📘 **Dokumentacja Techniczna Frontend — Project Manager**

📁 Struktura plików:

Project Manager/

├── CSS/

│ └── styles.css

├── Images/

│ └── computer.png

│ ├── eye-open.svg

│ └── eye-closed.svg

├── JS/

│ ├── add\_project.js

│ ├── add\_user.js

│ └── button\_reaction.js

│ └── log.js

│ └── log\_validation.js

│ └── projects.js

│ └── register.js

│ └── show\_password.js

│ └── update\_projects.js

│ └── update\_user.js

│ └── urls.js

│ └── user.js

├── add\_projects.html

└── index.html

└── log.html

└── projects.html

└── register.html

└── update\_projects.html

└── users.html

└── users\_add.html

└── users\_update.html

## **Pliki HTML:**

## **1. 📄 log.html – Strona logowania**

### **Opis:**

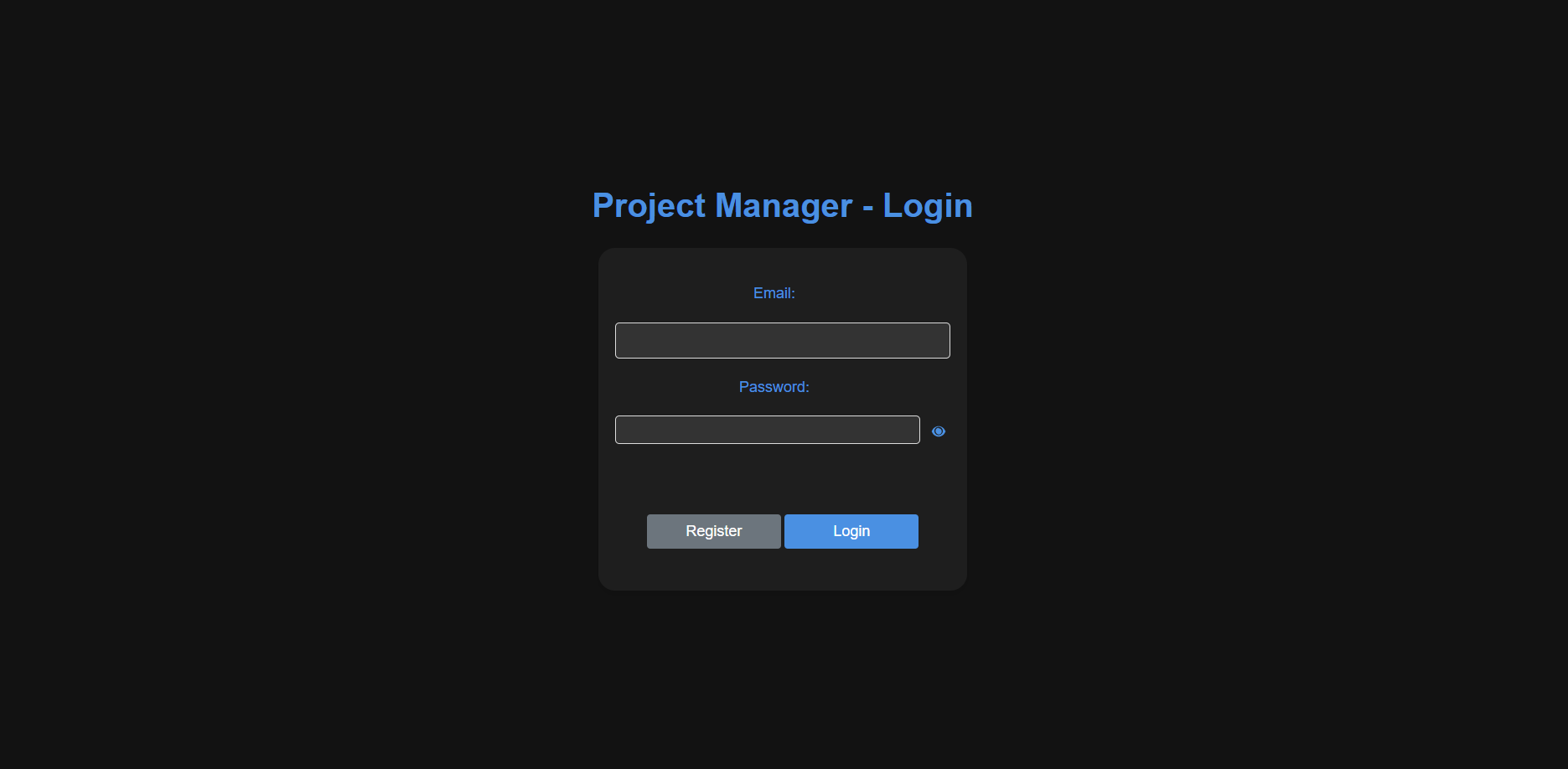
Strona umożliwia użytkownikowi zalogowanie się do systemu. Wczytuje dwa skrypty JS: obsługę logiki logowania (log.js) oraz przełączanie widoczności hasła (show\_password.js).

### **Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| <form> | Formularz logowania (bez action, obsługiwany przez JS). |
| #login | Pole tekstowe na e-mail użytkownika. |
| #password | Pole typu password z ikoną oka umożliwiającą pokazanie hasła. |
| #loginButton | Przycisk logowania, jego akcja definiowana jest w log.js. |
| .button2 | Przycisk do przekierowania na stronę rejestracji. |

### **Zależności:**

* **CSS**: CSS/styles.css – odpowiada za layout i styl.
* **JS**:
  + log.js – obsługuje proces logowania (brak implementacji w załączonych plikach, do uzupełnienia).
  + show\_password.js – umożliwia przełączanie widoczności hasła.



## **2. 📄 register.html – Strona rejestracji**

### **Opis:**

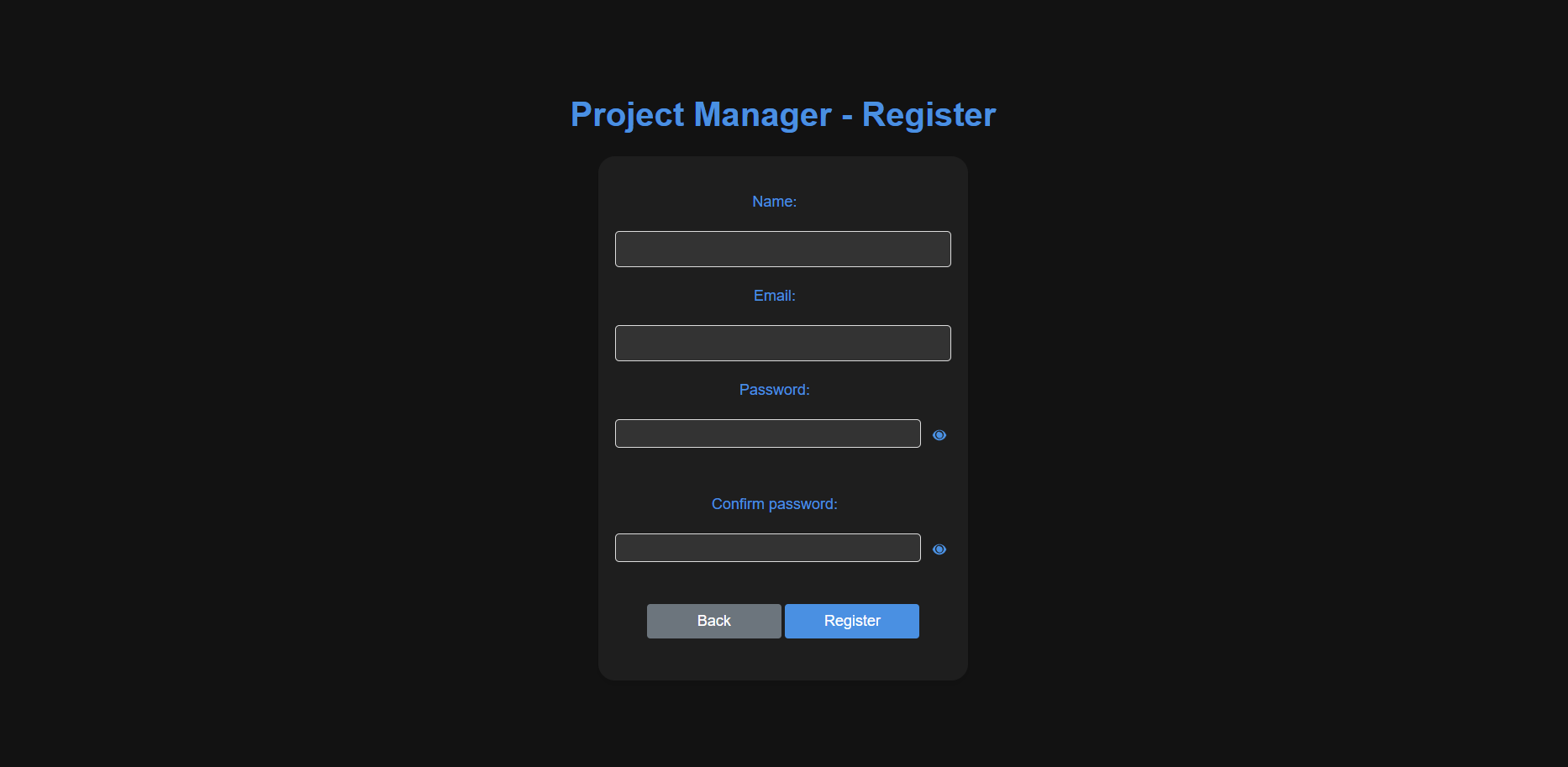
Formularz rejestracyjny dla nowych użytkowników. Zawiera weryfikację hasła oraz obsługę pokazywania/ukrywania haseł.

### **Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| <form id="registerForm"> | Formularz z kontrolkami do rejestracji. Obsługiwany przez register.js. |
| #name | Imię i nazwisko użytkownika. |
| #email | Adres e-mail. |
| #password | Hasło użytkownika. |
| #confirmPassword | Powtórzone hasło do walidacji. |
| #registerButton | Przycisk do wysłania danych rejestracyjnych. |
| .button2 | Przycisk powrotu do logowania. |

### **Zależności:**

* **CSS**: CSS/styles.css
* **JS**:
  + register.js – obsługuje walidację i logikę rejestracji użytkownika.
  + show\_password.js – umożliwia przełączanie widoczności hasła.



## **3. 📄 index.html – Strona główna aplikacji**

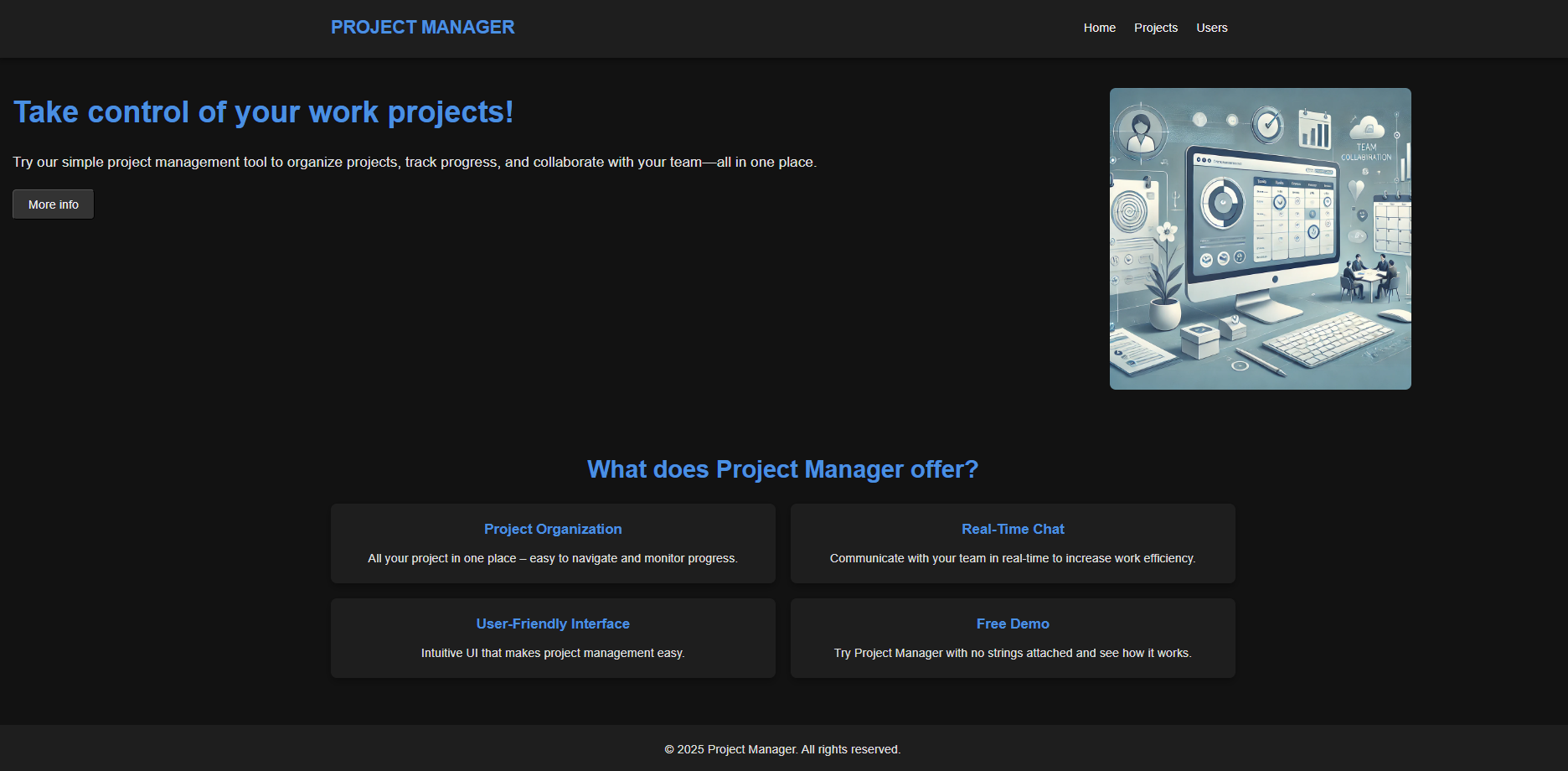
**Opis:**  
 Strona startowa aplikacji Project Manager, która prezentuje główne zalety i funkcjonalności systemu oraz umożliwia nawigację do kluczowych modułów (Projekty, Użytkownicy). Zawiera sekcję powitalną z przyciskiem „More info” rozwijającym dodatkowy opis.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| .hero | Sekcja główna z nagłówkiem, opisem i przyciskiem „More info”. |
| #moreInfoBtn | Przycisk, który po kliknięciu rozwija dodatkowy opis (#moreText). |
| #moreText | Ukryty na początku blok z dodatkowymi informacjami o aplikacji. |
| .features | Sekcja prezentująca kluczowe funkcjonalności aplikacji (organizacja projektów, czat, UI, demo). |
| <footer> | Stopka strony z prawami autorskimi. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja całej strony i sekcji.
* JS: JS/button\_reaction.js (moduł ES6, załadowany z defer) – obsługuje interakcję przycisku „More info”.
* Obrazy: Images/computer.png – ilustracja wspierająca wygląd sekcji hero.



## **4. 📄 add\_project.html – Formularz dodawania projektu**

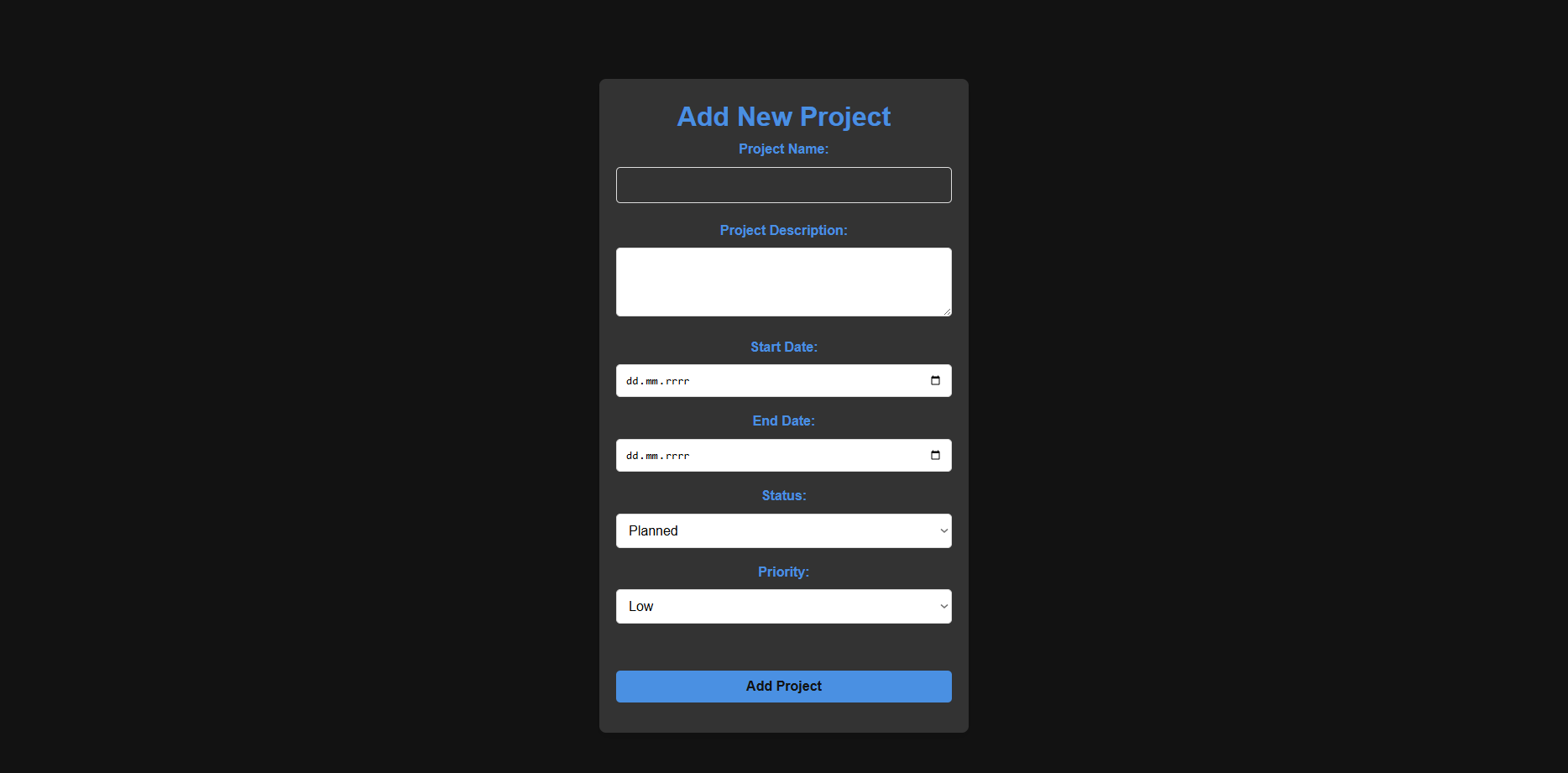
**Opis:**  
 Strona umożliwia użytkownikowi dodanie nowego projektu. Formularz zawiera pola do wprowadzenia nazwy, opisu, dat rozpoczęcia i zakończenia, statusu oraz priorytetu projektu. Dane przesyłane są do backendu w celu utworzenia nowego wpisu projektu.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| <form id="projectForm"> | Formularz do wprowadzania danych nowego projektu. |
| #name | Pole tekstowe na nazwę projektu. |
| #description | Pole tekstowe (textarea) na szczegółowy opis projektu. |
| #startDate | Pole wyboru daty rozpoczęcia projektu. |
| #endDate | Pole wyboru daty zakończenia projektu. |
| #status | Lista rozwijana z wyborem statusu projektu (Planned, In Progress, Completed, Cancelled). |
| #priority | Lista rozwijana z wyborem priorytetu (Low, Medium, High). |
| .btn | Przycisk do zatwierdzenia i wysłania formularza. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja formularza i elementów strony.
* JS: JS/add\_projects.js (moduł ES6, załadowany z defer) – obsługuje logikę formularza i komunikację z backendem.



## **5. 📄 projects.html – Strona zarządzania projektami**

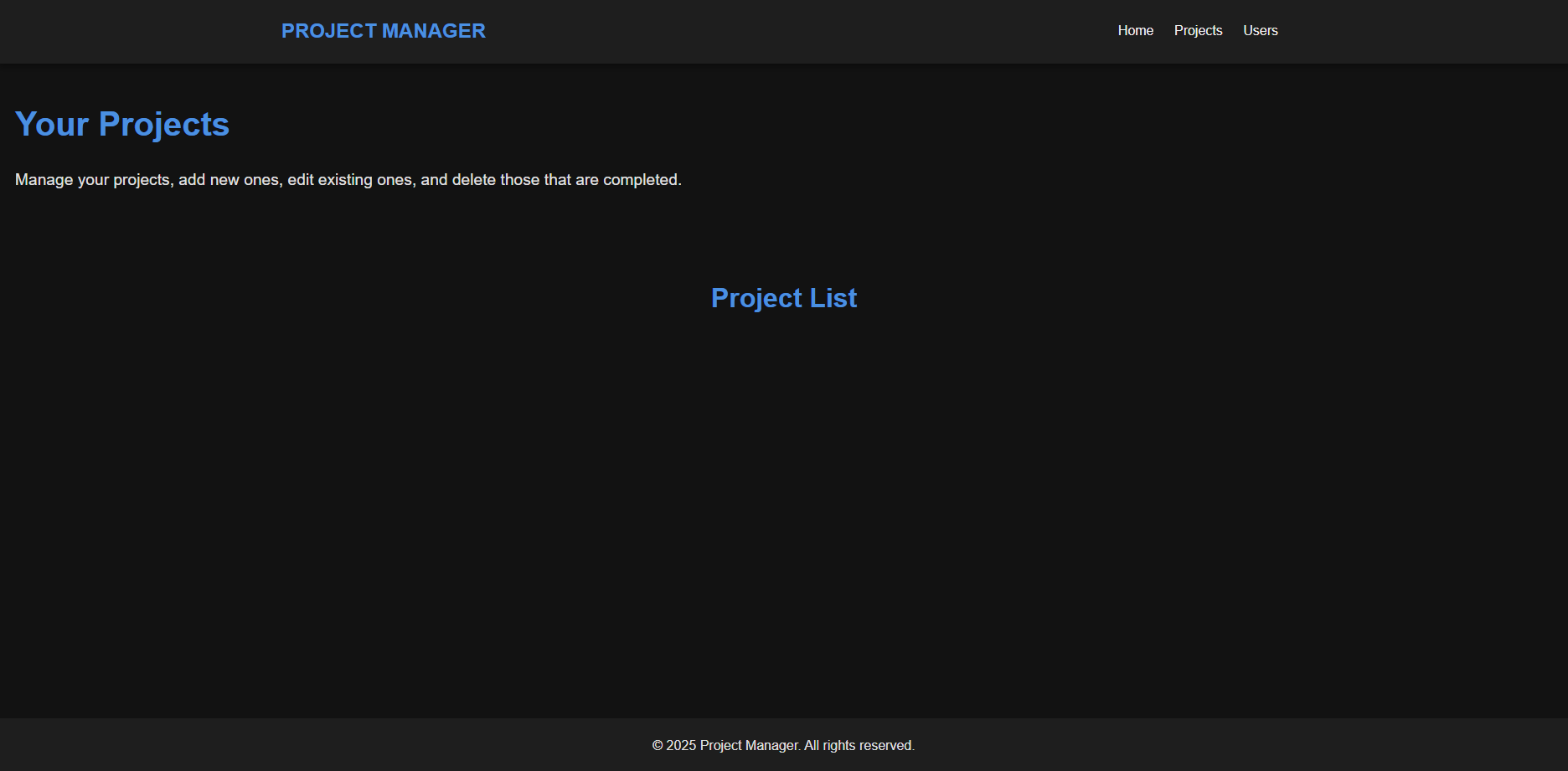
**Opis:**  
 Strona prezentująca listę projektów użytkownika. Umożliwia przeglądanie, dodawanie, edytowanie oraz usuwanie projektów. Lista projektów jest generowana dynamicznie przez skrypt JS.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| .nav-item.active | Link do bieżącej strony (Projects) oznaczony jako aktywny. |
| .hero | Sekcja powitalna z tytułem i opisem funkcji strony. |
| #project-list | Kontener, w którym dynamicznie generowana jest lista projektów przez projects.js. |
| <footer> | Stopka strony z prawami autorskimi. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja całej strony.
* JS: JS/projects.js (moduł ES6, załadowany z defer) – obsługuje pobieranie i wyświetlanie listy projektów oraz interakcje użytkownika z nimi.



## **6. 📄 update\_projects.html – Edycja istniejącego projektu**

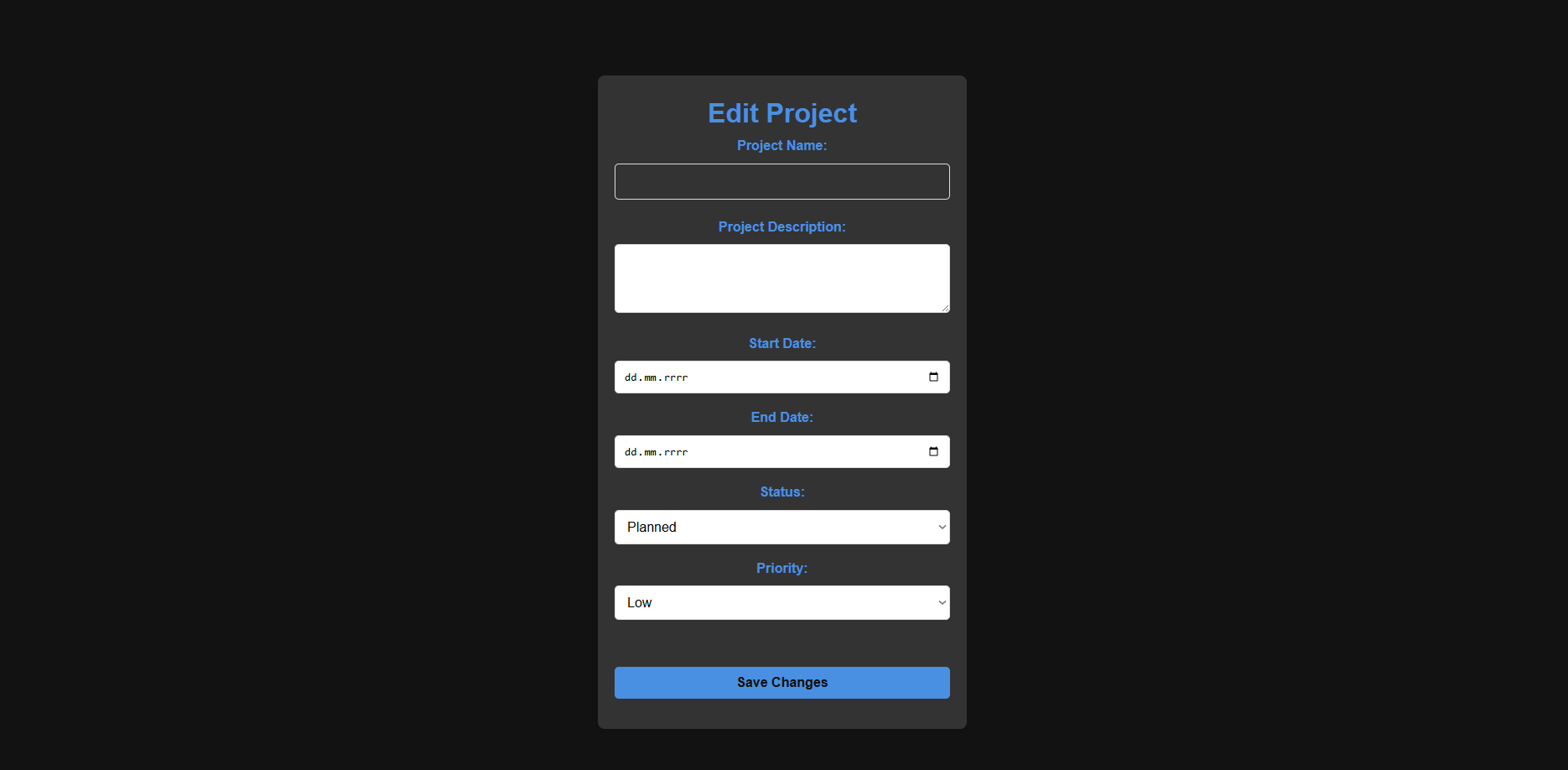
**Opis:**  
 Strona pozwalająca na edycję wybranego projektu. Formularz umożliwia zmianę nazwy, opisu, dat rozpoczęcia i zakończenia, statusu oraz priorytetu projektu. Dane są przesyłane do backendu w celu aktualizacji.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| <form id="editProjectForm" class="addForm"> | Formularz do edycji danych projektu. |
| #name | Pole tekstowe na nazwę projektu. |
| #description | Pole tekstowe (textarea) na opis projektu. |
| #startDate | Pole wyboru daty rozpoczęcia. |
| #endDate | Pole wyboru daty zakończenia. |
| #status | Lista rozwijana z wyborem statusu projektu. |
| #priority | Lista rozwijana z wyborem priorytetu projektu. |
| button[type="submit"] | Przycisk zatwierdzający zmiany i wysyłający formularz. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja formularza i strony.
* JS: JS/update\_projects.js (moduł ES6, załadowany z defer) – obsługuje logikę pobierania danych projektu do edycji oraz aktualizacji na backendzie.



## **7. 📄 users.html – Strona zarządzania użytkownikami**

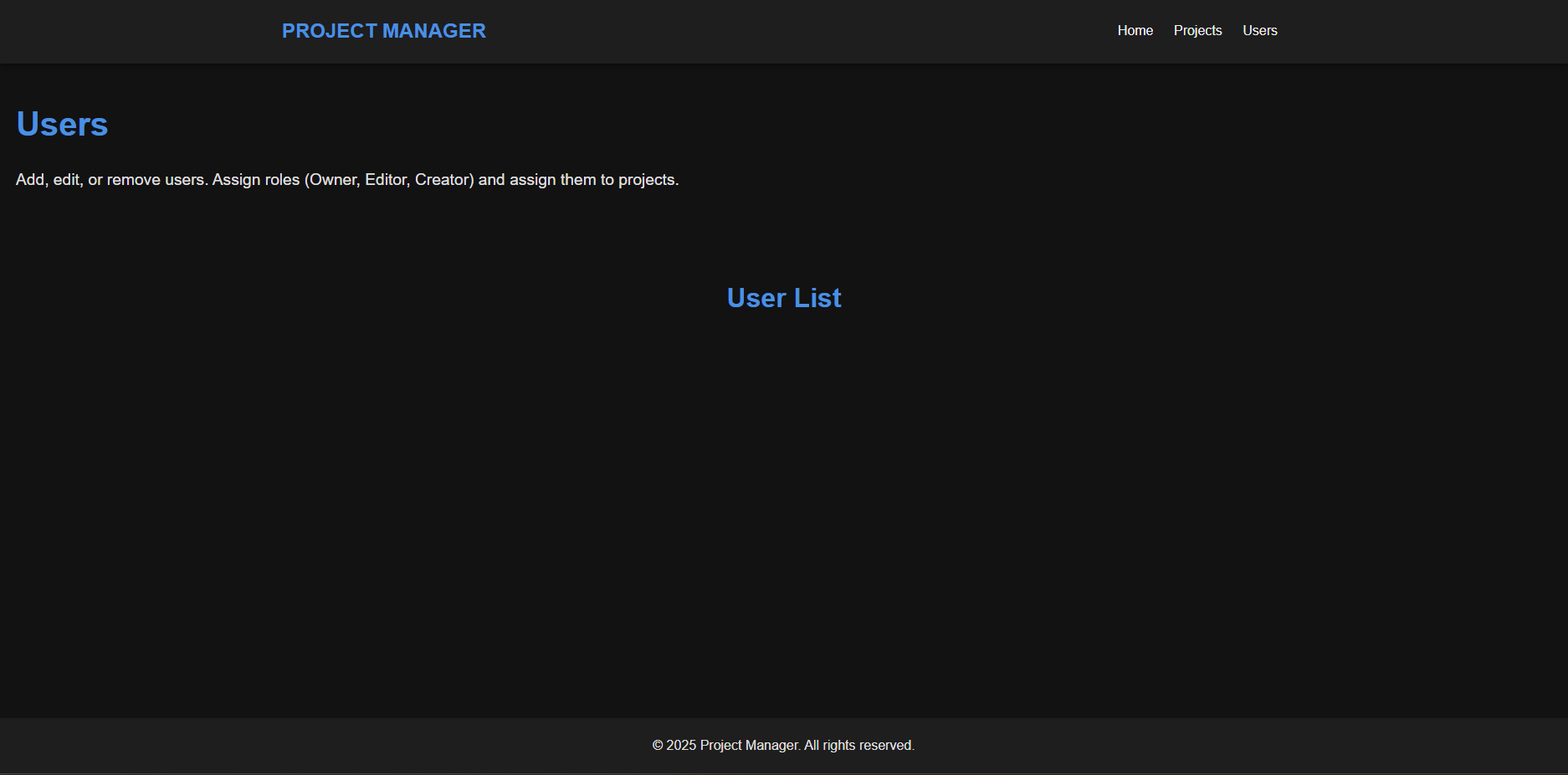
**Opis:**  
 Strona umożliwia zarządzanie użytkownikami systemu. Pozwala na przeglądanie listy użytkowników, edycję, usuwanie oraz przypisywanie ról (Owner, Editor, Creator) i przydzielanie użytkowników do projektów. Lista użytkowników jest generowana dynamicznie.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| .nav-item.active | Link do bieżącej strony (Users) oznaczony jako aktywny. |
| .hero | Sekcja powitalna z tytułem i krótkim opisem funkcji strony. |
| #user-list | Kontener, w którym dynamicznie generowana jest lista użytkowników przez users.js. |
| <footer> | Stopka strony z prawami autorskimi. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja całej strony.
* JS: JS/users.js (moduł ES6, załadowany z defer) – obsługuje pobieranie, wyświetlanie oraz zarządzanie użytkownikami.



## **8. 📄 users\_add.html – Dodawanie nowego użytkownika**

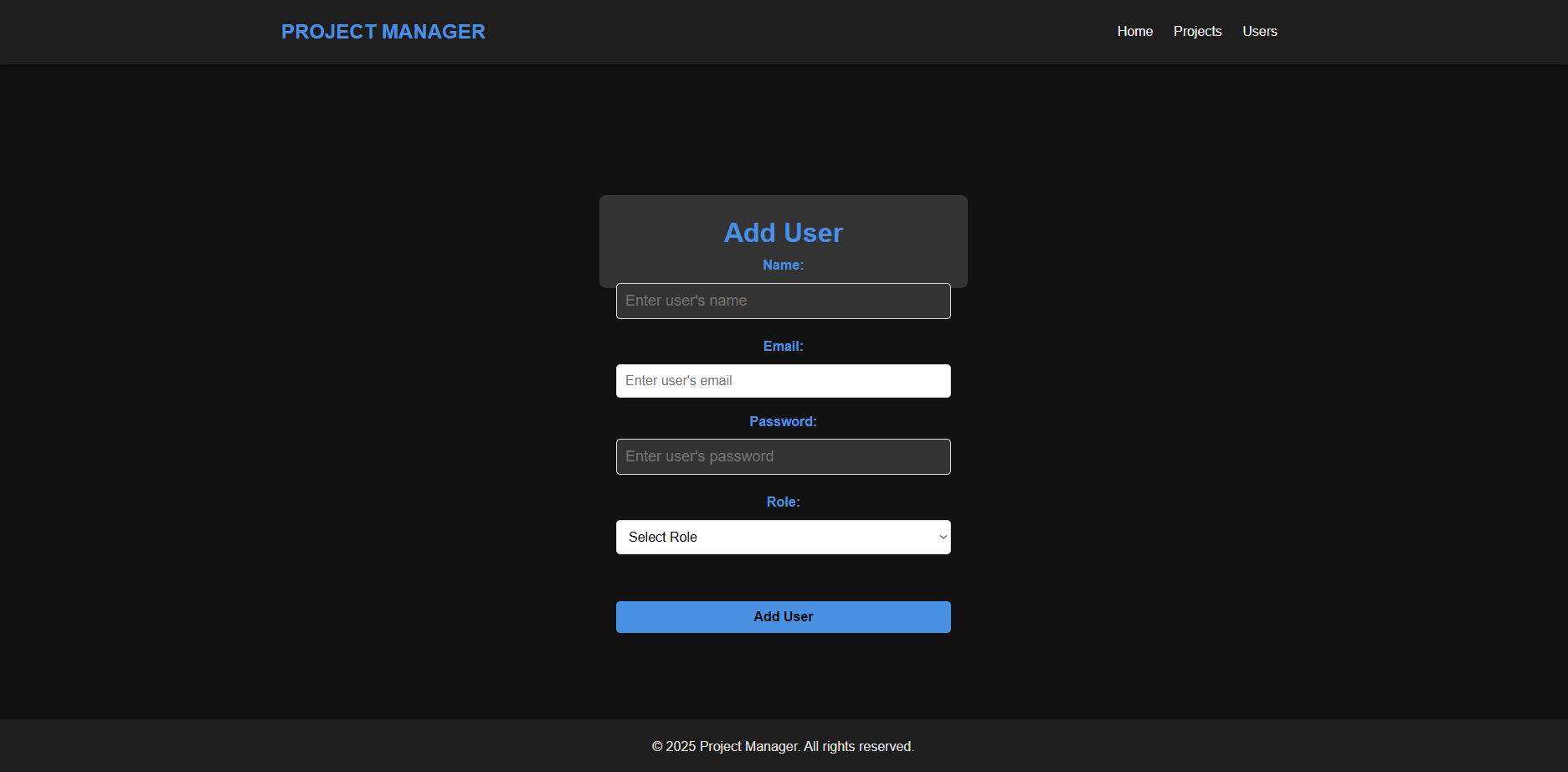
**Opis:**  
 Strona zawiera formularz do dodania nowego użytkownika do systemu. Pozwala wprowadzić dane użytkownika takie jak imię, e-mail, hasło oraz przypisać rolę. Formularz waliduje wymagane pola przed wysłaniem.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| <form id="addUserForm" class="addForm"> | Formularz do wprowadzania danych nowego użytkownika. |
| #name | Pole tekstowe na imię i nazwisko użytkownika. |
| #email | Pole na adres e-mail użytkownika. |
| #password | Pole do ustawienia hasła użytkownika. |
| #role | Lista rozwijana do wyboru roli użytkownika (Admin, Developer, Manager). |
| button.btn[type="submit"] | Przycisk wysyłający formularz. |
| <footer> | Stopka strony z prawami autorskimi. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja formularza i strony.
* JS: JS/add\_user.js (moduł ES6, załadowany z defer) – obsługuje walidację formularza i logikę dodawania użytkownika.



## **9. 📄 users\_update.html – Edycja danych użytkownika**

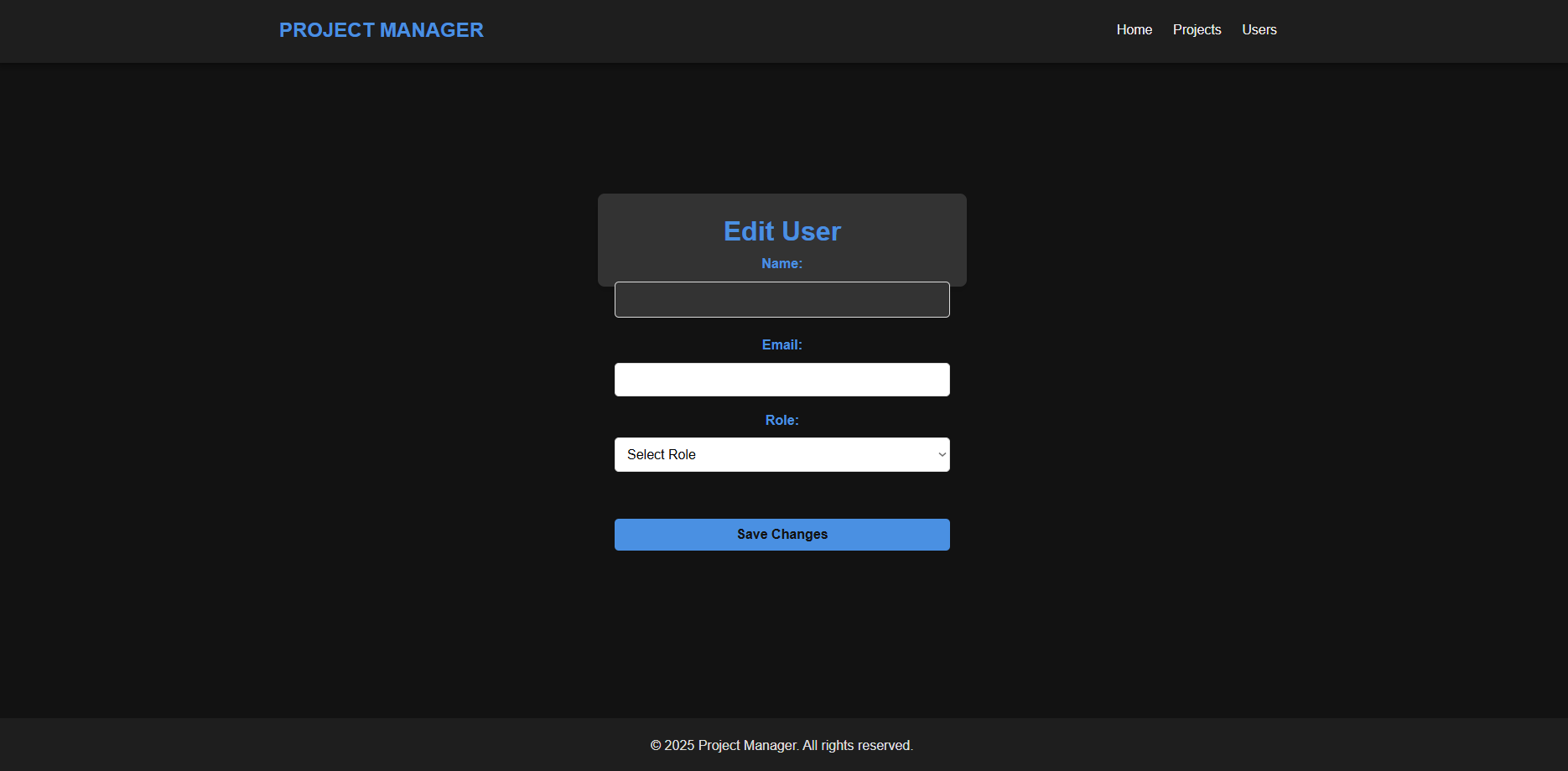
**Opis:**  
 Strona umożliwia edycję danych istniejącego użytkownika. Formularz pozwala zmienić imię, e-mail oraz rolę użytkownika. Dane są wczytywane i zapisywane za pomocą skryptu update\_user.js. Pola imię i e-mail są opcjonalne, natomiast wybór roli jest wymagany.

**Kluczowe elementy:**

|  |  |
| --- | --- |
| **Element** | **Opis** |
| .nav-item.active | Link do aktualnej strony (Users) oznaczony jako aktywny. |
| <form id="editUserForm" class="addForm"> | Formularz do edycji danych użytkownika. |
| #name | Pole tekstowe na imię i nazwisko użytkownika. |
| #email | Pole na adres e-mail użytkownika. |
| #role | Lista rozwijana do wyboru roli użytkownika (Admin, Developer, Manager). Pole wymagane. |
| button.btn[type="submit"] | Przycisk do zapisania zmian. |
| <footer> | Stopka strony z prawami autorskimi. |

**Zależności:**

* CSS: CSS/styles.css – stylizacja formularza i strony.
* JS: JS/update\_user.js (moduł ES6, załadowany z defer) – obsługuje wczytanie danych użytkownika, walidację i zapis zmian.



## **Pliki JS:**

## **1. 📄 add\_project.js – Obsługa dodawania nowego projektu**

**Opis:**  
 Skrypt obsługuje formularz dodawania projektu (projectForm). Po wczytaniu strony nasłuchuje na zdarzenie submit formularza, następnie waliduje dane (wywołując funkcję validate z log\_validation.js), pobiera wartości z pól i wysyła je metodą POST do API pod adresem addProjectUrl. Po pomyślnym dodaniu projektu formularz jest resetowany, a użytkownik przekierowywany jest do listy projektów.

**Kluczowe elementy:**

* Nasłuchiwanie na submit formularza o id projectForm.
* Pobranie wartości: name, description, startDate, endDate, status, priority.
* Wywołanie fetch z nagłówkami, w tym tokenem TokenId pobieranym z sessionStorage.
* Obsługa odpowiedzi i błędów, wyświetlanie alertów.

**Zależności:**

* urls.js – eksportuje addProjectUrl.
* log\_validation.js – funkcja validate() do walidacji formularza.

## **2. 📄 add\_user.js – Dodawanie nowego użytkownika**

**Opis:**  
 Skrypt obsługuje formularz dodawania użytkownika (addUserForm). Po wysłaniu formularza zbiera dane, tworzy obiekt użytkownika, i wysyła go metodą POST pod URL addUserUrl. Po sukcesie resetuje formularz i przekierowuje do listy użytkowników. W przypadku błędu wyświetla stosowny komunikat.

**Kluczowe elementy:**

* Funkcja addUser wywoływana przy submit formularza o id addUserForm.
* Pobranie pól: name, email, password, role.
* Wysłanie zapytania fetch z nagłówkami (w tym TokenId z sessionStorage).
* Obsługa odpowiedzi i błędów z alertami i konsolą.

**Zależności:**

* urls.js – eksportuje addUserUrl.

## **3. 📄 button\_reaction.js – Obsługa przycisku „More info” oraz weryfikacja sesji**

**Opis:**  
 Skrypt odpowiada za weryfikację sesji użytkownika (wywołuje funkcję validate z log\_validation.js) oraz obsługę przycisku „More info” (moreInfoBtn). Po kliknięciu przełącza widoczność dodatkowego tekstu i zmianę rozmiaru obrazka (heroImage). Tekst przycisku zmienia się dynamicznie na „Hide info” lub „More info”.

**Kluczowe elementy:**

* Wywołanie funkcji validate() przy załadowaniu dokumentu.
* Elementy: moreInfoBtn (przycisk), moreText (tekst do pokazania/ukrycia), heroImage (element obrazu).
* Nasłuchiwanie na kliknięcie przycisku i togglowanie klas CSS.
* Dynamiczna zmiana tekstu przycisku.

**Zależności:**

* log\_validation.js – funkcja validate() do sprawdzania poprawności sesji.

## **4. 📄 log.js – Obsługa logowania użytkownika**

**Opis:**  
 Skrypt obsługuje przycisk logowania (loginButton). Po kliknięciu blokuje domyślne zachowanie i wywołuje funkcję validateAndLogin z log\_validation.js, która zajmuje się weryfikacją danych logowania i pobieraniem tokena sesji.

**Kluczowe elementy:**

* Nasłuchiwanie na kliknięcie przycisku o id loginButton.
* Wywołanie validateAndLogin() z log\_validation.js w celu przetworzenia logowania.

**Zależności:**

* log\_validation.js – funkcja validateAndLogin() realizująca logikę logowania i pobierania tokena.

## **5. 📄 log\_validation.js – Walidacja sesji i logowanie użytkownika**

**Opis:**  
 Moduł zawiera dwie główne funkcje:

* validate() – sprawdza ważność tokena sesji przechowywanego w sessionStorage poprzez wysłanie POST na validateUrl. Jeśli sesja jest nieważna, wyświetla alert i przekierowuje do strony logowania (log.html).
* validateAndLogin() – pobiera dane z formularza logowania, wysyła je do API (getTokenUrl) i w przypadku sukcesu zapisuje token do sessionStorage oraz przekierowuje do strony głównej. W przeciwnym razie wyświetla odpowiednie komunikaty o błędach.

**Kluczowe elementy:**

* validate() – walidacja sesji użytkownika na podstawie tokena.
* validateAndLogin() – logowanie i pobieranie tokena sesji.
* Obsługa błędów i przekierowania.

**Zależności:**

* urls.js – eksportuje validateUrl oraz getTokenUrl.

## **6. 📄 project.js – Zarządzanie listą projektów (pobieranie, wyświetlanie, usuwanie, edycja)**

**Opis:**  
 Skrypt odpowiada za pobieranie wszystkich projektów z serwera (getAllProjectsUrl), wyświetlanie ich na stronie oraz obsługę usuwania i edycji projektów. Przy ładowaniu strony wywołuje funkcję validate() w celu sprawdzenia ważności sesji. Dodaje też przycisk do dodawania nowego projektu, który kieruje do formularza dodawania.

**Kluczowe elementy:**

* handleProjects() – wywołuje validate() i pobiera projekty.
* fetchProjects() – pobiera dane projektów przez GET z nagłówkiem TokenId.
* displayProjects() – dynamiczne tworzenie elementów HTML z danymi projektów i przyciskami Edytuj/Usuń.
* Obsługa przycisków:
  + Usuwanie projektu (fetch na deleteProjectUrl).
  + Przekierowanie do strony edycji projektu (update\_projects.html?id=...).
* Dodanie sekcji „Add new project” z linkiem do formularza dodawania.

## **7. 📄 register.js – Rejestracja nowego użytkownika**

**Opis:**  
 Skrypt obsługuje formularz rejestracji użytkownika, waliduje hasła (muszą być niepuste i zgodne), wysyła dane na serwer (registerUrl) i informuje o powodzeniu lub błędach rejestracji. Dodaje dynamiczny komunikat o błędzie pod polem potwierdzenia hasła.

**Kluczowe elementy:**

* Walidacja zgodności i obecności hasła i potwierdzenia hasła.
* Wysłanie POST z danymi: name, email, password.
* Obsługa odpowiedzi serwera (alerty o sukcesie lub błędzie).
* Dynamiczne dodanie elementu passwordError pod polem confirmPassword dla komunikatów błędów.
* Przypięcie handlera do przycisku registerButton.

## **8. 📄 show\_password.js – Przełącznik widoczności haseł w formularzach**

**Opis:**  
 Skrypt umożliwia przełączanie widoczności pól hasła na formularzach. Każdy przycisk z klasą .toggle-password posiada atrybut data-target, który wskazuje id pola hasła do przełączenia. Po kliknięciu zmienia typ inputa z password na text i na odwrót oraz dodaje klasę showing do przycisku.

**Kluczowe elementy:**

* Nasłuchiwanie kliknięć na przyciskach .toggle-password.
* Pobieranie docelowego inputa na podstawie data-target.
* Zmiana typu pola (password <-> text).
* Dodanie/Usunięcie klasy showing dla zmiany wyglądu przycisku.

## **9. 📄 update\_projects.js – Edycja projektu**

**Opis:**  
 Skrypt pobiera dane projektu po ID z URL i wypełnia formularz edycji. Umożliwia aktualizację projektu poprzez wysłanie zmienionych danych metodą POST do endpointu aktualizacji. Po udanej aktualizacji przekierowuje na listę projektów.

**Kluczowe elementy:**

* Formatowanie dat do formatu YYYY-MM-DD.
* Pobieranie ID projektu z URL i wczytywanie danych do formularza.
* Nasłuchiwanie submitu formularza i wysyłanie aktualizacji metodą POST.
* Obsługa odpowiedzi serwera i przekierowanie po sukcesie.

## **10. 📄 update\_user.js – Edycja użytkownika**

**Opis:**  
 Skrypt pobiera dane użytkownika po ID z URL i wypełnia formularz edycji. Pozwala na aktualizację danych użytkownika przez wysłanie zmienionych danych metodą POST z tokenem autoryzacyjnym. Po pomyślnym update przekierowuje do listy użytkowników.

**Kluczowe elementy:**

* Wywołanie funkcji walidującej token przy ładowaniu strony.
* Pobieranie danych użytkownika z API z nagłówkiem TokenId.
* Wypełnianie formularza danymi użytkownika.
* Nasłuchiwanie submitu formularza i wysyłanie aktualizacji z tokenem w nagłówku.
* Obsługa odpowiedzi i przekierowanie po sukcesie.

## **11. 📄 urls.js – Endpointy API**

**Opis:**  
 Plik eksportuje stałe URL-e do wywołań API dotyczących użytkowników, projektów oraz autoryzacji i rejestracji.

**Kluczowe elementy:**

* Endpointy do zarządzania użytkownikami (dodawanie, pobieranie, usuwanie, aktualizacja).
* Endpointy do zarządzania projektami (dodawanie, pobieranie, usuwanie, aktualizacja).
* Endpointy związane z autoryzacją i rejestracją użytkowników.

## **12. 📄 users.js – Zarządzanie listą użytkowników**

**Opis:**  
 Skrypt pobiera listę użytkowników z serwera i wyświetla ją na stronie. Umożliwia usuwanie i edytowanie użytkowników oraz dodanie nowego użytkownika poprzez przycisk kierujący do formularza rejestracji.

**Kluczowe elementy:**

* Wywołanie funkcji walidującej token przy ładowaniu strony.
* Pobieranie listy użytkowników z API z nagłówkiem TokenId.
* Obsługa błędów, w tym przekierowanie przy statusie 401 (nieautoryzowany).
* Generowanie elementów HTML z danymi użytkowników (imię, email, rola).
* Dodanie event listenerów do przycisków Edytuj i Usuń.
* Dodanie sekcji „Add new user” z linkiem do formularza rejestracji.

## **Plik CSS:**

## **1. 📄 style.css – Stylowanie interfejsu użytkownika**

**Opis:**  
 Arkusz stylów definiuje wygląd całej aplikacji, od resetu domyślnych marginesów i paddingów, przez stylizację nagłówka, sekcji hero, funkcji, formularzy, aż po responsywność dla urządzeń mobilnych. Utrzymany jest ciemny, nowoczesny motyw z akcentami niebieskiego (#4a90e2).

**Kluczowe elementy:**

* Reset stylów dla podstawowych elementów (body, nagłówki, paragrafy, linki) z box-sizing: border-box.
* Stylizacja body z ciemnym tłem (#121212) i jasnym tekstem (#f0f0f0), układ flex-column z min-height 100vh.
* Header z tłem #1e1e1e, pozycja sticky na górze, z cieniem i centralnym ułożeniem elementów.
* Sekcja hero z dwukolumnowym układem: tekst (60%) i obraz (40%), z responsywnym dostosowaniem do wąskich ekranów.
* Stylizacja przycisków .btn, .btn.secondary z zaokrąglonymi rogami i efektami hover.
* Sekcja features jako grid 2 kolumnowy z kartami cech, utrzymany w ciemnej kolorystyce i z niebieskimi nagłówkami.
* Formularze i elementy input mają spójny wygląd: ciemne tło, jasny tekst, zaokrąglone rogi, pełna szerokość i przestrzeń wewnętrzna.
* Klasy dla przycisków rejestracji/logowania .button1 i .button2 z kolorami i efektami hover.
* Stylizacja widoczności hasła: przycisk .toggle-password z ikonami, pokazującymi i ukrywającymi hasło.
* Responsywne poprawki dla ekranów do 768px: formularze i sekcje hero przechodzą do układu kolumnowego i pełnej szerokości.
* Dodatkowe efekty przejść, np. rozwijane teksty .more-text oraz powiększanie obrazów w .hero-image.expanded img.