. Tento vynález se stal klíčovým pro růst internetu jako způsobu sdílení informací. V průběhu let se celý svět propojil kabely. Vznikaly první počítačové viry. Jako důsledek tohoto propojení se rozvinul obor počítačová a internetová bezpečnost.

Technologie se nadále rychle vyvíjely. Webové stránky se staly dynamickými. Už nestačilo pouze prezentovat firmu, ale zákazníci s ní chtěli také komunikovat. Vznikaly sociální sítě, blogování a další alternativy. Mnoho podnikatelů doplňovalo své kamenné pobočky e-shopem.

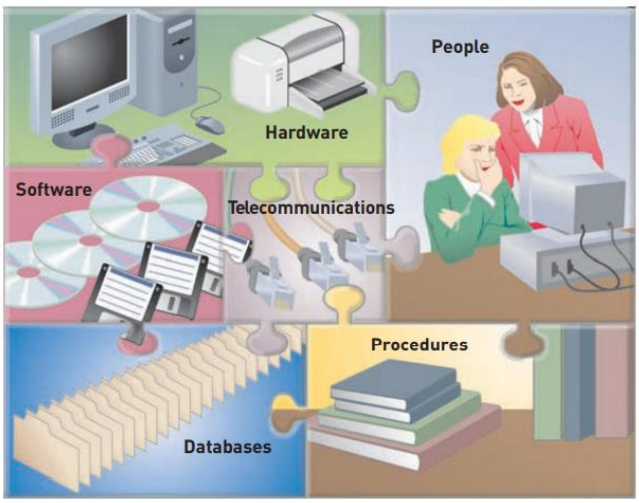


Obrázek 4 – Historie IS [8]

1.7 Počítačové IS

Celá podkapitola čerpá z [2].

Počítačové informační systémy (CBIS) je množina hardwaru, softwaru, databází, telekomunikací, lidí a postupů, která je nastavena pro zpracování dat do informací.



Obrázek 5 – Komponenty CBIS [2]

Hardware

Hardware je zařízení, které provádí zadávání vstupů, jejich zpracování a zobrazení výstupů. Mezi vstupní zařízení patří klávesnice, myš, web kamera a další. Jádro počítače tvoří operační paměť, procesor a trvalé úložiště dat. Jako výstupní zařízení se nejčastěji používá monitor nebo tiskárna.

Software

Software je každý počítačový program, který zpracovává HW. Rozlišují se dva typy. První typ je systémový SW, který má na starosti základní chod počítače a patří sem všechny operační systémy. Druhým typem je aplikační SW. Jedná se o programy, které slouží pro konkrétní úlohy a běží na systémovém SW. Patří sem např. Google Chrome, Malování a další.

Databáze

Jedná se o organizovaný soubor dat, který obvykle bývá uložen ve formě souborů, se kterými uživatel pracuje přes specializovaný SW. V databázi se typicky ukládají data o zaměstnancích, zákaznících, prodeji atd.

Telekomunikace, síť a internet

Telekomunikace

Telekomunikace umožňuje přenos elektrických signálu pro komunikaci. Slouží pro přenos signálu pro mobilní telefony, televizory atd. Pro přenos se obvykle používá metalický kabel, optický kabel nebo bezdrátový přenos.

Sítě

Sítě propojují počítače po celé zemi. Používají techniky telekomunikace, které umožňují elektronickou komunikaci.

Internet

Internet je největší celosvětový systém propojených počítačů.

Z jiného úhlu pohledu internet znamená síť počítačů, nazývaných též serverů, které jsou navzájem propojeny datovými kabely s vysokou rychlostí. Těmito kabely jsou přesouvány data pomocí nul a jedniček. Data se pohybují velmi rychle a využívají různé cesty podle momentálního stavu systému. Počítače si musí také „rozumět“. Pro tento účel se používají protokoly typu TPC/IP. Internet umožňuje nespočet služeb, mezi nejznámější patří WWW, e-mail, FTP a další. [9]

WWW

WWW je systém hypertextových dokumentů, které na sebe navzájem odkazují, a pro jejich prohlížení je třeba webový prohlížeč. Stránky obsahují text, multimediální data a odkazy, které umožňují vzájemnou provázanost stránek. [9]

WWW je technologie založená na HTTP a URL. HTTP byl vytvořen pro výměnu takzvaných hypertextových dokumentů mezi dvěma zařízeními, obvykle klient-server. URL určuje cestu, na které se požadovaný dokument nachází. [9]

Lidé

Personál informačních systémů zahrnuje osoby, kteří spravují, vyvíjí, užívají a udržují informační systém. Uživatelé IS jsou lidé, kteří využívají IS pro dosažení konkrétního cíle.

Postupy

Postupy zahrnují pravidla, metody, strategie pro fungování IS. Jako příklad může být systém přístupových práv.

1.8 Klasifikace IS

Celá podkapitola čerpá z [10].

Existuje několik druhů klasifikace IS, které se běžně používají. Problém je, že se jedná o obor, který se stále rychle vyvíjí. Při kategorizaci dochází často k tomu, že kategorie nejsou navzájem disjunktní, tudíž jeden typ IS může patřit do více kategorií. Z tohoto důvodu bude nadále zavedena velice abstraktní klasifikace.

Osobní informační systém

Osobní informační systém slouží pro správu informací pro účely jedné osoby. Může to být například evidence sbírky dané osob, rodokmen atd.

Veřejný informační systém

Na rozdíl od osobních informačních systémů může k těmto přistupovat i veřejnost. Příkladem takového informačního systému je databáze univerzitní knihovny, kde si veřejnost může vyhledat, jaké knihy jsou momentálně v knihovně dostupné.

Podnikový informační systém

Podnikový systém slouží pro podporu a zefektivnění firemních procesů. Typicky se sem řadí systémy pro účetnictví, docházku atd. Celý zbytek této práce se zabývá tímto typem IS.

1.9 Klasifikace podnikového IS

Celá podkapitola čerpá z [11].

„V každém podniku existuje několik organizačních úrovní, které požadují specifický způsob zpracování informací či specifický druh informací. Přitom se nejčastěji rozlišují strategická, řídicí, znalostní a provozní úroveň. Žádná z těchto úrovní sama o sobě nemůže poskytnout všechny informace, které managment potřebuje pro řízení. Podobně ale žádná z těchto úrovní nepředstavuje samostatnou ucelenou entitu, která by odrážela praktickou potřebu nasazení samostatného informačního systému (softwarová aplikace). Proto také často používaná klasifikace, která rozlišuje provozní, znalostní, řídící a strategické informační systémy, odráží výhradně teoretický náhled na fungování podniku. Jejím úkolem je charakterizovat hodnotu

automatizovaného zpracování informací pro pracovníky na jednotlivých organizačních úrovních, takže takovýto pohled na věc svůj smysl bezesporu má.“ [11]

Provozní úroveň

Provozní úroveň zpracovává informace na nejnižších úrovních. Jedná se převážně o každodenní činnosti, např. počet položek na skladě, zda odešly veškeré palety, dále jaký výrobek se má začít zpracovávat. Tuto úroveň IS používají obvykle střední management, technickohospodářští a provozní pracovníci.

Znalostní úroveň

„Znalostní úroveň zahrnuje nejen klientské aplikace podnikového informačního systému (ERP, CRM atd.), ale také prostředky osobní informatiky, jako jsou kancelářské aplikace, software určený pro týmovou práci (groupware atd). Tyto aplikace podporují růst znalostí báze organizace a řídí především tok dokumentů.“ [11]

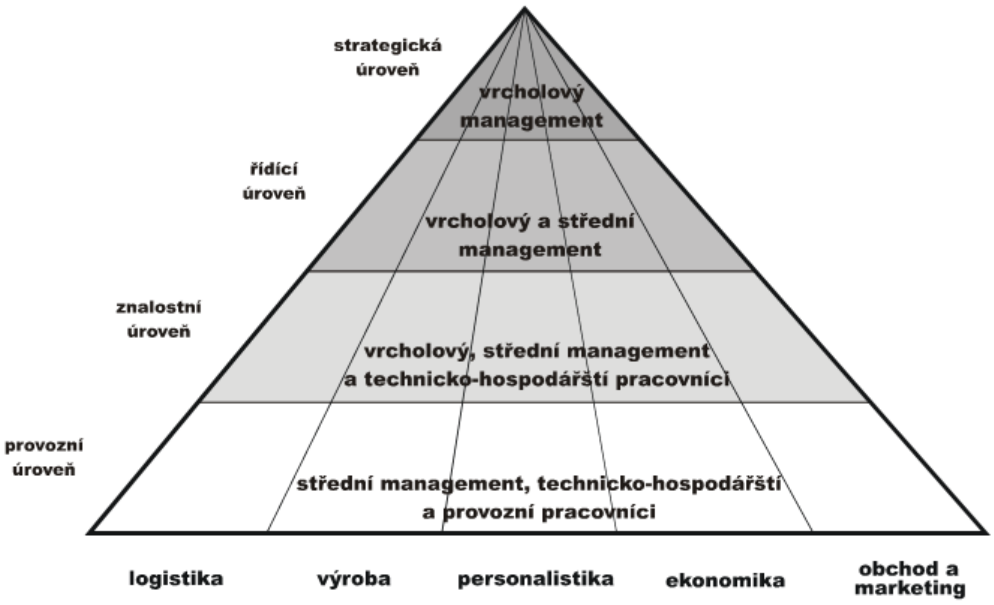
Typickou úlohou je zpětná reakce klientů, aktuální hospodaření podniku a podobně. Uživatelé na této úrovni jsou obvykle manažeři a technickohospodářští pracovníci na všech úrovních.

Řídící úroveň

Řídící úroveň má na starosti podporu rozhodování. Tuto úlohu plní pomocí reportingů. Uživatele tohoto systému jsou obvykle z vrcholového a středního managementu. Tento SW například odpovídá, co se stane, pokud něco firma požaduje atd.

Strategická úroveň

Strategická úroveň je určená pro vrcholový management a její hlavní účel je identifikování dlouhodobých trendů a popřípadě doporučit, jak na ně má společnost reagovat a jestli je toho vůbec schopná. Strategická úroveň odpovídá třeba na otázku, kolik bude chtít klient vyrobit výrobků za x let.



Obrázek 6 – Informační pyramida podle organizačních úrovní podniku [11]

1.10 Jednotlivé typy IS

Celá podkapitola čerpá z [11].

Jelikož existuje veliké množství typů IS, proto jsou popsány pouze ty nejběžnější. Název bude uveden v češtině a v závorce bude uvedena anglická zkratka, která je popsána v oddílu zkratek. Tento typ zápisu je v práci uveden z důvodu, že se tyto zkratky staly standardem i v ČR. Další typy IS lze dohledat ve zmíněné literatuře.

Plánování a řízení podnikových zdrojů (ERP)

Historicky se pro každou oblast procesů používal jeden IS. To znamenalo, že pro oblasti jako je řízení lidských zdrojů, účetnictví, prodej, marketing, management, logistika a dalších byla vždy pouze jedna aplikace, jelikož jednotlivé oblasti jsou na sobě částečně závislé a stejně tak používají i část stejných dat, tak vznikaly problémy v redundanci dat atd.

ERP odstraňuje výše zmíněný nedostatek a integruje veškeré standardní oblasti do jedné aplikace. Mezi hlavní výhody tohoto přístupu patří sjednocený pohled na veškerá data ve společnosti.

Nejpoužívanější ERP systém v dnešní době je od německé společnosti SAP, která nabízí stejnojmenný systém. Tuto aplikaci k roku 2009 používalo 43 000 firem.

Řízení vztahu se zákazníky (CRM)

V současné době se obchod a marketing orientují na konkrétní požadavky zákazníka. Pro tento účel slouží CRM.

CRM pomáhá uspokojit potřeby zákazníka pochopením jeho potřeb, segmentací zákazníků do skupin, přizpůsobování nabídky podle potřeby zákazníka, cílení reklamy a podobně.

Řízení lidských zdrojů (HRM)

HRM má na starosti celou agendu lidských zdrojů. Stará se hlavně o docházku, mzdy, jak si udržet zaměstnance atd.

Výrobní informační systém (MES)

„Při automatizaci výroby je nutné zabezpečit nejen vazbu na logistické procesy, ale také návaznost na samotný výrobní proces. K získávání provozních dat v reálném čase se zpravidla využívá tzv. výrobní informační systém. V hierarchii podnikových informačních systémů tvoří vrstvu mezi technologickou úrovní výroby a ERP systémy. MES se tedy zabývá detailním sběrem dat a jejich zpracování pro účely vyhodnocení výroby z mnoha různých úhlu pohledu a operativního řízení.“ [11]

2 Informační systémy v hotelnictví

Celá kapitola čerpá z [13], [14] a [15].

Jelikož cílem práce je implementovat hotelový informační systém, byla problematika hotelových IS v minulé kapitole vynechána a bude podrobněji probrána v této kapitole. Před samotným popisem problematiky budou popsány základní pojmy. Ne každý hotel má veškeré zde popsané části, jelikož velmi záleží na velikosti hotelu.

2.1 Globální distribuční systém

V dobách kdy se začínala rozvíjet letecká doprava, byl velký problém se zprostředkováváním aktuálních informací mezi leteckými společnostmi a agenturami, popřípadě cestovními kancelářemi dále pouze agenda. V případě, kdy bylo potřeba vytvořit novou rezervaci, musela se agentura nejprve spojit se zaměstnancem letecké společnosti. Problém nastal v případě, kdy chtěla jiná společnost provést stejnou rezervaci, protože neměla aktuální informace. Je třeba si také uvědomit náklady letecké společnosti za plat těchto zaměstnanců. Z tohoto důvodu v počátcích šedesátých let GDS, přes který společnosti komunikují a mají aktuální data. Postupem času se začínali připojovat další přilehlé služby jako jsou ubytovací, restaurační služby a tak dále. Mezi nejpopulárnější patří Sabre a Amadeus. Je třeba si také uvědomit, že přístup jakékoliv globální distribuce klientů vede také k více zákazníkům, a to v důsledku nadále vede k vyšším tržbám.

2.2 Rezervační portál

Rezervační portál je portál, který shromažďuje možnosti ubytování na určitém území. Může se jednat jak o systém, který je zaměřen pouze na malé území, tak i o systém zaměřený na celý svět. Provozovatelé ubytovacích zařízení mají v tomto případě smlouvu s ODA a nejčastějším modelem platby je platba provizí za každou rezervaci, kterou klienti provedou přes rezervační portál. Mezi nejpopulárnější patří například booking.com a expedia.com. Z českých rezervačních portálů stojí za zmínku například hotel.cz.

2.3 Meta vyhledávače

Meta vyhledávače nemají na rozdíl od OTA a GDS za cíl provést rezervaci, ale pouze porovnávají nabídky jednotlivých OTA, GDS a dalších forem distribuce. Potencionální klient si následně si může porovnat cenu, nabídku služeb a tak dále. Konkrétní rezervaci si však musí zařídit sám přes vybraný portál. Mezi nejznámější meta vyhledávače patří google.com a trivago.com.

2.4 Channel Manager

Channel manager slouží pro napojení hotelu mezi více OTA, GDS a další. Přes tento systém se v případě rezervace také aktualizuje nabídka pokojů v ostatních systémech. V dnešní době je to nepostradatelný nástroj pro podnikání v oblasti hotelnictví.

2.5 Recepční systém

Recepční systém je komplexní hotelový IS, který se stará převážně o rezervace, fakturace, správu hostů, základní statistiky atd. Tento systém může být modulem, který spolupracuje například s CRM nebo jinými typy IS. Výčet jeho funkcí, popřípadě modularizace záleží především na velikosti hotelu. [13]

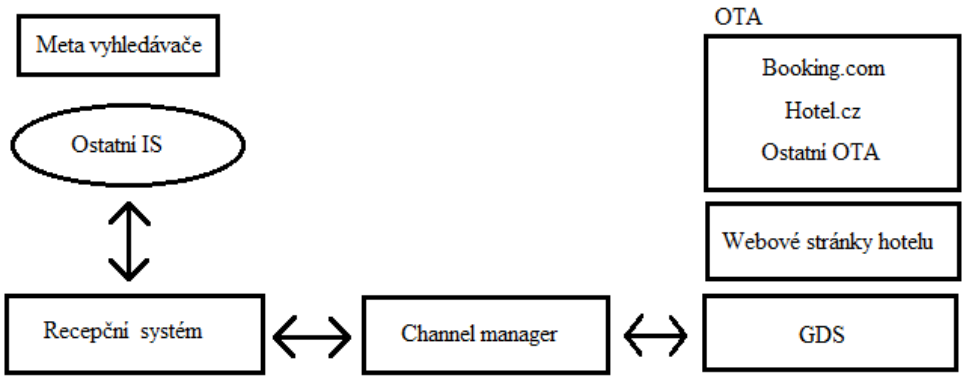
2.6 Struktura informačních systémů v moderním hotelu

Recepční systém slouží jako centrální správní jednotka pro nastavení ceníků, obsazenosti, ubytovacích jednotek hotelu. Také se zde prováděna většina analýz pro potřeby marketingu. Veškeré změny, které jsou provedeny v PMS jsou prostřednictvím channel manageru distribuovány do partnerských zařízení, kde tak mají přehled o aktuálních datech hotelu.

Channel manager v naprosté většině případů spolupracuje s webovými stránkami hotelu, kde si může klient zprostředkovat rezervaci bez další provize pro třetí stranu. Dalšími systémy, se kterými obvykle hotel spolupracuje jsou GDS a OTA.

Zvláštní roli v této struktuře mají meta vyhledávače, které bez účasti hoteliérů přivádějí také přivádějí nespočet klientů, a to obvykle hostů, kteří chtějí jet na někam na dovolenou a shání ubytování na určitém místě. Potencionální host obvykle hledá co nejlevnější hotel s co nejvyšší kvalitou služeb. Na webových stránkách Trivago či jiných zadá oblast, ve které by se měl hotel nacházet a poté si vybere z dostupných hotelových zařízení to nejvhodnější. Nakonec si vybere nabídku daného hotelu za nejpříznivější cenu, ať už se jedná o nabídku přímo ze stránek hotelu nebo od jeho partnerů.

Recepční systém neřeší například platy zaměstnanců, účetnictví a podobně, proto je obvykle ještě napojen na systémy typu CRM atd.



Obrázek 7 – Struktura IS v hotelu

3 Úvod do problematiky hotelnictví

Celá tato kapitola čerpá z [13].

Tato kapitola je věnována úvodu do problematiky hotelnictví s důrazem na popis nejdůležitějších údajů pro implementaci hotelového IS. Úvod kapitoly je věnován historii a důvodům vzniku dané služby. Dále jsou popsány základní pojmy a klasifikace hotelů. Následně přicházejí nejdůležitější podkapitoly pro implementaci. Zde čtenář získá poznatky o organizaci hotelu a fungování základních složek, povinnostech pro provozovatele ubytovacího zařízení, kterým podléhá podle zákona, a to především pro pobyt cizinců.

3.1 Historie

V dobách rozvoje směnného obchodu vznikl požadavek na ubytování obyvatel a cizinců mimo jejich domov. Lidé, kteří obchodovali se zbožím, jako jsou zlato, stříbro, potraviny, koření, zbraně, dobytek a podobně, byli nuceni pro svoji obživu cestovat do okolních měst a vesnic, někdy dokonce i do vzdálených zemí.

V této době samozřejmě neexistovala žádná pokročilá navigační zařízení, nebyla žádná pokročilá technika pro překonávání dlouhých vzdáleností a pro transport se používala výhradně zvířata. Obvyklým problémem bylo i to, že cesta nebyla detailně zmapovaná a obchodník mohl jednoduše zabloudit. Stejně tak byla poměrně velká pravděpodobnost napadení lupiči, splašení koní divokou zvěří atd. Z tohoto důvodu obchodníci, co podnikali několikadenní cesty, začali vyhledávat klidné a pohodlné ubytování ideálně i se stravou, kde by si mohli zároveň i odpočinout.

„Je to dlouhá historie, která dělí první nuzné ubytovací přístěnky z druhého tisíciletí před naším letopočtem od dnešních hotelových řetězců, poskytujících všemožný domácí i pracovní luxus včetně připojení na internet, kosmetických služeb, wellness a fitness provozů“ [13]

Klášterní azyl
Po rozpadu římské říše hrálo v ubytovacích zařízeních hlavní roli náboženství. Nejvýznamnější byl zákon o bližním svému. V dobách od 11. století bylo postaveno nespočet klášterů. Ty se staly středem vzdělání a pohostinství a bylo obvyklé, že obsahovaly ubytovny. Tyto budovy obsahovaly i odlehlou část hospitium, které sloužilo k ubytování cizinců, poutníku, ale i sirotků a nemocných lidí. Přestože byli zdraví lidé ubytováni odděleně, riziko nákazy bylo stále značné. Hospitium nemuselo být pouze u klášterů, ale stavila je i města, rytířské řády a podobně. Z počátku bylo ubytování zdarma, ale postupem času stalo čím dál více předmětem podnikání.

Zájezdní hostinec
Kolem začátku 14. století začaly vznikat hostince, kde se mohli lidé ubytovat. V tomto období bylo podnikání v pohostinství trochu chaotické, a to převážně z důvodu, že úroveň ubytování nebyla nijak klasifikována. Kromě klasických obecných zákazů, jako je například hraní hazardních her, nebyli činnosti v hotelu nějak omezovány.

Zákazník tak nemohl předem moc tušit o úrovní hotelu a je třeba brát potaz, že v této době nebyla dostupná tekoucí voda, elektřina atd. To mělo za následek ne zcela luxusní ubytování. Dalším problémem bylo získat licenci, protože hostinští byli shromážděni v cechu a do tohoto zřízení mohl vstoupit pouze určitý počet lidí. Toto právo se dědilo.

Vznik hotelu
Pro vznik hotelů, tak jak jsou známy dnes, byl nejdůležitější rozvoj železniční dopravy. S tímto objevem nastala nová situace v cestování, kdy se začalo přesouvat mnohem více lidí než dříve. Podnikatelé začali v tomto období stavět luxusní budovy pro ubytování, a to především u nádraží a v lázeňských městech. V těchto objektech se také vyvíjely další nové služby, jako byla například jídelna, která byla úplně oddělená od klasické hospody, kde hosté pili alkoholické nápoje. Standardem se začalo stávat, že hotely nabízely i jiné služby než pouze ubytování.

3.2 Hotel

„Hotel může být definován jako místo, kde se za úplatu poskytuje (zpravidla krátkodobé) ubytování všem kategoriím turistů. Vedle ubytovacích služeb jsou v závislosti na klasifikaci, respektive v závislosti na zařazení do jakostní třídy poskytovány další služby – stravovací, společensko-zábavní, relaxační, konferenční a obecně všechny služby, které jsou relevantní v této oblasti služeb. Obvyklá minimální kapacita je 10 pokojů, ale v závislosti na národní normě nebo historické zvyklosti může být vyžadován i větší počet pokojů.“ [13]

Rozdělení hotelů je možné mnoha způsoby. Zde jsou uvedeny pouze ty nejpoužívanější.

Podle umístění:

- přímořské,
- horské,
- městské,
- lázeňské,
- rekreační.

Toto dělení slouží pro představu, kde se hotel nachází.

Podle služeb:

- kongresové,
- konferenční či seminární,
- wellness,
- lázeňské,
- sportovní,
- relax,
- rodinné.

Toto dělení je důležité pro zákazníka, aby si udělal představu o hlavní funkci hotelu, do kterého se plánuje ubytovat.

Podle velikosti:

- malé,
- střední,
- velké,
- mega.

3.3 Kategorizace

Ministerstvo pro místní rozvoj ve vyhlášce č. 501/2006 Sb. o obecných požadavcích na využívání území zařazuje ubytovací zařízení do následujících kategorií.

Hotel

„Hotel, kterým se rozumí ubytovací zařízení s nejméně 10 pokoji pro hosty, vybavené pro poskytování přechodného ubytování a služeb s tím spojených.“ [16]

Motel

„Motel, kterým se rozumí ubytovací zařízení s nejméně 10 pokoji pro hosty, vybavené pro poskytování přechodného ubytování a služeb s tím spojených pro motoristy.“ [16]

Penzion

Penzion, kterým se rozumí ubytovací zařízení s nejméně 5 pokoji pro hosty, s omezeným rozsahem společenských a doplňkových služeb, avšak s ubytovacími službami srovnatelnými s hotelem; pro účely klasifikace je pension specifikován jako ubytovací zařízení s nejméně 5 a maximálně 20 pokoji pro hosty. [16]

Ostatní ubytování

„Ostatní ubytovací zařízení, kterými jsou zejména ubytovny, koleje, svobodárny, internáty, kempy a skupiny chat nebo bungalovů, vybavené pro poskytování přechodného ubytování.“ [16]

3.4 Klasifikace

Jak bylo uvedeno v historii hotelnictví, klasifikace hotelů slouží pro informování hostů, jaké služby mohou od zařízení dostat. V dřívějších dobách žádná vysoce standardizovaná klasifikace neexistovala. Nejklasičtější metodou se stala klasifikace podle počtu hvězdiček, ale existují i podle počtu klobouků, letadélek a další. Problémem v současnosti je nejednotná klasifikace hotelů v EU. Tento problém se snaží vyřešit systém Hotelstars Union. Tohoto systému jsou součástí české organizace, který mají na starosti podnikání v ubytovacích službách. Tento systém definoval konkrétní požadavky, které musí zařízení splňovat, aby dosáhla určité klasifikace.

Klasifikace se rozdělují na:

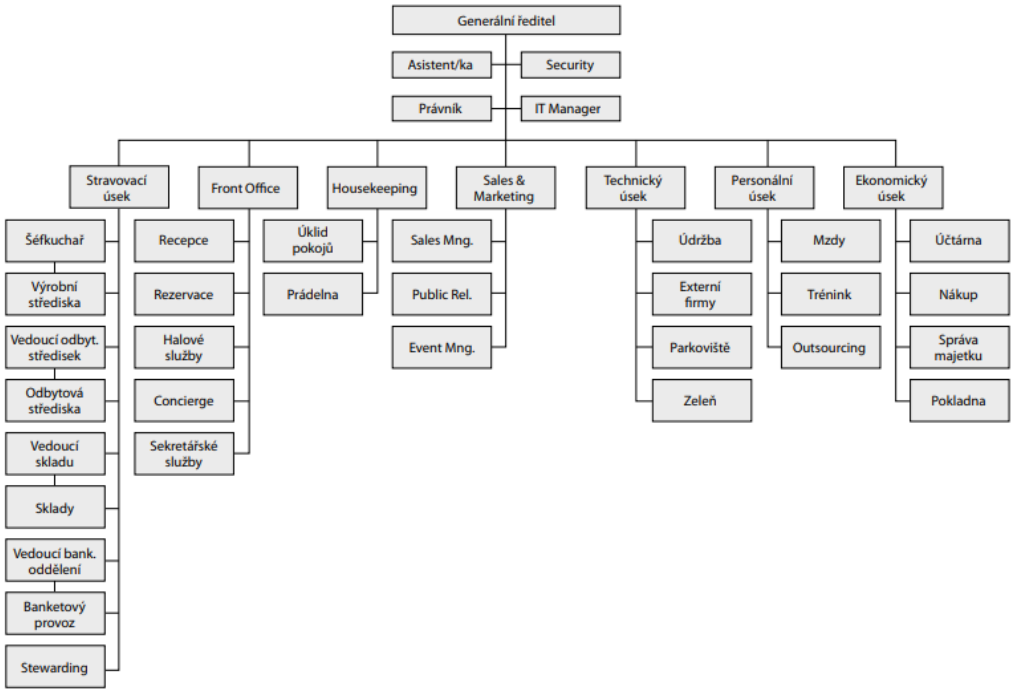
- * → tourist,

- ** → economy,
- *** → standart,
- **** → first class,
- ***** → luxury.

3.5 Organizace hotelu

Jako každá firma s více zaměstnanci, i v hotelnictví je potřebná určitá organizace jednotlivých oddělení. Složení této organizační hierarchie je velice problematické, protože velice záleží na konkrétním zařízení. Některé hotely mohou určité úseky úplně vynechat, popřípadě je nahradit outsourcingem. Největší vliv na tuto skutečnost má velikost, klasifikace a kategorie zařízení. Hlavním rozdílem mezi těmito úseky je, zda jsou ziskové nebo neziskové.

Klasický neziskový modul, který je ale nutným, je technický úsek. Veškeré úseky musí fungovat, protože mohou mít vliv na celkový dojem zákazníka a mohou zkazit dojem z celého hotelu. Níže uvedený výčet úseků není zdaleka konečný, existují i další.



Obrázek 8 – Organizační struktura středně velkého hotelu [13]

3.6 Základní úseky

Existuje velký počet úseků, proto zde budou uvedeny pouze základní a ty, které jsou zajímavé s ohledem na tvorbu požadavků.

Front office

Front office neboli recepce má na starosti hlavně rezervace, check-in, check-out, podávání informací klientům, vyřizování vzkazů a podobně. Forma recepce velice záleží na velikosti hotelu. V ubytovacích zařízeních, která jsou spíše malého charakteru, bývá na recepci pouze

jeden člověk, který se stará o vše. V takovém případě recepce nemusí být dostupná ani 24 hodin denně. Pokud se jedná o velký hotel, klienty obsluhuje několik lidí, kteří mají rozdělené role. V tomto případě je samozřejmostí nepřetržitá pracovní doba.

Recepce je jedna z nejdůležitějších složek hotelové organizace. Pro klienta zde dochází k prvnímu kontaktu se zaměstnanci hotelu, kteří reprezentují zbytek zaměstnanců a zároveň samotné zařízení, tudíž se jedná o rozhodující okamžik pro spokojenost hosta. V případě spokojenosti je větší pravděpodobnost, že odpustí další negativní zkušenosti. Z tohoto důvodu je velice důležitý výběr zaměstnanců na pozici recepčního. Zároveň je důležité, aby tito lidé měli požadované dovednosti a vlastnosti. Měli by být převážně milí, ochotní, komunikativní, reprezentativní atd. Také musí mít vysoké jazykové dovednosti, ovládat PC a schopnost koordinace.

Front office bývá umístěn v hotelové hale. Jak je uvedeno výše, zde dochází k prvotnímu dojmu, který si nadále nese klient do dalšího ubytování. Z tohoto důvodu musí být také hotelová hala dobře navržena.

Check-in neboli přijetí hosta nebo skupiny. Zde dochází k ústřednímu momentu, kdy je host přijímán do hotelu. Obsluha recepce musí být seznámena s aktuální nabídkou zařízení, aby nedocházelo k předání špatných informací. Samotný průběh přijetí hosta velice záleží na konkrétním zařízení. Obecný průběh probíhá v tomto pořadí. V prvé řadě dochází k registraci hosta a přiřazení pokoje, poté se vyřeší finanční záležitosti, kde bývá obvyklé placení předem, garance nebo platba zálohy. Následně se seznámí s nabídkami hotelu a pak dochází k ubytování hosta.

Recepce musí v průběhu ubytování hostů řešit jejich připomínky, vyřizovat vzkazy, poruchy zařízení, bezpečnost hostů a podobně. Zde je důležité, aby při těchto krocích nedocházelo k obtěžování hostů.

Check-out neboli odjezd hosta má na starosti převážně dorovnání finančních závazků, odevzdání přiřazeného majetku, zjištění zpětné reakce hosta.

Rezervace hosta může probíhat různými způsoby. Mezi základní a nejpoužívanější druhy rezervace patří přes webový formulář, email, telefon nebo osobní rezervace. V případě větší skupiny hostů je možné využít tzv. skupinové rezervace. V takovém případě je nutné mít informace o hostech předem, aby check-in trval rozumnou dobu i pro větší skupiny.

Housekeeping

Nejlépe vystihuje housekeeping [13].

„Housekeeping je další klíčový úsek ubytovacího úseku. Jeho význam spočívá především v tom, že na dobře uklizeném a připraveném pokoji nachází host onu kvalitu hmotné části ubytovací služby jako celkového produktu. Obsahem pojmu housekeeping však není jen úklid pokojů, ale jak z volného překladu vyplývá, jedná se o péči o dům, tedy o celý hotel, všechny hostovi přístupné prostory a samozřejmě i prostory v zázemí, kam host nevidí, ale užívají je

zaměstnanci. A ti mají také právo na kulturní pracovní prostředí, tedy dobře udržované a pravidelně a systematicky uklízené. K hlavním úkolům housekeepra patří též péče o prádelnu z hlediska provozního, má-li hotel vlastní prádelnu. V opačném případě se stará o sklad prádla a spolupracuje s dodavatelskou prádelnou." [13]

„Jedná se tedy o komplex všech hlavních činností vedoucího pracovníka od plánování přes komunikaci až po kontrolu a komunikaci. Z pohledu ekonomického je Housekeeping střediskem nákladovým, bez tvorby tržeb (s výjimkou praní hostova osobního prádla). Proto odpovědnost za výši těchto nákladů je vysoká, vezmeme-li v úvahu pouze objemy praného hotelového prádla. Ve výčtu povinností housekeepra nelze opomenout jeho význam ve sledování opotřebení vybavení pokoje, potřebu generálního úklidu pokoje nebo celého patra, malování, čištění koberců, mytí oken a plánování těchto akcí nejen v kontextu vytížení hotelu, ale i výše nákladů na tyto práce vynaložených. V rámci organizace práce na středisku musí housekeeper stanovit objektivní časový snímek dne, neboť počty pracovníků ve směně se promítají nejen ve kvalitě odvedené práce, ale též v objemu čerpaných mzdových prostředků. Tak jako v recepci je každodenním heslem vstřícné chování, tak u pracovníků housekeepingu platí, že všudypřítomný pocit čistoty dělá hotel hotelem. „ [13]

Stravovací úsek

Mezi základní potřeby lidí patří jídlo a pití. O tyto potřeby se stará stravovací úsek. Jak již bylo několikrát uvedeno, i tento úsek velice závisí na konkrétním zařízení. I přesto ho některé hotely zcela postrádají a ty největší mají naopak i několik restaurací, které jsou dostupné po celý den. Zde je veliký problém z hlediska ziskovosti tohoto modulu. Z důvodu, že do stravovacího zařízení chodí zpravidla pouze hosté hotelu, je častý problém s nedostatkem hostů z okolí. K tomuto faktu nepřispívá ani skutečnost, že přístup k restauračnímu zařízení bývá často jen přes recepci. Stravovací zařízení většinou nabízí jak klasický jídelní lístek, tak „denní menu". Pokud má hotel restaurační zařízení, zpravidla nabízí 5 druhů ubytování. Těmi jsou ubytování bez snídaně, ubytování se snídaní, polopenze, plná penze a all inclusive. Polopenze znamená, že v ceně ubytování je snídaně a buď večeře nebo oběd. V případě plné penze je v ceně snídaně, oběd a večeře. All inclusive znamená, že v ceně je zahnutá plná penze, vybrané jídlo a pití navíc za určitých podmínek. Častou praxí bývá, že od určité hodiny ráno do určité hodiny večer může host konzumovat cokoliv z nabídky a pít nealkalické nápoje.

3.7 Povinnosti hotelu ze zákona

Zde jsou ze zákona vypsány specifické povinnosti, které je potřeba řešit v implementaci systému. Zákony jsou zde popsány pouze obecně pro konkrétní znění a výpočty jsou dostupné ze zdrojů uvedených v popisu.

Místní poplatky

Místnímu úřadu, ke kterému se ubytovací zařízení vztahuje, může mít, ale také nemusí mít poplatky za ubytování podle *„zákon o místních poplatcích“*. Podle tohoto zákona je poskytovatel povinný vést knihu hostů a uchovávat ji alespoň 6 let po posledním zápisu. [16]

Televizní, rozhlasové a autorské poplatky

Dále je poskytovatel ubytování povinný platit rozhlasový a televizní poplatek za každý televizní přijímač, který vlastní. Také platí autorské poplatky organizacím OSA a Integram. [16]

Povinnosti ze zákona o pobytu cizinců

„Ze zákona o pobytu cizinců na území České republiky vyplývá mj. povinnost hlášení jejich místa pobytu na území České republiky.“ [13]

4 Webové technologie

Kapitola webové technologie se zabývá úvodem do problematiky webových aplikací. Seznámí čtenáře se základními technologiemi a přístupy nutnými pro pochopení dalších částí práce.

4.1 HTML

HTML je značkovací jazyk, který se používá pro vytváření webových stránek. Vznikl v devadesátých letech minulého století pro jednoduché tvoření dokumentů. HTML je nadále vyvíjeno a umožňuje stále další možnosti. Aktuální verze jazyka je HTML5. [19]

Pro zobrazení webových stránek slouží webové prohlížeče. V dnešní době mezi nejpoužívanější patří Google Chrome, Opera, Mozilla Firefox a Internet Explorer.

4.2 CSS

CSS slouží pro popis vzhledu HTML stránky. Hlavním důvodem vzniku této technologie je především oddělení definice vzhledu a obsahu dokumentu, což má za následek zvýšení přehlednosti kódu. Aktuální verze je CSS 3. Standardizaci má na starosti organizace W3C. [19]

4.3 JavaScript

JS je implementace standartu ECMAscript. JS se stala naprosto dominantní v porovnání s jinými jazyky, které tento standard implementuji, jako jsou například Jscript nebo ActionScript. JS byl vyvinut pro použití skriptování na straně prohlížeče, ale v dnešní době už je využíván i na straně klienta nebo v jiných technologií. Jedná se o platformě nezávislý interpretovaný jazyk. [20]

4.4 HTTP

Celá podkapitola čerpá z [21], [22] a [23].

Jedná se o protokol, který běží na aplikační vrstvě protokolu TPC/IP sloužící pro přenos objektů mezi webovým serverem a prohlížečem, popřípadě jiným klientem. Tento protokol přenáší například obrázky, soubory, formulářová data atd. HTTP je bezstavový. Velikou výhodou je, že přes HTTP je možné komunikovat skrz různé platformy.

Jak již bylo uvedeno, architektura tohoto protokolu je klient-server. Komunikace probíhá pomocí zpráv. Způsobem, že klient vytvoří požadavek pomocí dotazovací metody a hlaviček a klient poté zpracuje požadavek. Poté vygeneruje příslušné hlavičky, návratový kód a zašle data, pokud jsou vyžadována.

```
GET /clanky/obsah.html HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT)
Host: www.server.cz
```

Obrázek 9 – Požadavek [22]

```
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Wed, 06 Dec 2000 13:37:40 GMT
X-Powered-By: PHP/4.0.3pl1
Content-type: text/html

<!DOCTYPE HTML PUBLIC '-//W3C//DTD HTML 4.0 Transitional//EN'>
<html>
<head>
<title>Dobývání znalostí z databází 2000</title>
<link rel="stylesheet" type="text/css" href="base.css">
...
```

Obrázek 10 – Odpověď [22]

4.4.1 Zprávy

Zprávy používají standard RFC 822.

Obrázek 11 – Struktura zprávy

První řádek

První řádek se liší podle toho, jestli se jedná o zprávu adresovanou klientovi nebo serveru. Pokud se jedná o zprávu serveru, tak zde bude dotazovací metoda a v opačném případě návratový kód.

Hlavičky

Hlavičky slouží jako meta data o odpovědi, požadavku nebo datech. Existují čtyři výchozí typy. Lze definovat ale i vlastní hlavičky.

Obecné hlavičky

Hlavičky, které se dají použít jak pro dotaz, tak i pro odpověď.

Hlavičky dotazu

Hlavičky použitelné pouze pro dotaz.

Hlavičky odpovědi

Hlavičky použitelné pouze pro odpověď.

Hlavičky těla

Hlavičky, co jsou meta data o tělu zprávy.

Tělo

Pomocí těla se přenášejí data zprávy.

4.4.2 Požadavek

Struktura požadavku je znázorněna na obrázku č. 11. Vysvětlení jednotlivých bloků je uvedeno níže.

Obrázek 12 – Struktura Požadavku

Metoda

Na prvním řádku musí být uvedena dotazovací metoda a URL cíle. Základní HTTP metody jsou:

GET

Metoda GET slouží pouze pro načtení požadovaných dat a na jejich obsah by neměla mít žádný vliv.

POST

Podobné jak u GETU, s tím rozdílem, že se přenáší i data v těle HTTP požadavku.

PUT

Metoda PUT slouží pro úpravu cílových dat.

DELETE

Metoda DELETE slouží pro odstranění cílových dat.

4.4.3 Odpověď

Struktura odpovědi je znázorněna na obrázku č. 13.

Obrázek 13 – Struktura odpovědi

4.4.4 Stavový kód

Stavový kód neboli návratový kód upřesňuje, s jakým výsledkem byl požadavek zpracován. Tento kód obsahuje tři číslice, kde první je v rozsahu od 1 do 5. Zároveň toto první číslo znázorňuje, do jaké kategorie patří určitý soubor znaků. Dále jsou tyto kategorie rozepsány spolu s nejdůležitějšími stavovými kódy.

1xx

Informativní kód

2xx

Úspěšné vyřízení požadavku

- 201 OK – Požadavek byl úspěšně zpracován.
- 202 Created – Byl vytvořen nový zdroj.
- 204 No Content – Požadavek byl úspěšně zpracován, ale není co poslat.

3xx

Přesměrování

4xx

Chyba severu

- 400 Bad request – Požadavek nelze zpracovat z důvodu, že jsou špatně zadány parametry a podobně.
- 403 Forbidden – Uživatel nemá práva na přístup k tomuto požadavku.
- 404 Not found – Adresa se na této doméně nenachází.
- 405 Method Not Allowed – V případě, že server požaduje jednu z možných metod a v požadavku je uvedena jiná.

5xx

Chyba na straně serveru

- 500 Internal Server Error – Při zpracování požadavku se vyskytl nespecifikovaný error.

4.4.5 Hlavičky

Jelikož, že existuje veliké množství hlaviček, jsou zde popsány pouze ty základní nebo ty, které se nejvíce týkají této kvalifikační práce.

Allow

Hlavička allow určuje, jaké dotazovací metody server podporuje.

Content-type

Popisuje formát dat ve zprávě. Mezi nejčastější hodnoty patří:

- text/html,
- text/plain,
- image/png,
- image/jpeg,
- application/json.

Date

Datum a čas vytvoření zprávy.

Localization

Obsahuje URL adresu, na kterou je klient přesměrován.

Server

Server obsahuje verzi a typ webového serveru.

User-Agent

User agent je hlavička, co obsahuje typ a verzi prohlížeče na kterém si klient prohlíží webové stránky.

4.5 Formáty přenosu dat na webu

Pro přenos dat ve webovém prostředí se ustálily převážně formáty XML a JSON, popřípadě jejich modifikace.

4.5.1 XML

XML je značkovací jazyk, který byl navrhnut pro práci a přenášení dat. Mimo definici dat umožnuje také jejich popis. Nemá pevně danou množinu elementů jako některé jiné

značkovací jazyky. Jazyk je velice jednoduchý na syntaxi a má pouze pár základních pravidel. [23]

```
<?xml version="1.0" encoding="utf-8"?>
<student univerzita="upce">
    <titul>Bc.</titul>
    <jmeno>Tomáš</jmeno>
    <prijmeni>Tvrdý</prijmeni
</student>
```

Obrázek 14 – Ukázka XML

4.5.2 JSON

JSON je textový formát co slouží pro přenášení dat. Jeho syntaxe vychází z programovacího jazyka JavaScript, ale stejně jako XML lze využít i na jiné platfomy. JSON je na první pohled čitelný a jeho parsování je jednoduché a rychlé. Obsahuje pouze data. [24]

```
"student": {
  "titul": "Bc.",
  "jmeno": "Tomáš",
  "prijmeni": "Tvrdý",
  "univerzita": "upce"
}
```

Obrázek 15 – Ukázka JSON

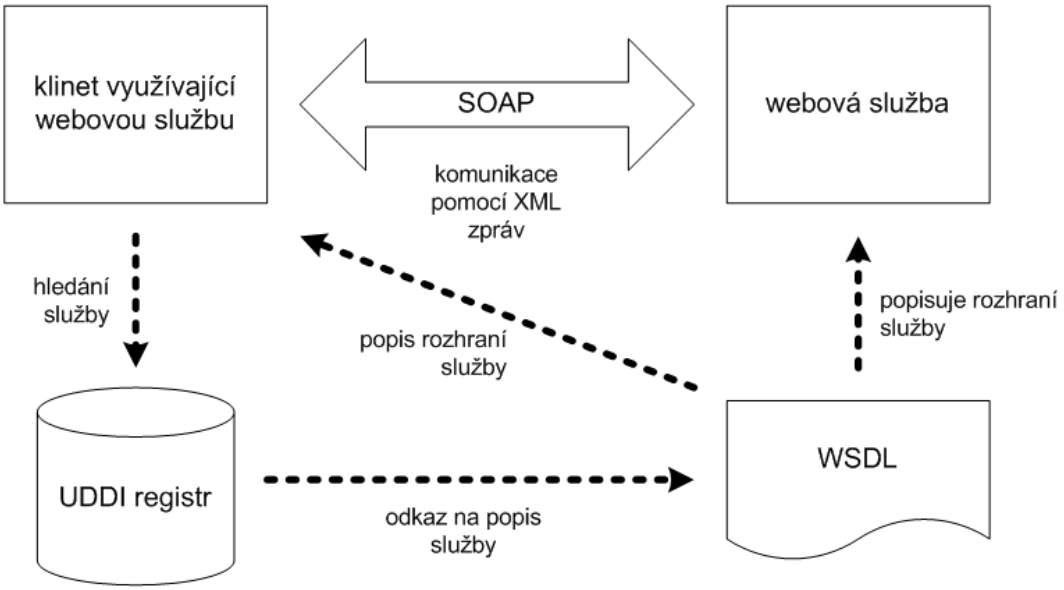
4.6 Webové služby

Webové služby jsou technologie pro propojení aplikací na síti. Velkou výhodou těchto služeb je, že používají protokol http. Z toho důvodu nezáleží, na jaké platformě je postavený klient nebo server a je umožněná komunikace i mezi různorodými aplikacemi.

4.6.1 SOAP

Celá podkapitola čerpá z [26]

Celý systém webových služeb má tři části, tj. SOAP, WSDL a UDDI. SOAP je protokol, který souží pro komunikaci a je založený na XML. WSDL popisuje jednotlivé rozhraní jednotlivých služeb a UDDI slouží pro jejich vyhledávání a vytvoření.



Obrázek 16 – Schéma webových služeb [26]

SOAP je středem této technologie, protože slouží pro samotnou výměnu zpráv. Vznikl jako první z těchto tří služeb. Metoda komunikace probíhá posláním zprávy na server, kterou tuto zprávu zpracuje a pošle odpověď.

Struktura zprávy

„Zpráva v SOAPu je jednoduchý XML dokument, který má kořenový element Envelope. *V této obálce jsou pak uzavřeny dva elementy* Header *(hlavička) a* Body *(tělo). Hlavička je přitom nepovinná a používá se pro přenos pomocných informací pro zpracování zprávy – například identifikaci uživatele, autentizační informace (jméno, heslo) apod. [26]*

O to nejdůležitější se stará tělo zprávy, v němž se přenášejí informace identifikující volanou službu a předávané parametry, resp. návratové hodnoty služby. SOAP používá jmenné prostory pro identifikování jednotlivých částí XML zprávy. Obálka, hlavičky a tělo zprávy patří do jmenného prostoru http://schemas.xmlsoap.org/soap/envelope/.“ [26]

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
   <SOAP-ENV:Body>
    <m:GetLastTradePrice xmlns:m="urn:x-example:services:StockQuote">
      <symbol>MOT</symbol>
    </m:GetLastTradePrice>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Obrázek 17 – SOAP požadavek [26]

```
<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
 SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
 <SOAP-ENV:Body>
  <m:GetLastTradePriceResponse
   xmlns:m="urn:x-example:services:StockQuote">
   <Price>14.5</Price>
  </m:GetLastTradePriceResponse>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Obrázek 18 – SOAP odpověď [26]

4.6.2 Rest

Celá podkapitola vychází z [27].

Rest architektura je datově orientovaná a slouží pro přístup a operaci s daty. Je založená na HTTP protokolu. Veškeré služby mají definovanou svoji adresu na serveru pomocí URL. Co se s daty provede, je určeno pomocí HTTP metody. Architektura definuje 4 základní metody zkráceně CRUD. V níže uvedené tabulce jsou tyto operace vysvětleny. Server po obdržení požadavku zahájí zpracování a následně vrátí příslušný stavový kód. Komunikace probíhá nejčastěji pomocí formátu JSON, ale může být použito i XML a další.

Zkratka (Význam)	HTTP metoda	Účel
C (Create)	POST	Vytvoří nové data
R (Read)	GET	Zisk dat
U (Update)	UPDATE	Úprava dat
D (Delete)	DELETE	Smazání dat

Tabulka 1 – CRUD operace

```
GET /vat/get-all HTTP/1.1
Host: localhost:8080
Authorization: Bearer 54cd642c-16a0-4334-b3a2-0cc7146afd40
Cache-Control: no-cache
```

Obrázek 19 – Požadavek

Status: 401 Unauthorized Time: 906 ms Size: 508 B

```
1 {
2     "error": "invalid_token",
3     "error_description": "Invalid access token: 54cd642c-16a0-4334-b3a2-0cc7146afd40"
4 }
```

Obrázek 20 – Odpověď

4.6.3 Srovnání

Celá podkapitola vychází z [28] a [29].

Před samotným porovnáním těchto přístupů je nutné zdůraznit, že není možné tyto technologie porovnávat přímo. Jelikož SOAP je protokol, který přesně definuje, jak probíhá komunikace a REST je architektonický vzor, tak je více abstraktní. REST může využívat pro komunikaci SOAP. SOAP je technologie, která pro komunikaci definuje výhradně formát XML. Na rozdíl od REST, který formát nemá definovaný a umožňuje použít JSON, XML, plain text a další.

REST využívá datový přístup, což znamená, že co se má provést je definováno URL adresou a HTTP metodou. Tato technologie se snaží co nejvíce využívat výhody HTTP protokolu. Na rozdíl od toho SOAP, pouze pro zaslání dat na server. Jelikož SOAP je orientovaný funkcionálně, tak co se má s obálkou provést je definováno až v těle obálky. U REST je velká výhoda, že z důvodu velkého využití HTTP protokolu je možné odpovědi cashovat a tím zvýšit jeho propustnost.

REST zmenšuje velikost datového toku, jelikož není nutné posílat další dodatečné informace a zasílají se pouze data. Tato výhoda se v dnešní době může zdát zanedbatelná, ale je potřeba si uvědomit, že pro aplikace, na které se připojuje veliké množství uživatelů nebo pro menší zařízení jako je mobilní telefon, které mají omezené množství dat, toto může hrát klíčovou roli při výběru spolu s možností cashování. REST je vždy bezstavový a při každém volání se musejí veškeré hlavičky posílat znovu na rozdíl od SOAPU, kde existují technologie se zachováním stavu.

Rest je vhodný pro použití v aplikacích, kde je kladen důraz na rychlost, jednotlivé volání nepotřebují informace z jiných a existuje několik zdrojů, které je vhodné cashovat. Stejně tak je velikou výhodou jeho jednoduchá implementace v základní podobě.

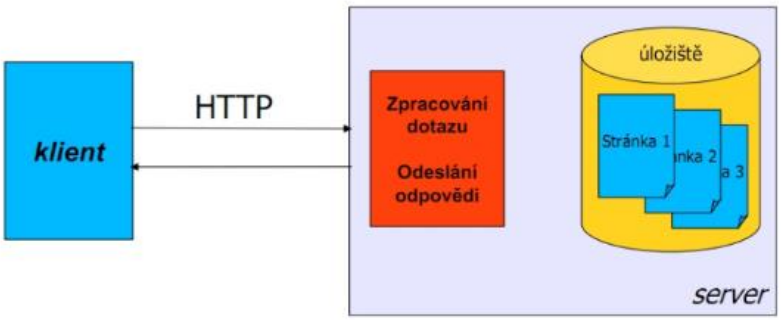
5 Architektura webové aplikace

Kapitola se zabývá možnými přístupy při tvorbě webové aplikace. Nejprve je představen historický přístup v podobě statických webových stránek a poté moderní přístupy tvorby webových aplikací.

5.1 Statické webové stránky

Celá podkapitola čerpá z [30] a [31].

Původní koncept webu byl převážně o přenosu souboru po síti. Název tohoto přístupu jsou statické webové stránky. Pro definování těchto dokumentu vznikly potřebné standarty jako jsou například HTTP, URL a další. Jedná se o aplikaci architektury klient-server, která se nazývá statické webové stránky. Komunikace probíhá tak, že klient pošle požadavek pomocí HTTP, kde je pomocí URL popsána adresa souboru na serveru. Server zpracuje požadavek a odešle požadovaný soubor klientovi, pokud je nalezen.

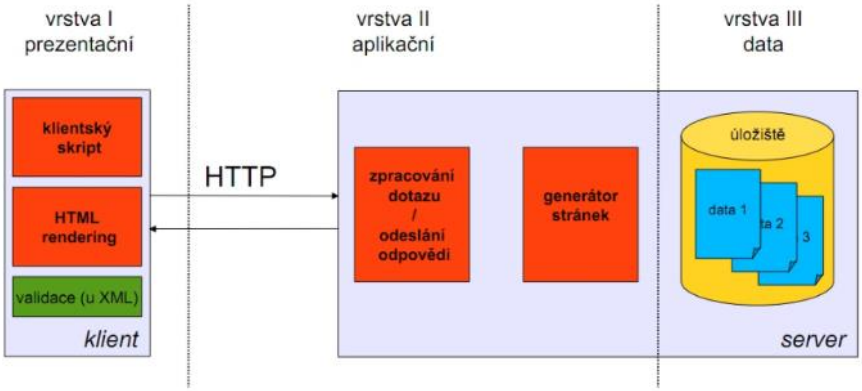


Obrázek 21 – Statický web [31]

Mezi největší výhody tohoto přístupu patří, že se změna obsahu souborů projeví okamžitě a jsou malé nároky na tvorbu webové stránky, kdy se používají pouze základní technologie, a tudíž je údržba také jednoduší v případě malé „aplikace“. Velké omezení této koncepce je zamezení dynamické generování odpovědí, ukládání dat atd., protože se v podstatě jedná pouze o posílání souboru po síti. V dnešní době je vhodný převážně pouze pro základní tvorbu prezentace formy bez ukládání dat, dynamické odpovědi atd.

5.2 MPA

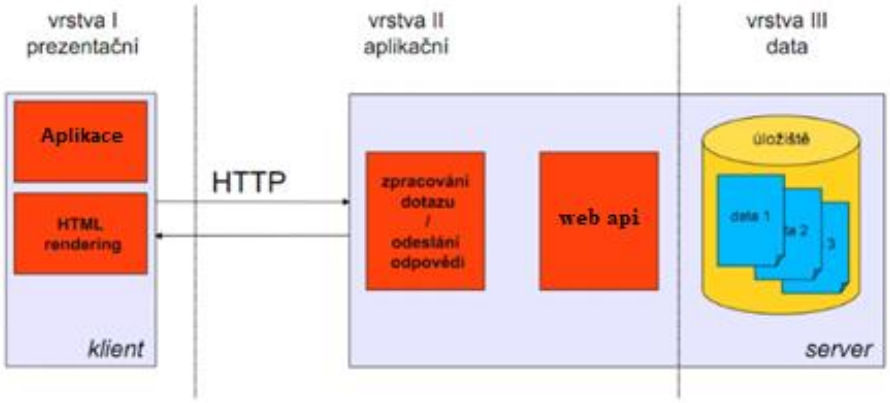
MPA se rozumí tradiční přístup k tvorbě webové aplikace. Na rozdíl od předchozího přístupu se na straně serveru nachází aplikace, která generuje výstup. V případě práce s daty aplikace standardně komunikuje s databázovým serverem, na kterém jsou data uložena. Komunikace probíhá tak, že klient pošle požadavek na server, který na základě požadavku vygeneruje celou HTML stránku. Při dalším požadavku se celá stránka přepíše nově příchozími daty. Této architektuře se také říká třívrstvá. Samozřejmostí je možnost nahradit databázový server pro uložení i jinými metodami, ale tyto přístupy se používají minimálně.



Obrázek 22 – MPA [31]

5.3 SPA

SPA jsou aplikace velice podobné jako MPA s jedním hlavním rozdílem. Na klientské části se nachází také aplikace, která pomocí webových služeb komunikuje s aplikací na serveru. Komunikace probíhá tak, že při prvním požadavku aplikační vrstva pošle klientovi aplikaci, a pak nadále probíhá komunikace přes webové služby a výsledná stránka se neobnovuje celá, ale pouze ta část, co je potřeba překreslit.

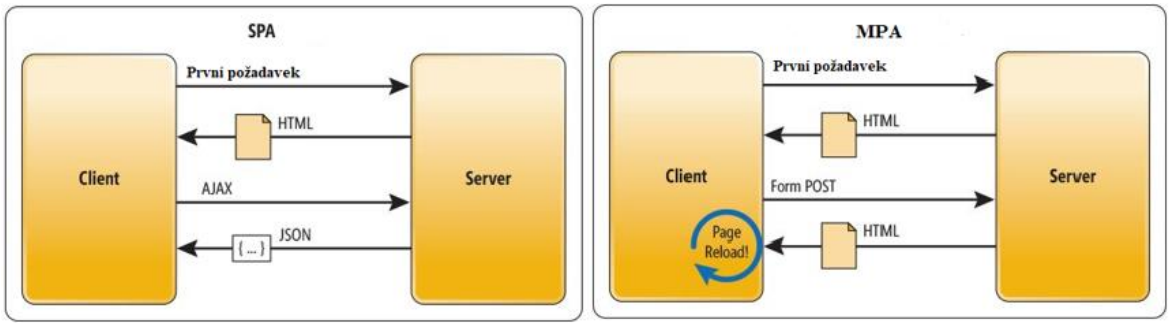


Obrázek 23 – SPA [31]

5.4 Srovnání

Celá podkapitola čerpá z [32], [33] a [34].

Výhody a nevýhody SPA vycházejí ze způsobu komunikace mezi klientskou a serverovou částí. MPA se po každém požadavku celá část klientské aplikace vykreslí znova a server musí pokaždé vygenerovat celý HTML kód znova. Na rozdíl od toho SPA při prvním požadavku pošle celou aplikaci, která má na starosti veškeré eventy od uživatele a přeposílá si z aplikační části pouze data.



Obrázek 24 – Srovnání komunikace SPA a MPA [35]

- **Výhody SPA**

Výhoda SPA architektury je výkon, jelikož server nemusí generovat žádné HTML a pracuje téměř jen s daty a renderování má na starosti klientská aplikace, takže se přenáší výkon na stranu klienta a s tím se snižuje i datový tok po síti. Jedná se o velkou výhodu převážně pro mobilní zařízení. S těmito vlastnostmi souvisí rychlejší odezva.

Klientská a serverová část jsou od sebe odděleny, tudíž aplikační část může nezávisle používat více webových, popřípadě jiných aplikací. Klasický případ pro využití aplikační části je rozšíření o mobilní aplikaci.

Z důvodu, že je na straně klienta jedna aplikace lépe se využívají klientská uložiště jako je například cookies a local storage. Tato vlastnost dokonce dává určitou, i když omezenou funkčnost aplikace i když je server ofline.

Dále je nutné uvést, že tato architektura je pro uživatele více příjemná, protože při zpracování požadavku MPA architektury se celá stránka načte znovu a uživateli to způsobí „přebliknutí“ na monitoru. Tato výhoda SPA se zvyšuje čím více je aplikace interaktivní.

Taková to aplikace se lépe testuje.

- **Nevýhody SPA**

Klient musí mít povolený JS, jinak je aplikace zcela nefunkční.

Zvyšuje časovou náročnost na vývoj aplikace z důvodu, že se vytvářejí dvě aplikace místo jedné, kde je zároveň nutné mít určité zkušenosti s JS.

Může být problém optimalizovat takovou aplikaci pro SEO.

První načtení stránky je velice pomalé, protože se musí stáhnout a spustit aplikace.

- **Výhody MPA**

Značnou výhodou je využití aplikace i pro prohlížeče bez podpory JS.

Tvorba takovéto aplikace je jednoduší, protože tvoříme pouze jednu aplikaci a pro její vývoj není nutné ovládat JS.

Také se lépe optimalizuje pro prohlížeče.

- **Nevýhody MPA**

Provázanost aplikační a klientské části.

Větší zátěž pro server a delší odezva.

Horší ovládání aplikace pro uživatele z důvodu častého „blikání“.

Použití

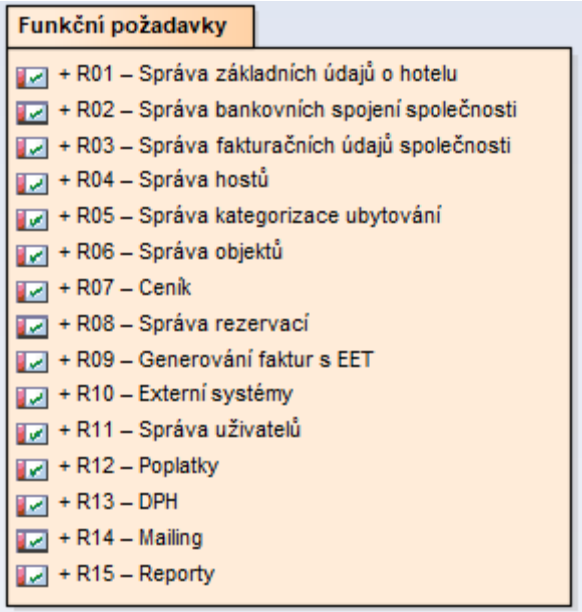
MPA architektura je vhodné použít pro menší aplikace, pokud je požadavek fungování stránky bez JS nebo popřípadě neznalosti JS. V komerčním prostředí také hraje velikou roli čas, kdy vývoj této aplikace bývá z pravidla rychlejší a tím méně nákladný.

SPA architektura je vhodná pro větší aplikace, kdy je možné aplikační část rozdělit na více aplikací a tím zvýšit udržitelnost systému, nebo v případě, kdy rozhraní aplikace budou využívat jiné systémy, pro tento účel není nutné psát moc dalšího kódu. Pokud je kritickou cestou budoucího systému výkon, propustnost, odezva nebo interaktivní ovládání.

6 Návrh aplikace

Kapitola návrh aplikace je stěžení částí práce. Je velice důležitá pro implementaci cílového systému. Slouží pro definování požadavků jak funkčních, tak nefunkčních na aplikaci. Dále se zabývá případy užití, analytickým modelem a datovým modelem.

6.1 Funkční požadavky



Obrázek 25 – Funkční požadavky

R01 – Správa základních údajů o hotelu

Nastavování základních údajů o hotelu. Eviduje se název hotelu, ulice, město, PSČ, plátce DPH a kraj.

R02 – Správa bankovní spojení společnosti

Nastavování aktuálních údajů o bankovním spojení. Eviduje se číslo účtu, kód banky, název banky, zkratka banky, IBAN a SWIFT.

R03 – Správa fakturačních údajů společnosti

Nastavování aktuálních fakturačních údajů. Eviduje se ulice, město, PSČ, IČO, DIČ, telefon, email a poznámky.

R04 – Správa hostů

Správa hostů obsahuje přidání, úpravu a mazání hostů. Host může být buď fyzická osoba nebo firma. U zákazníka se eviduje jméno, příjmení, telefonní číslo, email, pohlaví, datum narození, rodné číslo, titul před, titul za, trvalé bydliště a poznámky. U firmy se eviduje její adresa, IČO, DIČ a kontakt. K zákazníkům bude nutné přiřadit kategorie. U kategorií se zaznamenává název, slevy, popis, poznámky a věk. Veškeré úpravy v adrese se budou

uchovávat, aby v případě přístupu k historickým datům nedocházelo ke ztrátě konzistence dat. Tzn., že musí být evidován i rozsah platnosti.

R05 – Správa kategorií

Správa kategorií obsahuje přidávání, úprava a mazání objektů. Tyto kategorie představují typy objektů. Eviduje se název, počet lůžek, počet přistýlek, kapacita, typ rezervace a popis. Typ rezervace uvádí, jestli se bude rezervovat po dnech nebo minutách. Například wellness se rezervuje po minutách. Platba bude vždy pouze za celý pokoj. V případě wellnessu za dobu délky pobytu a poštu osob v zařízení. Zde bude také příznak, jestli se jedná o ubytovací zařízení, aby bylo možné určit, zda připočítat k ceně rekreační poplatky atd. Změna typu rezervace není možná.

R06 – Správa objektů

Správa objektů obsahuje vytváření, editace a mazání jednotlivých objektů. Každý objekt bude mít svojí kategorii. Eviduje se název a popis. Názvem je ve většině případů myšleno číslo pokoje, ale nemusí to tak být nutně a musí být unikátní.

R07 – Ceník

Ceníky se budou nastavovat na daný rozsah pro jednotlivé kategorie. Nastavování bude probíhat po určitých dnech (pondělí, úterý ...). Ceník se aplikuje podle počtu nocí strávených v hotelu. Toto platí pouze u ubytovacích objektů. Ceníky se vždy musejí nastavit na všechny dny, které se v ceníku nacházejí. Pokud je ceník již v platnosti, je nutné ho ukončit, aby se datumy nekryly s aktuálním dnem a vytvořit nový. V případě práce s rezervací, která nemá nastavenou cenu, je nutné, aby recepční byla upozorněna. Cena se vždy nastavuje za jednotku času nastavenou u typu objektu. Ceníky jsou nezávislé na konkrétních rezervacích slouží pouze pro generování položek na účet rezervace.

R08 – Správa rezervací

Rezervace budou dvojího typu. První typ bude pro rezervace ubytovacích zařízení a bude sloužit jako hlavní rezervace. Druhý typ je pro všechny ostatní objekty. Při vytvoření rezervace druhého typu je na účet hosta přidána položka na účet rezervace. Například pokud přijede rodina s jedním dítětem, tak se na objektu, na kterém je rodina ubytována vytvoří rezervace a při této příležitosti se vytvoří jedna položka na účtu. Zde je důležité, že na veškeré zaúčtované položky nemá žádný vliv jakákoliv úprava ceníku, této rezervace a podobně. Vše je nutné upravit ručně v účtu rezervace. V případě větší chyby na straně majitele hotelu je nutné vyhledat IT podporu. Na účet je možné také přidávat jakékoliv položky. V případě, že si rodina rezervuje bowling, wellness atd., je na účet vždy vložena položka s odpovídající cenou a popisem.

R09 – Generování faktur s EET

Systém bude umožňovat generování faktur. Platit bude možné pouze za celé rezervace. Faktury budou napojeny na EET a po vygenerování vytisknuty nebo zaslány na email. Veškeré doklady se uloží na disk. Při zadávání jednotlivých položek bude možné ručně upravovat ceny nebo přidávat položky. Je možné nastavit procentuální slevu. Měnit nebo přidávat ručně částky.

R10 – Externí systémy

Systém bude umožňovat připojení na externí systémy, jako je Trivago a další.

R11 – Správa uživatelů

Správa uživatelů obsahuje přidávání, úpravu a mazání uživatelů. Po přidání uživatele je na jeho email zaslán odkaz přes, který si nastaví heslo. Také je možné měnit heslo v průběhu. Minimální délka hesla je 5 znaků.

R12 – Poplatky

V systému bude možno nastavit poplatky, které se účtují ke každému dni jedné osoby na pokoji. V praxi se jedná o rekreační poplatky a podobně.

R13 – DPH

V aplikaci bude řešeno centrálně, jestli je zařízení plátce DPH.

Všude v aplikaci se bude zadávat cena s DPH. Cena bez DPH bude dopočítávána podle toho, jaká procentuální daň je na tuto položku vybrána.

R14 – Mailing

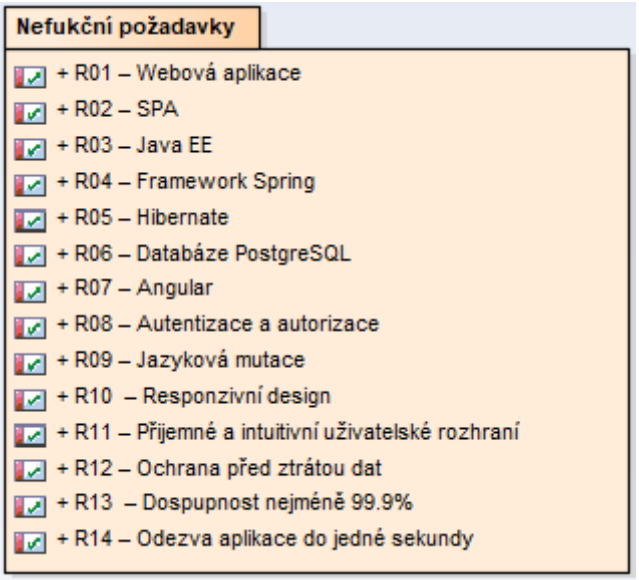
Bude možné rozesílat emaily konkrétním uživatelům ubytovaným k určitému datu.

R15 – Reporty

Reporty budou převážně pro cizineckou policii. Dalším důležitým reportem bude kniha hostů a obsazenost pokojů za určitá období.

6.2 Nefunkční požadavky

Nefunkční požadavky jsou v této kapitole zahrnuty z důvodu úplnosti tvorby požadavků. Podrobněji se jimi zabývají jiné kapitoly.



Obrázek 26 – Nefunkční požadavky

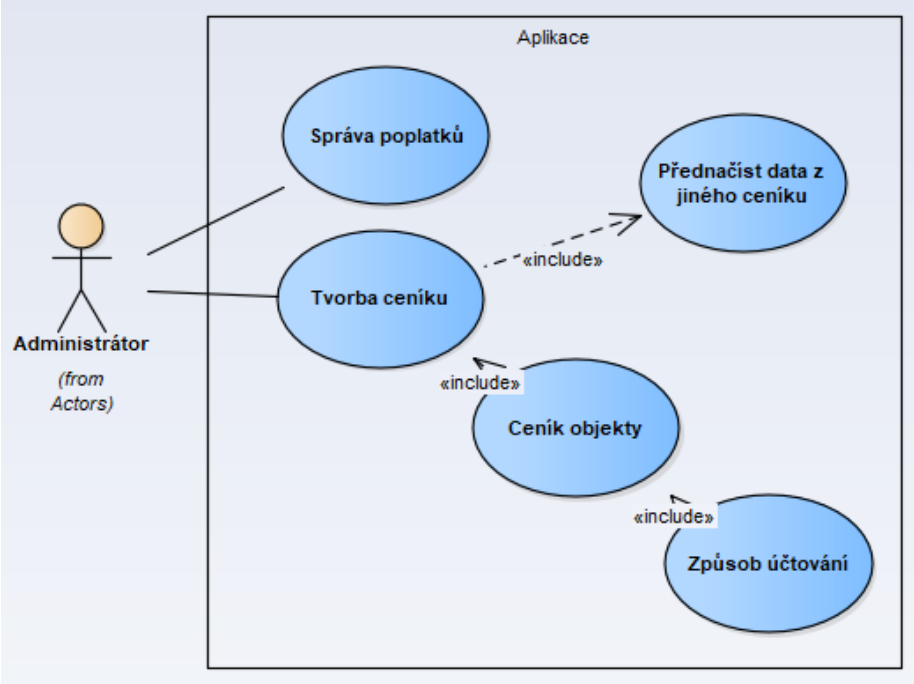
6.3 Případy užití

Zde budou uvedeny hlavní případy užití. Ostatní případy užití jsou dostupné v přiložených souborech přes aplikaci Enterprise Architect. Pod pojmem správa se skrývá zkratka operací přidat, upravit, zobrazit a smazat.

Aplikaci budou obsluhovat dva aktéři, a to recepční a administrátor. Tento výběr byl vybrán z důvodu, že IS je určen pro malé a střední ubytovací zařízení, kteří mají často pouze pár zaměstnanců.

Tvorba ceníků

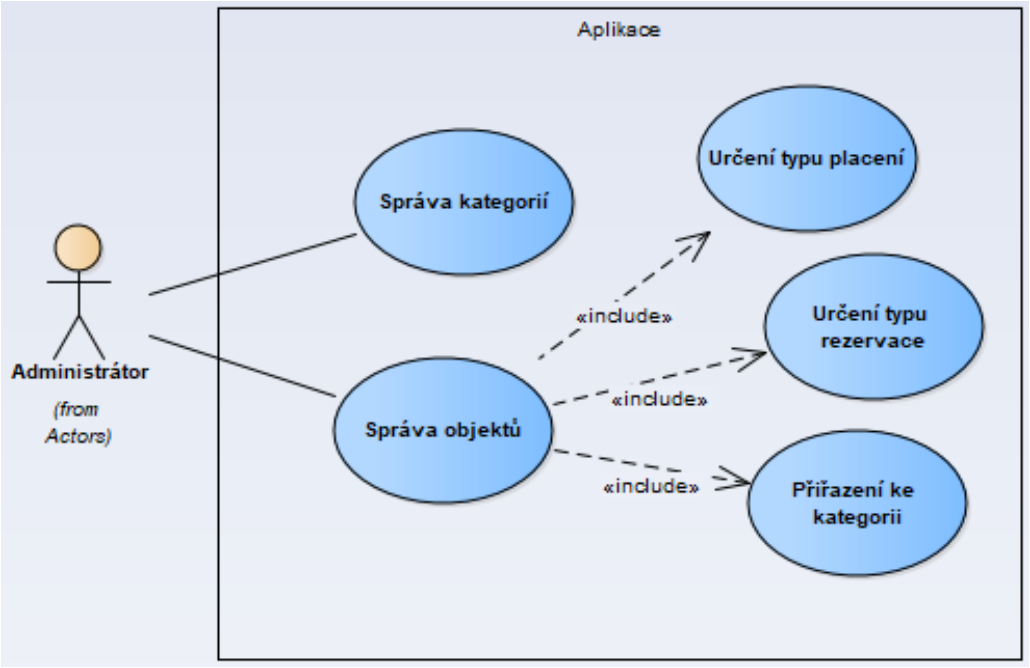
Tvorbu ceníků a poplatků má zpravidla na starosti administrátor. Ceny tvoří na jednotlivé období a pro jednotlivé objekty. To zahrnuje také příplatky.



Obrázek 27 – Use Case tvorba ceníků

Správa objektů

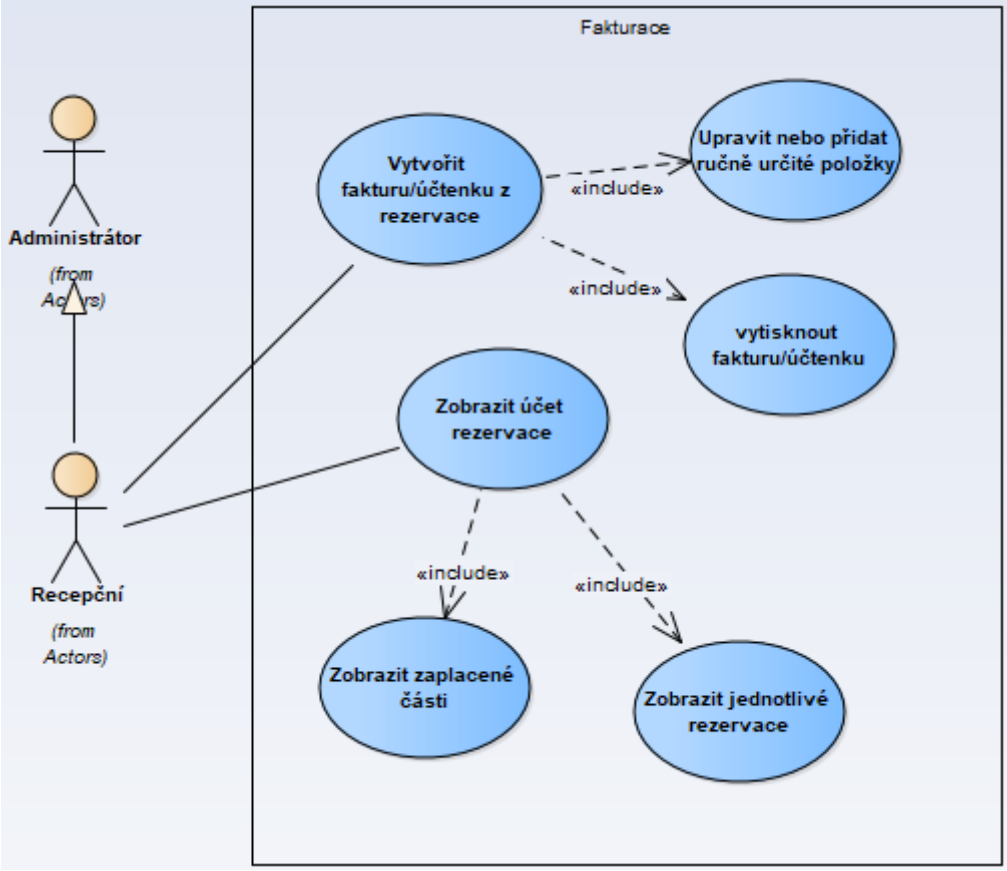
Na následujícím obrázku je znázorněna správa všech objektů. Tento diagram je již popsán ve funkčních požadavcích, tudíž je zde vynechán.



Obrázek 28 – Use Case tvorba objektů

Fakturace

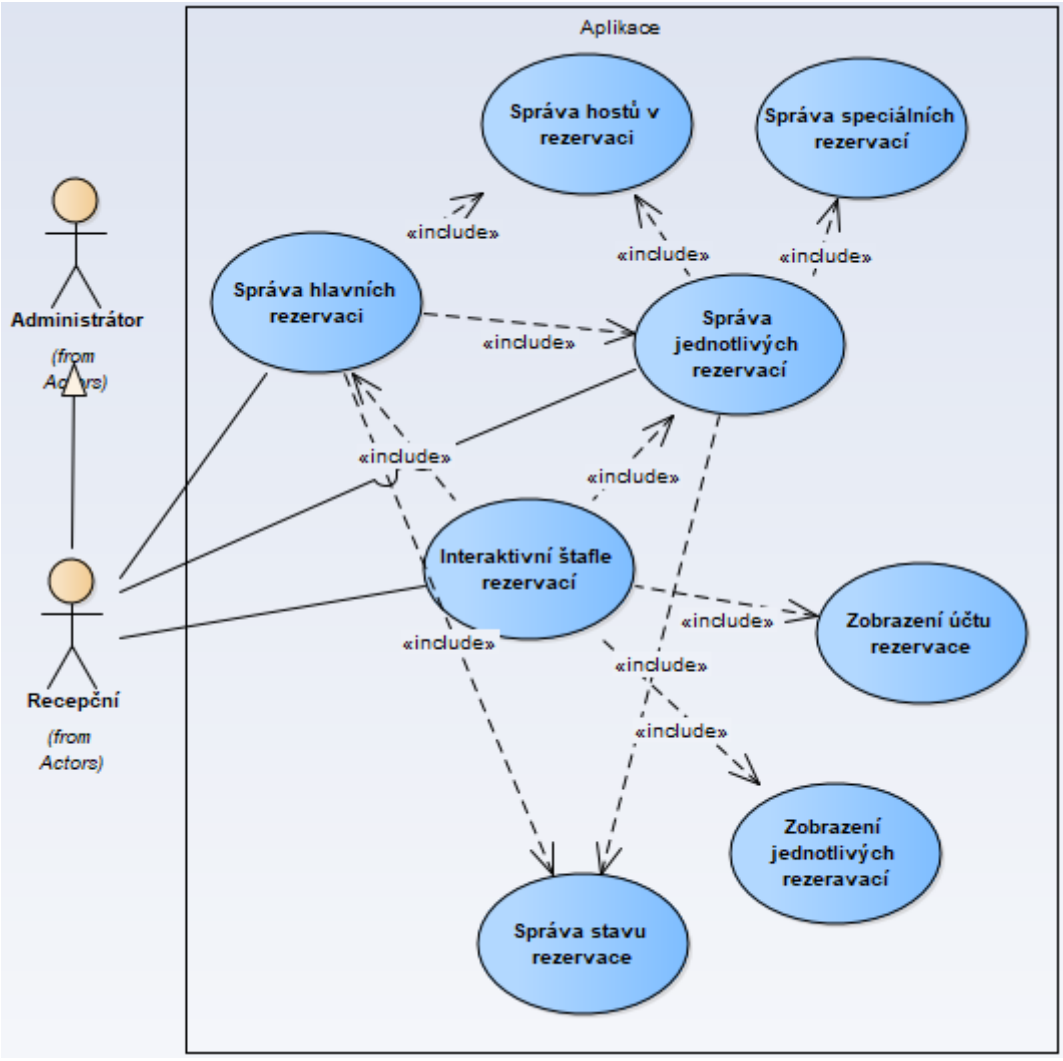
Popis fakturace byl popsán též v části funkčních požadavků, tudíž je zde pouze znázorněn diagram.



Obrázek 29 – Use Case fakturace

Rezervace

Recepční bude spravovat recepci převážně přes „interaktivní štafly“. Úprava bude možná také přes domovní knihu, dnešní přehled a seznam rezervací. Na kartě rezervace se ukáže seznam rezervací pro zadané období, následně uživatel systému bude moci kliknout na jednotlivé rezervace a po rozkliknutí se mu otevře modální okno, které bude obsahovat vše o rezervaci. Umožní ručně upravovat ceny, délky rezervace, přidávat rezervace, přidávat položky, psát poznámky například pro pokojskou atd. Z rezervací se zde budou vytvářet faktury. Také půjde přidávat a odebírat hosty z rezervace.



Obrázek 30 – Use Case rezervace

6.4 Analytický model

Analytický model byl vytvořen pomocí programu Enterprise architect. Tento diagram je z důvodu velikosti přiložen jako příloha – A. Je rozdělen do 6 oblastí, a to údaje o hotelu, hosté, ubytování, fakturace a rezervace. Zkratka operace _CRUD vychází z anglických slov create, read, update, delete a představují základní správu domény, kterou třída popisuje.

6.5 Databázový model

Databázový model byl vytvořen pomocí programu IntelliJ Idea. Z důvodu rozsáhlosti tohoto diagramu je přiložen jako příloha – B a jsou zde uvedeny pouze názvy tabulek a vztahy mezi nimi.

7 Klientská část

Kapitola klientská část rozebírá zvolené technologie na klientské části aplikace. Jelikož jsou některé technologie uvedeny již v kapitole číslo 3, nejsou zde dále rozebírány. Jedná se o HTML, CSS a JS. Na závěr kapitoly jsou popsány hlavní implementační části práce na klientské vrstvě.

7.1 LESS

LESS je preprocesor pro CSS. Přidává kaskádovým stylům nové funkce pro zjednodušení kódu a odstranění duplicity. Kód této technologie je velice podobný CSS a tudíž je velice jednoduchý na naučení pro vývojáře, kteří tento jazyk ovládají. Jedná se o open source projekt, který založil Alexis Sellier. [36] [37]

7.2 Bootstrap

Bootstrap je front-endový, výkonný, rychlý, intuitivní a responzivní framework. Patří v současnosti mezi nejpopulárnější. Výrazně napomáhá zrychlení vývoje webových stránek. Zakladateli jsou Mark Otto and Jacob Thornton. V současnosti je framework veden jako open source projekt. Byl vyvinut v HTML, CSS a JS za použití framworku JQuery. [38]

7.3 TypeScript

Jazyk JS byl původně vyvíjen pro práce s DOM na straně serveru. Nebyl navržen pro tvorbu samostatných velkých aplikací. Z tohoto důvodu vznikl TS. Stručně řešeno TS je JS s dalšími funkcemi. Mezi největší rozdíly patří silná typovost jazyka a rozšířená podpora objektově orientovaného programování. Při nasazení na web je nutná jeho kompilace do JS, protože prohlížeče tento jazyk nepodporují. Jeho velkou výhodou je, že v TS lze využít jakýkoliv program napsaný v JS. [38]

Obrázek 31 – TypeScript [39]

7.4 SPA

Jako architektura pro implementaci aplikace byla vybrána SPA jelikož, že s ní budou pracovat převážně recepční či administrátoři na stejných počítačích, a tudíž v tomto případě nevadí první pomalejší načtení klientské části. Zároveň je dobrý předpoklad, že bude povolený JS na frontendu. Problémy se SEO nejsou vůbec důležité, jelikož se jedná o interní aplikaci. Klíčovým prvkem pro výběr této architektury byl požadavek na tvorbu rozhraní pro restaurační systém, který při této architektuře bude vytvořen implicitně při tvorbě rozhraní pro

klientskou část a není nutné jakýmkoliv způsobem duplikovat tento požadavek. Rovněž pozitivně ovlivní v aplikaci nepotřebné znovunačítání stránek při každém požadavku na server, a to převážně při práci s rezervací pomocí interaktivní štafle.

7.5 Rest

Komunikace mezi klientem a serverem probíhá pomocí REST a formátu JSON. REST architektura byla vybrána pro plné využití HTTP protokolu, nepotřebnost stavu, rychlejší odezvu a jednodušší tvorbu rozhraní. Formát přenosu dat probíhá pomocí JSON pro další snížení odezvy a zmenšení dat přenášených po síti.

7.6 Angular

Celá podkapitola včetně obrázku čerpá z [40] a jeho podstránek.

Cílem této práce není popsat programování pomocí Angularu, ale představení technologie. Z tohoto důvodu je zde popsán pouze základní koncept tohoto frameworku. Pro jeho pochopení se předpokládá znalost TypeScript. Pokud by měl čtenář zájem o podrobnější vysvětlení je doporučená oficiální dokumentace [34], která je velmi dobře zpracována.

Před samotným popisem této technologie je třeba zdůraznit, že existují dvě podobně nazvané technologie. Jsou jimi AngularJs a Angular 2+. Pokud se v dalším textu této práce objeví Angular, vždy bude myšlen Angular 2+. Obě technologie vyvíjí společnost Google a jsou vhodné pro tvorbu klientské části SPA aplikace, ale jedná se úplně jiné jazyky. V případě bližšího zájmu o Angular popsaný v tomto textu je třeba se vyhnout Angularu, který je nižší než verze 2, protože v tomto případě se jedná o AngularJs.

Angular je framework napsaný pomocí jazyka TypeScript, který je určen pro tvorbu aplikací na webu. Je možné psát aplikaci pomocí Angularu i v JS, ale není to doporučeno, a i většina tutoriálů a knih se využívá jazyk TS. Framework je navržen převážně pro využití na klientské části aplikace pro architekturu SPA. Většina moderních aplikací se z Angularu zkompiluje do čistého JavaScriptu, který poté není příliš intuitivní ani čitelný.

7.6.1 Architektura

Základním prvkem Angularu jsou NgModules. Každý tento modul seskupuje určitý kontext pro komponenty. Komponenty mají šablony, které spolu s daty určují zobrazení komponenty v DOM a využívají pro tyto účely služby. Pro definování nejen komponent se v Angularu často využívají dekorátory, což je podobné anotacím v Javě. Moduly a komponenty mohou tvořit hierarchickou strukturu.

7.6.2 Moduly

Modul představuje základní stavební jednotku aplikace. Každá aplikace musí mít minimálně jeden tento element, který inicializuje aplikaci. Rozsáhlé aplikace jich ale mohou mít daleko více. Každý modul ve výchozím nastavení nesdílí svůj kontext s ostatními moduly. Moduly seskupují komponenty, direktivy, služby, roury atd.

Modul je třída, která obsahuje dekorátor @NgModule. Pomocí něho se nastavují vlastnosti modulu.

Declarations

Deklarace přidává komponenty, direktivy a služby do modelu. Tj. pokud vývojář chce použít v modulu komponentu a podobně, tak zde musí být uvedena.

Imports

Importy rozšiřují modul o další moduly. Zde se tedy rozšiřuje funkce modulu o funkčnost, která je viditelná z importovaných modulů.

Bootstrap

Bootstrap inicializuje aplikaci. Obsahuje pouze kořenová komponenta.

Providers

Pole providers slouží pro definování služeb, které můžou být i globální.

Ve zdrojovém kódu č. 1 je ukázána deklarace modulu, který má tři komponenty. V těchto komponentách jsou přístupné funkce z BrowserModule, BrowserAnimationsModule a HttpClientModule. Jelikož je zde i zmíněno klíčové slovo bootsrap, tak se jedná o rootovský modul aplikace. Modul také využívá služby třídy UserService.

```
@NgModule({
        declarations: [
                AppComponent,
                SignInComponent,
                SetPasswordComponent
        ],
        imports: [
                BrowserModule,
                BrowserAnimationsModule,
                HttpClientModule,
        ],
        bootstrap: [
                AppComponent
        ],
        providers: [
                UserService
        ],
})
export class UserModule{
}
```

Zdrojový kód 1 – Deklarace modulu

7.6.3 Komponenty

Komponenty tvoří po modulu další základní prvek. Komponenta má vždy definovanou šablonu a svůj vnitřní stav. Šablona definuje vzhled komponenty v prohlížeči v závislosti na datech. V nejjednodušším případě může být i nezávislá na vnitřním stavu. Syntaxe šablony je HTML rozšířený o selektory komponent a direktivy. Ve většině případů komponenty tvoří hierarchické uspořádání.

Selector

Selektor určuje, jakým tagem je komponenta vykreslena.

Template

Šablona definuje, jakým obsahem HTML bude tag nahrazen.

Ve zdrojovém kódu č. 2 je uveden příklad deklarace komponenty. Pro vykreslení této komponenty v jiných částech aplikace je určen selector reservation-guests-detail. Účelem třídy je vypsání všech rezervací hostů.

```
@Component({
        selector: 'reservation-guests-detail',
        template: `
                <h1>Hosté</h1>
                <guest-detail *ngFor="let g of guets" [guest] = "g">
                </guest-detail>
                `,
})
export class ReservationGuestsComponent implements OnInit {

  @Input('guests') guests: Guest[];

}
```

Zdrojový kód 2 – Deklarace komponenty

Roura

Roura slouží pro proměnu vypsané hodnoty na požadovaný formát.

Ve zdrojovém kódu č. 3 je uveden příklad, kdy je datum narození převeden pomocí roury do čitelného formátu. Tato roura je v práci nejpoužívanější.

```
<td> {{guest.dateBirthday | date 'dd.MM.yyyy'}}</td> [40]
```

Zdrojový kód 3 – Použití roury

Data binding

Angular poskytuje mechanismy pro posílání dat mezi komponentou a DOM tento proces se nazývá data binding. Pořadí definice je totožné se znázorněním na obrázku.

Interpolace

Interpolace slouží pro zobrazení dat v HTML.

Interpolace na úrovni komponenty

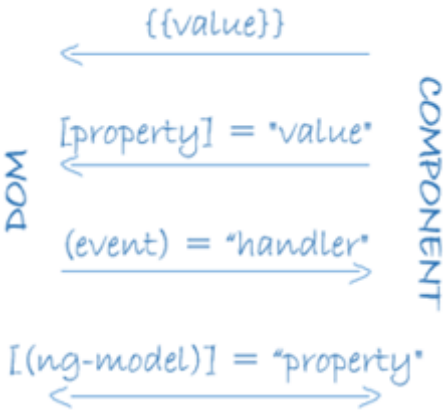
Interpolace na úrovní komponenty slouží pro výměnu dat mezi komponentami.

Eventy

Eventy slouží pro nastavení callbacků pro zpracování událostí jako je kliknutí, hover, focus, změna a podobně.

Obousměrný data binding

Obousměrný data binding slouží pro obousměrnou komunikaci. Největší význam má pro zpracování vstupů, kde okamžitě po změně hodnoty změní i data v komponentě, která jsou na tento vstup namapována.



Obrázek 32 – Data binding [40]

Direktiva

Šablony v Angularu jsou dynamické, tudíž se o jejich strukturu a konečnou podobu starají direktivy. Komponenta je zvláštní druh direktivy. Existují dva druhy direktiv strukturní a atributové.

Strukturální direktivy

Strukturální direktivy, které upravují strukturu DOM pomocí mazání, přepisování atd.

Mezi nejpoužívanější patří *ngFor a *ngIf. První z nich slouží pro procházení kolekcí a druhý je podmínka, zda se bude komponenta, popřípadě tag zobrazovat.

Příklad strukturální direktivy je uveden ve zdrojovém kódu č. 2, kdy jsou v cyklu procházeny veškeří hosté rezervace.

Atributové direktivy

Atributové direktivy pouze upravují vlastnosti direktiv.

Ve zdrojovém kódu č. 4 je uveden příklad atributové direktivy ngClass, která přidává třídu do svého elementu. Jestli tato třída bude do divu přidána závisí na ternárním operátoru a to konkrétně na tom, jestli se jedná o sobotu, neděli nebo pracovní den.

```
<div class="day" [ngClass]="data.isWeekend() ? 'week' : '' ">
```

Zdrojový kód 4 – Atributová direktiva

7.6.4 Vkládání závislostí

Vkládání závislostí bude podrobněji vysvětleno v další kapitole, zde je uveden pouze základní mechanismus pro pochopení architektury Angularu.

Angular automaticky vkládá závislosti, pokud má komponenta dekorátor @Injectable. Služby, funkce, nebo hodnoty, které je potřeba „injektnout“, musí být korektně načteny v modulu a poté uvedeny v konstruktoru v parametrech.

Ve zdrojovém kódu č. 5 je ukázán příklad vkládání závislosti třídy HttpClient do třídy ReservationDetaulCompont. Při takovéto definici konstruktoru je atribut třídy http vložen automaticky Angularem při vytvoření objektu.

```
@Injectable()
@Component({
        //…
})
export class ReservationDetailComponent {

        constructor(public activeModal: NgbActiveModal
        ,private http:HttpClient) {}
//…
}
```

Zdrojový kód 5 – Vkládání závislostí

7.6.5 HttpClient

Angular je framework vytvořený převážně pro SPA. Pro komunikaci se serverem se využívá služba HttpClient, která má na starosti tuto komunikaci pomocí XMLHttpRequestu. Odpovědí je třída Observable, která je rozšířenou implementací návrhového vzoru Iterátor.

Ve zdrojovém kódu č. 6 je vyobrazeno naplnění číselníku vat přes HttpClienta. Nejprve třída HttpClient odešle požadavek na adresu definovanou v URL_GET_VATS, kde jsou data, která byla přijata, umístěna a následně metodou subscribe předána do parametru data. V případě chyby do parametru err i s chybovými hláškami.

```
this.http.get<Vat[]>(this.URL_GET_VATS,{ headers: reqHeader }).
subscribe((data) => {
  this.vats = data;
},(err) => this.error = err);
```

Zdrojový kód 6 – HttpClient

7.6.6 Routování

Pro navigaci v aplikaci slouží routování. Pro pochopení této problematiky je třeba si uvědomit, že v klasických webových aplikacích (MPA) je po každém odeslání formuláře nebo kliknutí na odkaz uživatel přesměrován na jinou URL (za určitých okolností může zůstat stejná) a tato adresa dává uživateli jasnou odpověď v jaké sekci aplikace se nachází. URL se mění, ale jen zdánlivě. Fakticky se jen mění obsah, který se zobrazuje na stránce, která je od prvního okamžiku spuštěna. Pokud se uživatel chce dostat o krok zpět, tak je to možné. Takto

jednoduché by to bohužel v SPA aplikaci nebylo, jelikož co konkrétně prohlížeč zobrazí, není závislé na URL adrese, ale na datech uvnitř komponent. Z tohoto důvodu existuje routování, které přináší do aplikace navigaci.

Zdrojový kód č. 7 ukazuje příklad navigace v rootovské aplikaci, kdy adresa login vykresluje komponentu SignInComponent a adresa pages předává řízení navigace na modul PagesModule.

```
const routes: Routes = [
  {
    path: 'login',
    children: [{ path: '', component: SignInComponent }]
  },
  { path: 'pages',loadChildren: 'app/pages/pages.module#PagesModule' },
  …
];

@NgModule({
  imports: [RouterModule.forRoot(routes, config)],
  exports: [RouterModule],
})
export class AppRoutingModule {
}
```

Zdrojový kód 7 – Routování

7.7 PrimeNG

Celá podkapitola čerpá z [42].

PrimeNG je opensourcová UI knihovna pro Angular 2. Byla vyvinuta společností PrimeTek Informatics. Obsahuje více než 80 komponent, mezi které patří například textové vstupy, datepickery, dropdownboxy, slidery, našeptávače, tlačítka, komponenty pro zobrazování dat, různé druhy vyskakovacích dialogů, kalendář, souborové vstupy a mnoho dalších. Velkou výhodou této knihovny je responzivita.

7.8 Aplikace

Celá aplikace je rozdělena do dvou modulů.

První modul slouží pro definici všech nastavení a kalendáře pro objekty, které jsou definovány s rezervacemi po minutých. Veškeré komponenty v této části využívají knihovnu PrimeNg.

Druhý modul je implementace vlastního kalendáře pro správu a zobrazení rezervací pro ubytovací typ rezervací.

7.9 Data

Pro práci s daty byly využity dvě hlavní komponenty z PrimeNG, a to p-table a p-dialog. Tyto prvky jsou využity pro většinu správ uživatelů, daní, ceníků, hostů atd.

7.9.1 P-table

P-table slouží pro vypsání dat do tabulky. Pro práci s daty je možné nastavit filtrování, stránkování, řazení, export do csv, události, grupování, skrolování atd.

V níže uvedeném zdrojovém kódu je vyobrazen zkrácený příklad použití p-table na výpisu nastavení daní. Columns definuje klíče a názvy sloupců v tabulce ve vztahu k datům. Data se nastavují přes parametr value. U tabulky se aktivuje stránkování atributem paginator a možnosti jeho nastavení podle atributu rowsPerPageOptions. Rows určuje původní nastavení stránkování. OnRowSelect nastaví callback metodu, která zpracovává událost kliknutí na řádek.

V těle tohoto elementu je nejprve umístěn informativní nadpis s tlačítkem pro přidání daně, které otevře dialogové okno. Poté dochází k samotnému vykreslení hlavičky tabulky, kde je samozřejmostí možnost seřazení podle jakéhokoliv sloupce v případě, kdy je více sloupců. Ve většině ostatních případů následuje ng-template s filtrama podle hodnoty.

Poslední částí vysvětlení p-table je samotné vypsání dat.

```
<p-table #dt  [columns]="cols"  [value]="vats" [paginator]="true"
 [rowsPerPageOptions]="[10,20,50]"          [rows]="20"
selectionMode="single"         (onRowSelect)="onRowSelect($event)"
[(selection)]="selectedVat">
  <ng-template pTemplate="header" let-columns>
    <h4>
      <span style="text-align:left">
        <button type="button" pButton icon="fa fa-plus "
                (click)="showDialogToAdd()" label="Přidat"></button>
      </span>
    </h4>
    <tr>
      <th *ngFor="let col of columns" [pSortableColumn]="col.field">
        {{col.header}}
        <p-sortIcon [field]="col.field"></p-sortIcon>
      </th>
      <th style="width:6em"></th>
  </ng-template>
  <ng-template pTemplate="body" let-vat let-rowData>
    <tr [pSelectableRow]="rowData">
      <td>{{vat.percent}}</td>
        …
    </tr>
  </ng-template>
</p-table>
```

Zdrojový kód 8 – P-table

7.9.2 P-dialog

P-dialog je modální okno a je využíváno pro přidávání nebo úpravu dat.

Ve zdrojovém kódu č. 9 je znázorněna zkrácená ukázka dialogu pro tvorbu daně. Dialog je zobrazen v případě, že atribut komponenty displayDialog je nastaven na true. Tuto hodnotu nastavuje buď metoda onRowSelect a nebo showDialogToAdd. Tyto metody jsou znázorněny v předchozím zdrojovém kódu.

Následně jsou vypsány vstupy. V případě, že se přidává nová daň, tak se validátory aktivují až po prvním potvrzení formuláře. V případě úpravy stávající daně jsou aktivovány ihned.

```
<p-dialog header="Daň"
          [(visible)]="displayDialog" [responsive]="true"
          [modal]="true" [width]="400" >
  <form #vatForm="ngForm">
    <div class="ui-g ui-fluid" *ngIf="vat">
      <div class="ui-g-12">
        <div class="ui-g-4">
          <label for="percent">Daň</label>
        </div>
        <div class="ui-g-8">
          <input pInputText required id="percent"
                 #percent="ngModel" name="percent"
[(ngModel)]="vat.percent" />
        </div>
      </div>
      <div class="ui-g-12">
        <p-message severity="error" text="Daň je povinná."
                   *ngIf="percent.invalid&&displayValid"></p-message>
      </div>
      //dalsi validacni hlasky
    </div>
    <p-footer>
      <div class="ui-dialog-buttonpane ui-helper-clearfix">
          //tlacitka
      </div>
    </p-footer>
  </form>
</p-dialog>
```

Zdrojový kód 9 – P-dialog

7.10 Rezervace wellness

Pro rezervaci wellness pokojů byl využit modul z knihovny PrimeNG.

Ve zdrojovém kódu č. 10 je znázorněno nastavení kalendáře v komponentě. Z důvodu, že tato komponenta je velice komplexní a umožňuje zobrazení tří agend, je nejprve nutné nastavit, aby se zobrazovala pouze agenda akcí po minutách za týden. Dále je nastavena možnost orientovat se v kalendáři tlačítky další a předchozí.

V dalším kódu je ukázka vykreslení plánovače v šabloně komponenty. V této fázi jsou vypnuté veškeré funkce s událostmi v kalendáři, kromě funkce kliknutí na událost, kdy se zobrazí příslušné modální okno pro její správu.

```
this.header = {
  left: 'prev,next',
  center: 'title',
  right: '',
};
this.defaultView = 'agendaWeek';
```

Zdrojový kód 10 – Nastavení kalendáře

```
<p-schedule  *ngIf="object" class="scheduler" [allDaySlot]="false"
            (onEventClick)="handleEventClick($event)"
            [defaultView]="defaultView"
            [events]="events"
            locale="cs"
            [header]="header"
            [defaultDate]="today"
            [eventLimit]="4"
            [editable]="false"
            [droppable] = "false"
            [eventStartEditable] = "false"
            [eventDurationEditable] = "false">
</p-schedule>
```

Zdrojový kód 11 – Vykreslení kalendáře

7.11 Rezervace pokojů

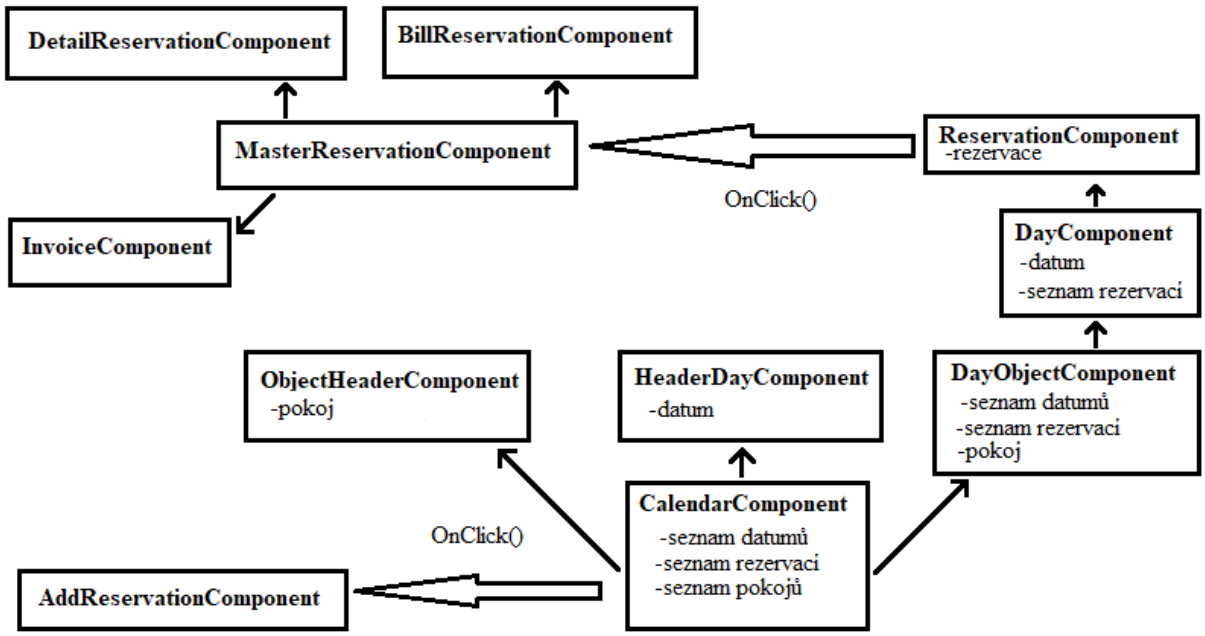
Pro správu a zobrazení kalendáře byl vytvořen vlastní modul, zjednodušené schéma této části je zobrazeno na obr. č. 33.

Rootovská komponenta modulu je CalendarComponent, která si po inicializaci stáhne veškerá data, jako jsou rezervace a pokoje. Rozsah těchto rezervací je při prvním spuštěním od 7. předcházejícího dne do měsíce dopředu. Tento interval je možné změnit filtrem.

Renderování kalendáře mají na starosti ObjectHeaderCoponent, HeaderDayComponent, DayObjectComponent, DayComponent a ReservationCompont, z nichž první tři jsou na stejné úrovni v hierarchickém uspořádání. ObjectHeaderComponent dostává od rootovského elementu pokoje, které vypisuje a touto činností tvoří levé menu. Hlavičku dní vypisuje HeaderDayComponenta, která obdobným způsobem dostává dny.

DayObjectComponent postupně vypisuje jednotlivé dny pro pokoje, kde instance jednoho dne je DayComponent. V případě renderování poslední zmíněné komponenty se vždy ověřuje, zda neexistuje pro tento pokoj na daný den rezervace. Pokud ano, je vykreslená přes ReservationComponent.

Správa jednotlivých rezervací probíhá po kliknutí na jejich vizualizaci v kalendáři, kde se aktivuje modální okno MasterReservationComponent. Tato komponenta přepíná mezi záložkami, kterými jsou účet, detail a faktura. Přidávání rezervací je umožněno přes AddReservationComponent.



Obrázek 33 – Modul kalendář

8 Aplikační část

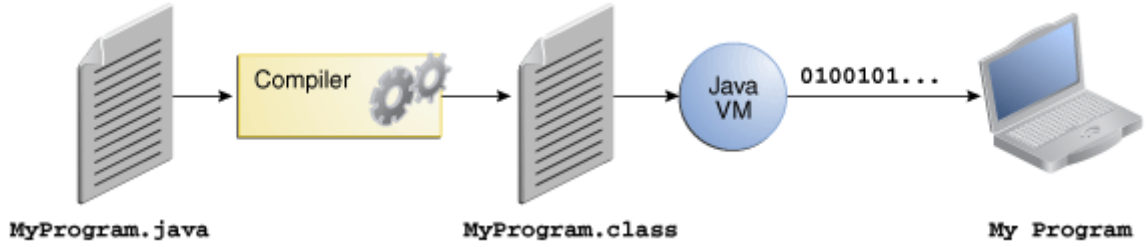
Kapitola aplikační část se zabývá technologiemi použité na aplikační vrstvě. Úvod se zabývá napřed jazykem Java, ve kterém je aplikace dle zadání naprogramována. Dále se pokračuje v definici webového serveru. Následně je vysvětlena koncepce frameworku Spring a v samém závěru je popsán celá architektura a stěžejní části aplikace.

8.1 Java

Celá podkapitola čerpá z [43] a [44].

Java je vysokoúrovňový multiplatformní objektově orientovaný jazyk. Při vývoji byl inspirován jazyky C a C++. Jazyk vytvořil tým pod vedením Jamesem Goslingem v roce 1990 s původním názvem Oak. Přelomovým rokem pro tuto technologii byl rok 1995, kdy Javu představila firma SunMicroSystems. Nové verze tohoto jazyka vycházejí dodnes.

Kompilování Java kódu probíhá tak, že zdrojový kód v souborech s příponou Java je zkompilován do ByteCode. Na této úrovni ještě není kód spustitelný na procesorech a je pořád multiplatformní. Pro spuštění aplikace na dané platformě se používá Java Virtual Machine, který interpretuje kód na příslušnou platformu. Tento postup má dvě hlavní výhody, těmi jsou, jeden kód pro všechny platformy a možnost sdílení kódu napříč jazyky, které podporují ByteCode.



Obrázek 34 – Kompilace [42]

Existuje více edicí Javy, zde jsou uvedeny pouze základní, které se týkají této práce.

Java SE

Java SE je základní edice, kde jsou definovány základní konstrukce, typy a podobně.

Java EE

Java EE je nadstavba nad SE. Jsou zde přidány části, které zjednodušují tvorbu rozsáhlejších aplikací.

8.2 Apatche Tomcat

Apatche Tomcat je opensourcový webový server, který implementuje Javovské webové technologie, jakými jsou Servlety, JSP a podobně. [44]

8.3 Maven

Maven je plugin pro správu závislostí, který byl vytvořen společností Apache Software Foundation. Umožňuje i buildování aplikací. Jeho hlavní výhoda je, že vývojář nemůsí do projektu přidávat žádné JAR soubory. Stačí vložit potřebnou komponentu, knihovnu a podobně na potřebné místo v pom.xml a vše ostatní zařídí Maven. [45]

Ve zdrojovém kódu č. 12 je ukázaný nejjednodušší případ definice závislostí pro aplikaci napsané v projektu Spring Boot.

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
         <version>2.0.3.RELEASE</version>
    </dependency>
</dependencies>
```

Zdrojový kód 12 – Maven závislosti

8.4 Spring

Celá tato podkapitola čerpá z [45] a [46].

Spring je opensourcový framework pro tvorbu robustných Java aplikací. Jeho typickou vlastností je možnost vývoje různorodých druhů aplikací, jako jsou například desktopové, webové a další. Framework vznikl v říjnu v roce 2002. Aktuální verze frameworku je 5.0.6.

Jelikož je framework velice robustní, bude zde popsána pouze základní koncepce s důrazem na vysvětlení části Spring využité v implementaci hotelového systému.

8.4.1 Moduly

Celá sekce čerpá z [47].

Spring se z důvodu rozsáhlosti a přehlednosti člení do zhruba 20 modulů. Ty jsou nadále členěny do skupin podle obrázku č. 38, kde jsou zvýrazněny moduly, které byli využity při implementaci.

Core conteiner

Core a Beans

Core a Beans představují střed frameworku. Modul Core musí mít každá Spring aplikace, protože se přes něj sdílejí potřebná data pro propojení modulů. Jsou zde také definovány IoC a Dependenci Injection. Beany slouží primárně pro konfiguraci a implementaci faktory implementace.

Kontext

Kontext rozšiřuje funkce Core modulu.

AoP

Slouží pro aspektově orientované programování.

Data Access/Integration

JDBC

JDBC Modul má na starosti práci s databází a umožňuje jak tvorbu nativní SQL, tak platformě nezávislé dotazy.

ORM

ORM rozšiřuje funkčnost JDBC. Zprostředkovává mapování mezi objektovým a relačním prostředím. Zahrnuje technologie, mezi které patří Hibernate, JDO, JPa a další.

Transactions

Modul transakce přidává možnost provádění transakcí nad POJO objekty.

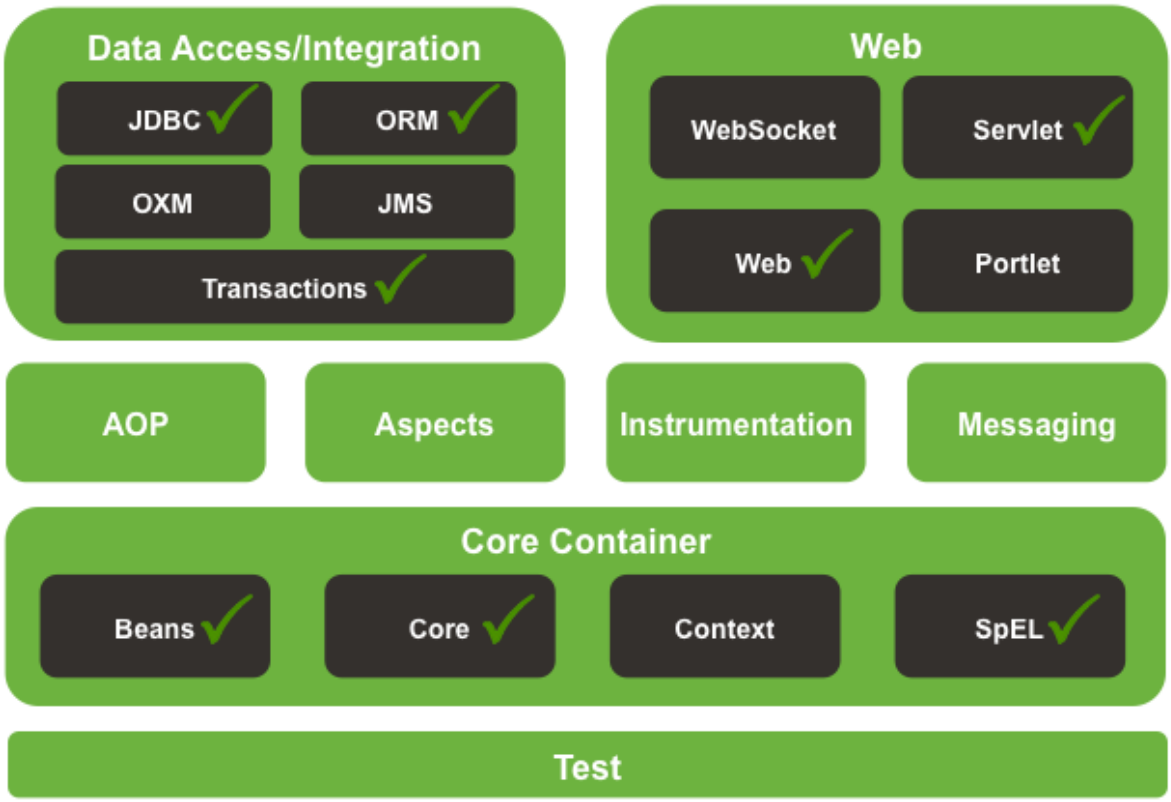
Web

Web

Modul web má na starosti integraci aplikace jako webového serveru, tj. hlavně nastavení a naslouchávání.

Web-mvc

Web-mvc rozšíří aplikaci o MVC architekturu a umožní použití webových služeb.



Obrázek 35 – Moduly [47]

8.4.2 IoC

Inversion of control je návrhový vzor, který říká, že potřebné závislosti vytváří externí zdroj, a tudíž se nemají vytvářet ve třídě, která pro svou činnost vyžaduje další třídy. Například existují třídy autoservis a mechanik. Autoservis má na starosti opravu aut a ke své činnosti potřebuje mechanika. V klasickém programování bude objekt mechanik vytvořen v konstruktoru třídy autoservis. Aplikací výše zmíněného návrhového vzoru bude tato závislost vyřešena pomocí externí služby. Způsoby řešení tohoto problému jsou uvedeny ve zdrojových kódech 13,14 a 15.

```
public class AutoServis{

        private IMechanik mechanik = new Mechanik();

        public AutoServis(){

        }

}
```

Zdrojový kód 13 – Vytváření závislostí bez DI

DI

Dependency Injection je návrhový vzor zabývající se vkládáním závislostí mezi objekty.

DI je konkrétní implementace IoC. Tyto pojmy bývají často zaměňovány, jako je tomu například v oficiální dokumentaci Springu. Pokud je debata na téma DI, je vždy myšleno IoC, ale opačně to neplatí. Existují i jiné implementace jako například návrhový vzor Service locator, Dependency pull a další.

Mezi největší výhody používání DI patří dobrá testovatelnost kódu, a to převážně z důvodu, že závislosti nejsou definovány napevno a můžeme tak využít jakoukoliv třídu implementující potřebné rozhraní.

Závislosti jsou definovány centrálně, což pomáhá při orientaci mezi nimi. Vyměnit závislost v třídě je mnohem jednoduší stačí vyhledat konfigurační soubor a změnit ji zde. Stejně tak podporuje lepší OOP návrh, jelikož implicitně dodržování tohoto návrhového vzoru slouží pro větší modularitu aplikace. Další výhodou je redukce množství a duplicity kódu. To je umožněno tím, že třída má své závislosti definované na jednom místě a pokud je potřeba v jiné části kódu není zde nutné tyto závislosti vytvářet znovu jak je tomu bez vyžití DI.

Existují dva hlavní způsoby DI vkládání konstruktorem a vkládání metodou.

Vkládání konstruktorem

Závislost je dodána v konstruktoru.

```
public class AutoServis{

        private IMechanik mechanik;

        public AutoServis(Mechanik mechanik){
            this.mechanik = mechanik;
        }

}
```

Zdrojový kód 14 – Vkládání konstruktorem

Vkládání metodou

Závislost je dodána setrem.

```
public class AutoServis{

    private IMechanik mechanik;

    public void setMechanik(Mechanik mechanik) {
        this.mechanik = mechanik;
    }

    public AutoServis(){}

}
```

Zdrojový kód 15 – Vkládání setrem

Jelikož vzniká ručním vkládáním závislostí mnoho „zbytečného“ kódu navíc, využívá se tzv. IoC kontejner, který vše inicializuje za vývojáře podle konfigurace.

DI ve Springu

Hlavní rozhraní ve Springu, které se stará o IoC kontejner, je BeanFactory. Veškeré třídy, které má tento kontejner na starosti, se nazývají Beany. Cílem tohoto přístupu je pouze řešení DI. Nevýhoda vlastní implementace tohoto rozhraní je jeho náročnost. Pro načtení konfigurace jsou připraveny třídy PropertiesBeanDefinitionReader, XmlBeanDefinitonReader a další. Pro konfiguraci nejen v IoC kontejneru slouží XML soubory, konfigurační třídy, anotace a Groody bean definiční metody. Přístupy ke konfiguracím se dají kombinovat.

Příklad základní implementace BeanFactory je uveden ve zdrojovém kódu č. 16. V tomto případě jsou závislosti ze souboru xml-bean-factory-config.xml.

```
public class XmlConfigWithBeanFactory {
    public static void main(String... args) {
        DefaultListableBeanFactory factory =
                        new DefaultListableBeanFactory();
        XmlBeanDefinitionReader rdr =
                        new XmlBeanDefinitionReader(factory);
        rdr.loadBeanDefinitions(
             new ClassPathResource("spring/xml-bean-factory-config.xml")
        );
        Oracle oracle = (Oracle) factory.getBean("oracle");
        System.out.println(oracle.defineMeaningOfLife());
    }
}
```

Zdrojový kód 16 – BeanFactory [48]

Další způsob, jak implementovat IoC kontejner, je třída ApplicationContext, ta je rozšířením BeanFactory a přidává funkce, jako jsou transakce, i18n, zpracování událostí AoP služby a další.

Beany získané z IoC kontejneru mají tzv. instanční mód, který určuje, životní cyklus Bean. Zde jsou probrány základní módy.

Singleton

Singleton je výchozí mód. Název podle návrhového vzoru Singleton. V systému vždy existuje pouze jedna instance a ostatní objekty na ní dostávají referenci.

Prototype

Prototype pokaždé se vytváří nová instance.

Request

Reguest slouží pro Spring MVC při každém požadavku je vytvořen object a následně zničen.

Session

Session podobně, jak request, pouze doba platnosti je na session.

8.4.3 Spring JDBC

JDBC je standardní rozhraní pro připojení k DB dostupné v Java SE edici. Mezi největší nevýhody tohoto rozhraní patří neustálé opakování kódu. Vývojáři Springu vytvořili vrstvu nad klasickým JDBC, která tento problém řeší a umožňuje lepší mapování parametrů přímo z POJO. Konkrétní nativní kód si vytvoří Spring JDBC automaticky. [48]

8.4.4 JPA

JPA je specifikace ORM mapování v Javě SE a Javě EE. V tomto standartu jsou definovány koncepty, anotace a rozhraní pro práci s ORM.

8.4.5 Hibernate

Hibernate je ORM framework. Jedná se o implementaci JPA. Stará se o mapování relačního světa do objektového. Umožňuje mapování vazeb 1:1, 1:M, M:1 a dokonce i M:N.

Ve zdrojovém kódu č. 17 je uveden příklad mapování 1:M mezi tabulkami user a role, a to přes tabulku user_role. Anotace OneToMany definuje kaskádový typ, který určuje, jak se má chovat relace v situacích jako jsou ukládání, mazání atd. Fechtype určuje, jestli se data relace načítají hned (EAGER) nebo až v případě potřeby (LAZY). Druhá anotace JoinTable definuje, přes jaké tabulky a atributy v DB je relace definována.

```
@OneToMany(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER)
@JoinTable(name = "user_role", joinColumns = @JoinColumn(name = "user_pk"),
inverseJoinColumns = @JoinColumn(name = "role_pk"))
private Set<Role> roles;
```

Zdrojový kód 17 – Hibernate 1:M mapování

8.4.6 Spring Data JPA

Spring Data JPA je rozšíření JPA ve Springu. Největší výhodou tohoto projektu je vygenerování základních CRUD operací bez psaní kódu. Postačí implementace příslušného rozhraní. Projekt Spring DATA JPA definuje tři základní repozitáře, těmi jsou CrudRepository, JpaRepository, a PagingAndSortingRepository.

Tato rozhraní mají hiearchickou strukturu. Na nejnižší úrovni je CrudRepository. CrudRepository rozhraní vygeneruje základní CRUD operace jako jsou například smazání záznamu podle klíče, načtení všech záznamu, uložení záznamu a tak dále. Funkci tohoto rozhraní rozšiřuje PagingAndSortingRepository, které implicitně přidává možnost stránkování a seřazení záznamu a poslední JpaRepository úak rozšiřuje o funkce specifické v JPA.

Ve zdrojovém kódu č. 18 je ukázáno deklarování JpaRespoitory. V tomto případě si lze všimnout, že definováním metod tohoto rozhraním lze docílit automatického generování kódu od JPA. Důležitým atributem v nastavení je exported, pokud by byl tento parametr true, tak třída User není automaticky serializována/deserialitována, tam kde je použita jako relace.

```
@RepositoryRestResource(path= "users",colletionResourceRel =
"users",exported = false)
public interface UserRepository extends JpaRepository<User, Integer> {
    Optional<User> findByLogin(String username);
    Optional<User> findByCode(String code);
}
```

Zdrojový kód 18 – JpaRepository

8.4.7 Validace

Celá podkapitola čerpá z [49].

Validace jsou součástí každé dobré aplikace. Je důležité si uvědomit, že v případě použití SPA architektury je nutné validovat vstupy jak na straně klienta, tj. v případě této práce Angular, tak na straně serveru. Mnoho lidí se nad tímto faktem podiví, ale je důležité si uvědomit, že uživatel aplikace, který chce server, jakkoliv zneužít, může kdykoliv poslat data bez využití frontendové aplikace, například třeba pomocí aplikace Postman. Tím může způsobit v případě absence validátorů na straně serveru v lepším případě pouze nekonzistenci dat, v krajním případě i smazání všech dat a podobně. Ve Springu existují dva hlavní způsoby implementace validátoru.

Výchozí

Spring nabízí sadu vlastních validátorů. Mezi základní patří:

- @NotNull – Nesmí být Null,
- @AssertTrue – Jestli je hodnota true,
- @Size – Interval, v jakém se může hodnota pohybovat,
- @Min – Minimální hodnota,
- @Max – Maximální hodnota,
- @Email – Kontrola formátu emailu.

Vlastní

Samozřejmostí je vytvoření vlastního validátoru například pomocí implementace ConstraintValidator rozhraní.

Ve zdrojovém kódu č. 19 je uvedena definice vlastního validátoru pro kontrolu zadání stejného hesla. Takový validátor musí definovat rozhraní ConstrainValidator. Metoda initializace slouží pro načtení dat z atributů třídy, kde je použita metoda isValid a zjisťuje, jestli jsou zadané údaje validní.

```
public class FieldsValueMatchValidator implements
ConstraintValidator<FieldsValueMatch, Object> {

    private String field;
    private String fieldMatch;

    public void initialize(FieldsValueMatch constraintAnnotation) {
        this.field = constraintAnnotation.field();
        this.fieldMatch = constraintAnnotation.fieldMatch();
    }

    public boolean isValid(Object value, ConstraintValidatorContext
context) {

        Object fieldValue = new
BeanWrapperImpl(value).getPropertyValue(field);
        Object fieldMatchValue = new
BeanWrapperImpl(value).getPropertyValue(fieldMatch);

        if (fieldValue != null) {
            return fieldValue.equals(fieldMatchValue);
        } else {
            return fieldMatchValue == null;
        }

    }

}
```

Zdrojový kód 19 – Vlastní validátor [45]

8.4.8 Spring MVC

MVC

MVC je návrhový vzor. Název tohoto vzoru je zkratkou jeho komponent model, controler a view.

Model

Model reprezentuje data jako taková. Tato data popisují aktuální stav, který se zobrazuje. V modulu se také provádějí veškeré operace nad daty jako jsou validace, uložení, načtení podle parametru, mazání, editace a podobně.

View

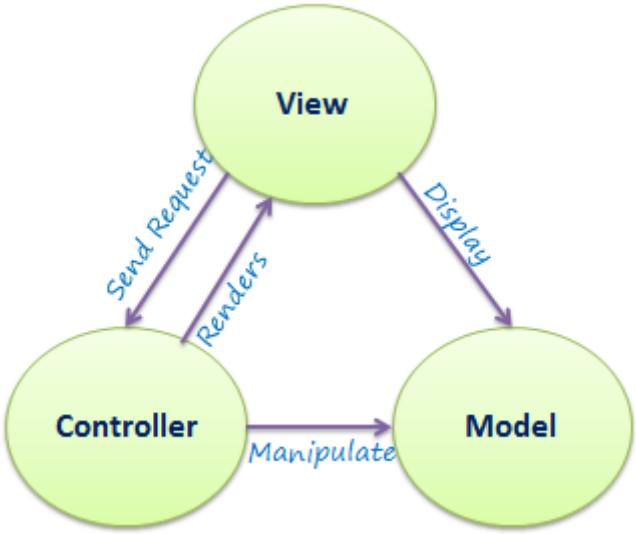
Pohled má na starosti zobrazení dat. Je zde možné použití i18n, validace na straně klienta a podobně.

Kontrolér

Kontrolér se stará o řízení procesu zobrazování, ukládání atd. Například pokud chce uživatel načíst detail auta s primárním klíčem 15, postup reakce MVC architektury bude:

1. Uživatel klikne na odkaz s URL adresou detailu auta, které si vybral.

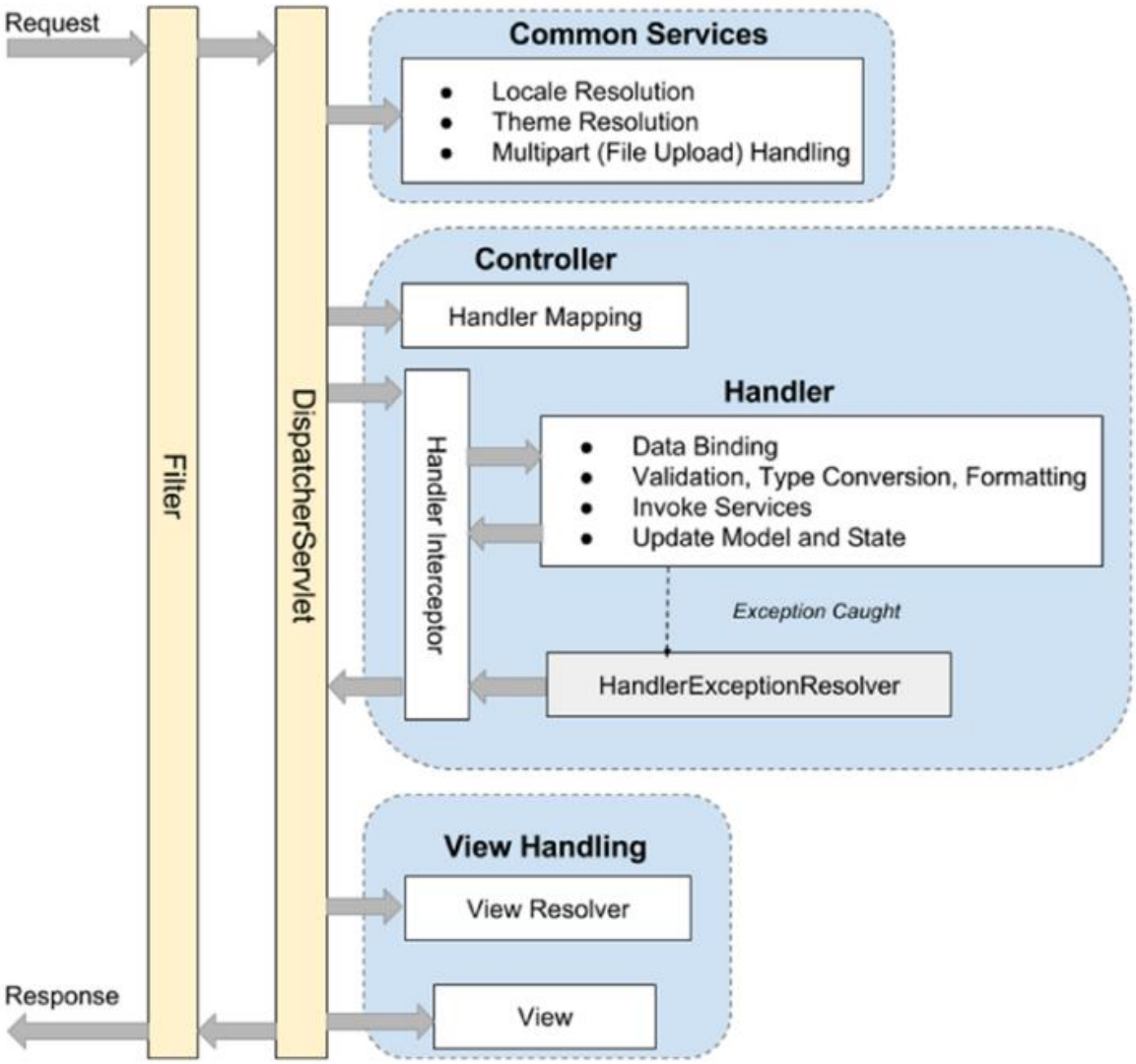
2. Webový server zpracuje požadavek a ten se pomocí mapování předán na příslušný kontrolér, který má tuto činnost na starost.
3. Metoda zpracování požadavku si pomocí objektu „Auto“ najde hledané auto.
4. Vyhledané auto vrátí v odpovědi.
5. Pohled vykreslí auto.



Obrázek 36 – MVC [50]

Spring MVC

Ve Spring má zpracování všech požadavků na starosti DispatcherServletu servlet. Těchto tříd může být více a spolu můžou tvořit hierarchickou strukturu. Před jakýmkoliv požadavkem je aplikován filtr, který ho může i zablokovat. Následně je doručen na instanci DispatcherServletu. Jedná se o implementaci návrhového vzoru Front Kontrolér. Tento návrhový vzor zjednodušeně říká, že hlavní kontrolér mapuje požadavky na další vedlejší kontroléry. Všem požadavkům na daném DispatcherServletu jsou dostupné common služby. Po rozhodnutí, na jaký kontrolér bude požadavek zpracován, určí Handler mapping, jaká metoda zpracovává požadavek. Následně metoda zpracuje pomocí beany požadavek a je a výsledek je odeslán klientovi. Tento výsledek může být za pomocí View Resolveru v případě MPA, kdy je potřebné nakonfigurovat, kde se načítají pohledy pro renderování. V případě SPA aplikace je objekt deserializován a vrácen v požadovaném formátu, což je v naprosté většině případů XML, JSON nebo od nich odvozené formáty.



Obrázek 37 – Spring WebApplicationContext [45]

8.4.9 Spring Boot

Při vytváření jakéhokoliv projektu ve Springu je velký problém s počáteční konfigurací, která zabere hodně časových prostředků nejen pro začínající vývojáře, kteří se Springem začínají. Z tohoto důvodu vývojáři vytvořili projekt Spring Boot. Tento projekt má přednastavené některé paramenty a je možné ho ihned používat pro vývoj bez dalších příslušných konfigurací. Řeší dokonce i závislosti mezi moduly. Vývojář má také možnost nastavit komponenty dle vlastní libosti.

Ve zdrojovém kódu číslo 16 je ukázkáno, jak se inicializuje aplikace. Nad třídou, která Spring spouští, je potřebné uvést anotaci SpringBootApplication a ve většině případů v main metodě zavolat statickou metodu run.

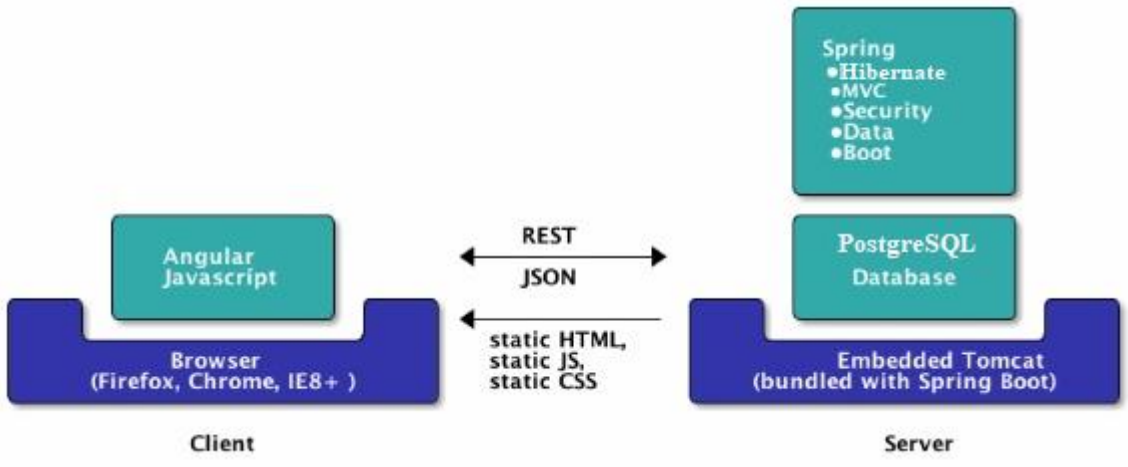
```
@SpringBootApplication()
public class DemoApplication {

    public static void main(String[] args) {
        SpringApplication.run(DemoApplication.class, args);
    }

}
```

Zdrojový kód 20 – Spuštění Spring aplikace

8.5 Architektura

Architektura aplikace má dvě hlavní části, kterými jsou klient a server. Na straně klienta je po prvním požadavku spuštěna Angular 6 aplikace, která komunikuje se serverem a vyměňuje si data pomocí formátu JSON.

Zpracování požadavků má na starosti Apache Tomcat, kde následně proběhne předání řízení Springu, který zkontroluje pomocí modulu Spring Security autorizace a autentizace. V případě, že tyto operace proběhnou úspěšně, je v případě potřeby aktivován Hibernate pro načtení dat z DB, která jsou následně poslána klientovi.



Obrázek 38 – Architektura aplikace [51]

8.6 Rozhraní pro napojení externích rezervačních a restauračních systémů

Před samotným popsáním rozhraní této aplikace zde bude rozebrán způsob jeho tvorby. Tento popis bude ukázán na zdrojovým kódu č. 21.

Kontrolér pro rozhraní se definuje pomocí anotace RestController. Dále je vhodné definovat doménovou adresu mapování na kontrolér, aby nedocházelo k zadávání redundantních adres.

Mapování na konečnou adresu je možné provést pomocí více způsobů. V tomto případě je použita anotace GetMapping. Následně se vyhledají všechny daně. Pokud není nalezena ani jedna daň je vracen stavový kód NO_CONTENT. V případě, že jsou daně již zadány, tak jsou odeslány se stavovým kódem OK.

```java
@RestController
@RequestMapping({"/vat"})
public class VatController {

    @Autowired
    private VatService vatService;

    @GetMapping("/get-all")
    public ResponseEntity<?> getAllVats() {
        List<Vat> vats = vatService.getAll();
        if(vats.isEmpty())
            return new ResponseEntity(HttpStatus.NO_CONTENT);

        return new ResponseEntity<>(vats,HttpStatus.OK);
    }
    //další metody
}
```

Zdrojový kód 21 – RestController

Hlavní rozhraní aplikace je zobrazeno v tabulce č. 2. Ověření uživatele probíhá pomocí tokenu, který obdrží po přihlášení.

Metoda	Adresa	Akce
GET	/guest/get-all	Vrátí všechny hosty
GET	/guest/get-all-in-day/{date}	Vrátí všechny hosty ubytované v datumu
GET	/guest/get-all-in-interval/{date_from}/{date_to}	Vrátí všechny hosty ubytované v itervalu
GET	/category/get-all	Vrátí všechny kategorie
GET	/object/get-all	Vrátí všechny objekty
GET	/object/get-all/{categoryPk}	Vrátí všechny objekty v kategorii
GET	/reservation/get-all	Vrátí všechny rezervace
GET	/reservation/get-by-number/{number}	Vrátí rezervaci podle jejího čísla
GET	/reservation/get-all-between-date/{date_from}/{date_to}	Vrátí všechny rezervace v dabém intervalu
POST	/reservation/create	Vytvoří rezervaci
PUT	/reservation/update	Upraví rezervaci
GET	/bill-item/get-all/{reservation}	Vrátí všechny položky na účtu
POST	/bill-item/create	Vytvoří položku na účtu
PUT	/bill-item/update	Upraví položku na účtu
DELETE	/bill-item/delete	Smaže položku na účtu
GET	/payment-method/get-all	Vrátí způsoby placení
GET	/price-list/get-all	Vrátí ceník
GET	/price-list/get-all/{categoryPk}	Vrátí ceník podle kategorie

Tabulka 2 – Rozhraní

8.7 EET

Připojení fakturačního systému na EET je provedeno přes open source knihovnu třetí strany a to z důvodu, že vlastní implementace knihovny pro tuto službu je velmi rozsáhlá. Příkladem může kvalifikační práce uvedena v seznamu literatury č. 52.

Ministerstvo financí podporuje dvě prostředí, produkční a testovací. Posílání dat do těchto prostředí má dva módy. V prvním se evidují tržby a ve druhém se pouze kontroluje, zda je správně navázána komunikace. [52]

Testování této knihovny proběhlo pouze v testovacím režimu. V případě využití systému na produkčním prostředí je třeba otestovat funkčnost v produkčních módech.

Na níže uvedeném obrázku je ukázáno základní vytvoření vybudování požadavku pro evidování obecné tržby.

```
EetRegisterRequest request=EetRegisterRequest.builder()
            .dic_popl("CZX")
            .id_provoz("X")
            .id_pokl("X")
            .porad_cis("X")
            .dat_trzby("yyyy-MM-ddThh:mm:ss+0Z:00")
            .celk_trzba(X)
            .rezim(X)
            .pkcs12(loadStream(getClass().getResourceAsStream("/X.p12")))
            .pkcs12password(X)
            .build();
```

Zdrojový kód 22 – Vytvoření EET požadavku

8.8 Emaily

Před samotnou tvorbou tříd pro zasílání emailu je nutné navit mailer. Zde je velká výhoda, že je použit projekt Spring Boot a stačí nastavit pouze základní vlastnosti viz. Zdrojový kód č. 22.

```
spring.mail.host=smtp.gmail.com
spring.mail.port=587
spring.mail.username=st43170@student.upce.cz
spring.mail.password=ENC(9S7B5Gu4jX/XLOm63Tyej5eWsHHhHb/9)
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.mail.default-encoding=UTF-8
```

Zdrojový kód 23 – Nastavení maileru

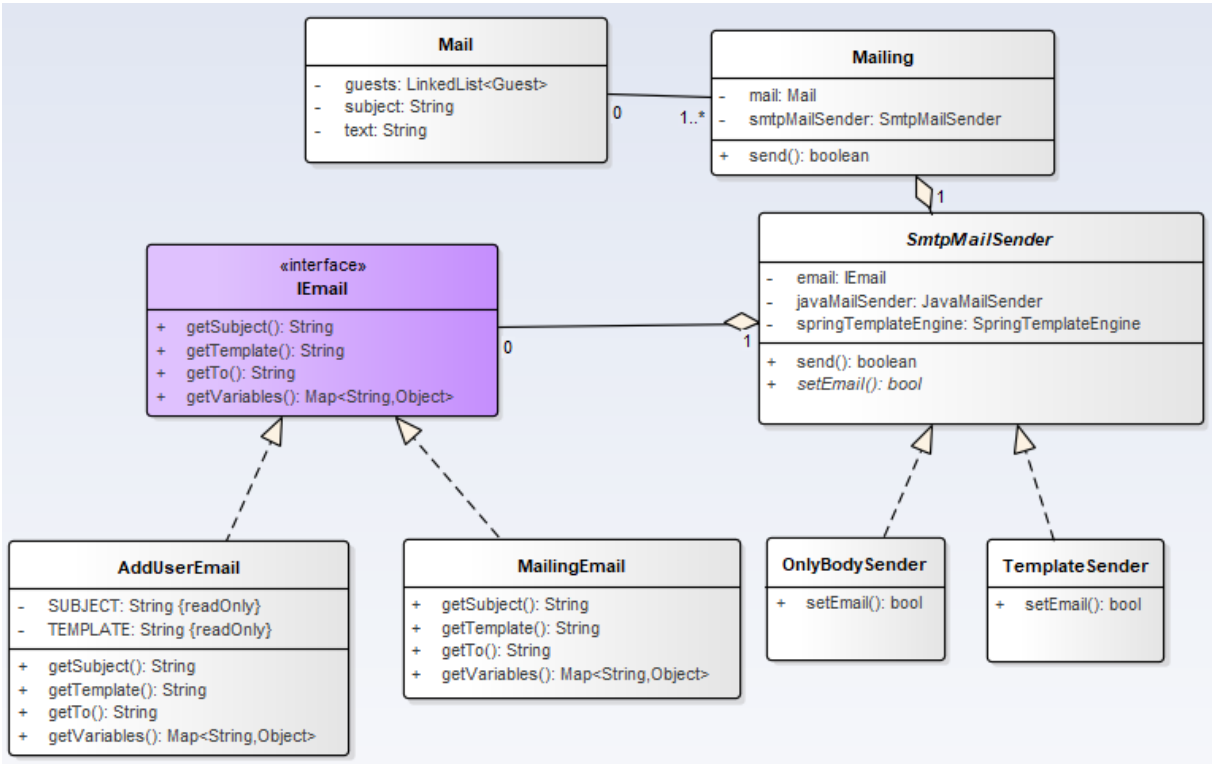
V aplikaci existují dva druhy emailu. První druh je informování, informuje o změně hesla uživatele. Druhým typem je mailing.

Pro generování emailu je využita vlastnost generování emailu podle HTML šablony. Pro převod mezi textovým řezězcem a HTML se používá template engine Springu, představený v kapitole Spring MVC.

Diagram tříd emailu je vysvětlen níže a je doprovázený zdrojovým kódem č. 33.

Základem celého návrhu tříd na posílání emailu je rozhraní IEmail, které definuje, které atributy bude mailer pro odeslání emailu požadovat. Abstraktní třída StmpMailerSender obsahuje potřebné třídy pro zaslání pošty a metodu setEmail. Tato metoda je abstraktní z důvodu, že email v případě mailingu nemá pevně definovanou šablonu, a proto je třeba rozdělit implementace metody setEmail. OnlyBodySender vloží jako šablonu text emailu, zatímco TemplateSender pouze doplní parametry do šablony emailu.

Jelikož v mailingu se obvykle posílá email více hostům, tak byla vytvořena třída mailing, která obsahuje metodu send. Ta v iterátoru volá postupně send nad všemi hosty v dané skupině hostů.



Zdrojový kód 24 – Diagram tříd mailing

9 Datová část

Kapitola datová část má za cíl seznámit čtenáře s datovou částí aplikace. Úvod se zabývá výběrem DB a následně jsou popsány techniky, které jsou v této práci použity pro lepší manipulaci s daty. V závěru kapitoly jsou popsány některé z procedur, které byli použity při implementaci práce.

9.1 Databáze

Výběr DB byl jednoduchý. Z důvodu velice kladných předchozích zkušeností s PostgreSQL, podporou všech funkcí, které jsou pro vývoj hotelového systému potřebné a způsobem jejího licencování jako open source, byla vybrána právě tato DB.

9.2 PostgreSQL

Celý zbytek kapitoly čerpá z [53] a její podstránek.

PostgreSQL je výkonná open source objektově relační DB. Je multiplatformní a lze ji využívat na všech nejpoužívanějších operačních systémech, jako jsou Linux, UNIX (AIX, BSD, HP-UX, macOS, Solaris), a Windows. Také je vyvíjena více než 30 let. Byla vytvořena profesorem Michaelem Stonebrakerem na UCB. PostgreSQL splňuje veškeré standardní požadavky na moderní DB a její popularita v posledních letech stále vzrůstá. [54]

Další část textu o této DB předpokládá základní znalost SQL a budou zde probrány funkce DB, které budou použity pro implementaci systému. Cílem není vysvětlit syntaxi DDL, DML, DCL, TCL a podobně, ale zdůraznit použití některých technik.

9.2.1 Omezení

Omezení neboli constraints slouží pro omezení množiny dat, která jsou validní pro sloupce tabulek. Pokud je toto pravidlo porušeno, není umožněno vložení nebo úprava dat a je vyhozena příslušná výjimka. Libovolné omezení lze kombinovat.

CHECK

Při použití CHECK je sloupec je omezen libovolnou podmínkou.

NOT NULL

Při použití NOT NULL sloupec nesmí být prázdný.

UNIQUE

Při použití UNIQUE sloupec musí být unikátní. Lze využít i pro kombinaci sloupců.

PRIMARY KEY

PRIMARY KEY je kombinace NOT NULL a UNIQUE. Unikátně identifikuje řádek v tabulce.

FOREIGN KEY

FOREIGN KEY dává DB najevo, že se jedná o odkaz na řádek z jiné tabulky.

Ve zdrojovém kódu č. 25 je uvedena definice cizího klíče v deklaraci tabulky, kdy po klíčovém slovu REFERENCES následuje cílová relace a název sloupce, na který bude v této tabulce odkazováno.

```
CREATE TABLE nabidka (
        nabidka_pk integer PRIMARY KEY,
        uzivatel_pk integer REFERENCES uzivatel(uzivatel_pk),
        cena integer CHECK (cena > 0),
        identifikacni_cislo integer UNIQUE NOT NULL
);
```

Zdrojový kód 25 – Foreign key

9.2.2 Sekvence

Sekvence jsou speciální tabulky v DB. Slouží převážně pro generování unikátních hodnot pro primární klíče. Nejzákladnější a nejpoužívanější využití je, když vkládáme nový seznam do DB a zavolá se nad sekvencí funkce nextval, která vrátí následující hodnotu sekvence.

Zvláštním datovým typem, který není ve všech DB je SERIAL, neboli integer, který se sám inkrementuje. Pokud má sloupec s primárním klíčem nastavený tento datový typ není nutné dodávat hodnotu primárního klíče přes nextval, protože jí nastaví PostgreSql sám. Při vytvoření tabulky s tímto datovým typem je také samozřejmostí, že se vytvoří i příslušná sekvence.

Ve zdrojovém kódu č. 26 je uveden příklad pro definici primárního klíče s datovým typem SERIAL.

```
CREATE TABLE nabidka(
        nabidka_pk SERIAL PRIMARY KEY
);
```

Zdrojový kód 26 – Použití Serial

9.2.3 Indexy

PostgreSQL podporuje více druhů indexů, tj. B-tree, Hash, GiST, SP-GiST, Gin a BRIN. Indexy slouží pro optimalizaci DB, převážně pro urychlení hledání menší skupiny záznamů. Pro účely této práce jsou používány pouze B-tree indexy, které PostgreSQL vytváří defaultně.

Ve zdrojovém kódu č. 27 je uveden příklad definice B-tree indexu na sloupci nabidka_pk v tabulce nabidka.

```
CREATE INDEX nabidka_index ON nabidka(nabidka_pk);
```

Zdrojový kód 27 – Vytvoření indexu

9.2.4 PL/pgSQL

PL/pgSQL je procedurální jazyk PostgreSQL DB. Rozšiřuje možnosti DB o procedurální programování a další kontrolu struktury DB. Vývojář s jeho pomocí může vytvářet své funkce, trigerry, typy, operátory atd.

V níže uvedeném zdrojovém kódu je zobrazena ukázka procedury na nastavení výchozí hodnoty daně u všech položek v systému. Nejprve se za pomoci perform a FOUND zjistí, jestli existuje daň se zadaným primárním klíčem a pokud ano, tak se u všech daní nastaví výchozí hodnota. Následně se u požadované daně nastaví výchozí hodnota na true.

```
create or replace function set_actual(in a_vat_pk integer)
  returns void
language plpgsql
as $$
BEGIN
  perform 1 from vat where vat_pk = a_vat_pk;
  IF NOT FOUND THEN
        return;
  END IF;
  update vat set is_default = false;
  update vat set is_default = true where vat_pk = a_vat_pk;
END;
$$;
```

Zdrojový kód 28 – Procedura

Ve zdrojovém kódu číslo 29 je uvedena zajímavá procedura aplikace, která má za úkol nastavit ceny za jednotlivé datumy podle zadaných cen po dnech. Nejprve se zjistí, jestli existuje ceník se zadaným primárním klíčem. V případě že existuje, jedná se o vložení nového ceníku. V opačném případě jde o úpravu stávajícího. Validace zadaných dat probíhá na aplikační vrstvě.

Při tvorbě nového ceníku je nejprve zjištěna hodnota primárního klíče nového záznamu, aby byl znám primární klíč budoucího záznamu v tabulce ceník. Následně je pomocí funkce generate_seriase vytvořen interval po dnech mezi zadanými datumy a podle operátoru case je vložena cena k tomuto dni.

V případě úpravy záznamu je postup podobný s rozdílem, že ceník je upraven a poté jsou smazány všechny ceny, které jsou následně vytvořeny znovu.

```
create or replace function save_price_list(a_price_list_pk integer,
a_date_from date, a_date_to date, a_vat_pk integer, a_mon numeric, a_tue
numeric, a_wed numeric, a_tru numeric, a_fri numeric, a_sat numeric, a_sun
numeric, a_category_pk integer)
  returns boolean
language plpgsql
as $$
DECLARE
  li_price_list_pk integer ;
  li_category_pk integer;
BEGIN

  perform 1 from price_list where price_list_pk = a_price_list_pk;
  IF NOT FOUND THEN
    ;
        li_price_list_pk := nextval('price_list_price_list_pk_seq');

        insert into price_list(price_list_pk,date_from, date_to, vat_pk,
          mon, tue, wed, thu, fri, sat, sun,category_pk)
values(li_price_list_pk,a_date_from,a_date_to,a_vat_pk,a_mon,a_tue,a_wed,a_
tru,a_fri,a_sat,a_sun,a_category_pk);
```

```
        insert into price(price_list_pk,date,price)
          SELECT li_price_list_pk,date_trunc('day', dd):: date,
            CASE EXTRACT(DOW FROM date_trunc('day', dd):: date)
                  when 0 then a_sun
                  when 1 then a_mon
                  when 2 then a_tue
                  when 3 then a_wed
                  when 4 then a_tru
                  when 5 then a_fri
                  when 6 then a_sat
            end
          FROM generate_series
          (  a_date_from::timestamp
          , a_date_to::timestamp
          , '1 day'::interval) dd
        ;
    else
        update price_list
          set
            date_from = a_date_from,
            date_to = a_date_to,
            vat_pk = a_vat_pk,
            mon = a_mon,
            tue = a_tue,
            wed = a_wed,
            thu = a_tru,
            fri = a_fri,
            sat = a_sat,
            sun = a_sun
        where price_list_pk = a_price_list_pk;

        delete from price where a_price_list_pk = price_list_pk;

      insert into price(price_list_pk,date,price)
        SELECT a_price_list_pk,date_trunc('day', dd):: date,
          CASE EXTRACT(DOW FROM date_trunc('day', dd):: date)
          when 0 then a_sun
          when 1 then a_mon
          when 2 then a_tue
          when 3 then a_wed
          when 4 then a_tru
          when 5 then a_fri
          when 6 then a_sat
          end
        FROM generate_series
             (  a_date_from::timestamp
            , a_date_to::timestamp
            , '1 day'::interval) dd;
  end if;
  return  true ;
END;
$$;
```

Zdrojový kód 29 – Ceník

10 Ukázka aplikace

Kapitola ukázka aplikace má za cíl seznámit čtenáře s implementovanou aplikací. Struktura této části je vždy nadpis, který uvádí, čeho se bude podkapitola týkat. Pod ním se nachází text, který popisuje hlavní části této problematiky a následně obrázek, který ukazuje, jaký má tato část aplikace vzhled. Vzhledem, že některé operace se v aplikaci opakují jsou zde popsány pouze jednou. Jedná se především o práci s daty v tabulkách.

10.1 Přihlášení

Před samotným přístupem do aplikace je nutné se přihlásit. Není možné odeslat formulář dříve, než uživatel zadá alespoň 5 znaků u uživatelského hesla a emailu. Až po zadání těchto znaků se stává tlačítko pro odeslání aktivním. V případě vyplnění validních údajů je uživatel přihlášen do aplikace. Dostupnost funkcí aplikace určuje role daného uživatele. Nejsou-li data validní, zobrazí se hláška, která klientovi oznámí zadání špatných přihlašovacích údajů.

Obrázek 39 – Přihlášení

10.2 Layout

Hlavní náplň aplikace je práce s interaktivní štaflí, z tohoto důvodu je možné levé menu schovat do podoby, která je na obrázku zobrazena napravo od menu přes příslušnou ikonu umístěnou v levým horním rohu. Zobrazení a přístup k jednotlivým položkám v menu je dán dle role přihlášené osoby, tedy podle toho, zda se jedná o admina či recepční. Admin vidí veškeré položky a recepční pouze správu hostů, rezervací, knihu hostů a reportu pro cizineckou policii.

Obrázek 40 – Layout

10.3 Uživatelé

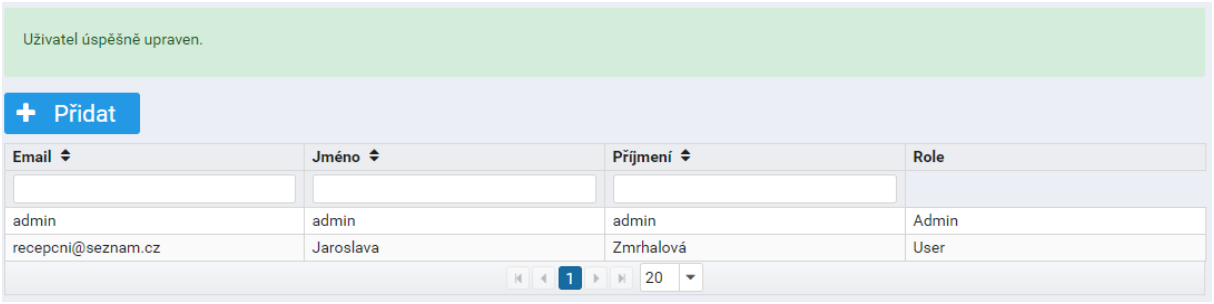
Přes položku v menu uživatelé je dostupná správa uživatelů, kteří mají přístup do aplikace. Po kliknutí na tlačítko „Přidat“ se zobrazí modální okno, kde uživatel vyplní příslušné údaje a odešle formulář. V případě validních údajů je uživatel přidán a zobrazí se hláška „Uživatel úspěšně přidán“, která po krátké době zmizí.

V opačném případě se pod danými vstupy zobrazí hláška, co a proč bylo špatně vyplněno. V případě potřeby záznam editovat nebo smazat je nutné kliknout na řádek v tabulce, kde se zobrazí stejné modální okno s rozdílem, že místo přidávání slouží k úpravě a je také možné přes toto okno smazat vybraný řádek. Po těchto událostech je samozřejmostí také příslušná hláška.

Data, která jsou v tabulce, je možné seřadit kliknutím na label vzestupně i sestupně. Pod popisem názvu sloupce u tabulky se nacházejí inputy, kterými je možné data filtrovat. Dalším užitečným prvkem je stránkování, kde je možné nastavit hodnoty 10,20 a 50. Tento postup je totožný u většiny ostatních správ, proto nebudou nadále rozebírány a ukázka modálního okna a editace budou zobrazeny na ostatních modelech.

Po přidání uživatele je odeslán na jeho email odkaz pro zadání hesla. Nové heslo zadává dvakrát pro potvrzení údajů. Recepční nebo admin si heslo může také změnit. Před vytvořením hesla není uživateli umožněn přístup do aplikace. Veškerá data jsou také defaultně řazena dle vhodného atributu.

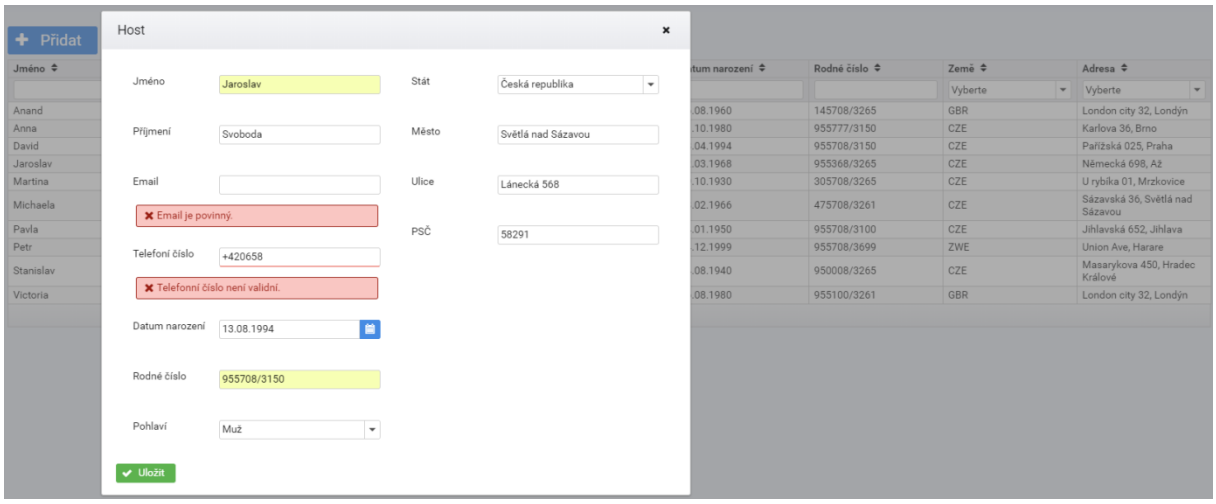
Email musí být vždy ve správném formátu, kromě emailu od uživatele admin, který je vložen do DB před prvním spuštěním používání aplikace.



Obrázek 41 – Správa uživatelů

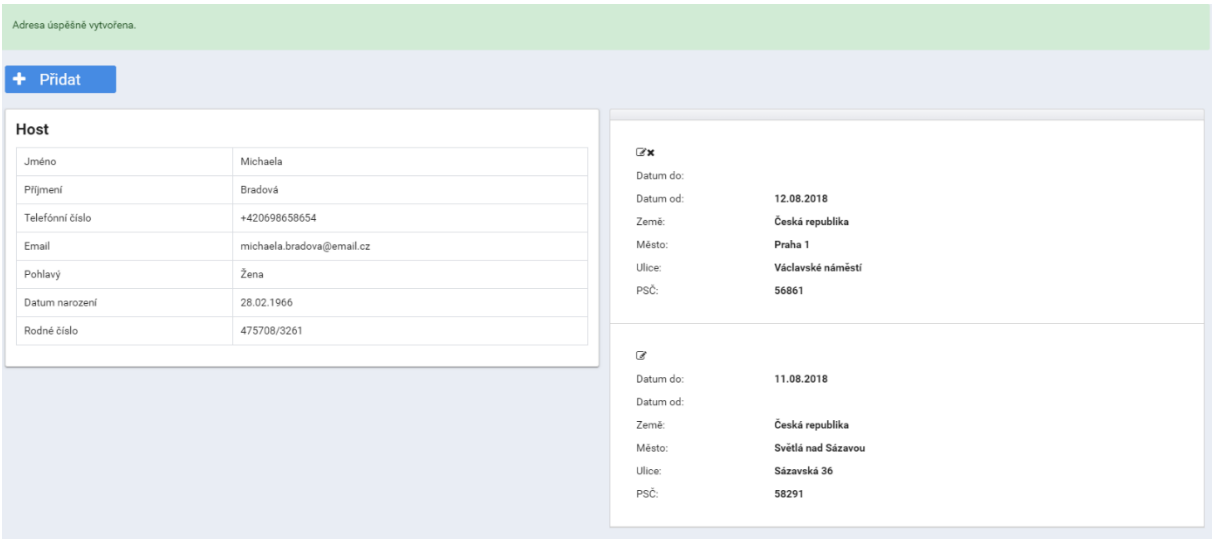
10.4 Hosté

Správa hostů probíhá stejným způsobem jako správa uživatelů. Zásadním rozdílem je možnost změny adresy v rámci hosta v případě jeho opětovné návštěvy ubytovacího zařízení, kdy došlo ke změně adresy. Detail hosta je dostupný po kliknutí na hosta v tabulce a tlačítko „detail“. Velkým rozdílem oproti původnímu funkčnímu požadavku na správu hosta je absence kategorie hosta, která nemá žádný význam z důvodu, že se vždy platí za celý pokoj ne za jednotlivé osoby na pokoji.



Obrázek 42 – Správa hostů

Detail hosta zobrazuje navíc všechny adresy hosta a je zde také jejich správa. Při vytvoření hosta se nastavuje jeho adresa, kdy platnost adresy je vždy prázdná. V případě dalšího přidávání adres se vždy nastaví datum do u předchozí adresy na den zpět od datumu nastaveného v přidané adrese. Smazat je možné pouze poslední adresu u hosta a editovat pouze údaje adresy nikoliv rozsah platnosti. První adresa má tudíž datum od vždy prázdný.



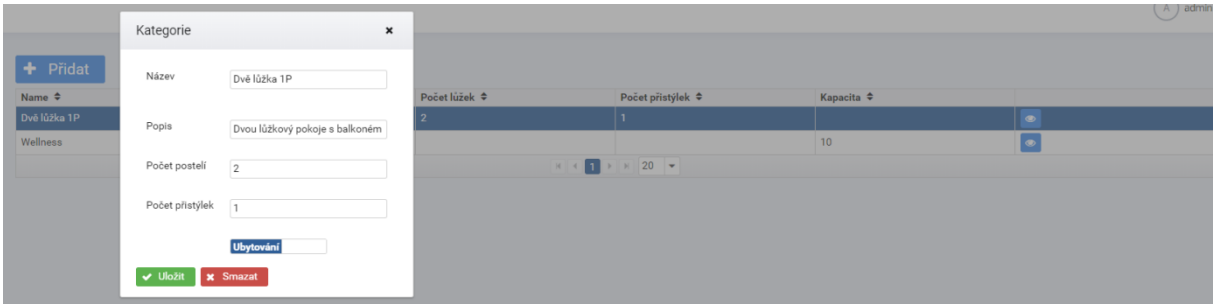
Obrázek 43 – Správa hosta

10.5 Ubytování

Základem tvorby ubytovacích jednotek hotelu jsou kategorie. Ty jsou dvojího typu. První typ je přímo ubytovací. Ten slouží pro definici pokojů podle počtu lůžek, vybavení atd. Druhý typ je „ostatní“ a slouží pro všechny ostatní objekty jako jsou například sportoviště, wellness a podobně.

Ubytovací kategorie slouží pro hlavní rezervace, ze kterých jsou brány údaje jako jsou právě ubytovaní hosté a podobně. Jejich rezervace probíhá po celých dnech. Na rozdíl od druhé kategorie, kde je tento interval minuta.

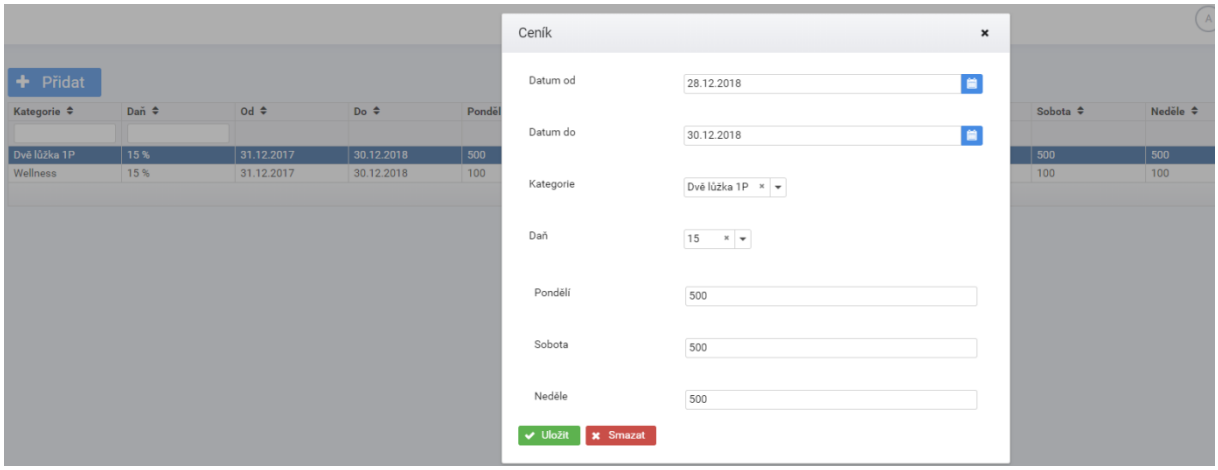
V případě rezervace hosta na ostatní kategorii je rezervace zaúčtována na hlavní rezervaci, kde je příslušný host ubytován. Po vytvoření kategorie je možné zobrazit její detail, kde se vytvářejí jednotlivé objekty kategorie.



Obrázek 44 – Ubytování

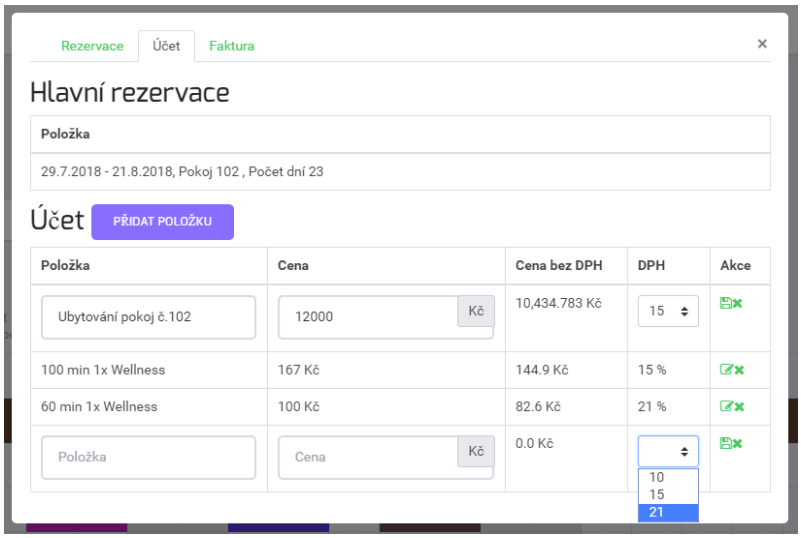
10.6 Ceník

Ceník se nastavuje podle intervalu. Intervaly se nesmí překrývat. K ceníku je nutné napárovat výši daně, která se bude uplatňovat. Pokud je kategorie typu pokoj, cena je uvedena za jeden den. V opačném případě je cena uvedena za hodinu pobytu jedné osoby v zařízení.



Obrázek 45 – Ceník

Veškeré tyto ceny jsou pouze výchozí a dají se v rezervaci pozměnit. Při vytvoření jakékoliv rezervace je vždy na účet hlavní rezervace přidána položka účtu, která obsahuje cenu, daň a název. Ta už nemá žádnou vazbu na ceník, kategorii, daň a podobně. Položku je možné, jakkoliv upravit. Po změně nastavení jakéhokoliv ceníků pokoje, k již vytvořené rezervaci tedy nenastane žádná změna v položkách účtu rezervace. Také je možné spravovat položky na účtu.



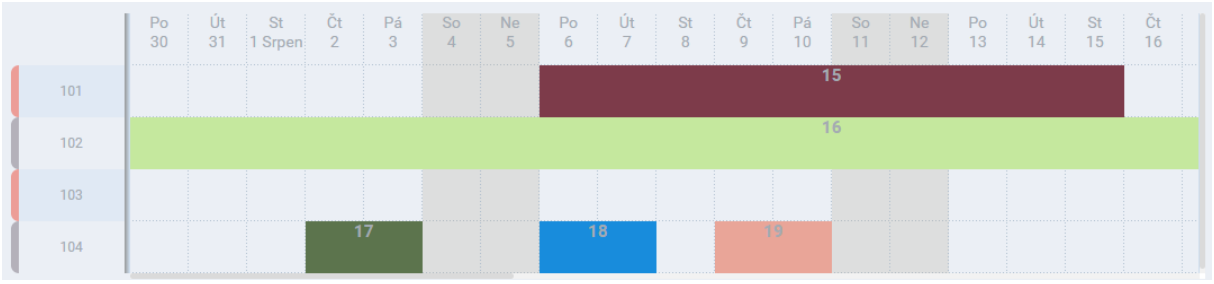
Obrázek 46 – Účet rezervace

10.7 Rezervace

Jelikož existují dva typy kategorií, kde se každá rezervuje po jiných intervalech, existují dva druhy kalendářů.

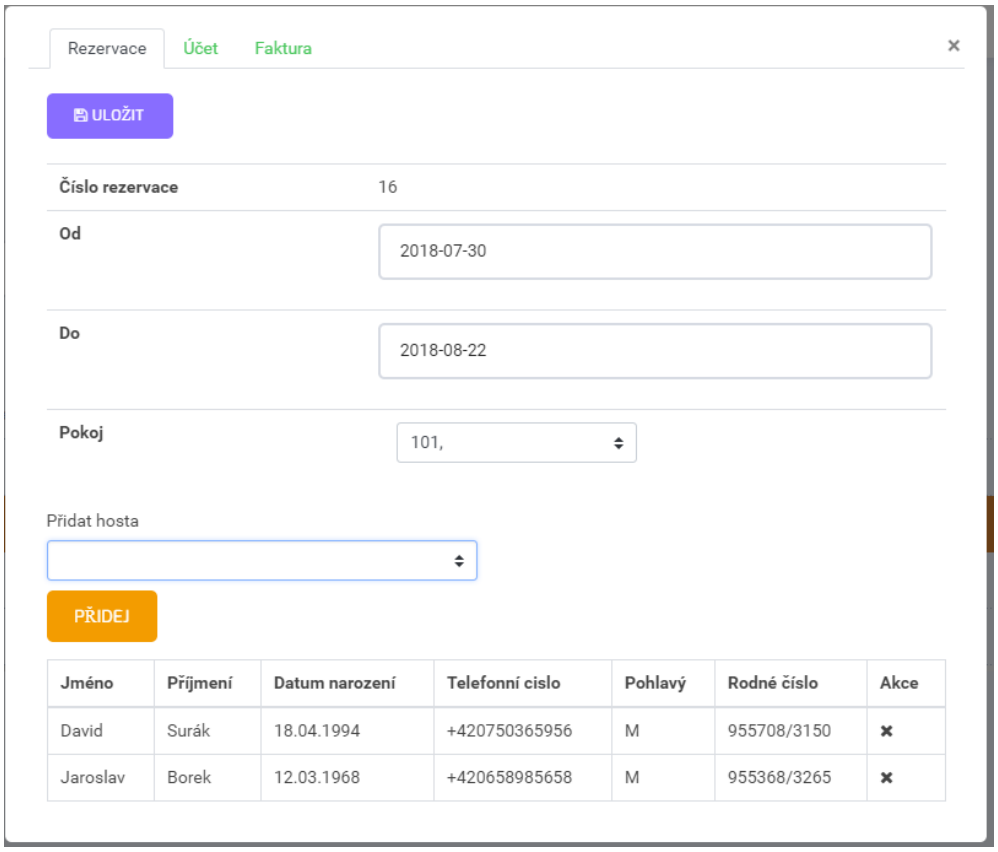
Ubytování

V tomto případě je kalendář zobrazen po jednotlivých dnech závisle na jednotlivých pokojích. Základní vlastností je také nastavené počátečního a koncového datumu pro zobrazování rezervací.



Obrázek 47 – Rezervace ubytování

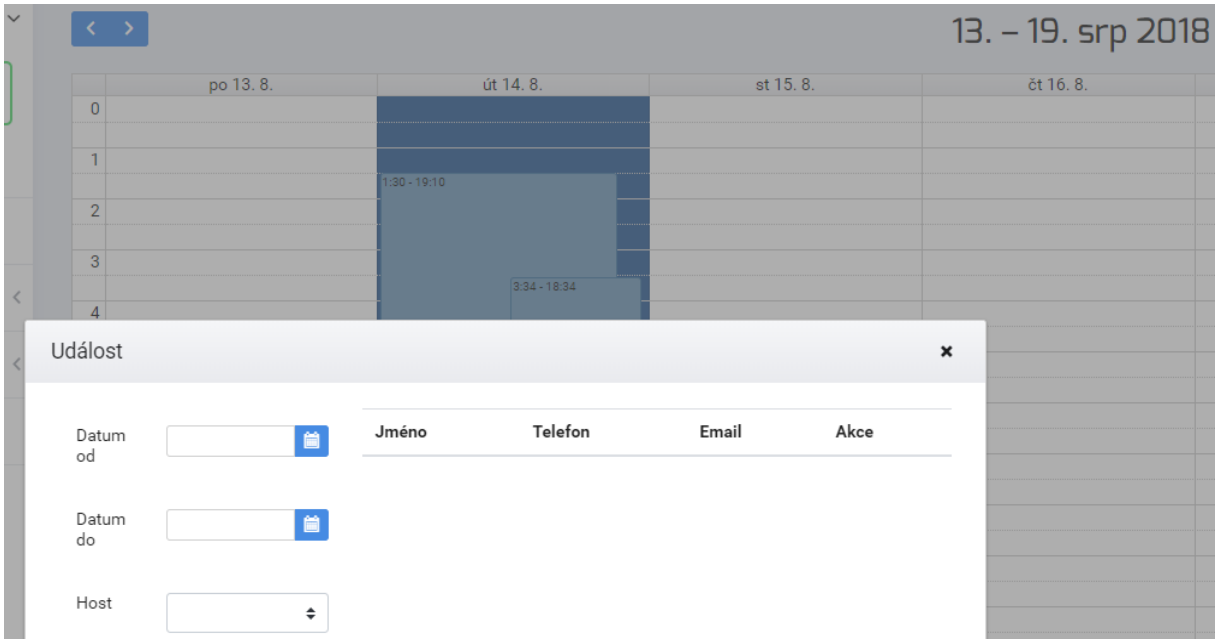
Veškeré rezervace je možné rozkliknout a nadále spravovat. Je možné editovat jejich účet, přidáváním, úpravou, mazáním položek. Je možné vytvoření faktury i s EET. V položce faktura se nastavují fakturační údaje, které je možné vyplnit nebo a načíst z hostů z rezervace, tj. aktuální adresa hostů v dané rezervaci. V případě, že byla faktura již vytvořena, tak je možné ji zde stáhnout.



Obrázek 48 – Správa rezervace

Ostatní

Rezervace ostatních objektů probíhá po minutách. Před samotným zobrazením rezervace je nutné zvolit, jakému objektu chce recepční spravovat rezervace Velikým rozdílem oproti ubytovacím rezervacím je možnost překrytí jednotlivých rezervací v jednom objektu.

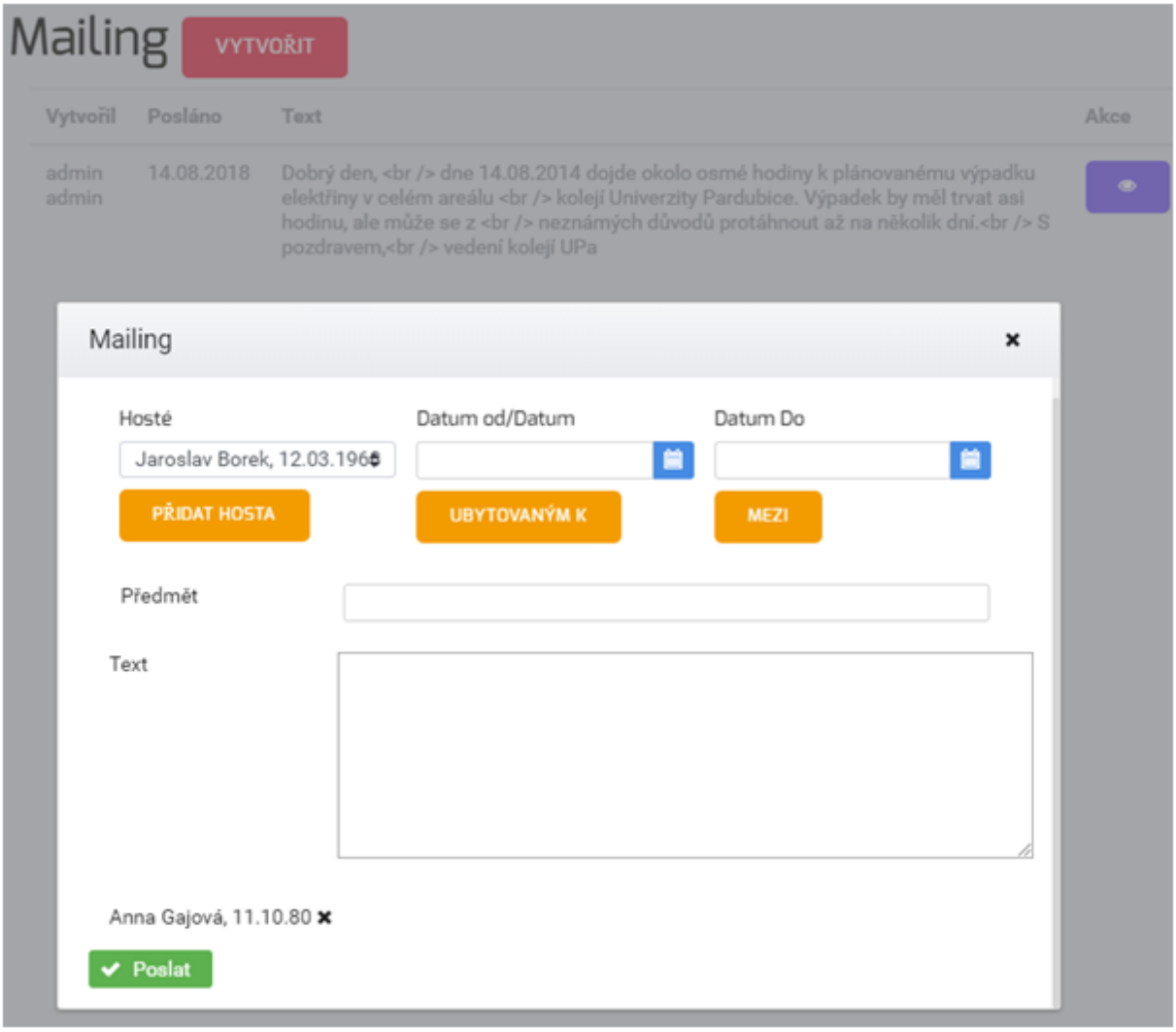


Obrázek 49 – Rezervace wellnes

10.8 Mailing

Aplikace umožňuje hromadné odeslání emailů hostům. Přidávání hostů je možné třemi způsoby. Po jednotlivcích hostech, všech hostech, kteří jsou ubytováni v určitém datu a všechny hosty, kteří jsou ubytováni v zadaném časovém intervalu.

Potencionální příjemci emailu jsou cyklicky vypisováni pod textem emailu, kde je možné je následně odebrat.



Obrázek 50 – Mailing

Závěr

Teoretická i praktická část práce byla splněna v plném rozsahu.

První kapitola se zabývá problematikou informačních systémů. Postupně jsou zde vysvětleny veškeré základní pojmy jako jsou data, informace, systém atd. Následně je v této kapitole definován informační systém a dle požadavků diplomové práce také jeho rozbor, kategorizace a členění jednotlivých typů.

Na první kapitolu navazuje část, kde jsou vysvětleny informační systémy v oblasti hotelnictví včetně vysvětlení funkce distribuovaných systémů a jejich typů, bez kterých se žádný dnešní hotel neobejde.

Před samotnou tvorbou požadavků bylo potřeba nejdříve zjistit, jak to chodí v hotelovém provozu a také, jaké požadavky jsou dány ze zákona. Tyto poznatky shromažďuje kapitola číslo 3.

Posledním požadavkem na kvalifikační práci v teoretické části je představení jednotlivých technologií a zdůvodnění volby těchto technologií. Tento bod zadání je zpracováván postupně v celé práci od kapitoly číslo 4, kde jsou nejprve rozebrány webové technologie jako takové a následně popsány možné architektury a jejich výhody a nevýhody. Konkrétní výběr technologií i se zdůvodněním je popsán v posledních kapitolách.

Velkou výhodou při implementaci aplikace byly funkční požadavky, datový model a případy užití. I po jejich vytvoření byly ještě několikrát upravovány z důvodu dalších návrhů k zamýšlení nad danou problematikou.

Volba technologií se ukázala také jako správná. Zvolení architektury SPA zjednodušilo tvorbu rozhraní, kde se implicitně vyřešily problémy s napojením na externí rezervační a restaurační systémy. Dokonce v jedné fázi vývoje byla snaha o napojení na externí channel manager. Pokus však ztroskotal z důvodu neposkytnutí testovacího prostředí.

Systém také splňuje dle zadání napojení na EET a přímé generování faktur. Zde je nutné podotknout, že není možné platit kartou, a to z důvodu, že nebylo možné zajistit platební terminál na vyzkoušení.

Rezervace je možné vytvořit jak na ubytovací buňky, tak na wellness. Systém nabízí vytváření i jiných „objektů“ než je wellness. Teoreticky je možné přes systém vytvořit i rezervace ke stolům v restauraci, sportoviště atd.

Aplikace obsahuje veškeré základní funkce, které jsou pro kategorie těchto software obvyklé, ale problematika hotelových systémů je rozsáhlejší a vždy se najde nějaká funkce, kterou by se hodilo přidat.

Literatura

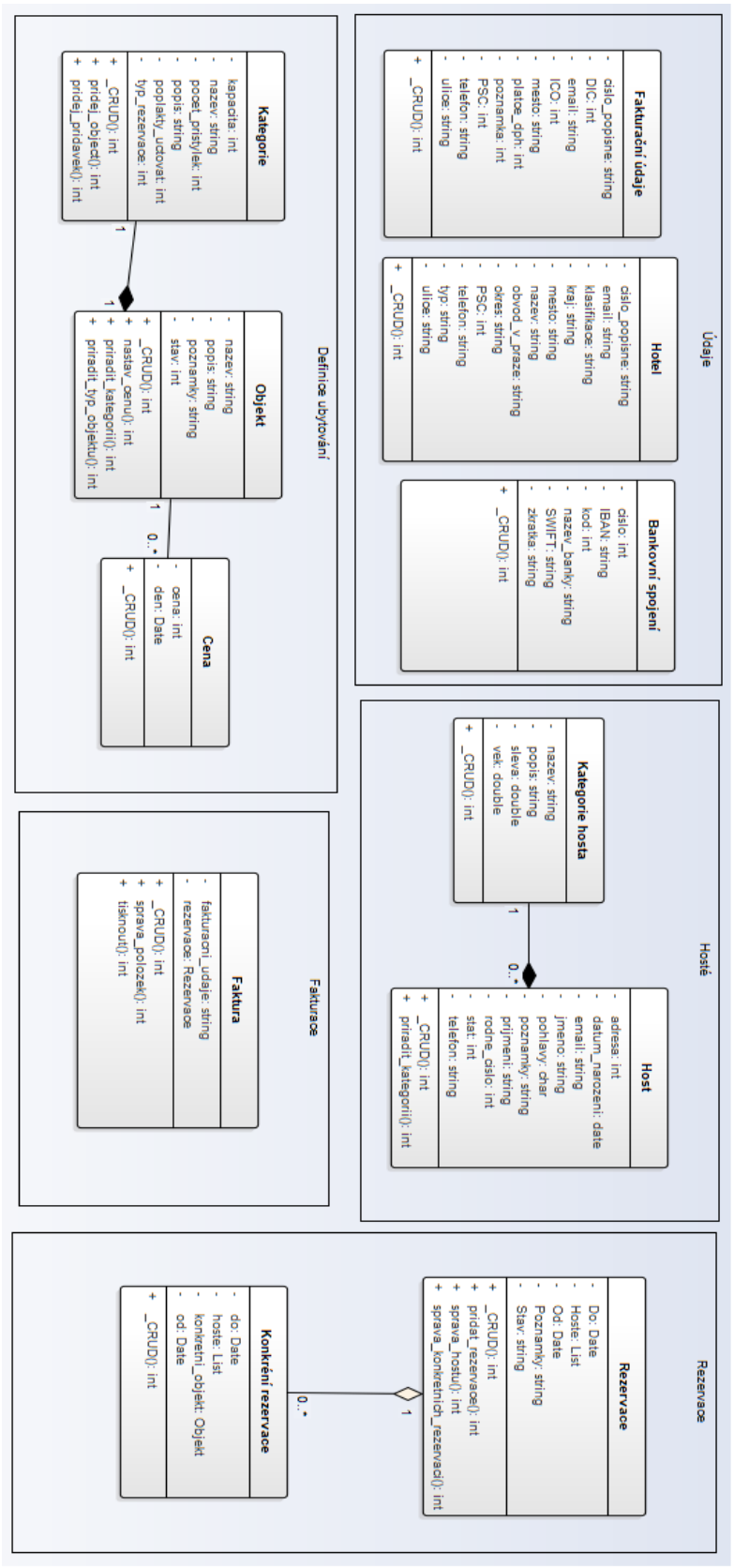
1. VYMĚTAL, Dominik. Informační systémy v podnicích: teorie a praxe projektování. Praha: Grada, 2009. ISBN 978-80-247-3046-2.
2. STAIR, Ralph M.; REYNOLDS, George W. *Principles of Information Systems: A Managerial Approach*. 9. vydání. Boston: 2010 [cit. 22. 4. 2018] ISBN-13:978–0-324–66528–4. Dostupné z: https://drive.uqu.edu.sa/_/fbshareef/files/principles%20of%20information%20systems%209th%20-stair,%20reynolds.pdf
3. MOLNÁR, Zdeněk. *Podnikové informační systémy.* 2. přeprac. vydání. Praha: České vysoké učení technické, 2009. ISBN 978-80-01-04380-6.
4. Pojem informačního systému. *Masarykova univerzita* [online]. Brno [cit. 22. 4. 2018]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsys.htm
5. BOURGEOIS, David T. *Information Systems for Business and Beyond.* [online]. Lulu.com, 2014. [cit. 22. 4. 2018]. ISBN 978-13-04-94348-4. Dostupné z: https://www.saylor.org/site/textbooks/Information%20Systems%20for%20Business%20and%20Beyond.pdf
6. Ibm704. In: *Geek.com* [online]. Ziff Davis, LLC. PCMag Digital Group 2010. [cit. 9. 8. 2018]. Dostupné z: https://www.geek.com/chips/mainframes-making-a-comeback-1055082/
7. Altair 8800. In: *Wikipedia* [online]. Wikimedia Foundation, Inc. ©2018. [cit. 9. 8. 2018]. Dostupné z: https://en.wikipedia.org/wiki/Altair_8800#/media/File:Altair_8800_Computer.jpgChromeHTML\Shell\Open\Command
8. MOLNÁR, Zdeněk. *Podnikové informační systémy* [přednáška]. Praha: ČVUT, Ústav řízení a ekonomiky podniku.
9. MCNAMEE, Joe et al. Jak funguje internet [online]. Praha: Iuridicum Remedium, [2014] [cit. 13. 08. 2018]. Dostupné z: http://www.slidilove.cz/sites/default/files/edripaper03_howtheinternetworks_cz.pdf
10. Course Hero, Inc. coursehero.com. *Because of the complex ity of contemporary* [online]. [cit. 13. 8. 2018]. Dostupné z: https://www.coursehero.com/file/p5gstqv/Because-of-the-complex-ity-of-contemporary-information-systems-and-rapidly/
11. SODOMKA, Petr a KLČOVÁ, Hana. *Informační systémy v podnikové praxi.* 2., aktualiz. a rozš. vyd. Brno: Computer Press, 2010. ISBN 978-80-251-2878-7.
12. Roman DANEL. *Informační systémy ECM, EAM, HRM, SCM* [online]. Ostrava, Institut ekonomiky a systémů řízení, Hornicko–geologická fakulta.
13. KŘÍŽEK, Felix; NEUFUS, Josef. *Moderní hotelový management.* 2. rozšířené vydání. Praha: Grada, 2014. ISBN: 978-80-247-4835-1.
14. iHOTELLIGENCE. *Hotel Management Systems* [prezentace]. Innovation Software

15. Slovník pojmů. *petragryczova.cz* [online]. [cit. 14. 8. 2018]. Dostupné z: https://www.petragryczova.cz/slovnik-pojmu/
16. Vyhláška č. 501/2006 Sb., o obecných požadavcích na využívání území (stavební zákon). In: *Sbírka zákonů*. 1. 1. 2013. ISSN
17. Jiří BUCHVALDEK. Povinnosti poskytovatele krátkodobého ubytování. *Hrubý & Buchvaldek advokátní kancelář* [online]. Praha: 2016. [cit. 22. 4. 2018]. Dostupné z: http://hblaw.eu/cz/aktuality/73-ubytovaci-sluzby.html
18. Vojtěch Fiala. Právní povaha krátkodobého ubytování. EPRÁVO.cz [online]. Praha: 2016 [cit. 22. 4. 2018]. Dostupné z: https://www.epravo.cz/top/clanky/pravni-povaha-kratkodobeho-ubytovani-102612.htmlChromeHTML/Shell/Open/Command
19. KOSEK, Jiří. *HTML: tvorba dokonalých WWW stránek: Podrobný průvodce*. Praha: Grada, 1998. ISBN 80-716-9608-0.
20. Tutorialspoint. *ES6*. 2016. [cit. 9. 8. 2018]. Dostupné z: https://www.tutorialspoint.com/es6/es6_tutorial.pdf
21. Tutorialspoint. HTTP Tutorial [online]. ©2018 [cit. 9. 8. 2018]. Dostupné z: https://www.tutorialspoint.com/http/index.htm
22. KOSEK, Jiří. *Protokol HTTP* [přednáška].
23. Jiří Kosek. Základy protokolu http. k*osek.cz* [online]. ©1999 [cit. 9. 8. 2018]. Dostupné z: https://www.kosek.cz/clanky/iweb/05.html
24. w3schools.com. Introduction to XML. *w3schools.com* [online]. ©2018 [cit. 9. 8. 2018]. Dostupné z: https://www.w3schools.com/xml/xml_whatis.asp
25. Internet Info, s.r.o. Nové standardy pro JSON. *ROOT.cz* [online]. 2013.©2018 [cit. 9. 8. 2018]. Dostupné z: https://www.root.cz/clanky/nove-standardy-pro-json/
26. Jiří Kosek. Využití webových služeb a protokolu SOAP při komunikaci. *kosek.cz* [online]. 2018 [cit. . 8. 2018]. Dostupné z: www.kosek.cz/diplomka/html/websluzby.html
27. VŠB-TU Ostrava. RESTful API. *Katedra informatiky* [online]. Ostrava. [cit. 12. 8. 2018]. Dostupné z: http://wiki.cs.vsb.cz/images/f/fd/TAMZ-cv-11-CZ.pdf
28. Guru99. guru99.com. *SOAP Vs. REST: Difference between Web API Services* [online]. ©2018 [cit. 12. 8. 2018]. Dostupné z: https://www.guru99.com/comparison-between-web-services.html
29. GitHub, Inc. github.com. *REST vs. SOAP* [online]. [cit. 12. 8. 2018]. Dostupné z: https://github.com/honzajavorek/jakpsatapi/blob/master/rest-soap.md
30. itnetwork.cz. ITnetwork.cz. *Lekce 1 - Úvod do PHP a webových aplikací* [online]. ©2018 [cit. 12. 8. 2018]. Dostupné z: https://www.itnetwork.cz/php/zaklady/php-tutorial-uvod-do-webovych-aplikaci
31. GitHub, Inc. github.com. *23. webové aplikace* [online]. 31.5. 2015. Aktualizováno 14. 6. 2015 [cit. 12.8. 2018]. Dostupné z: https://github.com/ludekvesely/szz-2015/wiki/23.-webov%C3%A9-aplikace

32. Medium. *Single-page application vs. multiple-page application* [online]. 1. 12. 2016 [cit. 12. 8. 2018]. Dostupné z: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
33. Medium. *Single-page application vs. multiple-page application* [online]. 1. 12. 2016 [cit. 12. 8. 2018]. Dostupné z: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58
34. Microsoft. *Volba mezi tradičními webovými aplikacemi a jednostránkové aplikace (SPA)* [online]. 28. 06. 2018. [cit. 12. 8. 2018]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/standard/modern-web-apps-azure-architecture/choose-between-traditional-web-and-single-page-apps
35. C# Corner. c-sharpcorner.com. *Single Page Application (SPA) Using AngularJS, Web API and MVC 5* [online]. 15. 12. 2014. [cit. 12. 8. 2018]. Dostupné z: https://www.c-sharpcorner.com/uploadfile/rahul4_saxena/single-page-application-spa-using-angularjs-web-api-and-m/
36. The core Less team. lesscss.org. *Overview* [online]. [cit. 12. 8. 2018]. Dostupné z: http://lesscss.org/
37. GitHub, Inc. github.com. *lesscss.org* [online]. [cit. 12.8. 2018]. Dostupné z: https://github.com/less/less-docs
38. tutorialspoint.com. *Bootstrap* [online]. ©2014 [cit. 12. 8. 2018]. Dostupné z: https://www.tutorialspoint.com/bootstrap/bootstrap_tutorial.pdf
39. tutorialspoint.com. *TypeScript - Overview* [online]. [cit. 12. 8. 2018]. Dostupné z: https://www.tutorialspoint.com/typescript/typescript_overview.htm
40. angular.io. *docs* [online].©2018 [cit. 12. 8. 2018]. Dostupné z: https://angular.io/
41. Oracle. docs.oracle.com. *About the Java Technology* [online]. ©2017. [cit. 12. 8. 2018]. Dostupné z: https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html
42. PrimeNG. *primefaces.org* [online]. [cit. 16.8. 2018]. Dostupné z: https://www.primefaces.org/primeng/#/
43. BALÍK, Miroslav. *Programování v jazyku Java* [přednáška]. Praha: České vysoké učení technické, Fakulta informačních technologií
44. Apache Software Foundation. *Apache Tomcat* [online]. ©2018 [cit. 12. 8. 2018]. Dostupné z: http://tomcat.apache.org/
45. COSMINA, Iuliana; HARROP, Rob; SCHAEFER, Chris; HO, Clarence. *Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools*. New York: Springer Science+Business Media, 2017. 5. Edice. ISBN: 1-4842-2807-3
46. docs.spring.io. *Spring Framework Documentation* [online]. Aktualizováno 26. 7. 2018 [cit. 13. 8. 2018]. Dostupné z: https://docs.spring.io/spring/docs/current/spring-framework-reference/index.html

47. docs.spring.io. *Overview of Spring Framework* [online]. Aktualizováno 24. 7. 2017 [cit. 13. 8. 2018]. Dostupné z: https://docs.spring.io/spring/docs/5.0.0.RC3/spring-framework-reference/overview.html#overview-getting-started-with-spring
48. Masarykova Univerzita. kore.fi.muni.cz. *Spring JDBC* [online]. Brno. Aktualizováno 28. 4. 2013 [cit. 13. 8. 2018]. Dostupné z: https://kore.fi.muni.cz/wiki/index.php?title=Spring_JDBC
49. baeldung.com. *Spring MVC Custom Validation* [online]. Aktualizováno 4. 4. 2018 [cit. 13. 8. 2018]. Dostupné z: https://www.baeldung.com/spring-mvc-custom-validator
50. tcBUT. In: imgur [online]. [cit. 13. 8. 2018] Dostupné z: https://i.stack.imgur.com/tcBUT.png
51. Testdrive AngularJS with Spring 4. In: slideshare.net [online]. LinkedIn Corporation 4. 10. 2015. [cit. 15. 8. 2018]. Dostupné z*: https://www.slideshare.net/owahlen/angularjs-spring4*
52. CÍGL, Jakub. Implementace EET pro e-shopy. České Budějovice, 2017. Diplomová práce. Jihočeská univerzita v Českých Budějovicích. Ekonomická fakulta. Katedra aplikované matematiky a informatiky.
53. The PostgreSQL Global Development Group. postgresql.org. *PostgreSQL 10.5 Documentation* [online]. [cit. 13.8. 2018]. Dostupné z: https://www.postgresql.org/docs/10/static/index.html
54. About. *POSTGRESQL* [online]. 2018 [cit. 22. 4. 2018]. Dostupné z: https://www.postgresql.org/about/

Příloha A – Analytický model



Příloha B – Databázový model