

# **Fotografický web a administrační systém pro amatérského fotografa**

---

**Jiří Vala - Student POJFM**

# **Obsah**

1. Vymezení pojmu
2. Úvod a zadání práce
3. Klíčové vlastnosti a požadavky projektu
  1. Serverless architektura
  2. Moderní techstack
    1. Frontend a UI/UX
      1. HTML & CSS a Javascript
      2. React.js
      3. SASS/SCSS
    2. Backend
      1. Node.js
      2. Shell scripty
      3. Firebase
  4. Struktura projektu s ohledem na DMP
  5. Průběh vývoje
  6. Webová prezentace
  7. Administrační systém
  8. VPS - Virtual Private Server
    1. Operační systém GNU/Linux
      1. Debian 11
      2. Nginx
      3. CI/CD
        1. CI/CD scrypty na VPS
    9. Závěr

# 1. Vymezení pojmu

1. **DMP** - Dlouhodobá maturitní práce.
2. **CMS** - Content Management System, v překladu Administrační systém, je systém, který umožňuje manipulaci s daty a obsahem webové stránky či webové (ale i desktopové) aplikace.
3. **Hosting** - Neboli webhosting, znamená pronájem prostoru na cizím webovém serveru za účelem spuštění webové stránky či aplikace.
4. **Continuous Deployment (CI/CD)** - CI - Continuous Intergration a CD - Continuous Delivery je proces, který umožňuje nepřetržité dodávání aktuální verze projektu k testování či produkčnímu spuštění. V praxi to znamená, že pokud tým vývojářů provede změnu a uloží ji do *remote repository* (místo pro ukládání zdrojových kódů v cloudu), tato změna se automaticky projeví (*deployne*) na produční server (server, na kterém je spuštěna aktuální verze projektu a je poskytována uživatelům z venčí).
5. **Fetching** - Proces během kterého se "stahuje" data z databáze. Tento anglický výraz doslova znamená "přinést".
6. **Backend (server)** - Logická část aplikace nebo webu. Požadavky, které uživatel aplikace provede (například *přejmenování záznamu v databázi*) a data (například *všechny záznamy*), které si vyžádá, tento server nejdříve zpracuje, případně vytáhne z databáze a pak je odešle uživateli aplikace, která je následně zobrazí, či případně upraví v databázi.
7. **Frontend** - Jedná se o grafické uživatelské rozhraní, v rámci webových aplikací a stránek většinou nejčastěji pomocí značkovacího jazyka HTML, stylovacího jazyka CSS a scriptovacího jazyka Javascript.
8. **Cloud Computing Provider** - Poskytovatel cloudových služeb a infrastruktury (serverů).
9. **(Public/Private) API** - API je soubor procedur, funkcí a protokolů zajišťující komunikaci mezi dvěma platformami, které si vzájemně vyměňují data (například dvě rozdílné aplikace). K public API má přístup kdokoliv, k private API jen autentifikovaní uživateli například pomocí unikátního ID tokenu.
10. **DDOS útok** - Distributed Denial Of Service - je útok jehož cílem je pomocí sítě mnoha virtuálních počítačů, které v jednu chvíli budou provádět požadavky na daný server, vyřadit tento server z provozu, jelikož nezvládne nápor takového počtu požadavků a zhroutí se.
11. **Framework** - Jedná se o podpůrnou softwarovou strukturu, která zjednoduší vývoj a organizaci projektu v daném jazyce. Může obsahovat další funkce, podpůrné programy nebo vzory a doporučené postupy ve vývoji.
12. **Knihovna** - Jedná se souhrn procedur a funkcí, často avšak také konstant a datových typů, které usnadňují vývojáři práci tím, že může použít již hotový kód.
13. **Open-source** - Jedná se o projekt s otevřeným zdrojovým kódem, což znamená, že na jeho vývoji se může podílet každý a každý si jej může stáhnout. Popularita open-source projektů v posledních letech velmi roste.
14. **CRUD Operace** - Create, read, update, delete (vytvořit, číst, změnit, smazat) jsou čtyři základní operace se záznamem v databázové tabulce.
15. **Shell** - Jedná se o vrstvu operačního systému, která provádí příkazy a spouští programy.
16. **Pagination** - Jedná se o proces rozdělování obsahu na jednotlivé stránky a podstránky. Nejčastěji ji vidíme ve formě čísel na konci webové stránky.
17. **Dropdown menu** - Jedná se o list záznamů, který se zobrazí až ve chvíli, kdy uživatel nějakým způsobem provede interakci s tímto menu, například kliknutím myši.
18. **Algoritmus** - Proces jednotlivých příkazů za účelem splnění zadaného úkolu.
19. **Push notifikace** - Notifikace, které se zobrazují uživateli na mobilním zařízení v notifikační listě nebo na zamýkací obrazovce.

## 2. Úvod a zadání práce

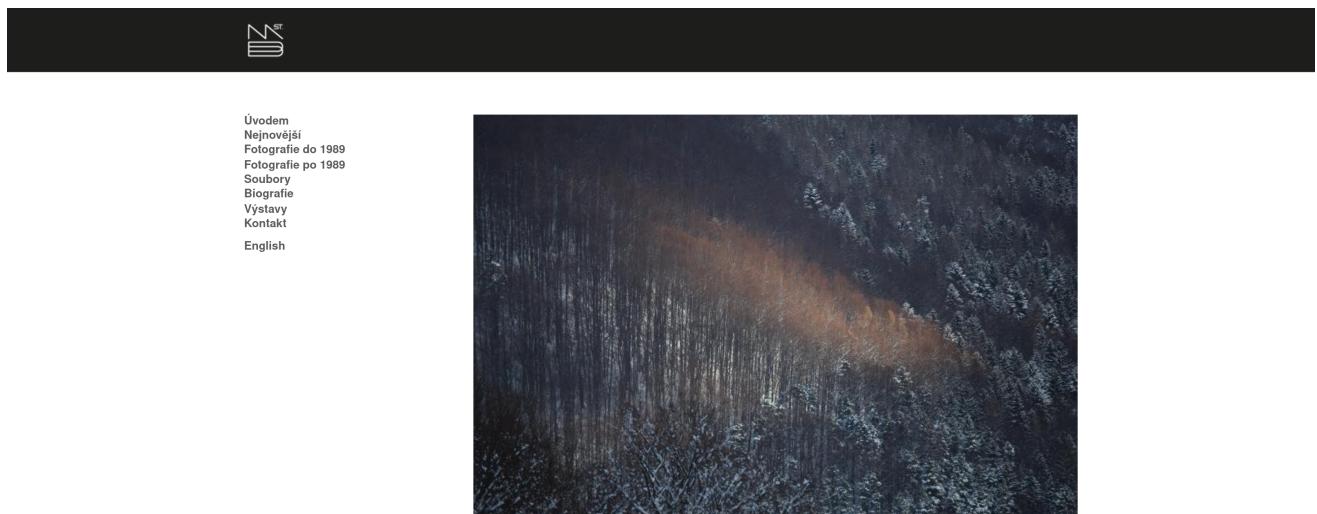
Pan Milan Bureš st. (dále jen *fotograf* nebo *základník*) je amatérský fotograf, který si již před pár lety nechal na zakázku vytvořit webovou stránku, na které by prezentoval své fotografie, které zachycují nádherné okamžiky jeho života. Nicméně postupem času se jeho web i administrační systém stali zastaralými a bylo potřeba celý systém zmodernizovat.

Proto mě v rámci dlouhodobé maturitní práce zprostředkované mým garantem kontaktoval a požádal, zda-li bych jeho webovou prezentaci nemohl modernizovat. Mým úkolem je tedy vytvořit web, který bude sloužit jako virtuální galerie jeho fotografií a bude zobrazitelná a funkční na všech zařízeních, a administrační systém (dále jen *CMS*), který bude umožňovat nahrávání, správu a manipulaci fotografií či alb, ve kterých se jednotlivé fotografie seskupují.

Dále bylo potřeba zajistit, aby má maturitní práce obsahovala část jiného předmětu, než jen programování (celý systém jsem naprogramoval) a databáze (data a fotografie, které fotograf nahraje/uloží). Rozhodl jsem se tedy, že si vytvořím vlastní server běžící na operačním systému Linux, na kterém budu hostovat vlastní instanci celého systému a bude obsahovat funkcionalitu tzv. *Continuous Deployment*, což přesně kopíruje funkcionality komerční instance projektu určené jen pro účely zákazníka.

## 3. Klíčové vlastnosti a požadavky projektu

Během vývoje webové prezentace byl velký důraz kladen na to, aby se co nejvíce podobala stávající verzi. Neveškeré elementy webu byly replikovatelné a také jsem se chtěl co nejvíce řídit pravidly dobrého webu. Pro porovnání, zde jsou screenshoty jednotlivých webových stránek.



2013 © Copyright Milan Bureš st.  
Webprograming by All Developers Studio  
Administrace

Obr.1 - Fotografova původní webová stránka



[Úvodem](#)  
[Nejnovější](#)  
[Soubory](#)  
[Výstavy](#)  
[Biografie](#)  
[Kontakt](#)



2021 © Mgr. Milan Bureš st.  
Made with ❤ by Orexin.

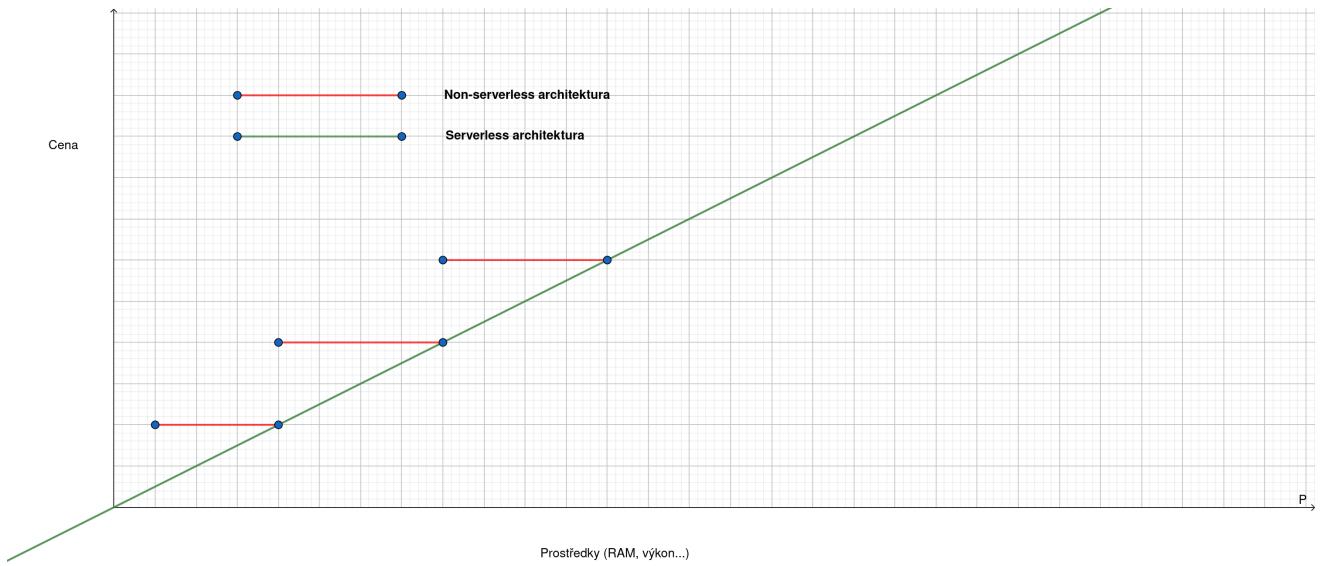
Obr.2 - Mé zpracování webové prezentace

### 3.1 Serverless architektura

Ačkoliv je samozřejmostí, že je jak webová prezentace tak CMS hostovaná na nějakém fyzickém serveru, k požadavkům jako fetchování dat z databáze nepoužívám žádný další (*backend*) server. To znamená, že jsem sice odkázaný kompletně na svého *cloud computing providera*, což je v mé případě Firebase od Google, ale na druhou stranu se nemusím (a ani zákazník) starat o údržbu vlastního serveru a ani jeho první konfiguraci, vše je tzv. *out-of-the-box* připravené k použití.

Díky serverless architektuře jsem nemusel vytvářet žádný *backend server* s *private API*, pomocí kterého bych dostával data z databáze a *cloud storage* do webové prezentace nebo administračního systému.

Serverless architektura má mnoho výhod, jedna z těch hlavních a největších je samozřejmě absence potřeby konfigurovat a později spravovat vlastní server, jelikož to kompletně zprostředkovává daný *cloud computing provider*, ale také celková cena "pronájmu", jelikož ta se počítá podle počtu požadavků na danou službu (například fetchování dat z databáze), bezpečnost, jelikož jsou tyto architektury zabezpečené například proti *DDOS* útokům, ale také celková škálovatelnost celého systému. *Cloud computing provider* automaticky škáluje prostředky (úložistě, výpočetní výkon apod.), které jsou k dispozici pro daný systém.



Obr.3 - Graf popisující škálovatelnost jednotlivých architektur

Na grafu jde vidět, že pokud si zvolíme cestu non-serverless architektury, musíme s narůstajícími požadavky výkon zvyšovat, což ale v praxi znamená dočasné odstavení systému od provozu a dodatečnou správu hardwaru a tento výpočetní výkon se nemění, pokud se opět nevylepsí, zatímco serverless řešení se automaticky škáluje.

## 3.2 Moderní techstack

Jelikož byla původní webová stránka i administrační systém postavena na poměrně zastaralých technologiích, bylo potřeba modernizovat a během vývoje jsem používal jen ty nejmodernější a mezi vývojáři velmi rozšířené technologie.

### 3.2.1 Frontend a UI/UX

Jedná se o grafické uživatelské rozhraní, v rámci webových aplikací většinou nejčastěji pomocí značkovacího jazyka HTML, stylovacího jazyka CSS a scriptovacího jazyka Javascript. Nicméně v dnešní době na tyto technologie, které existují delší dobu, existuje mnoho *frameworků* a *knihoven*, které nejen zjednodušují práci vývojářům, ale také celkově zlepšují *performance* aplikace či webu, která je díky tomu rychlejší, intuitivnější a použitelnější, což ve výsledku znamená, že například návštěvník webové stránky neztratí zájem, jelikož se stránka nebude dlouze načítat.

UI (user interface - uživatelské prostředí) a UX (user experience - v hrubém překladu uživatelská zkušenost) je sada pravidel a procedur, kterém mají za úkol zjednodušit uživatelskou interakci se systémem tak, aby se uživateli produkt používal co nejsnáze a byl co nejvíce intuitivní. Během vývoje administračního systému jsem se inspiroval návrhem, který jsem našel na designovém webu Dribble.com.

#### 3.2.1.1 HTML & CSS a Javascript

Jedná se o základní technologie, se kterými se dnes vyvíjejí webové stránky či aplikace a všechny ostatní technologie si z nich něco vzaly.

Základním stavebním kamenem webové stránky i aplikace je HTML dokument, který obsahuje všechny elementy, které má web zobrazovat.

Tyto elementy nastyluje stylovací jazyk CSS, což znamená, že v CSS dokumentu vybereme jednotlivý HTML element pomocí CSS selektoru a nastavíme mu například jinou barvu, šířku, či například nastavíme animaci, která se spustí po tom, co uživatel přes tento element přejede myší.

Pokud má mít web nějakou funkcionality, například po zadání dvou čísel vypočítat součet, použijeme Javascript. Ten je

vlastně logickým mozkem celého webu a stará se o tyto početní operace, získávání dat z databáze a jejich následné vložení do struktury HTML elementu (DOM) nebo interaktivitu s uživatelem .

### 3.2.1.2 React.js

React.js je Javascriptový *framework* vyvíjený firmou Meta (Facebook) a open-source komunitou vývojářů po celém světě a k dnešnímu dni se jedná o nejvíce používaný *framework* na světě pro jeho jednoduchost a modularitu.

Je optimální pro tvorby *SPA* (single page aplikací), jako je například administrační systém, protože velmi dobře pracuje s rychle měnícími se daty. Během vývoje react aplikace se celá aplikace rozdělí na jednotlivé komponenty, což má za následek jednoduchý a přehledný zdrojový kód a celou strukturu projektu, což je potřebné u velkého a komplexního projektu a já sám jsem během vývoje pocitil výrazný rozdíl v jednoduchosti vývoje oproti samotnému *vanilla* Javascriptu (bez použití frameworku či knihovny.)

```
1 // Component - tento blok kódu mohu použít kolikrát, kolikrát potřebuji
2 import React from "react";
3 import "../style/components/SummaryWidget.css";
4 class SummaryWidget extends React.Component {
5   constructor(props) {
6     super(props);
7   }
8   render() {
9     return (
10       <div className="summary-widget">
11         <div className="icon">
12           <>{this.props.icon}</>
13         </div>
14         <div className="data">
15           <p id="name">{this.props.data_name}</p>
16           <p id="value">{this.props.data_value}</p>
17         </div>
18       </div>
19     );
20   }
21 }
22
23 export default SummaryWidget;
```

Obr.4 - Ukázka jednoduché komponenty pomocí frameworku React

### 3.2.1.3 SASS/SCSS

**SASS** je *preprocessor* stylovacího jazyka CSS, což znamená, že styly psané v SCSS (sassy css) zpracuje a překompiluje to klasického CSS.

**SCSS** je další z mnoha stylovacích jazyků a jedná se o přímé rozšíření klasického CSS, přidává například proměnné, *nesting*, což znamená, že jednotlivé selektory můžeme psát přímo hierarchicky do sebe a pod sebe, což zlepšuje přehlednost jednotlivých stylů a například tzv. *mixins*, což jsou vlastně bloky stylů, které se dají použít vícekrát, což snižuje počet stylů, které musí vývojář napsat.

```

1  .element {
2      p {
3          color: $mainRed; // Proměnná
4      }
5      button {
6          @include submit-btn; // Použití mixin pomocí jeho unikátního jména
7      }
8  }

```

Obr.5 - Ukázka stylování v SCSS

```

1  .element > p {
2      color: #b30e2c
3  }
4  .element > button {
5      border: solid #b30e2c 1px;
6      background-color: #b30e2c;
7      color: white;
8  }

```

Obr.6 - Ukázka stylování v CSS

### 3.2.2 Backend

Termín *backend* označuje logickou část aplikace či webu a jeho úkolem je zpracování dat, se kterými uživatel manipuluje na *frontendu*. *Backend* má většinou na starost správu databází (*CRUD* operace), správu uživatelů a jejich autentizaci nebo například notifikace například ve formě mailu.

#### 3.2.2.1 Node.js

Node.js je tzv. *runtime* pro Javascript určený k vytváření vysoce škálovatelných *backend* aplikací a webových serverů. Toto má společné např. s jazykem PHP, který má stejné zaměření. JavaScript se tedy díky tomuto prostředí dá používat i na serveru a ne jen na druhém konci, u klienta. Avšak na rozdíl např. od zmíněného PHP je v Node.js kladen důraz na vysokou škálovatelnost, tzn. schopnost obsloužit mnoho připojených klientů naráz. Pro tuto vlastnost a vysokou výkonnost je dnes Node.js velmi oblíbený pro tvorbu tzv. API serverů pro klientské single page aplikace rovněž v Javascriptu.

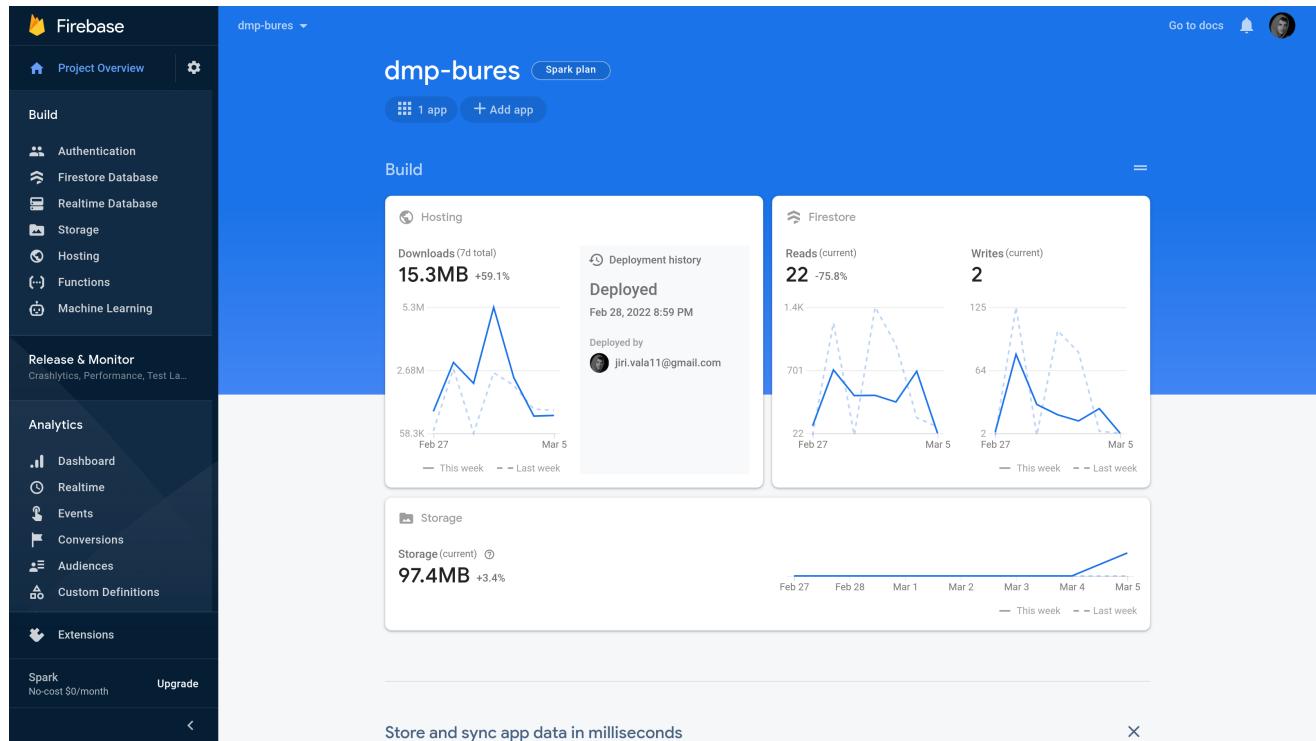
#### 3.2.2.2 Shell scripty

Shell scripty jsou sada příkazů, které po spuštění zpracovává a provede *shell*. Význam těchto scriptů v rámci tohoto projektu jsou *CI/CD* a *DevOps* Operace, které jsou detailněji popsány v bodu 8.3.

### 3.2.2.3 Firebase

Firebase je *cloud computing provider* nabízející služby jako hosting webových stránek, hosting databází, ale mnoho dalších jako je například *machine learning*, které však pro mě vzhledem k povaze projektu nejsou důležité. V tomto projektu zajišťuje všechny tyto zmíněné služby a nebýt Firebase, musel bych je všechny řešit sám, například pronájemem dalšího serveru, kde by byly databáze a podobně.

Databáze, které v rámci Firebase využívám, jsou tzv. *Firestore* a *Cloud Storage*.



Obr.7 - Nástěnka projektu ve Firebase Console

**Firestore** je NO-SQL databáze, což znamená, že jednotlivé záznamy uchovává v datovém typu Objekt, což je *key-value* datový typ, který slouží pro jednoduché popisování parametru daného předmětu, například člověka. Pro mé účely (ukládání záznamů o jednotlivých albech a fotografiích) je naprosto dostačující, avšak jedinou nevýhodou je, že NO-SQL databáze neumí vytvářet relace mezi jednotlivými tabulkami, tudíž, když jsem chtěl vytvořit relaci mezi albem a jeho fotografiemi, musel jsem v albu vytvořit pole s názvem *connectedImages* a v něm uchovávat *IDs* všech těchto fotografií.

The screenshot shows the Firebase Firestore interface. On the left, there's a sidebar with a collection named 'dmp-bures' containing 'albums', 'analytics', 'settings', and 'uploadedPictures'. The 'uploadedPictures' item is expanded, showing a list of document IDs: 2LpD6m3jC3Td8oBk8Xgu, 6hCDdHBhYyW3vbVQqpYF, B3hELcpARTn93ZMDAema, FPxyxS2XTlIouBB00WOZ, G7007uESHevSAd0eCfqM, KvrInwjtGtMzIDnSpwGY, LM1CXjvLPz7L21sL6nQI, M7xwYmxMsc9X9wPz0T5, MmmKJkyk1DSIhSSLJAPI, PLawcfjnZiX6wkmqGbRQ, QJqQf0jHGZMI1kwAaxu8, Rjs7LbAdvxMi9KFE0TDR, UdUqqTmEPBjUJcUXxnQdL, VlkGrXkCyJTndSt3c6Kjh. The middle panel has a header '+ Add document' and a list of these document IDs. The right panel shows the detailed view for the document '2LpD6m3jC3Td8oBk8Xgu', which has 0 fields. The data shown is:

```

id: "1642100551696"
name: "Hlavní"
description: "1984"
id: "2LpD6m3jC3Td8oBk8Xgu"
name: "011 Štramberk"
size: 4884001
total_likes: 0
type: "image/jpeg"
uploadDate: "Sun, Feb 06 2022"
url: "https://firebasestorage.googleapis.com/v0/b/dmp-bures.appspot.com/o/2LpD6m3jC3Td8oBk8Xgu?alt=media&token=0fd1566c-f90d-43af-a3b9-9442c2f2bc4c"

```

Obr.8 - Firebase Firestore záznamy jednotlivých fotografií

**Cloud Storage** je clouдовé úložiště pro všechny typy souborů. Avšak vzhledem k tomu, že by pan fotograf měl nahrávat jen fotografie, ve scriptu, který řídí nahrávání nových fotografií je implementováno jednoduché ověření, zda li soubor, který se snaží nahrát, doopravdy je typu obrázku/fotky. Nicméně kdyby si to zákazník přál, jsem schopen implementovat i nahrávání souborů jiných, než fotografií.

```

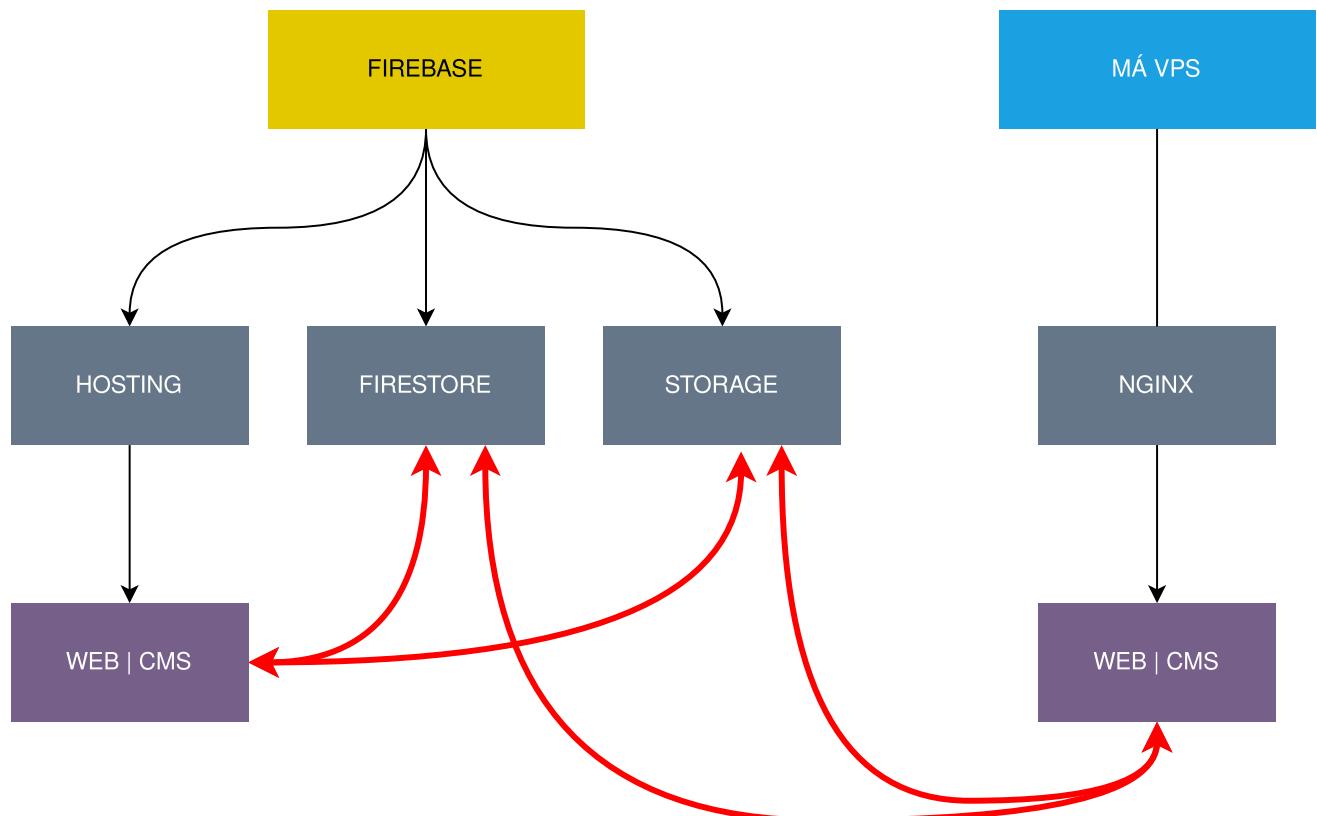
1  export const Images = {
2    Meta: {
3      /**
4       * @param {*} File File that is uploaded to `input` element
5       * @returns {bool} `true` if the file is type of image
6     */
7     isImage: (file) => {
8       if (file["type"].split("/")[0] === "image") {
9         return true;
10      } else return false;
11    },
12  },
13  // ... další helper funkce
14};
15

```

Obr.9 - Ukázka funkce, která zjišťuje, zda je nahraný soubor typu image

## 4. Struktura projektu s ohledem na DMP

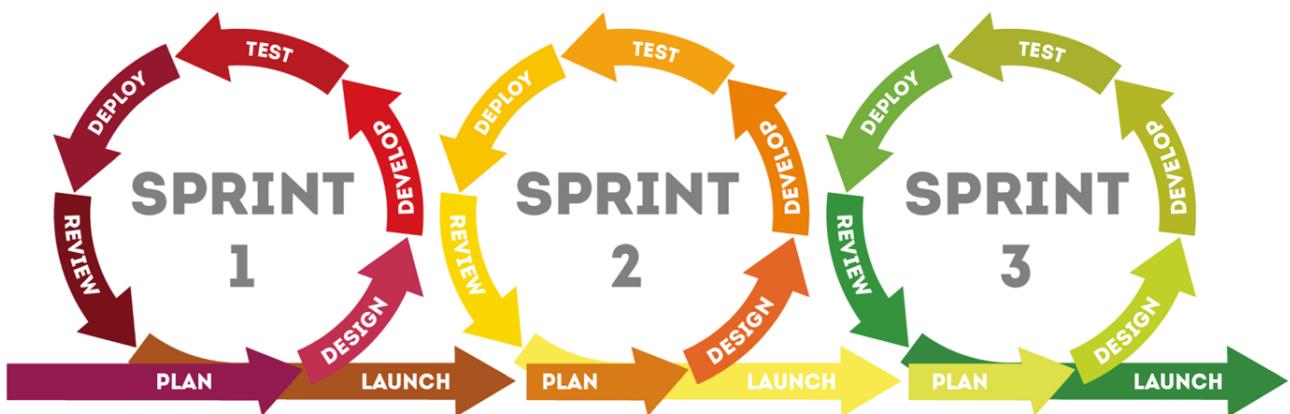
Projekt je rozdělen na dvě částí - komerční a školní (DMP). Co se týče části komerční, všechny služby okolo hostingu a databáze bude zprostředkovávat Firebase, jelikož je to v rámci dlouhodobého hlediska nejjednodušší a hlavně nejlevnější řešení. Školní část (DMP) budu zcela spravovat já, tudíž mám pronajatý vlastní server, na kterém budu zprostředkovávat hosting a backend služby, nicméně všechna data budou poskytována a požadována z databází komerční části.



Obr.10 - Diagram popisující kompletní strukturu projektu

## 5. Průběh vývoje

Vývoj webové prezentace začal někdy v dubnu roku 2021 a během té doby jsme si se zákazníkem vyměnili nespočet emailů a nespočet telefonních hovorů. A i přesto, že jsem se snažil o co nejvíce agilní vývoj, několikrát se stalo, že po tom co jsme daný vývojový *sprint* označili za ukončený a já se tak mohl přesunout na další část vývoje nebo další funkcionality, zákazník napsal email s tím, že by rád ještě něco změnil, což není až takový problém, nicméně některé z těchto změn byly poměrně velké a znamenaly například přetvoření celého schéma databáze.



Obr.11 - Princip agilního vývoje, který se odehrává v takzvaných sprintech

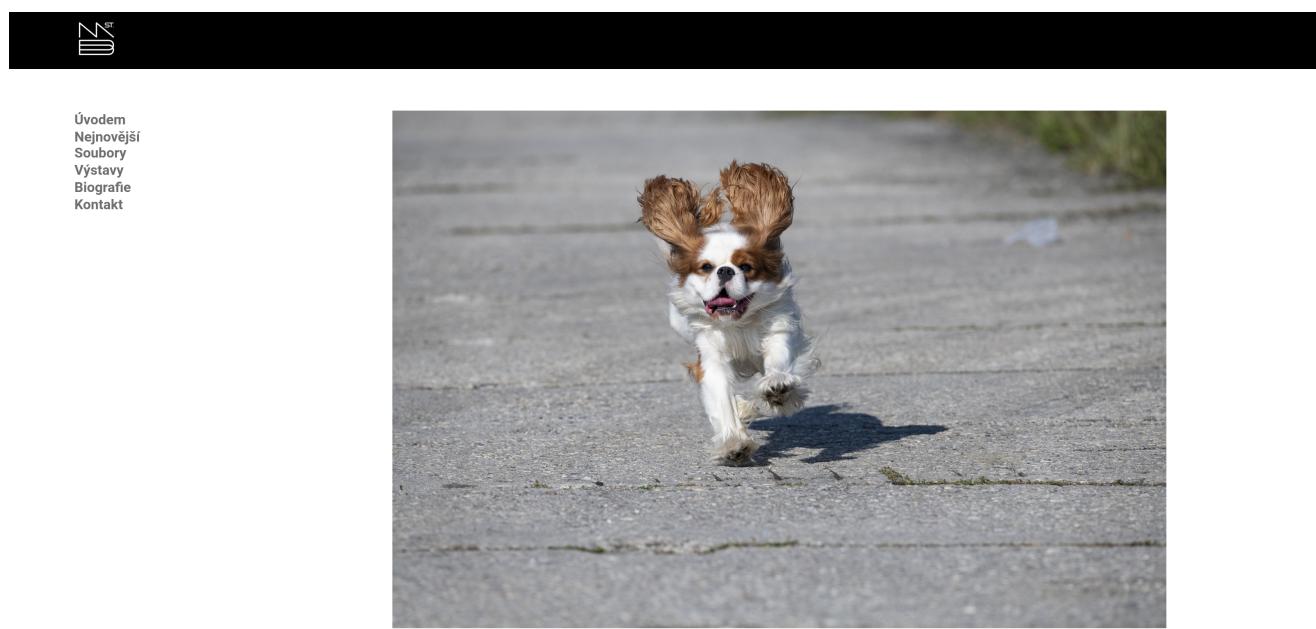
Vývoj obou systémů (webové stránky a administračního systému) začal nejprve ve vanilla Javascriptu, což se pro mě jevilo jako nejsnazší řešení, jelikož v té době jsem neuměl ovládat žádný *framework*, který by mi vývoj usnadnil. Čistý Javascript je sám o sobě pro projekt, jako je webová stránka naprostoto dostačující a i když bych nyní vývoj začal s *frameworkem*, jako je například Gatsby.js, není to až tak potřeba. Nicméně, jak jsem do administračního systému přidával dál složitější a složitější funkcionality, došlo mi, že se v tom začínám ztrácat a proto jsem se rozhodl, že celý administrační systém *refaktorují* a přepíši pomocí *frameworku* React.js, který nejen že zvýší celkovou *performance* a přehlednost nejen struktury projektu, ale i samotného kódu, což se vyplatilo, protože nyní mám přesný přehled o tom, kde co je a jak to funguje.

## 6. Webová prezentace

Webová prezentace bude panu fotografovi sloužit jako virtuální galerie fotek, aby lidé, které jeho tvorba zajímá, měli všechny jeho fotografie na "dosah ruky" a vzhledem k současné koronavirové situaci je více než vhodné, aby nemuseli chodit přímo na výstavu.

Samotná webová prezentace obsahuje přesně to, co obsahovala ta stávající, tedy, hlavní stránku, na které je jen galerie z alba "Hlavní", jelikož se jedná o fotografie, které se panu fotografovi líbí nejvíce a o kterých si myslí, že by mohly oslovit nejvíce lidí. Po kliknutí náhledovou fotografií se spustí galerie, která avšak nedovoluje rozkliknout informační box s více informacemi o dané fotografii.

Data, které web používá k svému fungování nejsou dynamická, jelikož by to z principu fungování statické webové stránky nemělo smysl. Data se stahují z databáze ve chvíli, kdy uživatel přijde na stránku poprvé. To znamená, že pokud uživatel nahraje novou fotografiu, změny na webu se projeví až ve chvíli, kdy uživatel stránku znova načte.

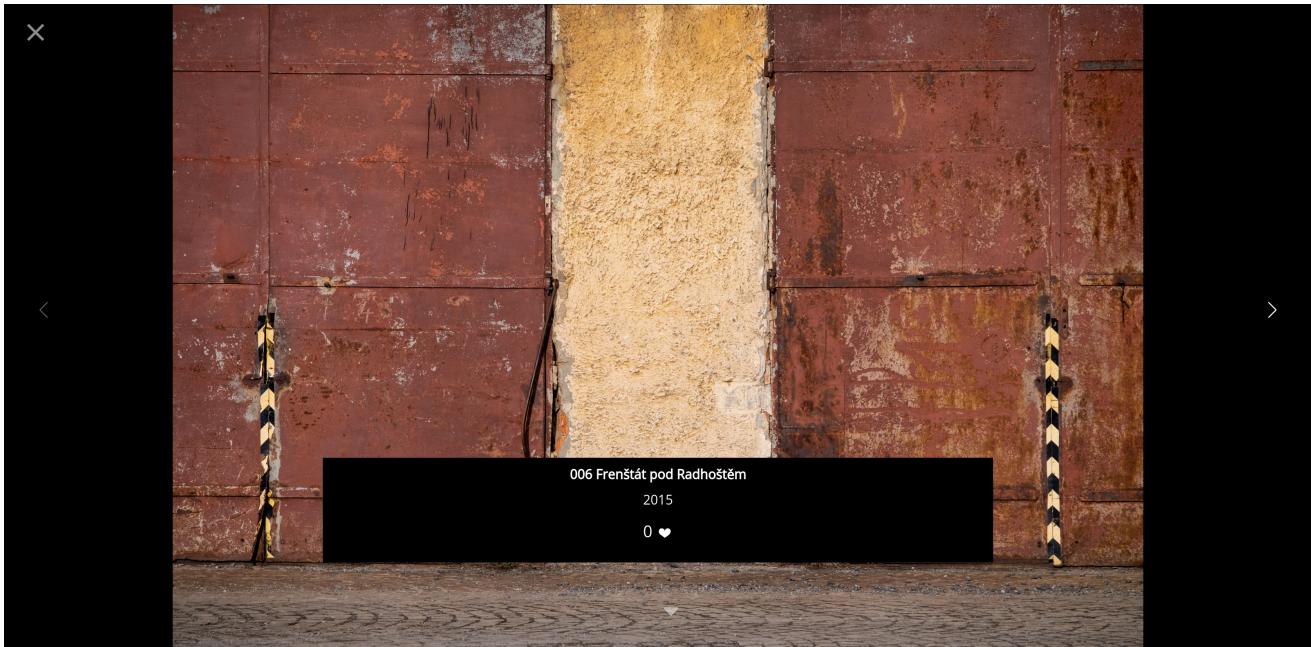


2021 © Mgr. Milan Bureš st.  
Made with ❤ by Orexin.

Obr.12 - Hlavní stránka webové prezentace

Dále stránku "Úvodem", která slouží jako předmluva pro celý web, který má simulovat umělecké dílo spolu s jeho uměleckými fotografiemi. Tento text je samozřejmě editovatelný v administračním systému.

Odkaz na stránku alba "Nejnovější". Nejedná se o, jak by se mohlo podle názvu zdát, album s fotografiemi, které byly nahrány nedávno, ale o fotografie, které byly nedávno vyfoceny, ostatní alba totiž mohou obsahovat i fotografie, které byly vyfoceny před rokem 2000. To znamená, že se jedná o výběr fotografií, které pan fotograf nedávno vyfotil a rozhodl se, že je bude sdílet. Při vývoji této podstránky jsem musel vymyslet systém, jakým budu fotografie na webu zobrazovat, jelikož si zákazník přál, aby místo galerie, která je použita u všech ostatních alb, byla použita tabulka s náhledy fotografií. Abych tohoto docílil, spolu se správnou *pagination*, musel jsem pole se všemi fotografiemi rozdělit na části po 6 fotografiích a poté je v těchto částečkách zobrazit na webu. Po kliknutí na náhled fotografie se však otevře galerie, jako u všech ostatních alb.



Obr.13 - Galerie po rozkliknutí náhledové fotografie

*Dropdown menu s odkazy na existující alba. Jelikož jsou alba dynamickými prvky v databázi, musel jsem vymyslet způsob, jakým budu vytvářet podstránky jednotlivých alb, nemohl jsem totiž ručně pro každé album vytvořit podstránku pokaždé, co ji uživatel administračního systému vytvoří. Implementoval jsem tedy *algoritmus*, který na základě *id* obsažené v URL linku alba vytáhne z databáze správný záznam o albu právě pomocí daného *id*.*

```
1 // This file manipulates with URL query params using which
2 // we show correct album
3
4 // Imports
5 import { async } from "regenerator-runtime";
6 import "../css/collections.css";
7 import { Collections, Images, Storage } from "./core";
8 import { Gallery } from "../components/gallery";
9
10 // Variables
11 let collectionId = new URL(document.location).searchParams.get("collectionId");
12 let collectionObject = Storage.getSpecific(collectionId);
13 const galleryWrapper = document.querySelector("#gallery-wrapper");
14
15 document.querySelector(".collectionName").innerText = collectionObject.name;
16 galleryWrapper.appendChild(new Gallery(collectionObject.images));
```

Obr.14 - Ukázka systému získávání správného alba

Stránku "Výstavy", která obsahuje seznam všech výstav, na kterých pan fotograf kdy vystavoval své fotografie. Samozřejmě se jedná o editovatelný text.

"Biografie" popisuje dosavadní život pana fotografa společně s jeho fotografií.

V poslední řadě "Kontakt", kde lze najít veškeré kontaktní možnosti. Formulář, který podstránka obsahuje, bude pomocí Firebase Functions posílat email a *push notifikaci* panu fotografovi vždy, když mu někdo napíše email, aby se nikdy nestalo, že nebude vědět o tom, že mu někdo psal.

The screenshot shows a contact form with the following fields:

- Vaše jméno a příjmení: Jan Novák
- Váš E-mail: jan-novak@priklad.cz
- Vaše telefonní číslo: +420 123 456 789
- Odesílání souhlasíte se zpracováním Vašich osobních údajů.
- Odkaz na "Odeslat" tlačítko.
- Icons for phone (+420 123 456 789), email (info@info.cz), Instagram (@milanbures), and Facebook (Milan Bureš).

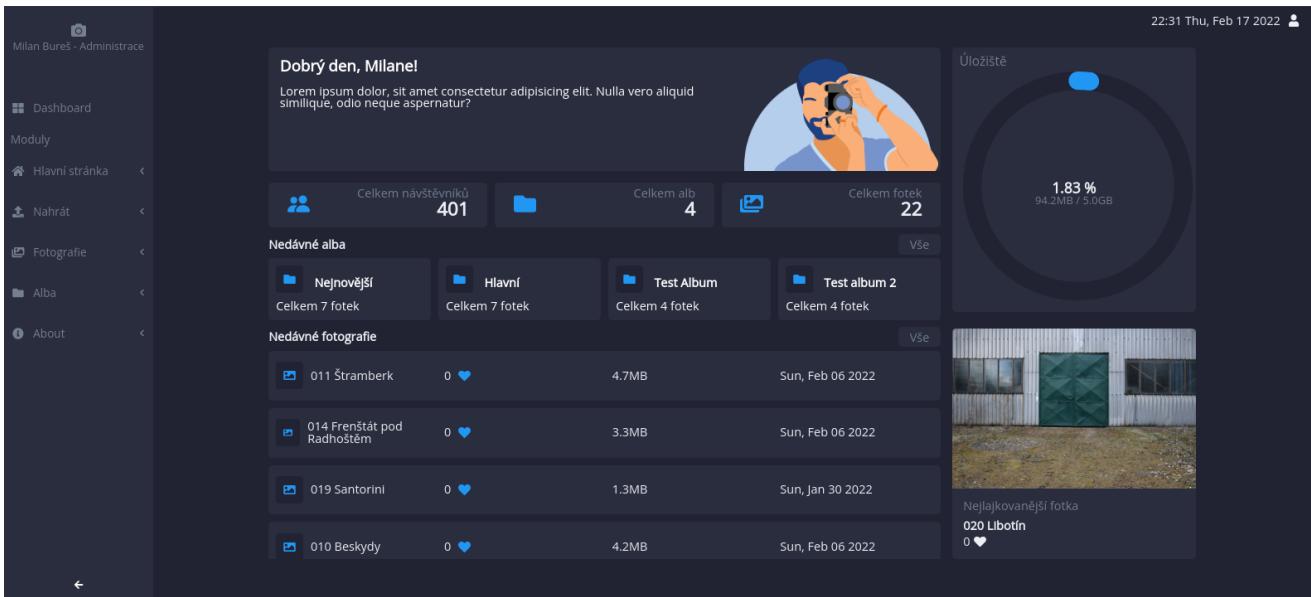
2021 © Mgr. Milan Bureš st.  
Made with ❤ by Orexin.

Obr.15 - Stránka s kontaktním formulářem.

Jelikož je webová prezentace napsána ve vanilla Javascriptu, v budoucnu mám v plánu ji přepsat pomocí *frameworku* Gatsby.js, který slouží pro generování statických HTML stránek za použití React.js, což bude mít za následek nejen mnohonásobně jednodušší budoucí vývoj, ale také rychlejší načítací časy.

## 7. Administrační systém

Administrační systém je systém které zprostředkovává manipulaci s daty, které se objevují na dané webové stránce. Jednoduše řečeno, díky administračnímu systému je uživatel schopen své fotografie nahrávat, mazat, upravovat jejich názvy a popisy, stejně tak jako uvidí, která z nich je nejvíce oblíbená na základě *lajků* a podobně. Dále administrační systém obsahuje funkcionality pro manipulaci s alby, do kterých bude uživatel své fotografie rozdělovat. Ty samozřejmě může obdobně vytvářet, mazat a upravovat jejich fotografický obsah. V neposlední řadě systém obsahuje i informační nástěnku, kde je velmi dobře vidět například kolik fotografií je v danou chvíli nahraných na webu a v jakých albech, kolik úložného prostoru jeho fotografie zabírají a nebo třeba seznam posledních nahraných fotografií, aby je nemusel hledat.



Obr.16 - Ukázka nástěnky administračního systému

Všechny data, se kterými uživatel pracuje, jsou zobrazována v reálném čase, což znamená, že pokud, například upraví název fotografie, tato změna se ihned projeví jak v samotném administračním systému, tak v databázi a tím pádem i na webové stránce.

Díky Firebase Auth se do systému nedostane uživatel, který **není** autentifikovaný, tedy přihlášený. Prozatím celý autentifikační systém funguje za pomocí emailu a hesla, jelikož web spravuje jeden člověk a není potřeba, abych implementoval například autentifikaci pomocí Google či jiného již existujícího účtu.

Celá aplikace je "obalená" v auth elementu, který po úspěšném přihlášení (ověří se, zda uživatel existuje na Firebase Auth serveru) uchovává informace o přihlášeném uživateli a pokud záznam o tomto uživateli existuje, auth element jej pustí dál, jinak bude přesměrován zpět na přihlašovací obrazovku.

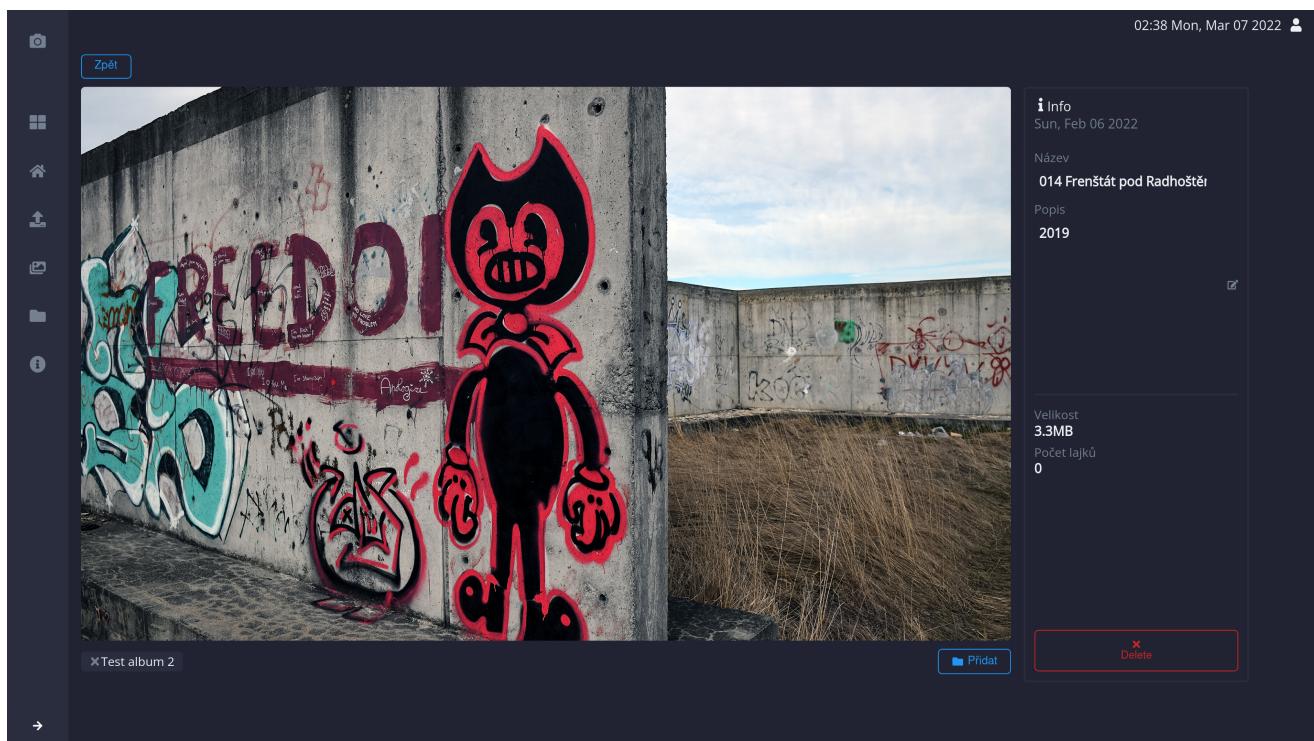
Administrační systém obsahuje veškeré nástroje k tomu, aby uživatel mohl manipulovat s fotografiemi a alby, které nahraje a vytvoří. Po nahrání jedné či více fotografií se tyto fotografie objeví na podstránce "Fotografie" v přehledném seznamu. Kliknutím na jednotlivé fotografie se uživatel dostane na stránku pro manipulaci s jednotlivou fotografií, kde ji může přejmenovat, přidat popis, přiřadit do alba nebo ji z alba odstranit, a nebo fotografií odstranit úplně. Také lze vidět metadata fotografie, jako například kdy byla nahrána, či případně kolikrát ji návštěvníci webové prezentace udělili "lajk".

Milan Bureš - Administrace 02:53 Mon, Mar 07 2022

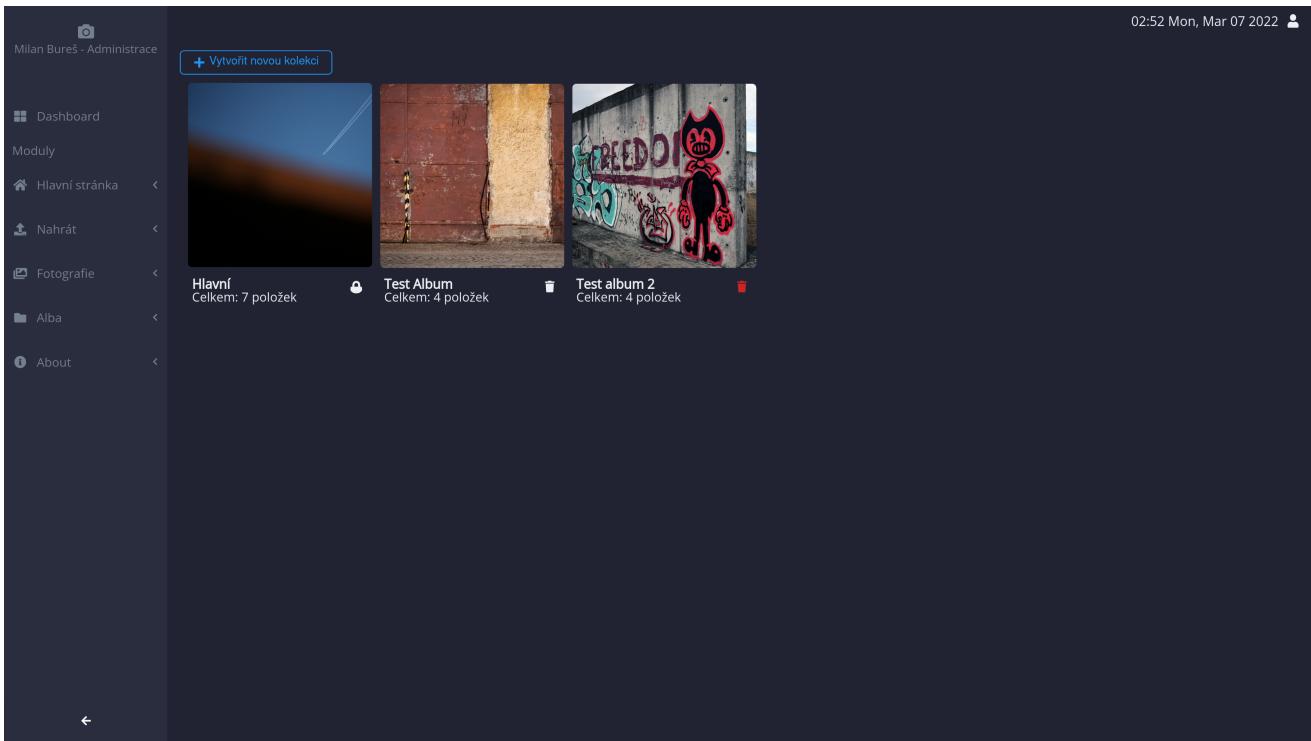
### Všechny fotografie

	Název	Počet lajků	Velikost	Datum nahrání
	011 Štramberk	0	4.7MB	Sun, Feb 06 2022
	014 Frenštát pod Radhoštěm	0	3.3MB	Sun, Feb 06 2022
	019 Santorini	0	1.3MB	Sun, Jan 30 2022
	010 Beskydy	0	4.2MB	Sun, Feb 06 2022
	022 Frenštát pod Radhoštěm	0	4.4MB	Tue, Feb 08 2022
	008 Větřkovice	0	6.4MB	Sun, Feb 06 2022
	002 Praha	0	7.0MB	Sun, Feb 06 2022
	001 Rhodos	0	3.0MB	Sun, Jan 30 2022
	003 Tichá	0	9.8MB	Wed, Jan 26 2022
	004 Lubina	0	5.7MB	Thu, Jan 27 2022

Obr.17 - Stránka se seznamem všech nahraných fotografií.



Obr.18 - Stránka pro manipulaci jednotlivé fotografie.



Obr.19 - Stránka se všemi aktuálně vytvořenými alby.

## 8. VPS - Virtual Private Server

VPS bude sloužit k realizaci školní části projektu a de facto bude kopírovat veškerou funkcionality Firebase. Bude zajišťovat *hosting* celého projektu pomocí webového serveru nginx, díky kterému bude uživatel moc přistoupit na webovou prezentaci i administrační systém, a *backend* služby, jako například *private API* pomocí *Node.js*, aby *Github Actions Hook* mohl provést *HTTP Request*, který spustí *CI/CD* scripty.

### 8.1 Operační systém GNU/Linux

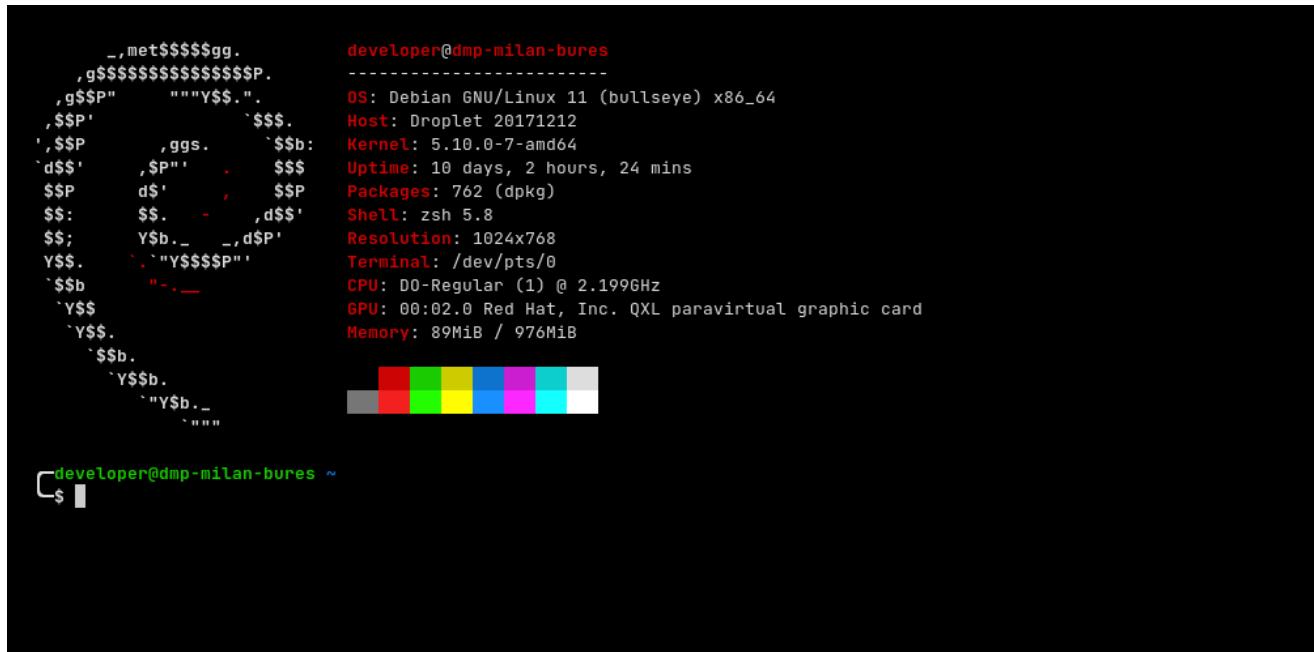
V dnešní době je jako serverový operační systém nejvíce rozšířený GNU/Linux pro jeho minimalističnost a poměrně jednoduchou konfiguraci. Podle [hostingtribunal.com](https://hostingtribunal.com) až 96% serverů z 1 milionu celosvětově nejpoužívanějších serverů používá Linux jako svůj operační systém a proto jsem se i já rozhodl použít Linux a nejen z toho důvodu, že jej dennodenně používám.

#### 8.1.1 Debian 11

Jako *cloud computing providera* pro můj VPS jsem si vybral DigitalOcean, vzhledem k jeho ceně a celkové nabídce se jeví jako nejlepší. Původně jsem na svůj VPS chtěl nainstalovat Arch Linux, jelikož se jedná o Linuxovou distribuci, kterou dennodenně používám, nicméně má jednu velkou nevýhodu - nemá vlastní instalátor (i když nyní již existuje instalacní script), což mě osobně při instalaci v domácích lokálních podmínkách vůbec nevadilo, ale v případě instalování Arche na VPS by to znamenalo, že bych musel mít vlastnoručně přizpůsobené ISO, které bych použil k instalaci, jelikož DigitalOcean nenabízí Arch k nainstalování na jejich VPS, které nazývají *Droplets*.

Volba tedy padla na *Debian 11*, jelikož je v jeho minimalistické instalaci poměrně "čistý" a tudíž neobsahuje zbytečné programy a podobně (*bloatware*). Vzhledem k tomu, že tento "počítač" bude sloužit jako server, není potřeba a je až zbytečné instalovat jakékoli desktopové prostředí *GUI* (Graphical User Interface), jelikož by to mělo nejen za následek zvýšení požadavků na výpočetní výkon serveru, což je nežádoucí, jelikož chceme, aby operační systém serveru zabíral co nejméně prostoru a prostředků, a zároveň celá konfigurace systému je proveditelná jen za pomoci příkazové řádky

*CLI* (Command Line Interface). Na obrázku pod lze vidět, jak *CLI-only* Debian v *idle* stavu (stavu, kdy neprovádí žádné operace) zabírá na paměti RAM jen 89MB z dostupných 1GB, tedy necelých 10%.



```
_met$$$$$gg.  
g$$$$$$$$$$$$$P.  
,$$P"    """$Y$.". .  
,$$P'      '$$$.  
'$$P     ,ggs.   '$$:  
'd$$'   ,$P''  .   $$$  
$$P   d$'   ,   $$P  
$$:   $$_. -   ,d$$'  
$$;   Y$b._ _ ,d$P'  
Y$$.   . "Y$$$P"  
'$$b   "-. __  
'Y$$.  
'$$.  
'$$.b.  
'Y$$.b.  
'"Y$b..  
`"'''  
  
developer@dmp-milan-bures ~
```

Obr.20 - Diagnostický nástroj neofetch zobrazující základní informace o systému po tom, co se na něj připojíme pomocí SSH

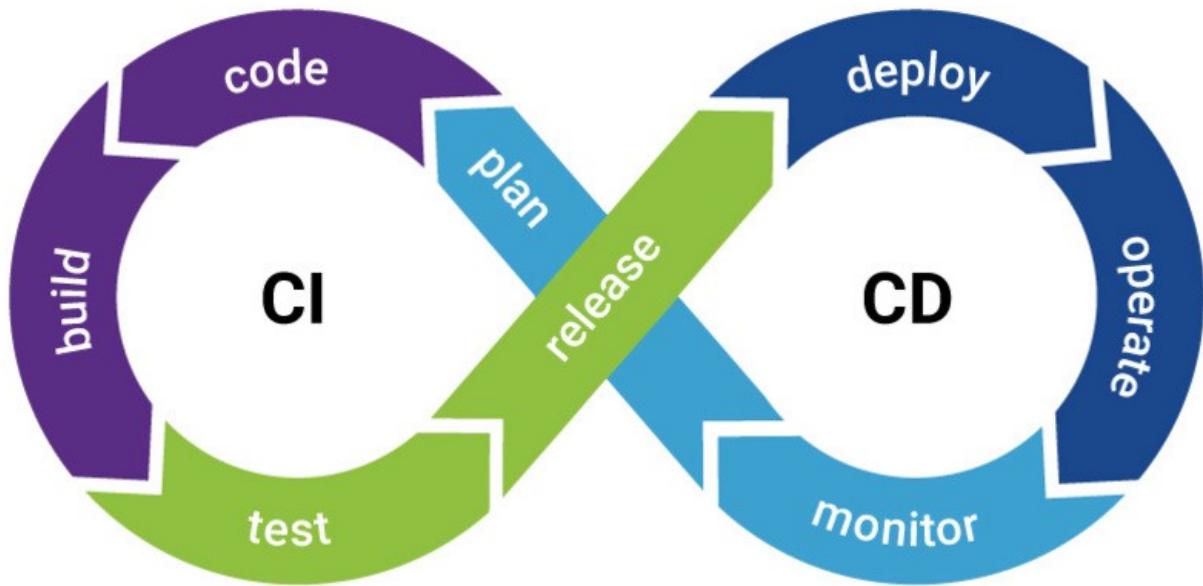
## 8.2 Nginx

Aby hardwarový server mohl poskytovat uživatelům z vnější sítě webové stránky či aplikace, musí na něm běžet softwarový webový server a optimálně *load balancer*, aby nápor uživatelů nepřesáhl výkon serveru.

Jeho základní výhodou oproti konkurenčnímu webovému serveru Apache je právě zmíněný *load balancing*, díky kterému nginx mnohem lépe zvládá nápor požadavků na server a rovnoměrně rozkládá zátěž.

## 8.3 CI/CD

CI (Continuous Integration) a CD (Continuous Delivery) je proces, který umožňuje nepřetržité dodávání aktuální verze projektu k testování či produkčnímu spuštění. V praxi to znamená, že pokud tým vývojářů provede změnu a uloží ji do *remote repositáře* (místo pro ukládání zdrojových kódů v cloudu), tato změna se automaticky projeví (*deployne*) na produkční server (server, na kterém je spuštěna aktuální verze projektu a je poskytována uživatelům z venčí).

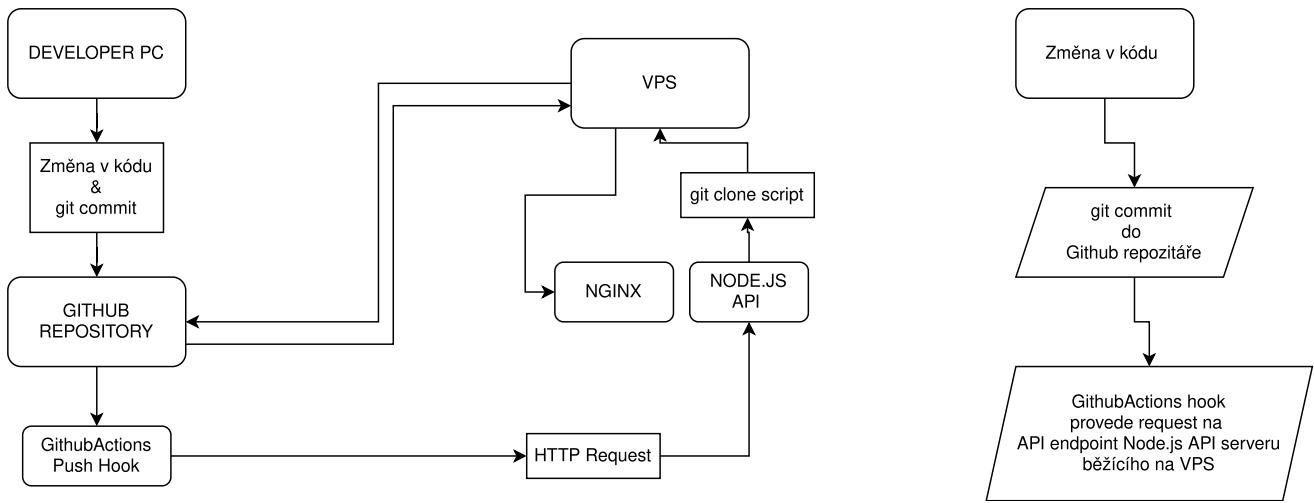


Obr.21 - Diagram znázorňující princip fungování CI/CD

### 8.3.1 CI/CD scripty na VPS

Po té, co provedu změnu v zdrojovém kódu webové prezentace nebo administračního systému a tuto změnu nahraji pomocí verzovacího systému *git* do Github repozitáře, je potřeba, aby se tato změna projevila na produkčním serveru. K tomu využiji *shell* scripty, Github actions hooky a Node.js API.

Celá operace bude probíhat tak, že jakmile *pushnu* (nahraji) změny do repozitáře, Github Actions zaznamenají, že se snažím o provedení nějaké změny ve zdrojovém kódu a spustí interní script, který provede požadavek (request) na mé Node.js API, které zase pro změnu spustí script přímo na VPS, který si *pullne* (stáhne) nejnovější verzi zdrojového kódu přímo do adresáře s soubory, odkud je poté webový server nginx nabízí uživatelům z vnější sítě.



Obr.22 - Diagram znázorňující princip stahování poslední změny na lokální úložiště VPS

## 9. Závěr

Během vývoje tohoto projektu jsem se naučil mnoho nových věcí, které věřím, že mě v mém profesionálním životě posunuly dál a jsem rád, že jsem si jako dlouhodobou maturitní práci vybral právě tento projekt. Mrzí mne však, že jsem začal pozdě s vývojem pomocí *frameworku React.js*, jelikož bych měl vývoj snazší a neztratil bych čas vyvíjením administračního systému ve *vanilla Javascriptu* a tím pádem bych se mohl soustředit na důležitější věci, než je opětovné vyvíjení systému, který jsem již jednou vytvořil.