UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Uživatelská nadstavba pro kontrolu procesů skladového hospodářství pivovaru

Lukáš Semorád

Bakalářská práce
2020

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Lukáš Semorád**
Osobní číslo:	**I16140**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Uživatelská nadstavba pro kontrolu procesů skladového hospodářství pivovaru**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Cílem bakalářské práce je analýza stávajících komponent, návrh a vytvoření funkčního modulu pro efektivní evidenci a přehled skladových pohybů v minipivovaru. Klíčovým požadavkem je přehledné grafické zobrazení nádob a stavů a jednoduchá obsluha přímo v provozu. Modul bude navázán na skladové hospodářství a restaurační systém klienta pro evidenci spotřeby surovin při výrobě a naskladnění výsledného produktu.

Modul bude umožňovat i další evidenční funkce, jako například tisk dat, evidenci mytí nádob, výpočet spotřební daně a generování příslušných dokumentů pro celní úřad.
V úvodní části je nutno provést rešerši potřeb uživatelů a současných SW komponent klienta. Součástí teoretické části bude také rešerše systémů zabývajících se podobnou problematikou.

Praktická část bude obsahovat popis použitých technologií při realizaci modulu, včetně návrhu databáze. Pro realizaci budou využity převážně jazyky JavaScript, PHP a HTML5.

Rozsah pracovní zprávy: **30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

BASL, Josef a Roman BLAŽÍČEK. *Podnikové informační systémy: podnik v informační společnosti.* 3., aktualiz. a dopl. vyd. Praha: Grada, 2012. Management v informační společnosti. ISBN 978-80-247-4307-3.

EELES, Peter a Peter CRIPPS. *Architektura softwaru.* Brno: Computer Press, 2011. ISBN 978-80-251-3036-0.

DEELEMAN, Pablo. *Learning Angular 2.* 1. Birmingham: Packt Publishing Limited, 2016. ISBN 1785882074.

Vedoucí bakalářské práce: **Ing. Jiří Lebduška**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. listopadu 2019**
Termín odevzdání bakalářské práce: **7. května 2020**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 17. prosince 2019

Prohlašuji:

Tuto práci jsem 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., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, 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.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 28. 7. 2020

Lukáš Semorád

PODĚKOVÁNÍ

Tímto bych rád poděkoval svému vedoucímu práce Ing. Jiřímu Lebduškovi za cenné rady, odbornou pomoc a ochotu při vedení této bakalářské práce. Veliké poděkování patří mým přátelům a rodině, především rodičům, za trpělivost a podporu během mého studia. V neposlední řadě bych rád poděkoval společnosti T - Solutions, s. r. o., která umožnila zpracování a zveřejnění praktické částí této bakalářské práce.

ANOTACE

Tato bakalářská práce je věnována návrhu a vývoji grafické uživatelské nadstavby pro snadnou obsluhu skladových pohybů minipivovaru. V teoretické části autor rozebírá klíčové požadavky a současné stavy softwarových komponent společnosti. Autor v práci popisuje technologie a nástroje použité při realizaci projektu. V práci je dále prakticky navržen a vytvořen funkční modul pro obsluhu daného minipivovaru.

KLÍČOVÁ SLOVA

Uživatelské prostředí, pivovar, GUI, intranet, modul, TypeScript

TITLE

User extension for control of brewery warehouse management processes

ANNOTATION

This bachelor thesis is dedicated to design and development of graphical user superstructure for easy operation of warehouse movements of a mini-brewery. In the teoretical part, the author analyzes the crucial requirements and current states of software components of the company. The author describes the technologies and tools used in the implementation of the project. In the thesis is also practically designed and created a functional module for operation the mini-brewery.

KEYWORDS

User interface, brewery, GUI, intranet, module, TypeScript

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZKRATEK

PDF	Document Format
HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
REST	Representational state transfer
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
JWT	JSON Web Token
API	Application Programming Interface
NPM	Node Package Manager

ÚVOD

Tato bakalářská práce se zabývá vývojem třívrstvého softwarového řešení se zaměřením na uživatelskou část. Právě na tuto část softwaru je často kladeno klienty mnoho požadavků. To je způsobeno především grafickým uživatelským rozhraním, které bývá nedílnou součástí každého vývoje softwaru určeného pro koncového uživatele. Poté slouží jako komunikační nástroj mezi uživatelem a vytvořeným programem. A právě tímto článkem se bude bakalářská práce primárně zabývat.

V úvodu bakalářské práce je popsán Intranet, který je využíván v provozovnách, pro celkové usnadnění jejich chodu. Celý intranet je složen z jednotlivých částí, které mohou často fungovat nezávisle na sobě. Těmto částem se říká moduly. Konkrétní provozovna, která využívá tento intranet, má ve svém komplexu také minipivovar. Na základě tohoto minipivovaru byl vytvořen požadavek na vytvoření nového modulu, který by zefektivnil jeho evidenci a přehled o skladových pohybech.

Hlavním cílem této bakalářské práce je splnění tohoto požadavku, tedy vytvoření funkčního modulu, který bude pomáhat při obsluze minipivovaru. Na základě daných požadavků je třeba provést analýzu a navrhnout efektivní způsob implementace. Na rozdíl od jiných modulů, bude na tento kladen větší důraz při tvorbě grafického rozhraní. Modul musí být například připraven na možnost obsluhy pomocí tabletu či mobilního telefonu.

Modul pivovaru není jediným, který byl autorem rámci zmíněného intranetu vytvářen, nebo na něm autor spolupracoval. Ovšem na rozdíl od jiných modulů je velmi unikátní, a to právě díky svému grafickému rozhraní. Jiné moduly pro obsluhu provozoven často obsahují převážně tabulky s daty. Možnost zpracování modulu pivovaru je tedy veliká příležitost pro prozkoumání dalších technologií a způsobů vývoje.

Jak již bylo zmíněno, začátek bakalářské práce je věnován Intranetu, do kterého bude modul zasazen. Následuje část, ve které jsou popsány nejvhodněji nástroje pro efektivnější vývoj. Zároveň je zde věnována část aktuálním technologiím, které jsou využívané při tvorbě takových systémů. V další kapitole jsou popsány požadavky a jejich zpracování ve formě návrhu. Poslední částí práce je samotná realizace grafického prostředí a implementace jednotlivých funkcí.

1 ANALÝZA

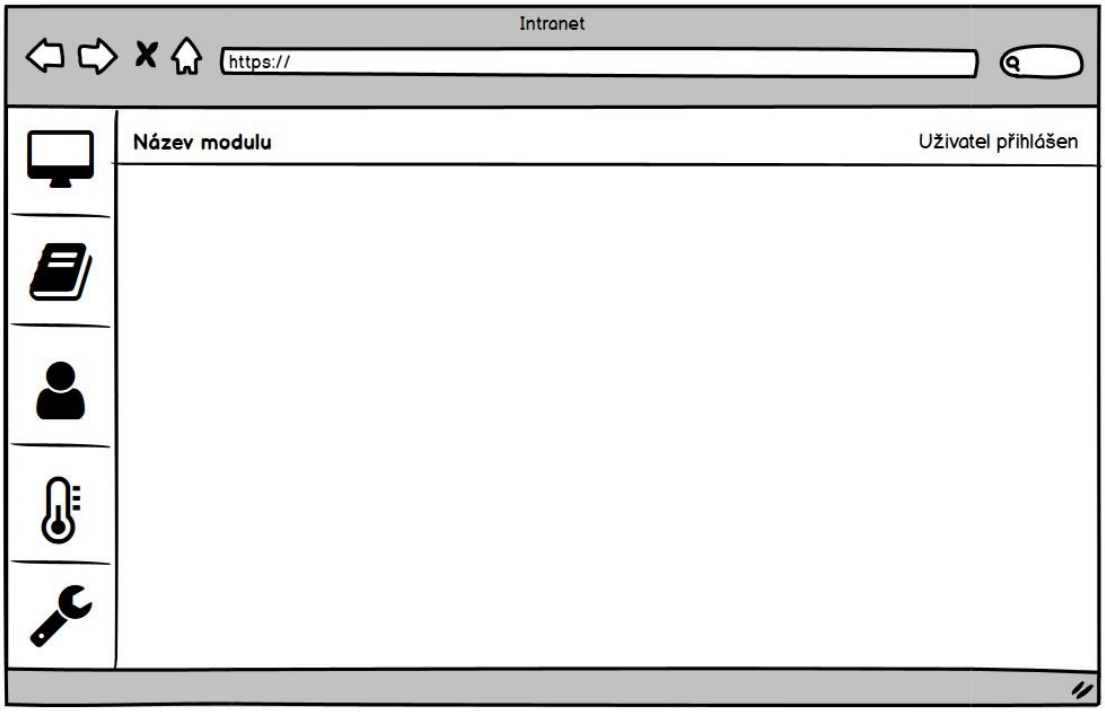
1.1 Hotelový intranet

Vyvíjený hotelový intranet je komplexní softwarový nástroj pro obsluhu a správu provozoven (např. hotelů). Intranet zajišťuje širokou škálu funkcionalit, které jsou pro hotely klíčové. Samozřejmostí je zabezpečené přihlašování s implementovanými rolemi uživatelů, díky kterému je intranet dostupný pouze oprávněným uživatelům a zajišťuje bezpečný přístup k informacím a ovládacím prvkům.

Jedna ze základních myšlenek tohoto hotelového intranetu byla sjednocení uživatelských účtů a integrace systémů třetích stran. Uživatel, který má přístup do intranetu, tedy nepotřebuje jiné aplikace či přístup do jiných systémů. Stačí se přihlásit do systému intranetu a podle oprávnění dostane přístup do určených modulů, se kterými může pracovat. Jednotlivé moduly mohou mít dále návaznosti do aplikací třetích stran, což již z pohledu uživatele není důležité.

1.1.1 Modularita intranetu

Základním požadavkem na architekturu řešení je modularita. Jedná se o to, že hotelový intranet je rozdělen do jednotlivých modulů. Moduly intranetu je poté možné různě kombinovat, dle potřeb zákazníků. Moduly je možné snadno integrovat a instalovat nezávisle v různých provozech. Právě modularita intranetu také zajišťuje snadnější vývoj a rozšiřování o další funkce, které jsou potřeba zajistit. Další předností, která souvisí s modularitou intranetu, je jeho nezávislost na dané lokalitě. Intranet, respektive jeho moduly, jsou navrženy tak, aby nebyly závislé na jednom hotelu, ale dala se snadno vytvořit nová instance pro jinou provozovnu. Tato instance se poté pouze přizpůsobí konkrétním požadavkům zákazníka. Na dále uvedeném obrázku je vidět rozložení prvků intranetu. Na levé straně se nachází lišta implementovaných modulů. Po zvolení modulu se načte do hlavní části obrazovky a je možné s ním dále pracovat. Pro větší pracovní plochu je také možné lištu modulů dočasně skrýt.

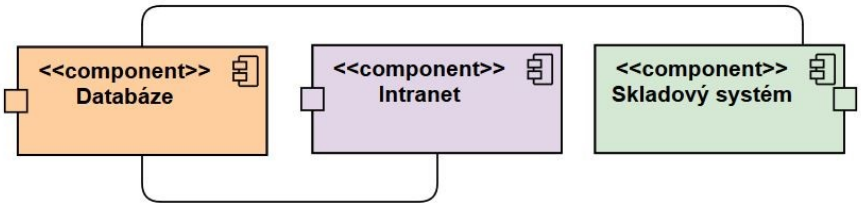


Obrázek 1: Model intranetu. Zdroj: Vlastní

1.1.2 Skladové hospodářství a modul pivovaru

Modul, kterým se dále podrobně zabývá tato bakalářská práce, je pivovar. Ten je dále navázán na systém skladového hospodářství. Systém skladového hospodářství je softwarové řešení pro každodenní správu skladových pohybů. Skladové hospodářství zajišťuje především přehled o celkovém skladovém provozu, včetně jeho kontroly. Mezi další vlastnosti skladového hospodářství patří například snížení pravděpodobnosti chyb spojených s manipulací s položkami. Systémy skladového hospodářství také nesou odpovědnost za kontrolu a optimalizaci prováděných operací v rámci skladu. [28] Celkovým cílem modulu je zjednodušení a zpřehlednění agendy, které dokáže pokrýt veškeré logistické a evidenční nároky kladené cílovým zákazníkem.

Samotné moduly se systémem skladového hospodářství nekomunikují přímo. Viz obrázek níže. Jako mezičlen existuje databáze pro intranet. Tato databáze je poté navázána na systémy třetích stran, jako je tomu u systému skladového hospodářství.



Obrázek 2: Diagram komponent. Zdroj: Vlastní

1.2 Konkurenční systémy

Intranet, jakožto vnitřní systém podniku, je často jeho nezbytnou součástí. Intranet umožnuje celkové propojení podniku. Mimo jiné zajišťuje také komunikační kanály mezi zaměstnanci a poskytuje přehled o běhu podniku. Díky těmto a mnoha dalším výhodám jsou intranety více a více rozšířené, a to ve velkých i malých podnicích. Každý podnik má ovšem různé potřeby, proto nejčastěji vznikají systémy přímo pro jednotlivé podniky na základě jejich požadavků. Tato skutečnost se dnes efektivně řeší právě díky modularitě. Klient si poté vybere, jaké moduly bude pro podnik potřebovat a sestaví si je dle svých preferencí.

1.2.1 Systémy pro pivovar

V dnešní době existuje mnoho malých soukromých pivovarů. Často se tyto pivovary rozrostou a pro efektivnější práci vyhledávají systémy, které by usnadnily evidenci a zajistily větší přehled. Mezi společnosti zabývající se těmito systémy je například KARAT. Tato společnost nabízí mimo jiné právě systém pro výrobu nápojů, a to včetně piva. Jejich systém je velmi komplexní a zajišťuje pokročilé funkce, jako podpora více poboček, či podpora vratek. Tento systém je navržen již pro relativně veliké podniky, což dokazuje i skutečnost že tento systém využívá pivovar BERNARD. [29]

Oproti těmto systémům je v našem případě nutno dbát na grafickou stránku. V minipivovarech, kde pracuje málo lidí je nutné zajistit, aby bylo uživatelské prostředí vhodné i pro lidi, kteří nejsou s podobnými systémy více seznámeni.

2 TECHNOLOGIE VYUŽITELNÉ PRO VÝVOJ

Trendem posledních let je přechod z klientských aplikací na aplikace webové. Klientské aplikace je zpravidla zapotřebí instalovat na každý počítač, kde jsou potřeba. Při nutných aktualizacích aplikace musí každé zařízení instalovat novou verzi, což je náročné především časově. Další překážka nastává při potřebě využívání aplikace na různých typech přístrojů, jako jsou mobilní telefony, tablety nebo notebooky. [32]

Webové aplikace mají v těchto případech nesporné výhody. Je možné k nim přistupovat z jakéhokoliv zařízení a operačního sytému. Není potřeba aplikaci instalovat a při využití vhodné architektury při vývoji je možné aplikace velice snadno rozšiřovat a měnit. [32]

2.1 Vývojové nástroje

Jak již bylo zmíněno v úvodu, práce je věnována převážně klientské části programu, takzvanému front-endu. K tvorbě této části existuje výběr z velikého množství nástrojů a frameworků, které vývoj aplikace umožnují a také usnadňují.

Tyto nástroje často obsahují sady komponent a vizuálních prvků. Jelikož jsou tyto nástroje populární, tak jejich sady komponent jsou velmi rozšířené a uživatelé si na ně zvykli. Díky využívání těchto komponent se aplikace uživateli jeví jako přehlednější a intuitivnější na ovládání. Takovéto aplikace poté nemusí využívat pro každý prvek nápovědu, jak tomu bývalo dříve.

První z takových nástrojů, který byl využit, je Bootstrap. Bootstrap je volně dostupná sada nástrojů pro usnadnění vývoje klientských částí aplikace. Bootstrap také klade veliký důraz na responzivitu webu, což je v dnešní době velice důležité. Obsahuje především šablony pro HTML a CSS, čímž dělá práci s těmito technologiemi efektivnější. [1]

V neposlední řadě byl při vývoji klientské části využíván TypeScript. TypeScript je objektově orientovaný skriptovací jazyk, který se využívá pro vývoj webové aplikace. Je to klíčová technologie pro interakci s klientem. TypeScript je nadstavbou JavaScriptu, což je jeden z nejpoužívanějších nástrojů svého typu na světě. TypeScript se stará o obsluhu veškerých interaktivních části uživatelského rozhraní. Ovšem pro efektivní používání JavaScriptu a TypeScriptu je doporučené využívat určený framework. Framework neboli aplikační rámec je programový nástroj, který poskytuje nástroje a knihovny pro zjednodušení běžných vývojových operací. Výsledkem je snadnější a přehlednější cesta vývoje aplikací. Nejrozšířenější webové frameworky jsou dále popsány podrobněji. [2], [3]

2.1.1 Angular

Angular je framework pro tvorbu klientské části webových aplikací. Poskytuje především komplexní nástroje pro efektivnější a udržitelnou tvorbu webových aplikací a zároveň definuje určité předpisy a pravidla, podle kterých by se měly webové aplikace tvořit. [6]

Předchůdce Angularu je AngularJS, z kterého Angular zcela vychází. AngularJS, stejně jako Angular jsou vytvořeny a spravovány firmou Google. Předešlý AngularJS byl poprvé vydán v roce 2010 a je postaven čistě na JavaScriptu. V roce 2016 byla poprvé vydána verze nového frameworku s názvem Angular 2, dnes již známého pouze jako Angular. Angular se svým předchůdcem není kompatibilní, a to především z důvodu, že nový Angular je postaven na jiném jazyku, a to na TypeScriptu. TypeScript je nadstavbou JavaScriptu, což je jeden z důvodů, proč je Angular náročnější na pochopení pro začátečníky. Angular je komplexní framework s vlastní filozofií, a to, že využívá jiný jazyk než většina jeho konkurentů, bývá pro nové uživatele překážka. Je možné vyvíjet aplikace pomocí Angular a používat moderní verzi jazyka JavaScript, ovšem není to doporučeno a velmi málo lidí využívá tento způsob. [4], [5], [7]

2.1.2 Vue.js

Vue.js je rozšířený JavaScriptový framework pro vývoj uživatelského rozhraní. Je považován za jeden z nejlehčích frameworků svého druhu a je vhodný pro začínající vývojáře. Je snadný na implementaci a umí se snadno přizpůsobit. Je to ideální nástroj při využití v menším měřítku. Je ovšem zcela možné vytvářet pokročilé jednostránkové aplikace s kombinací s dalšími moderními nástroji a knihovnami. Na rozdíl od Angularu je méně rozšířený a je vhodný pro menší projekty. Na druhou stranu Vue.js je díky své jednoduchosti méně náročné na výkon a snazší na naučení i používání. [8], [9]

2.1.3 React

React je JavaScriptová knihovna, která poskytuje nástroje pro snazší tvorbu interaktivních uživatelských rozhraní. React je zcela jistě nejpopulárnější JavaScriptový framework. Poprvé byl představen v roce 2011 a je spravován firmou Facebook. [10]

Díky Reactu je možné jednodušeji vytvářet dynamické webové aplikace a oproti samotnému JavaScriptu poskytuje více možností s nižší náročností na vývoj. Pokud je uživatel již seznámen se základy vývoje klientských částí aplikací, je React velmi snadný na naučení, jelikož využívá

principy JavaScriptu a HTML, které rozšiřuje o další funkcionality. Samozřejmostí jsou poté některé pokročilejší funkce a principy, kterým je potřeba věnovat nějaký čas na pochopení a naučení. [10], [11]

2.2 REST

REST (Representational State Transfer) je architektonický styl využívaný při vývoji a práci s webovými službami. REST je velice populární pro svou jednoduchost. Tato skutečnost je dána především tím, že staví nad již zavedenými koncepty, jako je HTTP (Hypertext Transfer Protocol). REST poprvé představil v roce 2000 Roy Fielding jako součást své disertační práce. [12]

Při využívání REST architektury REST Server umožnuje přístup ke zdrojům. Klient poté ke zdrojům přistupuje a je mu umožněno je měnit. Zdroje v tomto případě mohu být nejčastěji reprezentovány prostým textem nebo pomocí JSON (JavaScript Object Notation). Pokud chce tedy klient přístup ke zdrojům na serveru, musí odeslat žádost, která musí mít určitou strukturu. [13]

Další pojem, který je potřeba zmínit, je REST API. API (Application Programming Interface) je v tomto případě součástí webové služby, která čeká a reaguje na dotazy klienta. Webové služby tedy využívají API pro komunikaci. Moderní přístup k tvorbě API je využití právě konceptu REST. REST má také několik pravidel, které je při využívání jeho architektury potřeba dodržovat. Dvě základní pravidla jsou rozepsána podrobněji v následujících kapitolách. [30]

2.2.1 Client-Server

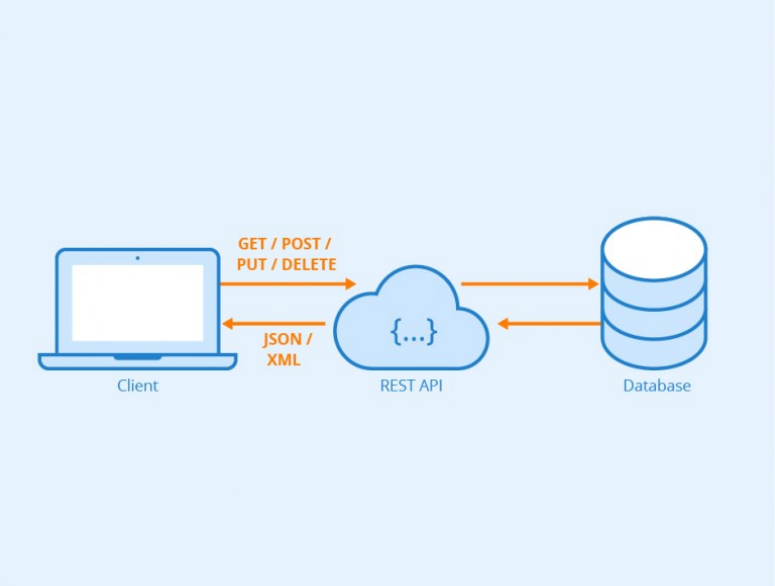
První z těchto pravidel je „Client-Server“. Toto pravidlo říká, že uživatelské prostředí by mělo být odděleno od úložiště dat. Díky tomu se zvýší kompatibilita klientské části s více platformami. V neposlední řadě dodržování tohoto pravidla umožnuje nezávislé rozšiřování jednotlivých částí. [33]

2.2.2 Stateless

Toto pravidlo říká, že veškerá komunikace mezi klientem a serverem je bezstavová. Klient tedy při komunikaci nevyužívá žádných uložených dat na straně serveru, jako je tomu u stavové komunikace. Požadavek od klienta obsahuje veškeré informace pro ověření a zpracování. [33]

2.2.3 REST komunikace

Webová služba využívající REST API se nazývá „RESTful“. REST API poté komunikuje pomocí HTTP metod, díky čemu se stává nezávislé a flexibilní. Dále na obrázku číslo 3 je možné si povšimnout, že API funguje jako prostředník pro komunikaci. Díky využití principů REST není API závislé na komunikujících stranách a je dnes již velmi žádaným standardem. [30]



Obrázek 3: Princip REST API. Zdroj: [31]

Žádost zpravidla začíná úvodním řádkem, který obsahuje typ HTTP metody, verzi protokolu a adresu cíle. Další částí je hlavička žádosti. Ta obsahuje dodatečná data jako informace o odesílateli nebo požadovaný typ odpovědi. Dále hlavička zpravidla obsahuje informace o těle, jako je například jeho délka a typ. Poslední částí žádosti je tělo. Tato část není povinná a metody jako GET nebo DELETE ji často nemají. Ovšem metody jako POST nebo PUT tělo zpravidla mají, jelikož v jeho rámci odesílají data klienta pro server. [14], [18]

Poté, co server obdrží žádost, zpravidla odešle odpověď. Odpověď začíná stavovým řádkem, který obsahuje verzi protokolu, stavový kód a stavový text. Stavové kódy jsou předepsané a nesou informaci o výsledném stavu zpracované žádosti. Mezi nejčastější patří kód „200 OK“, který značí úspěšné zpracování žádosti. Naopak kód „404 Not Found“ značí, že server nenašel požadovaný zdroj. Odpověď dále obsahuje hlavičku a tělo. Tyto části podléhají stejné struktuře jako u žádosti. [14], [15]

V následujících kapitolách jsou podrobněji popsány nejčastěji používané HTTP metody pro přístup a manipulaci se zdroji.

2.2.4 Metoda GET

GET je nejznámější a nejvíce používaná metoda. Tato metoda se využívá pouze pro získání zdrojů ze serveru. Metoda má povoleno data pouze číst a nelze tedy pomocí této metody data upravovat nebo mazat. Díky této skutečnosti je metoda GET považována za bezpečnou. Další takovou bezpečnou metodou je metoda HEAD, která může data také pouze číst. [16], [17]

2.2.5 Metoda POST

Metoda POST se využívá pro vytvoření nového zdroje. Na rozdíl od většiny ostatních metod se v tomto případě neuvádí identifikátor zdroje, se kterým je zamýšleno pracovat. Metodou POST se vytváří nový zdroj, tedy identifikátor není předem známý. Ve většině případů je identifikátor navrácen po úspěšném vytvoření zdroje v odpovědi. Tato metoda není považována za bezpečnou, jelikož vytvoření nového zdroje je považováno za modifikaci dat. Metoda POST má ovšem další vlastnost, díky které se liší od většiny ostatních. Metoda není idempotentní, což znamená, že po opětovném volání metody se stejnými parametry se dosáhne vždy jiného výsledku. Po každém následujícím volání se vytvoří další zdroj s jiným identifikátorem. Jiné, zde zmíněné metody způsobí po opětovném volání stejný výsledek. Na základě toho je potřeba postupovat s metodou POST opatrněji a opětovné volání příslušně ošetřit. [16], [17]

2.2.6 Metoda PUT

PUT je metoda, která se využívá pro změnu již existujícího zdroje. V určitých situacích, pokud zdroj neexistuje, metoda PUT jej může vytvořit. V tomto případě je nutné informovat klienta o vytvoření nového zdroje spolu s novým identifikátorem, pokud existuje. Tato metoda není považována za bezpečnou, jelikož zasahuje do dat. Ovšem na rozdíl od POST je idempotentní, což znamená, že po opětovném volání se stejnými parametry docílíme stejného výsledku. Tedy pokud již daný zdroj existuje, pouze se touto metodou aktualizuje. [16], [17]

2.2.7 Metoda DELETE

Metoda DELETE slouží pro odstranění specifického zdroje. Patří mezi běžné metody a je také snadné ji používat. Stačí pouze definovat identifikátor zdroje a metodou DELETE jej snadno

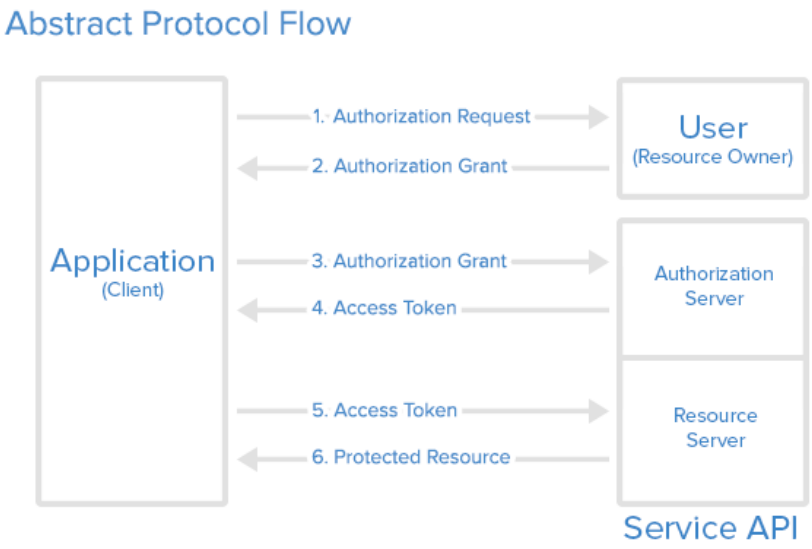
odstranit. Metoda DELETE není považována za bezpečnou, ale je idempotentní. Po pokusu odstranit daný zdroj vícekrát tedy nevznikne změna v datech. Ovšem pravděpodobně se ohlásí chybový návratový kód od databáze, jelikož zdroj již nebude existovat. [16], [17]

2.3 OAuth

OAuth (Open Authorization) je otevřený standard pro autorizaci na internetu, který může být využit na jakékoliv platformě. V roce 2007 se prvně objevila první verze tohoto standardu. Tento nový přístup se začal hojně využívat a později vznikla verze OAuth 2.0, která se využívá dodnes. Tato verze není zpětně kompatibilní a pokud se dnes mluví o protokolu OAuth, ve většině případů se jedná o aktuální verzi OAuth 2.0. [19], [20]

OAuth umožnuje aplikacím získat zabezpečený částečný přístup k datům uživatele, které se nacházejí v jiné aplikaci, bez nutnosti opětovného zasílání uživatelských údajů. S OAuth se lze na internetu setkat vcelku často. Časté využití je například u známého dialogu s otázkou „Chcete se přihlásit na naše stránky pomocí účtu ze stránky jiné?“ V tomto případě se první stránka spokojí s tím, že je uživatel přihlášen na nějaké známé aplikaci, jakou mohou být například Google či Facebook. [21]

Při popisu standardu OAuth je potřeba definovat tři role. První je uživatel neboli vlastník zdroje. Jedná se o někoho, kdo vlastní účet a může udělovat oprávnění jiným aplikacím přistupovat k tomu účtu. Druhou rolí je server, který uchovává data klienta a zajišťuje nad nimi ochranu. Poslední rolí je aplikace třetí strany, která požaduje přístup k uživatelově účtu. Celá interakce poté začíná žádostí aplikace třetí strany, ve které specifikuje uživatelovi, jaká oprávnění žádá. Uživatel schválí žádost a aplikace získá potvrzení. Toto potvrzení spolu se svou identitou poté předá serveru, ke kterému žádá přístup. Server ověří přijaté informace a při schválení vydá aplikaci autorizační token. Pomocí tohoto tokenu se poté aplikace autorizuje při žádání dat ze serveru. Průběh této komunikace je pro lepší představení znázorněn na obrázku 4. [21], [22]



Obrázek 4: Průběh komunikace využívající OAuth. Zdroj: [22]

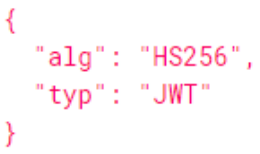
2.4 JWT

JWT (JSON Web Token) je otevřený internetový standard pro vytváření ověřovacích tokenů pro aplikace. JWT slouží jako bezpečně přenosný objekt pro komunikaci mezi stranami. Aby byla zajištěna důvěryhodnost, je nutné JWT digitálně podepsat. Pro tyto účely se využívá především princip asymetrického šifrování. Ovšem je možné použít i méně bezpečný princip s jedním tajným klíčem. [23], [24]

JWT má největší využití při autentizaci. Existují dvě hlavní metody autentizace. První je starší technologie, při které se serveru prvně poskytnou údaje pro autentizaci a server vytvoří unikátní identifikátor. Tento identifikátor pak drží spolu s dalšími informacemi server, ale také i klient. Klient poté při přístupu k serveru použije identifikátor a server ho porovná s jeho uloženými záznamy. Jelikož server musí udržovat data veškerých aktivních klientů, snižuje to jeho výkonost a paměťový prostor. Další nevýhoda nastane, pokud jedna služba využívá více serverů. V takovémto případě je nutné implementovat mezi servery algoritmus, který zajistí provázání jednotlivých identifikátorů a uživatelů. Poslední zmíněná nevýhoda se týká sdílení informací dalším aplikacím. Tato problematika byla již zmíněna spolu s OAuth. Aplikace třetích stran totiž nemají způsob ověřovat uživatelovo správné přihlášení, jelikož tato část probíhá až na straně serveru. [25], [26]

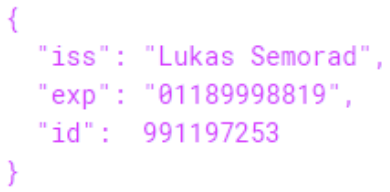
Na základě těchto nevýhod vznikl další způsob autentizace. Ten byl popsán v předešlé kapitole v rámci OAuth. Hlavní princip spočívá v tom, že server na dotaz vygeneruje token, který podepíše a předá klientovi. Ten poté při komunikaci se serverem přiloží token a server pouze ověří pravost tokenu. Server tedy nemusí uchovávat žádné informace o klientovi. Tokeny mají zpravidla určitou dobu platnosti, která v definovaný čas vyprší. Souvisí to s nevýhodou, kterou je to, že server nemůže zneplatnit ověřené zrušení jako mohl u první metody ověření. Kvůli této skutečnosti je tedy token zpravidla platný vždy po danou omezenou dobu. [25]

Samotný token se skládá z hlavičky, těla a podpisu. Jak je vidět na obrázku níže, celý token je strukturován podle standardu zápisu JSON. Hlavička tokenu typicky obsahuje dva záznamy. První je typ tokenu, což je JWT, a druhý je použitý algoritmus pro podpis tokenu. V tomto případě byl použit algoritmus HS256. [27]



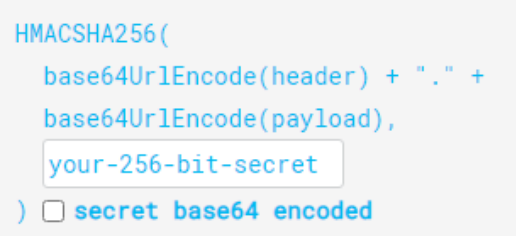
Obrázek 5: Struktura hlavičky JWT. Zdroj: Vlastní

Tělo tokenu, které je vidět na obrázku níže, může obsahovat uživatelská data. Existují také určité rezervované položky, jako je „iss“, což je vydavatel, nebo „exp“, což je informace o datu expirace tokenu. [24]



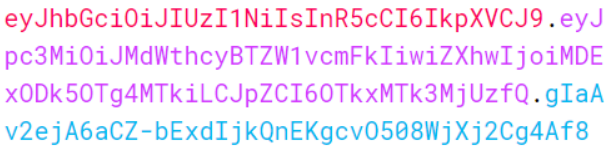
Obrázek 6: Struktura těla JWT. Zdroj: Vlastní

Poslední částí tokenu je podpis. Prvně se vezme hlavička a tělo tokenu, které se upraví pomocí URL kódování a poté jsou obě části oddělené tečkou spolu s vlastním heslem zašifrovány pomocí metody uvedené v hlavičce tokenu. Tento postup zobrazuje obrázek níže. [27]



Obrázek 7: Struktura podpisu JWT. Zdroj: Vlastní

Všechny tři části oddělené tečkou poté tvoří celý JWT. Na obrázku níže je možné vidět jednotlivé části tokenu barevně rozlišené.



Obrázek 8: Vygenerovaný JWT. Zdroj: Vlastní

3 POŽADAVKY

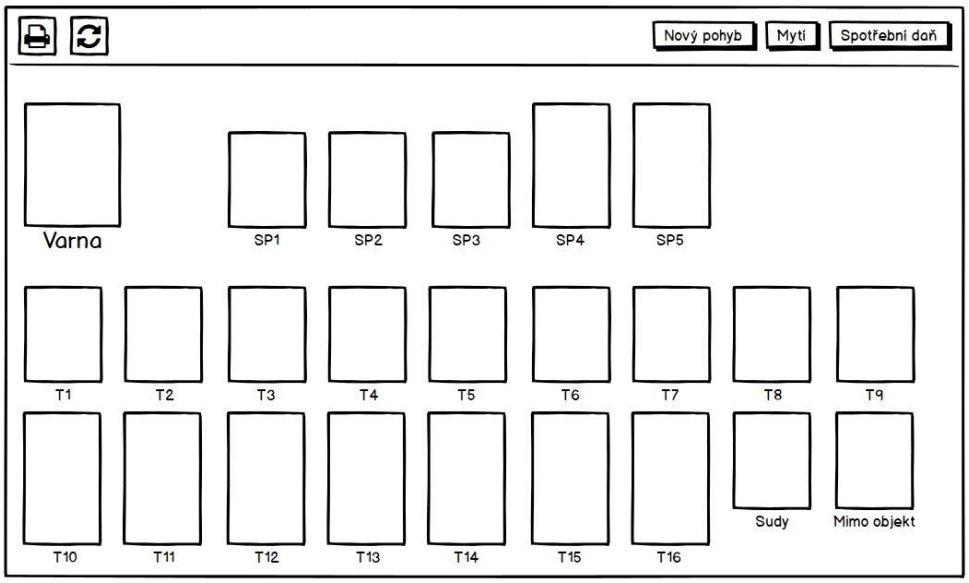
Po seznámení se se systémem bylo potřeba definovat veškeré požadavky na daný modul. Modul je již modelován pro stávající provozovnu, tedy veliká část požadavků je dána koncovými uživateli. To se týká především grafického zpracování systému a implementování daných funkcí. V neposlední řadě bylo také potřeba zajistit správnou návaznost na celý intranet a zachování typické modularity.

Konkrétní modul pivovaru bude obsluhovat především sládek. To je osoba, která se stará o veškeré procesy spojené s výrobou piva. Tyto procesy je ovšem nutné zaznamenávat a zanášet do systému, a to nejen pro evidenci. Sládek má často letité zkušenosti s veškerými procesy při přípravě piva. Ovšem přes tyto zkušenosti nemusí být vždy snadné zanášet tyto procesy do evidence. Z tohoto důvodu je nutné vytvořit nadstavbu, která tuto bariéru odstraní. Do systému bude mít zpravidla přístup také manažer. Pro toho bude důležitá možnost zjištění stavu skladů. Nadstavba bude graficky zobrazovat jednotlivé sklady v reálném čase, a díky tomu bude mít manažer vždy plný přehled.

3.1 Grafické prostředí

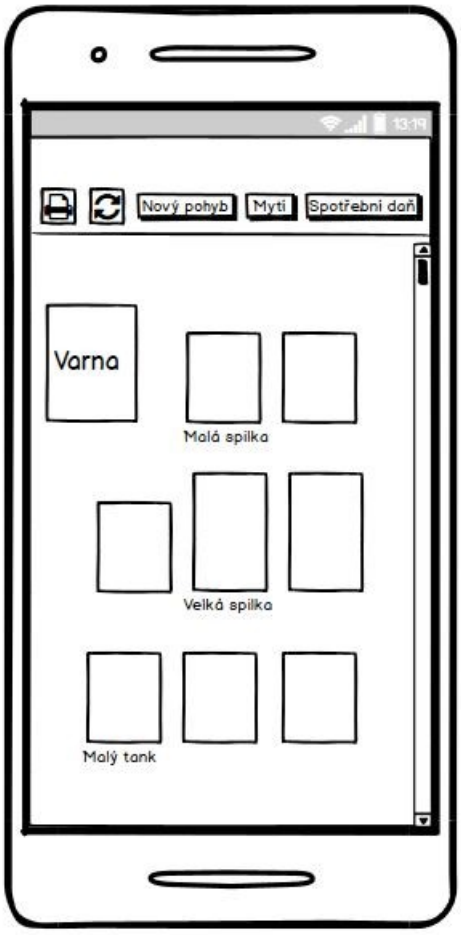
Mezi nejdůležitější požadavky v rámci návrhu patří zcela jistě uživatelsky jednoduché, moderní a responsivní uživatelské prostředí. Je nutné nezapomínat, že pro uživatele bývá grafické prostředí jedna z nejdůležitějších, ne-li nejdůležitější část celého projektu.

Další věcí je uvědomit si, kdo a za jakých podmínek bude grafické prostředí využívat a obsluhovat. Na základě těchto informací bylo potřeba vytvořit grafické uživatelské rozhraní, které by se snadno obsluhovalo v každodenním provozu. Za tuto obsluhu zodpovídá především sládek, který s podobnými systémy často nepracuje. Bylo nutné zajistit veškeré nutné funkce a přehlednost potřebných dat současně s uživatelsky intuitivním a velmi jednoduchým designem. Jak je viditelné dále na obrázku číslo 9, v horní části byla navržena lišta s ovládacími prvky. Zbytek obrazovky je vyhrazen jednotlivým objektům pivovaru.



Obrázek 9: Návrh grafického prostředí modulu pivovaru. Zdroj: Vlastní

V neposlední řadě je nutné vymezit zařízení, která budou systém využívat a na jejich základě grafické prostředí přizpůsobit. Základní vývoj je zpravidla určen pro stolní počítače s monitory s rozlišením Full HD. Ovšem v tomto konkrétním případě bylo nutné zajistit dodatečnou optimalizaci také pro tablety a mobilní telefony. Ty v podobném provozu zaměstnanci standardně využívají. Při změně obrazovky je tedy nutné objekty pivovaru přeskládat, jak je viditelné dále na obrázku číslo 10. Objekty pivovaru bude tedy nutné dynamicky upravovat na základě aktuální velikosti obrazovky. Nelze také opomenout nutnost přizpůsobit jednotlivé prvky dotykovým displejům.

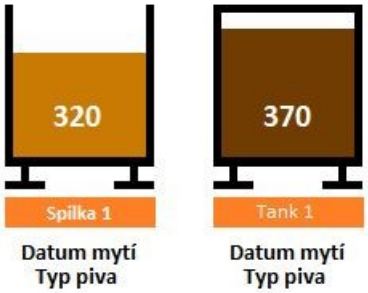


Obrázek 10: Návrh grafického prostředí v mobilním zobrazení. Zdroj: Vlastní

3.1.1 Objekty

Největší část grafického rozhraní tvoří objekty spilek a tanků. Grafická stránka těchto objektů byla navržena pro zajištění nejvyšší přehlednosti a optimální funkčnosti. Jednotlivé nádoby bylo potřeba navrhnout, aby zobrazovaly reálnou hladinu piva, a to vždy v poměru k velikosti dané nádoby. Barva vykreslovaného piva je dána jeho typem. Pokud je v nádobě pivo, je nutné zobrazit ve středu nádoby jeho objem v litrech. Pod nádobou bude viditelné datum posledního pohybu s danou nádobou a aktuální typ piva.

Na následujícím obrázku číslo 11 je vytvořený návrh dvou hlavních typů objektů pivovaru. Objekty se skládají ze dvou primárních částí. První je grafické znázornění objektu. To ukazuje aktuální množství piva a zároveň zobrazuje reálný poměr piva k nádobě. Pod tímto grafickým zobrazením se nacházejí další informace o objektu, které jsou potřebné pro uživatele.



Obrázek 11: Návrh objektů pivovaru. Zdroj: Vlastní

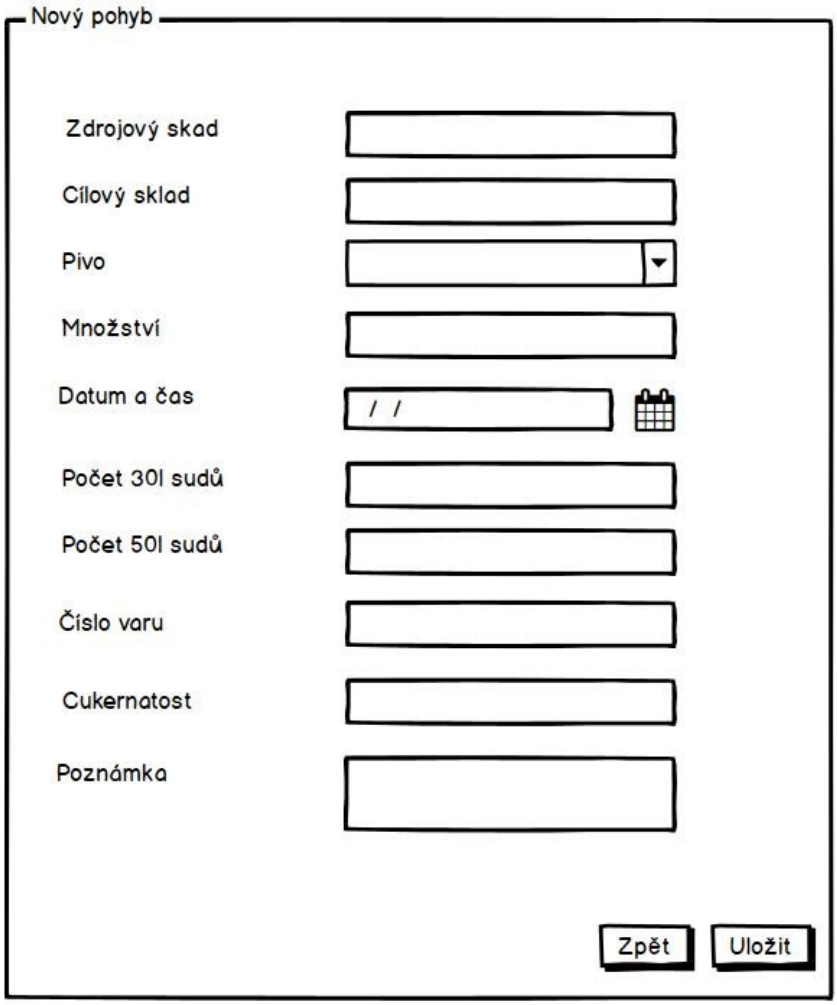
3.2 Tisk stránky

Dalším klíčovým požadavkem je možnost tisku obsahu dat na webové stránce. Stránku bude potřeba příslušně přizpůsobit k danému tisku, jelikož se na stránce budou nacházet objekty, které jsou v tiskové podobě nežádoucí. Především bude nutné stránku zjednodušit, a zbavit nepodstatných grafických prvků. Dále je žádoucí zaměřit se na potřebnou výslednou strukturu dat, která musí mít určitou předepsanou formu.

3.3 Přesun piva

Jednou z nejpoužívanějších funkcí v pivovaru bude přesun piva. Jedná se o funkci, která se bude využívat pro několik druhů přesunů. Základní z nich je přesun z varny do spilky. Tímto pohybem se zaznamená uvaření nového piva. Mezi další typy pohybů patří přesuny mezi objekty v pivovaru. Například přesun piv ze spilky do tanku. Dále bude možné pivo přesunout do sudů.

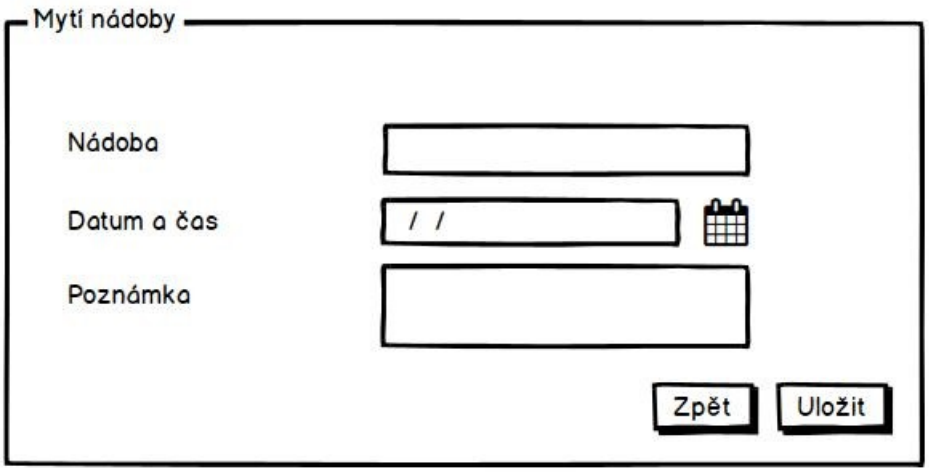
Pro veškeré tyto pohyby bude nutné navrhnout dialogové okno, které zajistí správný průběh přesunu a přizpůsobí se dle aktuálního typu přesunu. Pole, která jsou vždy viditelná, jsou: zdrojový a cílový sklad, typ piva, množství piva pro přesun, datum a čas přesunu a poznámka. Ostatní pole se mění v závislosti na zvoleném typu přesunu. Následující obrázek číslo 12 zachycuje dialogové okno se všemi možnými parametry. V reálné situaci tedy budou zobrazena pouze vybrané pole a dialog bude mít zpravidla menší velikost.



Obrázek 12: Návrh dialogu pro obsluhu přesunu piva. Zdroj: Vlastní

3.4 Evidence mytí nádob

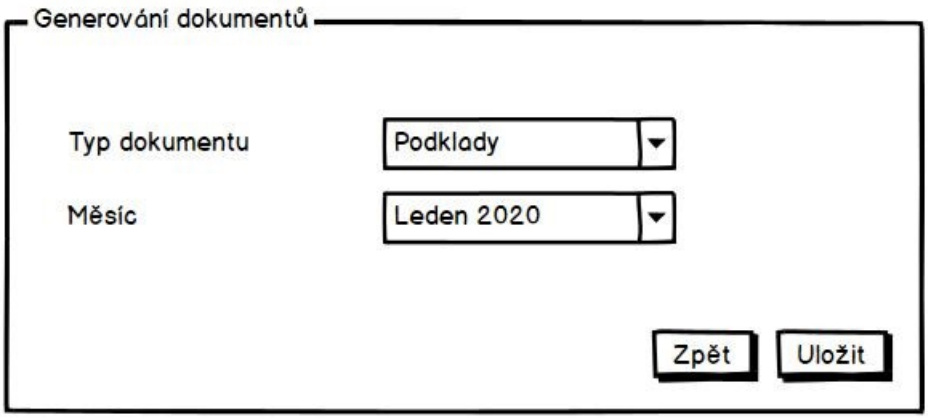
Zde se již jedná o specifičtější požadavek, jelikož je přímo spojený s potřebami pivovaru. Po zvolení příslušné nádoby, nad kterou bude uživatel potřebovat provést danou funkci, se vyvolá dialogové okno. Dialogové okno bude předvyplněno známými informacemi. Uživatel bude mít možnost některé informace pozměnit a odeslat požadavek. Návrh tohoto dialogu je vidět níže na obrázku.



Obrázek 13: Návrh dialogu pro obsluhu mytí nádoby. Zdroj: Vlastní

3.5 Generování dokumentů pro úřady

Jeden z požadavků modulu je také možnost generování dokumentů pro úřady. Respektive se jedná o možnost stáhnutí určitých dokumentů mapujících a agregujících vybrané skladové pohyby s případným pozdějším tiskem. Po výběru parametrů dokumentu se jednoduše soubor stáhne a uživatel má poté možnost dokument odeslat, vytisknout nebo pouze zobrazit. Na základě požadavků dialogové okno předvyplní vždy minulý měsíc. Uživatel může zvolený měsíc upravit a dále zvolit typ dokumentu, jak je zachyceno na obrázku číslo 14.



Obrázek 14: Návrh dialogu pro generování dokumentů. Zdroj: Vlastní

4 REALIZACE ŘEŠENÍ

Na základě zadaných požadavků bylo potřeba navrhnout a vytvořit ideální strukturu, která by byla uživatelsky přívětivá a snadná na ovládání. Při návrhu a následné tvorbě uživatelského prostředí bylo potřeba využít moderních technologií, které se využívají pro responsivní design webových aplikací. Především díky použití těchto technologií byla možná optimalizace zobrazení pro různé typy zobrazovacích zařízení. Zároveň bylo nutné postupovat s ohledem na optimální funkčnost v nejrozšířenějších internetových prohlížečích. V neposlední řadě je uživatelské prostředí plně přizpůsobeno možnosti ovládání pomocí dotykových displejů.

4.1 Třívrstvá architektura

Při realizaci byl využit princip třívrstvé architektury. Díky této skutečnosti je aplikace celkově flexibilnější. Jednotlivé vrstvy mohou být dále upravovány nezávisle na sobě za předpokladu, že nedojde ke změně komunikačních rozhraní. Díky využití tohoto principu se aplikace stává lépe rozšiřitelnou. Mezi další výhody patří skutečnost, že jednotlivé vrstvy zpravidla běží na různých zařízeních. Díky tomu dochází k efektivnějšímu rozdělení výkonu a klientská část může být používána i s levným koncovým vybavením.

4.1.1 Klientská část

Jelikož se jedná o webovou aplikaci, uživatel může přistupovat ke klientské části odkudkoliv, nezávisle na použitém zařízení či platformě a bez nutnosti instalace a aktualizace klientského programu. Pomocí svého účtu se přihlásí do intranetu, či do samotného modulu na libovolném počítači nebo mobilním zařízení s přístupem na Internet. Vývoji klientské části jsou věnovány další kapitoly, jelikož se jedná o hlavní část, kterou se tato bakalářská práce primárně zabývá. Jsou zde popsány jednotlivé ovládací prvky spolu s celkovým návrhem a vyhotovením uživatelského prostředí. Viz kapitola 4.2.

4.1.2 API

Prostřední vrstva se stará o výměnu informací mezi klientskou a datovou vrstvou a funguje pro ně jako most. Při práci s API byla využívána architektura REST. Tato architektura byla podrob-

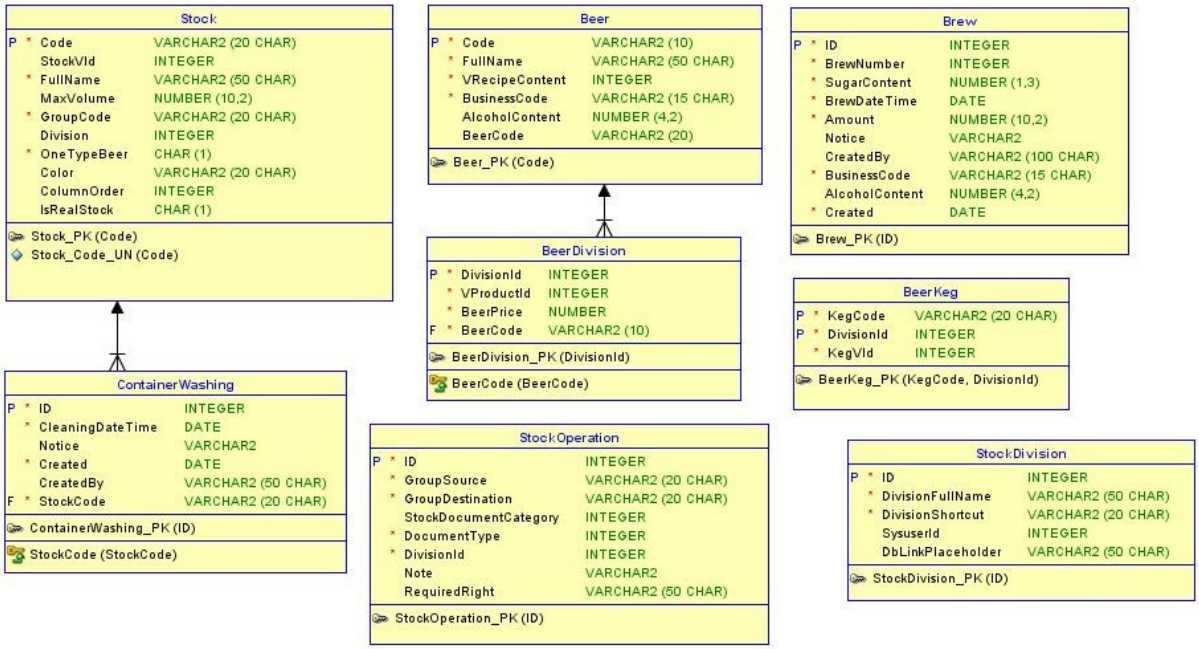
něji popsána v kapitole 2.2. Na základě REST principů byly vytvořeny základní metody CRUD (Create, Read, Update, Delete). V našem případě se jednalo o HTTP metody POST, GET, PUT a DELETE.

Další částí je zajištění autentizace. Pro zajištění autentizace se využívá protokol OAuth. Ten nabízí řešení pomocí tokenů. Více lze zjistit v kapitole 2.3. Pro tyto účely byl využit takzvaný JSON Web Token. Pro více informací o JWT, viz kapitola 2.4. Poté co se uživatel pomocí svých údajů přihlásí, je mu zaslán právě JWT. Ten dále zůstává uložen na straně klienta. V našem případě se jedná o uložení přímo ve webovém prohlížeči, formou takzvaných cookies. Poté, co si uživatel přeje zaslat požadavek pomocí jedné z výše zmíněných HTTP metod, je nutné zaslat také JWT. Token je automaticky přiložen ke každému požadavku při jeho odesílání. Jak je vidět níže na obrázku, JWT je zaslán v hlavičce, jako „Authorization“. Server díky tomu poté dokáže ověřit věrohodnost požadavku, a to s nízkými nároky na jeho výkon.

Obrázek 15: Hlavička HTTP metody. Zdroj: Vlastní

4.1.3 Databáze

Následující obrázek číslo 16 zachycuje relační model databáze, která byla použita při realizaci modulu pivovaru. Jedná se o část databáze, kde jsou uložena data, které pivovar přímo využívá, a to především při vykonávání akcí v jeho rámci. Tato databáze je dále provázána s proprietárním systémem třetí strany. Jedná se o restaurační a skladový systém, který umožnuje propojení s jinými moduly v rámci intranetu a zajištuje komplexnější kontrolu provozu.



Obrázek 16: Relační model. Zdroj: Vlastní

Obrázek zachycuje celkem 8 tabulek. První tabulka „Stock“ slouží pro uchování informací o jednotlivých nádobách na pivo. Tabulka obsahuje například informace o velikosti nádoby nebo kódové označení, pod kterým se zobrazuje uživateli. Další důležitá tabulka je „Beer“. Jedná se o číselníkovou tabulku, která umožňuje do systému snadno přidat další druh piva. Základní informace v této tabulce jsou mimo jiné podnikové označení piva, podíl alkoholu a plný název piva. Tabulka s názvem „ContainerWashing“ slouží k uchovávání záznamů o mytí jednotlivých nádob. Další zajímavá tabulka je pro nový var piva. Tabulka „Brew“ obsahuje veškeré záznamy o nově uvařených pivech. Mezi její atributy patří například přesné množství alkoholu daného varu, číslo varu, nebo obsah cukru.

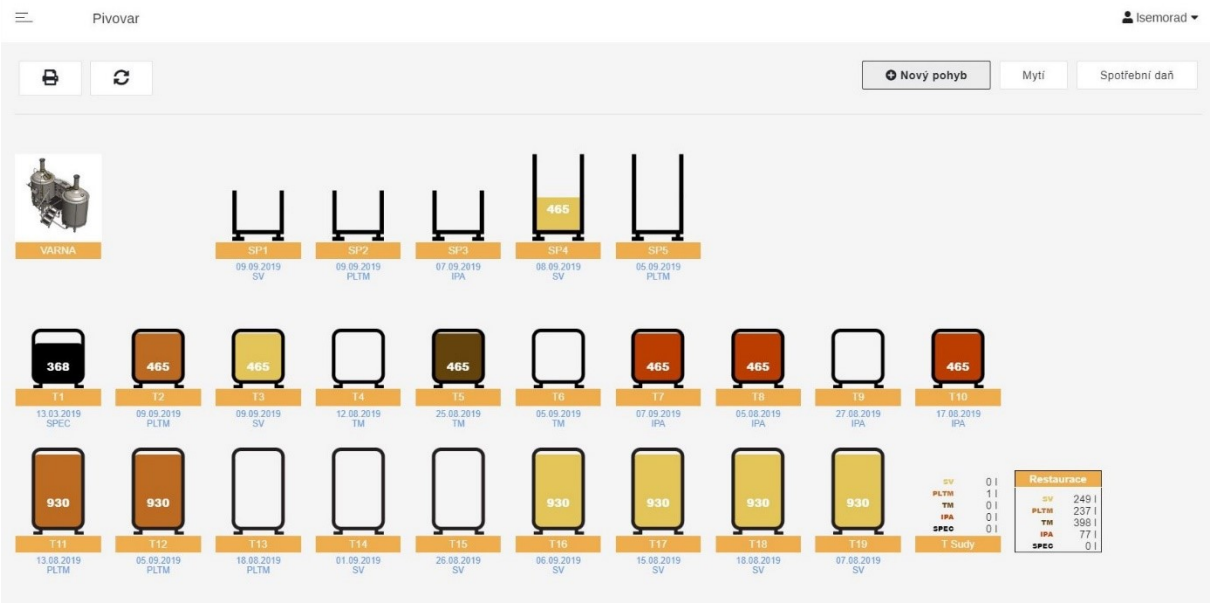
4.2 Ovládací prvky

Pro jednotlivé požadované funkce bylo potřeba navrhnout nejoptimálnější ovládací prvky. Pro základní funkce byl v horní části uživatelského rozhraní vymezen prostor, ve kterém se budou nacházet tlačítka pro obsluhu akcí. Levá část obslužného prostoru obsahuje tlačítko s možností tisku stránky a následně tlačítko sloužící k obnovení dat na stránce. Po stisku tlačítka pro tisk stránky se vyvolá speciálně upravený náhled stránky aplikace, přizpůsobený právě pro tisk.

Uživatel po kontrole náhledu dostane možnost stránku vytisknout, uložit ve vhodné formě, popřípadě zrušit akci zavřením náhledu. Tlačítko pro obnovení dat je zde umístěno pro vynucení opětovného načtení dat z databáze. Aplikace ovšem klade důraz na stálou automatickou aktuálnost všech dat. Z toho vyplývá, že akci pro vynucenou aktualizaci dat není nutné využívat, ovšem v určitých případech je z uživatelského hlediska komfortní.

V pravé části ovládacího prostoru se nacházejí tlačítka pro obsluhu zbylých tři akcí, které budou popsány dále. Obrázek číslo 17 níže, již zachycuje výslednou podobu grafického uživatelského prostředí hlavní stránky modulu pivovaru. V horní části obrazovky je vidět výše zmíněná ovládací lišta. Dále jsou zde vyobrazeny uspořádané objekty pivovaru již se zkušebními hodnotami piv.

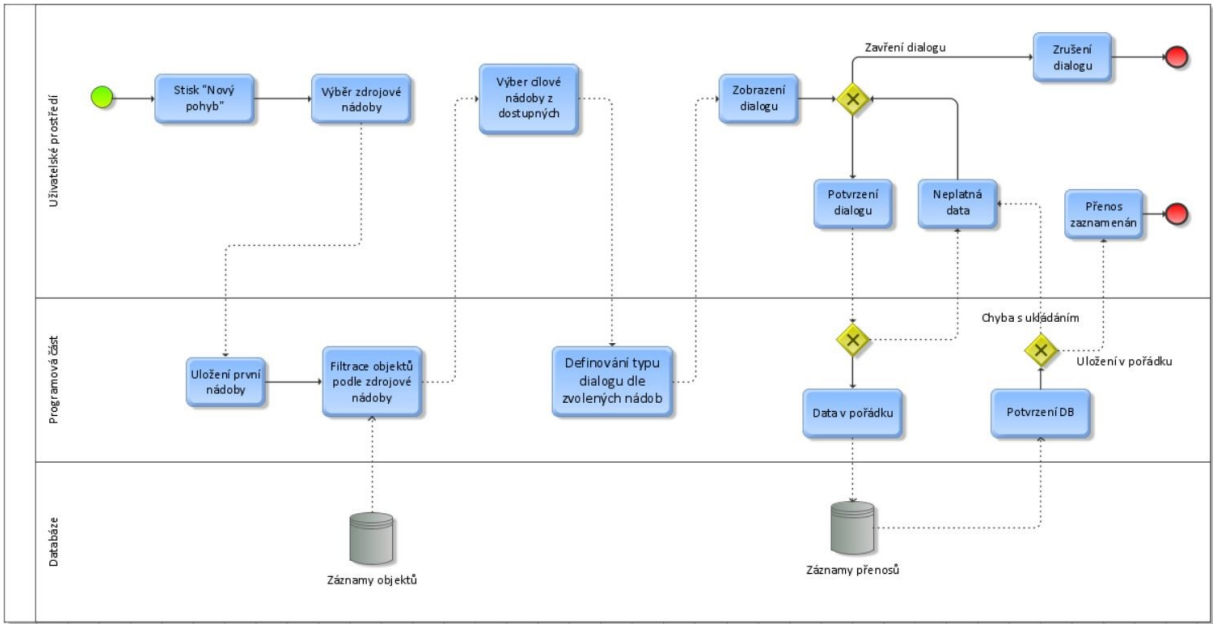
Díky využívání veřejných rozšíření pro efektivnější vývoj, jako je Bootstrap 4, bylo možné využívat předdefinované sady nástrojů. Mezi takovéto nástroje pařily například ikony. Ikona je univerzální zástupný symbol, který aplikaci dodá přehlednější vzhled. Na základě využívání těchto známých sad nástrojů, také nebylo nutné implementovat zastaralé způsoby nápověd, které se dříve skrývaly v každém rohu každého prvku aplikace. V našem případě byl zvolen modernější přístup, pomocí kontextových nápověd.



Obrázek 17: Hlavní stránka modulu pivovaru. Zdroj: Vlastní

4.2.1 Nový pohyb

První z akcí je pravděpodobně jedna z nejdůležitějších, jelikož se jedná o akci, která zajišťuje nový pohyb piva. Tato akce je vcelku jednoduchá na ovládání, jelikož se jedná o často využívanou funkci, ovšem je také velmi komplexní a zajištuje několik různých režimů. Po stisku tlačítka pro nový pohyb se aktivuje jakýsi mód pro výběr objektů pivovaru, který je signalizován v horní části obrazovky. Základním principem je zvolení zdrojového a následně cílového objektu, na základě kterých se zobrazí dialogové okno s upřesňujícími parametry pro přesun piva. V jakémkoli bodě je možné mód výběru objektů zrušit stiskem nápisu, který tento mód signalizuje. Názorně tyto procesy zachycuje obrázek číslo 18.



Obrázek 18: Diagram procesů nového pohybu. Zdroj: Vlastní

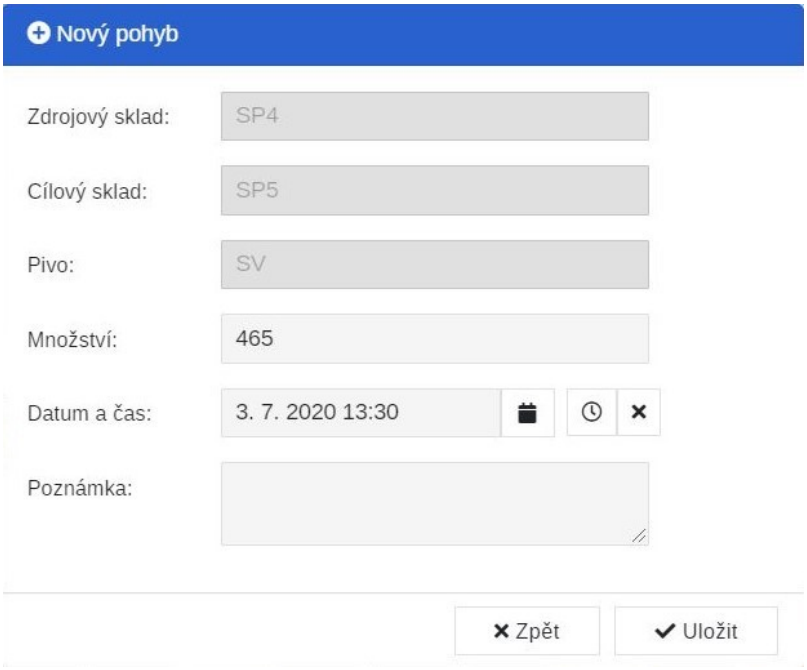
Prvním režimem pohybu by se dalo nazvat uvaření nového piva. Uživatel po výběru nového pohybu vybere jako zdroj přímo varnu piva. Vzápětí se zvýrazní pouze určité objekty, do kterých je možné převést nově uvařené pivo. Po zvolení vhodného cílového objektu, se uživateli zobrazí dialogové okno s výběrem dalších podrobností.

V dialogovém oknu se nacházejí upřesňující informace o pohybu piva. První, již předvyplněné informace, jsou zdrojový a cílový sklad. Tyto informace jsou v dialogu již neměnné, a slouží pouze pro kontrolu. Dalším parametrem je typ přenášeného piva. To je velmi důležitý parametr

při pohybu piva právě z varny. Zde má uživatel na výběr z databázového seznamu všech evidovaných typů piv. Následně je potřeba zkontrolovat množství piva pro přenos. Tento parametr se automaticky přizpůsobí cílové nádrži, ovšem je možné množství libovolně upravit s omezením, které udává objem cílové nádoby. Následujícím parametrem je datum a čas přesunu piva. Tento parametr je také předvyplněný, a to momentálním časem. Čas i datum je samozřejmě možné také upravit dle potřeb.

Při přesunu nového piva z varny je také nutné doplnit číslo varu. Číslo varu je jakési označení, které je pro uživatele v dialogu taktéž předvyplněno, ovšem je možné tento parametr ručně přepsat. V neposlední řadě je u nově uvařeného piva nutné zapsat jeho hodnotu cukernatosti. Zadání této hodnoty je kontrolováno a porovnáno s aktuálně zvoleným typem piva. Pokud zadaná cukernatost nespadá do obvyklých hodnot pro zvolený typ piva, je uživatel informován o neobvyklé kombinaci a je vyzván o dodatečné potvrzení správnosti dat. Nakonec je možné k pohybu přidat libovolnou poznámku informativního charakteru.

Další režim přenosu piva je jednoduchý přesun mezi nádrži na pivo. Pokud uživatel po zvolení nového pohybu vybere jako zdroj přesunu nádrž s pivem, automaticky se zvýrazní pouze ty nádrže, které jsou buď prázdné, nebo obsahují stejný typ piva a nejsou plné. Uživatel dále vybere jednu z dostupných nádrží a vzápětí se stejně jako v předešlém režimu zobrazí dialogové okno s dalšími podrobnostmi o přesunu. V tomto případě se dialog liší v několika věcech. Přesněji se v tomto případě nevybírá číslo varu a cukernatost, a to z důvodu, že se nejedná o nově uvařené pivo. Dále typ piva je dán zdrojovou nádrží a v dialogu se zobrazuje pouze pro informativní účely, podobně jako zdrojová a cílová nádrž. Na obrázku číslo 19 je zachycen příklad dialogu pro přesun piva mezi nádržemi.



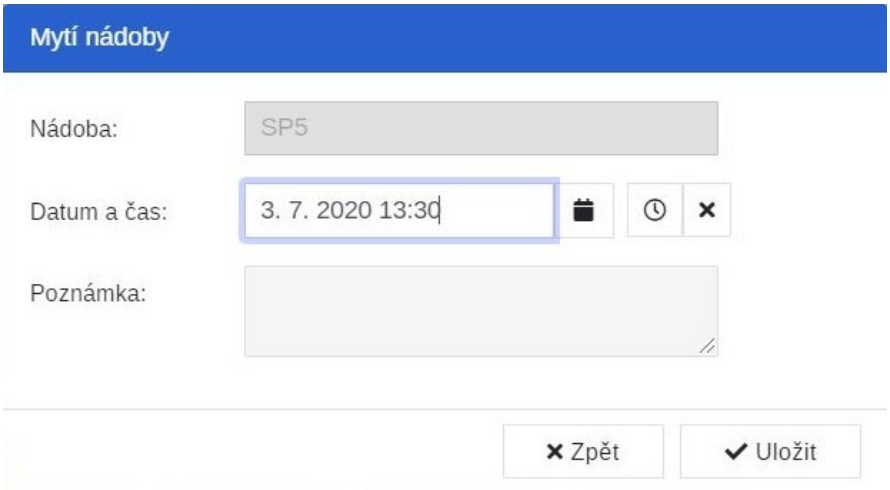
Obrázek 19: Dialog pro přesun piva mezi objekty. Zdroj: Vlastní

Posledním režimem by se dal nazvat přesun piva do sudů. Uživatel v tomto případě zvolí zdroj jako libovolnou nádrž s pivem a jako cílový slad zvolí právě objekt sudy. Stejně jako v předešlých případech se objeví dialogové okno s upřesňujícími parametry. V tomto případě dialog ovšem zobrazuje počty jednotlivých typů sudů. Uživatel tedy standardně vyplní množství piva k přesunu a na základě toho se automaticky dopočítá a zobrazí počty jednotlivých typů sudů, které se naplní daným množstvím piva.

Po doplnění všech informací v dialogovém oknu uživatel odešle požadavek. Po provedení několika nezbytných kontrol a ověření se požadavek odešle ke zpracování a následně se zobrazí upozornění uživateli o výsledném stavu požadavku. Pokud je požadavek úspěšně přijat, dojde k aktualizaci dat a zobrazení příslušné pozitivní zprávy. Pokud se žádost o požadavek nezdaří, dojde k informování uživatele o případném problému, kvůli kterému nebylo možné akci provést.

4.2.2 Mytí nádoby

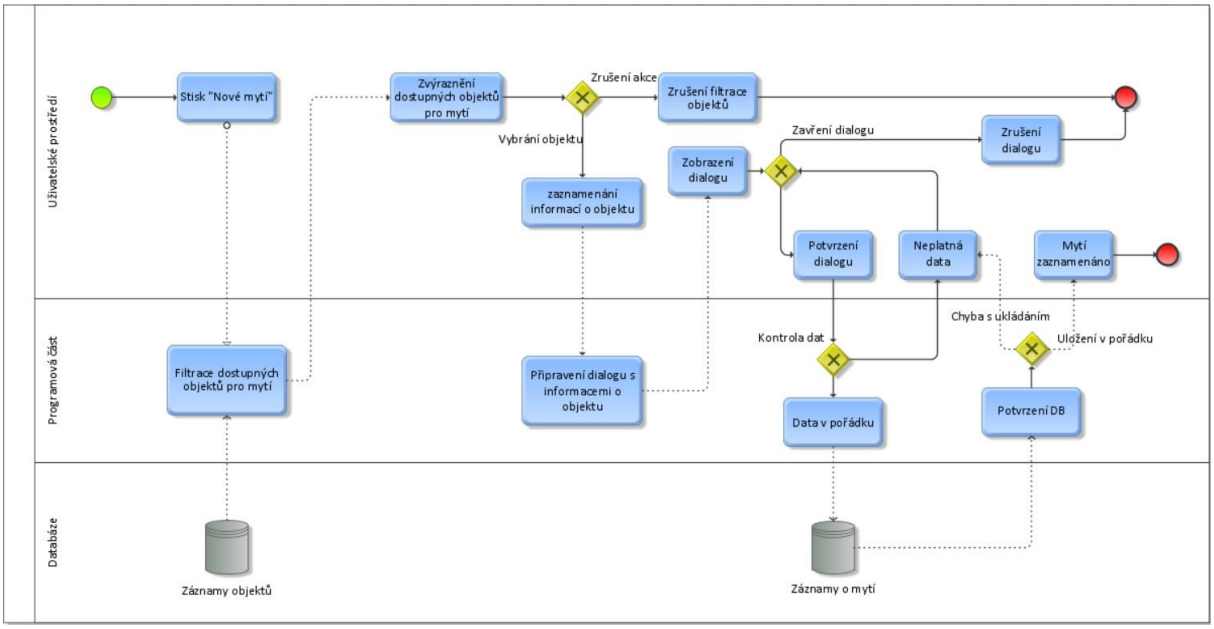
Další akcí, kterou lze provést, je mytí nádoby. Jedná se o možnost evidovat mytí jednotlivých nádob v pivovaru. Akce je jednoduchá a velmi intuitivní na použití. Po stisku tlačítka mytí nádoby, se zvýrazní pouze objekty, nad kterými je možné danou akci mytí nádoby provést. Uživatel je dále vyzván ke zvolení jednoho z těchto objektů. Po zvolení se otevře dialogové okno, které obsahuje parametry pro upřesnění požadavku, jak je možné vidět na obrázku číslo 20.



Obrázek 20: Dialog pro nové mytí nádoby. Zdroj: Vlastní

První povinný parametr dialogu je zvolená nádoba pro mytí. Tento parametr je předvyplněn, a je pouze informačního charakteru. Tento parametr v tuto chvíli již nelze upravovat. Následujícím parametrem je datum a čas mytí nádoby. Zde je datum a čas předvyplněn aktuálním časem a uživatel má možnost tuto hodnotu libovolně změnit. V poslední řadě je možné přidat k novému mytí nádoby vlastní poznámku.

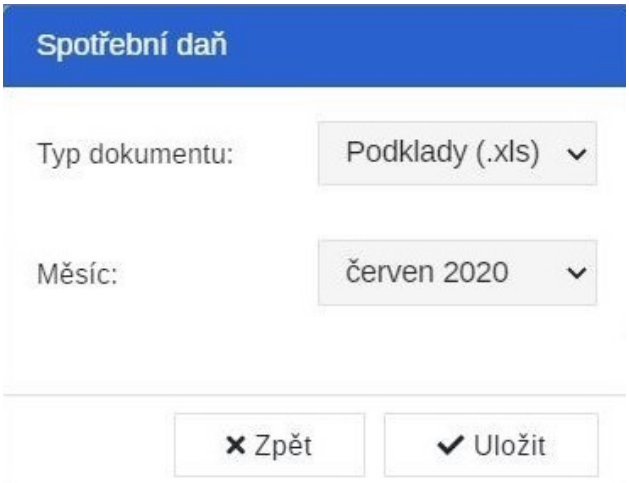
Po vyplnění a zkontrolování parametrů uživatel potvrdí požadavek. Následně je požadavek zkontrolován a odeslán ke zpracování. Při úspěšném zpracování požadavku dojde k obnovení vyobrazených dat a uživatel je informován i úspěchu. Při selhání požadavku je uživatelovi zobrazena zpráva o problému, kvůli kterému nebylo možné požadavek úspěšně provést. Celý průběh procesů je pro lepší představu také zachycen níže na obrázku číslo 21.



Obrázek 21: Diagram procesů mytí nádoby. Zdroj: Vlastní

4.2.3 Spotřební daň

Tato akce slouží ke stažení veškerých potřebných dokumentů. Po stisku daného tlačítka se uživateli zobrazí dialogové okno s možností výběru typu dokumentu a měsíce, pro který daný dokument potřebuje. Typy dokumentů se evidují dva typy. První typ jsou „Podklady“, ty se stahují ve formátu Microsoft Excel s příponou xls. Druhým dostupným typem dokumentu pro stažení je „Přiznání“. Tyto dokumenty se stahují ve formátu PDF. Níže je vidět příklad předvyplněného dialogu.



Obrázek 22: Dialog pro zobrazení a stažení dokumentů. Zdroj: Vlastní

4.2.4 Tisk dat

Pro upravení stránky pro tisk bylo potřeba vybrat a zobrazit pouze určité údaje, které jsou pro tisk důležité. Znamenalo to především vytvoření celého souboru stylů pro úpravu stránky při tisku. Pokud je tedy požadován tisk stránky, aplikuje se alternativní soubor pro styly stránky. Toho bylo docíleno především díky CSS pravidla @media, za které poté stačilo přidat klíčové slovo „print“. Viz dále na obrázku číslo 23. Díky tomu se styly obsažené v tomto soubory vyberou jako prioritní vždy při tisku. Bylo nutné zbavit se veškerých pokročilých grafických prvků a vytvořit alternativní vzhled objektů, které se hodí pro tisk. Místo graficky znázorněných objektů se tedy zobrazují pouze informace v textové podobě. Veškeré informace na výsledné tiskové stránce jsou černobílé a informace o daném objektu jsou od sebe vždy odděleny viditelnou mezerou. Řádky poté odděluje přerušovaná čára pro větší přehlednost.

```
@media print {

.row{
    font-family: 'Courier New', Courier, monospace;
}
.row div{
    height: 100px;
    padding-top: 20px;
    padding-bottom: 10px;
    text-align: center;
    text-transform: uppercase;
    border-bottom: 1px dashed black;
}
```

Obrázek 23: Ukázka stylů pro tisk. Zdroj: Vlastní

4.3 Objekty pivovaru

Největší část grafického rozhraní zabírají samotné objekty pivovaru. Jedná se o nejdůležitější vizuální část celého projektu. Po zvážení všech požadavků byla použita responzivní struktura, která umožňuje vkládání objektů. Díky této vlastnosti je snadné stránku upravit pro zobrazení na různých zařízeních. Na obrázku číslo 24 je vidět začátek struktury, která obsahuje jednotlivé objekty pivovaru. Jednotlivé objekty jsou dále zapouzdřeny do struktury „inmod-brew-object“, která komunikuje skrze parametry, které jsou také zachyceny na následujícím obrázku.

```
<div *ngIf="tanksAndFermentationsData">

    <div class="row">
        <div class="col-lg-1 col-md-2 col-4 d-print-none">
            <inmod-brew-object [data]="{Code: 'Varna'}"
            [paramPohyb]="novyPohyb" [paramMyti]="myti" [paramFrom]="from"
            [paramFromObject]="fromObject" (click)="onClickOject('Varna')">
            </inmod-brew-object>
        </div>
        <div class="col-lg-1 col-md-2 col-4 d-none d-lg-block">
        </div>
        <div class="col-lg-1 col-md-2 col-4">
            <inmod-brew-object [data]="getItem('SP1')"
            [paramPohyb]="novyPohyb" [paramMyti]="myti"[paramMyti]="myti"
            [paramFrom]="from" [paramFromObject]="fromObject" (click)="onClickOject('SP1')">
            </inmod-brew-object>
        </div>
        <div class="col-lg-1 col-md-2 col-4">
            <inmod-brew-object [data]="getItem('SP2')"
                [paramPohyb]="novyPohyb" [paramMyti]="myti"[paramMyti]="myti"
            [paramFrom]="from" [paramFromObject]="fromObject" (click)="onClickOject('SP2')">
            </inmod-brew-object>
        </div>
```

Obrázek 24: Ukázka struktury kódu pro objekty pivovaru. Zdroj: Vlastní

Samotný objekt je tedy představován strukturou, která je snadno rozšiřitelná a není tedy problém objekty přidávat, popřípadě odebírat dle potřeb zákazníků. V tomto případě byl podle potřeb pivovaru přidán na první místo objekt varny piva. Ten slouží pro vytvoření nového varu piva. Na stejné rovině jsou vytvořeny objekty pro uložení nových varů piva, takzvané „Spilky“. Tyto objekty jsou dále rozdělené podle velikostí na malé a velké.

Ve spodní řadě v pravé části obrazovky se nachází dva speciální objekty. Tyto objekty jsou zachyceny níže, na obrázku číslo 25. První z nich slouží pro možnost přesunutí piva do sudů. Druhý objekt poté slouží pro přesun piva mimo objekt pivovaru. Zde je vidět možnost přesunu piva do restaurace. Zároveň je zde ukazatel množství piva v tomto externím objektu.

Obrázek 25: Příklad speciálních objektů pivovaru. Zdroj: Vlastní

Při standardním zobrazení se zobrazují objekty ve třech řádcích. První řádek obsahuje na prvním místě vždy varnu. Následuje několik spilek jako dočasné nádoby na pivo. Druhý a třetí řádek obsahují především nádrže, se kterými je možné standardně interagovat. Tyto nádrže slouží pro dlouhodobější uskladnění piva. Je možné nad nimi provádět zejména akce typu umytí nádoby a přesunu piva. Poslední řádek dále obsahuje jeden specifický objekt, díky kterému se pivo pomocí nového pohybu přesouvá do sudů.

Jednotlivé nádrže na pivo dále obsahují veškeré potřebné popisky, jako například název nádrže, aktuální typ a množství piva, popřípadě datum posledního mytí nádrže. Podrobnější informace lze zobrazit při najetí kurzoru myši na objekt, popřípadě stiskem nádrže při používání dotykového zařízení. Tato akce vyvolá okno typu nápovědy, ve kterém se zobrazí veškeré podrobnější informace o nádrži. Toto okno je zachyceno níže na obrázku. Mezi přednosti grafického rozhraní patří také skutečnost, že jednotlivé nádrže opravdu simulují množství piva, a to se graficky zobrazuje v reálném čase. Jednotlivé druhy piv mají také přiřazenou svou barvu a díky tomu se pivo v nádržích vykresluje podle této vlastnosti.

Obrázek 26: Detail nádrže s popiskem. Zdroj: Vlastní

4.4 Nasazení modulu do provozu

Po realizaci všech komponent a naplnění požadavků klienta bylo potřeba modul dále otestovat. Testování je fáze, kterou musí projít většina aplikací. Při testování bylo potřeba simulovat reálný provoz pivovaru a opakovaně testovat veškeré funkce. Testování probíhalo na několika fyzických zařízeních, aby se zajistily co nejpřesnější výsledky. Bylo také potřeba otestovat zobrazení na obrazovkách s různou velikostí a rozlišením. Mezi časté nedostatky, které se při takovémto testování odhalí, jsou například drobné změny v chování aplikace v jiném internetovém prohlížeči. Po odhalení všech nalezených nedostatků došlo k jejich opravě a modul byl připraven na nasazení do reálného provozu.

V tomto případě byl modul, na základě přání zákazníka, přidán do již používaného intranetu. Tento intranet funguje na serveru, který vlastní daná provozovna. Intranet je ovšem možné nabízet jako službu formou hostovanou vzdáleně. Zákazník poté nemusí vlastnit žádný server pro běh intranetu.

Moduly, a to včetně pivovaru, jsou navrženy pro samostatný chod. Díky architektuře, která byla použita při vývoji, je možné modul pivovaru využívat bez nutnosti celého intranetu. Celý modul se poté chová jako knihovna, která se dá snadno nainstalovat. K tomu se využívá NPM (Node Package Manager). NPM je správce balíčků a s patřičnou licencí, je díky němu snadné modul nainstalovat a využívat kdekoliv.

Konkrétní provozovna, pro kterou byl modul primárně vytvořen, již určitý čas modul pivovaru využívá. Po krátké době byla k dispozici také zpětná vazba uživatelů modulu. Na jejím základě se v modulu pivovaru provedly drobné úpravy, jako zvětšení písma na určitých místech nebo úprava zobrazení potvrzení o dokončení akce.

Díky struktuře modulu je možné provádět úpravy bez větších obtíží pro uživatele, který takovou změnu verze často ani nezjistí. Změna verze se ovšem vždy předem ohlašuje a upozorňuje se na ni, aby nedošlo k nežádaným nedorozuměním.

ZÁVĚR

Práce se zabývala vývojem uživatelské nadstavby pro modul pivovaru v rámci intranetu. V první části je popsán systém, do kterého je práce zasazena, spolu s porovnáním s jiným systémem. V práci jsou následně popsány aktuálně využívané nástroje, které přispívají k efektivnějšímu vývoji podobných typů aplikací. Dále jsou podrobně popsány technologie, využívané při tvorbě aplikace.

Následující část práce byla věnována samotnému návrhu a realizaci modulu. Na začátku bylo potřeba rozebrat veškeré požadavky a vytvořit návrhy jednotlivých komponent. Tyto návrhy byly následně schváleny. Na základě těchto návrhů bylo dále vymodelováno grafické uživatelské rozhraní. Toto rozhraní obsahovalo funkční prvky, které jsou v práci podrobně popsány a dále implementovány.

Modul pivovaru byl určitou dobu testován a po celkové kontrole byl nasazen do intranetu k fungování v reálném provozu. Na základě zpětné vazby od klienta proběhly v modulu pivovaru dále drobné úpravy, aby lépe vyhovovaly koncovému uživateli. Díky struktuře rozhraní je možné dále pivovar snadno upravovat dle potřeb dalších potenciálních klientů.

Jak bylo z počátku předpovídáno, tvorba modulu pivovaru se lišila od jiných, a to především díky bohatému grafickému rozhraní. Bylo zde nutné vytvořit plochu složenou z interaktivních objektů, které musí reagovat na vstupy uživatele, a to v reálném čase. Objekty se také dokážou přizpůsobit rozměrům obrazovky a je tedy možné využívat pro obsluhu například dotykový display tabletu.

Při tvorbě tohoto unikátního modulu autor také pochopil, že uživatelské grafické rozhraní je většinou to jediné, co koncový uživatel vidí nebo chápe. Na základě toho je nutné tuto část aplikace nepodceňovat, ale právě naopak je potřeba se dokázat vžít do někoho, kdo danou aplikaci bude využívat. Díky tomu se aplikace stanou uživatelsky přívětivějšími a snazšími na obsluhu.

POUŽITÁ LITERATURA

[1] Bootstrap 4 Get Started. *W3schools* [online]. Norwegia: Refsnes Data, 1998 [cit. 2020-07-10]. Dostupné z: https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp

[2] DREIMANIS, Gints a Olga BOLGURTSEVA. Why You Should Choose TypeScript Over JavaScript. *Serokell* [online]. 18. 6. 2020 [cit. 2020-07-11]. Dostupné z: https://serokell.io/blog/why-typescript

[3] RYABTSEV, Alexander. Web Frameworks: How To Get Started. *Django Stars* [online]. 2008, 11. 1. 2017 [cit. 2020-07-11]. Dostupné z: https://www.goodfirms.co/glossary/web-framework/

[4] MORRIS, Scott. WHAT IS A JAVASCRIPT FRAMEWORK? HERE'S EVERYTHING YOU NEED TO KNOW. *Skillcrush* [online]. c2012 - 2020 [cit. 2020-07-11]. Dostupné z: https://skillcrush.com/blog/what-is-a-javascript-framework/

[5] MÁCA, Jindřich. Lekce 1 - Úvod do Angular frameworku. *ITnetwork.cz* [online]. c2020 [cit. 2020-07-11]. Dostupné z: https://www.itnetwork.cz/javascript/angular/zaklady/uvod-do-angular-frameworku

[6] Introduction to Angular concepts. *Angular* [online]. c2010-2020 [cit. 2020-07-11]. Dostupné z: https://angular.io/guide/architecture

[7] GAVIGAN, Dave. The History of Angular. *Medium* [online]. c2020, 3. 4. 2018 [cit. 2020-07-11]. Dostupné z: https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7

[8] Introduction. *Vue.js* [online]. c2014-2020 [cit. 2020-07-11]. Dostupné z: https://vuejs.org/v2/guide/index.html#What-is-Vue-js

[9] KRUPIČKA, David. #1 Vue.js český návod zdarma. *StarkMedia* [online]. c2020, 26. 01. 2019 [cit. 2020-07-11]. Dostupné z: https://starkmedia.cz/blog/vue-js-zdarma-kurz-navod-cesky

[10] PANDIT, Nitin. What and Why React.js. *C# Corner* [online]. c2020, 5. 3. 2020 [cit. 2020-07-11]. Dostupné z: https://www.c-sharpcorner.com/article/what-and-why-reactjs/

[11] SUFIYAN, Taha. What is React? *Simplilearn Solution* [online]. c2009-2020, 10. 3. 2020 [cit. 2020-07-11]. Dostupné z: https://www.simplilearn.com/what-is-react-article

[12] ROUSE, Margaret, Kevin FERGUSON a Cameron MCKENZIE. REST (REpresentational State Transfer). *TechTarget* [online]. c2020 [cit. 2020-07-11]. Dostupné z: https://searchapparchitecture.techtarget.com/definition/REST-REpresentational-State-Transfer

[13] RESTful Web Services Introduction. *Tutorialspoint* [online]. c2020 [cit. 2020-07-11]. Dostupné z: https://www.tutorialspoint.com/restful/restful_introduction.htm

[14] MDN [Mozilla Developer Network]. HTTP Messages. *MDN web docs* [online]. c2005-2020, 4. 7. 2020 [cit. 2020-07-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Messages

[15] MDN [Mozilla Developer Network]. HTTP response status codes. *MDN web docs* [online]. c2005-2020, 7. 7. 2020 [cit. 2020-07-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

[16] RICHARDSON, Leonard, Michael AMUNDSEN a Sam RUBY. RESTful Web APIs. Sebastopol: O'Reilly, 2013. ISBN 978-1-4493-5806-8.

[17] PAUL, Javin. What are Idempotent and Safe methods of HTTP and REST. *Javarevisited* [online]. c2010-2018, 31. 5. 2016 [cit. 2020-07-13]. Dostupné z: https://javarevisited.blogspot.com/2016/05/what-are-idempotent-and-safe-methods-of-HTTP-and-REST.html

[18] HTTP headers. *MDN web docs* [online]. c2005-2020, 27. 4. 2020 [cit. 2020-07-13]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers

[19] RAIBLE, Matt. What the Heck is OAuth? *Okta* [online]. c2020, 21. 6. 2017 [cit. 2020-07-17]. Dostupné z: https://developer.okta.com/blog/2017/06/21/what-the-heck-is-oauth

[20] ROUSE, Margaret. OAuth. *TechTarget* [online]. Newton (Massachusetts), 1999, únor 2020 [cit. 2020-07-17]. Dostupné z: https://searchapparchitecture.techtarget.com/definition/OAuth

[21] BIHIS, Charles. Mastering OAuth 2.0. Birmingham: Packt Publishing, 2015. ISBN 978-1-78439-540-7.

[22] ANICAS, Mitchell. An Introduction to OAuth 2. *DigitalOcean* [online]. New York, 2011, 21. 7. 2014 [cit. 2020-07-17]. Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

[23] COPES, Flavio. JSON Web Token (JWT) explained: Learn the basics of JWT and how to use them. *FlavioCopes* [online]. c2020, 13. 12. 2018 [cit. 2020-07-17]. Dostupné z: https://flaviocopes.com/jwt/

[24] ŠÍPEK, Robert. JSON Web Tokens (JWT). *Zooom.cz* [online]. [2015], 20. 4. 2019 [cit. 2020-07-17]. Dostupné z: https://zooom.cz/json-web-tokens-jwt/

[25] CHOI, Kenneth. Stateful and stateless authentication. *Medium* [online]. 2012, 14. 3. 2018 [cit. 2020-07-17]. Dostupné z: https://medium.com/@kennch/stateful-and-stateless-authentication-10aa3e3d4986

[26] BRENNENSTUHL, Jan. The Purpose of JWT: Stateless Authentication. *Zalando* [online]. Německo, 2008, 26. 7. 2017 [cit. 2020-07-17]. Dostupné z: https://engineering.zalando.com/posts/2017/07/the-purpose-of-jwt-stateless-authentication.html

[27] KALANSURIYA, Prathap. What Is a JWT Token? *DZone* [online]. 2005, 24. 1. 2020 [cit. 2020-07-17]. Dostupné z: https://dzone.com/articles/what-is-jwt-token

[28] What is a Warehouse Management System? *ClarusWMS* [online]. Anglie, 2016, 18. 9. 2018 [cit. 2020-07-17]. Dostupné z: https://www.claruswms.co.uk/what-is-warehouse-management-system/

[29] *Informační systém KARAT* [online]. c2006 [cit. 2020-07-20]. Dostupné z: https://www.karatsoftware.cz/

[30] MASSÉ, Mark. *REST API design rulebook*. Sebastopol: O'Reilly, c2012, s. 2-6. ISBN 978-1-449-31050-9.

[31] NAEEM, Tehreem. REST APIs: Definition, Working, Benefits, and Design Principles. *Asteria* [online]. 2009, 24. 6. 2020 [cit. 2020-07-23]. Dostupné z: https://www.astera.com/type/blog/rest-api-definition/

[32] KOLOMIIETS, Ksenija. 9 Reasons that will convince you to redevelop your Desktop app to a cloud-based application. *GBKSOFT* [online]. Ukrajina, 2011, 10. 6. 2020 [cit. 2020-07-03]. Dostupné z: https://gbksoft.com/blog/convert-desktop-app-to-web-app/

[33] FIELDING, Roy Thomas. *Architectural Styles and the Design of Network-based Software Architectures*. Irvine, 2000. Disertační práce. Kalifornská univerzita.