UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Fakturační aplikace technická dokumentace frontend

Semestrální práce

Obsah

1. Úvod

Fakturka je webová aplikace určená pro správu faktur, která poskytuje jednoduché a intuitivní uživatelské rozhraní přizpůsobené potřebám firemního managementu. Aplikace umožňuje:

- **Vytvářet a upravovat faktury** podle předdefinovaných šablon.
- **Spravovat uživatelské účty**, včetně přihlášení, registrace a správy profilů.
- **Zobrazovat a organizovat faktury** prostřednictvím interaktivních tabulek.

Aplikace je postavená na moderních technologiích, které zajišťují vysoký výkon a flexibilitu:

- **Frontend framework**: React.js, umožňuje rychlou a efektivní tvorbu komponent a správu stavu.
- **Grafické prvky**: Použití knihovny Material-UI (MUI) zaručuje profesionální a responsivní design.
- **Komunikace s backendem**: REST API pro načítání, ukládání a manipulaci s daty.
- **Docker**: Snadné spuštění a nasazení aplikace pomocí kontejnerizace.

Hlavní vlastnosti

1. **Evidování faktur**:
- Možnost zobrazení všech faktur uživatele.
- Vyhledávání a filtrování podle klíčových hodnot (např. variabilní symbol).

2. **Správa šablon**:
- Uživatelsky přívětivý výběr šablon faktur.
- Editace šablon s automatickým doplňováním dat.

3. **Uživatelský účet**:
- Registrace a přihlášení uživatelů.
- Možnost správy profilu.

4. **Responsivní rozhraní**:
- Optimalizované pro práci na desktopu i mobilních zařízeních.

Architektura frontendové aplikace

- **React.js**: Jednostránková aplikace (SPA), která zajišťuje rychlé načítání a vysokou interaktivitu.
- **Redux**: Použití pro správu globálního stavu, jako je autentizace uživatele.
- **MUI**: Moderní knihovna pro stylování a tvorbu UI komponent.
- **RTK Query**: Pro komunikaci s API a správu dat na klientské straně.

Cílová skupina

Tato dokumentace je určena především pro vývojáře, kteří budou aplikaci udržovat, rozšiřovat nebo upravovat. Dokumentace poskytuje popis struktury projektu, použitých technologií, klíčových komponent a jejich funkcionality. Dále je určena pro DevOps inženýry, kteří se podílí na nasazení a správě prostředí pro běh aplikace, a testery, kteří potřebují pochopit hlavní aspekty funkčnosti a struktury aplikace pro efektivní testování. Cílem dokumentace je zajistit snadné porozumění kódu, rychlou orientaci v projektu a efektivní spolupráci mezi členy vývojového týmu.

2. Struktura projektu

Frontend aplikace je strukturován tak, aby byla zajištěna přehlednost a jednoduchost při vývoji i údržbě. Hlavní adresáře a jejich účel:

Kořenový adresář

- public/
  - Obsahuje statické soubory, které jsou přímo přístupné z prohlížeče (např. obrázky, favicon, základní HTML šablona index.html).
- src/
  - Obsahuje zdrojový kód aplikace. Toto je hlavní pracovní adresář pro vývojáře.

Adresáře v src/

- components/
  - Sdílené React komponenty, které se používají napříč aplikací. Například:
  - Header.js: Hlavička aplikace.
  - CustomTable.js: Tabulka pro zobrazení dat.
- functions/
  - Obsahuje utility a pomocné funkce, které jsou využívány v různých částech aplikace.
- layouts/
  - Obsahuje rozložení stránek aplikace (např. hlavní layout, který obaluje všechny stránky).
- pages/
  - Jednotlivé stránky aplikace odpovídající konkrétním URL. Například:
  - LoginPage.js: Stránka pro přihlášení uživatelů.
  - InvoiceEditor.js: Stránka pro úpravu faktur.
- services/
  - Obsahuje logiku pro volání API a správu dat na klientské straně. Například:
  - authService.js: Autentizační služby (přihlášení, registrace, odhlášení).
  - invoiceService.js: Operace spojené s fakturami.
- utils/redux/
  - Složka pro správu globálního stavu aplikace:
  - authSlice.js: Redux slice pro správu uživatelské autentizace.
  - store.js: Centralizovaný Redux store.

3. Architektura projektu

Frontend aplikace Fakturka je navržen tak, aby byla zajištěna modularita, snadná rozšiřitelnost a přehlednost. Aplikace využívá standardní architekturu Reactu, doplněnou o Redux pro správu stavu a MUI pro vizuální komponenty.

3.1. Hlavní architektonické prvky

React SPA (Single Page Application)

Aplikace funguje jako jednostránková aplikace, což znamená, že veškerá navigace mezi stránkami probíhá na klientské straně. Data jsou načítána prostřednictvím REST API.

Správa stavu pomocí Reduxu

Redux zajišťuje centralizovanou správu globálního stavu aplikace, například:

- Informace o přihlášeném uživateli.
- Data faktur.
- Stav autentizace.

Komunikace s API

Data jsou získávána pomocí Redux Toolkit Query (RTK Query), což umožňuje efektivní práci s daty (načítání, ukládání, aktualizace a mazání).

Stylování a uživatelské rozhraní

- **MUI**: Používá se pro stylování komponent a zajištění responsivního designu.
- **CSS-in-JS**: Stylování je integrováno přímo v komponentách.

Modularita

Aplikace je rozdělena na logické části:

- Komponenty (components/): Sdílené UI prvky.
- Stránky (pages/): Odpovídají jednotlivým URL aplikace.
- Služby (services/): API klienti a další pomocné třídy.

3.2. Styčné body mezi komponentami

Aby byla aplikace přehledná a snadno rozšiřitelná, jsou komponenty navrženy tak, aby mezi sebou komunikovaly pouze prostřednictvím jasně definovaných rozhraní (props, Redux, nebo callbacky).

Přehled styčných bodů

Header.js

- **Vstup**:
  - Redux stav (useSelector): Informace o přihlášeném uživateli.
- **Výstup**:
  - Navigace uživatele (useNavigate): Přesměrování na různé stránky (např. /dashboard).
  - Redux akce (clearUser): Odhlášení uživatele.

CustomTable.js

- **Vstup**:
  - columns (props): Definice sloupců tabulky.
  - data (props): Data pro zobrazení.
  - onRowClick (props): Callback pro akci při kliknutí na řádek.
- **Výstup**:
  - Callback onRowClick: Informuje rodičovskou komponentu o vybraném řádku.

TemplateRenderer.js

- **Vstup**:
  - htmlTemplate (props): Šablona pro vykreslení.
  - jsonData (props): Data pro naplnění šablony.
  - editable (props): Povolení editace šablony.
- **Výstup**:
  - Callback onFieldChange: Informuje rodičovskou komponentu o změnách v šabloně.
  - Callback onUpdateData: Aktualizace dat při přidávání nebo mazání položek.

TemplatesList.js

- **Vstup**:
  - jsonTemplate (z useLoadJsonTemplate): Data o šablonách.
- **Výstup**:
  - Navigace na detail šablony (TemplatesListItem přesměrovává na /template/:id).

UserInvoicesTable.js

- **Vstup**:
  - Faktury (useGetItemsQuery): Data faktur načítaná z API.
  - searchValue (lokální stav): Hodnota pro filtrování faktur.
- **Výstup**:
  - Callback handleRowClick: Přesměrování na detail faktury.
  - mazání faktury (useDeleteInvoiceMutation): Odstraňuje fakturu z databáze.

3.3. Tok dat

Zobrazení dat

- Data jsou načítána z backendového API pomocí RTK Query a ukládána do Reduxu.
- Komponenty získávají data z Redux store nebo přímo z API query.

Uživatelské interakce

- Kliknutí na řádky tabulky (CustomTable) nebo šablony (TemplatesListItem) spouští navigaci na detail.
- Změny v šablonách (TemplateRenderer) jsou zpětně odesílány do Reduxu nebo API.

Změna stavu

- Redux slouží jako styčný bod pro globální stav (autentizace, seznam faktur).
- Lokální stavy (např. searchValue v UserInvoicesTable) slouží pro specifické interakce.

4. Hlavní funkce aplikace

Tato sekce popisuje klíčové funkce aplikace Fakturka, jejich účel a způsoby implementace. Cílem je poskytnout vývojářům jasný přehled o tom, co aplikace nabízí a jak jsou jednotlivé funkce technicky zajištěny.

4.1. Autentizace uživatele

Popis

- Uživatelé se mohou registrovat, přihlašovat a odhlašovat.
- Autentizační údaje jsou zpracovávány pomocí backendového API.

Funkce

- **Registrace uživatele**:
  - Uživatel zadá jméno, e-mail a heslo, které jsou odeslány na API.
- **Přihlášení**:
  - Po přihlášení získá uživatel autentizační token, který je uložen v Redux stavu.
- **Odhlášení**:
  - Token je odstraněn z Reduxu a uživatel je přesměrován na přihlašovací stránku.

Implementace

- Použití Reduxu (authSlice.js) pro správu autentizačního stavu.
- Komunikace s API prostřednictvím authService.js.

4.2. Správa faktur

Popis

- Aplikace umožňuje vytvářet, upravovat, mazat a zobrazovat faktury.
- Data faktur jsou spravována na serveru a načítána prostřednictvím API.

Funkce

- **Zobrazení faktur**:
  - Přehled všech faktur uživatele v tabulkovém formátu.

- **Vyhledávání**:
  - Uživatel může hledat faktury podle variabilního symbolu.
- **Mazání faktur**:
  - Každá faktura obsahuje tlačítko pro odstranění.
- **Editace faktur**:
  - Detail faktury umožňuje uživateli měnit její obsah.

Implementace

- Použití komponent UserInvoicesTable a CustomTable pro zobrazení faktur.
- Redux Toolkit Query (useGetItemsQuery) pro načítání faktur.
- useDeleteInvoiceMutation pro mazání faktur.

4.3. Šablony faktur

Popis

- Šablony faktur poskytují základní strukturu, kterou lze snadno upravovat.
- Uživatelé si mohou vybrat z dostupných šablon nebo je upravovat dle potřeby.

Funkce

- **Zobrazení šablon**:
- Seznam všech dostupných šablon zobrazený v gridovém rozložení.
- **Výběr šablony**:
- Kliknutím na šablonu se uživatel přesměruje na stránku s podrobnostmi.
- **Editace šablon**:
- Dynamické úpravy obsahu šablony přímo v aplikaci.

Implementace

- Komponenty TemplatesList a TemplatesListItem zajišťují zobrazení šablon.
- TemplateRenderer umožňuje interaktivní úpravy obsahu šablony.

4.4. Dynamická interaktivita

Popis

- Aplikace je navržena tak, aby podporovala dynamické interakce bez nutnosti obnovování stránky.
- Příkladem je stránkování, hledání nebo kliknutí na řádky tabulky.

Funkce

- **Klikání na řádky tabulky**:
  - Při kliknutí na řádek v tabulce je uživatel přesměrován na detail faktury.
- **Vyhledávání**:
  - Interaktivní vyhledávání v tabulce faktur podle klíčových slov.
- **Stránkování šablon**:
  - Dynamické stránkování položek na šabloně faktury.

Implementace

- Použití MUI komponent (Table, TextField, Grid) pro UI.
- Callbacky (onRowClick, onFieldChange) pro interakce mezi komponentami.

4.5. Responsivní design

Popis

- Aplikace je optimalizována pro zobrazení na různých zařízeních, včetně mobilních telefonů a tabletů.
- Rozvržení se automaticky přizpůsobuje velikosti obrazovky.

Funkce

- **Přizpůsobení UI**:
- Využití MUI Grid a Stack komponent pro flexibilní rozvržení.
- **Responsivní prvky**:
- Tabulky, formuláře a tlačítka jsou optimalizovány pro dotykové ovládání.

Implementace

- MUI komponenty pro grid layout (Grid2, Stack).
- CSS-in-JS stylování pro dynamické změny vzhledu.

5. Správa stavu aplikace

Správa stavu je jednou z klíčových částí frontendové aplikace Fakturka. Použití **Redux Toolkit** zajišťuje efektivní správu globálního stavu, který zahrnuje autentizaci, data faktur a další informace potřebné napříč aplikací.

5.1. Hlavní koncepty Reduxu v aplikaci

Redux Store

- Centrální úložiště, které drží veškerý stav aplikace.
- Kombinuje jednotlivé „slices“ pro různé části stavu.

Slices

- **authSlice.js**: Správa uživatelské autentizace.
- **invoiceSlice.js**: Správa dat faktur.

RTK Query

- Používá se pro asynchronní volání API a ukládání výsledků do Reduxu.
- Zajišťuje automatickou správu stavu (např. načítání, úspěch, chyba).

Dispatch a Selectory

- **Dispatch**: Posílá akce, které aktualizují stav.
- **Selector**: Čte data z Redux store.

5.2. Architektura Reduxu

Store (store.js**)**

- Centralizované úložiště.
- Kombinuje všechny slices.

Slices:

- **authSlice.js**:
  - Obsahuje informace o uživateli (např. uživatelské jméno, token).

- Akce:
  - setUser(user): Nastavuje aktuálního uživatele.
  - clearUser(): Maže data uživatele při odhlášení.
- **invoiceSlice.js**:
  - Obsahuje seznam faktur a stav jejich načítání.
  - Akce:
    - addInvoice(invoice): Přidá novou fakturu.
    - updateInvoice(invoice): Aktualizuje existující fakturu.

RTK Query (invoicesApi.js**):**

- Definuje endpointy pro komunikaci s API.
- Endpointy:
  - getItems: Načítá faktury.
  - deleteInvoice: Maže fakturu.

5.3. Jak Redux interaguje s aplikací

- Autentizace uživatele
  - Když se uživatel přihlásí, akce setUser uloží uživatelské informace do Reduxu.
  - Při odhlášení akce clearUser odstraní uživatelská data.
- Načítání faktur
  - Komponenta UserInvoicesTable volá endpoint getItems z RTK Query.
  - Načtená data se automaticky ukládají do Reduxu.
- Interaktivní akce
  - Při kliknutí na tlačítko mazání faktury:
  - Volá se endpoint deleteInvoice.
  - Redux aktualizuje seznam faktur po úspěšném smazání.
- Selektory pro čtení stavu
  - useSelector je použitý v komponentách jako Header nebo UserInvoicesTable pro získání stavu uživatele nebo seznamu faktur.

5.4. Výhody implementace Reduxu

- **Centralizovaná správa stavu**:
  - Stav je spravován na jednom místě, což usnadňuje jeho údržbu a sdílení mezi komponentami.
- **Jednoduchá práce s asynchronními požadavky**:
  - RTK Query automaticky spravuje stavy načítání a ukládání dat.
- **Modularita**:

- Každý slice spravuje pouze určitou část stavu, což zlepšuje čitelnost a testovatelnost.