**Česká zemědělská univerzita v Praze**

**Technická fakulta**

**Katedra technologických zařízení staveb**

# Diplomová práce

## Webová aplikace pro evidenci chovného dobytka

**Bc. Jan Kott**

# ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Technická fakulta

# ZADÁNÍ DIPLOMOVÉ PRÁCE

Bc. Jan Kott

Informační a řídicí technika v agropotravinářském komplexu

Název práce

**Webová aplikace pro evidenci chovného dobytka**

Název anglicky

**Web application for registration of breeding cattle**

---

**Cíle práce**

Cílem práce je analyzovat a na základě provedeného rozboru realizovat aplikaci pro evidenci dobytka založené na webovém prostředí.
V průběhu realizace je potřeba definovat pojmy a dostupné technologie vztažené k vlastní aplikaci. Dále popsat zvolený přístup z pohledu architektury a bezpečnosti. Uvést problematiku systému řízení báze dat. V praktické části práce vytvořit návrh systému. Na základě návrhu vytvořit vlastní aplikační výstup, na kterém budou následně provedeny aplikační testy. Na závěr uvést aplikaci do produkčního režimu.

**Metodika**

1. Úvod
2. Cíl práce
3. Metodika práce
4. Webová aplikace
5. Technologie pro tvorbu webových aplikací
6. Vývoj webových aplikací z hlediska architektury a přístupu
7. Systém řízení báze dat
8. Návrh praktického řešení a realizace
9. Závěr

**Doporučený rozsah práce**

50 – 60

**Klíčová slova**

webová aplikace, evidenční systém, objektově orientovaná architektura

---

**Doporučené zdroje informací**

BÖHMER, M., Návrhové vzory v PHP, Computer Press, ISBN 9788025133385
CASTRO, E., HYSLOP, B., HTML5 a CSS3, COMPUTER PRESS, ISBN 9788025137338
THOMSON, L., WELLING, L., Mistrovství – PHP a MySQL, Computer Press, 2017, ISBN 9788025148921
ZAKAS, N., Listování JavaScript pro webové vývojáře, COMPUTER PRESS, 2009, ISBN 9788025125090

---

**Předběžný termín obhajoby**

2021/2022 LS – TF

**Vedoucí práce**

Ing. Jan Lešetický, Ph.D.

**Garantující pracoviště**

Katedra technologických zařízení staveb

**Konzultant**

Ing. Marek Pačes

Elektronicky schváleno dne 3. 2. 2021

**doc. Ing. Jan Malaťák, Ph.D.**

Vedoucí katedry

Elektronicky schváleno dne 10. 2. 2021

**doc. Ing. Jiří Mašek, Ph.D.**

Děkan

V Praze dne 22. 09. 2021

**Čestné prohlášení**

Prohlašuji, že svou diplomovou práci Webová aplikace pro evidenci chovného dobytka jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31.3.2022 _____

**Poděkování**

Rád bych touto cestou poděkoval Ing. Janu Lešetickému, Ph.D. a Ing. Marku Pačesovi za poskytnutý čas a cenné rady při psaní diplomové práce.

# Webová aplikace pro evidenci chovného dobytka


## Abstrakt

Diplomová práce řeší vývoj webových aplikací. Cílem bylo vytvořit webovou aplikaci pro evidenci chovného dobytka na základě provedené analýzy. Dále uvést čtenáře do problematiky webových aplikací a jejich vývoje. Teoretická část diplomové práce byla vypracována na základě poznatků získaných z odborné literatury a ostatních informačních zdrojů. Praktická část byla věnována vývoji webové aplikace pro evidenci chovného dobytka. Proces vývoje webové aplikace zahrnoval analýzu, průzkum trhu, definováni požadavků, návrh, implementaci, testování a nasazení. Výsledkem diplomové práce bylo vyhotovení webové aplikace dle vytyčených požadavků a vytvoření přehledu o vývoji webových aplikací. V závěru jsou uvedeny veškeré dosažené výsledky a směr, kterým by se mohla webová aplikace dále ubírat.



**Klíčová slova:** webová aplikace, evidenční systém, chovný dobytek, react, php, cloud computing

# Web application for registration of breeding cattle

**Abstract**

This diploma thesis deals with development of web applications. The aim was to create a web application that encapsulates registry of lifestock based on the performed analysis. Furthermore, to introduce the reader to the issue of web applications and their development. The theoretical part of the diploma thesis was prepared on the basis of knowledge obtained from professional literature and other information sources. The practical part was devoted to the development of the actual web application for livestock registry. The web application development process included analysis, market research, requirements definition, design, implementation, testing, and deployment. The result of the diploma thesis was the creation of a web application according to the specified requirements and the creation of an overview of the development of web applications. In conclusion, all the achieved results and the direction in which the web application could be developed further are listed.

**Keywords**: web application, registration system, breeding cattle, react, php, cloud computing

# Obsah

# Seznam obrázků

# Seznam tabulek



# Seznam zdrojových kódů

# 1 Úvod

S rostoucí digitalizací napříč všemi obory lidské činnosti došlo k rozvoji využívání informačních technologií i v zemědělství. Informační technologie v zemědělství mají za cíl především usnadnit a zefektivnit práci zemědělců.

Dnes již prakticky všichni zemědělci alespoň částečně ke své činnosti některé informační technologie využívají. Ať už se jedná o ty méně sofistikované technologie, jakými jsou například meteostanice a RFID čipy. Až po vysoce sofistikované technologie, mezi které patří družicové snímky, navigační a autonomní systémy. Do budoucna lze očekávat další rozvoj v informačních technologií a užší integraci těchto technologií v zemědělství.

Prostřednictvím diplomové práce bych chtěl přispět k digitalizaci zemědělské činnosti v oboru živočišné výroby. Vytvořením webové aplikace umožňující zemědělskému subjektu vést online evidenci chovného dobytka. Vytvořená webová aplikace by měla být vhodnou bezplatnou alternativou k ručně vedené evidenci nebo k desktopovým aplikacím pro evidenci dobytka. Dále bych chtěl uvést čtenáře do problematiky vývoje moderních webových aplikací. Definovat technologie a používané přístupy pro tvorbu webových aplikací. Seznámit čtenáře s možnostmi ukládáni dat ve webových aplikacích pomocí systémů řízení báze dat.

# 2 Cíl práce a metodika

## 2.1 Cíl práce

Cílem diplomové práce je vytvořit moderní webovou aplikaci umožňující vést bezplatnou evidenci chovného dobytka. Dále uvést čtenáře do problematiky webových aplikací a jejich vývoje. Definovat pojmy a technologie vztažené k vývoji webové aplikace. V praktické části pak popsat jednotlivé kroky vývoje webové aplikace pro evidenci chovného dobytka. A na závěr webovou aplikaci spustit v produkčním režimu.

## 2.2 Metodika

Pro zpracování teoretické časti diplomové práce bylo využíváno odborných publikací i vlastních poznatků o tvorbě webových aplikací. Z těchto zdrojů byl vytvořen základní přehled o problematice webových aplikací a jejich vývoji.

V praktické části diplomové práce byly využívány standardní metody tvorby softwaru. Zahrnující analýzu, průzkum trhu, definováni požadavků, návrh, implementaci, testování a nasazení.

# 3 Webová aplikace

Webová aplikace je aplikační software, který je poskytován uživatelům z webového serveru. Uživatel (klient) komunikuje s webovým serverem přes počítačovou síť Internet, případně přes Intranet (vnitropodniková síť). Na rozdíl od desktopového softwaru není aplikace přítomna lokálně, tedy nahrána na disku počítače uživatele. Přístup k aplikaci je realizován skrze webový prohlížeč. Webový prohlížeč umožňuje uživateli webovou aplikaci obsluhovat. Komunikace je založena na síťové architektuře Klient – Server. Klient reprezentuje počítač uživatele webové aplikace. Ten přes webový prohlížeč zasílá požadavky na webový server. Webový server reprezentuje Server v architektuře Klient-Server a vrací klientovy odpovědi (data). [1, 2, 3, 46, 47]

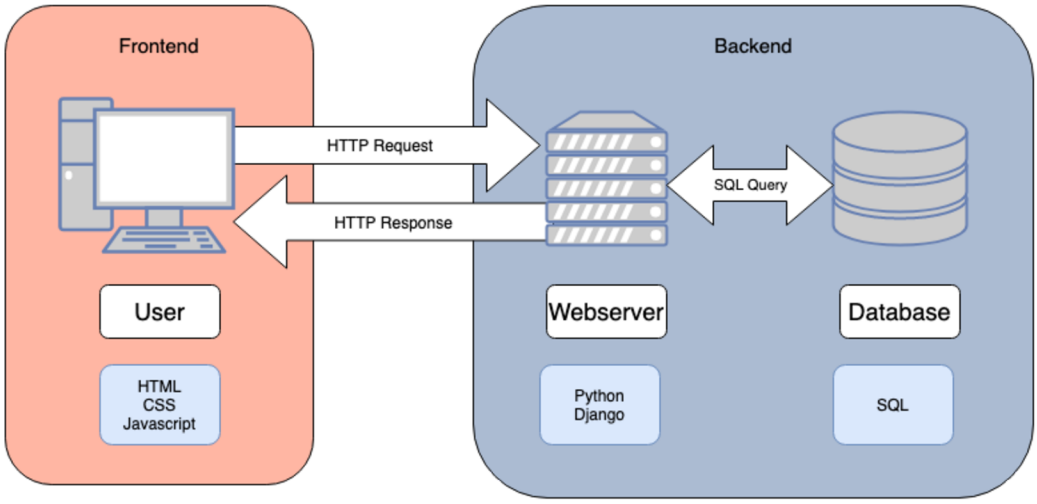
## 3.1 Front-end

Ve vývoji webových aplikací je rozlišována grafická část webové aplikace (front-end) od logické části webové aplikace (back-end). Výraz front-end je používán pro grafickou část webové aplikace. Grafická část aplikace je tvořena html kódem, který určuje strukturu aplikace. Dále CSS jenž doplňuje grafické prvky do aplikace. A také JavaScriptem, ten dodává webové aplikaci dynamické chování. Zjednodušeně lze říci, že front-end webové aplikace tvoří ta část aplikace, která je visuálně vidět. Hlavními cíli front-endu je vytvořit stránku visuálně atraktivní a přehlednou. Zajistit rychlé vykreslování a kompatibilitu napříč jednotlivými zařízeními: chytrý telefon, počítač a další. V posledních letech dochází rozvojem javascriptových frameworků k přesahům front-endové části do back-endové části. Například při využívání architektury REST API. [4]

## 3.2 Back-end

Oproti front-endu je back-end, ta část webové aplikace, kterou jako uživatelé nevidíme. Jedná se o část služeb, které jsou spouštěny na serveru (viz obr. 1). Typicky je to programovací jazyk na straně serveru například: PHP, Ruby, Java, Python. Dále pak nějaký systém řízení báze dat například MySQL nebo PostgreSQL. Cílem back-endu je zpracovávat data, aby na straně klienta (front-endu) vše správně fungovalo. [4]

*Obrázek 1 Schéma frontend a backend*




*Zdroj: https://xyzcoding.com/course/the-internet/how-the-internet-works/front-end-vs-back-end/*

## 3.3 Historie

Historie webových aplikací se skládá z několika významných milníků, zmíněné milníky v kapitole historie zformovali webové aplikace do podoby, v jaké je známe dnes. Začátek éry webových aplikací odstartovalo vydání jazyku JavaScript firmou NetScape Communications v roce 1995. JavaScript umožnil vývojářům přidávat dynamické prvky do webových stránek a tím zlepšit celkovou odezvu stránky. [5,6]

V roce 1996 společnost Macromedia představila Flash. Jednalo se o revoluční technologii, která výrazně zvýšila interaktivitu webových stránek. Pomocí technologie Flash byly do stránek vkládány vektorové animace, hry, vektorová grafika, poslech hudby a sledovaní videa. V roce 1999 se koncept webových aplikací objevil také v jazyce Java. [5,6]

Dalším milníkem ve vývoji webových aplikací byl rok 2005, kdy byla představena technologie AJAX. Technologie AJAX umožnila komplexní řešení pro asynchronní volání serveru a začaly tak vznikat první asynchronní webové aplikace. První prohlížeč, který tuto technologii podporoval byl Internet Explorer později se přidaly další prohlížeče Opera, Mozilla a Safari. Dále technologii využívaly i první webové aplikace Gmail a Google Maps. [5,6]

Další vývojovou fází webových aplikací bylo uvedení specifikace HTML 5, jejíž finální specifikace byla vydána v roce 2014. Specifikace HTML 5 podporuje přehrávání medií bez přídavných pluginů v prohlížeči a podporu aplikací fungujících i bez připojení k Internetu. [5,6]

## 3.4 Současnost

V současnosti se oblast webových aplikací velmi rychle vyvíjí. Objevují se neustále nové technologie, se kterými musí vývojáři webových aplikací držet krok. Například donedávna používaná technologie Flash, která byla plně nahrazena HTML 5. Dále se ve vývoji webových aplikací objevují nové programovací jazyky, frameworky a přístupy k návrhu. Spousta služeb, které byly dříve dostupné jako desktopové či mobilní aplikace se dnes přesouvá do webového prostředí. Do budoucna bude trend v popularitě webových aplikací nadále stoupat s očekávaným příchodem Webu 3.0. [1,3]

## 3.5 Rozdíl mezi webovou stránkou a webovou aplikací

Webová stránka je soubor dat zobrazován uživateli ve webovém prohlížeči. Data jsou nejčastěji ve formě textu, obrázků a videí. Data jsou zobrazena na jedné či více stránkách. Tyto znaky platí i pro webovou aplikaci. Rozdíly spočívají především v samotném fungování. Prvním a největším rozdílem je interaktivita. Webové stránky umí obsah stránky pouze vizualizovat. Webové aplikace umožňují dynamické chování, uživateli je umožněno část obsahu dynamicky měnit. Dalším rozdílem je zobrazování obsahu jednotlivým uživatelům. Webové stránky zobrazují stejný obsah všem uživatelům bez rozdílu. Zatímco webové aplikace umožňují obsah filtrovat na základě oprávnění uživatele. Oprávnění uživatele – autentizace a s ní spojená autorizace je přidanou hodnotou webových aplikací. [7, 8, 1]

Z výše uvedených rozdílů je patrné, že webové aplikace jsou více komplexní, na rozdíl od webových stránek. Webové stránky jsou určeny jenom pro prezentaci dat. Například: články, blogy, profesní stránky, odborné publikace. Zatímco co webové aplikace jsou určeny jak pro

prezentaci dat, tak i pro editaci, správu a tvoření. Příklady webových aplikací: Gmail, Facebook, Amazon, Seznam. [7, 8, 1]

## 3.6 Bezpečnost

Webové aplikace se často stávají terčem útoků. Útočníci mohou být jak jednotlivci, tak organizované skupiny. V informatice je útočník označován termínem hacker. Hackeři si za své cíle vybírají webové aplikace nejčastěji ze dvou důvodů. Prvním je získání citlivých dat, která webové aplikace může obsahovat. Dále pak za účelem sabotáže celé aplikace čili jejího odstavení z provozu. S rostoucí popularitou webových aplikací a počtem uživatelů aplikací se webové aplikace stávají čím dál tím větším terčem útoků. Proto je na bezpečnost kladen větší důraz, než tomu bylo dříve. [9, 10]

### 3.6.1 OWASP

OWASP (Open Web Application Security Project) je v oblasti zabezpečení webových aplikací neziskový projekt, který mapuje bezpečností hrozby a snaží se zlepšovat zabezpečení webových aplikací. Mezi členy OWASPu patří největší odborníci v problematice bezpečnosti webových aplikací. Tito odborníci sestavují dokumentaci k bezpečnosti webových aplikací a vytvářejí nástroje ke kontrole zranitelností webových aplikací. [9, 10]

### 3.6.2 OWASP Top Ten

OWASP Top Ten je standardizovaný dokument vytvořen vývojáři a odborníky na bezpečnost webových aplikací z projektu OWASP. Tento dokument obsahuje žebříček deseti nejzávažnějších hrozeb za daný časový úsek. Většina hrozeb se opakuje jen jejich pořadí v žebříčku se mění. Pořadí, v jakém je hrozba umístěna ovlivňují následující faktory: frekvence s jakou se hrozba vyskytuje, závažnost hrozby a velikost jejich potenciálních dopadů. Pro rok 2021 byly zveřejněny následující rizika. [9, 10]

#### Broken Access Control

Broken Access Control je zranitelnost spojená se špatně nastavenými přístupovými právy uživatele. Uživatel díky nevhodnému nastavení přístupových práv získá možnost přístupu k citlivým datům. [9, 10]

### Cryptographic Failures

Tato zranitelnost vzniká při špatné implementaci kryptografických prostředků nebo jejich úplným vynecháním. Takového rizika může využít útočník, který tak získá citlivá data například z nezabezpečené komunikace. [9, 10]

### Injection

Zranitelnost na úrovní databázové vrstvy, kdy útočník podsune aplikaci škodlivý SQL příkaz. Škodlivý SQL příkaz pak může mít za následek únik citlivých informací nebo ztrátu dat. [9, 10]

### Insecure Design

Zranitelnost spočívá v chybném návrhu webové aplikace a nedostatkům v architektuře webové aplikace. [9, 10]

### Security Misconfiguration

Chybná konfigurace (Security Misconfiguration) je zranitelnost plynoucí z použití výchozí konfigurace, úniku systémových chybových hlášek a špatně nastavené hlavičky. [9, 10]

### Vulnerable and Outdated Components

Zranitelné a zastaralé komponenty (Vulnerable and Outdated Components) je zranitelnost vyplývající z používání zranitelných technologií a zastaralých komponent. U takových to komponent a technologií už byly v minulosti prokázané bezpečnostní mezery. Jako řešení těchto rizik je nutné používat aktuální a ověřené komponenty a technologie. [9, 10]

### Identification and Authentication Failures

Zranitelnost vyplývající z chybně implementované autentizace nebo nedostatečně implementované autentizace. Útočník zneužívající tuto zranitelnost tak získává přístup citlivým údajům. [9, 10]

### Software and Data Integrity Failures

Zranitelnost plynoucí z užívání služeb a dat ve webové aplikaci od neověřených poskytovatelů. Webové aplikace často využívají služby a data od neověřených poskytovatelů. Data a služby od takových poskytovatelů mohou být často zdrojem škodlivého kódu. [9, 10]

### Security Logging and Monitoring Failures

Security Logging and Monitoring Failures jsou bezpečnostní zranitelnosti spojené s nedostatečným sledováním (logováním) a zaznamenáváním pohybu uživatelů ve webové aplikaci. Útočník se díky těmto bezpečnostním rizikům může bezstarostně pohybovat po webové aplikaci a hledat její bezpečnostní mezery a útočit na ně bez vědomosti provozovatele webové aplikace. [9, 10]

### Server-Side Request Forgery

Server-Side Request Forgery je druh zranitelnosti, při kterém útočník zneužije funkcionalitu na serveru k čtení či změně dat z interních zdrojů aplikace nebo použije server jako prostředníka pro externí volání na jiné servery. [9, 10]

# 4 Technologie pro tvorbu webových aplikací

V kapitole Technologie pro tvorbu webových aplikací jsou definovány pojmy a technologie vztažené k vývoji vlastní webové aplikace.

## 4.1 HTML

Hypertext Markup Language zkráceně HTML, je značkovací jazyk pro tvorbu webových stránek a aplikací. Jedná se o základní stavební prvek webu, značky jazyka HTML, tvoří strukturu webové stránky. Samotný jazyk je zapisován do běžných textových dokumentů, které nativně končí koncovkou .html. [11, 12]

### 4.1.1 DOM

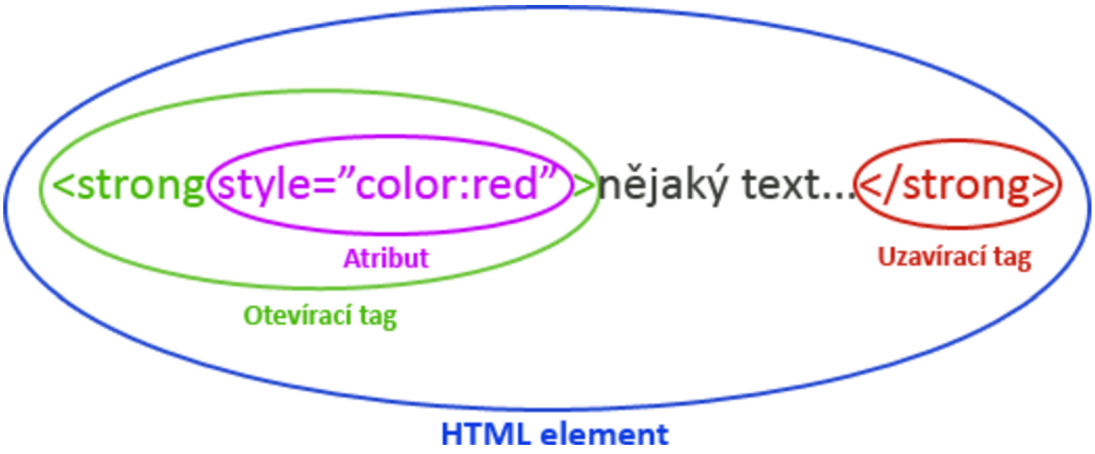
DOM (Document Object Model) je programové rozhraní (API) pro práci s HTML dokumentem. DOM reprezentuje dokument HTML jako datovou strukturu strom, kde každý uzel je objekt reprezentující část dokumentu. DOM umožňuje měnit strukturu, vzhled a obsah HTML dokumentu. Jazyk JavaScript využívá objektově orientovaných vlastností DOMU, což mu umožňuje dynamicky měnit strukturu, vzhled a obsah webové aplikace. [13, 14]

### 4.1.2 Značky v HTML

Značky v jazyce HTML definují, jak bude prohlížeč text uvnitř značek vykreslovat. Běžný dokument se skládá z několik značek, jenž definují jeho strukturu. Samotná značka je tvořena otevírací značkou tvořenou špičatými závorky. Dále obsahem ve formě prostého textu. A nakonec uzavírací značkou tvořenou špičatými závorky a lomítkem. V běžné praxi je otevírací značka nazývána otevírací tag a zavírací značka je nazývána uzavírací tag. Obsah obalený těmito závorky, nazýváme HTML element. [11, 12]

Na ukázce (viz obr. 2) je znázorněna syntaxe značky  . Značka strong převede běžný text na silný. Doplněna o atribut style.

*Obrázek 2 Ukázka syntaxe HTML elementu*




*Zdroj: https://www.klikzone.cz/HTML-navod/HTML-tagy.php*

V roce 2021 obsahoval jazyk HTML 132 značek. Značky jsou párové či samostatné. V případě párových značek je vždy přítomná ještě ukončující značka, ta je doplněna lomítkem před špičatou závorkou. [11, 12]

Na ukázce je zobrazena jednoduchá ukázka syntaxe jazyka HTML (viz obr. 3). První řádek HTML souboru obsahuje deklaraci dokumentu *DOCTYPE*, ten přiřazuje definici typu dokumentu. Na dalším řádku se nachází značka *html* označuje, kde začíná a končí celý dokument HTML.

*Obrázek 3 Ukázka syntaxe jazyka HTML*

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1> Toto je nadpis</h1>
    <p>Toto je odstavec</p>
</body>
</html>
```

*Zdroj: autor*

Dále značka *head*, ta je označením pro hlavičku souboru, do které se dále vkládají například *meta* značky a značka *title*. Značka *meta* obsahuje důležité informace například o kódování stránky, klíčových slovech a autorovi. Značka *title* definuje titulek stránky. Za značkou *head* následuje značka *body*, jedná se o značku obalující tělo dokumentu. Obsahuje všechen

zobrazovaný obsah. Značka *body* obaluje značky, které už tvoří konkrétní text na stránce. První značkou je značka h1, ta definuje textový nadpis. Dále značka p, která definuje odstavec textu.

### 4.1.3 HTML atributy

HTML atributy jsou klíčová slova deklarována v otevírací značce HTML elementu. Tyto slova modifikují výchozí chování HTML elementu. U některých elementů jsou atributy volitelné, tedy doplňují pouze přidané chování elementu. Dále atributy povinné, bez kterých HTML element nemůže správně fungovat. A atributy událostí, které spouští určité skripty na základně určitých událostí. [11, 12]

```
<a href="www.seznam.cz"> Seznam odkaz </a>
```

*Zdrojový kód 1 HTML element <a> </a>*

Na ukázce (viz zdroj. kód 1) je zobrazen HTML element tvořený značkou <*a*>. Značka <*a*> vytváří z textu uvnitř závorek odkaz. Značka je doplněna o atribut, který definuje kam bude po kliknutí na element prohlížeč přesměrován. V tomto případě se jedná o stránku seznam.cz.

### 4.1.4 HTML 5

HTML 5 je nejaktuálnější vydanou verzí jazyka HTML. Finální specifikace této verze byla vydána 28.října 2014. Oproti předchozí verzi HTML 4 z roku 1997 přináší HTML 5 podstatné změny, k těm nejzásadnějším patří podpora multimédií ve webovém prohlížeči. Dále stojí za zmínku tyto novinky. [15]

**File API** v HTML 5 umožňuje nahrávat neomezené množství souborů i rozkouskování větších souborů a postupný upload jednotlivých kousků. Tímto způsobem lze obcházet restrikce maximálního uploadu v nastavení serveru. [15]

**CSS3** rozebráno v kapitole 4.2 CCS

**Geolocation API** umožňuje bez nutnosti jakýchkoliv doplňků požádat o sdělení vaši globální pozice. Ta je vypočítána z parametrů, kterými webový prohlížeč disponuje. [15]

**Media API** umožňuje přehrávání medií (video a hudba) v prohlížeči. A také naprogramování celého prohlížeče medií bez použití pluginu Flash. Zahrnující metody play, pause, load a také možnost celé obrazovky v kombinaci s Fullscreen API. [15]

**SVG** podpora formátu SVG umožnuje vykreslovat v prohlížeči obrázky bez ztráty kvality při zvětšení stránky webového prohlížeče. [15]

**Sémantika** HTML 5 zavedlo spoustu nových tagů, které jsou určeny pro zlepšení přehlednosti a optimalizaci webových stránek. Místo dělení stránky do divů lze použit značky přímo podle toho, jaký obsah bude značka obsahovat. Například část internetový článek neobalíme značkami div, ale novou značkou article. Dalším příkladem pak může být hlavička stránky, pro kterou můžeme použít značku header. [15]

## 4.2 CSS

CSS (Cascading Style Sheets) je zkratka pro jazyk, který graficky upravuje HTML dokument a jeho značky. V českém jazyce je CSS známo pod názvem Kaskádové styly. Samotné značky v HTML jsou naformátované a jejich vzhled je předem daný. Lze ho upravovat jen pomocí několika málo HTML značek, které nemění strukturu, ale grafické zobrazení elementu. Proto se vývojáři webových stránek rozhodli pro vytvoření jazyka, jenž bude upravovat jen grafickou podobu HTML dokumentu. A tím se jím podařilo oddělit vzhled dokumentu od jeho struktury a obsahu. To sebou neslo několik výhod, větší přehlednost v kódu, zlepšení použitelnosti a snazší správu. Dnes je technologie CSS, rozšířená na většině webových stránek. Za pomoci CSS se upravuje především barva textu, rozložení prvků, typografie a mnoho dalších grafických prvků. [16, 17]

### 4.2.1 Syntaxe a způsob implementace

Kaskádové styly lze implementovat třemi způsoby. Prvním způsobem je deklarace stylu na přímo, a to vložení atributu *style* přímo do špičatých závorek formátovaného elementu (viz zdroj. kód 2). Jedná se o nejzákladnější formu implementace. [16, 17]

```
<h1 style="color:red"> Nadpis upravený </h1>
```

*Zdrojový kód 2 Deklarace stylu na přímo*

Další možností, jak styl deklarovat je pomocí stylopisu (stylesheet) umístěného v hlavičce html stránky (viz zdroj. kód 3). Pod pojmem stylopis se skrývá seznam stylů. V seznamu stylů jsou uvedeny jednotlivé styly, ty určují, jak jaký element má být zformátován. Například nadpis *h1*, má být podtržený a červený. Stylopis je ohraničen značkami <*style*> a </*style*>. [16, 17]

```
<style>
  h1 {
    color: red;
  }
</style>

<h1>Nadpis upravený</h1>
```

*Zdrojový kód 3 Stylopis umístěný v hlavičce stránky*

Poslední možností je použití externího souboru, kde bude stylopis deklarován. Soubor, kde je stylopis deklarován musí končit příponou .css. Na daný soubor se v HTML dokumentu odkazuje pomocí značky <*link*> (viz zdroj. kód 4). Výhodou tohoto řešení je větší přehlednost a čitelnost jak v HTML kódu, tak v kódu CSS. [16, 17]

```
<link rel="stylesheet" href="styly.css"> </link>
<h1> Nadpis upravený </h1>
```

*Zdrojový kód 4 Odkaz na externí stylopis*

### 4.2.2 CSS 3

CSS 3 je třetí a zároveň nejnovější verzí jazyka CSS. CSS 3 je velkým skokem v grafickém návrhu webových aplikací a stránek. Umožňuje tvořit většinu grafických prvků stránky pouze svépomocí a tím urychlit proces načítání. Grafika tvořená v CSS je vektorová a nedochází k ztrátě kvality při jejím zvětšení. Verze CSS3 je stále vyvíjená, kompatibilita jednotlivých verzí ve webových prohlížečích se může lišit. Ke kontrole kompatibility je vývojáři často používán online nástroj caniuse.com. Caniuse kontroluje kompatibilitu CSS vlastností napříč prohlížeči. CSS 3 přináší následující nové klíčové vlastnosti. [18, 19]

Vlastnost zaoblené rohy border-radius v CSS 3 umožňuje nastavit velikost zaoblení rohů rámečku, díky této vlastnosti se nemusí řešit zaoblení pomocí obrázků. Tuto vlastnost lze použít s jednou, dvěma nebo i čtyřmi hodnotami. [18, 19]

Dále vlastnost stín u blokového prvku pomocí vlastnosti box-shadow se nastavuje velikost stínění u blokového prvku. U této vlastnosti se nastavují čtyři hodnoty. První nastavovanou hodnotou je horizontální posun stínu od objektu. Druhou hodnotou je vertikální posun stínu od

objektu. Třetí hodnotou je okraj stínu, kde stín přechází do ztracena. Poslední čtvrtá hodnota je určena pro barvu stínu. [18, 19]

Vlastnost stín u textu (text-shadow) nastavuje stín u obyčejného textu. U této vlastnosti se také nastavují čtyři hodnoty. A to horizontální posuv stínu, vertikální posun stínu, okraj stínu a barva stínu. [18, 19]

Vlastnost transformace umožňuje transformovat element. Jsou rozlišovány tři transformace a to translate, rotate a scale. Vlastnost translate umožňuje pohyb elementu po ose x a ose y. Vlastnost rotate umožňuje rotaci prvku a vlastnost scale umožňuje změnu velikosti prvku. [18, 19]

## 4.3 JavaScript

Jedná se o programovací jazyk používaný pro vývoj webových aplikací a stránek. Na rozdíl od HTML a CSS se jedná o programovací jazyk (objektově-orientovaný) tudíž má blíže ke klasickým objektovým programovacím a objektově-orientovaným jazykům jako je PHP, Java, Python a mnoha dalším. Kód v jazyce JavaScript je označován jako skript. Proto je JavaScript označován jako skriptovací jazyk. JavaScript umožňuje webovým stránkám a aplikacím dynamické chování. Tedy částečně možnost měnit jejich obsah a vzhled stránky na základě pokynů od uživatele. [20, 21, 22]

### 4.3.1 Fungování JavaScriptu

Společně s HTML je stažen z web serveru, kde je následně interpretován (přeložen) webovým prohlížečem. Pomocí JavaScriptu jsou dále ovládány různé elementy na stránce například: tlačítka, animace elementů, textové pole, efekty obrázků. Většina funkcí, které JavaScript umožňuje jsou spouštěny na základě interakcí uživatele s danou stránkou či aplikací. [20, 21]

### 4.3.2 Syntaxe a příklad použití

Chceme-li na webové stránce či aplikaci využívat JavaScript, je několik možností, jak kód JavaScriptu implementovat. První možností je implementace kódu přímo do HTML souboru, a to za pomocí HTML značek <script> </script>. Mezi tyto značky vkládáme přímo kód jazyka JavaScript. Kód jazyka JavaScript má obdobnou syntaxi jako ostatní objektově-orientované jazyky. Je možné vytvářet posloupnosti příkazů a větvení a data zapouzdřovat do objektů. Pro méně rozsáhlé aplikace a stránky, si vývojář vystačí s několika málo příkazy. [20, 21]

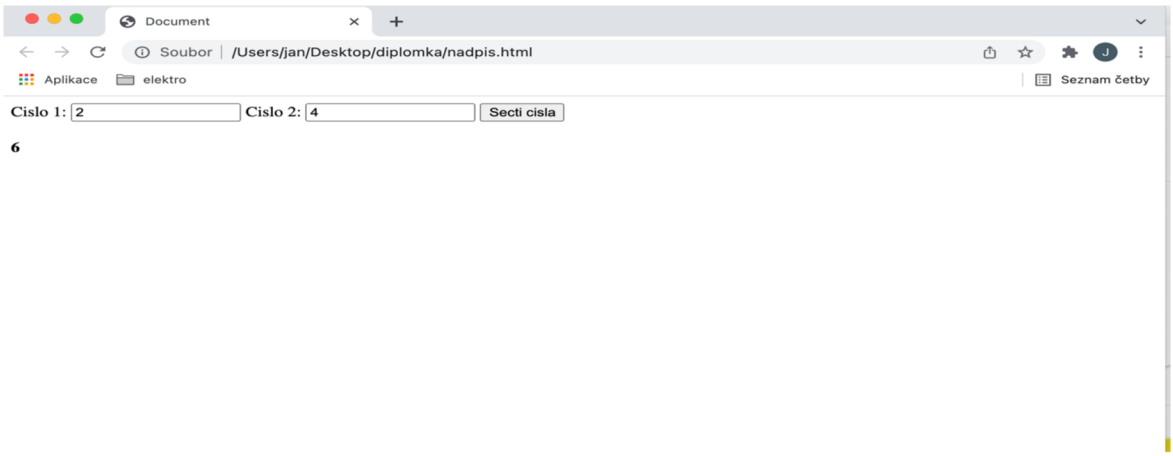
Na jednoduché ukázce je zobrazen dokument HTML a v něm je vložený skript jazyka JavaScript (viz zdroj. kód 5). Tento skript má za účel sečíst dvě čísla, která jsou uživatelem zadána do textových polí v kódu označenými značkami *<input> </input>*. Po zadání čísel a kliknutí na *tlačítko Sečti Čísla.* Je spuštěn skript v značkách *<script> </script>*. V skriptu je nadefinována funkce, která čísla od uživatele sečte a vrátí jejích součet a vypíše je do HTML kódu.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>

  <script>
    function sectiCisla() {
      let x = parseInt(document.getElementById("cislo1").value);
      let y = parseInt(document.getElementById("cislo2").value);

      let vysledek = x + y;
      document.getElementById("vysledek").innerHTML = vysledek;
    }
  </script>
  <body>
    Cislo 1: <input id="cislo1" /> Cislo 2: <input id="cislo2" />
    <button onclick="sectiCisla()">Secti cisla</button>

    <h4 id="vysledek"></h4>
  </body>
</html>
```

*Zdrojový kód 5 Ukázka syntaxe jazyka JavaScript*

Podle toho, jaké uživatel zadá čísla, dostane od prohlížeče výsledek (viz obr. 4). Tím se stránka stává dynamická a mění svůj obsah na základě nějaké interakce s uživatelem. Za tímto účelem je využíván jazyk JavaScript.

*Obrázek 4 JavaScript sečtení dvou čísel*




*Zdroj: autor*

### 4.3.3 ECMAScript

ECMAScript je standard ze kterého vychází jazyk JavaScript. ECMAScript zajišťuje interoperabilitu webových aplikací a stránek napříč webovými prohlížeči. Je normován společností ECMA International. ECMAScriptu bylo doposud vydáno 12 verzí. Jednotlivé verze jsou označovány zkratkou ES a číslem verze. [23]

### 4.3.4 TypeScript

TypeScript je nadstavbou jazyka JavaScript, který rozšiřuje jazyk JavaScript o statické typování, rozhraní a moduly. Byl vytvořen firmou Microsoft v roce 2012 a je vydán pod licencí open source. Na rozdíl od čistého JavaScriptu je TypeScript objektově-orientovaný. TypeScript není pro webové prohlížeče čitelný, proto je kompilován a konvertován do jazyka JavaScript. Ke kompilaci je používán Transpiler. Transpiler zpracovává kód v jednom jazyce (TypeScript) a stejně fungující kód vytváří v jazyce druhém (JavaScript). Soubory obsahující kód TypeScript musí končit příponou .ts. [31, 32, 33]

**Rozdíly mezi JavaScript a TypeScript**

*Tabulka 1 Porovnání JavaScript a TypeScript*

| JavaScript | TypeScript |
|---|---|
| Skriptovací jazyk | Objektově-orientovaný |
| Dynamicky typovaný | Staticky typovaný |
| Nepodporuje moduly | Podporuje moduly |
| Neumožňuje rozhraní (interface) | Umožňuje rozhraní (interface) |

*Zdroj: https://www.stxnext.com/blog/typescript-pros-cons-javascript/*

**Výhody**

- Zachycení bugů při samotném vývoji (kompilaci)
- Udržitelnost kódu
- Vysoká popularita

**Nevýhody**

- Nutnost kompilace
- Komplikovaný typovací systém [31]

### 4.3.5 Frameworky a knihovny

JavaScriptové frameworky jsou kolekce knihoven jazyka JavaScript. Poskytující připravené části kódu pro zjednodušení a zrychlení vývoje webové aplikace. Pomocí frameworků lze dosáhnout stejné kvality aplikace a kódu jako při použití čistého JavaScriptu. Pro jazyk JavaScript existuje několik frameworků mezi nejznámější patří frameworky Vue.js. [24]

**Vue.js**

Vue.js je open-source javascriptový framework sloužící pro vytváření uživatelských rozhraní a single-page aplikací. Framework Vue.js byl publikován v roce 2014 Evane Youem a řadí se mezi nejpoužívanější JavaScriptové frameworky. Vue.js reprezentuje přírůstkově adaptibilní

architekturu se zaměřením na deklarativní renderování a kompozici komponent. Jádro frameworku se zaměřuje pouze na zobrazovací vrstvu. [25, 26]

**React**

React je moderní javascriptová knihovna pro tvorbu uživatelského rozhraní. Knihovně je věnována samostatná kapitola 4.4 React. [29]

**jQuery**

jQuery je nejrozšířenější javascriptová knihovna, která je navržena k zjednodušení procházení a manipulaci s HTML DOMem. Umožňuje zpracování událostí, AJAX a CSS animace. Velkou výhodou jQuery je obrovské množství jQuery pluginů, které si vývojář pro svoje stránky může stáhnout. Získá tak například: rozšířené formulářové prvky, interaktivní galerie, carousely a mnoho dalších doplňků. Syntaxe jQuery je navržena, tak aby co nejvíce ulehčovala navigaci v dokumentu, výběr elementů DOMum, AJAX, vytváření animací a zpracování událostí. [27]

## 4.4 React

React je knihovna pro jazyk JavaScript. Knihovna definuje několik přídavných funkcí a procedur, které jsou pro programátora k dispozici a ulehčují mu vývoj webové aplikace. Knihovna React je jednou z nejrozšířenějších knihoven pro jazyk JavaScript. Jedná se o open-source knihovnu. Je vyvíjena a spravována společností Meta Platforms. React využívá technologii virtuálního DOMu na doplnění dat do HTML DOMu, tím dokáže měnit pouze ty části stránky, které jsou zrovna třeba a nemusí načítat celou webovou stránku znovu. [28, 29, 30]

### 4.4.1 Komponenty

Základní stavební prvek Reactu tvoří komponenty. Komponenty jsou znovupoužitelné části skládající se z HTML elementů a případně rozšířené o kód JSX. Z těchto komponent je složená celá webová aplikace. Vývojář si tak může například připravit komponentu vytvářející tlačítko. Je-li na stránce potřeba více stejných tlačítek, stačí mu pouze vložit komponentu tlačítko a nemusí kopírovat celý kód. Komponenty lze do sebe také vnořovat a tím vytvářet komplexnější celky. [28, 29, 30]

### 4.4.2 Vlastnosti a stavy

Každá komponenta si sebou nese vlastnosti a stavy. Ty představují nějaké dodatečné informace, které si komponenta pamatuje a může je předávat. Vlastnosti se nazývají props z anglického termínu properities. Props umožňují předávat data z jedné komponenty do druhé. Na příklad komponenta formulář může předat vyplněná data od uživatele do komponenty tabulka. Stavy nazývané states, jsou data o uložená v komponentách. Ty si komponenta pamatuje a může je zobrazovat. Stavy jsou nastaveny na výchozí hodnoty při vytváření kódu. Stavy jsou měněny interakcí s webovou stránkou, například získáním dat od uživatele či získáním dat voláním serveru na základně nějaké události od uživatele. [28, 29, 30]

### 4.4.3 Výhody

Výhodou knihovny React oproti čistému JavaScriptu je tvorba komponent. Komponenty jsou znovupoužitelné a vytváření aplikací je s nimi snadnější a zároveň přehlednější. Dalším plusem komponent je zapouzdření logiky do jednoho celku. Další výhodou knihovny React je výkonnost v situacích, kdy dochází k volání serverových API, a následném zpracování dat získaných ze serveru. Po zpracování přijaté odpovědi se nemusí znovu načítat celý HTML DOM, ale pouze ta část DOMU, která byla pozměněna. To umožňuje technologie virtuálního DOMU, kterou React disponuje. Dalšími výhody je sada nástrojů pro ladění aplikací, které je možno integrovat do webových prohlížečů. A velká sada doplňujících balíčků. [28]

### 4.4.4 Nevýhody

Mezi nevýhody lze řadit především rychlé tempo vývoje celé knihovny. Vývojáři často mění funkce a zaběhlé způsoby v knihovně a tím vzniká nutnost neustále sledovat poslední trendy ve vývoji. Další nevýhodou je slabá dokumentace knihovny, která je z části zaviněna rychlým vývojem knihovny. [28]

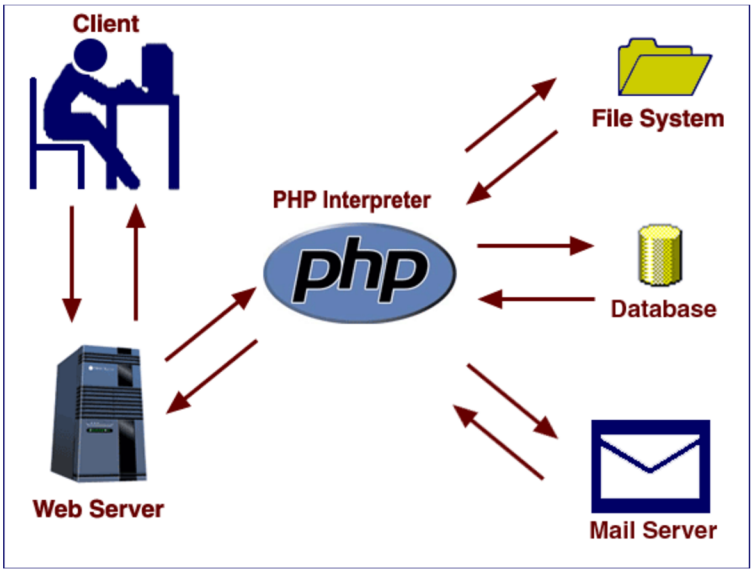
## 4.5 PHP

PHP je programovací jazyk používaný především pro tvorbu webových aplikací. Jedná se o objektově-orientovaný skriptovací jazyk. Jazyk PHP umožňuje tvořit dynamické stránky, aplikace a interagovat s databází. Jazyk PHP je obvykle interpretován na serveru za pomoci PHP enginu. Jazyk PHP využívá také nejrozšířenější redakční systém WordPress. [34, 35, 67]

### 4.5.1 Princip činnosti PHP a PHP enginu

Klient zasílá požadavky z webové aplikace či stránky na webový server, na serveru musí být nainstalován a spuštěn PHP engine, který kód jazyka PHP interpretuje (viz obr. 5). PHP engine daný požadavek zpracuje a web server vrátí klientovy odpověď ve formě http. Při zpracování požadavků může php engine přistupovat do databáze, souborového systému a využívat emailový server. [34, 35, 67]

*Obrázek 5 Činnost PHP*




*Zdroj: https://elmcip.net/platformsoftware/php*

### 4.5.2 Syntaxe

Kód v jazyce PHP lze vkládat přímo do HTML dokumentu. Dokumenty obsahující kód PHP, musí končit koncovkou .php, to proto aby bylo jasně deklarováno, že soubor obsahuje PHP skript. Samotný skript v HTML dokumentu se pak vkládá mezi značky *<?php "vlastní kód" ?>*. Kód PHP se velice podobá kódu ostatních objektových a objektově-orientovaných programovacích jazyků. Pracuje s datovými typy, funkcemi, třídami a objekty. Jazyk umožňuje používat objektová paradigmata. Tedy zapouzdření dat, dědičnost, abstraktní třídy, rozhraní a další prvky objektové přístupu. Pro psaní kódu PHP existuje několik vývojových prostředí. Mezi nejznámější a nejvíce používané vývojové prostředí patří například: PHPStorm a Visual Code Studio. [34, 35, 67]

### 4.5.3 PHP a databáze

Velké uplatnění má jazyk PHP při komunikaci s databází. Web server ani klient nemůžou s databází přímo komunikovat, musí proto využít rozhraní programovacího jazyka, například jazyka PHP. PHP nabízí několik abstrakčních vrstev pro komunikaci. Mezi nejvíce používané patří MySQLi, to je používané výhradně pro systém řízení báze dat MySQL. Dále pak rozhraní PDO, které podporuje 12 systému řízení báze dat. PDO také umožňuje zjednodušené zadávání SQL dotazů. [34, 35, 67]

### 4.5.4 Výhody

Jazyk PHP se velice snadno učí na rozdíl od některých známých objektových jazyků. Syntaxe je logická a dobře organizovaná. Tím ulehčuje vývoj a optimalizaci webových aplikací pro mnoho vývojářů. Další výhodou je vysoká flexibilita jazyka. Mezi největší výhody patří kompatibilita s operačními systémy, web hostingy a servery. [36, 37]

### 4.5.5 Nevýhody

Není silně typovaný. Kombinací mnoha knihoven a frameworků dochází k jeho zpomalení. Nezachycuje tolik potenciálních chyb jako ostatní jazyky. [36, 37]

## 4.6 Nette framework

Je český open source PHP framework pro tvorbu webových aplikací. Skládá se z několika samostatně použitelných komponent. Podporuje principy DRY, KISS eliminuje některá bezpečností rizika. Klade důraz na znovupoužitelnost. Využívá softwarovou architekturu MVP. Je často využíván pro tvorbu e-shopů, redakčních systému a rezervačních systémů. Nette Framewok se řadí mezi open source software. Výhodou frameworku nette je dokumentace v českém jazyce a rozsáhlá komunita vývojářů v Česku. [38, 39]

## 4.7 SQL

SQL (Structured Query Language) je standardizovaný dotazovací jazyk sloužící pro práci s daty v relačních systémech řízení báze dat. Byl vyvinut společností IBM v 70.letech 20. století. Mezi relační systémy řízení báze dat používající dotazovací jazyk SQL patří například: MySQL, SQL Server, Oracle, MS Access a PostgreSQL. Příkazy v jazyce SQL se dělí na čtyři základní skupiny. [67]

**Příkazy pro definici dat**

Příkazy pro definici dat vytvářejí či upravují struktury databáze např. tabulky, indexy a pohledy. Jedná se o příkazy CREATE, ALTER, DROP. [40, 41]

**Příkazy pro manipulaci s daty**

Příkazy sloužící pro získávaní, ukládání a mazání dat v databázi. Jedná se o příkazy SELECT, INSERT, UPDATE, MERGE, DELETE A SHOW. [40, 41]

**Příkazy pro řízení dat**

Příkazy sloužící pro nastavování přístupových práv. Jedná o příkazy GRANT a REVOKE. [40, 41]

**Příkazy pro správu databázových transakcí**

Do této skupiny spadají příkazy START TRANSACTION, COMMIT, ROLLBACK. [40, 41]

**Ukázka syntaxe jazyka SQL**

*Obrázek 6 Ukázka syntaxe jazyka SQL*

```
CREATE TABLE zvire
(
    cislo INT NOT NULL PRIMARY KEY,
    nazev_druhu VARCHAR(100) NOT NULL,
    hmotnost INT NOT NULL,
    barva VARCHAR(100) NOT NULL
)
```

*Zdroj: autor*

## 4.8 Git

Git je informatice termín pro distribuovaný systém správy verzí. Systém správy verzí je v informatice způsob, jak provádět kontrolu verzí softwaru a spravovat verze softwaru při jeho vývoji. Systém Git poskytuje uživatelům (programátorům) možnost pracovat zároveň na jednom projektu bez sdílení počítačové sítě. Za vznikem Gitu stojí Linus Torvalds vývojář

operačního systému Linux. Cílem Gitu je zrychlení vývoje, datová integrita, nelineární tok práce (několik větví softwarů běžících na odlišných systémech). [44, 45]

### 4.8.1 Centralizovaný systém verzování

Centralizovaný systém využívá architektury server-klient. V centralizovaném systému verzování je server hlavním repozitářem obsahující veškeré verze kódu. K tomu, aby mohl vývojář na projektu pracovat musí nejdříve získat kód z hlavního repozitáře na serveru. Po provedení změn na softwaru musí klient (vývojář) celý kód opět poslat na hlavní repozitář umístěný na serveru. Toto řešení pracuje pouze z jedním repozitářem obsahujícím veškeré verze, historii a větvě softwaru. [45]

### 4.8.2 Distribuované verzování

Způsob distribuovaného verzování je založen na architektuře peer-to-peer. Místo jednoho repozitáře na serveru, má každý vývojář vlastní "server" na osobním počítači a na něm repozitář s veškerým kódem včetně jeho verzí a větví. To umožňuje vývojáři pracovat lokálně na svém zařízení. Synchronizace v distribuovaném verzování probíhá výměnou patchů (kolekcí změn). Tím odpadá potřeba centrálního repozitáře, kde se uživatelé synchronizují. Z čehož vyplývají následující rozdíly. [45]

**Hlavní rozdíly oproti centralizovanému systému**

- Absence referenční kopie kódové báze
- Běžné operace jsou rychlejší (nemusí komunikovat s centrálním serverem)
- Komunikace probíhá jen při sdílení změn mezi vývojáři
- Každá kopie funguje jako záložní kopie kódů, změn a historie [42]

### 4.8.3 Github

Webová služba GitHub poskytuje uživatelům distribuovaný systém správy verzí (Git). Aktuálně má GitHub přes 73 miliónů uživatelů a více než 200 miliónů repozitářů. Kromě verzování umožňuje uživatelům tvorbu dokumentace, systém sledování prostředků, bug tracking, tvorbu úkolů a mnoho dalších služeb. [68, 69, 42]

### 4.8.4 GitLab

GitLab je další webovou službou poskytující Git. Stejně jako GitHub poskytuje mnoho dalších vývojářských nástrojů. GitLab má přes 30 milionů registrovaných uživatelů. Využívají ho organizace jako NASA, CERN a Alibaba. [43, 42]

### 4.8.5 Porovnání GitHub a GitLab

*Tabulka 2 Porovnání GitHub a GitLab*

| Parametr | GitHub | GitLab |
|---|---|---|
| Druh licence | Není open-source | Open source (komunitní edice) |
| Veřejný repozitář | Ano (neomezený počet) | Ano (omezený počet) |
| Privátní repozitář | Ano (maximálně 3 vývojáři) | Ano (neomezený počet) |
| Analýza projektu | Zobrazuje historii commitů | Zobrazuje historii commitů a poskytuje graf vývoje projektu |
| Ostatní parametry | Velká komunita | Vysoce zabezpečený |

*Zdroj: https://www.geeksforgeeks.org/difference-between-gitlab-and-github/*

# 5 Vývoj webových aplikací z hlediska architektury a přístupu

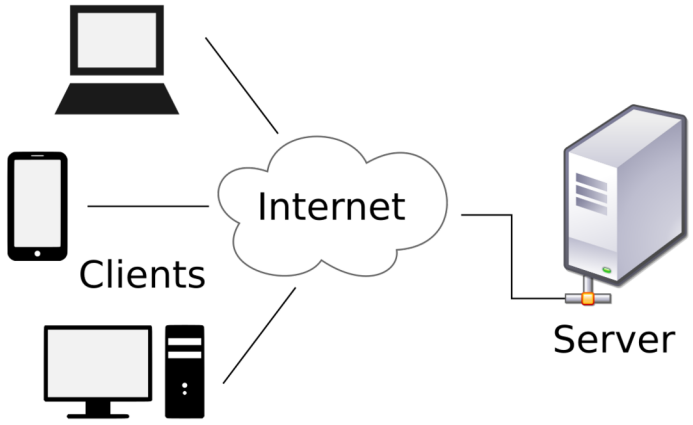
V kapitole Vývoj webových aplikací z hlediska architektury a přístupu jsou definovány pojmy vztažené k webové aplikaci týkající se architektury a bezpečnosti aplikace.

## 5.1 Klient – server

Klient–server model je struktura oddělující klienta od serveru pomocí počítačové sítě (viz obr. 7). Na serveru je obvykle spuštěno několik služeb, které klient může využívat. Klient zasílá požadavky na server a server mu zasílá odpovědi. Server může poskytovat několik druhů služeb. Mezi nejvíce používané patří: web servery, souborové servery, emailové servery a databázové servery. [46, 47]

Klienta nezajímá, jaké operace na serveru probíhají, zajímá ho pouze odpověď. Aby server mohl s klientem komunikovat potřebují si vyměňovat zprávy. K výměně zpráv slouží komunikační protokoly, každá služba využívá vlastní komunikační protokol. Například web server používá známý a rozšířený protokol http. [46, 47]

*Obrázek 7 Architektura klient-server*




*Zdroj: https://en.wikipedia.org/wiki/Client%E2%80%93server_model*

### 5.1.1 Web Server

Web server je software a hardware využívají komunikační protokol k odesílání odpovědí na požadavky klienta. Na straně hardwaru webového serveru se stará o ukládání souborů webové aplikace či stránky například: HTML dokumenty, kaskádové styly, javascriptové soubory a PHP soubory. Dále je hardwarová část fyzicky připojena k internetu a podporuje výměnu dat s dalšími zařízeními (klienty) na síti. Kvalita a robustnost hardwaru určuje kolik požadavků dokáže server obsloužit za jednotku času. Na straně softwarové server řídí správný a bezpečný průběh požadavků a odpovědí skrze komunikační protokol HTTP. Dnešní servery především využívají pokročilejší protokol HTTPS, který je oproti standardnímu protokolu http šifrovaný. [48]

## 5.2 REST

REST (Representational state transfer) je softwarová architektura definující určité zásady chování distribuované architektury webu, používaná k snadnému způsobu, jak číst, vytvářet, mazat a editovat data na serveru. K dosažení cílů používá jednoduchých HTTP volání. Architektura REST klade důraz na škálovatelnost interakcí mezi komponenty. Komponenty představují v případě distribuované architektury webu počítač klienta a služby běžící na serveru, které pro svojí vzájemnou komunikaci využívají internetovou síť a komunikační protokol HTTP. Architektura REST definuje šest fundamentálních zásad. [49, 50, 51]

### 5.2.1 Zásady REST architektury

**Server-klient:** zásada dodržení architektury server-klient, zajišťující oddělení těchto entit od sebe. [49, 50, 51]

**Bezstavovost:** každý http požadavek musí obsahovat veškeré informace k jeho vykonání. [49, 50, 51]

**Cache:** cashevotelnost odpovědi musí být jasně definovány jako casheovatelná či necacheovatelná odpověď. [49, 50, 51]

**Vrstevnatost**: umožňuje skládat několik serverových služeb do sebe za účelem zvýšení variabilnosti. [49, 50, 51]

**Jednotné rozhraní:** dodržení jednotného rozhraní, které umožňuje nezávislý vývoj aplikace na vrstvě API. [49, 50, 51]

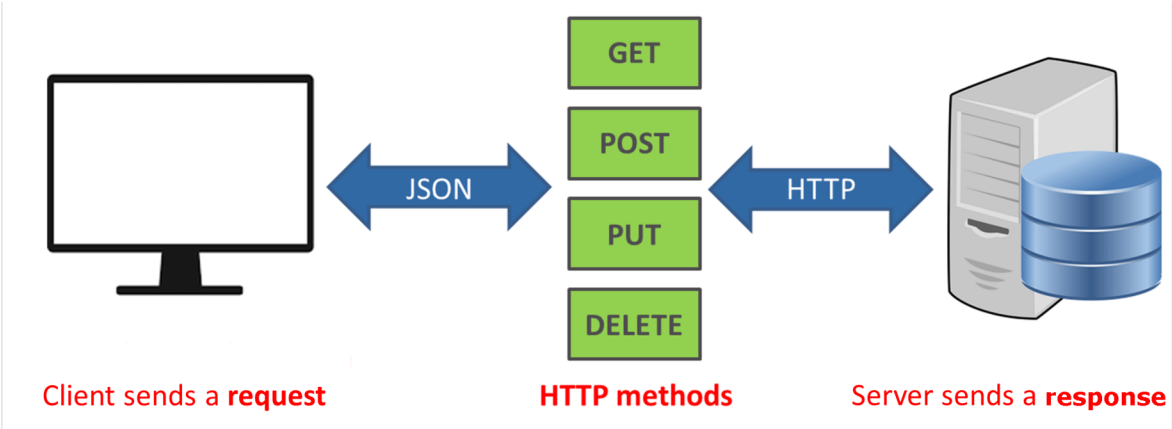
**Code-On-Demande:** server může dočasně rozšířit či upravit funkcionality klienta kódem, který zašle server. [49, 50, 51]

### 5.2.2 RESTful API

Komunikace více softwarů mezi sebou je v informatice umožněna pomocí API (Application Programming Interface). Toto rozhraní je využíváno i při tvorbě webových aplikací. Kde je využíváno pro komunikaci klienta se serverem. Při tvorbě webových aplikací je často požadováno na serveru provádět tyto operace: získávat data, mazat data, editovat a vytvářet data. Tyto požadavky na server jsou často označovány zkratkou CRUD (create, read, update, delete). Pro splnění požadavků je v některém z jazyků běžících na serveru (např. PHP, Python) podmínkou vytvořit API, které bude tyto operace na serveru obsluhovat. Přijímat požadavky od klienta a zasílat klientovy odpovědi. Při dodržení zásad softwarové architektury REST můžeme takové API označit za RESTful API. [49, 50, 51]

RESTful API používá metody http požadavků pro obsluhu uživatelů API (viz obr. 8). Pro každou operaci ze skupiny operací CRUD, je v REST architektuře přidělena vlastní HTTP metoda. Každá HTTP metoda sebou nese stavové kódy, které informují uživatele o výsledku jejich požadavku. Ty se dělí do několika kategorií podle významu. Data získána ze serveru jsou odesílané ve formě JSON. JSON je datový formát sloužící pro přenos dat nezávislý na počítačové platformě. [49, 50, 51]

*Obrázek 8 REST API*




*Zdroj: https://phpenthusiast.com/blog/what-is-rest-api*

### 5.2.3 HTTP metody a kódy

**Metoda POST:** HTTP metoda sloužící k vytváření záznamu na serveru. Požadavky POST nelze cachovat. Často používaná pro přenos dat z formulářů, data putují skrytě na rozdíl od metody GET. [52]

**Metoda GET:** HTTP metoda sloužící k získání záznamu na serveru. Data odesílané touto metodou jsou zobrazeny v URL adrese. [52]

**Metoda PUT:** HTTP metoda sloužící k úpravě záznamu na serveru. [52]

**Metoda DELETE:** HTTP metoda sloužící k smazání záznamu na serveru. [52]

**Stavové kódy**

- 1xx – informační – požadavek byl přijat a zpracovává se
- 2xx – úspěšná – akce byla přijata, akceptována a zpracována
- 3xx – přesměrování – je třeba provést další akci
- 4xx – chyba klienta – problém na straně klienta
- 5xx – chyba serveru – server nedokáže požadavek provést [74]

## 5.3 MVC

MVC je architektonický vzor, který rozděluje webové aplikaci do tří oddělených částí (viz obr. 9). Tyto části jsou nazývány komponenty. Každá webová aplikace postavené na architektonickém vzoru MVC obsahuje komponenty třech typů a to Modely, Controllery (kontrolery) a View (pohledy). [53]

### 5.3.1 Model

Model provádí veškerou logiku aplikace. Do oblasti logiky webové aplikaci patří: databázové dotazy, výpočty, autentizace, validace a další logické operace. Model spravuje data webové aplikace. Model pouze data zpracovává, nestará se o to, kam budou data přeposlána či vykreslena. [53]

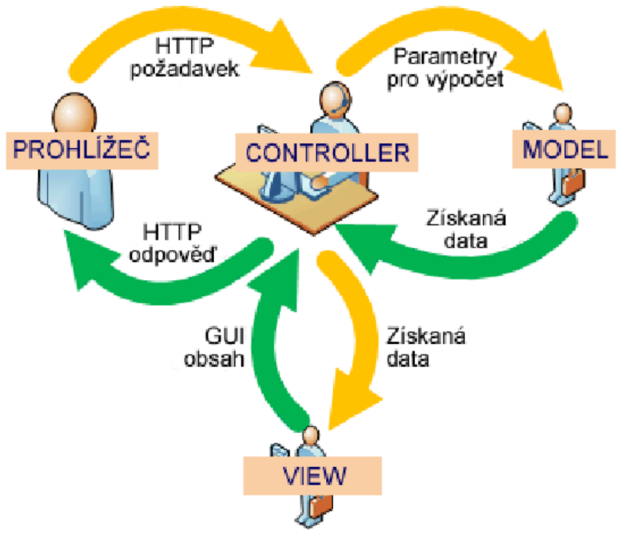
### 5.3.2 View

Komponenta view představuje v MVC architektuře reprezentační část. Reprezentuje získaná data a generuje grafické prostředí (GUI). Data přijímá od komponenty controller, který získává data modelu. Data tak nejsou předávány přímo mezi modelem a pohledem, ale přes prostředníka. [53]

### 5.3.3 Controller

Komponenta controller umožňuje propojení modelu a pohledu. Contoller komunikuje jak s modely, tak pohledy. Řídí veškerý tok dat v aplikaci a staví se do role prostředníka, který umožňuje nezávislost modelu na pohledu a obráceně. [53]

*Obrázek 9 MVC schéma*




*Zdroj:https://igloonet.cz/blog/zprijemnete-si-vyvoj-webovych-stranek-s-frameworkem-ruby-on-rails-dil-2/*

## 5.4 Single-page application

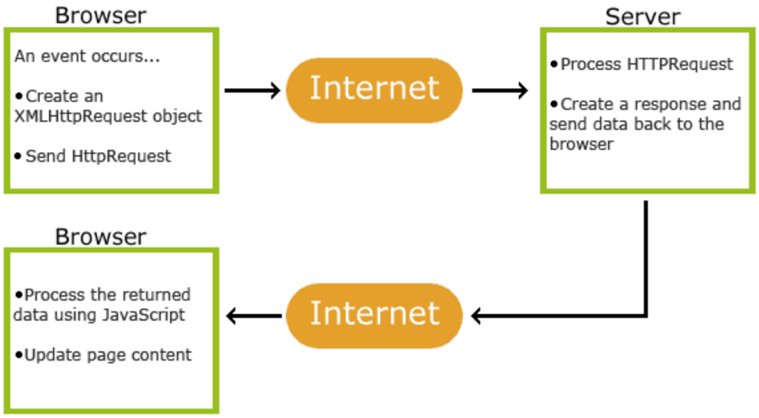
Jedná se o druh webové aplikace nebo stránky, která mění dynamicky obsah bez toho, aby se musela celá stránka znovu načítat. Vždy když webová aplikace přijme data ze serveru přepíše pouze tu část stránky, kde je třeba vykreslit přijímaná data ze serveru. Tento druh aplikací je velice populární. Stránky jsou rychlejší a uživatel je může ovládat jako klasickou desktopovou aplikaci. Mezi nejznámější single-page aplikace patří například: Facebook, Gmail, Google Mapy a GitHub. [54]

Pro vývoj takových to aplikací se nejčastěji používají javascriptové frameworky, které umožňují asynchronní volání serveru za pomoci technologie AJAX. Mezi nejrozšířenější patří React, Angular a Vue.js. [54]

### 5.4.1 AJAX

Technologie AJAX (Asynchronous JavaScript and XML) využívá rozhraní XMLhttpRequest nebo modernějšího rozhraní Fetch API (viz obr. 10). Přes tyto rozhraní odesílá http požadavky na server. Web server zpracuje požadavky a odešle odpověď http s daty například ve formátu JSON, čistého textu či XML. JavaScript nebo některý javascriptový framework přečte odpověď. Po přečtení dat spustí akci na základě získaných dat, kde změní pouze tu část stránky, kde se přidávají nebo mění získaná data. [55, 56]

*Obrázek 10 Technologie AJAX*




*Zdroj: https://www.w3schools.com/js/js_ajax_intro.asp*

### 5.4.2 JSON

JSON (JavaScript Object Notation) je otevřený standardizovaný datový formát sloužící k zápisu dat a výměně dat nezávisle na počítačové platformě. JSON je jedním z nejpoužívanějších datových formátu jeho syntaxe je odvozena z javascriptové objektové notace, ale data jsou ukládána v podobě textu. Data mohou být uspořádána v polích nebo agregována v objektech. JSON soubory musí končit příponou .json. [57, 58]

V syntaxi JSON jsou data uloženy vždy ve dvojících klíč-hodnota. Data jsou oddělena čárkami. Hranaté závorky jsou určeny pro ukládaní dat do polí a složené závorky pro ukládání objektů. [57, 58]

**Datové typy v JSONu**

JSON umožňuje ukládat jen hodnoty následujících datových typů:

- JSONNumber – číslo
- JSONBoolean – logická hodnota
- JSONString – textový řetězec
- JSONObject – objekt
- JSONArray – pole
- JSONNull – null, prázdná hodnota [58]

**Ukázka JSONu**

Na ukázce je zobrazen obsah a syntaxe jednoduchého JSON dokumentu (viz zdroj. kód 6). Jedná se o pole, které je naplněno třemi objekty. Každý objekt v poli nese dva atributy.

```
[
    {
      "druh_zvířete": "kůň",
      "barva": "hnědá"
    },
    {
     "druh_zvířete": "ovce",
      "barva": "bílá"
    },
    {
     "druh_zvířete": "kůň",
      "barva": "černý"
    }
]
```

*Zdrojový kód 6 Ukázka JSONu*

## 5.5 JSON WEB Tokens

JSON Web Token (JWT) je otevřený standard, který definuje způsob, jak bezpečně přenášet informace mezi dvěma stranami v datovém formátu JSON. Takto odesílané informace lze považovat za ověřené a důvěryhodné. Pro zajištění důvěryhodností je JWT digitálně podepsán pomocí HMAC algoritmu nebo podepsán veřejným klíčem a soukromým za použití šifrování RSA, ECDSA. [59]

### 5.5.1 Použití JWT

Standard JWT je v praxi využíván pro autorizaci a přenos informací. Nejčastějším scénářem pro využití standardu JWT je autorizace. U webových aplikací využívajících standard JWT je po přihlášení uživatele do prohlížeče zaslán ze serveru JWT token. Tento token v sobě nese identitu přihlášeného uživatele. Token se ukládá do cookies nebo do úložiště prohlížeče. Při každém dotazu klienta na server je společně s dotazem zasílán i token. Server token dešifruje tím ověří identitu na základě ověření identity umožňuje využívat služby, pro které je uživatel autorizován. [59]

### 5.5.2 Struktura JWT

JWT se skládá ze tří částí oddělených tečkami (viz obr. 11). První část je header (hlavička) za ní následuje obsah (payload) a nakonec signature (podpis). [59]

**Header (Hlavička)**

Hlavička se skládá ze dvou částí. První částí je typ tokenu. Explicitně je typ tokenu nastaven na JWT. V druhé části je typ použitého šifrovacího algoritmu. [59]

**Payload (Obsah)**

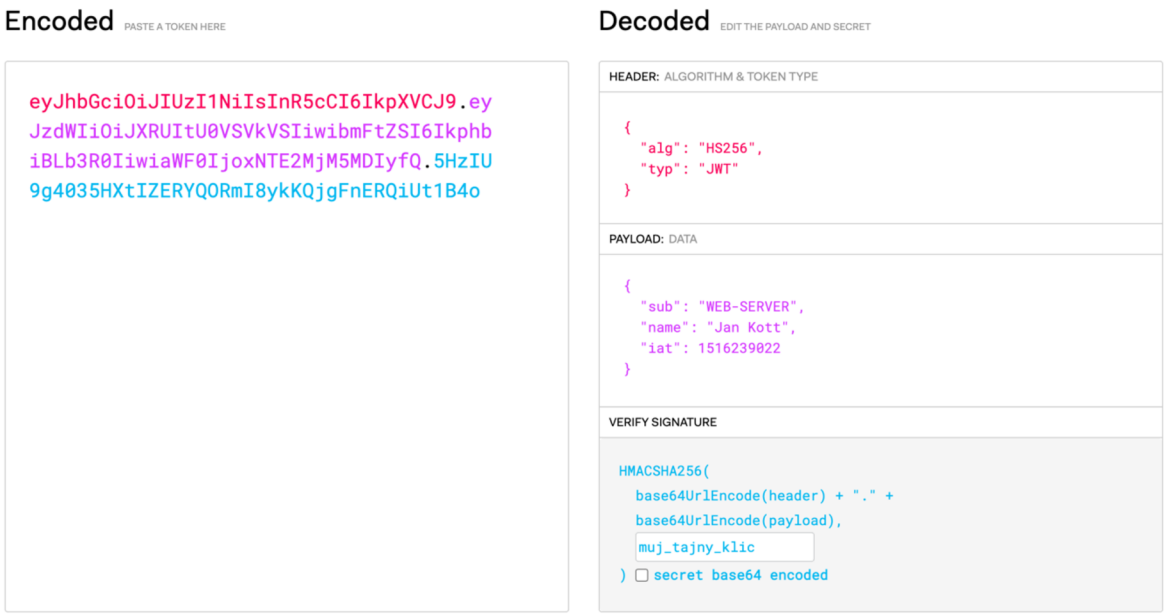
Payload obsahuje informace o uživateli případně další dodatečné informace. Mezi dodatečné informace patří, například vystavitel tokenu, časová expirace tokenu a subjekt. [59]

**Signature (Podpis)**

Signature se skládá ze zakódované hlavičky, zakódovaného obsahu a vlastního tajemství. Tajemství je symetrický klíč, který zná jak příjemce, tak odesílatel. Podpis je určen

k verifikování pravosti tokenu. Zaručuje, že token nebyl upraven a odesílatel je opravdu ten za koho se vydává. [59]

*Obrázek 11 Struktura JWT*

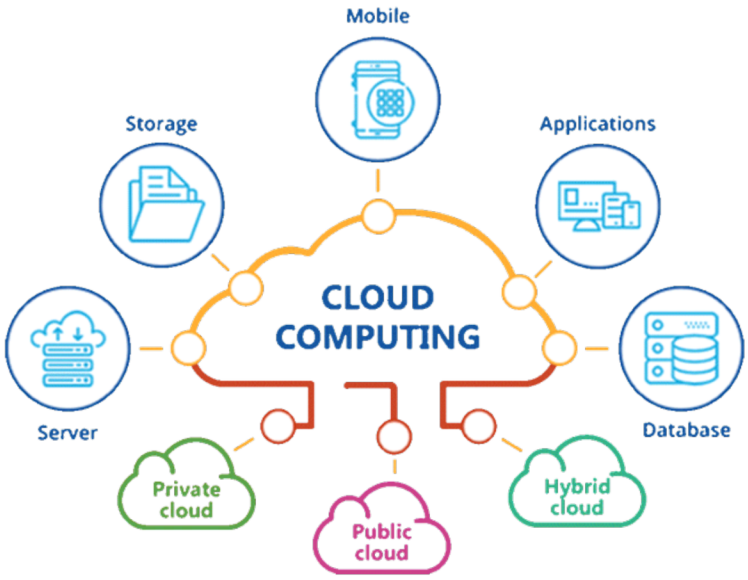



*Zdroj: https://jwt.io/*

## 5.6 Cloud Computing

Cloud computing je v informatice termín pro model, který poskytuje služby, programy a výpočetní techniku formou pronájmu (viz obr. 12). Pronájem těchto služeb a technologií je realizován prostřednictvím internetové sítě. Uživatel takového modelu nepotřebuje fyzicky vlastnit žádný server s databází a uložištěm. Veškeré služby a programy pronajaté uživatelem u poskytovatele Cloud Computingu, jsou spuštěny v datových centrech po celém světe. Cloud Computing je řešení, které je často používáno pro nahrání a ostrý provoz webových aplikací na internetu tzv. deploy. [60, 61, 62 63]

*Obrázek 12 Cloud computing*




*Zdroj: https://www.geetoo.com/cloud-computing/*

### 5.6.1 Model nasazení

Model nasazení definuje, jak jsou služby cloudu poskytovány, existují tři typy cloudového nasazení.

**Veřejný cloud**

Na veřejném cloudu je veškerá infrastruktura (výpočetní technika, služby a programy) umístěna na datových centrech poskytovatele Cloud Computingu. Poskytovatel tak kontroluje všechen hardware a software. Zákazník využívá služeb a výpočetního výkonu přes internetovou síť. Cloudová infrastruktura je uživateli poskytována za poplatek, který bývá nejčastěji spjatý v objemem využitých služeb. Poplatek za objem využitých služeb je označován pod zkratkou PAYG (pay-as-you-go). Zákazníci veřejného cloudu spolu sdílí stejné služby a stejnou výpočetní techniku. Takové uspořádání je nazýváno jako tzv. multitenant-architektura. Kdy jeden tenant je prostředí jednoho zákazníka. [61, 62, 63, 64]

Například tenant (prostředí jednoho zákazníka), kterému je poskytována databáze a datové úložiště, kde má nahrána všechny data sdílí úplně stejný hardware s jinými tenanty a jejich daty. Ale zároveň však multi-tenant architektura umožňuje oddělení a zabezpečení dat od ostatních tenantů. [61, 62, 63, 64]

### Výhody veřejného cloudu

- Snížení nákladů na software a hardware
- Téměř veškerá údržba je poskytována provozovatelem
- Snadná a téměř neomezená škálovatelnost [64]

### Privátní cloud

Na rozdíl od veřejného cloudu jsou všechny prostředky daného nasazení vztaženy a používány jedním uživatelem (tenantem). V privátním cloudu služby a infrastruktura běží na dedikované privátní síti. Veškerý hardware a software je vyhrazen pouze pro jednoho zákazníka. [61, 62, 63, 64]

Tento typ řešení je využíván především velkými organizacemi jako jsou finanční instituce, vládní agentury a další středně velké až velké organizace. Ty volí privátní cloud pro jeho zvýšenou kontrolu nad prostředím. [61, 62, 63, 64]

### Výhody privátního cloudu

- Vyšší flexibilita - nastavení cloudového prostředí na míru požadavkům zákazníka či organizace
- Lepší kontrola – vyšší úrovně kontroly a ochrany údajů
- Vysoká škálovatelnost, stejně jako u veřejného cloudu [64]

### Distribuční model

Distribuční model rozděluje cloud computing do třech kategorií. Jednotlivé kategorie definují, jak jsou služby či výpočetní technika klientům poskytovány. [61, 62, 63, 64]

### IaaS

IaaS (Infrastructure as a Service) je nezákladnější kategorií služeb. Zákazník si v případě zvolení řešení IaaS pronajímá infrastrukturu na bázi pravidelných průběžných plateb. Do infrastruktury jsou zahrnovány servery, virtuální počítače, úložiště a sítě. Iaas je vhodný pro zákazníky, kteří si nepotřebují pronajímat software a zároveň potřebují hardware a jeho správu. [61, 62, 63, 64]

### PaaS

PaaS (Platform as a Service) poskytuje zákazníkovi služby doručující vývojářské nástroje mobilních nebo webových aplikací, dále databáze a middleware, umělou inteligenci, chatovací boty, blockchain a mnoho dalších služeb. Celý PaaS je navržen tak, aby poskytoval robustní infrastrukturu pro vývoj aplikací. [61, 62, 63, 64]

### SaaS

SaaS (Software as a Service) poskytuje zákazníkovi softwarové aplikace přes internet. Nejčastěji pomocí předplatného. Poskytovatelé SaaS spravují a hostují softwarovou aplikaci včetně její údržby. Takové řešení eliminuje potřebu vlastního aplikačního softwaru na počítačích zákazníka. [61, 62, 63, 64]

# 6 Systém řízení báze dat

Systém řízení báze dat je v informatice termín pro softwarový systém sloužící uživateli pro manipulaci s daty a jejich správu. Tento systém je často zkráceně označován jako DBMS (Database Management System). Systém tvoří rozhraní mezi uloženými daty a aplikačním softwarem. Systémy řízení báze dat jsou využívány při tvorbě webových aplikací pro snadnou a efektivní práci s uloženými daty. Systémy řízení báze dat jsou klasifikovány podle toho, jaký databázový model podporují. V současné době mají největší zastoupení databázové modely relační, objektově-relační a objektový model. [70]

Systém řízení báze dat umožňuje provádět s daty několik operací. Mezi které patří vytváření nových datových záznamů, mazání a editaci záznamů. Dále správu klíčů, autentizaci, autorizaci, integritu dat a mnoho dalších funkcí. [70]

## 6.1 MySQL

MySQL je relační systém řízení báze dat (RDBMS) registrovaný pod licencí open source. Jedná se o druhý nejrozšířenější systém řízení báze dat na světě. Tento systém je založen na relačním databázovém modelu. MySQL využívá dotazovací jazyk SQL. [67, 71]

### 6.1.1 MySQL workbench

MySQL workbench je vizuální nástroj pro vývojáře a databázové architekty určený pro systém řízení báze dat MySQL. Umožňuje uživatelům vytvářet vizuální modely databází, spravovat databáze, spouštět SQL skripty, generovat SQL skripty na základě ER diagramu a obráceně. [72]

### 6.1.2 Relační databázový model

Data v relačním databázovém modelu jsou strukturována do jedné či více tabulek (relací). Tabulky (relace) jsou složeny z řádků a sloupců. Každý řádek tabulky obsahuje unikátní klíč, kterým je řádek identifikován. Sloupce tabulek jsou nazývány atributy, řádky tabulky jsou nazývány záznamy. Jednotlivé atributy musí mít definovaný datový typ. Každá tabulka reprezentuje určitý typ entity například *tabulka zvíře* reprezentuje celou množinu zvířat. Jednotlivé záznamy (řádky) pak reprezentují entity například: kůň, ovce, koza a apod. (viz obr. 13). [67]

*Obrázek 13 Ukázka relační tabulky*

| id | nazev_druhu | hmotnost | barva |
|---|---|---|---|
| 1 | ovce | 30 | cerna |
| 2 | kun | 150 | hneda |
| 3 | koza | 20 | bila |
| 4 | kun | 200 | cerna |

*Zdroj: autor*

Každý záznam v tabulce má svůj vlastní unikátní klíč (primární klíč). Řádky v jedné tabulce je možné propojit s řádky v jiných tabulkách. Přidáním atributu v tabulce druhé nazvanou jako cizí klíč. V záznamu druhé tabulky je pak v atributu cizí klíč uvedena hodnota unikátního klíče (primárního klíč) z tabulky první. [67]

## 6.2 PostgreSql

PostgreSql je další open-source systém řízení báze dat. Na rozdíl od MySQL se nejedná o čistě relační systém řízení báze dat (RDBMS), ale o objektově-relační databázový systém (ORDBMS) založený na objektově-relačním modelu. Tento systém byl vytvořen v roce 1996 na Californské univerzitě. Je vyvíjen především pro operační systém Linux, ale je dostupný i pro macOS server a Microsoft Windows. [66, 73]

### 6.2.1 Objektově-relační databázový model

Objektově-relační databázový model je kombinací relačního databázového modelu a objektového databázového modelu. Kombinací těchto modelů je dosažena podpora některých vlastností objektového modelu a vlastností relačního modelu. Objektově-relační model podporuje tabulkovou strukturu a datové typy jako relační model. Zároveň podporuje ukládaní objektů do databáze a dědičnost jako je tomu v objektovém modelu. [65, 73]

Velkou výhodou objektově-orientovaného modelu je odstranění složitosti komunikace mezi relačními databázemi a softwarem, jenž je naprogramován v objektovém či objektově-orientovaném jazyce. [65, 73]

# 7 Návrh praktického řešení a realizace

Návrh praktického řešení pro webovou aplikaci umožňující evidenci chovného dobytka a její realizaci je rozčleněn do několika menších kapitol, které popisují jednotlivé kroky k vytvoření webové aplikace. Vychází se zde z již zavedených postupů, které jsou využívány pro tvorbu softwaru. Konkrétně byl pro vývoj webové aplikace zvolen agilní přístup. U agilního přístupu je při vývoji kladen menší důraz na plánování a dokumentaci a více času je věnováno programování aplikace. Za použití agilního přístupu si webová aplikace prošla několika vývojovými cykly, kdy na konci každého cyklu byla do aplikace implementována určitá funkcionalita.

## 7.1 Analýza

V problematice analýzy týkající se vývoje webové aplikace pro evidenci chovného dobytka byla řešena jednotlivá úskalí vedení evidence chovného dobytka. Celá problematika byla rozdělena na několik menších problémů.

Prvním řešeným problémem bylo zjištění případné proveditelnosti, a to, zda za daných prostředků a v daném časovém rámci je možno webovou aplikaci pro evidenci chovného dobytka vytvořit a nasadit do ostrého provozu. Po důkladné analýze byl vyvozen závěr, že při optimálním nastavení požadavků na webovou aplikaci je její realizace proveditelná v daném časovém rámci.

Druhým řešeným problémem v analýze webové aplikace bylo zanalyzování, jak se chovný dobytek na území České republiky eviduje. Pro vytvoření analýzy tohoto problému byly použity dokumenty a legislativa z webových stránek Českomoravské společnosti chovatelů. Konkrétně tyto dokumenty: Metodika – Pokyny pro chovatele k vedení ÚE skotu platné od 1. 11. 2021 a Metodika – Pokyny pro chovatele k vedení ÚE ovcí a koz.

Ze zmíněných dokumentů byl vypracován základní přehled o způsobu evidenci chovného dobytka. Zahrnující, jaké druhy zvířat jsou evidovány a jaké atributy týkající se zvířat bude ve webové aplikaci případně potřeba uchovávat. Zjištěné informace z pokynu pro chovatele budou sloužit jako základ pro tvorbu databáze. Dále byly zanalyzovány silné stránky a slabé stránky způsobu evidence podle Českomoravské společnosti chovatelů, které poslouží, jako poklad pro návrh evidenčního systému.

## 7.2 Průzkum trhu

Po úspěšném vytvoření analýzy problému bylo přistoupeno k dalšímu kroku při vývoji webové aplikace. A to kroku týkajícího se zmapování a prozkoumání, jaké webové aplikace a jaký aplikační software je pro evidenci chovného dobytka aktuálně dostupný na trhu. S cílem zjištění požadavků potenciálního zákazníka na webovou aplikaci a jejich aktuálním uspokojením na trhu se zemědělským softwarem zaměřeným na evidenci chovného dobytka.

Nejdříve byl proveden průzkum specifikovaný na zemědělský software a webové aplikace poskytující evidenci chovného dobytka v českém jazyce. Byly vybrány a prozkoumány následující: aplikační software WinFAS – evidence zvířat, aplikační software AG info – evidence skotu, hospodářský registr skotu (ovcí, koz) a webová aplikace Stájový registr na Portálu Farmáře.

Dále byl proveden průzkum specifikovaný na zemědělský software a webové aplikace poskytující evidenci chovného dobytka v anglickém jazyce. Zde byla zmapována webová aplikace FarmBrite – livestock managment.

V průzkumu byly sledovány především tato kritéria: forma softwaru, počet funkcí, uživatelské rozhraní, druh evidovaných zvířat, cena. Výstup v podobě uceleného přehledu z průzkumu trhu umožní přesnější a optimálnější stanovení požadavků na webovou aplikaci. Výsledkem průzkumu trhu byl ucelený přehled o tom, jak je chovný dobytek evidován dostupným softwarem na trhu.

## 7.3 Definování požadavků

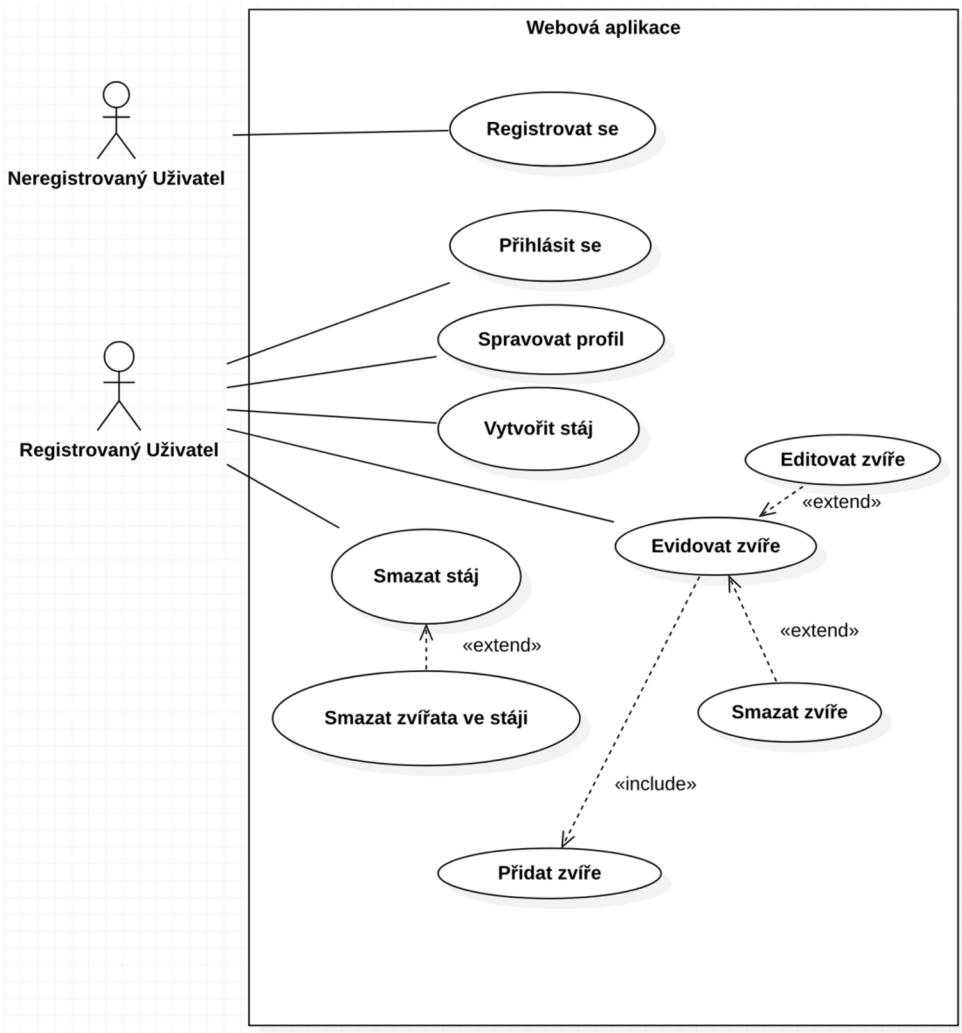
Ve vývojové části nazvané jako definování požadavků, byly definovány všechny požadavky, které má webová aplikace splňovat. Požadavky vycházeli především z poznatků zachycených při analýze a průzkumu trhu

### 7.3.1 Funkční požadavky

Pro zachycení funkcionalit systému byla použita technika pro zdokumentování a zjištění veškerých funkcionalit systému zvaná Use case (případ užití).

**Use Case diagram**

*Obrázek 14 Use case diagram*




*Zdroj: autor*

### 7.3.2 Designové požadavky

Při definování požadavků v oblasti designu webové aplikace, byl kladen důraz na jednoduchost ovládaní celé webové aplikace a její přehlednost. S cílem vytvořit webovou aplikaci s příjemným uživatelským rozhraním a uživatelským prožitkem.

Byly stanoveny následující požadavky na design:

- Dynamické načítání dat
- Minimalistický design
- Jednoduché ovládání

Tyto požadavky vycházeli především z analýzy některých dostupných softwarů na evidenci chovného dobytka, u kterých byly zjištěny nedostatky v přehlednosti celých systému a v komplikovanosti ovládání. Stanovením takových požadavků se má docílit odlišením od dostupného komerčního i nekomerčního softwaru na trhu.

### 7.3.3 Výkonnostní požadavky

Další důležitou oblastí požadavků na webovou aplikaci jsou požadavky vztahující se k výkonu webové aplikace. Proto při vývoji webové aplikace pro evidenci chovného dobytka, byly stanoveny následující výkonnostní požadavky na aplikaci vycházející z metrik nástroje Lighthouse od společnosti Google.

Prvním výkonnostním požadavkem je dosažení hodnot FCP (The First Contentful Paint) do $\leq$ 0,9 s. FCP určuje čas prvního vykresleného obsahu. Druhým výkonnostním požadavkem je dosažení hodnot TTI (Time To Interactive) do $\leq$ 2,5 s. TTI určuje čas kdy budou interaktivní prvky webové aplikace plně k použití. Třetím výkonnostním požadavkem je dosažení hodnoty Speed Indexu do $\leq$ 1,3 s. Speed index určuje, jak rychle je viditelná část stránky plně vykreslena.

### 7.3.4 Požadavek na škálovatelnost

Dalším požadavkem na webovou aplikaci pro evidenci chovného dobytka byl požadavek na škálovatelnost. Škálovatelností má být dosaženo jednoduchého rozšiřování webové aplikace do budoucna s možností snadné implementace nových funkcionalit do webové aplikace.

### 7.3.5 Požadavky na bezpečnost

Webová aplikace by měla dostatečně zabezpečit citlivá data všech uživatelů. A zajistit bezpečnou a robustní autentizaci a autorizaci uživatelů za použití ověřených a používaných standardů.

## 7.4 Návrh webové aplikace

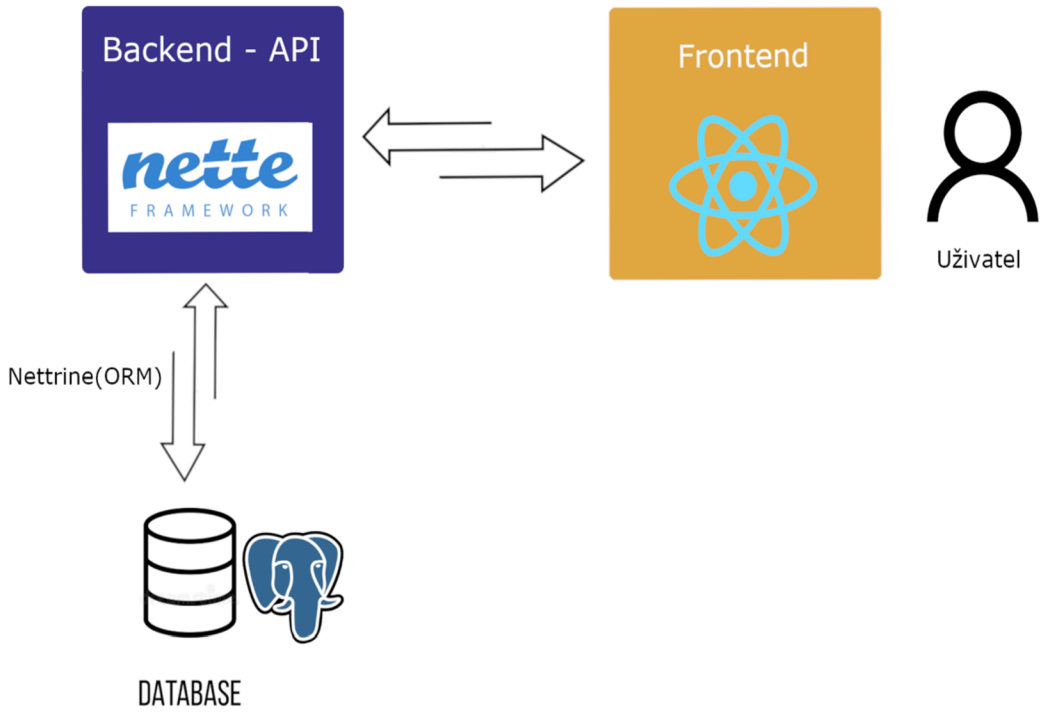
V kapitole návrh webové aplikace je vytvořen koncept webové aplikace zahrnující návrh architektury, entitně relační diagram a použité technologie. Cílem návrhu je vytvořit konceptuální model splňující vytyčené požadavky. Vytvořený konceptuální model bude sloužit pro samotnou implementaci webové aplikace.

### 7.4.1 Zvolená architektura a technologie

Návrh webové aplikace vycházel ze stanovených požadavků na webovou aplikaci. Na základě požadavků na aplikaci byl zvolen druh architektury webových aplikací SPA (Single-Page application). Architektura SPA zajistí dynamické načítání dat ze serveru do pohledové části aplikace.

Pro komunikaci mezi serverem a pohledovou části bylo zvoleno řešení vycházející z architektury REST API. Zvolení takové přístupu vyžadovalo vytvoření dvou oddělených systému postavených na architektuře klient-server a komunikujících pomocí HTTP požadavků (viz obr. 15). Systém na straně klienta je pohledovou (frontendovou) částí webové aplikace a systém na straně serveru je backendovou částí aplikace. Zvolením přístupu vycházejícího z architektury REST API, lze do budoucna webovou aplikaci snadno rozšířit o další funkcionality pomocí vytvoření dalších endpointů v backendové části.

*Obrázek 15 Zvolená architektura webové aplikace*




*Zdroj: autor*

### Frontendová část

Pro tvorbu pohledové (frontendové) části aplikace byla zvolena javascriptová knihovna React. Knihovna React splňovala všechny požadavky kladené na design a výkonnost webové aplikace. React zajistí dynamické vykreslování dat do webové aplikace získaných z jednotlivých API díky technologii virtuálního domu.

### Backendová část

Pro backendovou část aplikace byl zvolen PHP framework Nette. Nette framework zastává roli ideálního nástroje (technologie) pro vytvoření jednotlivých API webové aplikace. Dále také jako vhodný nástroj pro operace CRUD nad databází.

### Databáze

Pro databázovou část byl použit open-source systém řízení báze dat PostgreSQL. Komunikace mezi PostgreSQL a Nette frameworkem bude probíhat přes vrstvu ORM (objektově relačního mapování), kterou zajistí knihovna Nettrine. Nettrine umožní pracovat s jednotlivými záznamy v databázi jako s objekty.

### Autorizace a autentizace uživatelů

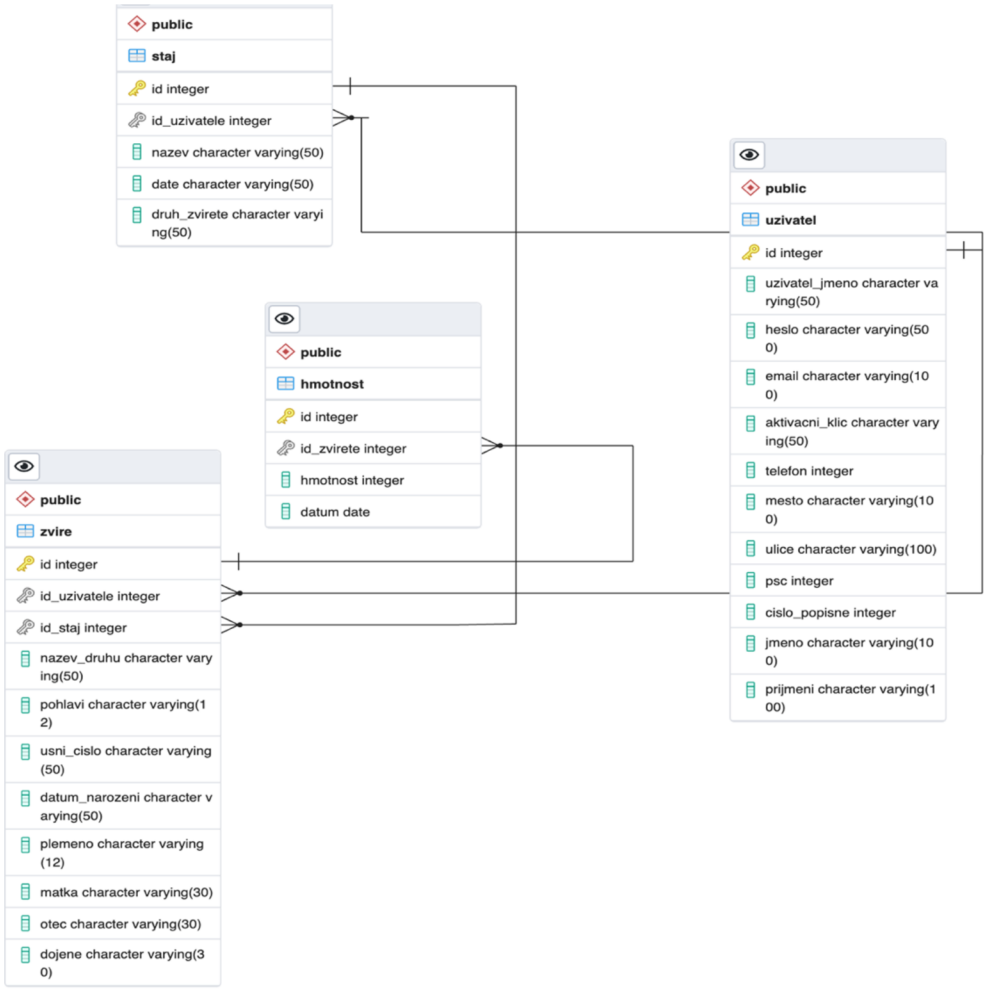
Pro autentizaci uživatelů byl zvolen přístup využívající technologii JWT (Json Web Token). Po odeslání přihlašovacích údajů a jejich ověření, budou pro uživatele na serveru vygenerovány dva JSON web tokeny. A to access token s krátkou časovou platností a refresh token s delší časovou platností. Oba tokeny budou uloženy do cookies webového prohlížeče.

Pro autorizaci budou využívány oba získané tokeny, kdy se s každým požadavkem uživatele na server zasílá access token, který je na serveru dekódován a verifikován. Pokud verifikace proběhne v pořádku je uživatel autorizován. V případě, že doba platnosti access tokenu expirovala, je na serveru dekódován refresh token a v případě jeho úspěšné verifikace vydán nový access token.

### Entitně relační diagram

Na entitně relačním diagramu je vyobrazen návrh databáze webové aplikace pro evidenci chovného dobytka. Návrh zahrnuje tabulky jejich atributy a vztahy mezi tabulkami.

*Obrázek 16 Entitně relační diagram*




*Zdroj: autor*

## 7.5 Implementace webové aplikace

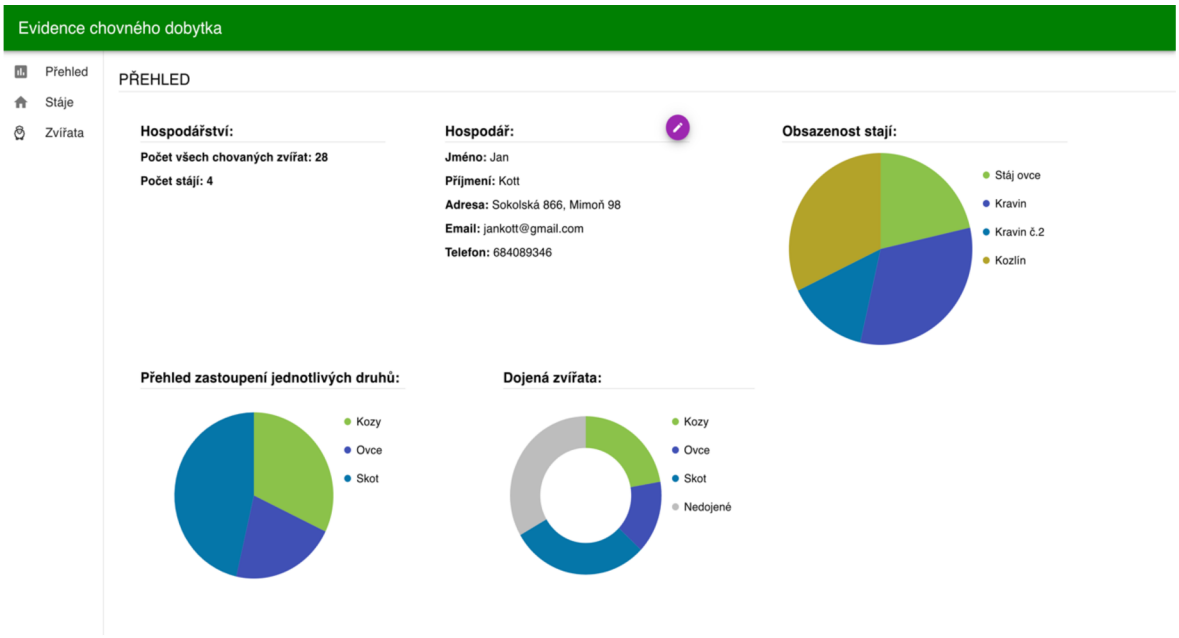
Implementace webové aplikace byl proces, kdy se z analýz, požadavků a návrhu realizovalo samotné řešení webové aplikace pro evidenci chovného dobytka. Byly implementovány dva systémy. A to systém zajišťující frontendovou část aplikace a systém zajišťující backendovou část aplikace.

### 7.5.1 Implementace frontendové části

Pohledová část webové aplikace byla naprogramována za pomoci knihovny React v jazyce JavaScript. Knihovna React nahradila vrstvu View v architektuře MVP používanou frameworkem Nette. V Reactu byly vytvořeny jednotlivé komponenty aplikace, ze kterých je pohledová část aplikace jako celek poskládána. Některé z komponent byly rozšířeny o funkci fetch umožňující komunikaci s API napsaných ve frameworku Nette.

Na obrázku 17 se nachází ukázka stránky přehled z pohledové části webové aplikace. Stránka přehled je složená z jednotlivých komponent vytvořených v Reactu.

*Obrázek 17 Ukázka stránky přehled*




*Zdroj: autor*

Na následujících ukázkách zdrojového kódu webové aplikace jsou vyobrazeny metody sloužící pro získávání dat, přidávání dat a mazání dat v databázi z pohledové části webové aplikace. Na ukázkách jsou všechny metody vztaženy ke komponentě stáj a tabulce stáj. Způsob, jakým jsou tyto metody implementovány pro komponentu stáj je shodný se způsoby implementace, jakými byli implementovány i ostatní komponenty v pohledové části.

### Implementace zobrazení stájí

Na ukázce (viz zdroj. kód 7) je vyobrazen útržek kódu z pohledové části aplikace znázorňující, jak probíhá získaní dat z vytvořeného API ve frameworku Nette za pomoci javascriptové funkce fetch s nastavenou HTTP metodou GET. Jedná se konkrétně o získaní všech stájí vlastněných daným uživatelem. HTTP požadavek je z pohledové části zaslán na příslušnou URL adresu (API), kde dochází ke zpracování požadavku. Při úspěšném zpracování požadavku dojde k odeslaní odpovědi ze serveru. Pohledová část obdrží odpověď se seznamem stájí uživatele ve formátu JSON. Stáje jsou poté uloženy do state komponenty Stáj a vykresleny ve webovém prohlížeči.

```
fetch(process.env.REACT_APP_APISERVER + "staj", {
      method: "GET",
      credentials: "include",
    })
      .then((res) => res.json())
      .then((result) => {
        this.setState({
          staj: result,
        });
      });
```

*Zdrojový kód 7 Fetch s metodou GET*

Získaná data v JSONu jsou obdržena jako pole objektů (viz obr. 18). Atributy objektů nesou data získaná z databáze, tyto data se přímo vykreslují ve webovém prohlížeči na příslušná místa. Hranaté závorky označují pole a složené závorky jednotlivé objekty (stáje).

*Obrázek 18 Získaná data v JSONu*


```
[
    {
        "id":5,
        "nazev":"Stáj ovce",
        "datum":"23. 3. 2022",
        "druh":"Ovce",
        "pocet_zvirat":6
    },
    {
        "id":6,
        "nazev":"Kravin",
        "datum":"23. 3. 2022",
        "druh":"Skot",
        "pocet_zvirat":9
    }
]
```


*Zdroj: autor*

### Implementace přidání stáje

Na ukázce je vyobrazena metoda createStaj (viz zdroj. kód 8) z pohledové části aplikace znázorňující, jak probíhá přidání nové stáje do databáze pomocí javascriptové funkce fetch s nastavenou HTTP metodou POST. HTTP požadavek je z pohledové části zaslán na příslušnou URL adresu (API), kde dochází ke zpracování požadavku. Tělo HTTP požadavku obsahuje název nové stáje a druh chovaných zvířat v nové stáji. Při úspěšném zpracování požadavku dojde k přidání nového záznamu v tabulce stáj a do pohledové části je odeslána HTTP odpověď obsahující aktualizovaný seznam stájí ve formátu JSON. Seznam stájí je uložen do state komponenty stáj a vykreslen webovém prohlížeči.

```
createStaj = async (event) => {
    event.preventDefault();
    this.setState({ open: !this.state.open });
    const data = new FormData(event.target);
    await hasAccess();
    fetch(process.env.REACT_APP_APISERVER + "staj", {
      method: "POST",
      credentials: "include",
      body: data,
    })
      .then((res) => res.json())
      .then((result) => {
        this.setState({
          staj: result,
        });
      });
  };
```

*Zdrojový kód 8 Metoda createStaj*

### Implementace smazání stáje

Na ukázce je zobrazena metoda deleteStaj (viz zdroj. kód 9) z pohledové části aplikace znázorňující, jak probíhá smazání stáje pomocí javascriptové funkce fetch s nastavenou HTTP metodou DELETE. HTTP požadavek je z pohledové části zaslán na příslušnou URL adresu (API), kde dochází ke zpracování požadavku. Při úspěšném zpracování požadavku dojde k smazání záznamu v tabulce stáj a do pohledové části je odeslána HTTP odpověď obsahující aktualizovaný seznam stájí ve formátu JSON. Seznam stájí je uložen do state komponenty stáj a vykreslen webovém prohlížeči.

```
deleteStaj = async (event) => {
    this.setState({ open: !this.state.open });
    await hasAccess();
    fetch(
      process.env.REACT_APP_APISERVER + "staj/" + this.state.oznacene_staje,
      {
        method: "DELETE",
        credentials: "include",
      }
    )
      .then((res) => res.json())
      .then((result) => {
        this.setState({
          staj: result,
        });
      });
  };
```

*Zdrojový kód 9 Metoda deleteStaj*

### 7.5.2 Implementace backednové části

V backendové části webové aplikace byla naprogramována jednotlivá API pro pohledovou část aplikace. Dále byla naprogramována vrstva ORM za pomoci knihovny Nettrine umožňující komunikaci s databázi. Jednotlivé presentery v Nette frameworku zachycují HTTP požadavky z pohledové části. Tyto požadavky se zpracovávají za pomoci modelů a vrstvy ORM. Po úspěšném zpracování požadavku jsou odeslána data ve formátu JSON zpět do pohledové části. Samotnému zpracováni vždy předchází autorizace.

Na následujících ukázkách kódu je zobrazeno, jak backendová část aplikace zpracovává požadavky zaslané z pohledové části aplikace. Jedná se konkrétně o ukázky kódu zpracovávající požadavky týkajících se stájí a tabulky stáj. Obdobným způsobem jsou implementovány i ostatní presentery a modely v backendové části webové aplikace.

#### Implementace získání stájí

Na ukázce (viz zdroj. kód 10) je vyobrazena část kódu PHP z presenteru Stáj. Tato část kódu zajišťuje zpracování HTTP požadavku s metodou GET z pohledové části aplikace (viz

implementace zobrazení stájí). Je-li požadavek úspěšně zpracován je do pohledové části aplikace odeslán seznám stájí daného uživatele ve formátu JSON.

```
// vrati staje uzivatele
if ($_SERVER['REQUEST_METHOD'] == 'GET' && !$cislo_staje) {
 $res = $this->stajrepo->vratStajeUzivatele2($id_uzivatel);
 $this->sendJson($res);
}
```

*Zdrojový kód 10 Zpracování HTTP požadavku GET*

Na ukázce (viz zdroj. kód 11) je vyobrazena část kódu z modelu StajRepository, část kódu získává záznamy z databáze konkrétně z tabulky stáj pomocí ORM a vrací je do presenteru Stáj. Funkce obsahující tuto část kódu v modelu StajRepository je volána v preseneteru Stáj, při zpracovávání požadavku GET.

```
$qb = $this->decorator->createQueryBuilder();
        $qb->select('u, COUNT(g)')
            ->from('App\Model\Entity\Staj', 'u')
            ->where('u.id_uzivatele = :id')
            ->setParameter('id', $id_uzivatele2)
            ->LeftJoin('App\Model\Entity\Zvire', 'g', 'WITH', 'u.id = g.id_staj')
            ->groupBy('g.id_staj, u');

        $query = $qb->getQuery();
        $res = $query->getArrayResult();
```

*Zdrojový kód 11 Získání záznamů z tabulky stáj*

### Implementace přidání stáje

Na ukázce (viz zdroj. kód 12) je vyobrazena část kódu PHP z presenteru Stáj. Tato část kódu zajišťuje zpracování HTTP požadavku s metodou POST z pohledové části aplikace. Je-li požadavek úspěšně zpracován dojde k přidání nového záznamu do tabulky stáj. A zároveň odeslání aktualizovaného seznamu stájí ve formátu JSON zpět do pohledové části.

```
//pridat staj
if ($_SERVER['REQUEST_METHOD'] == 'POST') {
      $this->stajrepo->pridatStaj($_POST['staj_name'],$id_uzivatel,
$_POST["vyber_zvirete"]);
       $res = $this->stajrepo->vratStajeUzivatele2($id_uzivatel);
       $this->sendJson($res);
}
```

*Zdrojový kód 12 Zpracování HTTP požadavku POST*

Na ukázce (viz zdroj. kód 13) je vyobrazena část kódu z modelu StajRepository, funkce pridatStaj provádí přidání nového záznamu do tabulky stáj pomocí ORM. Tato funkce je volána v presenetru Stáj, při zpracovávání požadavku POST.

```
public function pridatStaj($nazev, $id_uzivatele, $druh_zvirete): void
{
    $id_uzivatele =   $this->decorator->getRepository("App\Model\Entity\Uzivatel")-
>findOneBy(["id" => $id_uzivatele]);
    $my_date = date("j. n. Y");
    $staj = new Staj();
    $staj->setNazev($nazev);
    $staj->setDruh_zvirete($druh_zvirete);
    $staj->setId_uzivatele($id_uzivatele);
    $staj->setDatum($my_date);

    $this->decorator->persist($staj);

    $this->decorator->flush();
}
```

*Zdrojový kód 13 Funkce pridatStaj*

### Implementace smazání stáje

Na ukázce (viz zdroj. kód 14) je vyobrazena část kódu PHP z presenteru Stáj. Tato část kódu zajišťuje zpracování HTTP požadavku s metodou DELETE z pohledové části aplikace. Je-li požadavek úspěšně zpracován dojde k smazání záznamu z tabulky stáj. A zároveň odeslání aktualizovaného seznamu stájí ve formátu JSON zpět do pohledové části.

```
//smaze staj
            if ($_SERVER["REQUEST_METHOD"] == "DELETE") {
                $res = $this->stajrepo->smazatStaj($id_uzivatel, $cislo_staje);
                $res = $this->stajrepo->vratStajeUzivatele2($id_uzivatel);
                $this->sendJson($res);
            }
```

*Zdrojový kód 14 Zpracování požadavku HTTP DELETE*

Na ukázce (viz zdroj. kód 15) je vyobrazena metoda smazatStaj z modelu StajRepository, funkce smazatStaj provádí smazání záznamu z tabulky stáj pomocí ORM. Tato funkce je volána v presenetru Stáj, při zpracovávání požadavku DELETE.

```
public function smazatStaj($id_uzivatele, $id_staje)
{
        $id_uzivatele=$this->decorator->getRepository("App\Model\Entity\Uzivatel")-
>findOneBy(["id" => $id_uzivatele]);

        $query = $this->decorator->createQuery('DELETE App\Model\Entity\Staj u WHERE
u.id = :id AND u.id_uzivatele= :id_uzivatele');
        $query->setParameters(array(
            'id' => $id_staje,
            'id_uzivatele' => $id_uzivatele
        ));
        return $query->getResult();
}
```

*Zdrojový kód 15 Metoda smazatStaj*

## 7.6 Testování

Po dokončení implementace byla webová aplikace otestována. Testování webové aplikace zahrnovalo kontrolu zranitelností a kontrolu správného fungování jednotlivých funkcionalit webové aplikace. Kontrola zranitelností zahrnovala otestování aplikace vůči útokům SQL injection, Broken Access Control, Cryptographic Failures a Server Misconfiguration. Kontrola fungování jednotlivých funkcionalit webové aplikace spočívala v otestování veškerých vytyčených funkcionalit z Use Case diagramu a ověření jejich správného fungování po implementaci webové aplikace.

## 7.7 Nasazení aplikace

Po dokončení implementace a testování byla webová aplikace pro evidenci chovného dobytka uvedena do produkčního režimu. Pro nasazení do produkčního režimu bylo zvoleno cloudové řešení Google Cloud Platform. Oba systémy (frontend a backend) byly nahrány na výpočetní platformu Google App Engine. Google App Engine je založen na modelu PaaS a zajišťuje hostování webových aplikací společností Google. Výhodou tohoto řešení je téměř neomezená škálovatelnost a ušetření starostí ohledně správy výpočetní infrastruktury.

# 8 Závěr

V této diplomové práci byla vytvořena webová aplikace pro evidenci chovného dobytka a popsána problematika vývoje moderních webových aplikací. V rámci teoretické části práce byl vytvořen přehled o pojmu webová aplikace. Dále byly popsány použité technologie pro tvorbu webové aplikace a definovány architektury a přístupy. V poslední kapitole teoretické části diplomové práce byla rozebrána problematika ukládání dat. V praktické části diplomové práce byl popsán postup vývoje webové aplikace pro evidenci chovného dobytka. Popis postupu zahrnoval jednotlivé kroky vývoje od analýzy až po nasazení do produkčního režimu. Dále byla vytvořena webová aplikace na základě stanovených požadavků a cílů. Vytvořená aplikace splňuje stanovené nároky na výkonnost, design, ovládání, bezpečnost a škálovatelnost.

Webová aplikace pro evidenci chovného dobytka je přínosem pro zemědělské subjekty, které se věnují chovu chovného dobytka. Přínos vytvořené webové aplikace spočívá v možnosti online evidence, která je dostupná z více míst a zařízení skrze internetovou síť. Což v případě desktopových aplikací a papírově vedené evidence není možné. Webová aplikace je postavena na moderních technologiích, díky kterým je obsah načítán dynamicky a velmi rychle. Z hlediska designu je aplikace navržena tak, aby bylo ovládání co nejjednodušší a uživatelsky přívětivé. Z bezpečnostního hlediska je webová aplikace navržena tak, aby co nejlépe chránila citlivá data uživatelů. Poslední velkou výhodou je snadná škálovatelnost celé aplikace a modulárnost, které bylo dosaženou díky použité architektuře a nasazením do produkčního režimu za pomoci cloud computingových služeb.

Do budoucna lze vytvořenou webovou aplikace snadno rozšířit i o další funkcionality. Mezi možná rozšíření připadá v úvahu například komunikace s čtečkami, váhami a případně dalšími zařízeními v zemědělství skrze internet věcí.

# 9 Seznam použitých zdrojů


1. WEB, WEBOVÁ STRÁNKA A WEBOVÁ APLIKACE, V ČEM JE ROZDÍL?. Rascasone [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://www.rascasone.com/cs/blog/web-webova-aplikace-rozdil

2. Informace o webových aplikacích. Adobe [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html

3. Webová aplikace. Wikipedia [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Webov%C3%A1_aplikace

4. Frontend vs Backend. GeekforGeeks [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://www.geeksforgeeks.org/frontend-vs-backend/

5. From History of Web Application Development. Devsaran [online]. 2016 [cit. 2022-03-26]. Dostupné z: https://www.devsaran.com/blog/history-web-application-development

6. Web application. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://en.wikipedia.org/wiki/Web_application

7. Difference Between Web application and Website. GeekforGeeks [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.geeksforgeeks.org/difference-between-web-application-and-website/

8. Webová stránka. Wikipedia [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Webov%C3%A1_str%C3%A1nka

9. OWASP Top Ten. Owasp [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://owasp.org/www-project-top-ten/

10. 10 nejzávažnějších zranitelností webových aplikací podle OWASP. Zdroják [online]. 2018 [cit. 2022-03-26]. Dostupné z: https://zdrojak.cz/clanky/10-nejzavaznejsich-zranitelnosti-webovych-aplikaci-podle-owasp/

11. HTML Introduction. W3schools [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.w3schools.com/html/html_intro.asp

12. HTML příručka. Jakpsatweb [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://www.jakpsatweb.cz/html/

13. Introduction to the DOM. Mdn [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction
14. Document Object Model. Wikipedia [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Document_Object_Model

15. HTML5: co přináší a proč se o něj zajímat. Root [online]. 2012 [cit. 2022-03-26]. Dostupné z: https://www.root.cz/clanky/html5-co-prinasi-a-proc-se-o-nej-zajimat/CSS3

16. CSS. Jakpsatweb [online]. [cit. 2022-03-26]. Dostupné z: https://www.jakpsatweb.cz/css/

17. Kaskádové styly. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Kask%C3%A1dov%C3%A9_styly

18. 10 CSS3 Properties You Need to Be Familiar With. *Envatotuts* [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://code.tutsplus.com/tutorials/10-css3-properties-you-need-to-be-familiar-with--net-16417

19. CSS3. Wikipedia [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/CSS3

20. JAVASCRIPT PRO ZAČÁTEČNÍKY: CO TO JE A JAK FUNGUJE. Rascasone [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.rascasone.com/cs/blog/co-je-javascript-pro-zacatecniky

21. JavaScript. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/JavaScript

22. Lekce 1 - Úvod do JavaScriptu. Itnetwork [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://www.itnetwork.cz/javascript/zaklady/javascript-tutorial-uvod-do-javascriptu-nepochopeny-jazyk

23. JavaScript language resources. Mdn [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Language_Resources

24. WHAT IS A JAVASCRIPT FRAMEWORK?. GeneralAssembly [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://generalassemb.ly/blog/what-is-a-javascript-framework/

25. Introduction. Vue.js [online]. [cit. 2022-03-26]. Dostupné z: https://vuejs.org/guide/introduction.html

26. Vue.js. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Vue.js

27. Lekce 1 - Úvod do jQuery. Itnetwork [online]. 2013 [cit. 2022-03-26]. Dostupné z: https://www.itnetwork.cz/javascript/jquery-zaklady/javascript-tutorial-funkcionalni-programovani-a-jquery-webova-kalkulacka

28. Pros and Cons of ReactJS. Javapoint [online]. [cit. 2022-03-26]. Dostupné z: https://www.javatpoint.com/pros-and-cons-of-react

29. React (webový framework). Wikipedia [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/React_(webov%C3%BD_framework)

30. Lekce 1 - Úvod do React. Itnetwork [online]. 2019 [cit. 2022-03-26]. Dostupné z: https://www.itnetwork.cz/javascript/react/zaklady/uvod-do-react/

31. What Is TypeScript? Pros and Cons of TypeScript vs. JavaScript. Stxnext [online]. [cit. 2022-03-26]. Dostupné z: https://www.stxnext.com/blog/typescript-pros-cons-javascript/

32. Lekce 1 - Úvod do TypeScriptu. *Itnetwork* [online]. 2018 [cit. 2022-03-26]. Dostupné z: https://www.itnetwork.cz/javascript/typescript/uvod-do-typescriptu

33. Difference between TypeScript and JavaScript. *GeekforGeeks* [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.geeksforgeeks.org/difference-between-typescript-and-javascript/

34. PHP. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/PHP

35. PHP. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://en.wikipedia.org/wiki/PHP

36. Top 6 Advantages Of Php Over Other Programming Languages. GoodWork [online]. 2015 [cit. 2022-03-26]. Dostupné z: https://www.goodworklabs.com/top-6-advantages-of-php-over-other-programming-languages/

37. Advantages and Disadvantages of PHP. GeekforGeeks [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-php/

38. Proč používat Nette?. Nette [online]. [cit. 2022-03-26]. Dostupné z: https://doc.nette.org/cs/10-reasons-why-nette

39. Nette Framework. Nette [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Nette_Framework

40. Jazyk SQL. Voho [online]. [cit. 2022-03-26]. Dostupné z: http://voho.eu/wiki/sql/

41. Příkazy jazyka SQL. Wikipedia [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/P%C5%99%C3%ADkazy_jazyka_SQL#P%C5%99%C3%ADkazy_pro_definici_dat

42. Difference Between GitLab and GitHub. GeekforGeeks [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://www.geeksforgeeks.org/difference-between-gitlab-and-github/

43. GitLab. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/GitLab

44. Git. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Git

45. Centralized vs Distributed Version Control: Which One Should We Choose?. GeekforGeeks [online]. 2021 [cit. 2022-03-26]. Dostupné z:

https://www.geeksforgeeks.org/centralized-vs-distributed-version-control-which-one-should-we-choose/

46. Client Server Architecture. Cio-wiki [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://cio-wiki.org/wiki/Client_Server_Architecture

47. Client–server model. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://en.wikipedia.org/wiki/Client%E2%80%93server_model

48. What is a web server?. Mdn [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

49. What is REST. Restfulapi [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://restfulapi.net/

50. Representational State Transfer. Wikipedia [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Representational_State_Transfer

51. What is a RESTful API?. Mulesoft [online]. [cit. 2022-03-26]. Dostupné z: https://www.mulesoft.com/resources/api/restful-api

52. HTTP request methods. Mdn [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods

53. MVC architektura. Itnetwork [online]. 2013 [cit. 2022-03-26]. Dostupné z: https://www.itnetwork.cz/navrh/mvc-architektura-navrhovy-vzor

54. CO JE JEDNOSTRÁNKOVÁ WEBOVÁ APLIKACE (SPA) A KDY JI VYUŽÍT?. Rascasone [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://www.rascasone.com/cs/blog/jednostrankova-webova-aplikace-spa

55. AJAX Introduction. W3schools [online]. [cit. 2022-03-26]. Dostupné z: https://www.w3schools.com/js/js_ajax_intro.asp

56. AJAX. Wikipedia [online]. 2021 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/AJAX

57. JSON - Introduction. W3schools [online]. [cit. 2022-03-26]. Dostupné z: https://www.w3schools.com/js/js_json_intro.asp

58. JSON : jednotný formát pro výměnu dat. Zdroják [online]. 2008 [cit. 2022-03-26]. Dostupné z: https://zdrojak.cz/clanky/json-jednotny-format-pro-vymenu-dat/

59. Introduction to JSON Web Tokens. Jwt [online]. [cit. 2022-03-26]. Dostupné z: https://jwt.io/introduction

60. Cloud Computing: Co to je a komu se vyplatí. Algotech [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://www.algotech.cz/novinky/2020-04-21-cloud-computing-co-to-je-a-komu-se-vyplati

61. Cloud Computing. Investopedia [online]. 2020 [cit. 2022-03-26]. Dostupné z: https://www.investopedia.com/terms/c/cloud-computing.asp

62. Cloud Computing. Wikipedia [online]. 2022 [cit. 2022-03-26]. Dostupné z: https://cs.wikipedia.org/wiki/Cloud_computing

63. What is Cloud Computing?. Oracle [online]. [cit. 2022-03-26]. Dostupné z: https://www.oracle.com/cz/cloud/what-is-cloud-computing/

64. Co jsou veřejné, privátní a hybridní cloudy?. Azure [online]. [cit. 2022-03-26]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-are-private-public-hybrid-clouds/#benefits

65. Object-relational Data Model. Tutorialspoint [online]. 2018 [cit. 2022-03-26]. Dostupné z: https://www.tutorialspoint.com/Object-relational-Data-Model

66. PostgreSQL is an ORDBMS. What does that mean?. ArcType [online]. 2021 [cit. 2022-03-28]. Dostupné z: https://arctype.com/blog/postgres-ordbms-explainer/

67. WELLING, Luke a Laura THOMSON. PHP a MySQL Kompletní průvodce vývojáře. Albatros media, 2017. ISBN 9788025148921.

68. What Is GitHub? A Beginner's Introduction to GitHub. Kinsta [online]. 2021 [cit. 2022-03-29]. Dostupné z: https://kinsta.com/knowledgebase/what-is-github/

69. GitHub. Wikipedia [online]. 2022 [cit. 2022-03-29]. Dostupné z: https://en.wikipedia.org/wiki/GitHub

70. Systém řízení báze dat. Wikisofia [online]. [cit. 2022-03-29]. Dostupné z: https://wikisofia.cz/wiki/Syst%C3%A9m_%C5%99%C3%ADzen%C3%AD_b%C3%A1ze_dat

71. MySQL. Wikipedia [online]. 2022 [cit. 2022-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/MySQL

72. MySQL workbench. MySQL [online]. [cit. 2022-03-29]. Dostupné z: https://www.mysql.com/products/workbench/

73. PostgreSQL. Wikipedia [online]. 2022 [cit. 2022-03-29]. Dostupné z: https://cs.wikipedia.org/wiki/PostgreSQL

74. Stavové kódy a hlášení v odpovědi protokolu HTTP. Interval [online]. 2002 [cit. 2022-03-31]. Dostupné z: https://www.interval.cz/clanky/stavove-kody-a-hlaseni-v-odpovedi-protokolu-http/