Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Chytrá aplikace pro správu fotografií

Bakalářská práce

Autor: Daniel Strach

Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Michal Macinka

Hradec Králové duben 2021

Prohlášení:

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

vlastnoruční podpis

V Hradci Králové 30.04.2021 Daniel Strach

Poděkování:

Děkuji vedoucímu bakalářské práce Ing. Michalu Macinkovi za metodické vedení, za pomoc a rady při zpracování této práce.

Anotace

Tato bakalářská práce se zabývá představením problematiky vzniklé při návrhu a implementaci webových aplikací. Práce je rozdělena do pěti částí. První část je věnována popisu technologií pro tvorbu webových stránek. Druhá část seznámí čtenáře se stručnou historií zpracování obrazových dat a způsobu jejich využití. Třetí část se zabývá přehledem platforem, na kterých mohou být aplikace vyvíjeny. Čtenář se seznámí s jejich výhodami a nevýhodami. Ve čtvrté části je věnována pozornost návrhu aplikace pro správu fotografií. Jsou zde představeny řešení datové struktury, případy užití a grafický návrh aplikace. Poslední část je věnována problémům, které vznikly při implementaci aplikace pro správu fotografií. Jedná se o rozeznání objektů v obraze, zabezpečení aplikace a další.

Annotation

Title: Smart Application for Photos Management

This bachelor thesis deals with the issues arising in the design and implementation of web applications. The work is divided into five parts. The first part is devoted to the description of technologies for creating websites. The second part introduces the reader to a brief history of image data processing and how to use them. The third part deals with an overview of platforms on which applications can be developed. The reader will get acquainted with their advantages and disadvantages. In the fourth part, attention is paid to the design of a photo management application. Data structure solutions, use cases and graphical application design are presented here. The last part is devoted to problems that arose during the implementation of a photo management application. These include object recognition in the image, application security and more.

Obsah

1 Úvod

V dnešní době používá chytré mobilní zařízení takřka 50 % procent světové populace. Pro rok 2021 je počet uživatelů přibližně 3 800 000 000 [1]. Fotoaparát se stal neodmyslitelnou součástí tohoto přístroje. Každý rok se světoví výrobci předhání ve stálém zdokonalování fotoaparátu v zařízení spolu s přidáváním nových funkcí. Chytré mobilní telefony už v některých odvětvích dokonce nahradily známé zrcadlové fotoaparáty používané zejména profesionálními fotografy. Není se čemu divit – chytré zařízení je skladné a snadné na obsluhu.

Trendem posledního desetiletí se stalo zachytávání každodenních okamžiků v našich životech. Každý si přeje své životní události zvěčnit a jednoho dne se k nim vrátit. Mobilní zařízení mají ovšem několik nevýhod. Jednou z nich je omezená velikost úložiště, která se ve většině případů nedá fyzicky navýšit. Další nevýhodou je vyhledání konkrétní fotografie. Snadno se stane, že uživatel zapomene, kdy byla fotografie pořízena, a dokáže si pouze vybavit, co na fotografii bylo. V tomto případě by uživatel musel projít stovky, ne-li tisíce fotografií, než najde fotografii, kterou hledá. Snadno se může stát, že mezi všemi fotografiemi tu hledanou přehlédne.

Tato práce se zabývá implementací webové aplikace pro správu fotografií, která řeší výše zmíněné problémy. Aplikace nabídne uživatelům možnost nahrát své fotografie na úložiště. Po nahrání mohou fotografie odstranit ze svého zařízení a tím vytvořit místo pro nové. Aplikace také umožní uživatelům setřídit fotografie do galerií pro lepší přehled. Každá fotografie může mít přiřazený „tag“, který popisuje objekt na snímku. Pomocí vyhledávače může uživatel nalézt snímky s konkrétním popisem. Pomocí správného výrazu zobrazí aplikace všechny fotografie, na kterých je třeba strom.

2 Cíl práce

Cílem bakalářské práce je navrhnout a implementovat chytrou aplikaci pro správu fotografií. Aplikace si klade za hlavní cíl usnadněné vyhledávání a snadnou organizaci fotografií.

Součástí práce bude analýza a zvolení vhodných technologií, na kterých bude aplikace implementována. Pozornost je především věnována platformě webových aplikací, kde budou analyzovány a vysvětleny technologie na straně serveru i klienta. Práce uvede čtenáře do základů zpracování obrazu, jeho stručné historii a základních formátů.

V částech Návrh aplikace a Implementace aplikace bude rozebrána problematika při navrhování a následné implementaci aplikace. Každý problém, který v jednotlivých částech vývoje nastal, bude vysvětlen a následně bude představeno řešení daného problému.

Výstupem praktické části práce bude aplikace, která slouží pro správu fotografií. Aplikace bude postavena na technologiích, které budou vybrány z analýzy.

3 Vývoj aplikací

Vývoj softwaru je komplexní projekt jedné i více osob s jasně vyznačenými hranicemi času, finančního rozpočtu a lidskými zdroji, které produkují nový nebo vylepšený zdrojový kód, který přidává značnou obchodní hodnotu k novému nebo již existujícímu obchodnímu procesu. [2]

Tato práce se zabývá vývojem softwaru postaveného na architektuře klient-server. Jak už název napovídá, veškerá funkcionalita je rozdělena mezi dvě základní komponenty - klienta a server [3]. Tyto komponenty se mohou nacházet současně v jednom fyzickém systému, ale mnohem častěji jsou tyto komponenty rozděleny a komunikují spolu skrze počítačovou síť. Komunikace mezi klientem a serverem je poté založena na principu požadavek-odpověď (request-response) a řídí se běžnými komunikačními protokoly, které definují formální pravidla, jazyk a dialogové vzory. Nejběžněji používaný protokol pro komunikaci je TCP/IP – sada standardizovaných pravidel, která umožňují počítačům komunikovat v síti, jako je internet. [4]

3.1 Nástroje pro podporu vývoje

K vývoji moderní a kvalitní webové aplikace je nutné si vybrat správné technologie. V této kapitole budou popsány technologie, které budou následně využity při implementaci aplikace.

3.1.1 NPM

Node Package Manager (dále jen NPM) je největší softwarový registr JS balíčků. Slouží open-source vývojářům k sdílení a stažení těchto balíčků [5]. NPM bylo původně napsáno pro správu balíčků v prostředí Node.js, kterého je NPM nedílnou součástí. V dnešní době spravuje i všechny balíčky pro klienta (frontend) a je z něho možné spouštět další užitečné nástroje pomocí příkazové řádky [5]. Jako alternativu k NPM je možné zvolit Yarn.

NPM se skládá ze třech částí:

- Webová stránka – umožňuje vývojářům vyhledávat potřebné balíčky. Nabízí možnost vytvoření profilu a správy vlastních balíčků. Vývojářům jsou poskytnuty informace jako jsou popularita, kvalita a četnost údržby daného balíčku nebo seznam dalších balíčků, které jsou nutné k funkci určitého balíčku. [5]
- CLI – příkazový řádek slouží k instalaci a manipulaci balíčků přímo ve vyvíjené aplikaci.

- Registr – jedná se o databázi veřejně dostupného softwaru psaném v JS spolu s dalšími důležitými informacemi.

Struktura souborů

├── node_modules

├── package.json

├── package-lock.json

node_modules – složka, ve které jsou nainstalované balíčky. Je potřeba správně nastavit soubor `.gitignore`, který dovoluje uživateli nastavit složky a soubory, které mají být záměrně ignorovány [6]. V opačném případě by došlo k zbytečnému nahrávání poměrně velkých souborů na Git.

Package.json – jedná se o soubor s konfigurací NPM. Obsahuje základní informace o projektu a seznam veškerých potřebných balíčků.

Package-lock.json – slouží k zachování jednotlivých verzí balíčků, aby byla zaručena kompatibilita pro všechny členy týmu, kteří na projektu pracují.

Inicializace projektu

K inicializaci nového projektu slouží příkaz `npm init`. Projekt může vzniknout i zkopírováním souboru `package.json` z jiného projektu. Po zadání příkazu je v konzoli spuštěn dotazník, ve kterém je možné vyplnit autora, název balíčku, popis, verzi projektu a další. Po dokončení se v souborové struktuře objeví soubor `package.json`.

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "REST server for web-app Photom",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Daniel Strach",
  "license": "ISC",
}
```

Ukázka kódu 1 – Obsah souboru package.json. Zdroj: [Autor].

3.1.2 REST

Representional State Transfer (dále jen REST). Ta to zkratka může být vyložena jako: reprezentace dat klientovi ve formátu, který je pro něj nejvhodnější. Jedná se o architektonický styl, který poskytuje standardy komunikace pro model klient-server. Tento architektonický typ poprvé představil ve své disertační práci Roy Fielding a to roku 2000. [7]

REST je postaven na řadě architektonických principů:

- Nezávislost serveru na klientovi – kód na straně klienta lze kdykoli změnit bez možného dopadu na provoz serveru. Klient může být okamžitě nahrazen jinou technologií, protože rozhraní zůstává neměnné a z tohoto důvodu nedochází k uložení stavu klienta na serveru [8].
- Jedinečnost adres zdrojů – kterákoli jednotka dat (včetně libovolné úrovně vnoření) má přesně definovanou jedinečnou adresu URL, která je jednoznačným identifikátorem zdroje [7].
- Nezávislost formátu – jedná se o nezávislost formátu ukládání dat na formátu jejich přenosu. Server může obsahovat celou množinu formátů (JSON, XML atd.) pro zaručení největší kompatibility. Data ovšem musejí být ukládána ve vlastním vnitřním formátu bez ohledu podpory dalších formátů [8].
- Metadata v odpovědi – spolu se samotnými daty v odpovědi musejí být klientovi poskytnuty další informace. Server tedy vrací podrobnosti ohledně zpracování požadavku jako jsou zprávy o chybách, vlastnosti zdroje či počet záznamů pro správné zobrazení navigace na klientovi [7].

Tyto principy umožňují, aby jeden server mohlo využívat několik klientů postavených na rozdílných technologií a běžících na různých platformách.

3.1.3 Babel

Babel je překladač (compiler) zdrojového kódu psaného podle normy ECMAScript 2015+ do starší verze, aby byla zaručena podpora starších prohlížečů a prostředí. [9]

```
// Zápis funkce v ES6
var a = () => {};
var a = (b) => b;

// Zápis funkce map pomocí ES6 zápisu.
const double = [1,2,3].map((num) => num * 2);
console.log(double); // [2,4,6]
```

Ukázka kódu 2 – Zdrojový kód psaný dle standardu ES6 před použitím překladače Babel. Zdroj: [10].

```
// Funkce byla konvertována na ES5
var a = function () {};
var a = function (b) {
  return b;
};

// Konverze funkce map na starší verzi ES5
const double = [1, 2, 3].map(function (num) {
  return num * 2;
});
console.log(double); // [2,4,6]
```

Ukázka kódu 3 – Zdrojový kód po překladu skrze Babel. Zdroj: [10].

3.1.4 Webpack

Webpack je technologie, která dovoluje vytvářet statické svazky modulů. Webpack provede analýzu struktury balíčků, na jejíž základě vytvoří graf závislostí (dependency graph). Ten obsahuje několik modulů, které jsou nutné pro správné fungování aplikace. Na základě tohoto grafu je vytvořen nový balíček, který je vytvořen z naprostého minima potřebných souborů mnohdy pouze jednoho souboru bundle.js, který může být velice snadno začleněn do HTML souboru. [11]

3.1.5 JSON

JavaScript Object Notation (dále jen JSON) je formát pro výměnu dat založený na jazyce JS. Díky tomuto formátu je umožněn přenos objektů a dalších datových struktur. JSON slouží především k serializaci (proces, který převádí libovolně složitý objekt do jeho jednorozměrné podoby.) a následnému přenosu strukturovaných dat mezi serverem a klientem pomocí síťového připojení. JSON je hlavním způsobem výměny dat u jednostránkových aplikací. [12]

Soubory využívající formát JSON lze poznat podle přípony `.json`. Jednou z výhod tohoto formátu je jeho snadnost přenosu mezi klientem a serverem [12]. Ovšem jeho hlavní vlastností je vysoká kompatibilita mezi různými programovacími jazyky. Na serveru a klientovi mohou být použity velice rozdílné technologie, ale JSON zaručí konzistentní komunikaci. [13]

Douglas Crockford navrhl a zpopularizoval tento formát. Původ formátu byl úzce spjatý s JS, ale je plně nezávislý na tomto programovacím jazyce a může být použit takřka s libovolným programovacím jazykem [12]. JSON je postaven na obecných programovacích konvencích, které jsou známé většině programátorů [13].

JSON svými výhodami nahrazuje Extensible Markup Language (dále jen XML). K zapsání dat potřebuje méně kódu a výsledný soubor má menší velikost. Díky tomu je rychlost zpracování a přenosu nižší. Zásadní rozdíl je ve čtení těchto dvou formátů. JSON byl navržen na základě textu a umožňuje snadné získání dat i pouhým okem. XML soubor se musí syntakticky analyzovat pomocí XML analyzátoru. Ke čtení formátu JSON stačí standardní funkce JS. JSON může do své struktury zahrnout pole. [13]

3.2 Technologie na straně klienta

Frameworky jsou softwarové struktury, nad kterými jsou vyvíjena robustní softwarová řešení. Pyšní se především znuvupoužitelností částí systému a signifikantní úsporou času při vývoji. Na druhou stranu jsme nuceni přistoupit na jeho restrikce a syntaxi. V dnešní době se veliké oblibě těší frameworky napsané v jazyce JavaScript (dále jen JS) a z nich především tyto tři technologická řešení: React (knihovna), Vue.js a Angular.

3.2.1 React

React je JS knihovna používaná k vývoji uživatelského rozhraní (UI) webových aplikací s použitím interaktivních prvků. K dnešnímu dni je knihovna React spravována společností Facebook. Vznikla v roce 2011 a jejím autorem je Jorda Walke. [14]

Tato knihovna klade důraz na znovupoužitelnost částí kódu. Tuto část kódu poté nazýváme „komponenta“. Každou funkcionalitu webu se tedy snažíme dekonstruovat na menší komponenty, což umožňuje produkovat kód rychleji a s menší šancí na vznik chyb. Za zmínku stojí především dvě funkcionality, na kterých je React postaven – JSX a Virtual DOM.

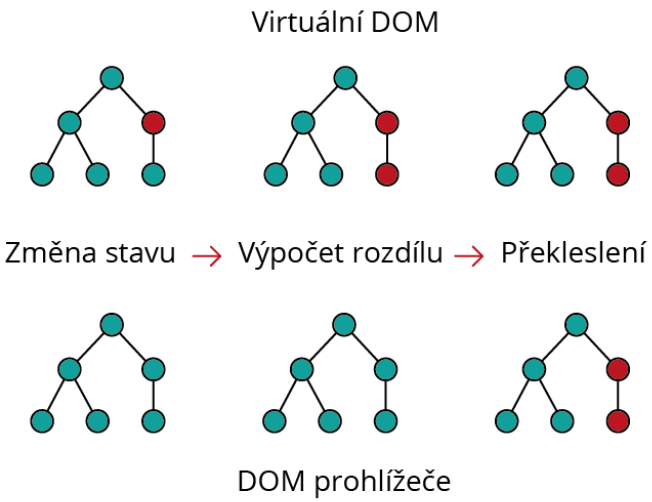
JSX

Každá webová stránka je v základu tvořena dokumenty psanými v jazyce HTML. Po otevření dokumentu ve webovém prohlížeči se vytvoří takzvaný „Document Object Model“ (dále jen DOM). Jedná se o reprezentaci složení webu pomocí stromové struktury. DOM je spojen s procesy, při kterých je možné měnit elementy a hodnoty na stránce bez toho, aniž by bylo nutné ručně aktualizovat stránku. To vše díky využití skriptovacího jazyka jako je JS.

JavaScript eXtension (dále jen JSX) je rozšíření, které React využívá, aby usnadnil vývojářům práci s úpravou DOM pomocí kódu, který je založen na syntaxi jazyku HTML. Používání JSX k manipulaci s DOM vede k zvýšení výkonu a efektivitě, a to díky s ním spojené technologii „Virtual DOM“.

Virtual DOM

Problém s běžným DOM je ten, že při jakékoliv změně dojde ke kompletnímu překreslení DOM. Pokud je při vývoji využita technologie JSX, tak je vytvořen tzv. Virtual DOM. Jak už název napovídá, dojde k vytvoření kopie DOM. React tuto kopii využívá k porovnávání s pravým DOM, jestli nedošlo k vyvolání události, která by způsobila změnu (třeba kliknutí na klávesu enter). Technologii Virtual DOM využívají i ostatní JS frameworky. [15]



Obrázek 1 - Stromová reprezentace vykreslování elementů DOM. Zdroj: [Autor].

Pro představu může uživatel vstoupit na stránky s filmovou kritikou a chce k filmu, který ho zaujal, napsat komentář. Do formuláře napíše jeho dojmy a stiskne tlačítko „Komentovat“. Bez použití knihovny React by se musela celá stránka obnovit, a to bude

stát čas a zdroje. Na druhou stranu, pokud stránka využije React, dojde k porovnání skutečného a virtuálního DOM, aby vyhodnotil, co změnila uživatelova akce. Následně provede selektivní změnu pouze těch částí, kterých se to týká.

Tento selektivní update zabere mnohem méně výpočetního výkonu a sníží se potřebný čas pro načtení stránky. V případě přidání komentáře se jedná jen o nepatrné zlepšení. Signifikantní zlepšení přichází u komplexních webových aplikací.

Komponenty

Komponenty nám umožňují rozdělit uživatelské rozhraní na nezávislé, opakovaně použitelné části kódu a přemýšlet o každé části izolovaně. [14]

Svým konceptem komponenty připomínají JS funkce. Přijímají libovolné vstupy (zvané „props“) a vrací elementy knihovny React. Tyto elementy popisují, co by se mělo na obrazovce objevit. [16]

Třídní komponenty

V dřívějších verzích knihovny React se třídní komponenty používaly především jako komponenty, které uchovávají data (uživatelská, UI nebo informace ze serveru).

```
class Greetings extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Ukázka kódu 4 - Třídní komponenta v knihovně React. Zdroj: [Autor].

Funkční komponenty

Hned na první pohled je možné si z ukázky kódu (Ukázka kódu 5) povšimnout zjednodušeného a přehlednějšího zápisu kódu než u třídní komponenty (Ukázka kódu 4). Je to i jeden z důvodů, proč se v dnešní době doporučuje používat výhradně funkcionální komponenty.

```
function Greetings(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Ukázka kódu 5 – Funkcionální komponenta v knihovně React. Zdroj: [Autor].

Od verze 16.8. lze ve funkcionálních komponentách uchovávat stav (state) pomocí React Hooks – jedná se zejména o rozhraní, která umožňují použití a změnu lokálního stavu v komponentách bez nutnosti použití tříd, a pro vyvolávání „side effects“ uvnitř komponenty ještě před jejím samotným vykreslením. [14]

Tato ukázková funkce je validní React komponenta, protože přijímá objekt „props“ jako argument s daty a vrací React element. Takto vypadající komponenty se nazývají funkcionální komponenty, protože to jsou čisté JS funkce.

3.2.2 Vue

Vue.js (dále jen Vue) je progresivní framework se zaměřením na tvorbu moderního uživatelského rozhraní v jednostránkových webových aplikacích. Autorem tohoto frameworku je Evan You, který první verzi tohoto frameworku oficiálně vydal v roce 2014 [17]. Vue není podporováno žádnou korporací, jako je tomu například u knihovny React. Framework je vhodně navržený, aby neobsahoval zbytečné nepoužité moduly pro daný projekt, což způsobilo, že je výsledná aplikace velikostně malá. Tato vlastnost dovoluje vývojářům postupně rozšiřovat projekt podle potřeb, proto je vhodné framework použít od malých aplikací až po ty komplexní. [18]

Syntaxe

I tento framework je založen na skládání komponent, kde reprezentace, funkcionalita a vizuální stránka jsou definovány v jednom souboru s příponou `.vue`. Tento soubor využívá syntaxi příbuznou jazyku HTML. Soubor můžeme rozdělit do tří částí: šablona (template), script (skript) a style (styly).

Template zastupuje reprezentativní část kódu, která je zodpovědná za zobrazení dat uživateli. Jedná se o validní HTML kód obohacený o komponenty, u kterého využíváme syntaxe dvojitých složených závorek pro následné zobrazení dat. Podobně jako v knihovně React je i zde možnost podmíněného vykreslování a vložení JS funkcí. [19]

```
<template>
<h1 class=“welcome“>Welcome, {{username}} </h1>
</template>
```

Ukázka kódu 6 – ukázka syntaxe ve frameworku Vue. Zdroj: [Autor]

V části script se nachází obchodní pravidla (bussiness logic) komponent. Právě v této části dochází k volání API serveru pro zaslání potřebných dat. Každá komponenta také nabízí metody k přístupu životního cyklu komponenty. Vývojář je tedy schopen reagovat na změny stavu komponenty.

V poslední části je definován vzhled elementů ze šablony, a to pomocí CSS.

Životní cyklus komponent

Stejně jako konkurenční knihovna React, tak i Vue má životní cyklus komponent zvaný „Lifecycle hooks“. Funkce popsané v následujících odstavcích slouží vývojářům k zachycení stavu komponent a dovolují jim na něj reagovat. Každá Vue komponenta prochází sérií několika fází:

`beforeCreate` – jedná se o úplně první funkci, která je zavolána okamžitě po inicializaci instance.

`created` – v tuto chvíli je již instance plně inicializovaná, začala pracovat s vlastnostmi dat a již je možné získat datový typ proměnných.

`beforeMount` – jedná se o chvíli těsně předtím, než je instance začleněna do DOM. Template a script jsou plně zkompilovány, ale stále není možné manipulovat s elementy v DOM.

`mounted` – v této fázi začne být komponenta plně funkční a je možné ji použít. Data jsou předána šabloně a původní DOM element je nahrazen elementem s vyplněnými daty. S těmito daty může být dále manipulováno.

`beforeUpdate` – funkce je zavolána po tom, co dojde ke změně stavu aplikace, ale předtím, než jsou změny provedeny v DOM. Vývojáři využívají tuto funkci k přístupu DOM před provedením změn.

`updated` – v tomto případě došlo ke změně stavu komponenty a následné aktualizaci DOM. V tuto chvíli je možné vykonávat operace, které jsou na DOM závislé. Ve většině

případů je doporučeno se vyhnout editaci stavu v této funkci. Není zaručeno vykreslení všech potomků této komponenty.

`beforeUnmount` – tato funkce je volána těsně před odstraněním z DOM. Komponenta je v tuto chvíli stále funkční.

`unmounted` – komponenta byla úspěšně odstraněna z DOM i se všemi svými potomky.

3.2.3 Angular

Angular je framework, který cílí na poskytování moderních nástrojů k snadnějšímu vývoji moderních SPA. Jedná se o open-source projekt spravovaný společností Google, která v pravidelných půlročních intervalech aktualizuje tento projekt [20]. Předchůdcem tohoto frameworku byl AngularJS. Hlavním rozdílem mezi těmito frameworky je jazyk, ve kterém byly napsány. Angular je psaný v nadstavbě jazyka JS známé pod názvem TypeScript (dále jen TS).

Oproti konkurenci je vývojář donucen řídit se základy objektově orientovaného programování (dále jen OOP). Konkurence dovoluje vývojářům možnost volby mezi verzemi JS a TS. Angular se rozhodl pro podporu nadstavby TS.

Syntaxe

Stejně jako již zmíněné technologie React a Vue je i Angular založen na komponentovém přístupu. Komponenta je základním nápadem stojícím za vznikem frameworku. Komponenta exportuje třídu (class – objekt jazyka TS) a je definována direktivou `@Component` [21]. Použitím této direktivy je možné předat komponentě specifické informace:

- CSS selektor – definuje použití komponenty v šabloně. HTML tag, který se shoduje s názvem selektoru se stane instancí dané komponenty.
- HTML šablona – určuje způsob vykreslení dat v komponentě.
- CSS styly – definují vzhled komponenty v HTML šabloně

Tento framework především využívá komponenty (components) a šablonu (template).

```
import { Component } from '@angular/core';
@Component({
      selector: 'hello-world',
      //Definování HTML šablony pomocí přímého zápisu
      template: `
            <h2>Hello World</h2>
            <p>This is my first component!</p>
            `,
})
export class HelloWorldComponent {
// Kód vykonaný uvnitř této třídy řídí chování komponenty.
}
```

Ukázka kódu 7 – Základní komponenta ve frameworku Angular. Zdroj: [21].

Šablona

Šablona (template) je součástí každé komponenty a určuje, jakým způsobem budou uživateli vykreslena data. Šablona je definována pomocí cesty k souboru nebo pomocí přímého zápisu do komponenty (in-line). Stejně jako u předchozích frameworků je i zde možnost podmíněného vykreslení a dynamických změn dat. Na základě změny stavu komponenty následně dojde k přepsání DOM.

Služby

Pro samotnou logiku aplikace využívá Angular tzv. služby (services), které slouží pro komunikaci a distribuci mezi klientem a serverem. Jednotlivé komponenty využívají tyto služby pomocí principu Dependenci Injection (dále jen DI) [21]. DI je návrhový vzor sloužící k předání závislostí uvnitř aplikace. Závislost můžeme definovat jako potřebu objektů komunikovat [22].

3.2.4 Next.js

Next.js je open-source framework pro tvorbu jednostránkových aplikací vybudovaný na knihovnách Webpack, Babel a React. Framework byl vytvořen společností Vercel (dříve Zeit) v roce 2016 [23]. Stejná společnost poskytuje server-less hosting optimalizovaný právě pro tento framework. Server-less – bezserverová architektura, kde poskytovatel služeb automaticky zřizuje, škáluje a spravuje infrastrukturu, která je vyžadována ke spuštění kódu. Kombinace knihoven, kterou framework využívá, umožňuje vývojářům založit projekt takřka s nulovou potřebou konfigurace. *Styled-JavaScript* je systém, který

umožňuje psaní CSS v JS souboru. To s sebou přináší optimalizaci rychlosti načítání aplikace, kdy se ze serveru stahují jen styly, které jsou používané na dané stránce. [23]

Způsoby vykreslení

Vykreslení (rendering) znamená, že framework dokáže generovat HTML v předstihu místo toho, aby vše zpracovával JS na klientově straně. Předkreslením (pre-rendering) je důležité se zabývat nejenom kvůli výkonu, ale také kvůli jeho dopadu na SEO. Předkreslení je možné díky procesu „hydration“. Do vygenerované HTML stránky je přiřazeno naprosté minimum potřebného JS kódu. Jakmile prohlížeč načte HTML stránku, spustí JS, který zaručí, že stránka bude plně interaktivní. [24]

Next.js poskytuje dva způsoby předkreslení obsahu. Framework dovoluje vývojářům každou stránku generovat jiným způsobem. To vede ke vzniku hybridní aplikace, kde většina stránek je staticky generovaná a tam, kde je to opravdu potřeba, je využito vykreslení na straně serveru. [24]

Statické generování

Statické generování je doporučováno tvůrci frameworku. Vygenerované HTML soubory mohou být snadno cachovány (uloženy do mezipaměti) pomocí CDN serveru. Při statickém generování dochází k vytvoření HTML stránky pro každou stránku ve složce `Pages`, a to při spuštění příkazu `next build`. Tento soubor je poté použit při každém dotazu na tuto konkrétní stránku. Níže vidíme stránku, která je staticky generována. [24]

```
function About() {
  return <div>About</div>;
}

export default About;
```

Ukázka kódu 8 – Ukázka stránky, která je staticky generovaná. Zdroj: [Autor].

Vykreslení na straně serveru

Vykreslení na straně serveru z anglického spojení „server-side rendering“ (dále jen SSR). Pomocí této metody vykreslení je veškerá logika dané stránky zpracována na serveru nikoli na klientovi. Díky tomu je možné se vyhnout zasílání velkých JS souborů na klienta. Přímým následkem je zkrácení času potřebného k dosažení interaktivity stránky neboli Time to Interactive (TTI) [25]. Měření TTI je jeden z důležitých parametrů při

posuzování výkonu stránky. Některé stránky dávají přednost zobrazení stránky uživateli dříve, než je možné se stránkou interagovat. V uživateli může toto chování vyvolat pocit, že stránka je plně funkční a po pokusu o interakci může být zmaten a stránku opustit s dojmem její nefunkčnosti. [26]

Next.js řeší SSR pomocí exportování asynchronní metody `getServerSideProps()`, která může být použita pouze uvnitř stránky (Ukázka kódu 9). Funkce `getServerSideProps()` se volá při každém požadavku. Data získaná v této funkci jsou následně předána stránce pomocí „props“. Tato funkce je využita, pokud se na stránce nachází data, která se často mění. Při požadavku dojde k vykreslení na serveru a klientovi je předána vykreslená HTML stránka. To přináší výhodu pro SEO, jelikož vyhledávače jsou lépe uzpůsobeny na zpracování webů v HTML. Na druhou stranu nemůže být stránka uložena pomocí CDN. [24]

```
function Page({ data }) {
    // Vykreslení dat...
  }

  // Tato funkce je volána při každém požadavku
  export async function getServerSideProps() {
    // Žádost o data z externích API
    const res = await fetch(`https://.../data`)
    const data = await res.json()

    // Předání dat stránce skrze props
    return { props: { data } }
  }

  export default Page
```

Ukázka kódu 9 – renderování na straně serveru s požadavkem o data ze serveru. Zdroj: [24]

3.3 Technologie na straně serveru

Následující kapitola se bude zabývat představením technologií, které se v dnešní době využívají na serverové části aplikací.

3.3.1 Node.js

Node.js (dále jen Node) je zástupcem JS runtime (běhového) prostředí, které dokáže vytvářet aplikace na straně serveru [27]. Jedná se o open-source technologii dostupnou na různých operačních systémech (Windows, Linux, Unix, Mac OS X, atd.).

Veškerý kód je spouštěn asynchronně, což dovoluje aplikaci provádět operace nezávisle na jejich pořadí. Díky tomuto principu dokáže Node zpracovávat velké množství

požadavků souběžně. Právě asynchronním zpracováním nejvíce vyniká oproti svým konkurentům. Uživatelé, kteří využívají systém postavený na Node, se nemusejí obávat blokace systému (dead-lock). Skoro žádná funkce nepracuje se vstupně výstupními zařízeními napřímo a díky tomu nemůže dojít k blokaci. [28]

Node neoperuje v prostředí webového prohlížeče. Z toho důvodu by bylo zbytečné zachovávat přístup k rozhraním, které se na serveru nedají využít (geolokace, přístup k úložišti webového prohlížeče). Naopak implementuje klasické rozhraní pro webové servery jako jsou souborové systémy a HTTP. Díky tomu je umožněna práce se soubory a reakce na HTTP požadavky uživatelů. [27]

3.3.2 PHP

PHP je skriptovací jazyk na straně serveru. Jedná se o široce využívaný jazyk, což potvrzují i statistické údaje [29]. PHP je určen především k tvorbě dynamických webových stránek, ale je možné ho použít i k tvorbě desktopových a konzolových aplikací.

Jazyk PHP je nezávislý na platformě až na pár omezení kvůli funkcím závislých na operačním systému. U syntaxe je patrná inspirace od dalších programovacích jazyků jako je Pascal nebo Java. Nejčastější využití PHP je možné vidět u webových stránek, kde nejčastější kombinací je operační systém Linux, databázový systém MySQL a webový server Apache.

3.4 Výběr technologií

Tato kapitola se bude zabývat výběrem technologií pro implementaci aplikace pro správu fotografií. Analýza bude vyhodnocena podle následujících vlastností – popularita, zastoupení na trhu práce, komunitní podpora a požadavky aplikace.

3.4.1 Server

Tato kapitola se bude věnovat analýze technologií vhodných pro použití na straně serveru.

Popularita

Celkově můžeme popularitu vybraných technologií hodnotit podle počtu webů, které danou technologii používají. Z tabulky níže můžeme vidět, že PHP je využíván na drtivé většině webových stránek.

PHP	JavaScript
79,2 %	1,3 %

Tabulka 1 – procentuální zastoupení webů psaných v JS nebo PHP. Zdroj: [30].

Trh práce

Z údajů dostupných z aplikace Google Trends je jasně patrné, že mezi JS a PHP je pouze malý rozdíl v poptávce po zaměstnání. Na základě analýzy dat od dubna 2020 do dubna 2021 se ujal vedoucí pozice JS [31].

PHP	JavaScript
43,7 %	56,3 %

Tabulka 2 – procentuální zastoupení poptávaných pozic. Zdroj: [31].

Komunitní podpora

Z následující tabulky je na první pohled vidět, že obě technologie mají v komunitě vývojářů velkou podporu. Velkou převahu má ovšem JS.

PHP	Node.js / JavaScript
1 399 731	2 584 212

Tabulka 3 – Počet kladených dotazů na stránce Stackoverflow. Zdroj: [32].

Požadavky aplikace

Hlavním rozdílem mezi PHP a JS je způsob zpracování operací. PHP vykonává kód synchronně. To znamená, že nemůže být vykonána další operace, dokud není plně dokončena předchozí. V tomto aspektu splňuje požadavky JS, který bude pracovat prokazatelně lépe při větším zatížení.

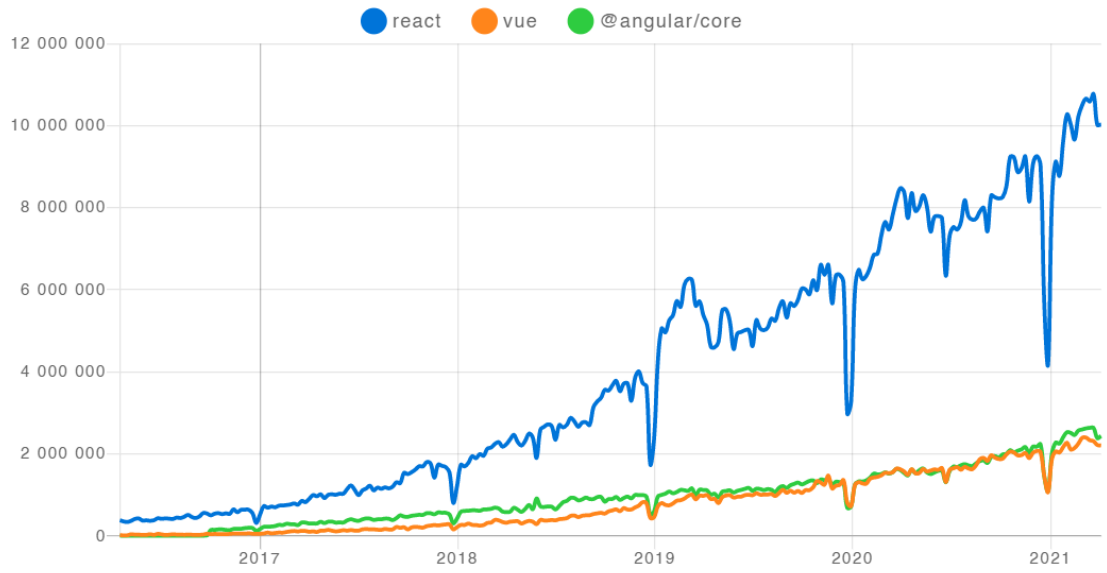
Zvolením JS oproti PHP je možné dosáhnout jednotnosti při vývoji. Serverová i klientská část bude psaná ve stejném jazyce. To zapříčiní nižší čas na osvojení daných technologií.

3.4.2 Klient

Tato kapitola se bude věnovat analýze technologií vhodných pro použití na straně klienta.

Popularita

Na následujícím grafu je možné vidět popularitu vyjádřenou počtem instalací balíčků pomocí NPM. Z grafu je jasně patrné, že React se těší velké oblibě v programátorské komunitě z čehož je možné usuzovat jeho optimisticky vyhlížející budoucnost.



Obrázek 2 – Graf počtu stažení NPM balíčků technologií React, Vue a Angular. Zdroj: [33].

Trh práce

Podle výsledků průzkumu trhu z roku 2018 se nejvíce dařilo frameworku Angular, v těsném závěsu se nachází knihovna React. Z údajů dostupných z aplikace Google Trends je jasně patrné, že o React je mezi vývojáři velký zájem. Na základě analýzy dat od dubna 2020 do dubna 2021 se ujal vedoucí pozice React.

	React	Angular	Vue.js
Nabízená místa	105 123	107 396	15 842
Poptávka po místě	61,7 %	31,9 %	6,4 %

Tabulka 4 – Přehled aktuálního stavu na trhu práce. Zdroj: [31, 34].

V tomto aspektu analýzy by bylo nejlepší zvolit knihovnu React, o kterou je značný zájem. Tímto by se razantně snížily náklady na hledání vývojářů, kteří by vyvíjeli danou aplikaci.

Komunitní podpora

Komunitní podpora je jedním z důležitých faktorů při výběru jakékoli technologie. Z následující tabulky je možné vidět, že odborná populace se velké oblíbenosti těší React a Vue, kdežto Angular tolik uznávaný není. Z údajů je patrné, že nejsilnější komunitní podporu má knihovna React, která, co se týče počtu použití, nechává konkurenci daleko za sebou.

	React	Angular	Vue.js
Počet hvězdiček	167 000	72 500	182 000
Počet uživatelů	6 200 000	1 800 000	142 000
Forks	36 600	19 000	28 700
Počet kontributorů	1 542	1389	399

Tabulka 5 – Přehled dat získaných z Github. Zdroj: [35].

Požadavky aplikace

K vývoji ve frameworku Angular oproti knihovně React je potřeba komplexnějšího a delšího kódu. Framework obsahuje již od začátku všechny funkce, které jsou pro JS frameworky standardní. Funkce, které nepotřebujeme, jsou v tomto projektu stále přítomny. React je zaměřen na tvorbu uživatelského rozhrání. Dovoluje vývojářům dodat do projektu přesně to, co potřebují, díky obrovské dostupnosti balíčků třetích stran. Výsledná aplikace pak obsahuje pouze balíčky, které jsou nezbytné pro její správné fungování.

Vue.js je inspirováno frameworkem Angular.js a za svůj cíl si klade převzetí těch nejlepších vlastností z tohoto frameworku. Přes odlehčenost, skvělý výkon a přehlednou syntaxi zde neexistují řešení osvědčená v praxi (best practice). Oproti knihovně React to může vést k horší údržbě složitějších aplikací.

Na základě analýzy a faktu, že autor je obeznámen s problematikou vývoje aplikací pomocí knihovny React, bude v aplikaci využit framework Next.js postavený právě na této knihovně.

4 Zpracování obrazových dat

Tato kapitola se bude zabývat historií zpracování obrazu, důvody, které vedly lidstvo k zachytávání a získávání informací z obrazu. Dělení podle způsobu vytvoření a následné prezentaci na digitálním zařízení a segmentaci obrazu.

4.1 Historie zpracování obrazu

Mnoho způsobů digitálního zpracování obrazu bylo objeveno v 60. letech 19. století institucemi JPL (Jet Propulsion Laboratory), MIT (Massachusetts Institute of Technology), Bell Laboratories a pár dalšími, které se zaměřovaly na využití v oblastech satelitního snímání, konverze standardů pro fotografie, lékařství, videohovorů, rozpoznání písma a mnohých dalších. [36]

Cena zpracování obrazu byla vzhledem k dostupnému výpočetnímu výkonu vysoká. To se ovšem změnilo v 70. letech 19. století, kdy se začala technologie zpracování obrazu šířit, především díky snížení cen. Digitální obraz bylo v tuto dobu možné zpracovávat v reálném čase třeba pro televizní vysílání. S rostoucím výkonem se začaly běžné počítače využívat k více specializovaným a sofistikovanějším operacím. Na přelomu století spolu s příchodem digitálního signálového procesoru se digitální zpracování obrazu stalo jednou z nejvíce používaných forem zpracování obrazu obecně, a to především díky cenové dostupnosti a širokému využití. [37]

Důvody zpracování obrazu

Z hlediska lidského vnímání informací či reality obecně je obrazové podání pro lidstvo to nejpřirozenější a nejsnáze pochopitelné. Analýza, klasifikace obrazu a počítačové vidění se pokouší docílit zachycení reality lidským pohledem.

Zájem o tuto technologii pramení ve dvou principech – zlepšení obrazových informací pro interpretaci člověka a zpracování obrazových dat pro autonomní strojové vnímání. [36]

Jako příklad mohou posloužit satelity jakožto nástroj pro snímání informací jak o vesmíru, tak i na Zemi. Problém nastává, pokud je potřeba data získaná satelitem zobrazit přímo člověku. Satelity ukládají snímky ve formě RGB číselných kombinací. K pochopení těchto dat musíme tuto kombinaci převést na kombinaci barev, které je člověk schopen vnímat, a také zvolit správný formát, jakmile jsme se zpracováním hotovi.

Satelity posílají tyto snímky v podobě digitálního signálu, který je zpracován počítačem. [37]

4.2 Digitální zpracování obrazu

Jedná se o vědeckotechnickou disciplínu zabývající se zpracováním obrazových dat z různých zdrojů, např. kamery, fotoaparátu, ultrazvuku, satelitu atd. [37]

Samotné zpracování a rozpoznání obrazu je rozděleno do několika samostatných kroků, které nemusejí být vždy využity všechny nebo ve stejném pořadí. Jednoznačná konvence, která by nám říkala, v jakém pořadí by měly být tyto kroky provedeny, není stanovena. Kroky pro zpracování obrazu jsou následující:

1. snímání a digitalizace obrazu,
2. předzpracování,
3. segmentace obrazu,
4. popis objektů,
5. klasifikace.

4.3 Snímání a digitalizace obrazu

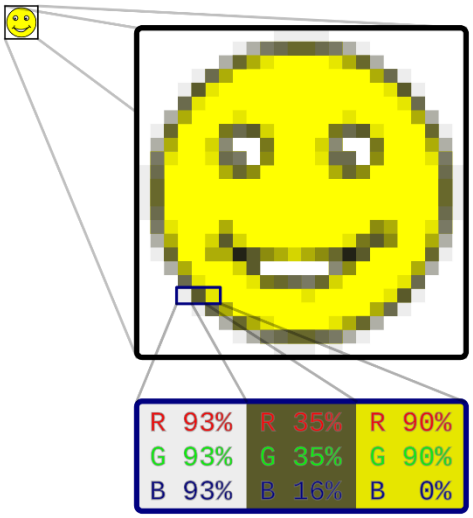
V dnešní době je možné zachytit realitu širokou paletou zařízení, jako je například skener dokumentů či fotoaparát. Veškerá data zachycená těmito zařízeními musejí být převedena na odpovídající formát, který bude snadné zobrazit člověku. U dat, která jsou snímána pomocí světlocitlivých snímačů, to není takový problém jako u zařízení z lékařského odvětví. Příkladem je EKG – tato data nemají odraz ve skutečném světě. Digitální signál je tedy převeden do podoby matematických funkcí, které jsou následně zobrazeny uživateli v podobě grafu.

Digitální obraz je číselná interpretace (nejčastěji v binární soustavě) dvourozměrného obrazu. Digitální obrázek můžeme dělit na 2 typy, a to podle rozlišení a způsobu zachycení, popř. vytvoření. Jedním typem je vektorový obraz a druhým rastrový.

Rastrová grafika

Rastrový obraz má konečný rozsah digitálních hodnot nazývajících se pixely. V rastrové grafice je celý obrázek uložen do mřížky (rastru). Každý bod má přesně definovanou polohu vůči rastru a barvu v jednom z barevných modelů, nejčastěji RGB. Tento formát obrázku bývá uložen v počítači jako rastrová mapa (dvourozměrné pole typu integer).

Tyto hodnoty jsou následně převedeny a uloženy v kompresní formě. Rastrový obrázek může být zachycen pomocí digitálních kamer, skenerů a dalších zařízení.



Obrázek 3 – Ukázka rastrového obrázku a zápisu v RGB. Zdroj: [38].

Výhody

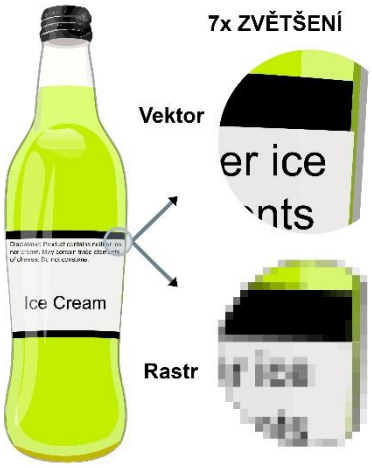
- Snadnost pořízení – vzhledem k dnešním možnostem se dá obrázek pořídit velice snadno, a to pomocí fotoaparátu, mobilního telefonu, výstřižku obrazovky či skeneru.
- Fotografie – vhodná volba reprezentace pro digitální fotografii.

Nevýhody

- Změna velikosti – změna měřítek obrázku, tj. jeho zmenšování či zvětšování vede ke zhoršení kvality. Velkou opatrnost je nutné klást především u zvětšování snímku, kdy se může velikost dostat až za hranici, kde bude možno vidět rastr pouhým okem.
- Velikost souboru – velké paměťové nároky pojící se s rozlišením a hloubkou barev. Při použití bezztrátové komprese a formátu JPEG s rozlišením snímku 1920 na 1080 pixelů může velikost souboru dosahovat až 3 megabajty. U vyššího rozlišení 3840 na 2160 pixelů je potřeba uchovat informaci o 8,294,400 bodech. Zde se velikost souboru pohybuje okolo hodnoty 12 megabajtů.

4.3.1 Vektorová grafika

Vektorová grafika funguje na principu složení obrazu pomocí geometrických primitiv. Primitivy jsou myšleny následující prvky: přímky, křivky, kružnice, trojúhelníky, ale především vektory, které mají vše, co je potřeba (velikost, délku a směr).



Obrázek 4 – Porovnání rastrového a vektorového obrazu. Zdroj: [38].

Výhody

- Velikost – změna velikosti bez následků ztráty kvality,
- Jednoduchost – menší náročnost na paměť,
- možnost práce s dílčími částmi obrázku.

Nevýhody:

- Dostupnost – horší možnosti pořízení snímku ve vektorovém formátu, oproti rastrové grafice.
- Výpočetní náročnost – zvyšující počet jednotlivých komponent může překonat paměťovou náročnost rastrové grafiky.
- Viditelný rozdíl mezi rastrovou a vektorovou grafikou je možné vidět na snímku níže. Tento snímek porovnává rastrovou a vektorovou grafiku při sedminásobném přiblížení.

4.4 Předzpracování

Po úspěšném nasnímání obrázku a jeho digitalizaci je objekt v digitální podobě, ale během skenování nebo pořizování snímku mohlo obecně dojít k zhoršení kvality. Např. u pořizování snímku fotoaparátem mohla být špatně zvolena jedna ze složek parametrů expozice (ISO, clona a čas). Obrázek tedy může být podexponovaný (nízký jas snímku).

V tuto chvíli jsou na řadě transformace, které dokážou tyto problémy vyřešit. Ovšem záleží na závažnosti deformace obrazu. Pokud je obraz příliš poškozený, transformace nebude mít žádný pozitivní přínos.

Transformace:

- Afinní transformace – natočení, zkosení, měřítko, posun
- Transformace barev – změna odstínu barev, jas, kontrast

4.5 Segmentace obrazu

Jedná se o proces, jehož výsledkem je rozdělení komplexního obrazu do menších zjednodušených částí (segmentů). Tyto části mají stejnou sadu vlastností. Díky této reprezentaci obrazových dat jsme schopni snadněji provést analýzu celkového obrazu, která se používá pro nalezení objektů a hranic v obraze (čáry, křivky, kružnice atd.). [38]

Pro lepší detekci kontur snímku lze použít algoritmus pro detekci hran, který je založen na matematických modelech, kde se mění jasová složka snímku a zkoumá se ostrost bodu při přechodu na jinou jasovou úroveň. Jedním z algoritmů je „Canny edge detector“.

5 Přehled platforem

Tato kapitola se zabývá představením nejpoužívanějších platforem, na kterých jsou aplikace vyvíjeny. U každé platformy jsou představeny její výhody a nevýhody.

5.1 Webové aplikace

Využívání webových aplikací zažívá v posledních desetiletích markantní nárůst. Největší výhodou webové aplikace je její provoz bez potřeby instalace na koncovém zařízení a dostupnost z každého zařízení podporující internetový prohlížeč. Díky vysoké adaptabilitě této technologie se razantně sníží náklady na vývoj aplikace, protože není nutno aplikaci dále reprodukovat pro ostatní platformy. Aplikace je nasazena na straně serveru, ke kterému uživatel přistupuje přes již zmíněný internetový prohlížeč. Díky tomu uživatel téměř vždy přistupuje k aktuální verzi a ke svým datům může přistoupit odkudkoli. Webová aplikace s sebou přináší i některá úskalí v podobě úniku citlivých informací, pokud není správně zabezpečena (kupříkladu nekvalitním naprogramováním nebo chybějícím SSL certifikátem). Aby aplikace mohla fungovat, musí být zařízení připojeno k internetu. Z toho vyplývá, že rychlost aplikace je úměrně závislá na rychlosti a kvalitě připojení. Webové aplikace můžeme rozdělit do několika dalších skupin, viz následující kapitoly. [39]

5.1.1 Jednostránkové aplikace

Single page aplikace (Single Page Appliccation, dále jen SPA) je webová aplikace z nových trendů vývoje, která běží čistě v internetovém prohlížeči na straně klienta. Díky tomu během svého užívání nevyžaduje opětovné vykreslení celé stránky. Tento typ aplikace je využíván velkými společnostmi v aplikacích jako je třeba Gmail, Google Mapy nebo Facebook. SPA volíme v případě, kdy chceme uživateli poskytnout maximální uživatelský prožitek (User Experience, dále jen UX). Ve chvíli, kdy uživatel čeká na odpověď ze serveru, můžeme uživateli zobrazit animaci načítání nebo kostru textu místo dlouhého načítání „bílé“ stránky, která může evokovat chybu aplikace. [40]

Na druhou stranu, užívání přináší jisté oběti. Jednou z nich může být právě používání JS, který klade větší nároky na vývojáře. Díky principu asynchronního načítání dat ze serveru není SPA vhodná k optimalizaci pro vyhledávače (Search Engine Optimalization, dále jen SEO) [41]. Tuto problematiku částečně řeší renderování na straně serveru (Server Side Rendering, dále jen SSR). Postup SSR probíhá následovně:

1. Bot nebo uživatel poprvé navštíví stránku a pošle požadavek na server.
2. Server zašle oproti běžnému webu JS soubor. Zde nastává problém se SEO. Vyhledávač Google sice dokáže indexovat SPA, ale je to velice zdlouhavý a složitý proces. [42] V tuto chvíli by bot opustil naši stránku.
3. Prohlížeč načte React ze souboru.
4. Aplikace je funkční a asynchronně stahuje požadovaná data.
5. Uživatel může plně využívat aplikaci.

Výhody

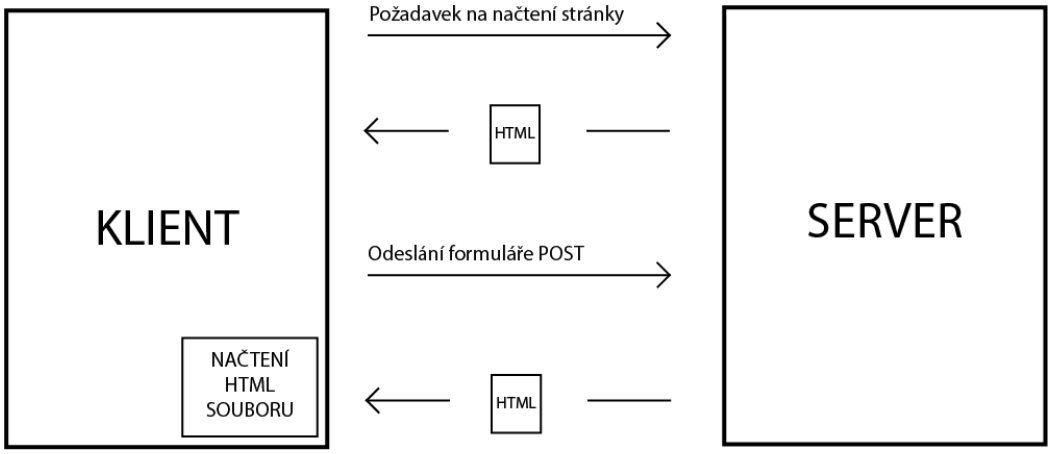
- Rychlost – za největší benefit SPA se dá považovat rychlost, jakou probíhá její načtení dat ze serveru. Aplikace je schopna oproti MPA zaslat požadavek o data, který je zpracován v rámci jednoho stejného HTML souboru. Tím odpadá potřeba stahovat celý HTML soubor pro každý odkaz na webu.

Nevýhody

- SEO – indexování SPA aplikací je v dnešní době možné, ale jedná se stále o problematickou část vývoje.

5.1.2 Vícestránkové aplikace

Vícestránkové aplikace (Multi-page application, dále jen MPA) využívají jeden z nejběžnějších a nejstarších způsobů renderování stránky. V praxi to znamená, že stránka se musí znovu načíst po každé změně, kterou uživatel provede (klikne na tlačítko, přidá komentář) nebo se pokusí zobrazit nová data. Nejčastější technologie použité pro stavbu dnešních MPA aplikací jsou HTML, CSS a JavaScript. MPA jsou stále ve velké míře využívány pro velké webové aplikace s širokou nabídkou služeb, produktů či obecně jakéhokoli katalogového systému. Zdárným příkladem může být e-shop společnosti Amazon. [43]



Obrázek 5 – Princip fungování MPA. Zdroj: [Autor].

První webové aplikace byly vyvíjeny právě jako MPA. Hlavním důvodem byla absence technologie asynchronních dotazů na server (AJAX). Princip fungování je možno vidět na obrázku (Obrázek 5 – Princip fungování MPA. Zdroj: [Autor]

1. Uživatel zašle svůj požadavek na webový server.
2. Server zašle uživateli HTML soubor spolu s dalšími soubory jako například CSS.
3. Po úspěšném načtení stránky může uživatel vykonat akci. V tomto případě vyplní formulář a odešle ho.
4. Server požadavek zpracuje a odpoví novým HTML souborem. Po přijetí klientem je starý soubor HTML nahrazen a stránka vykreslena. [43]

Výhody a nevýhody

I mezi dnešními technologiemi najde MPA svoje využití.

- Náklady – MPA je mnohem levnější na vývoj a údržbu než SPA. SPA je mnohem mladší než její předchůdce a z toho důvodu mají MPA technologie mnohem větší zastoupení na trhu práce. V praxi se nám sníží i náklady na provoz aplikace. Když si porovnáme náklady na hosting u e-shopu vybudovaného na PHP frameworku oproti e-shopu, kde využijeme například Java framework Spring společně s knihovnou React, vychází levněji první varianta (tedy PHP).
- SEO – je největší výhodou MPA, protože bylo optimalizováno právě pro stránky tohoto typu. Ačkoliv se SEO algoritmy výrazně během let změnily, pozice MPA zůstala stejná.

5.2 Mobilní aplikace

Mobilní aplikace běžně označována jako App je druh aplikačního softwaru, který je vytvořen pro operace na mobilních zařízeních, jimiž je chytrý telefon, tablet nebo chytré hodinky. [44] Poskytují uživateli podobné nebo stejné služby jako jejich desktopové alternativy. Oproti desktopové verzi mají několikanásobně menší velikost a pomocí služeb jako je App Store nebo Google Play je jejich instalace, aktualizace a správa mnohem intuitivnější a snadnější.

Trend poslední doby stále více naznačuje na potencionál mobilních aplikací, které vytlačují nejen desktopové aplikace, ale i ty webové. Webová aplikace doplněná právě tou mobilní je kombinace, kterou můžeme ve velké míře vídat právě díky výhodám obou technologií. Ukázkou nám může být internetové bankovnictví, kde uživatel může využít pro získání informací o svém kontě webovou aplikaci a mobilní aplikace poslouží jako autorizační prvek ke vstupu bez nutnosti zaslání SMS klíče.

Výhody

- Pohodlí – aktuální statistiky ukazují na nárůst popularity mobilních aplikací vůči webovým z důvodu pohodlnějšího ovládání. Oproti webovým aplikacím poskytují lepší UX, doba načtení obsahu je kratší a snáze se ovládají. Disponují push notifikacemi, možností sdílení, speciálními funkcemi. To všechno zvyšuje uživatelovu důvěru v aplikaci, která si ho snáze udrží. Není nutno řešit responsivní zobrazení na různé druhy zařízení jako tomu je u webových aplikací. Odpadá tedy nevzhledný rámeček prohlížeče a design působí čistěji.
- Personalizace – díky faktu, že uživatelé mají chytré zařízení většinu času u sebe, se velice daří aplikacím, které jsou navrženy k pravidelnému užití. Tyto aplikace dovolují uživatelům nastavit předvolby, profil a vždy mít své informaci po ruce. Díky kolekci těchto osobních údajů mají z obchodního hlediska obrovský potenciál. Dovolují totiž přesněji marketingově cílit na určitou skupinu uživatelů.
- Práce offline – stěžejní výhodou je zajisté možnost pracovat s aplikací offline. Díky tomu, že aplikace je fyzicky přítomna na daném zařízení a může data ukládat přímo do zařízení, tak není nutné, aby byla připojena k internetu.

Nevýhody

- Kompatibilita – mobilní aplikace musí splňovat požadavky konkrétního OS (operačního systému), na kterém aplikace běží. Z toho vyplývá, že je potřeba mít vytvořeno několik verzí aplikací pro různé OS (Windows, iOS). Tuto nevýhodu lze vyřešit pomocí vývoje multiplatformního řešení, které by mělo vyhovovat každému operačnímu systému. Realita je však mnohem komplikovanější a s tímto řešením jsou spojeny i další problémy.
- Podpora a údržba – tím, že aplikace může mít více verzí se výrazně zvyšují časové a finanční náklady na jejich údržbu. V praxi to znamená, že vývojáři musí poskytnout aktualizace – opravit chyby pro každý typ zařízení v pravidelných časových intervalech, informovat o těchto změnách své uživatele a přimět je stáhnout si novou verzi aplikace.

5.3 Desktopové aplikace

Desktopová aplikace je forma softwaru, kterou je nutné nainstalovat na harddisk zařízení, na kterém bude spuštěn. K jeho vývoji je nutná znalost vyššího programovacího jazyka (Java, C# a další). Aplikaci lze spustit pouze na platformě (operačním systému), pro kterou byla naprogramována. Je spuštěn v rámci systému, na kterém je závislý. Hlavním důvodem použití desktopové aplikace je nezávislost technologie na připojení k internetu. Vše ovšem záleží na účelu dané aplikace. Data této aplikace potom zůstávají na pevném disku, pokud si je uživatel nenahraje na cloudové úložiště, flashdisk a další.

Výhody

- Aplikace je většinou připravena k použití bez ohledu na to, zdali máme internetové připojení. Hodně také záleží na účelu aplikace.
- Aplikace je nainstalována na pevném disku zařízení, z toho důvodu je její spuštění velice rychlé. Pro uživatele je rovněž rychlejší využít ikonu na ploše, která spustí aplikaci.

Nevýhody

- Desktopové aplikace bývají častým cílem hackerů. Aby se předcházelo velké zranitelnosti aplikace, musejí vývojáři vydávat často bezpečnostní aktualizace, které musí uživatel většinou sám stáhnout a nainstalovat.

- Složitější mechanismus zálohy a ukládání dat. Uživatel musí manuálně vytvořit kopii svých dat, přičemž hrozí vznik nekonzistentnosti dat. Při pádu systému či aplikace uživatel přijde o všechna neuložená data.
- Týmová spolupráce není většinou možná z důvodu, jakým jsou desktopové aplikace navrženy. K synchronizaci je nutné využít aplikaci třetích stran nebo přejít na cloudové řešení (Microsoft Office 365).

5.4 Progresivní webové aplikace

Progresivní webová aplikace (dále jen PWA) je pokus o zkombinování kladných vlastností webových a nativních aplikací. Stavebními kameny jsou HTML, CSS a JavaScript, stejně jako je tomu u standardních webových aplikací. Díky tomu je vývoj aplikace rychlejší a snazší oproti vývoji nativních aplikací. PWA může na první pohled vypadat jako obyčejná aplikace, ale funkcionalita je to, co ji činí odlišnou. Na rozdíl od klasického webu je aplikace schopna využívat geolokaci, push notifikace, práci s daty online a mnohem více. [45]

PWA získává velice rychle na popularitě. Důkazem je fakt, že ji začali využívat internetoví giganti jako je Twitter, YouTube, StarBucks či Uber. Pokud uživatel navštíví PWA stránku, má možnost si tuto aplikaci nainstalovat na plochu počítače nebo domácí obrazovku chytrého zařízení. Po spuštění se aplikace jeví jako nativní, tzn. bez rámečku prohlížeče, a zpřístupní uživateli další funkce.

Výhody

- Není nutnost instalovat aplikaci na odlišných platformách z různých zdrojů (iOS App Store, Android Google Play).
- Oproti nativním aplikacím je jejich vývoj levnější, protože si vystačíme s webovými technologiemi a nemusíme spravovat několik verzí aplikací kvůli každé platformě (zařízení, operační systémy), kterou by měla podporovat.
- Díky webovým technologiím je snadné aplikaci vyvíjet responzivní, tzn. že se aplikace přizpůsobí většině obrazovkám. Výjimku tvoří atypická zařízení.
- PWA aplikace je mnohem snadnější propagovat potencionálním uživatelům díky internetovým vyhledávačům. Pro PWA platí stejná pravidla jako pro klasické webové stránky. Navíc aplikace může být stále dostupná skrze obchody s mobilními aplikacemi.

Nevýhody

- U komplexních a výpočetně náročnějších aplikacích lze pozorovat výraznější nárůst spotřeby baterie, pokud je aplikace používána na některém z chytrých zařízení.
- Ačkoliv dokážou PWA fungovat jako nativní aplikace, stále nedokážou přistupovat k některým dalším funkcionalitám zařízení, jako např. NFC, Bluetooth, pokročilé ovládání kamery a další.
- Omezená podpora prohlížečů – PWA nefungují na zařízeních se starší verzí prohlížečů, které nepodporují ServiceWorker. [46]

6 Návrh aplikace

Tato kapitola se zabývá problémem návrhu aplikace pro správu fotografií.

6.1 Existující řešení pro práci s fotografiemi

Existující řešení, která pracují s fotografiemi, je možné rozdělit do tří skupin (viz níže). Jednou ze skupin jsou aplikace, které dovolují spravovat a organizovat fotografie. Můžeme sem zařadit i cloudová úložiště. Využíváním těchto služeb získají uživatelé přístup k funkcím jako je snadnější sdílení souborů, zálohování dat a možnost kolaborace více uživatelů.

OneDrive (dříve SkyDrive)

Jedná se o cloudovou službu, která je spravována společností Microsoft. Poprvé byla spuštěna byla 1. srpna 2007 [47]. Přihlášeným uživatelům je umožněno ukládat data a osobní soubory. Uživatel může tyto soubory sdílet a synchronizovat se svými dalšími zařízeními. Služba nabízí svým uživatelům zdarma prostor o velikosti 5 GB. Výhodou služby je úzké propojení s dalšími produkty firmy Microsoft jako je Office 365.

Google Disk

Google disk byl poprvé spuštěn 24. dubna 2012 a je, jak již název napovídá, spravován společností Google. Oproti konkurenční aplikaci OneDrive, Google Disk svým uživatelům nabízí 15 GB prostoru zdarma a také poskytuje lepší zabezpečení dat. Ovšem navýšení velikosti úložiště je spojeno s vyšší cenovou nabídkou, než jak je tomu u konkurence.

Druhou skupinou jsou aplikace určené k publikaci fotografií, přesněji se jedná o sociální sítě, které dovolují sdružování lidí se společnými zájmy prostřednictvím online prostředí.

Facebook

Facebook je spravován stejnojmennou společností. Spuštěn byl 4. února 2004 [48]. Jedná se o rozsáhlý webový systém sloužící k sdílení mediálních dat (fotografií, videí a souborů) a komunikace mezi uživateli. K práci s fotografiemi slouží aplikace Fotky. Při nahrání fotografie mohou uživatelé vykonávat základní úpravy jako je ořezání fotografie a změna orientace.

Instagram

Zakladateli aplikace jsou Kevin Systrom a Mike Krieger. Aplikace se poprvé objevila v obchodě App Store společnosti Apple 6. října 2010. V roce 2011 byla aplikace odkoupena společností Facebook. [49]

Jedná se o sociální síť zaměřenou čistě na sdílení fotografií a videí. Oproti Facebooku zde není možné přidávat textové příspěvky. Rozšiřuje, ale možnosti úpravy fotografií, jako je editace kontrastu, jasu, struktury, teploty sytosti atd. Rovněž dovoluje použití přednastavených úprav, takzvaných "filtrů". Uživatelé mohou hodnotit fotky ostatních pomocí ikony srdíčka nebo se k fotografii vyjádřit pomocí komentáře.

Třetí skupinu tvoří aplikace pro amatérské a profesionální editace fotografií.

Photoshop

První verze aplikace Photoshop vyšla v únoru roku 1990 a je do dnešního dne spravována společností Adobe. Původně se jednalo o projekt bratrů Thomase a Johna Knolla. [50]

Photoshop je software určený pro editaci obrázků a retušování fotografií. Aplikace dovoluje uživatelům tvořit, vylepšovat nebo kreslit obrázky, umělecká díla a ilustrace. Jedná se o nejvíce rozšířený software tohoto druhu. Dovoluje upravovat obrázky různých formátů a následně je exportovat.

GIMP

GIMP je open-source projekt od studentů Kalifornské univerzity v Berkeley. Veřejnosti byl zpřístupněn 15. února 1996. [51]

Zásadními rozdíly mezi aplikacemi GIMP a Photoshop jsou:

- GIMP je poskytován zdarma, kdežto Photoshop je placený,
- obě řešení nabízejí velké množství dodatečného obsahu,
- GIMP obsahuje méně nástrojů než Photoshop,
- GIMP nemá aplikaci pro mobilní telefony,
- GIMP je používán především pro amatérské účely, kdežto Photoshop spíše pro profesionály.

6.2 Funkční a nefunkční požadavky

Funkční a nefunkční požadavky jsou takové, které se musejí identifikovat na samotném začátku vývoje aplikace a jsou očekávaným výstupem funkcionality již hotové aplikace.

Funkční požadavky objasňují a definují veškerou funkcionalitu dané aplikace, tzn. co je nutné udělat, a identifikuje veškeré úkony a akce, které se od aplikace vyžadují. Ve zkratce – co by aplikace měla umět a co nikoliv.

Nefunkční požadavky jsou takové požadavky, které nemají přímý dopad na funkcionalitu aplikace (design, rychlost atd.)

Funkční požadavky

Dle zadání této bakalářské práce je žádoucí, aby aplikace dokázala pracovat s metadaty, která jsou k fotografii vložena ve formátu EXIF. Jako hlavní cíl si aplikace klade usnadněnou správu fotografií. Toho je docíleno pomocí galerií, do kterých uživatel může roztřídit fotografie, a chytrého vyhledávání pomocí tagů, které jsou přiřazovány během nahrání fotografie.

Během procesu nahrávání fotografie se uživateli zobrazí samotná fotografie a lokace na mapě, kde byla fotografie pořízena. Toho je docíleno pomocí GPS souřadnic uložených v metadatech. Dojde také k zobrazení jednotlivých údajů o fotografii – informace o zařízení, na kterém byla fotografie pořízena, parametry expozice, čas, datum a mnohé další. Zobrazení těchto informací je nejvíce příhodné profesionálním i amatérským fotografům, kteří tyto informace potřebují znát. Uživatel má možnost si zvolit, zda chce fotografii označit tagem ručně nebo přenechá tuto operaci na samotné aplikaci.

Shrnutí funkčních požadavků v bodech:

- registrace a přihlášení pomocí emailu nebo skrze Google účet,
- nahrání fotografie,
- aplikace poskytne možnost automaticky označit fotografii tagem nebo dovolí uživateli provést tuto akci ručně,
- vytvoření galerií, které seskupují více fotografií dohromady.

Nefunkční požadavky

Aplikace bude obsahovat jednu jazykovou mutaci, a to anglický jazyk. Hlavním důvodem je velikost základny potencionálních uživatelů. Dle odhadů se počet anglicky hovořících

lidí pohybuje mezi 400 miliony a 1.3 miliardou [52]. To je minimálně 40x větší základna uživatelů, než je počet obyvatelů České republiky.

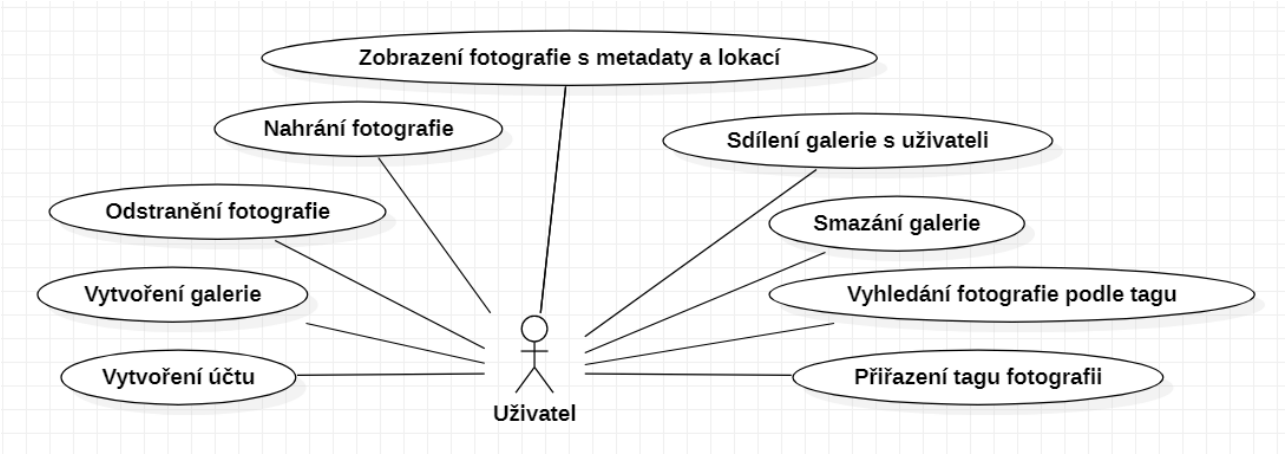
Uživatelské rozhraní bude jednoduché a intuitivní na ovládání. Cílem je zamezení odchodu nových uživatelů, kteří by měli problém s adaptací na ovládání aplikace.

Shrnutí nefunkčních požadavků v bodech:

- aplikace by měla mít jednoduchý a intuitivní design, se kterým se bude uživateli snadno pracovat,
- aplikace má jednu jazykovou mutaci (anglický jazyk).

6.3 Use-Case diagram

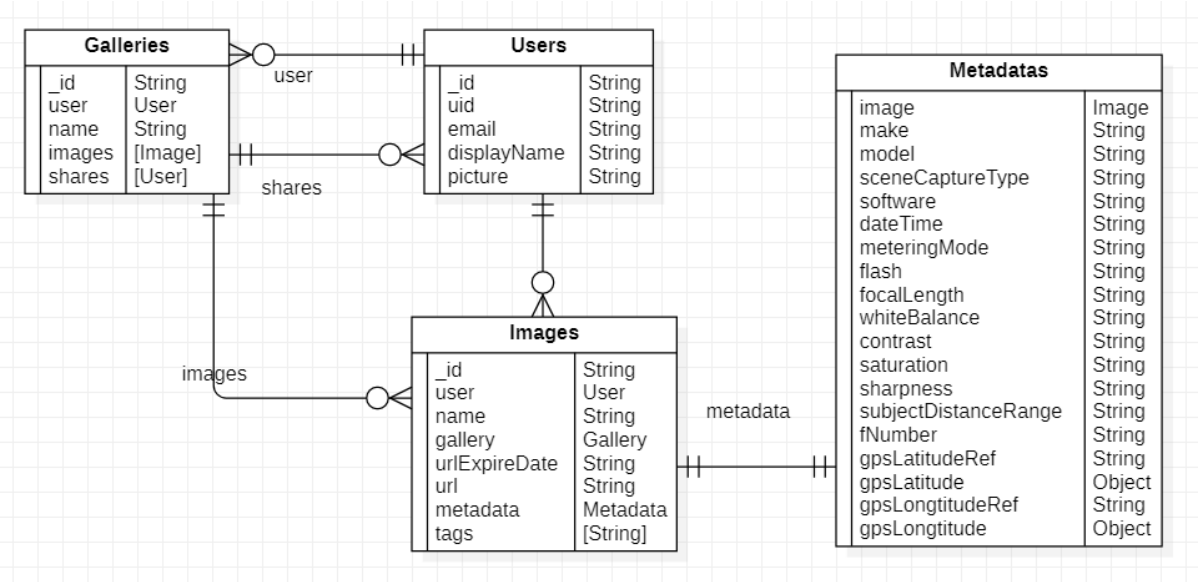
Pomocí Use-Case diagramu (Obrázek 6) je znázorněna funkcionalita pro přihlášeného uživatele. Nepřihlášený uživatel může zobrazit pouze stránku, kde má možnost se přihlásit nebo si vytvořit účet.



Obrázek 6 – Diagram případů užití. Zdroj: [Autor].

6.4 Návrh databázové struktury

Databázová struktura mapuje vztahy mezi entitami. Entita je objekt, jehož data je možné zachytit a uložit ve formě popisu vlastností, pracovního postupu nebo tabulky. Tato struktura byla navržena pro NoSQL databázi. Konkrétně se jedná MongoDB, které poskytuje možnost cloudového hostingu zdarma.



Obrázek 7 – Návrh databázové struktury. Zdroj: [Autor].

7 Implementace aplikace

Tato kapitola se bude zabývat problematikou implementace webové aplikace pro správu fotografií. Větší pozornost bude věnována službě Firebase, Google Mapy a Google Cloud Vision API. Implementace vychází z pečlivě zhotoveného návrhu aplikace.

Hlavními prostředky pro tvorbu aplikace se na základě průzkumu staly technologie Next.js a Node.js, přesněji framework Express.

7.1 Architektura aplikace

Hlavní koncept architektury spočívá v realizaci serverové a klientské části nezávisle na sobě. Komunikace je realizovaná pomocí REST API, kde dochází k výměně dat ve formátu JSON. Tento přístup dovoluje škálovat jednotlivé části podle potřeby.

Server

Serverová část je zhotovena ve frameworku Express.js, který je nadstavbou samotného Node.js. Server je logicky rozdělen do několika částí – Kontroler (controller), Router, Služby (services). Na první pohled se může zdát rozdělení zbytečné kvůli množství kódu, které je nutné napsat navíc. Konečný kód je ale lépe udržovatelný a vyměnitelný díky separaci logiky. Souborová struktura vypadá následovně:

```
├─── models
│   ├─── user.model.js
├─── routers
│   ├─── user.route.js
├─── services
│   ├─── user.service.js
├─── controllers
│   ├─── user.controller.js
```

Models – slouží k definování datových modelů knihovny „mongoose“.

Routers – uchovává jednotlivé routery, které definují API aplikace. Požadavek následně předají kontroleru.

Controllers – v této složce jsou uloženy všechny kontrolery, které zodpovídají za zpracování požadavků (validace parametrů požadavků, zaslaní odpovědi zpět klientovi, …).

Services – obsahuje funkce (služby) zodpovědné za zpracování veškeré bussiness logiky a vykonání databázových dotazů. Kontroleru, který službu zavolal, se vrátí požadovaný objekt nebo bude obeznámen o selhání funkce pomocí chybové zprávy.

Klient

Klientská část je vyvinuta ve frameworku Next.js jehož základ tvoří knihovna React. Rozdělení klientské části je následovné:

├── components
├── conf
├── hooks
├── out
├── pages
├── public
├── utils

Components – slouží k uložení komponent (znovupoužitelné části kódu).

Conf – v této složce jsou uloženy globální údaje ke knihovnám (Firebase, Axios).

Hooks – zde jsou uloženy vlastní „hooks“. Například `useAuth`, který slouží jako rozhraní pro autentizaci uživatele.

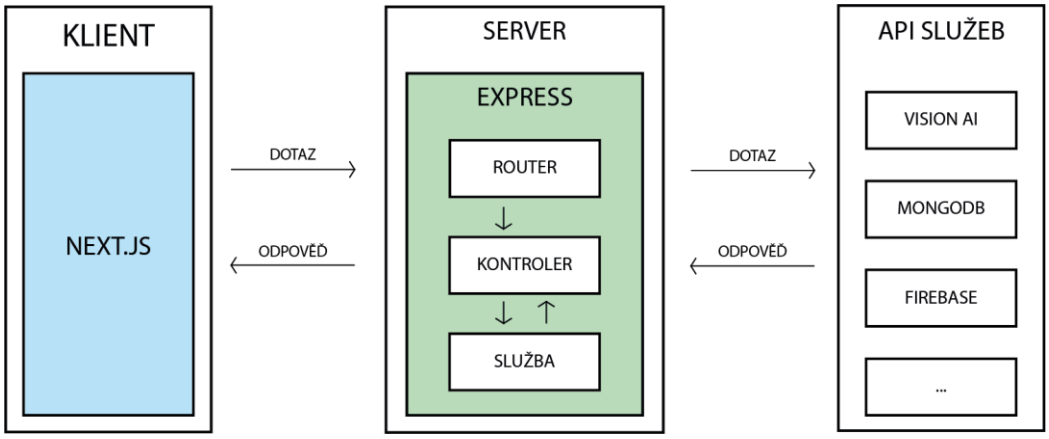
Out – složka obsahující vygenerované soubory, které jsou připraveny pro umístění na hosting. Jedná se o statické generování.

Pages – obsahuje JS soubory, které definují URL adresy aplikace.

Utils – pomocné funkce.

Fungování aplikace je vysvětleno na následujícím snímku (Obrázek 6). Jednotlivé kroky jsou vyjádřeny pomocí seznamu.

1. Klient zašle svůj dotaz na server, kde správu nad dotazem převezme framework Express.
2. Express se pokusí naleznout router podle URL, na kterou byl dotaz zaslán. V případě, že cesta nevede k žádnému routeru, dojde k zaslání chybové hlášky na klienta.
3. Router určí, který kontroler má nad danou URL adresou správu, a předá mu dotaz.
4. Kontroler zpracuje dotaz na jehož výsledku provede volání jednotlivých služeb.
5. Služba vykoná potřebné operace s daty nebo požadavek zašle ke zpracování na API navazujících služeb. Výsledek operace je sdělen kontroloru.
6. Kontroler vrátí výsledek klientovi.
7. Klient zobrazí výsledek uživateli.



Obrázek 8 – Architektura aplikace pro správu fotografií. Zdroj: [Autor].

7.2 Routování v aplikaci

Výhodou frameworku Next.js je mechanismus, který se stará o obsluhu routování. Nutno podotknout, že knihovna React (na které je tento framework postaven) postrádá tento mechanismus. Vývojáři jsou tedy nuceni používat knihovny třetích stran. React je používán především pro tvorbu SPA aplikací, které byly popsány v přechozích kapitolách. Jedním z problémů, proč web tvoří pouze jedna stránka, je právě routování. Stránka mění svůj obsah na základě uživatelových vstupů, ale nemění hodnotu URL adresy. Klasický uživatel webových stránek je zvyklý používat tlačítka zpět a vpřed, která

v tomto případě neplní svojí funkci. Toto nestandardní chování aplikace by mohlo vést ke ztrátě uživatele. Framework Next.js naštěstí tento problém odstranil.

Next.js řeší routování pomocí „pages“ neboli stránek. Jedná se o složku v souborové struktuře, která obsahuje soubory s příponou `.js`. Každý soubor má stejně pojmenovanou funkci, která vrací komponenty nebo elementy JSX k renderování.

Aby uživatel mohl navštívit URL adresu `https://photom.com/profile`, musí ve složce `pages` existovat stejnojmenný soubor `profile.js` obsahující podobnou funkci jako v ukázce kódu (Ukázka kódu 10).

```
import React from "react";
import UploadPost from "../components/post/UploadPost/UploadPost";

export default function profile() {
  return <UploadPost />;
}
```

Ukázka kódu 10 – Náhled do souboru ve složce „Pages“. Zdroj: [Autor].

Dynamické routování

V aplikaci bylo nutné vyřešit zobrazení jednotlivých galerií, kde každá galerie má svoji přístupovou adresu URL. Z předchozí ukázky jasně vyplývá, že pro každý náhled galerie by ve složce pages musel být vytvořen soubor. Tento postup by však byl velice neefektivní. Naštěstí Next.js nabízí možnost tzv. dynamického routování, které dovoluje definovat měnící se parametry v URL adrese. Stránka si pak tento parametr dokáže extrahovat z URL a použít k volání na server.

```
├── pages
│   ├── gallery
│   │   ├── index.js
│   │   ├── [id].js
```

Tato struktura souborů by dokázala zpracovat následující URL adresy.

- `https://photom.com/gallery` – nasměrování do souboru index.js ve složce gallery
- `https://photom.com/gallery/1` - nasměrování do souboru [id].js, ve kterém můžeme přistoupit k parametru 1.

7.3 Firebase

Aplikace využívá služeb společnosti Google, především platformu Firebase, která poskytuje širokou paletu nástrojů pro tvorbu aplikací. Firebase je cloudové řešení backend-as-a-service (BaaS), což je možné volně přeložit do spojení server-jako-služba. V praxi to znamená vývoj aplikace bez potřeby serverové části aplikace, protože služby Firebase dokážou nahradit nejvíce používané funkcionality. Služby využité v aplikaci pro správu fotografií jsou: – hosting, funkce, autentikace, úložiště souborů.

Aplikace využívá dvou npm balíčků, které dovolují přistupovat k službám na platformě Firebase. U klientské části se jedná o `firebase` a na serverové části se jedná o `firebase-admin`, který oproti prvnímu balíčku dovoluje přistupovat k funkcím s administrátorským oprávněním.

Hosting

K hostingu pro serverovou i klientskou část je využita již zmíněná platforma Firebase. Pro klientskou část stačí použít službu Hosting, která dovoluje umístit staticky vygenerované soubory. K získání těchto souborů je nutno upravit příkaz `build` v souboru `package.json`. Poté stačí spustit příkaz `npm run build`.

```
"scripts": {
    "dev": "next dev",
    "build": "next build && next export",
    "start": "next start"
  }
```

Ukázka kódu 11 – Upravený soubor package.json pro statické generování. Zdroj: [Autor].

Po dokončení procesu se v souborové struktuře objeví nová složka `out`, která obsahuje soubory k nahrání na hosting.

Pro serverovou část je využita služba Cloudové Funkce (Cloud Functions). Jedná se o serverless framework, který dovoluje automaticky spouštět serverový kód na základě spouštěče (HTTP požadavek). JS kód je uložen na cloudovém úložišti. Není tedy třeba škálovat svůj vlastní server vše probíhá automaticky dle aktuálních potřeb aplikace.

7.4 Zabezpečení aplikace

O zabezpečení aplikace se stará technologie JSON Web Token (dále jen JWT). Jedná se o šifrovaný způsob komunikace mezi klientem a serverem. Aplikace využívá služeb Authentication platformy Firebase. Tato služba dovoluje vývojářům spravovat

uživatelské účty a provádět autorizaci požadavků na server. Registrace je možná skrze email a heslo nebo pomocí jednoho z ověřených providerů (Google, Facebook a další).

Aplikace pro správu fotografií využívá tuto službu k autentizaci uživatelů a k autorizaci jejich požadavků na server. Způsob přihlášení je pomocí emailu a hesla nebo pomocí existujícího Google účtu.

7.5 Využití Google map

Pokud je u fotografie přiřazena lokace (ve formátu EXIF), ve které byla pořízena, tak aplikace detekuje tuto informaci a zobrazí pomocí Google Maps API lokalitu na mapě. V knihovně React je práce s mapami ulehčena, jelikož existuje knihovna `google-maps-react`, která poskytuje již hotové komponenty, se kterými je možné dále pracovat. Jednou z těchto komponent je GoogleMapReact.

```
...
  //Výpočet GPS souřadnic vzhledem k zěmepisné šířce a výšce
  const GPSCoordinates = formatGpsFromExif(
    JSON.stringify(props.exif, function replacer(key, value) {
      return value;
    })
  );

  return (
    //Vykreslení mapy do kontejneru společně s ukazatelem, kde byla foto. poří
zena
    <div style={{ height: "500px", width: "500px" }}>
      <GoogleMapReact
        bootstrapURLKeys={{ key: "<GOOGLE_MAPS_API_KEY>" }}
        defaultCenter={{
          lat: GPSCoordinates[0],
          lng: GPSCoordinates[1],
        }}
        defaultZoom={11}
      >
        <Marker lat={GPSCoordinates[0]} lng={GPSCoordinates[1]}></Marker>
      </GoogleMapReact>
    </div>
  );
}
```

Ukázka kódu 12 – GoogleMap komponenta využívající google-maps-react. Zdroj: [Autor].

7.6 Rozpoznání objektů v obraze – Vision AI

Aplikace využívá k popsání objektů v obraze funkci prostředí Google Cloud. Jedná se o Cloud Vision API, které nabízí předtrénované modely pomocí strojového učení. Dokáže detekovat objekty a jejich pozici, obličeje nebo převést psaný text do digitální formy.

V aplikaci je konkrétně využívána funkce popsání objektu. Při nahrání fotografie si uživatel může zvolit automatický popis fotografie, který obstará výše zmíněné API. Server nahraje uživatelovu fotografii na Firebase úložiště a získá URL přes kterou může Cloud Vision API získat přístup k fotografii. Odpověď Cloud Vision API je možné vidět v následující ukázce kódu ve formátu JSON.

Aby aplikace mohla komunikovat s Cloud Vision API je potřeba na straně serveru nainstalovat npm balíček `google-cloud`. Skrze tuto knihovnu je možné přistoupit k funkcím na platformě Google Cloud. Viz funkce v následující ukázce kódu.

```
// Importování Google Cloud knihovny
const vision = require("@google-cloud/vision");

/**
 * Funkce pro získání popisu objektů pomocí veřejné URL fotografie
 * @param {String} publicURL
 * @returns [String]
 */
exports.getLabelsFromCloudImage = async (publicURL) => {
  // Vytvoření nového klienta
  const client = new vision.ImageAnnotatorClient();
  // konstanta slouží jako hranice pro přesnost rozeznaných objektů (85%)
  const treshold = 0.85;

  const [result] = await client.labelDetection(publicURL);
  const filteredLabels = result.labelAnnotations.filter(
    (label) => label.score > treshold
  );
  // slouží k přeformátování slov do tagu pes -> #Pes
  return formatLabelsToTags(filteredLabels);
};
```

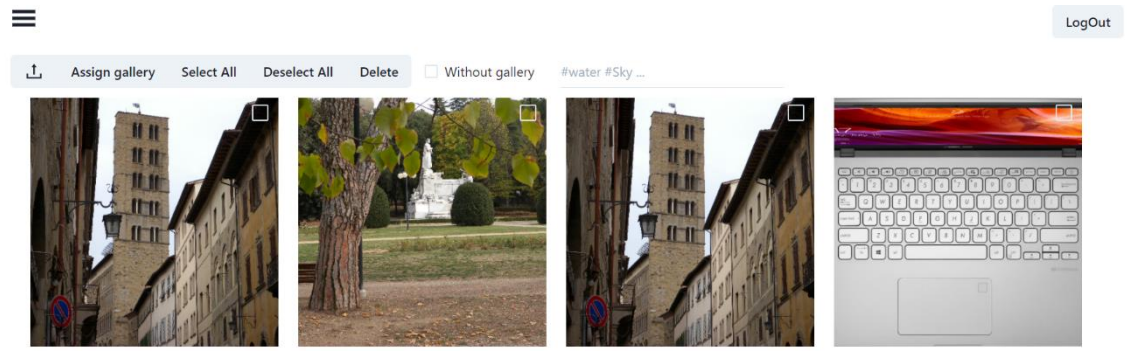
Ukázka kódu 13 – Funkce pro získání tagů z Cloud Vision API. Zdroj: [Autor].

```
{
  "faceAnnotations": [],
  "landmarkAnnotations": [],
  "logoAnnotations": [],
  "labelAnnotations": [
    {
      "locations": [],
      "properties": [],
      "mid": "/m/05s2s",
      "locale": "",
      "description": "Plant",
      "score": 0.9373241662979126,
      "confidence": 0,
      "topicality": 0.9373241662979126,
      "boundingPoly": null
    },
    {
      "locations": [],
      "properties": [],
      "mid": "/m/0h8nqmc",
      "locale": "",
      "description": "Outdoor bench",
      "score": 0.8946077823638916,
      "confidence": 0,
      "topicality": 0.8946077823638916,
      "boundingPoly": null
    }
  ]
}
```

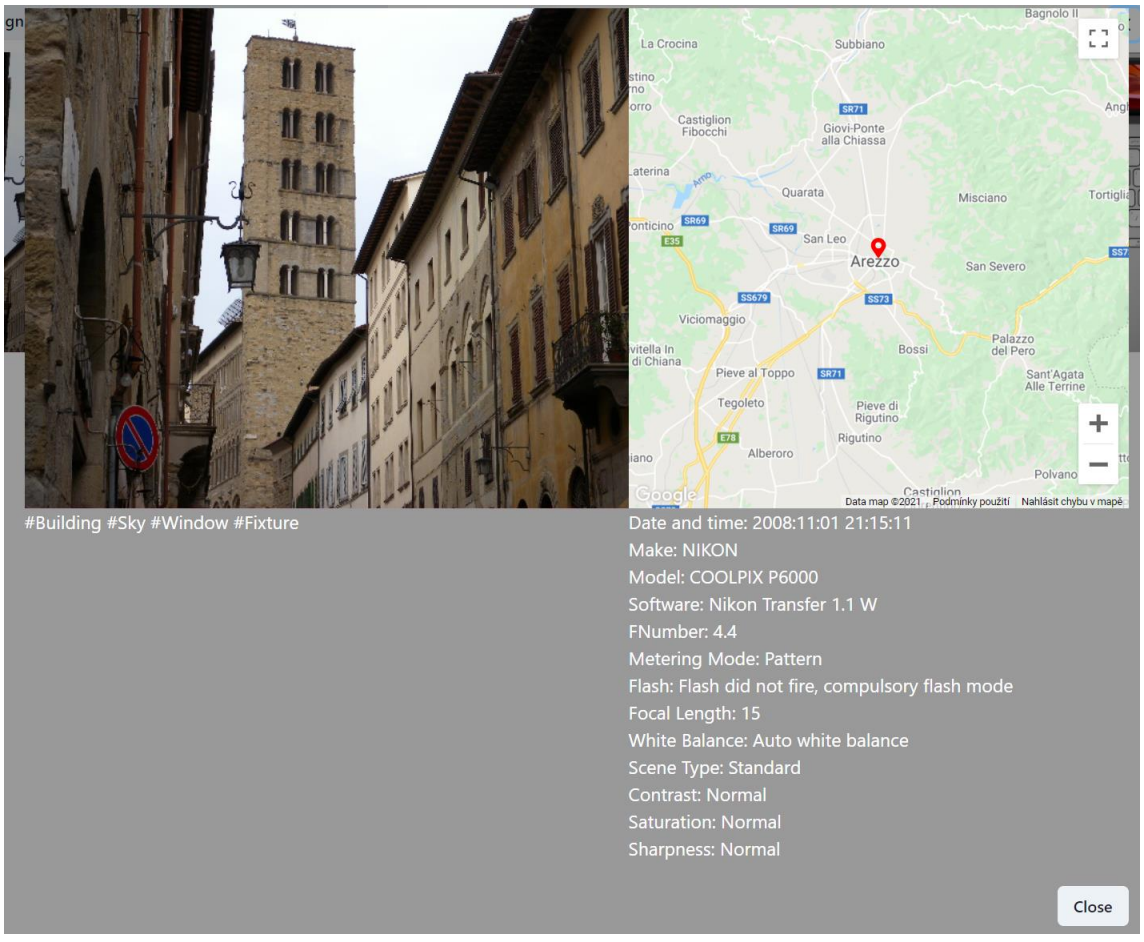

Ukázka kódu 14 – ukázka odpovědi ze služby Cloud Vision API. Zdroj: [Autor].

7.7 Vzhled aplikace

Na následujících snímcích lze vidět grafický vzhled finální aplikace.



Obrázek 10 – Náhled na galerii aplikace. Zdroj: [Autor].



Obrázek 9 – Panel s detaily o snímku. Zdroj: [Autor].

8 Závěr

Cílem této bakalářské práce bylo vytvoření aplikace pro správu fotografií. Pro implementaci aplikace byly zvoleny technologie Next.js na klientské straně aplikace a Express na straně serveru.

V práci byly představeny nástroje, které byly použity pro podporu vývoje. Jedním z těchto nástrojů je REST, který slouží jako architektonický typ, který poskytuje standardy pro model klient-server na kterém je aplikace pro správu fotografií postavena.

Dále byly představeny technologie na straně serveru i klienta. Pro serverovou část byly zvoleny technologie Node.js a PHP a pro klientskou část React, Vue, Angular a Next.js. Následně byla provedena analýza těchto technologií. Na základě této analýzy byly vybrány technologie Next.js a Node.js.

V další části práce je představena stručná historie a důvody, které vedli lidstvo ke zpracování obrazu. Je zde popsán rozdíl mezi vektorovou a rastrovou grafikou.

Platformou, na které byla aplikace vyvíjena, byla zvolena platforma webová. V práci jsou představeny výhody a nevýhody nejznámějších platforem pro které jsou aplikace vyvíjeny.

Zbytek práce byl věnován návrhu a implementaci, kde byly popsány funkční a nefunkční požadavky aplikace, databázová struktura, případy užití a návrh vzhledu aplikace. Poté následovala samotná implementace aplikace, kde byla popsána její architektura a komunikace mezi klientskou a serverovou částí. V této části byla řešena problematika zabezpečení aplikace, routování v aplikaci, rozpoznání objektů v obraze a využití metadat k zobrazení lokace, kde byla fotografie pořízena.

Aplikace uvažuje budoucí rozšíření. Mezi tato rozšíření patří optimalizace uživatelského rozhraní pro mobilní zařízené, potažmo implementace PWA technologie pro snadnější ovládání z mobilních zařízení.

9 Seznam použité literatury

[1] TURNER, Ash. *How Many People Have Smartphones Worldwide (April 2021)* [online]. 10. červenec 2018 [vid. 2021-04-04]. Dostupné z: https://www.bankmycell.com/blog/how-many-phones-are-in-the-world

[2] WYSOCKI, Robert K. Effective Software Project Management. 2010, **2010**, 671.

[3] *Architektura Klient-Server - Vojtěch Hordějčuk* [online]. [vid. 2021-04-05]. Dostupné z: http://voho.eu/wiki/klient-server/

[4] *What is Client-Server? Definition and FAQs | OmniSci* [online]. [vid. 2021-04-05]. Dostupné z: https://www.omnisci.com/technical-glossary/client-server

[5] *About npm | npm Docs* [online]. [vid. 2021-04-16]. Dostupné z: https://docs.npmjs.com/about-npm

[6] *Git - gitignore Documentation* [online]. [vid. 2021-04-16]. Dostupné z: https://git-scm.com/docs/gitignore

[7] WEBBER, Jim, Savas PARASTATIDIS a Ian ROBINSON. *REST in Practice: Hypermedia and Systems Architecture*. 1st edition. Beijing: O'Reilly Media, 2010. ISBN 978-0-596-80582-1.

[8] BIEHL, Matthias. *RESTful web API design: APIs your consumers will love*. 2016. ISBN 978-1-5147-3516-9.

[9] *What is Babel? · Babel* [online]. [vid. 2021-04-29]. Dostupné z: https://babeljs.io/

[10] COPES, Flavio. *A short and simple guide to Babel* [online]. [vid. 2021-04-18]. Dostupné z: https://flaviocopes.com/babel/

[11] Concepts. *webpack* [online]. [vid. 2021-04-16]. Dostupné z: https://webpack.js.org/concepts/

[12] *Beginning JSON | BEN SMITH | Apress* [online]. nedatováno [vid. 2021-04-16]. Dostupné z: https://www.apress.com/gp/book/9781484202036

[13] MARRS, Tom. *JSON at Work: Practical Data Integration for the Web*. 1st edition. Sebastopol, CA: O'Reilly Media, 2017. ISBN 978-1-4493-5832-7.

[14] *React – A JavaScript library for building user interfaces* [online]. [vid. 2021-02-04]. Dostupné z: https://reactjs.org/

[15] React: The Virtual DOM. *Codecademy* [online]. [vid. 2021-04-29]. Dostupné z: https://www.codecademy.com/articles/react-virtual-dom

[16] *Components and Props – React* [online]. [vid. 2021-04-29]. Dostupné z: https://reactjs.org/docs/components-and-props.html

[17] *Meet the Team — Vue.js* [online]. [vid. 2021-03-26]. Dostupné z: https://vuejs.org/v2/guide/team.html

[18] Understanding the Critical Rendering Path. *bitsofcode* [online]. 17. leden 2017 [vid. 2021-03-26]. Dostupné z: https://bitsofco.de

[19] *Template Syntax — Vue.js* [online]. [vid. 2021-03-26]. Dostupné z: https://vuejs.org/v2/guide/syntax.html

[20] *Angular - Angular versioning and releases* [online]. [vid. 2021-04-14]. Dostupné z: https://angular.io/guide/releases

[21] *Angular - What is Angular?* [online]. [vid. 2021-04-14]. Dostupné z: https://angular.io/guide/what-is-angular

[22] A quick intro to Dependency Injection: what it is, and when to use it. *freeCodeCamp.org* [online]. 18. říjen 2018 [vid. 2021-04-30]. Dostupné z: https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/

[23] *Next.js by Vercel - The React Framework* [online]. [vid. 2021-04-04]. Dostupné z: https://nextjs.org

[24] *Basic Features: Pages | Next.js* [online]. [vid. 2021-04-04]. Dostupné z: https://nextjs.org/docs/basic-features/pages#pre-rendering

[25] Rendering on the Web. *Google Developers* [online]. [vid. 2021-04-12]. Dostupné z: https://developers.google.com/web/updates/2019/02/rendering-on-the-web?hl=cs

[26] Time to Interactive. *web.dev* [online]. [vid. 2021-04-12]. Dostupné z: https://web.dev/interactive/

[27] NODE.JS. About. *Node.js* [online]. [vid. 2021-04-17]. Dostupné z: https://nodejs.org/en/about/

[28] *Node.js - Introduction - Tutorialspoint* [online]. [vid. 2021-04-30]. Dostupné z: https://www.tutorialspoint.com/nodejs/nodejs_introduction.htm

[29] *Usage Statistics and Market Share of PHP for Websites, April 2021* [online]. [vid. 2021-04-17]. Dostupné z: https://w3techs.com/technologies/details/pl-php

[30] *PHP vs. JavaScript usage statistics, April 2021* [online]. [vid. 2021-04-30]. Dostupné z: https://w3techs.com/technologies/comparison/pl-js,pl-php

[31] Google Trends. *Google Trends* [online]. [vid. 2021-04-18]. Dostupné z: https://trends.google.com/trends/explore?cat=31&q=Vue%20jobs,React%20jobs,Angular%20jobs

[32] Stack Overflow - Where Developers Learn, Share, & Build Careers. *Stack Overflow* [online]. [vid. 2021-04-18]. Dostupné z: https://stackoverflow.com/

[33] *react vs vue vs @angular/core | npm trends* [online]. [vid. 2021-04-17]. Dostupné z: https://www.npmtrends.com/react-vs-vue-vs-@angular/core

[34] Tech Trends Showdown🏆: React vs Angular vs Vue. *Zero To Mastery* [online]. [vid. 2021-04-17]. Dostupné z: https://zerotomastery.io/blog/tech-trends-showdown-react-vs-angular-vs-vue

[35] GitHub: Where the world builds software. *GitHub* [online]. [vid. 2021-04-18]. Dostupné z: https://github.com/

[36] ROSENFELD, Azriel. *Picture Processing by Computer*. B.m.: Academic Press, 1969. ISBN 978-0-12-597350-2.

[37] GONZALEZ, Rafael C. a Richard Eugene WOODS. *Digital Image Processing*. B.m.: Prentice Hall, 2008. ISBN 978-0-13-168728-8.

[38] SHAPIRO, Linda G. a George C. STOCKMAN. *Computer Vision*. 1st edition. Upper Saddle River, NJ: Pearson, 2001. ISBN 978-0-13-030796-5.

[39] MANAGEMENTMANIA. Webová aplikace (Web Application). *ManagementMania.com* [online]. [vid. 2021-02-13]. Dostupné z: https://managementmania.com/cs/webova-aplikace-web-application

[40] *Single page aplikace* [online]. [vid. 2021-02-13]. Dostupné z: https://jecas.cz/spa

[41] SEO Starter Guide: The Basics | Google Search Central. *Google Developers* [online]. [vid. 2021-02-13]. Dostupné z: https://developers.google.com/search/docs/beginner/seo-starter-guide?hl=cs

[42] *SPA SEO: Mission Impossible?* [online]. [vid. 2021-02-13]. Dostupné z: https://www.magnolia-cms.com/blog/spa-seo-mission-impossible.html

[43] NEOTERIC. Single-page application vs. multiple-page application. *Medium* [online]. 30. březen 2018 [vid. 2021-02-13]. Dostupné z: https://medium.com/@NeotericEU/single-page-application-vs-multiple-page-application-2591588efe58

[44] What is a Mobile Application? - Definition from Techopedia. *Techopedia.com* [online]. [vid. 2021-02-14]. Dostupné z: http://www.techopedia.com/definition/2953/mobile-application-mobile-app

[45] *Progressive Web Apps | Software AG* [online]. [vid. 2021-04-30]. Dostupné z: https://techradar.softwareag.com/technology/progressive-web-apps/

[46] *Is service worker ready?* [online]. [vid. 2021-03-23]. Dostupné z: https://jakearchibald.github.io/isserviceworkerready/

[47] OneDrive. *Microsoft Wiki* [online]. [vid. 2021-04-30]. Dostupné z: https://microsoft.fandom.com/wiki/OneDrive

[48] *Facebook* [online]. [vid. 2021-04-30]. Dostupné z: https://www.firstversions.com/2015/04/facebook.html

[49] BLYSTONE, Dan. The Story of Instagram: The Rise of the # 1 Photo-Sharing Application. *Investopedia* [online]. [vid. 2021-04-30]. Dostupné

z: https://www.investopedia.com/articles/investing/102615/story-instagram-rise-1-photo0sharing-app.asp

[50] Adobe Photoshop | software. *Encyclopedia Britannica* [online]. [vid. 2021-04-30]. Dostupné z: https://www.britannica.com/technology/Adobe-Photoshop

[51] *GIMP - A Brief (and Ancient) History of GIMP* [online]. [vid. 2021-04-30]. Dostupné z: https://www.gimp.org/about/ancient_history.html

[52] *Learntalk | How Many People in the World Speak English? | Learntalk* [online]. [vid. 2021-04-05]. Dostupné z: https://learntalk.org/en/blog/how-many-people-in-the-world-speak-english

10 Seznam obrázků

11 Seznam tabulek

12 Seznam ukázek kódů

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání bakalářské práce

Autor:	**Daniel Strach**
Studium:	I1700141
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Chytrá aplikace pro správu fotek**
Název bakalářské práce AJ:	Smart Application for Photos Management

Cíl, metody, literatura, předpoklady:

Cílem této bakalářské práce je navrhnout a implementovat chytrou aplikaci pro správu fotografií, využívající informace z metadat a analýzy samotného snímku. Osnova: 1. Úvod, 2. Vývoj aplikací, 3. Zpracování obrazových dat, 4. Přehled platforem, 5. Návrh aplikace, 6. Implementace aplikace, 7. Závěr

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Michal Macinka
Oponent:	Ing. Bruno Ježek, Ph.D.
Datum zadání závěrečné práce:	14.1.2018