

# E-shop s CMS

***KIV/WEB – Semestrální práce***

student:	<i>Martin Riedl</i>
osobní číslo:	<i>A22B0242P</i>
email:	<i>marty69@students.zcu.cz</i>
datum:	<i>28.11.2023</i>

## ***Nutné nastavení pro správnou funkcionalitu:***

Ve složkách `.env` a `.env.local` nastavit `secrets` a `variables`.

```
STRIPE_SECRET_KEY=""
STRIPE_WEBHOOK_SECRET=""
NEXT_PUBLIC_CLOUDINARY_CLOUD_NAME=""
STRIPE_SECRET_KEY= STRIPE_PUBLISHABLE_KEY=
NEXTAUTH_SECRET=""
GOOGLE_CLIENT_ID=
GOOGLE_CLIENT_SECRET=
```

**Předvedení práce:**

<https://youtu.be/rw0xj6yN5TY>

**Struktura složek:**

<https://youtu.be/2ffQJXQXsdM>

**Github:**

<https://github.com/real-marty/sp-web>

## **Úvod:**

V této semestrální práci se zaměřuji na vývoj moderního e-shopu s využitím pokročilých technologií pro front-end a back-end. Cílem je vytvořit funkční online obchodní platformu, kde super uživatel může definovat produkty, které jsou následně prezentovány koncovým uživatelům. Tento projekt se zaměřuje na implementaci celého procesu od prezentace produktů po finální objednávku s využitím platební brány Stripe.

Hlavní technologický stack tohoto projektu zahrnuje Next.js 13 na straně front-endu, PostgreSQL pro databázové řešení a ORM Prisma pro efektivní správu databáze. Tato kombinace technologií byla zvolena s ohledem na jejich moderní charakter a schopnost poskytnout robustní, ale zároveň flexibilní řešení pro náročné webové aplikace.

Tento projekt je nejen praktickým cvičením v oblasti webového vývoje, ale také příležitostí se naučit a efektivně využít nejnovější technologie a postupy. Klíčovým prvkem práce je vytvoření uživatelsky přívětivého rozhraní s důrazem na intuitivní navigaci a efektivní zobrazování produktů, stejně jako na bezproblémový proces objednávání a placení.

Semestrální práce je strukturována do několika hlavních částí. První část se věnuje stručnému popisu použitých technologií, včetně případných ukázkových implementací z webové stránky. Následně je podán popis adresářové struktury aplikace, který poskytuje přehled o obsahu jednotlivých adresářů a souborů. Dále popis architektury aplikace a rozvržení databáze, přičemž databázové závislosti jsou ilustrovány pomocí UML diagramu. Na závěr je uveden seznam defaultních uživatelů a schválené zadání práce.

Cílem této práce je nejen vytvořit funkční e-shop, ale také prohloubit porozumění moderním technologiím a postupům v oblasti webového vývoje, což je klíčové pro mé další profesní rozvoj v oboru programování.

## Použité technologie:

### Next.js (next):

Populární framework pro React určený pro vytváření aplikací s vykreslováním na straně serveru a statických webových aplikací. Nabízí funkce jako souborové směřování, API cesty a integraci s různými nástroji pro front-end.

### React (react, react-dom):

JavaScriptová knihovna pro vytváření uživatelských rozhraní, známá především díky své komponentové architektuře. Je široce používána pro vývoj jednostránkových aplikací.

### Tailwind CSS (tailwindcss):

Utility-first CSS framework pro rychlé vytváření vlastních designů přímo v HTML.



Kód 1- nastavení pozadí pomocí tailwindcss

### Pluginy Tailwind CSS (@tailwindcss/forms, @tailwindcss/aspect-ratio, @tailwindcss/typography):

Tyto pluginy pro Tailwind CSS poskytují další utility pro formuláře, poměr stran a typografii.

### Prisma (@prisma/client, prisma):

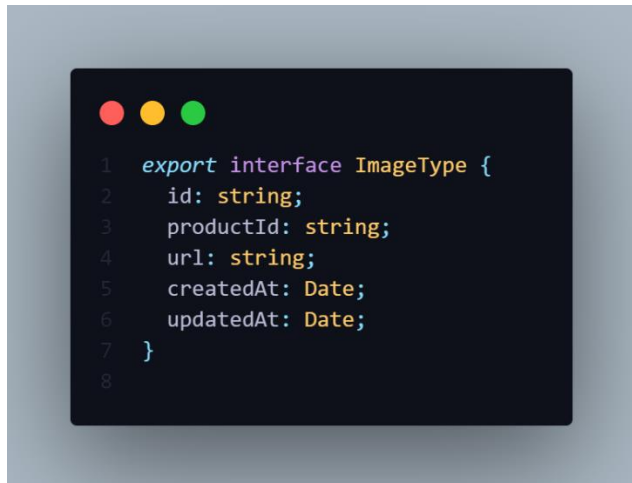
Open-source databázový toolkit, který zahrnuje ORM (Object-Relational Mapping) pro Node.js a TypeScript. Zjednodušuje přístup k databázi a poskytuje robustní nástroj pro vytváření dotazů.



Kód 2 - vytvoření záznamu v tabulce color

### TypeScript (typescript):

Nadstavba JavaScriptu, která přidává statické typy. TypeScript je navržen pro vývoj velkých aplikací a je překládán do JavaScriptu.

A screenshot of a code editor with a dark background and light-colored text. The code defines an interface named 'ImageType' with several properties: 'id' of type 'string', 'productId' of type 'string', 'url' of type 'string', 'createdAt' of type 'Date', and 'updatedAt' of type 'Date'. The code is as follows:

```
1 export interface ImageType {  
2   id: string;  
3   productId: string;  
4   url: string;  
5   createdAt: Date;  
6   updatedAt: Date;  
7 }  
8
```

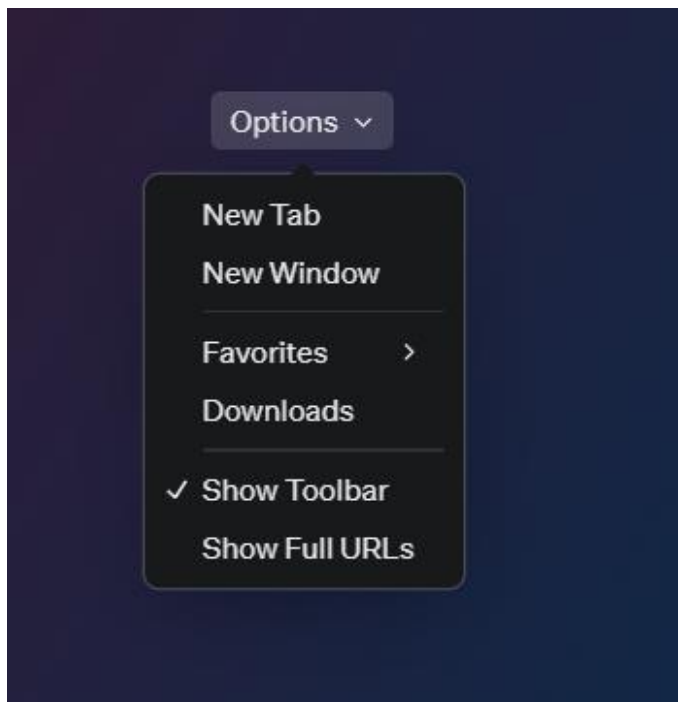
*Kód 3- definování interface ImageType pro typovou korekci typescriptu*

### ESLint (eslint-config-next):

Nástroj pro identifikaci a hlášení vzorů v kódu ECMAScript/JavaScript, s konfigurací specifickou pro projekty Next.js.

### Radix UI (@radix-ui/\*):

Řada nízkoúrovňových UI primitiv pro vytváření kvalitních, přístupných designových systémů a webových aplikací. (uživatel si pak může přizpůsobit vzhled těchto primitiv)  
Primitivum pro menu:



### Framer Motion (framer-motion):

Knihovna pro React určená pro animace.

### React Hook Form (react-hook-form):

React Hook Form je knihovna pro React, která zjednodušuje práci s formuláři a zároveň zvyšuje výkon aplikace tím, že minimalizuje počet překreslování komponent. Tato knihovna využívá přímou práci s DOM formulářových prvků, což eliminuje potřebu ukládat hodnoty formuláře do interního stavu. Díky integraci s React hooky je její zapojení do existujících aplikací snadné a intuitivní. Kromě toho podporuje integraci s knihovnami pro validaci, jako je Yup, což umožňuje efektivní a real-time validaci formulářů. React Hook Form je kompatibilní s TypeScriptem, což přináší výhody statického typování, a nabízí různé režimy validace pro flexibilní řízení chování formulářů. Výsledkem je lepší zkušenost pro vývojáře a efektivnější práce s formuláři ve složitějších React aplikacích.

```
1  const mailSchema = z.object({  
2    email: z.string().email("Neplatná e-mailová adresa."),  
3  });
```

Kód 4 - definice email schématu pomocí zod, chybová hláška

```
1  const {  
2    register,  
3    handleSubmit,  
4    formState: { errors },  
5    reset,  
6  } = useForm({  
7    resolver: zodResolver(mailSchema),  
8    defaultValues: { email: "" },  
9  });
```

Kód 5 - "vytažení" potřebných funkcionalit, nastavení resolveru; defaultní hodnota

```

1 <input
2     disabled={loading}
3     type="text"
4     id="email"
5     {...register("email")}

```

*Kód 6 - registrace email inputu pro ReactHookForm*

### Zustand (zustand):

Malé, rychlé a škálovatelné řešení pro správu stavu s použitím zjednodušených principů flux. Když například potřebuji otevřít a zavřít menu z více míst aplikace, může být velice užitečné.

```

1 import { create } from "zustand";
2
3 interface UseStoreNavigationMobileMenuStore {
4   isOpen: boolean;
5   onClose: () => void;
6   onOpen: () => void;
7 }
8
9 export const UseStoreNavigationMobileMenu =
10   create<UseStoreNavigationMobileMenuStore>((set) => ({
11     isOpen: false,
12     onOpen: () => set({ isOpen: true }),
13     onClose: () => set({ isOpen: false }),
14   }));
15

```

Axios (axios):

Populární JavaScriptová knihovna používaná k provádění HTTP požadavků.

```
1  const onDelete = async () => {
2    try {
3      await axios.delete(`/api/role/${row.id}`);
4      toast.success("Uživateli byly úspěšně odebrány práva.");
5    } catch (error) {
6      toast.error("Nedostatečná práva, nebo uživatel neexistuje.");
7    } finally {
8      router.refresh();
9    }
10  };
```

*Kód 7- posílání delete requestu pomocí knihovny Axios*

Stripe (stripe):

JavaScriptová knihovna pro integraci Stripe API pro zpracování plateb.

## Architektura a kořenová struktura

Od verze Next.js 13 byla představena nová kořenová složka s názvem app, která přináší nové možnosti pro routování, sdílení komponent a definování layoutů. Tato dokumentace poskytuje přehled funkcí a nejlepších postupů pro používání složky app.

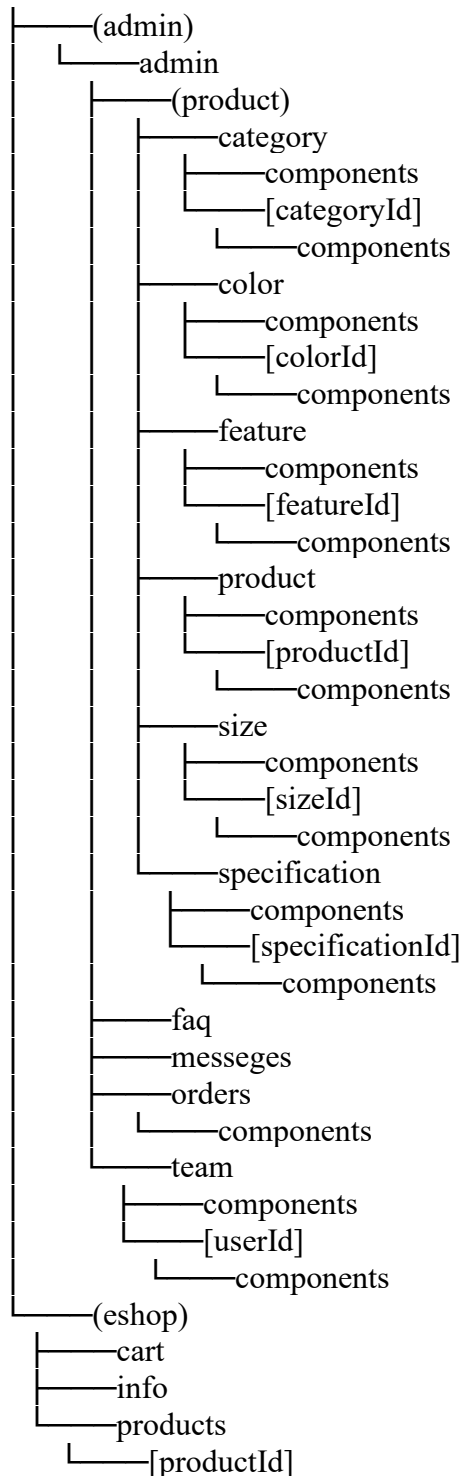
V jádru této změny je nový systém routování, který umožňuje vývojářům vytvářet strukturované a intuitivní cesty ve svých aplikacích. Složka app používá složkové routování, kde hierarchie složek a souborů přímo odráží strukturu URL cest. To umožňuje vývojářům snadno navigovat a spravovat strukturu svých aplikací.

Dalším klíčovým aspektem složky app je možnost definovat a používat opakované layouty. Vývojáři mohou vytvářet různé layouty pro různé části své aplikace, což umožňuje konzistentní vzhled a chování napříč různými stránkami a sekce. Toto je zvláště užitečné pro větší projekty, kde je důležitá jednotnost a udržitelnost kódu. Složka app také usnadňuje sdílení a opětovné použití komponent. Vývojáři mohou centralizovat běžně používané komponenty a funkcionality, což zvyšuje efektivitu a snižuje redundanci v kódu. Kromě toho, integrace s různými stylovacími přístupy a knihovnami je nyní jednodušší a flexibilnější.

## Rozložení eshopu:

Hlavní rozdělení e-shopu je realizováno prostřednictvím dvou zásadních složek: „(admin)“ a „(eshop)“. Tato struktura poskytuje jasnou oddělenost mezi dvěma klíčovými funkcemi: CMS (Content Management System) pro správu obsahu e-shopu a samotným e-shopem, který je určen pro konečné uživatele. V kořenové struktuře projektu tedy naleznete tyto dvě složky, které efektivně organizují a rozlišují tyto dvě oblasti aplikace.

D:.





Kromě hlavního rozdělení na složky admin a **e-shop**, projekt obsahuje několik dalších klíčových top-level složek, které plní důležité funkce:

- **components**: Tato složka obsahuje všechny React komponenty používané v aplikaci. Je to centrální místo pro ukládání opakovaně použitelných UI prvků.
- **actions**: Složka určená pro fetchovací operace. Zde jsou definovány funkce pro načítání dat z backend REST endpointů, což umožňuje oddělit logiku dat od UI komponent.
- **api**: V této složce jsou definovány REST endpointy. Je to klíčové místo pro správu a konfiguraci všech backendových rozhraní, které aplikace používá.
- **hooks**: Tato složka slouží pro globální state management a využívá knihovnu Zustand. Zde jsou umístěny všechny custom hooks, které spravují sdílený stav napříč aplikací.
- **prisma**: Složka prisma obsahuje vše potřebné pro práci s databází – od migrací a schémat až po samotné připojení k databázi. Je to zásadní část projektu pro správu datové vrstvy.
- 

### ***Databázové schéma:***

Z důvodu velkých rozměrů, přiloženo zvlášť, k dokumentaci.

## **Zadání Semestrální práce: Vývoj E-shop Systému**

**Cíl Projektu:** Vývoj webového e-shop systému s následujícími uživatelskými rolemi a funkcionalitami.

### **Role Uživatelů:**

#### **1. Admin/SuperAdmin:**

- Správa uživatelů, včetně přiřazování rolí.

### **Dodavatel:**

- Vytváření a správa produktů v systému.

### **Přihlášený Uživatel:**

- Přidávání produktů do košíku a simulace objednávky (bez reálné transakce).

### **Nepřihlášený Uživatel:**

- Omezený přístup k zobrazení obsahu e-shopu.
- Možnost registrace pro získání role přihlášeného uživatele.

### **Zjednodušení:**

### **Platební Brána:**

- Neimplementovat platební bránu; objednávky budou pouze simulovány.

### **Databáze Produktů:**

- Použít jednu tabulku produktů pro zjednodušení struktury databáze.
- Produkty mohou být služby, fyzické produkty nebo kombinace, ale vše v jedné tabulce.

### Další Požadavky:

- Vývoj přehledného uživatelského rozhraní.
- Zajištění základní bezpečnosti systému.
- Dokumentace kódu a funkčnosti systému.

### Výstup:

- Funkční webová aplikace e-shopu.
- Dokumentace obsahující popis implementace a uživatelskou příručku.

### Závěr:

Musím říct, že jsem se hodně naučil a jsem velmi spokojen s přístupem učitelů. Oceňuji také povolení využívat framework, který mi nejlépe vyhovuje. Toto období bylo pro mě skutečně přínosné a inspirující.

- Vývoj přehledného uživatelského rozhraní.
- Zajištění základní bezpečnosti systému.
- Dokumentace kódu a funkčnosti systému.

#### Výstup:

- Funkční webová aplikace e-shopu.
- Dokumentace obsahující popis implementace a uživatelskou příručku.

## **Závěr:**

Musím říct, že jsem se hodně naučil a jsem velmi spokojen s přístupem učitelů. Oceňuji také povolení využívat framework, který mi nejlépe vyhovuje. Toto období bylo pro mě skutečně přínosné a inspirující.