DOKUMENTACJA TECHNICZNA APLIKACJI WEBOWEJ DOCSEE

Spis treści

Wojciech Charemski
Programowanie Aplikacji Webowych na Akademii Górniczo-Hutniczej AGH w Krakowie.

Kraków 2023 r.

1. WPROWADZENIE

1.1 Cel Dokumentacji

Celem niniejszej dokumentacji jest dostarczenie kompleksowej informacji dotyczącej struktury, funkcji, i zasady działania aplikacji webowej Docsee. (Reszta dokumentacji zawiera szczegółowe opisy poszczególnych sekcji i komponentów aplikacji.)

1.2 Cel Aplikacji

Aplikacja Docsee została stworzona, jako projekt końcowy studiów podyplomowych o kierunku Programowanie aplikacji webowych na Akademii Górniczo Hutniczej AGH w Krakowie.
Niektóre funkcje aplikacji są w wersji demo, czyli nie działają tak jak w normalnej aplikacji webowej stworzonej dla większego odbioru.

Docsee to aplikacja internetowa zaprojektowana w celu usprawnienia przechowywania i organizacji kluczowych dokumentów osobistych. Dzięki Docsee użytkownicy mogą bezpiecznie przesyłać i przechowywać różne ważne dokumenty, w tym skany dowodów osobistych, dyplomów, certyfikatów, akt zatrudnienia, życiorysów i innych. Platforma kładzie nacisk na przyjazną dla użytkownika nawigację i oferuje intuicyjne funkcje kategoryzacji, dzięki czemu użytkownicy mogą łatwo zarządzać niezbędną dokumentacją i uzyskiwać do niej dostęp w jednym scentralizowanym miejscu. Docsee ma na celu uproszczenie zarządzania dokumentami i zwiększenie dostępności, zapewniając użytkownikom wygodny i bezpieczny sposób śledzenia ich najważniejszych zapisów.

1.3 Zakres Dokumentacji

Dokumentacja techniczna ma na celu dostarczenie kompleksowej wiedzy na temat aplikacji, a jej zakres obejmuje następujące obszary:

1.3.1 Opis funkcji aplikacji

W tej sekcji opisane są główne funkcje i możliwości aplikacji. Każda funkcja jest szczegółowo przedstawiona, wskazując, jakie problemy rozwiązuje dla użytkownika końcowego i jakie korzyści przynosi.

1.3.2 Instrukcja instalacji

Szczegółowo opisuje kroki niezbędne do poprawnej instalacji i konfiguracji aplikacji na środowisku lokalnym.

1.3.3 Struktura katalogów

Opisuje strukturę katalogów projektu, pomagając programistom i administratorom w zlokalizowaniu konkretnych elementów aplikacji. Jest to szczególnie istotne dla zespołów pracujących nad projektem.

1.3.4 Frontend

Opisuje strukturę i organizację warstwy frontendowej, w tym wykorzystane biblioteki, frameworki, zarządzanie stanem, komunikację z backendem, komponenty i inne istotne elementy interfejsu użytkownika.

1.3.5 Backend

W tej sekcji znajdują się informacje na temat architektury backendu, wykorzystanych technologii, obsługi bazy danych, autentykacji i innych kluczowych elementów związanych z logiką działania

1.3.6 Bibliografia

Zbiór wszystkich źródeł pomocnych w tworzeniu aplikacji Docsee

2. ARCHITEKTURA SYSTEMU

1.2 Działanie Aplikacji Docsee

Aplikacja została napisana w języku JavaScript, wykorzystując framework React. Treść całej aplikacji jest napisana w języku angielskim. Strona główna Docsee informuje użytkownika w 2-3 zdaniach o tym, do czego ona pokrótce służy, z przyciskiem przekierowującym do głównej części aplikacji,(jeśli użytkownik jest zalogowany, jeżeli nie jest zostanie przekserowany do formularza z logowaniem). Znajduje się tutaj również nawigacja do kart logo Docsee, '**PRICING**','**ABOUT US**','**LOG IN**'. Logo Docsee przenosi nas do strony startowej aplikacji.

Karta 'PRICING' przedstawia 3 oferty planu działania aplikacji (wybór planu ma charakter demonstracyjny, gdyż przyciski tych ofert przekierowują na stronę główną).

Karta 'ABOUT US' zawiera rozszerzony opis działania i funkcjonowania aplikacji Docsee. Opis ten dotyczy tego, jak miałaby działać i zachowywać się aplikacja wydana docelowo do powszechnego działania.

Karta 'LOG IN' przenosi użytkownika bezpośrednio do formularza logowania. Jeżeli użytkownik nie posiada konta w aplikacji, może stworzyć je za pomocą formularza rejestracji. Gdy użytkownik się zaloguję link ‘LOG IN’ zniknie.

Jeżeli logowanie/rejestracja przebiegły pomyślnie użytkownik dostaje dostęp do głównej części aplikacji. W wersji desktopowej są dwa panele lewy do nawigacji i prawy do wyświetlania dokumentów oraz plików w danych kategoriach. W wersji mobilnej panel z nawigacją znajduję się u góry ekranu, natomiast panel główny pod nim.

Panel nawigacji składa się z przycisków ‘**DOCS**’ oraz ‘**CATEGORIES**’. Dzięki zastosowaniu odpowiedniej nawigacji z pomocą biblioteki react-router, przycisk DOCS jest domyślnie aktywny. Powoduję to,że poniżej zostają wyświetlone nazwy dokumentów przesłanych przez użytkownika na serwer. Jeżeli użytkownik nie posiada żadnych plików na serwerze wyświetlany jest napis zachęcający do dodania pierwszego dokumentu. Każdy dokument po dodaniu można usunąć. Pod tą listą jest przycisk otwierający formularz dodawania nowego dokumentu. Pozwala on na wybranie dowolnego pliku w formacie .jpg, .jpeg, .png, .pdf. Zastosowałem tu walidację formularza, przez co nie ma możliwości wysłania formularza bez uprzedniego dodania pliku oraz wyboru odpowiadającej kategorii, którą można wybrać z listy, (jeżeli jakaś z kategorii istnieje) lub stworzyć samemu. Po

dodaniu nowego dokumentu zostanie wyświetlona jego nazwa w powyższej liście. Każdy element listy jest klikany i przekierowywuje użytkownika do nowej karty, jednak nie przeładowywuje strony, co jest zasługą odpowiednich bibliotek reacta. Gdy element z listy zostanie kliknięty, do adresu URL zostanie dopisane id danego dokumentu oraz jego nazwa np.: „**app/docs/84?name=38493f78-50d3-42ef-81cd-eaac757b4aee-cv.pdf**". Dzięki temu możliwe jest namierzenie id danego dokumentu poprzez odczytanie adresu URL i wyświetlenie go na głównym panelu. Pozwala to na swobodne przeglądanie danych dokumentów oraz ich pobieranie z serwera. Na panelu nawigacji natomiast zamiast wcześniejszej listy, wyświetlane są teraz główne informacje o danym dokumencie, takich jak jego nazwa, kategoria, typ, rozmiar oraz data ostatniej modyfikacji. Pod listą jest przycisk '**BACK**', który cofa nas do wcześniejszego miejsca.

Przycisk '**CATEGORIES'** wyświetla wszystkie kategorie, które zostały przez nasz utworzone, w prostokątnych bloczkach wraz z informacją o ilości dokumentów należących do danych kategorii. Również jest to element klikany przenoszący nas do nowej karty (nowego URL i bez przeładowywania strony).Każdą kategorię można usunąć tylko wtedy, gdy liczba dokumentów, które się w niej znajdują będzie równa 0, w przeciwnym razie przycisk do usuwania kategorii nie będzie widoczny. Po kliknięciu w dany bloczek kategorii uruchamiana jest nowa lista w panelu nawigacji zawierająca nazwę kategorii, a poniżej wszystkie nazwy dokumentów, które ona zawiera. Na panelu głównym natomiast zostają wyświetlone bloki z nazwą dokumentu, datą utworzenia, ostatnią modyfikacją oraz linkiem do dokumentu, który zostanie wyświetlony w nowej karcie.

Dodatkowo w części głównej aplikacji znajduję się stopka z informacją o prawach autorskich. W górnym prawym rogu jest wyświetlony e-mail obecnie zalogowanego użytkownika oraz przycisk służący do wylogowania się z aplikacji.

3. INSTRUKCJA INSTALACJI

3.1 Wymagania Systemowe

Do uruchomienia aplikacji lokalnie zaleca się korzystanie z programu Visual Studio Codę. Dodatkowo system musi spełniać następujące wymagania:

- Node.js (wersja >= v18.13.0)
- Npm (wersja >= 8.19.3)

Wymagane jest zainstalowanie bądź zaktualizowanie bieżących programów w celu uruchomienia aplikacji w środowisku VS Code.

3.2 Instrukcje Instalacji

Aplikacje w wersji LIVE można bezpośrednio zobaczyć pod adresem https://docsee.netlify.app. Dzięki serwisowi Netlify.com aplikacja została umieszczona na serwerze.
W celu uruchomienia aplikacji lokalnie na komputerze należy postąpić następująco:

a) Uruchom program VS Code, następnie otwórz folder główny z aplikacją Docsee lub sklonuj repozytorium: `git clone https://github.com/Rockwood98/docsee
b) Proszę włączyć terminal, a następnie przejść do katalogu projektu poleceniem: 'cd nazwa_katalogu' (np. 'cd docsee')
c) Zainstaluj zależności 'npm install' lub' npm i'
d) Uruchom aplikację poleceniem 'npm run dev'

4. STRUKTURA KATALOGÓW

4.1 Opis Struktury Katalogów Projektu

Struktura katalogów projektu została zorganizowana zgodnie z wytycznymi przedstawionymi w warunkach zaliczenia na kierunku Programowanie Aplikacji Webowych na AGH.

Główny katalog to ‘docsee/’. W nim znajduję się ‘/src’ zawierający wszystkie źródła projektu. Katalog ‘/auth’ zawiera pliki służące do zarządzania i konfigurowania kontem użytkownika, oraz zapewniania odpowiedniej autoryzacji.

Katalog ‘/components’ zawiera dosłownie wszystkie komponenty znajdujące się w aplikacji, przykładowo komponent PageNav.jsx oraz PageNav.module.css (plik z CSS) odpowiedzialny za nawigacje w aplikacji, Button.jsx + Button.module.css tworzący wszystkie przyciski znajdujące się na stronie.

Folder ‘/pages’ przechowuje kod odpowiedzialny za budowę wszystkich podstron w aplikacji, przykładowo Homepage.jsx/Homepage.module.css jest to kod strony głównej, witającej w Docsee, lub Login.jsx/Login.module.css odpowiadający stronie z formularzem logowania.

Katalog ‘/services’obejmuję dokumenty w formacie JavaScrpit, które odpowiadają za wysyłanie odpowiednich zapytań/komend do bazy danych w serwisie Supabase.com

Folder ‘/imgs’ zawiera wszystkie obrazy w formacie .png i .jpg znajdujące się na stronie, natomiast katalog ‘/assets’ zachowuje w sobie domyślne SVG dostarczone przez sam react.

Poza katalogami w folderze ‘/src’ są pliki takie jak App.js, jest to kod budujący całą aplikację w jedną całość. Index.css składa się z głównych styli odpowiadających całemu programowi oraz main.jsx, który dotyczy importowania biblioteki react.

Poza tym w katalogu głównym znajdują się pliki konfiguracyjne wymagane do uruchomienia aplikacji tj. .eslintrc- do konfiguracji narzędzia ESLint służącego do statystycznego analizowania kodu źródłowego w języku JavaScrpit, .gitignore dla serwisu Github, index.html, package.json/package-lock.json do odpowiedniego skonfigurowania środowiska w czasie instalacji, vite.confg- konfiguracja Vite, README.md oraz katalog ‘/docs’ z dokumentacją techniczną.

5. FRONTEND

5.1 Struktura Aplikacji

Docsee zbudowana jest za pomocą biblioteki react. Wszystkie elementy tej aplikacji podzielone są na oddzielne pliki tzw. moduły. W głównym pliku App.jsx importowane są wszystkie strony budujące aplikacje i składane w całość, te strony pobierają natomiast wszystkie napisane wcześniej oddzielne komponenty, które je budują. Kod HTML jest tak naprawdę kodem JavaScript pisanym w języku JSX, jest to składnia, która umożliwia pisanie struktur UI w sposób podobny do HTML, jednak jest to nadal JavaScript. JSX jest bardziej deklaratywny i zwięzły niż czysty JavaScript, co sprawia ,że kod jest łatwiejszy do zrozumienia, umożliwia również korzystanie z wyrażeń JavaScript bezpośrednio wewnątrz struktury JSX, co ułatwia dynamiczne renderowanie treści. Jedną z dodatkowych funkcji jest obsługa zdarzeń, umożliwiająca bardziej rozbudowane interakcje w interfejsie użytkownika. Dzięki tworzeniu komponentów ‘wielorazowego użytku’ możemy umieścić je na stronie w

wielu miejscach, np. w przypadku komponentu Button, wystarczy w danym miejscu w aplikacji zaimportować ten komponent następnie określić jego typ wyznaczony poprzez style CSS i nie tworzymy nowych zbędnych linijek kodu.

Jeżeli chodzi o część wizualną aplikacji jest ona pisana w ‘czystym’ CSS bez użycia zewnętrznych bibliotek typu Bootstrap czy Tailwind. W projekcie style zostały stworzone poprzez CSS Modules, jest to technika modularyzacji stylów w projektach internetowych. Pomaga ona unikać konfliktów nazw klas, zwiększa izolację komponentów oraz ułatwia zarządzanie stylami w dużych projektach. CSS Modules wprowadzają lokalny zasięg dla klas. Każdy moduł stylów jest traktowany jako osobny zakres, co oznacza ,że nazwy klas są unikalne w obrębie danego modułu. Rozwiązanie to eliminuje globalne zasięgi, co jest szczególnie istotne w dużych projektach, gdzie konflikty nazw klas mogą prowadzić do nieprzewidywalnych stylizacyjnych skutków ubocznych. Stylizacja jest importowana bezpośrednio do plików JavaScript/JSX, co ułatwia przypisanie klas do elementów w kodzie komponentu. Importowanie odbywa się przy użyciu specjalnej składni przykładowo: ‘import styles from ‘./nazwa_komponentu.module.css’. Dużym plusem jest możliwość korzystania z dynamicznych klas w zależności od kontekstu.

5.2 Wykorzystane Biblioteki/Frameworki

Frontend aplikacji webowej korzysta z biblioteki react + Vite. React to biblioteka stworzona przez Facebooka, służąca do budowy interfejsów użytkownika. Oparty jest na koncepcji komponentów, co oznacza ,że interfejs użytkownika jest rozbijany na mniejsze, samodzielne elementy zwane komponentami. Każdy komponent może posiadać swój własny stan (state) oraz metody, co zapewnia reużywalność i skalowalność kodu. React używa wirtualnego DOM-u, co pozwala na minimalizację liczby manipulacji w jego rzeczywistą strukturę, co przekłada się na lepszą wydajność. Dane przekazywane są w jednym kierunku, ułatwia to debugowanie i zarządzanie stanem.

Vite natomiast dostarcza środowisko do budowy aplikacji opartych na JavaScript, TypeScript, React oraz Vue. Zapewnia szybkość budowy, wbudowany serwer deweloperski, brak zbędnych konfiguracji np. ESLint oraz lepszą optymalizację.

Oba te narzędzia możemy łączyć w celu budowy nowoczesnych i wydajnych aplikacji internetowych. React dostarcza potężną bibliotekę do budowy interfejsów użytkownika, a Vite oferuje środowisko do szybkiego rozwoju projektów JavaScript.

5.2.1 React-Query

React Query to biblioteka stworzona do zarządzania stanem oraz obsługi żądań http w aplikacjach React. Głównym jej celem jest ułatwianie pracy z danymi asynchronicznymi, takimi jak pobieranie, zapisywanie i aktualizowanie danych, zarówno lokalnie, jak i poprzez żądania sieciowe. Dzięki tej bibliotece aplikacja zyskała bardzo szybki dostęp do danych pomiędzy różnymi komponentami w zoptymalizowany sposób, nie powodując przeciążeń. Przykładem jest wyświetlanie listy dokumentów w panelu nawigacji oraz dokumentów PDF w panelu głównym. React Query eliminuje konieczność ręcznego zarządzania stanem oraz dostarcza narzędzia do efektywnego obsługiwania danych operacji.

Dodatkowo w projekcie została pobrana uzupełniająca biblioteka dla React Query, tj. React Query Devtools. Narzędzie to dostarcza dodatkowe funkcje do debugowania i monitorowania stanu oraz działania biblioteki React Query w trakcie tworzenia aplikacji React. Stanowi ona przejrzysty interfejs dla użytkownika ułatwiający efektywne zarządzanie danymi asynchronicznymi.

5.2.2 React-Router-Dom

React Router DOM to biblioteka dla React, która zapewnia nawigację między widokami (komponentami/stronami) w aplikacji. Umożliwia tworzenie dynamicznych jednostronicowych aplikacji (Single Page Applications SPA), gdzie zmiana URL-a nie powoduje przeładowywania całej strony, a jedynie renderowanie nowego komponentu na istniejącej stronie. Docsee również korzysta z tej biblioteki, dzięki umieszczeniu kodu z głównego pliku App.jsx pomiędzy znacznik <BrowserRouter> a następnie przypisaniu odpowiednich <Route>, aplikacja staje się SPA, bez zbędnych przeładowań strony do nowych kart i komponentów

5.2.3 React-Hook-Form

React Hook Form to kolejna biblioteka dla Reacta, umożliwia łatwe zarządzanie formularzami w aplikacjach React przy użyciu hooków. Głównym celem tej biblioteki jest uproszczenie i usprawnienie zarządzania stanem formularzy, dostarczając intuicyjny interfejs programistyczny. W aplikacji Docsee React Hook Form został zaimplementowany w formularzach na stronie, logowania/rejestracji oraz dodawania nowego dokumentu na serwer.

5.2.4 React-Icons

React Icons jest biblioteką dla projektów opartych na React. Umożliwia łatwe dodawanie ikon do aplikacji, oferując dostęp do popularnych zestawów ikon dostępnych w sieci takich jak Font Awesome, Material Icons czy Hero Icons. Dzięki prostemu systemowi importu bezpośrednio z poziomu kodu jesteśmy w stanie szybko dodawać i zmieniać interesujące nas ikony. Niektóre parametry ikon takie jak kolor czy wielkość możemy dowolnie modyfikować poprzez odpowiednie stylowanie.

5.2.5 React-Hot-Toast

Kolejna biblioteka, która tym razem służy do tworzenia oraz wyświetlania na stronie powiadomień (toastów). Są to ładnie stylizowane już komunikaty o sukcesie, błędzie czy ostrzeżeniu do interfejsu użytkownika, poprzez responsywne i estetyczne komponenty toastów.

5.2.6 React-Spinner

Biblioteka ta dostarcza gotowe komponenty do renderowania animowanych spinnerów(obrotowe, zmienne wskaźniki) w aplikacji. Wskaźniki te służą w aplikacji do sygnalizowania ładowania danych lub oczekiwania na operacje asynchroniczne.

5.2.7 UUID

Za pomocą tej biblioteki możemy generować identyfikatory składające się z 32 heksadecymalnych cyfr, podzielonych myślnikami na pięć grup w formie 8-4-4-4-12. Zastosowanie w aplikacji Docsee tej biblioteki pomogło w tworzeniu unikatowych nazw przesyłanych plików na serwer, co zniwelowało problem posiadania dwóch o identycznej nazwie.

6 BACKEND

6.1 Opis Platformy

Supabase to platforma służąca do budowana aplikacji opartych na bazie danych. Opiera się na PostgreSQL, jednej z najbardziej zaawansowanych baz danych relacyjnych. Oferuje kompleksowe narzędzia do budowy nowoczesnych aplikacji bez konieczności zarządzania wieloma aspektami backendu. Dzięki funkcji realtime, elastycznemu API GraphQL i wbudowanym usługom serverless, Supabase pozwala na efektywne tworzenie zaawansowanych aplikacji internetowych.

6.2 API

W katalogu '/services' znajdują się pliki zawierające kod API do pobierania odpowiednich elementów z bazy danych. Dzięki odpowiednim regułom API, możemy w aplikacji Docsee wysyłać na serwer nasze dokumenty a następnie pobierać te dane z powrotem wraz z informacjami dodatkowymi o danym pliku. Dodatkowo dzięki ustawieniom odpowiednich opcji zabezpieczeń i autoryzacji każdy plik oraz utworzoną przez nas kategorię możemy usunąć.

6.3 Baza Danych

Struktura bazy danych składa się z tabeli 'docs' i tabeli 'category'. Pierwsza tabela zawiera następujące wiersze: 'id',' created_at' (te dwa generowane automatycznie),'name', 'type', 'size', 'category', 'path' oraz 'lastModified'. 'Name', 'type', 'size' oraz 'lastModified' pobierane są z danych pliku przesyłanego na serwer. Kolumna 'category' połączona jest kluczem głównym z id tabeli 'category', zawierającej kolumne z nazwą danej kategorii, którą możemy stworzyć bezpośrednio w aplikacji. Połączenie kluczem zapewnia, że użytkownik może wybrać odpowiednią kategorię z już istniejących lub stworzyć nową i przypisać do niej dany plik. Kolumna 'path' zawiera ścieżkę do odpowiedniego pliku przesłanego na serwer znajdującego się w magazynie, dzięki zastosowaniu biblioteki UUID każdy plik ma unikatową nazwę. Po dodaniu pliku na serwer zostaje on przeniesiony do magazynu się na nim znajdującym i nadana zostaje mu nowa nazwa jest to 'identyfikator UUID+nazwa_pliku'. Wszystkie dane są odczytywane z pliku i umieszczane w tabeli 'docs'. W przypadku utworzenia nowej kategorii pojawi się ona w tabeli 'category', a jej id zostanie połączone z wierszem 'category' z tabeli 'docs'

Dzięki zastosowaniu odpowiednich zasad autoryzacji dostęp do bazy danych mają tylko zalogowani użytkownicy. Nawet jeżeli ktoś dokona zmiany w kodzie i będzie chciał wyświetlić komponent dostępny z danymi z serwera poza głównym obszarem aplikacji, dane nie zostaną wyświetlone.

Użytkownik z poziomu aplikacji może stworzyć nowe konto lub zalogować się na już istniejące. Do stworzenia nowego konta wymagane jest podanie adresu e-mail oraz hasła. Weryfikacja poprzez adres mailowy jest wyłączona dlatego adres e-mail niekoniecznie musi być prawdziwy, ma to na celu przyśpieszenie korzystania z tej demonstracyjnej wersji aplikacji. Dane użytkownika po poprawnym utworzeniu konta zostają przesłane na serwer gdzie możemy zarządzać danymi kontami.

7 Bibliografia

React.js: [Dokumentacja React](https://reactjs.org/docs/getting-started.html)

- React-hook-form: [GitHub - React Hook Form](https://github.com/react-hook-form/react-hook-form)
- React-hot-toast: [GitHub - React Hot Toast](https://github.com/timolins/react-hot-toast)
- React-icons: [GitHub - React Icons](https://github.com/react-icons/react-icons)
- React-router-dom: [Dokumentacja React Router](https://reactrouter.com/web/guides/quick-start)
- React-spinner: [GitHub - React Spinner](https://github.com/davidhu2000/react-spinners)
- Uuid: [GitHub - UUID](https://github.com/uuidjs/uuid)
- React-query: [Dokumentacja React Query](https://react-query.tanstack.com/)
- Supabase: [Dokumentacja Supabase](https://supabase.io/docs)
- CSS Modules - Dokumentacja: [GitHub - CSS Modules](https://github.com/css-modules/css-modules)
- React Docs - Obsługa CSS: [Create React App - Adding a CSS Modules Stylesheet](https://create-react-app.dev/docs/adding-a-css-modules-stylesheet/)
- JavaScript MDN Web Docs:** [JavaScript MDN](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide)
- [@vitejs/plugin-react](https://github.com/vitejs/vite-plugin-react/blob/main/packages/plugin-react/README.md)
- Baza danych (https://supabase.com/)
- Grafiki w aplikacji (https://pixabay.com/pl/)