Projekt zaliczeniowy przedmiotu Frameworki front-endowe

Dokumentacja projektowa:

**Autor:**

Rafał Kwiecień

nr indeksu:13515

Ścieżka Programowanie aplikacji IS/WSEI N

Spis treści

[Wprowadzenie 3](#_Toc158465724)

[Cel Projektu 3](#_Toc158465725)

[Technologie 3](#_Toc158465726)

[Oczekiwane Korzyści 3](#_Toc158465727)

[Instalacja i Uruchamianie projektu 4](#_Toc158465728)

[Wymagania Systemowe: 4](#_Toc158465729)

[Struktura Katalogów i Plików 5](#_Toc158465730)

[Lista Funkcjonalności 7](#_Toc158465731)

[Opis Plików Klas 8](#_Toc158465732)

[App.tsx 8](#_Toc158465733)

[Opis: 8](#_Toc158465734)

[Metody: 8](#_Toc158465735)

[Elementy Interfejsu Użytkownika: 8](#_Toc158465736)

[Nawigacja: 8](#_Toc158465737)

[MainView.tsx 9](#_Toc158465738)

[Opis: 9](#_Toc158465739)

[Stan Komponentu: 9](#_Toc158465740)

[Metody: 9](#_Toc158465741)

[Elementy Interfejsu Użytkownika: 9](#_Toc158465742)

[Posts.tsx 11](#_Toc158465743)

[Opis: 11](#_Toc158465744)

[Stan Komponentu: 11](#_Toc158465745)

[Metody: 11](#_Toc158465746)

[Elementy Interfejsu Użytkownika: 11](#_Toc158465747)

[Profile.tsx 12](#_Toc158465748)

[Opis: 12](#_Toc158465749)

[Stan Komponentu: 12](#_Toc158465750)

[Metody: 12](#_Toc158465751)

[Elementy Interfejsu Użytkownika: 12](#_Toc158465752)

[Search.tsx 13](#_Toc158465753)

[Opis: 13](#_Toc158465754)

[Stan Komponentu: 13](#_Toc158465755)

[Metody: 13](#_Toc158465756)

[Elementy Interfejsu Użytkownika: 13](#_Toc158465757)

[ApiCalls.ts 14](#_Toc158465758)

# Wprowadzenie

Projekt powstał w ramach realizacji przedmiotu Frameworki front-endowe, ma na celu stworzenie strony font-endowej portalu do dzielenia się zdjęciami i postami. Galeria Zdjęć umożliwia użytkownikom łatwe przeglądanie, dodawanie i usuwanie zdjęć. Daje też możliwość przeglądania twórczości innych użytkowników portalu i dzielenia się przemyśleniami w postach i komentarzach.

### Cel Projektu

Celem tego projektu jest dostarczenie narzędzia umożliwiającego dziekenie się i zarządzanie zdjęciami. W projekcie skoncentrowałem się na prostocie obsługi, jednocześnie oferując wystarczające funkcjonalności do zaspokojenia podstawowych potrzeb użytkowników w zakresie przechowywania i przeglądania zdjęć.

### Technologie

Projekt został zrealizowany przy użyciu technologii takich jak React do budowy interfejsu użytkownika, TypeScript dla lepszej kontroli nad typami, oraz Node.js jako platformy do obsługi logiki serwerowej. Struktura projektu została zaprojektowana z myślą o małym skomplikowaniu projektu.

### Oczekiwane Korzyści

* Łatwa nawigacja.
* Intuicyjny interfejs.
* Budowanie relacji biznesowych użytkowników.
* Możliwość tworzenia albumów zawierających zestawy zdjęć i udostępniania ich publicznie.
* Dzielenie się poradami na forum.
* Zapewnienie dostępu do galerii zdjęć z dowolnego miejsca, co umożliwia pracę poza domem.

# Instalacja i Uruchamianie projektu

### Wymagania Systemowe:

Upewnij się, że Twój system spełnia następujące wymagania:

* Node.js (zalecana wersja LTS)
* npm (menedżer pakietów dla Node.js)

Sklonuj repozytorium ze strony Git-a lub użyj załączonych plików:

* <https://github.com/Arcanagos/Proj-frontend-studia-3.1>

Otwórz Wiersz poleceń i przejdź w nim do katalogu projektu, następnie zainstaluj niezbędne zależności przy użyciu poniższej komendy:

* npm install

W wierszu poleceń uruchom serwer lokalny następującym poleceniem:

* npm start

Po poprawnym uruchomieniu, projekt będzie dostępny pod adresem domyślnym http://localhost:3000 lub innym, podanym w oknie konsoli, jeżeli skonfigurowano inaczej lub port jest niedostępny.

Do zalogowania można użyć następującego loginu:

* Bret

Lub dowolnego innego username z pliku json z adresu:

* <https://jsonplaceholder.typicode.com/users/>

# Struktura Katalogów i Plików

* .gitignore
  + Plik konfiguracyjny dla systemu kontroli wersji Git, określający, które pliki i katalogi mają być ignorowane podczas śledzenia zmian.
* galeriaprojfrontframeworkrk.esproj
  + Plik projektu dla środowiska Visual Studio.
* galeriaprojfrontframeworkrk.sln
  + Plik rozwiązania dla środowiska Visual Studio
* nuget.config
  + Plik konfiguracyjny dla narzędzia NuGet, które jest menedżerem pakietów dla projektów .NET.
* package-lock.json
  + Plik używany w projektach Node.js, zawierający informacje o zablokowanych wersjach zależności.
* package.json
  + Plik konfiguracyjny dla projektu Node.js, zawiera informacje o zależnościach i skryptach.
* tsconfig.json
  + Plik konfiguracyjny dla TypeScript, zawierający ustawienia kompilacji.
* public/
  + Katalog zawierający pliki publiczne, takie jak ikony, manifesty, pliki HTML. Jest dostępny publicznie ze strony serwera. Zaiwera niezbędne elemnty Reacta.
* public/index.html
  + Główny plik HTML aplikacji React, punkt wejścia dla renderowania komponentów.
* public/manifest.json
  + Plik manifestu Progressive Web App (PWA), zawierający informacje o aplikacji.
* src/
  + Katalog źródłowy projektu, zawierający pliki źródłowe, komponenty, usługi i widoki.
* src/index.css, src/index.js, src/index.tsx
  + Pliki związane z punktem wejścia aplikacji React. Zawierają routowanie aplikacji w React.
* src/react-app-env.d.ts, src/reportWebVitals.js, src/reportWebVitals.ts, src/setupTests.js, src/setupTests.ts
  + Pliki związane z narzędziami deweloperskimi, testowymi i monitorowaniem wydajności.
* src/Sites/
  + Katalog zawierający pliki związane z różnymi widokami (stronami) aplikacji.
* src/Sites/ApiCalls.ts
  + Moduł obsługujący komunikację z API. Zawiera dzielone interfejsy.
* src/Sites/App.css, src/Sites/App.tsx
  + Pliki związane z głównym widokiem aplikacj. Ekran Logowania
* src/Sites/MainView.css, src/Sites/MainView.tsx
  + Pliki związane z głównym widokiem galerii zdjęć.
* src/Sites/Posts.css, src/Sites/Posts.tsx
  + Pliki związane z widokiem postów w aplikacji.
* src/Sites/Profile.css, src/Sites/Profile.tsx
  + Pliki związane z widokiem profilu użytkownika.
* src/Sites/Search.css, src/Sites/Search.tsx
  + Pliki związane z widokiem wyszukiwania w aplikacji.

# Lista Funkcjonalności

1. Logowanie użytkowników – logowanie po username
2. Strona główna z feedem:
   1. Wybór katalogu
   2. Wybór zdjęcia
   3. Dodawanie zdjęć
   4. Usuwanie swoich zdjęć
   5. Filtorwanie po ID
3. Strona danych użytkownika:
   1. Lista informacji o userze z możliwością edycji
   2. Widok dodanych zdjęć
   3. Widok dodanych albumów
   4. Widok dodanych postów
4. Strona wyszukiwania:
   1. Wyszukiwanie informacji o userze
   2. Wyszukiwanie zdjęć po id
   3. Wyszukiwanie albumów po id
5. Strona z postami:
   1. Czytanie postów wraz z komentarzami
   2. Możliwośc usunięcia swoich postów

# Opis Plików Klas

## App.tsx

### Opis:

Główny komponent aplikacji, który odpowiada za logikę logowania użytkownika do aplikacji "Galerianka". Komponent ten wykorzystuje React w połączeniu z React Router oraz funkcje z pliku ApiCalls.ts do komunikacji z serwerem.

### Metody:

1. **Login:**
   * Metoda asynchroniczna obsługująca proces logowania użytkownika.
   * Pobiera wartość loginu z elementu o identyfikatorze "Login".
   * Wywołuje funkcję getAllUsers z pliku ApiCalls w celu pobrania danych wszystkich użytkowników.
   * Sprawdza, czy istnieje użytkownik o podanym loginie.
   * Jeśli użytkownik zostanie znaleziony, przekierowuje do widoku głównego (/MainView) z identyfikatorem użytkownika.
   * W przypadku niepoprawnych danych wyświetla alert informujący o błędzie.

### Elementy Interfejsu Użytkownika:

* **Div z Klasą "App":**
  + Kontener obejmujący całą aplikację.
* **Paragraph z Id "Galerianka":**
  + Nagłówek prezentujący nazwę aplikacji.
* **Input z Id "Login":**
  + Pole do wprowadzania loginu użytkownika.
* **Button z Id "LoginButton":**
  + Przycisk wywołujący funkcję logowania (Login).

### Nawigacja:

* **Login:**
  + Użytkownik wpisuje swój login w pole "Login".
  + Po naciśnięciu przycisku "Zaloguj", uruchamiana jest funkcja Login.
  + W przypadku poprawnych danych, użytkownik jest przekierowywany do widoku głównego (/MainView/:userId).

## MainView.tsx

### Opis:

MainView.tsx to główny komponent reprezentujący widok strony głównej aplikacji "Galerianka". Komponent ten obsługuje wyświetlanie albumów, zdjęć oraz umożliwia użytkownikowi dodawanie nowych zdjęć do albumów.

### Stan Komponentu:

* **allAlbums:**
  + Stan przechowujący wszystkie albumy użytkownika.
* **filteredAlbums:**
  + Stan przechowujący albumy po zastosowaniu filtru.
* **selectedAlbum:**
  + Stan przechowujący aktualnie wybrany album.
* **expandedPhoto:**
  + Stan przechowujący informacje o rozwiniętym zdjęciu.
* **filterId:**
  + Stan przechowujący wartość filtru według ID.

### Metody:

* **scrollToTop:**
  + Funkcja przewijająca stronę do góry.
* **handleAlbumClick:**
  + Funkcja obsługująca kliknięcie na album, ustawiająca go jako wybrany.
* **handleShowAddingClick:**
  + Funkcja obsługująca pokazanie formularza dodawania zdjęcia.
* **handlePhotoClick:**
  + Funkcja obsługująca kliknięcie na zdjęcie, ustawiająca je jako rozwinięte.
* **handleCloseClick:**
  + Funkcja obsługująca zamknięcie rozwiniętego zdjęcia.
* **handleDeleteClick:**
  + Funkcja obsługująca kliknięcie przycisku usuwania zdjęcia.
* **handleAddPhotoClick:**
  + Funkcja obsługująca kliknięcie przycisku dodawania nowego zdjęcia.
* **goToProfile, goToPosts, goToSearch:**
  + Funkcje obsługujące nawigację do innych widoków aplikacji.

### Elementy Interfejsu Użytkownika:

* **Div z Id "MainViewContainer":**
  + Główny kontener widoku.
* **Div z Id "Menu":**
  + Kontener menu z nagłówkiem "Galerianka" i listą nawigacyjną.
* **Div z Id "Content":**
  + Główna treść widoku, zawiera albumy, zdjęcia i formularz dodawania zdjęć.
* **Div z Id "ZdjTitleDiv":**
  + Nagłówek z przyciskiem dodawania zdjęć i formularzem dodawania.
* **Div z Id "Photos":**
  + Kontener wyświetlający miniatury zdjęć.
* **Div z Id "Albumes":**
  + Kontener wyświetlający albumy użytkownika.

## Posts.tsx

### Opis:

Posts.tsx to komponent reprezentujący widok postów oraz komentarzy użytkownika w aplikacji "Galerianka". Komponent ten pobiera i wyświetla posty wraz z komentarzami, a także umożliwia użytkownikowi usuwanie własnych postów.

### Stan Komponentu:

* **postsWithComments:**
  + Stan przechowujący posty wraz z komentarzami.

### Metody:

* **deletePost:**
  + Funkcja usuwająca post o podanym identyfikatorze.

### Elementy Interfejsu Użytkownika:

* **Div z Id "MainViewContainer":**
  + Główny kontener widoku.
* **Div z Id "Menu":**
  + Kontener menu z nagłówkiem "Galerianka" i listą nawigacyjną.
* **Div z Id "Content":**
  + Główna treść widoku, zawierająca posty i komentarze.
* **Div z Id "PostsDiv":**
  + Kontener wyświetlający posty wraz z komentarzami.

## Profile.tsx

### Opis:

Profile.tsx to komponent reprezentujący widok profilu użytkownika w aplikacji "Galerianka". Komponent ten umożliwia użytkownikowi zarządzanie danymi swojego profilu, jak również prezentuje aktywności takie jak zdjęcia, albumy i posty użytkownika.

### Stan Komponentu:

* **userData:**
  + Stan przechowujący dane użytkownika.
* **userPhotos:**
  + Stan przechowujący zdjęcia użytkownika.
* **userAlbums:**
  + Stan przechowujący albumy użytkownika.
* **userPosts:**
  + Stan przechowujący posty użytkownika.

### Metody:

* **fetchData:**
  + Funkcja pobierająca asynchronicznie dane użytkownika, zdjęcia, albumy i posty.
* **sendPutRequest:**
  + Funkcja wysyłająca zaktualizowane dane użytkownika do serwera.
* **updateValues:**
  + Funkcja aktualizująca wartości pól formularza danymi użytkownika.

### Elementy Interfejsu Użytkownika:

* **Div z Id "MainViewContainer":**
  + Główny kontener widoku.
* **Div z Id "Menu":**
  + Kontener menu z nagłówkiem "Galerianka" i listą nawigacyjną.
* **Div z Id "Content":**
  + Główna treść widoku, zawierająca dane użytkownika, formularz edycji oraz aktywności.
* **Div z Id "ProfileDiv":**
  + Kontener wyświetlający sekcje z danymi, formularzem i aktywnościami użytkownika.
* **Div z Id "UserInfo":**
  + Sekcja z danymi użytkownika oraz formularzem edycji.
* **Div z Id "UserStuff":**
  + Sekcja z aktywnościami użytkownika, takimi jak zdjęcia, albumy i posty.

# Search.tsx

### Opis:

Search.tsx to komponent aplikacji "Galerianka" umożliwiający użytkownikowi wyszukiwanie informacji na podstawie różnych kryteriów. Komponent ten obsługuje wyszukiwanie użytkowników, zdjęć oraz albumów na podstawie różnych parametrów.

### Stan Komponentu:

* **user:**
  + Stan przechowujący dane znalezionego użytkownika.
* **photo:**
  + Stan przechowujący dane znalezionego zdjęcia.
* **album:**
  + Stan przechowujący dane znalezionego albumu.

### Metody:

* **handleFindUserByName:**
  + Funkcja obsługująca wyszukiwanie użytkownika na podstawie imienia lub nazwiska.
* **handleFindPhotoById:**
  + Funkcja obsługująca wyszukiwanie zdjęcia na podstawie ID.
* **handleFindAlbumById:**
  + Funkcja obsługująca wyszukiwanie albumu na podstawie ID.

### Elementy Interfejsu Użytkownika:

* **Div z Id "MainViewContainer":**
  + Główny kontener widoku.
* **Div z Id "Menu":**
  + Kontener menu z nagłówkiem "Galerianka" i listą nawigacyjną.
* **Div z Id "SearchDiv":**
  + Główny kontener sekcji wyszukiwania.
* **Div z Id "FindUserDiv":**
  + Sekcja wyszukiwania użytkownika na podstawie imienia lub nazwiska.
* **Div z Id "FindPhotoDiv":**
  + Sekcja wyszukiwania zdjęcia na podstawie ID.
* **Div z Id "FindAlbumDiv":**
  + Sekcja wyszukiwania albumu na podstawie ID.
* **Inputy i Buttony:**
  + Pola tekstowe i przyciski umożliwiające wprowadzanie kryteriów wyszukiwania oraz uruchamianie wyszukiwania.
* **Div z Id "FoundUserDiv":**
  + Wyświetlana, gdy znaleziono użytkownika, zawiera informacje o użytkowniku.
* **Div z Id "FoundPhotoDiv":**
  + Wyświetlana, gdy znaleziono zdjęcie, zawiera informacje o zdjęciu.
* **Div z Id "FoundAlbumDiv":**
  + Wyświetlana, gdy znaleziono album, zawiera informacje o albumie.

## ApiCalls.ts

Plik ApiCalls.ts zawiera zestaw funkcji do obsługi komunikacji z serwerem JSONPlaceholder, który udostępnia fałszywe dane dla testowania. Poniżej znajduje się opis każdej z funkcji:

1. **getAllPhotos(): Promise<Photo[]>**
   * Pobiera wszystkie zdjęcia z serwera JSONPlaceholder.
   * Zwraca listę obiektów reprezentujących zdjęcia.
2. **getAllUsers(): Promise<UserData[]>**
   * Pobiera wszystkich użytkowników z serwera JSONPlaceholder.
   * Zwraca listę obiektów reprezentujących użytkowników.
3. **getModifiableAlbumes(senderId: string | undefined): Promise<Album[]>**
   * Pobiera albumy z serwera JSONPlaceholder, dodaje informacje o edycji (canEdit) i przypisuje zdjęcia do odpowiednich albumów.
   * Zwraca listę obiektów reprezentujących albumy.
4. **addPhoto(title: string, filePath: string, albumId: string, senderId: string | undefined): Promise<boolean>**
   * Dodaje nowe zdjęcie na serwer JSONPlaceholder.
   * Zwraca true, jeśli operacja zakończy się sukcesem, w przeciwnym razie false.
5. **getUserData(senderId: string | undefined): Promise<UserData>**
   * Pobiera dane użytkownika na podstawie ID z serwera JSONPlaceholder.
   * Zwraca obiekt reprezentujący dane użytkownika.
6. **putUserData(senderId: string | undefined, updatedData: UserData): Promise<boolean>**
   * Aktualizuje dane użytkownika na serwerze JSONPlaceholder.
   * Zwraca true, jeśli operacja zakończy się sukcesem, w przeciwnym razie false.
7. **getUserPhotos(senderId: string | undefined): Promise<Photo[]>**
   * Pobiera zdjęcia użytkownika, które można edytować (mają canEdit: true).
   * Zwraca listę obiektów reprezentujących zdjęcia.
8. **getUserAlbumes(senderId: string | undefined): Promise<Album[]>**
   * Pobiera albumy użytkownika na podstawie ID.
   * Zwraca listę obiektów reprezentujących albumy.
9. **findUserByName(name: string): Promise<UserData | undefined>**
   * Wyszukuje użytkownika po imieniu na serwerze JSONPlaceholder.
   * Zwraca obiekt reprezentujący znalezionego użytkownika lub undefined, jeśli nie znaleziono.
10. **findPhotoById(id: string): Promise<Photo>**
    * Wyszukuje zdjęcie po ID na serwerze JSONPlaceholder.
    * Zwraca obiekt reprezentujący znalezione zdjęcie.
11. **findAlbumById(id: string): Promise<Album>**
    * Wyszukuje album po ID na serwerze JSONPlaceholder.
    * Zwraca obiekt reprezentujący znaleziony album.
12. **getUserPosts(senderId: string | undefined): Promise<Post[]>**
    * Pobiera posty użytkownika na serwerze JSONPlaceholder.
    * Zwraca listę obiektów reprezentujących posty.
13. **getAllPosts(): Promise<Post[]>**
    * Pobiera wszystkie posty z serwera JSONPlaceholder.
    * Zwraca listę obiektów reprezentujących posty.
14. **getAllComments(): Promise<Comment[]>**
    * Pobiera wszystkie komentarze z serwera JSONPlaceholder.
    * Zwraca listę obiektów reprezentujących komentarze.
15. **getPostsAndComments(senderId: string | undefined): Promise<Post[]>**
    * Pobiera posty i komentarze użytkownika, przypisuje komentarze do odpowiednich postów.
    * Zwraca listę obiektów reprezentujących posty z przypisanymi komentarzami.
16. **DeletePicture(id: string): Promise<boolean>**
    * Usuwa zdjęcie na serwerze JSONPlaceholder na podstawie ID.
    * Zwraca true, jeśli operacja zakończy się sukcesem, w przeciwnym razie false.
17. **DeletePost(id: string): Promise<boolean>**
    * Usuwa post na serwerze JSONPlaceholder na podstawie ID.
    * Zwraca true, jeśli operacja zakończy się sukcesem, w przeciwnym razie false.
18. **Interfejs: UserData**
    * Definiuje strukturę obiektów reprezentujących dane użytkownika.
19. **Interfejsy: Photo, Album**
    * Definiują strukturę obiektów reprezentujących zdjęcia (Photo) i albumy (Album).
20. **Interfejsy: Post, Comment**
    * Definiują strukturę obiektów reprezentujących posty (Post) i komentarze (Comment).