Dokumentacja Projektu - Serwis Komputerowy

Skład Grupy:

- Mateusz Mogielnicki
- Jakub Matyszek
- Przemysław Rutkowski

1. Architektura komponentów, przekazywane właściwości i metody

Komponent: LoginForm

- **Funkcje:**
    - LoginForm jest komponentem React służącym do logowania użytkowników.
    - Używa hooków useState do zarządzania stanem lokalnym dla emaila, hasła i ewentualnych błędów.
    - Wykorzystuje funkcję login z AuthService do autentykacji użytkownika.
    - Przekierowuje użytkownika na stronę raportów po pomyślnym zalogowaniu.

- **Stan:**
    - **email**: przechowuje adres email wprowadzony przez użytkownika.
    - **password**: przechowuje hasło wprowadzone przez użytkownika.
    - **errors**: przechowuje błędy związane z logowaniem.

- **Metody:**
    - handleSubmit: obsługuje zdarzenie submit formularza. Wykonuje logowanie i obsługuje błędy.

- **Nawigacja:**
    - Używa useNavigate z react-router-dom do nawigacji po pomyślnym logowaniu.

- **Elementy UI:**
    - Formularz logowania z polami email i hasło oraz przyciskiem do logowania.
    - Link do strony rejestracji.

Komponent: AddReport

- **Funkcje:**
  - AddReport to komponent React do tworzenia nowych raportów.
  - Używa hooka useState do zarządzania stanem formularza raportu, który obejmuje opis i priorytet.
  - Wykorzystuje funkcję createReport z ReportService do dodawania nowego raportu.

- **Stan:**
  - report: obiekt przechowujący dane raportu, w tym opis i priorytet.
  - errors: obiekt do śledzenia błędów walidacji formularza.

- **Metody:**
  - validateForm: funkcja do walidacji danych formularza przed ich przesłaniem.
  - handleInputChange: obsługuje zmiany w polach formularza i aktualizuje stan report.
  - handleAddReport: obsługuje proces dodawania raportu, w tym walidację, tworzenie raportu i nawigację po pomyślnym dodaniu.

- **Nawigacja:**
  - Używa useNavigate z react-router-dom do nawigacji po pomyślnym dodaniu raportu.

- **Elementy UI:**
  - Formularz do dodawania raportu z polami tekstowymi dla opisu i selektorem priorytetu.
  - Przyciski do dodawania raportu i anulowania.

Komponent: EditReport

- **Funkcje:**
  - EditReport służy do edycji istniejących raportów.
  - Używa hooków useState i useEffect do zarządzania stanem i pobierania danych raportu.
  - Wykorzystuje useParams do pobrania ID raportu z URL.

- **Stan:**
  - report: stan przechowujący aktualne dane raportu, takie jak cena i status.
  - currentReport: aktualnie edytowany raport.
  - errors: stan do śledzenia błędów walidacji formularza.

- **Metody:**
    - handleInputChange: aktualizuje stan raportu podczas zmian w formularzu.
    - validateForm: sprawdza poprawność danych formularza przed ich zapisaniem.
    - handleUpdateReport: obsługuje proces aktualizacji raportu.

- **Nawigacja:**
    - Używa useNavigate do nawigacji po zakończeniu edycji.

- **Elementy UI:**
    - Formularz edycji raportu z polami dla ceny i statusu.
    - Przyciski do zapisu zmian i anulowania edycji.

Komponent: LogsTable

- **Funkcje:**
    - LogsTable wyświetla tablicę logów i obsługuje ich usuwanie.
    - Używa funkcji deleteLog z LogService do usuwania logów.

- **Stan:**
    - logs: Tablica obiektów Log, które mają być wyświetlone.
    - isServiceman: Flag, która określa, czy użytkownik to serwisant, aby kontrolować widoczność przycisku usuwania.

- **Metody:**
    - handleDelete: Obsługuje usuwanie logu. Wywołuje deleteLog i następnie onDelete do aktualizacji widoku.

- **Elementy UI:**
    - Tabela z kolumnami dla ID, ID raportu, statusu, ceny i daty logu.
    - Przycisk "Delete" dla każdego logu, widoczny tylko dla serwisantów.

Komponent: RegisterForm

- **Funkcje:**
    - RegisterForm służy do rejestracji nowych użytkowników.
    - Wykorzystuje hook useState do zarządzania stanem formularza, w tym danych takich jak imię, nazwisko, email, hasło i numer telefonu.
    - Używa funkcji register z AuthService do rejestracji nowego użytkownika.

- **Stan:**
    - firstName, lastName, email, password, phoneNumber: Stany przechowujące dane wprowadzane przez użytkownika.
    - errors: Obiekt do śledzenia błędów walidacji formularza.

- **Metody:**
    - validateForm: Waliduje dane formularza przed wysłaniem.
    - handleSubmit: Obsługuje proces rejestracji, w tym walidację i wywołanie register.

- **Elementy UI:**
    - Formularz rejestracji z polami dla danych osobowych i przyciskiem rejestracji.

Komponent: ReportsTable

- **Funkcje:**
    - ReportsTable służy do wyświetlania tabeli raportów. Jest to komponent prezentacyjny.
    - Obsługuje logikę do wyświetlania szczegółów raportu oraz opcji edycji i przyjmowania raportu zależnie od roli użytkownika.

- **Stan:**
    - reports: Tablica obiektów Report, które mają być wyświetlone.
    - isServiceman: Wartość boolean określająca, czy użytkownik to serwisant.
    - servicemanId: ID serwisanta, służy do identyfikacji, czy raport należy do danego serwisanta.

- **Elementy UI:**
    - Tabela z kolumnami dla ID, opisu, priorytetu, statusu, ceny, dat rozpoczęcia i zakończenia oraz ID użytkownika.
    - Przyciski "Edit" i "Take report" w zależności od roli użytkownika i stanu raportu.

Komponent: ShowLogs

- **Funkcje:**
    - ShowLogs jest odpowiedzialny za wyświetlanie listy logów systemowych lub użytkownika.
    - Wykorzystuje hook useEffect do pobierania logów przy inicjalizacji komponentu.
    - Używa serwisu LogService do pobierania danych logów

- **Stan:**
    - logs: Stan przechowujący pobrane logi.
    - isServiceman: Określa, czy zalogowany użytkownik jest serwisantem.

- **Metody:**
    - fetchLogs: Metoda asynchroniczna do pobierania logów.
    - Przy każdym usuwaniu logu, lista jest odświeżana, co powoduje ponowne wywołanie fetchLogs.

- **Elementy UI:**
    - Wykorzystuje komponent LogsTable do wyświetlania logów.
    - Zapewnia przyciski nawigacyjne do powrotu do raportów i wylogowania.

Komponent: ShowReports

- **Funkcje:**
    - ShowReports odpowiada za wyświetlanie listy raportów.
    - Używa hooków useState i useEffect do zarządzania stanem raportów oraz filtrów.
    - Pobiera dane raportów za pomocą getReports z ReportService.

- **Stan:**
    - reports: Tablica raportów do wyświetlenia.
    - filters: Obiekt z filtrami stosowanymi do listy raportów.
    - isServiceman: Określa, czy zalogowany użytkownik jest serwisantem.
    - servicemanId: ID zalogowanego serwisanta.

- **Metody:**
    - fetchReports: Asynchroniczna metoda do pobierania i filtrowania raportów.
    - handleTakeReportClick: Obsługa akcji przyjęcia raportu.
    - handleFilterChange: Aktualizacja stanu filtrów.

- **Elementy UI:**
    - Wykorzystuje FilterReports do filtracji raportów.
    - Wykorzystuje ReportsTable do wyświetlania listy raportów.
    - Przyciski do dodawania nowego raportu i wyświetlania logów.

Komponent: FilterReports

- **Funkcje:**
    - FilterReports umożliwia filtrowanie raportów na podstawie różnych kryteriów, takich jak status, priorytet, data rozpoczęcia i zakończenia.
    - Wykorzystuje hook useState do przechowywania obecnie wybranych filtrów.

- **Stan:**
    - filters: Stan zawierający aktualnie stosowane filtry. Jest to obiekt typu Partial<Report>.

- **Metody:**
    - handleFilterChange: Funkcja wywoływana przy zmianie każdego z elementów filtrujących, aktualizująca stan filters i wywołująca onFilterChange.

- **Elementy UI:**
    - Cztery elementy do filtracji: dwa selektory dla statusu i priorytetu oraz dwa pola daty dla daty rozpoczęcia i zakończenia.

Katalog Reusable Components:

- **Komponent Header:**
    - **Funkcje:** Definiuje funkcjonalny komponent Header, który przyjmuje jeden prop content (string).
    - **Elementy UI:** UI składa się z tagu <h3>, wyświetlającego zawartość prop content.
- **Komponent Link:**
    - **Funkcje:** Definiuje komponent Link, przyjmujący dwa propsy: content (string) i link (string).
    - **Elementy UI:** UI to tag kotwicy (<a>), używający prop link jako atrybutu href i wyświetlający content jako tekst linku.
- **Komponent LogoutBtn:**
    - **Funkcje:** Definiuje funkcję lub komponent logoutBtn bez przyjmowanych propsów.
    - **Elementy UI:** Głównym elementem UI jest przycisk (<button>) z klasą logout, zawierający kotwicę (<a>) i ikonę SVG.
- **Komponent TableHeaders:**
    - **Funkcje:** Definiuje komponent TableHeaders, który przyjmuje dwa propsy: titles (tablica stringów) i optionalLast (Boolean).
    - **Elementy UI:** UI to nagłówek tabeli (<thead>), dynamicznie tworzący komórki nagłówka (<th>) dla każdego tytułu w tablicy titles, z warunkiem opcjonalnego pominięcia ostatniego tytułu.

2. Ścieżki i komponenty związane z routingiem

- /addReport - AddReport
- /edit/:reportId - EditReport
- / (domyślnie) - LoginForm
- /register - RegisterForm
- /reports - ShowReports
- /logs - ShowLogs

3. API serwera

Nasze API serwera zostało zaimplementowane przy użycia narzędzia **json-server**, które umożliwia nam szybkie i łatwe udostępnianie danych za pomocą protokołu HTTP.

Dostępne Endpointy:

- http://localhost:3000/servicemen
- http://localhost:3000/reports
- http://localhost:3000/clients
- http://localhost:3000/Logs

Wykorzystanie Endpointów:

Autentykacja Użytkowników

- AuthService.tsx
    - login
        - GET http://localhost:3000/servicemen?email=${email}&password=${password}
        - GET http://localhost:3000/clients?email=${email}&password=${password}
    - register
        - GET http://localhost:3000/servicemen?email=${email}
        - GET http://localhost:3000/clients?email=${email}
        - POST http://localhost:3000/clients

- getUserById
  - GET http://localhost:3000/servicemen/${id}
  - GET http://localhost:3000/clients/${id}

Operacje na Logach

- LogService.tsx
  - getLogs
    - GET http://localhost:3000/logs?reportIds=${reportsIds}
    - GET http://localhost:3000/logs
  - addLog
    - POST http://localhost:3000/logs
  - deleteLog
    - DELETE http://localhost:3000/logs/${logId}

Operacje na Raportach

- ReportService.tsx
  - getReports
    - GET http://localhost:3000/reports?userId=${userId}
    - GET http://localhost:3000/reports
  - createReport
    - POST http://localhost:3000/reports
  - getReport
    - GET http://localhost:3000/reports/${reportId}
  - updateReport
    - PATCH http://localhost:3000/reports/${reportId}

4. Szczególnie ciekawe fragmenty kodu

Asynchroniczne Ładowanie i Filtrowanie Danych w ShowReports:

- useState do Zarządzania Stanem:
  - Używa useState do przechowywania stanu raportów (reports) oraz filtrów (filters).
  - Stan reports zawiera listę raportów do wyświetlenia, a filters obiekt z bieżącymi filtrami.

- useEffect do Asynchronicznego Ładowania:
  - Hook useEffect jest używany do inicjowania żądania do API po załadowaniu komponentu oraz przy każdej zmianie filtrów.
  - Wewnątrz useEffect, wywoływana jest asynchroniczna funkcja fetchReports, która ładuje dane.

- Ładowanie i Filtrowanie Danych:
    - Funkcja fetchReports wykonuje asynchroniczne zapytanie do serwera (przy użyciu getReports z ReportService), aby pobrać raporty, które następnie są filtrowane na podstawie aktualnych wartości w filters.
    - Po pobraniu i przefiltrowaniu danych, stan reports jest aktualizowany, co powoduje ponowne renderowanie komponentu z nowymi danymi.

- Reakcja na Zmiany Filtrów:
    - Każda zmiana w filtrach (np. zmiana statusu lub daty w FilterReports) aktualizuje stan filters, co wywołuje ponowne wykonanie useEffect i ponowne załadowanie i przefiltrowanie raportów.

Zastosowanie komponentu prezentacyjnego do wyświetlania zgłoszeń:

- **ShowReports:**
    - ShowReports odpowiada zarządzanie listą raportów.
    - Wykorzystuje asynchroniczne zapytania do API do pobierania raportów i stosuje filtry określone przez użytkownika.
    - Zawiera logikę do obsługi filtrów i aktualizacji wyświetlanych danych.
    - Zintegrowany z komponentami takimi jak FilterReports i ReportsTable, tworzy kompleksowy widok zarządzania raportami.

- **ReportsTable:**
    - ReportsTable to komponent stricte prezentacyjny, który wyświetla dane raportów w formie tabeli.
    - Odbiera przefiltrowane raporty jako prop i renderuje je, wyświetlając kluczowe informacje w poszczególnych wierszach tabeli.
    - Zawiera elementy interaktywne, takie jak przyciski edycji lub przyjęcia raportu, które są wyświetlane tylko dla pracowników serwisu.

5. Wypunktowane elementy techniczne

- własna walidacja danych wprowadzanych przez użytkownika
    - RegisterForm.tsx
    - AddReport.tsx
    - EditReport.tsx

      Zostało to zrealizowane za pomocą Hooku useState, który oczekuje odpowiedniego interfejsu - RegistrationErrors/ReportErrors oraz funkcji validateForm, która uruchamia się w momencie próby zatwierdzenia formularza.

- użycie TypeScript

- wykorzystanie komponentów prezentacyjnych
    - ReportsTable.tsx
    - LogsTable.tsx

      Do tych komponentów przekazywane są tablice z raportami/logami, które następnie są zwracane w formie tabeli.

- dwukierunkowa komunikacja pomiędzy komponentami
    - ShowReports.tsx
    - FilterReports.tsx

      Dwukierunkowa komunikacja pomiędzy tymi komponentami działa na tej zasadzie, że ShowReports.tsx przekazuje do FilterReports.tsx funkcję przeznaczoną do modyfikacji wykorzystywanych filtrów, a następnie w zależności od poczynań użytkownika ulega ona tam modyfikacji, co ma odzwierciedlenie w wyświetlanych raportach.

- co najmniej 4 komponenty reużywalne

- modyfikacja danych odbywa się tylko w jednym komponencie

- operacje modyfikacji danych za pomocą 4 rodzajów żądań http
    - ReportService.tsx
        - GET (Pobieranie zgłoszeń):
            a. Metoda do pobierania danych (getReports), realizuje żądania GET do serwera API, aby pobrać listę raportów.

        - POST (Tworzenie nowych zgłoszeń):
            a. Metoda do tworzenia nowego raportu (createReport), używa żądania POST do wysłania nowego raportu do serwera i zapisania go w bazie danych.

        - PATCH (Modyfikowanie istniejących zgłoszeń):
            a. Metoda do aktualizacji istniejącego raportu, (updateReport), używa żądania PUT do przesłania zmodyfikowanych danych raportu na serwer i zaktualizowania go w bazie danych.

    - AuthService.tsx
        - GET (Pobieranie użytkowników):
            a. Metoda login używa żądań GET do pobierania użytkowników z serwera. Sprawdza, czy użytkownik istnieje w bazie danych i zwraca użytkownika.

        - POST (Tworzenie nowych użytkowników):
            a. Metoda register implementuje żądanie POST do tworzenia nowego użytkownika. Najpierw sprawdza, czy użytkownik już istnieje (za pomocą GET), a następnie, jeśli nie, rejestruje nowego użytkownika wysyłając jego dane.

    - LogService.tsx
        - GET (Pobieranie logów):
            a. Metoda getLogs: Realizuje żądanie GET do pobierania logów. Zwraca listę logów z bazy danych.

        - POST (Tworzenie nowych logów):
            a. Metoda addLog: Wykorzystuje żądanie POST do dodawania nowego logu.

        - DELETE (Usuwanie logów):
            a. Metoda deleteLog: Używa żądania DELETE do usunięcia konkretnego logu.

- żądania do serwera są zapisane w jednym oddzielnym pliku
    - ReportService.tsx
    - AuthService.tsx
    - LogService.tsx

      W plikach tych znajdują się metody pozwalające na komunikację z naszym API.

- routing
    - App.tsx

      Korzystamy tam z Route, które stanowią kontener dla zawartości, która ma być wyrenderowana pod wskazanym adresem URL

- wykorzystanie dwóch zmiennych właściwości routingu
    - navigate
        - AddReport.tsx
        - EditReport.tsx
        - LoginForm.tsx
        - RegisterForm.tsx

          Wykorzystujemy tam Hook useNavigate służący do przejścia pod dany adres URL.

    - params
        - EditReport.tsx

          Wykorzystujemy tam Hook useParams służący do uzyskania przesyłanego przez parametry id raportu, który ma ulec modyfikacji.

- brak błędów/ostrzeżeń w konsoli przeglądarki

6. Dodatkowe biblioteki

- React Router
    - Opis biblioteki - stworzona dla framework React, umożliwia nawigację i zarządzanie trasami w aplikacjach internetowych.
    - Cel użycia - nawigacja po poszczególnych widokach naszej aplikacji.

- JSON-SERVER
    - Opis biblioteki - narzędzie umożliwiające stworzenie “sztucznego” serwera API opartego na JSON.
    - Cel użycia - wykorzystaliśmy je w celu składowania i przeprowadzania operacji na danych - raportach, logach, klientach, serwisantach naszego serwisu komputerowego.

7. Podział pracy w zespole

- **Przemysław Rutkowski**
    - FilterReports.tsx
    - RegistrationForm.tsx
    - LoginForm.tsx
    - RegistrationErrors.ts
    - AuthService.tsx
    - LogService.tsx

- **Mateusz Mogielnicki**
    - ReportTable.tsx
    - LogsTable.tsx
    - AddReport.tsx
    - ReportErrors.tsx
    - Katalog Styles
    - Katalog Reusable Components

- **Jakub Matyszek:**
    - ShowReports.tsx
    - ShowLogs.tsx
    - EditReport.tsx
    - ReportService.tsx
    - UserLocalStorage.tsx