

AI1	Dokumentacja projektu
Autor	Sebastian Król, 132537
Kierunek, rok	Informatyka i ekonometria, II rok, st. stacjonarne (3,5-l)
Temat projektu	<i>Kalkulator PIT</i>

Spis treści

1. Wstęp	3
2. Wymagania.....	3
3. Narzędzia i technologie	4
3.1. Języki programowania	4
3.2. Framework.....	5
3.3. Baza danych.....	5
3.4. Biblioteki PHP z Composer	5
3.5. Frontend i stylowanie	5
3.6. Środowisko programistyczne	5
4. Baza Danych	6
4.1. Diagram ERD.....	6
4.2. Opis bazy danych – pit_calculator	6
4.3. Relacje pomiędzy tabelami.....	7
5. GUI – prezentacja	8
5.1. Panel Administratora.....	8
5.2. Główne funkcje i elementy panelu:.....	8
5.3. Zarządzanie użytkownikami	10
5.4. Elementy interfejsu:	10
5.5. Responsywność	11
6. Uruchomienie aplikacji	12
6.1. Wymagania systemowe i oprogramowanie	12
6.2. Kroki konfiguracyjne – pierwsze uruchomienie aplikacji	13
Po uruchomieniu	14
6.3. Automatyczna konfiguracja	14
6.4. Dane logowania	14
7. Funkcjonalności aplikacji	14
7.1. Proces rejestracji i logowania użytkownika.....	15
7.2. Przykładowy CRUD przeprowadzany przez użytkownika	16
7.3. Zarządzanie użytkownikami przez administratora	17
7.4. Przeglądanie ogólnodostępnych zasobów (bez logowania).....	18
7.5. Zarządzanie swoimi kalkulacjami przez użytkownika.....	19
7.6. Zarządzanie swoimi danymi przez użytkownika.....	20
7.7. Walidacja danych – przykłady	21
8. Podsumowanie	29

1. Wstęp

Tematem mojego projektu jest „**System kalkulacji podatkowej PIT**”. W założeniu miała powstać aplikacja webowa usprawniająca obliczanie podatku dochodowego od osób fizycznych na podstawie danych finansowych użytkownika.

System odwzorowuje podstawowe procesy związane z rozliczaniem podatku PIT w Polsce. Głównym założeniem projektu jest umożliwienie użytkownikom szybkiego i prostego obliczenia należnego podatku oraz zapisu historii przeprowadzonych kalkulacji. Aplikacja została również wyposażona w panel administratora oraz tryb demonstracyjny (demo), pozwalający na przetestowanie systemu bez konieczności zakładania konta.

Użytkownicy aplikacji po zalogowaniu uzyskują dostęp do następujących funkcjonalności:

- Wprowadzanie danych finansowych (np. dochody, koszty uzyskania przychodu, ulgi)
- Automatyczne obliczanie wysokości podatku
- Przeglądanie historii swoich kalkulacji
- Edytowanie profilu użytkownika
- Możliwość wylogowania się lub zmiany hasła

Panel administratora zapewnia dostęp do danych statystycznych oraz historii kalkulacji wykonanych przez użytkowników. Administrator może także zarządzać kontami użytkowników i dbać o prawidłowe działanie systemu.

Projekt zakłada również możliwość wygenerowania przykładowych danych z wykorzystaniem seederów i factory, co wspomaga testowanie funkcji aplikacji w warunkach zbliżonych do rzeczywistych.

Aplikacja wspiera użytkowników w podejmowaniu decyzji podatkowych oraz pozwala im na bezpieczne i szybkie obliczenie PIT bez potrzeby korzystania z zewnętrznych kalkulatorów czy usług księgowych. Dodatkowo, system umożliwia sprawne zarządzanie kontami oraz historią obliczeń, co zwiększa transparentność i użyteczność całego rozwiązania.

2. Wymagania

Od strony technologicznej aplikacja działa na Laravelu w wersji $\geq 12.x$, wykorzystuje zarówno **MySQL** z kluczami obcymi, jak i **seedy** oraz **migracje** do przygotowania przykładowych danych. Baza danych zawiera tabele powiązane z użytkownikami, kalkulacjami PIT, źródłami dochodów oraz historią zmian. System wspiera zapisy transakcyjne, zapewniające spójność danych przy wykonywaniu obliczeń podatkowych.

W zakresie funkcjonalności:

- **Tworzenie kont i ról:** aplikacja umożliwia rejestrację i logowanie użytkowników oraz definiuje uprawnienia dla administratora. Administrator posiada pełen dostęp do zarządzania użytkownikami i ich kalkulacjami. Użytkownicy mogą edytować swoje dane (nazwa, e-mail, hasło, zdjęcie) oraz przeglądać własną historię kalkulacji.
- **Kalkulacje i zarządzanie danymi:** użytkownik może dodawać dane finansowe (dochody, ulgi, koszty), a następnie przeprowadzić kalkulację PIT. Kalkulacje są zapisywane w historii z podziałem na użytkownika i datę wykonania. Administrator ma wgląd do wszystkich kalkulacji.
- **UI/UX:** interfejs wykorzystuje komponenty typu toggle (np. widoczność danych), formularze z walidacją (date-pickery do wyboru roku podatkowego), paginację list wyników i statystyk, interaktywne tabele oraz przejrzysty layout dopasowany do przeglądarek opartych na Chromium i Firefox.
- **Walidacja i zabezpieczenia:** walidacja danych odbywa się zarówno po stronie frontendu (HTML5 + JavaScript), jak i backendu (Form Requesty i reguły Laravela). Zastosowano ochronę CSRF, system autoryzacji oparty na middleware i policy, poprawne kody błędów HTTP oraz dedykowane widoki błędów. Hasła są haszowane, a zdjęcia profilowe użytkowników walidowane i przechowywane zgodnie z zasadami Laravela (symlink storage:link).
- **Asynchroniczność i automatyzacja:** aplikacja korzysta z kolejki (queue:work) do obsługi zadań w tle (np. powiadomień) oraz z harmonogramu (schedule:work) do okresowego czyszczenia danych testowych i innych zaplanowanych zadań. Umożliwia to płynne działanie bez zakłóceń dla użytkownika.
- **DevOps:** aplikacja zawiera kompletny skrypt start.bat, który umożliwia szybkie uruchomienie środowiska deweloperskiego – od utworzenia bazy danych, przez instalację zależności (composer, npm), aż po migracje, seedy i uruchomienie niezbędnych komponentów (worker, scheduler, serwer dev).

3. Narzędzia i technologie

Poniżej przedstawiam zestawienie technologii wykorzystanych w projekcie **System kalkulacji podatkowej PIT**:

3.1. Języki programowania

- **PHP** **8.2+**
Główny język backendowy wykorzystywany do realizacji logiki aplikacji, przetwarzania danych podatkowych oraz integracji z bazą danych.
Dokumentacja: <https://www.php.net/docs.php>
Licencja: PHP License, darmowy
- **JavaScript**
Wykorzystywany w aplikacji do podstawowych funkcjonalności po stronie klienta – np. walidacji formularzy czy prostych interakcji.
Dokumentacja: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
Licencja: ECMAScript, darmowy

3.2. Framework

- **Laravel** **12.x**
Nowoczesny framework PHP, który zapewnia strukturę MVC, obsługę tras, migracji, walidacji danych, autoryzacji oraz narzędzi developerskich.
Dokumentacja: <https://laravel.com/docs/12.x>
Licencja: MIT, darmowy

3.3. Baza danych

- **MySQL**
Relacyjna baza danych służąca do przechowywania użytkowników, kalkulacji PIT, źródeł dochodu oraz historii zmian.
Dokumentacja: <https://dev.mysql.com/doc/>
Licencja: GPL v2 (darmowy)

3.4. Biblioteki PHP z Composer

- **Laravel** **Tinker**
Interaktywna konsola do testowania kodu i manipulowania danymi aplikacji.
<https://github.com/laravel/tinker>
- **Faker** + **Factory** + **Seeder**
Narzędzia Larela służące do generowania przykładowych danych użytkowników, kalkulacji i historii, przydatne w fazie testów i prezentacji systemu.

Wszystkie wymienione biblioteki są **open-source** i darmowe.

3.5. Frontend i stylowanie

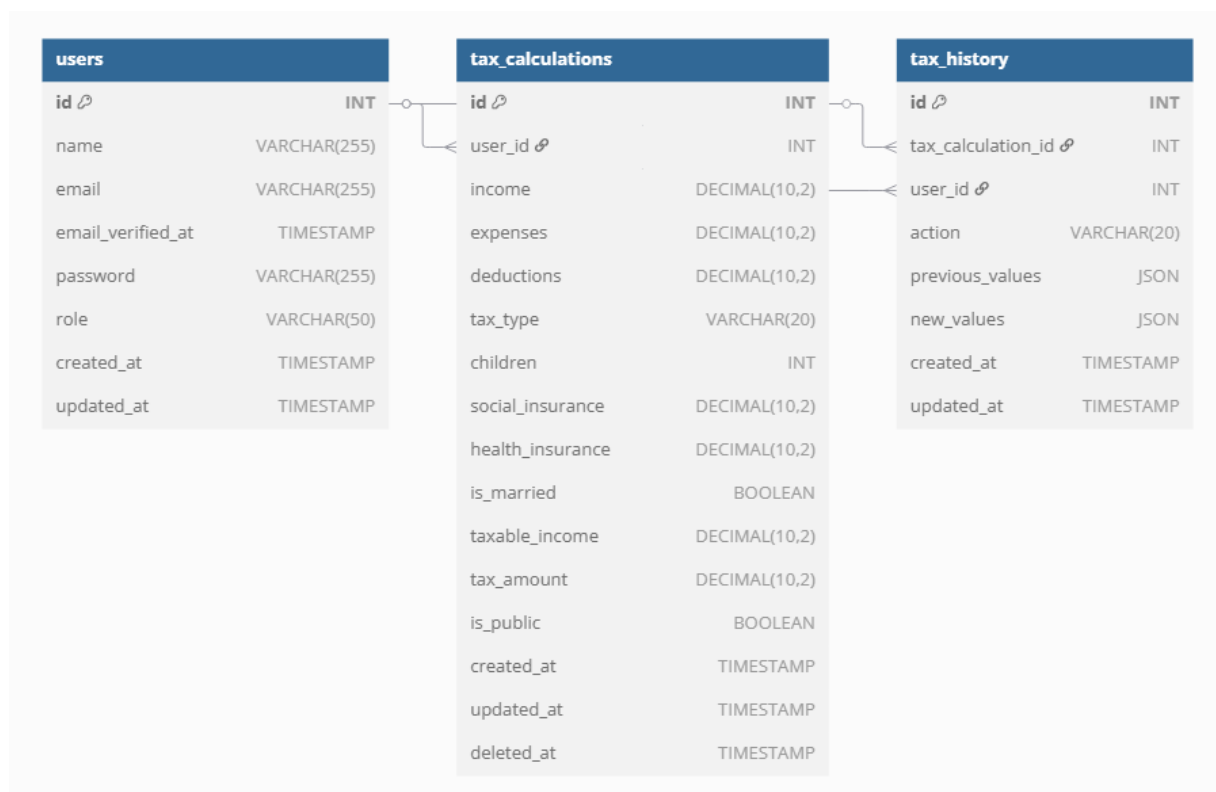
- **Własny CSS** oraz **komponenty Blade (Laravel)**
Interfejs użytkownika został opracowany przy użyciu standardowego HTML i CSS oraz szablonów Blade – bez wykorzystania zewnętrznych frameworków CSS.

3.6. Środowisko programistyczne

- **Visual Studio** **Code**
Edytor kodu wykorzystywany do tworzenia aplikacji.
<https://code.visualstudio.com>
Licencja: Freeware
- **XAMPP**
Lokalne środowisko programistyczne (Apache, MySQL, PHP) używane do uruchamiania projektu podczas developmentu.
<https://www.apachefriends.org>
Licencja: GNU

4. Baza Danych

4.1. Diagram ERD



4.2. Opis bazy danych – pit_calculator

Baza danych została zaprojektowana w sposób relacyjny, aby umożliwić bezpieczne i przejrzyste przechowywanie informacji niezbędnych do obliczania podatku dochodowego (PIT) oraz zarządzania użytkownikami systemu. Główne elementy aplikacji — użytkownicy, kalkulacje podatkowe oraz historia zmian — zostały zaimplementowane jako odrębne tabele powiązane między sobą za pomocą kluczy obcych.

Tabela users (Użytkownicy)

Tabela przechowuje dane wszystkich zarejestrowanych użytkowników systemu. Zawiera takie informacje jak:

- imię i nazwisko,
- adres e-mail (unikalny),
- hasło (zahaszkowane),
- znaczniki czasowe (created_at, updated_at).

Użytkownik może wykonywać obliczenia podatkowe, przeglądać historię oraz edytować dane swojego konta.

Tabela `tax_calculations` zawiera dane pojedynczych kalkulacji wykonanych przez użytkowników. Każda kalkulacja obejmuje:

- dochód (`income`),
- koszty uzyskania przychodu (`costs`),
- wyliczoną wartość podatku (`tax`),
- rok podatkowy (`year`),
- powiązanie z użytkownikiem (`user_id` – opcjonalnie),
- znacznik `is_public` – informujący, czy kalkulacja jest dostępna w trybie demo.

Tabela ta stanowi główny obszar funkcjonalny systemu, odpowiadający za rejestrację i wyświetlanie obliczeń.

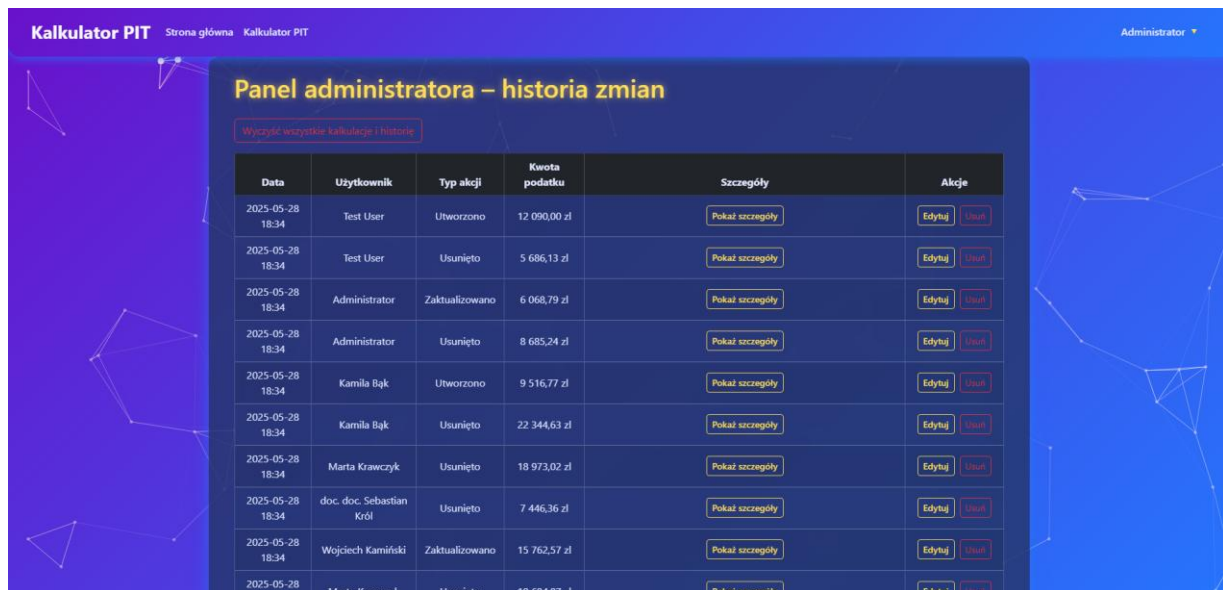
Tabela `tax_history` przechowuje historię modyfikacji kalkulacji PIT. Zawiera informacje o wcześniejszych wartościach `income` i `costs`, które zostały zmienione. Każdy wpis jest powiązany z konkretną kalkulacją (`tax_calculation_id`) i zapisywany w momencie edycji. Tabela Rezerwacje (`reservations`)

4.3. Relacje pomiędzy tabelami

- **Jeden użytkownik może wykonać wiele kalkulacji PIT**
Relacja 1:N między `users` a `tax_calculations` (`user_id` może być null w wersji demo).
- **Jedna kalkulacja może mieć wiele wersji historycznych**
Relacja 1:N między `tax_calculations` a `tax_history`.
- **Historia kalkulacji jest przypisana do jednej kalkulacji PIT**
Dzięki temu możliwe jest wersjonowanie zmian dla konkretnego rekordu.

5. GUI – prezentacja

5.1. Panel Administratora



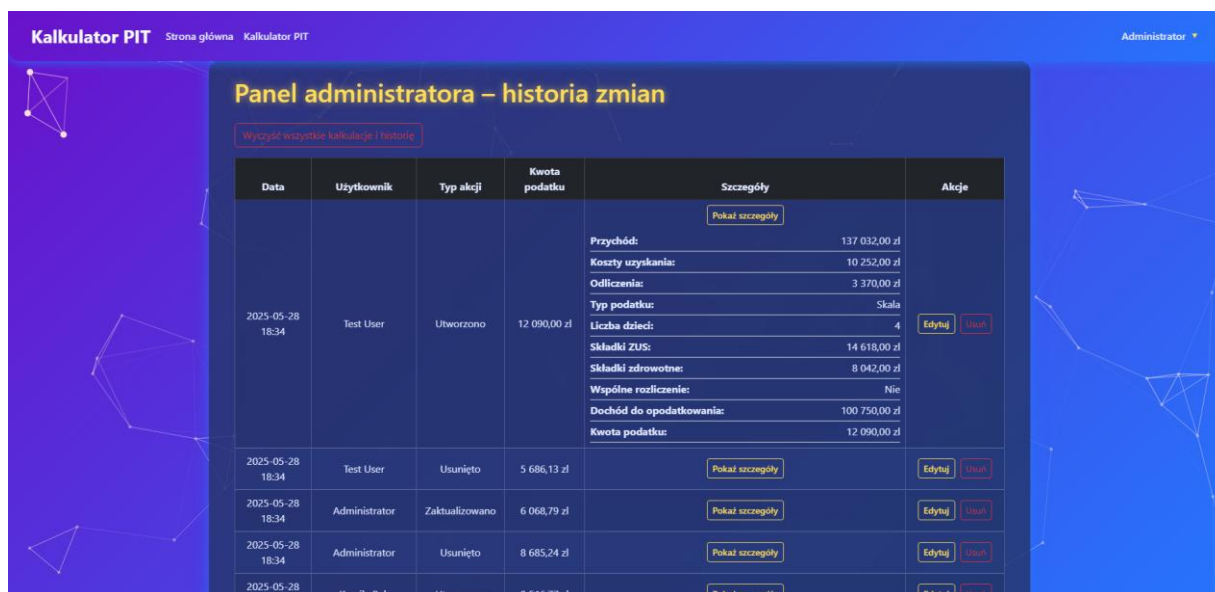
Rysunek 1 Przykładowe GUI

Widok przedstawia **panel administratora** aplikacji PIT, zawierający kluczowe narzędzia i informacje umożliwiające sprawne zarządzanie użytkownikami i historią obliczeń. Interfejs użytkownika utrzymany jest w nowoczesnej stylistyce z gradientowym tłem oraz responsywnym układem.

5.2 Główne funkcje i elementy panelu:

1. **Nagłówek** – zawiera nazwę aplikacji oraz menu nawigacyjne:
 - o odnośniki do „Strona główna” i „Kalkulator PIT”,
 - o rozwijane menu z nazwą aktualnie zalogowanego administratora (z możliwością przejścia do ustawień konta lub wylogowania).
2. **Tytuł sekcji** – wyświetlany jest nagłówek: „Panel administratora – historia zmian”, który informuje o bieżącym kontekście administracyjnym.
3. **Przycisk czyszczący historię** – przycisk „Wyczyść wszystkie kalkulacje i historię”, widoczny nad tabelą, umożliwia administratorowi usunięcie całej historii obliczeń w bazie danych (akcja potwierdzana przez użytkownika).
4. **Tabela historii kalkulacji** – zawiera szczegółową listę zdarzeń związanych z działaniami użytkowników:

Kolumna	Opis
Data	Data i godzina wykonania akcji
Użytkownik	Nazwa użytkownika wykonującego akcję
Typ akcji	Rodzaj operacji (Utworzono, Zaktualizowano, Usunięto)
Kwota podatku	Obliczona wartość podatku PIT w danym wpisie
Szczegóły	Przycisk do wyświetlenia danych kalkulacji
Akcje	Dwa przyciski: Edytuj oraz Usuń dla każdej pozycji historii



Rysunek 2 Przykładowe GUI

Po kliknięciu przycisku „**Pokaż szczegóły**” w tabeli historii zmian administrator może zobaczyć pełen zestaw danych wprowadzonych przez użytkownika oraz wynik kalkulacji podatkowej. Funkcja ta umożliwia szybki audyt poprawności danych i ocenę podstawy obliczeń bez konieczności przechodzenia do osobnego widoku.

Po rozwinięciu szczegółów, dla każdej kalkulacji wyświetlane są:

Pole	Opis
Przychód	Całkowita kwota przychodu użytkownika
Koszty uzyskania	Koszty uzyskania przychodu (np. koszty pracy)
Odliczenia	Kwoty ulg i odliczeń (np. darowizny, IKZE)
Typ podatku	Forma rozliczenia: Skala / Podatek liniowy
Liczba dzieci	Liczba dzieci wykazana w rozliczeniu
Składki ZUS	Kwota składek emerytalnych i rentowych
Składki zdrowotne	Wartość składek zdrowotnych
Wspólne rozliczenie	Informacja, czy użytkownik rozlicza się wspólnie z małżonkiem

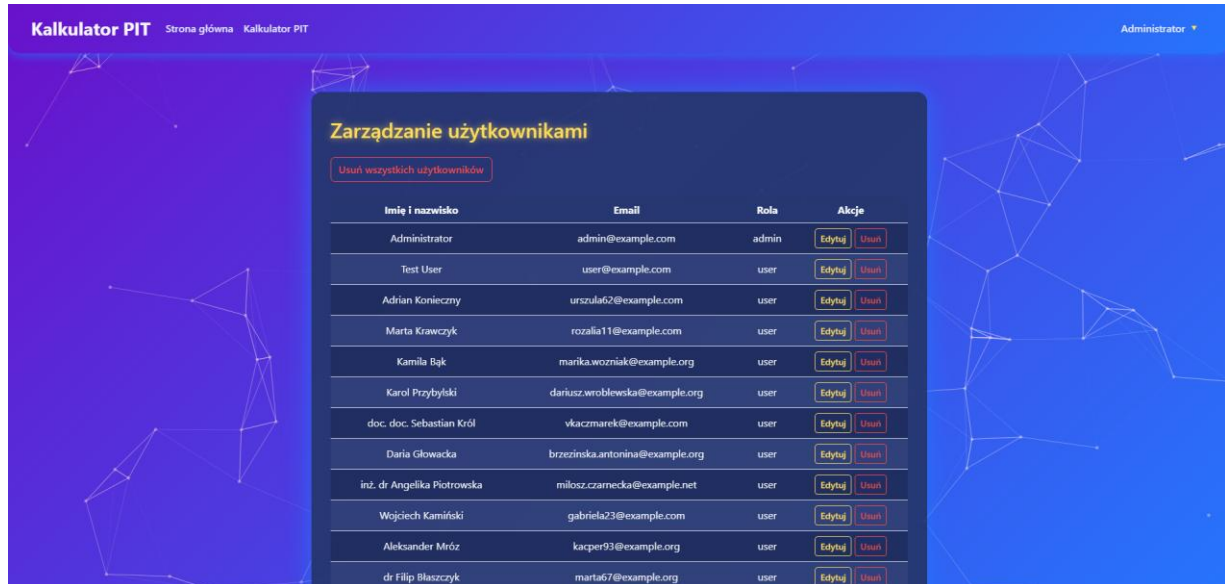
Pole

Opis

Dochód do opodatkowania Dochód po uwzględnieniu odliczeń

Kwota podatku Ostatecznie obliczona należność PIT

5.3. Zarządzanie użytkownikami



Rysunek 3 Przykładowe GUI

Widok ten przedstawia **panel administratora** umożliwiający przegląd oraz zarządzanie wszystkimi kontami użytkowników zarejestrowanych w systemie.

5.4 Elementy interfejsu:

Na górze widoczny jest tytuł: „**Zarządzanie użytkownikami**”, który jednoznacznie informuje o przeznaczeniu strony. Poniżej umieszczony jest przycisk:

- „**Usuń wszystkich użytkowników**” – umożliwia masowe usunięcie kont (po potwierdzeniu akcji).

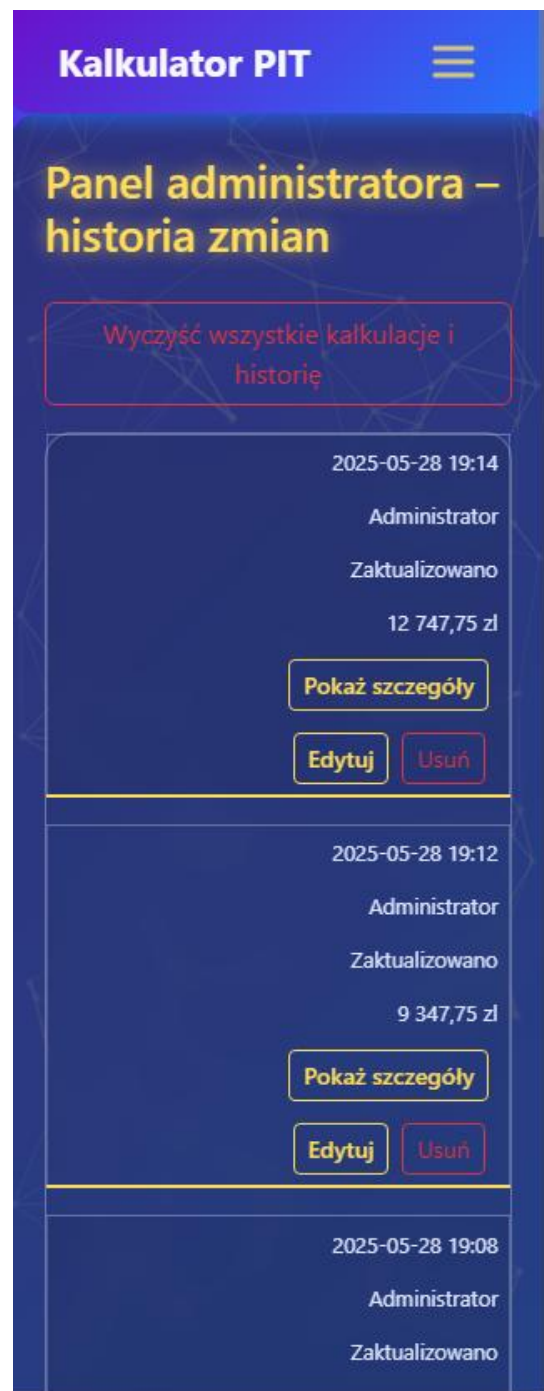
Lista użytkowników zaprezentowana jest w formie tabeli z następującymi kolumnami:

Kolumna	Opis
Imię i nazwisko	Pełna nazwa użytkownika
Email	Adres e-mail (login)
Rola	Rola w systemie: admin lub user
Akcje	Dwa przyciski: Edytuj i Usuń

- Przycisk **Edytuj** umożliwia administratorowi przejście do formularza edycji użytkownika.

- Przycisk **Usuń** pozwala na trwałe usunięcie użytkownika po potwierdzeniu.

5.5 Responsywność



Aplikacja została zaprojektowana w sposób responsywny, co oznacza, że automatycznie dostosowuje się do rozmiaru ekranu urządzenia, na którym jest wyświetlana – zarówno komputerów, jak i urządzeń mobilnych (smartfonów, tabletów).

Na urządzeniach mobilnych menu główne zostaje ukryte i zastąpione intuicyjnym przyciskiem typu „hamburger”. Po jego kliknięciu rozwija się lista opcji nawigacyjnych, umożliwiających szybki dostęp do głównych funkcjonalności aplikacji – m.in. kalkulatora PIT, strony głównej i panelu administratora.

Widoki dla urządzeń mobilnych zostały zoptymalizowane tak, aby:

- tabele automatycznie przewijały się poziomo na małych ekranach,
- dane były przedstawione w czytelny sposób – z dopasowaną wielkością czcionek, przycisków oraz marginesów,
- układ sekcji był pionowy, co ułatwia obsługę jedną ręką.

W celu weryfikacji responsywności przeprowadzono testy na widokach zarówno **dla użytkownika**, jak i **administratora**.

Dla użytkownika – strona główna oraz kalkulator podatkowy prezentują się w uproszczonym, zwartym układzie. Wyświetlane są ostatnie anonimowe kalkulacje, a użytkownik może w prosty sposób uruchomić kalkulację podatku poprzez widoczny przycisk „**Oblicz podatek**”.

Dla administratora – widok panelu historii zmian został dostosowany do ekranów mobilnych. Lista operacji (dodanie, edycja, usunięcie kalkulacji) jest wyświetlana w formie kart z przyciskami akcji („Pokaż szczegóły”, „Edytuj”, „Usuń”), które są czytelne i łatwe do kliknięcia.

Responsywność stanowi kluczowy element projektu aplikacji, zapewniając wygodę korzystania z systemu niezależnie od urządzenia. Dzięki zastosowaniu technik takich jak @media, flexbox, table-responsive, aplikacja zachowuje funkcjonalność i estetykę w każdej rozdzielczości.

6. Uruchomienie aplikacji

Aby uruchomić aplikację **Kalkulator PIT** lokalnie, należy spełnić określone wymagania systemowe oraz wykonać następujące kroki instalacyjne.

6.1. Wymagania systemowe i oprogramowanie

Do poprawnego działania aplikacji potrzebne są:

- **PHP** w wersji min. **8.2**
- **Composer** – do zarządzania zależnościami PHP
- **MySQL** lub **MariaDB** – jako silnik bazy danych
- **Laravel** (instalowany automatycznie przez Composer)
- **XAMPP** lub inne środowisko z serwerem Apache i bazą danych
- **Visual Studio Code** (lub inne IDE) – do edycji kodu
- (Opcjonalnie) **Git** – jeśli pracujesz z repozytorium

6.2. Kroki konfiguracyjne – pierwsze uruchomienie aplikacji

1. **Rozpakuj projekt ZIP**
Użyj programu np. **7-Zip** lub wbudowanego eksploratora Windows.

2. **Otwórz folder projektu w edytorze**
Przejdź do katalogu projektu np. za pomocą **Visual Studio Code** lub terminala.

3. **Zainstaluj zależności PHP (Laravel):**

```
bash
KopiujEdytuj
composer install
```

4. **Utwórz symlink do folderu storage:**

```
bash
KopiujEdytuj
php artisan storage:link
```

5. **Skopiuj plik środowiskowy i wygeneruj klucz aplikacji:**

```
bash
KopiujEdytuj
cp .env.example .env
php artisan key:generate
```

6. **Ustaw dane dostępowe do bazy w .env:**

```
ini
KopiujEdytuj
DB_DATABASE=pit_calculator
DB_USERNAME=root
DB_PASSWORD=twoje_haslo
```

7. **Uruchom migracje do utworzenia struktury bazy danych:**

```
bash
KopiujEdytuj
php artisan migrate
```

8. **Zapełnij bazę przykładowymi danymi (seedy):**

```
bash
KopiujEdytuj
php artisan db:seed
bash
```

9. **Uruchom serwer deweloperski Laravel:**

```
php artisan serve
```

Po uruchomieniu

- Otwórz przeglądarkę i przejdź do: <http://127.0.0.1:8000>
- Możesz zalogować się jako administrator (jeśli dane dostępne zostały zseedowane) lub utworzyć konto użytkownika.

6.3 Automatyczna konfiguracja

W projekcie znajduje się plik start.bat, który automatyzuje powyższe kroki. Aby uruchomić projekt:

1. Dwukrotnie kliknij plik start.bat.
2. Skrypt samodzielnie uruchomi lokalny serwer Laravel, stworzy bazę danych i wykona migrację.
3. Po zakończeniu skryptu aplikacja będzie dostępna pod adresem: <http://localhost:8000>

Rozwiązanie to pomaga uniknąć konieczności używania terminala lub ręcznego wpisywania poleceń.

6.4 Dane logowania

Po wykonaniu seeda, automatycznie tworzony jest użytkownik na prawach administratora z poniższymi danymi:

- Email: admin@example.pl
- Hasło: admin

Zalogowanie się tymi danymi umożliwia dostęp do funkcjonalności administratora.

Można również zalogować się na konto użytkownika na prawach user danymi:

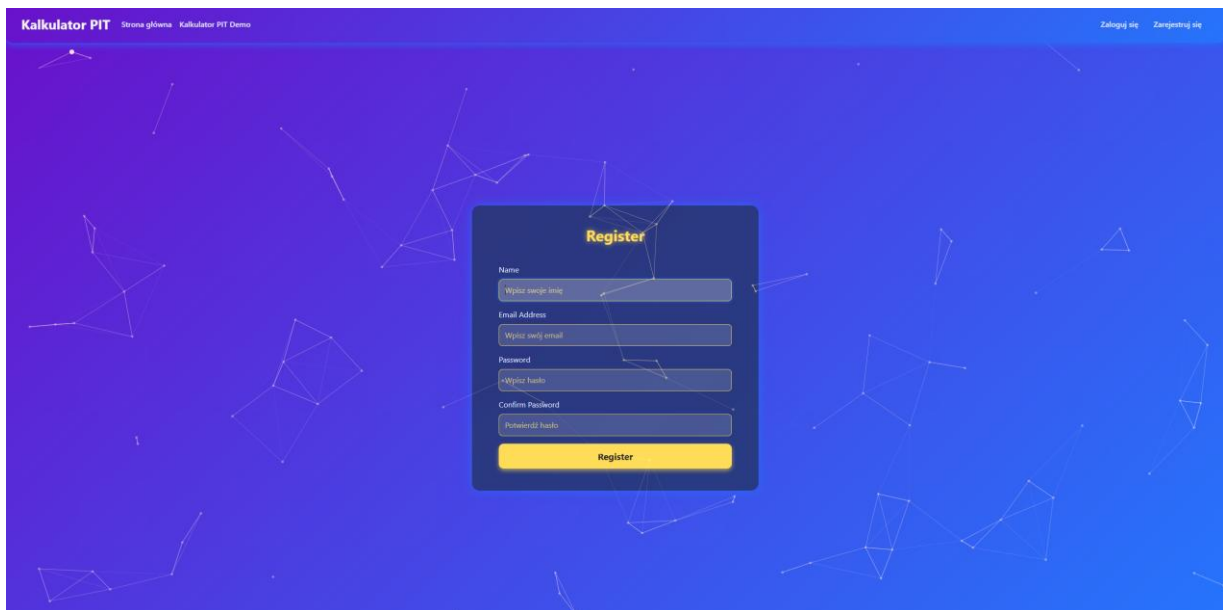
- Email: user@example.pl
- Hasło: password

Dane te tworzone są osobnym seederem w porównaniu z innymi użytkownikami na stronie tworzonymi przez factory.

7. Funkcjonalności aplikacji

Aplikacja oferuje zestaw funkcjonalności zarówno dla zwykłych użytkowników, jak i administratorów. Poniżej przedstawiam główne możliwości wraz z wizualną prezentacją działania.

7.1. Proces rejestracji i logowania użytkownika



The screenshot shows the 'Register' form on the 'Kalkulator PIT' website. The form is centered on a blue background with a white geometric pattern. It includes fields for Name, Email Address, Password, and Confirm Password, each with a placeholder text. A yellow 'Register' button is at the bottom. The website header shows 'Kalkulator PIT' and navigation links 'Strona główna' and 'Kalkulator PIT Demo'. The top right corner has links 'Zaloguj się' and 'Zarejestruj się'.

Kalkulator PIT Strona główna Kalkulator PIT Demo Zaloguj się Zarejestruj się

Register

Name
Wybierz nazwę konta

Email Address
Wybierz swój email

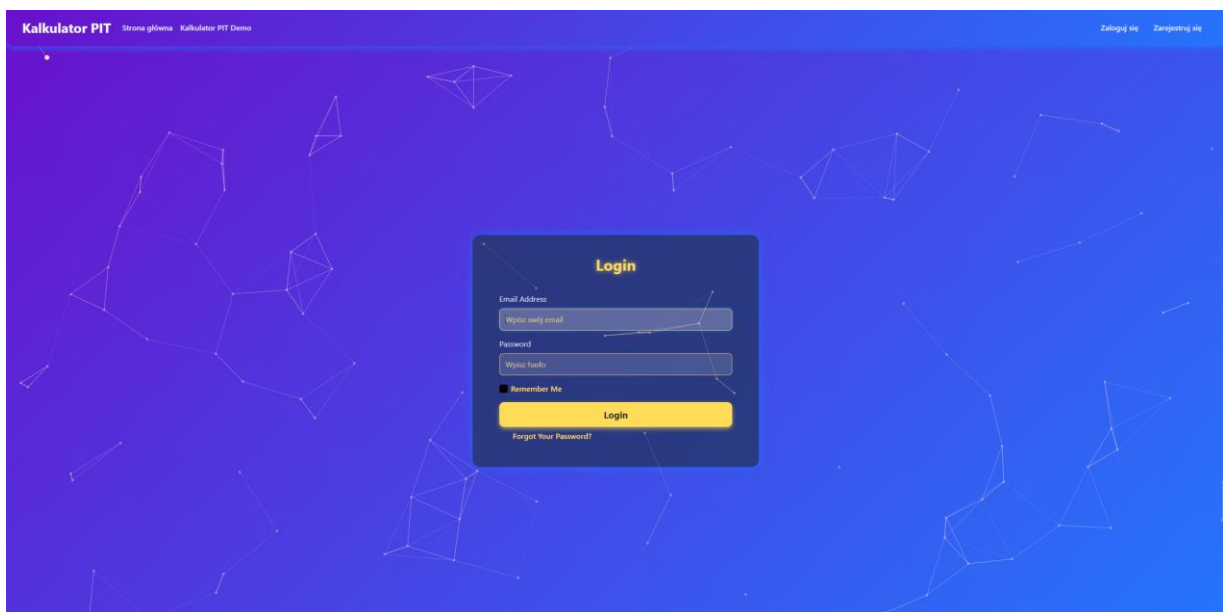
Password
Wybierz hasło

Confirm Password
Potwierdź hasło

Register

Rysunek 4 Rejestracja

- Nowy użytkownik może założyć konto podając nazwę konta, adres e-mail i hasło.



The screenshot shows the 'Login' form on the 'Kalkulator PIT' website. It is centered on the same blue background with a white geometric pattern. It includes fields for Email Address and Password, each with a placeholder text. There is a 'Remember Me' checkbox and a yellow 'Login' button. A link 'Forgot Your Password?' is at the bottom. The website header is identical to the previous screenshot.

Kalkulator PIT Strona główna Kalkulator PIT Demo Zaloguj się Zarejestruj się

Login

Email Address
Wybierz swój email

Password
Wybierz hasło

☐ Remember Me

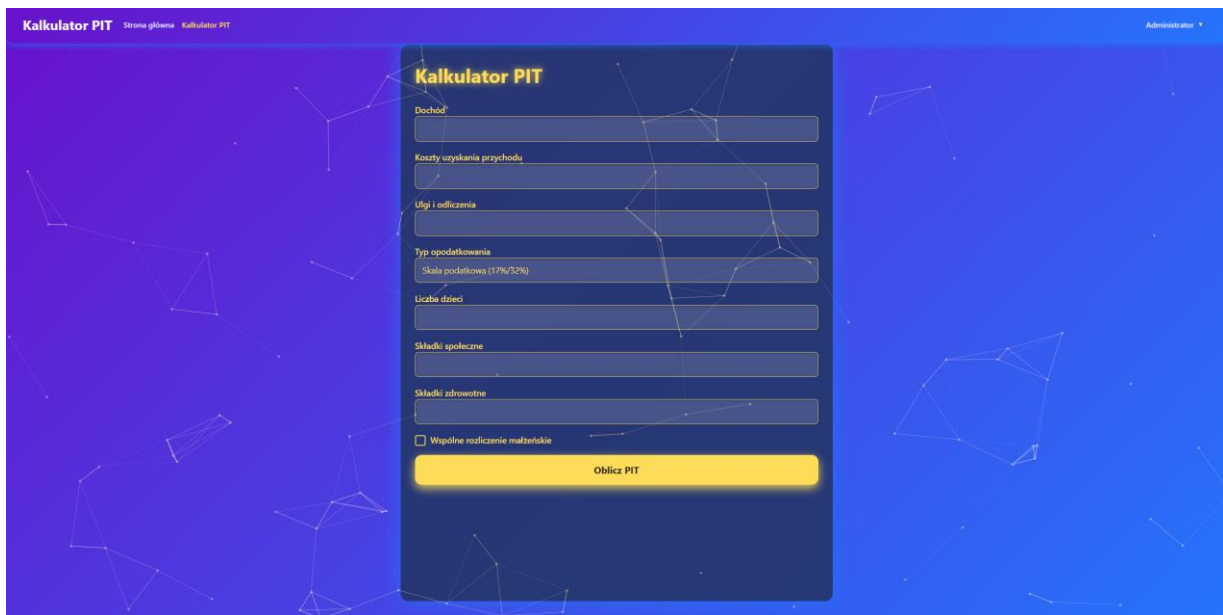
Login

[Forgot Your Password?](#)

Rysunek 5 Logowanie

- Obecni użytkownicy logują się, aby uzyskać dostęp do swojej historii kalkulacji i konta.

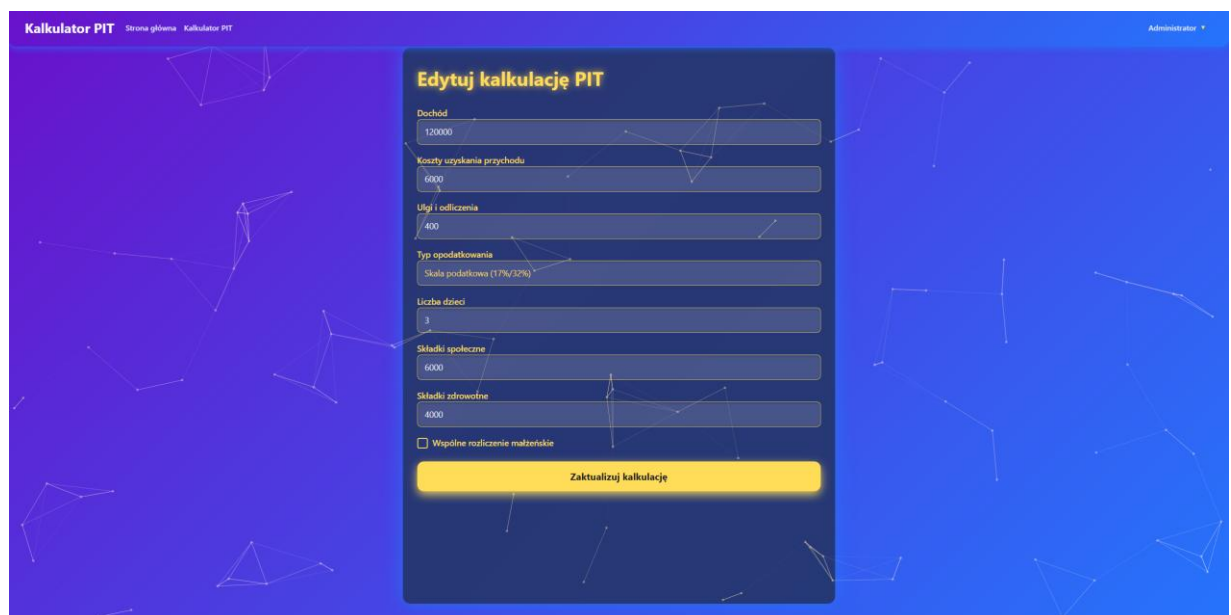
7.2. Przykładowy CRUD przeprowadzany przez użytkownika



The screenshot shows a web application titled 'Kalkulator PIT' with a navigation bar containing 'Strona główna' and 'Kalkulator PIT'. A user is logged in as 'Administrator'. The main form, titled 'Kalkulator PIT', contains the following fields: 'Dochód' (empty), 'Koszty uzyskania przychodu' (empty), 'Ułgi i odliczenia' (empty), 'Typ opodatkowania' (dropdown menu with 'Skala podatkowa (17%/32%)' selected), 'Liczba dzieci' (empty), 'Składki społeczne' (empty), 'Składki zdrowotne' (empty), and a checkbox for 'Wspólne rozliczenie małżeńskie' (unchecked). A yellow 'Oblicz PIT' button is at the bottom.

Rysunek 6 Dodawanie kalkulacji

- Po uzupełnieniu formularza użytkownik może zapisać nową kalkulację PIT, która pojawia się na liście.

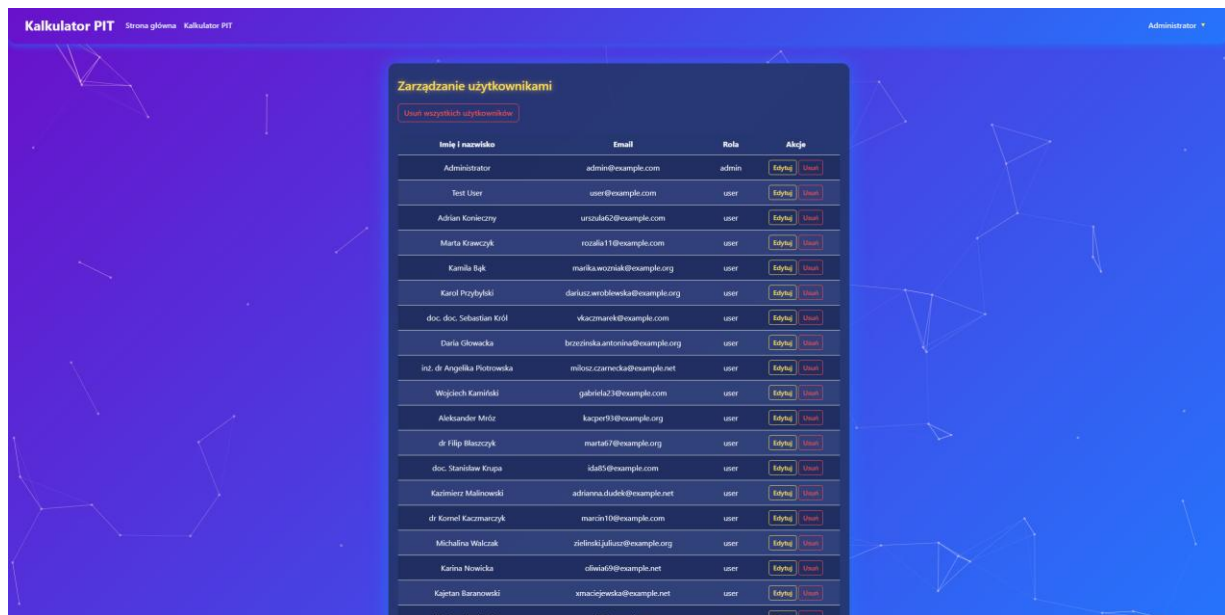


The screenshot shows the 'Edytuj kalkulację PIT' form. The fields are pre-filled with values: 'Dochód' (120000), 'Koszty uzyskania przychodu' (6000), 'Ułgi i odliczenia' (400), 'Typ opodatkowania' (Skala podatkowa (17%/32%)), 'Liczba dzieci' (3), 'Składki społeczne' (6000), and 'Składki zdrowotne' (4000). The 'Wspólne rozliczenie małżeńskie' checkbox remains unchecked. A yellow 'Zaktualizuj kalkulację' button is at the bottom.

Rysunek 7 Edytowanie łowiska

- Edytowanie kalkulacji możliwe jest po przejściu do formularza edycji z odpowiednimi uprawnieniami (np. przez administratora), co pozwala na poprawienie danych wejściowych oraz przeliczenie wartości podatku.

7.3. Zarządzanie użytkownikami przez administratora

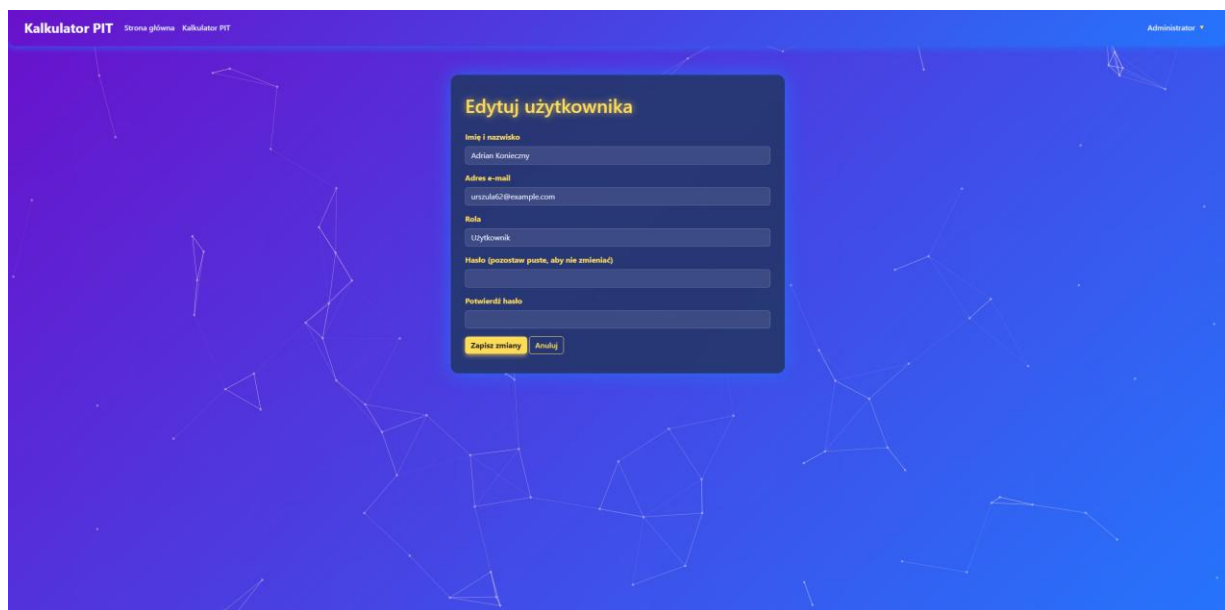


Zarządzanie użytkownikami

Usuń wszystkich użytkowników

Imię i nazwisko	Email	Rola	Akcje
Administrator	admin@example.com	admin	Edytuj Usuń
Test User	user@example.com	user	Edytuj Usuń
Adrian Koniczny	urszula62@example.com	user	Edytuj Usuń
Marta Krawczyk	rozalia1@example.com	user	Edytuj Usuń
Kamila Bąk	marika.wozniak@example.org	user	Edytuj Usuń
Karol Przybyliki	dariusz.wroblewski@example.org	user	Edytuj Usuń
doc. doc. Sebastian Kobi	vlaszmarek@example.com	user	Edytuj Usuń
Daria Glowacka	boczinska.antonina@example.org	user	Edytuj Usuń
inż. dr Angielika Piotrowska	milosz.czarnecka@example.net	user	Edytuj Usuń
Wojciech Kamiński	gabrielaz@example.com	user	Edytuj Usuń
Aleksander Miś	kasper9@example.org	user	Edytuj Usuń
dr Filip Błaszczak	martal@example.org	user	Edytuj Usuń
doc. Stanisław Kupa	ida85@example.com	user	Edytuj Usuń
Kacimierz Malinowski	adrianna.dudek@example.net	user	Edytuj Usuń
dr Kornel Kaczmarczyk	marcin10@example.com	user	Edytuj Usuń
Michalina Walczak	zielenki.juliusz@example.org	user	Edytuj Usuń
Karolina Nowicka	olivia69@example.net	user	Edytuj Usuń
Kajetan Baranowski	arnoldojewski@example.net	user	Edytuj Usuń
Prof. Krzysztof Kubiś	Prof@example.net	user	Edytuj Usuń

Rysunek 8 Przeglądanie użytkowników



Edytuj użytkownika

Imię i nazwisko

Adrian Koniczny

Adres e-mail

urszula62@example.com

Rola

Użytkownik

Hasło (pozostaw puste, aby nie zmieniać)

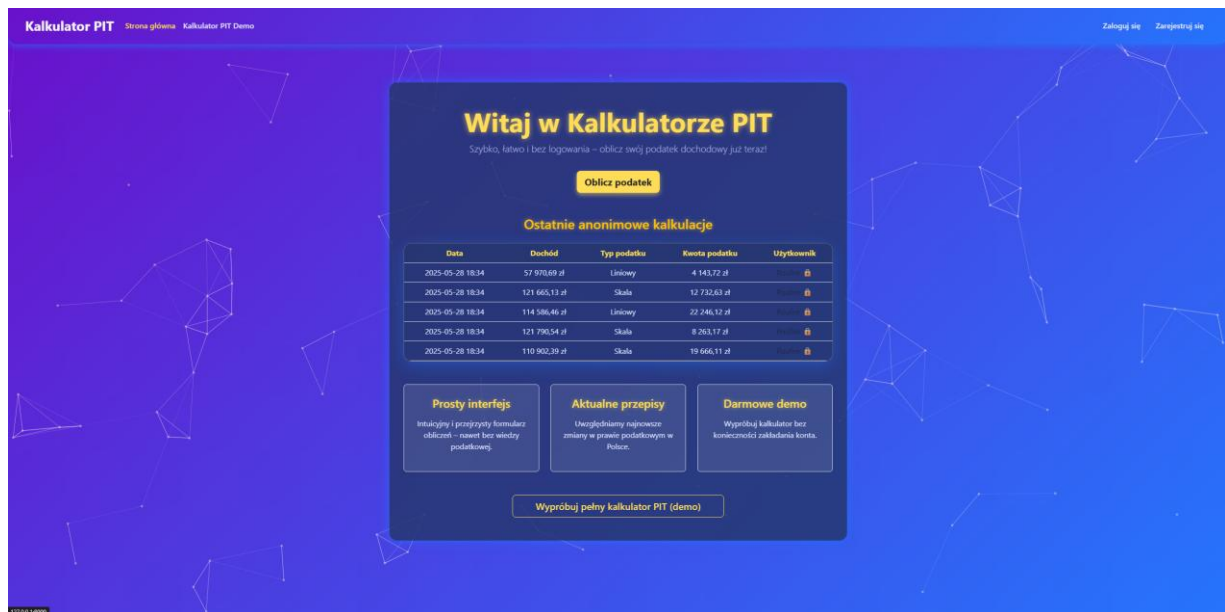
Potwierdź hasło

Zapisz zmiany Anuluj

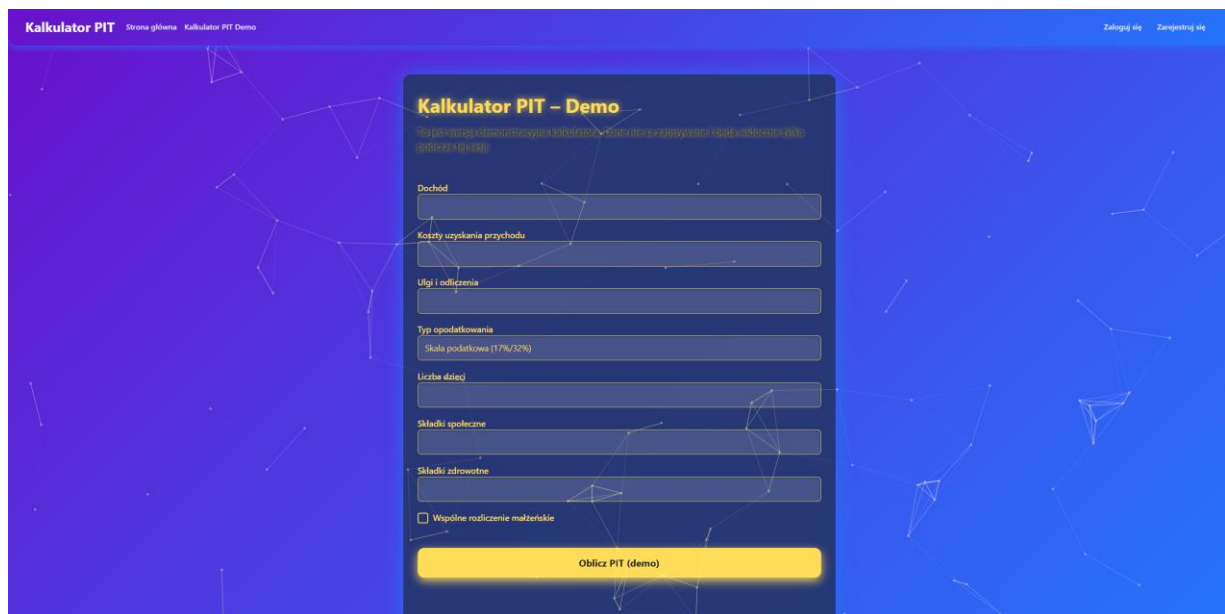
Rysunek 9 Edycja użytkownika

- Administrator ma dostęp do panelu zarządzania użytkownikami oraz historią kalkulacji podatkowych. Może przeglądać, edytować i usuwać rekordy utworzone przez użytkowników.
- Możliwość tworzenia, edycji i usuwania kalkulacji (CRUD) została zaprezentowana w poprzednich sekcjach.

7.4. Przeglądanie ogólnodostępnych zasobów (bez logowania)



Rysunek 10 Strona Powitalna



Rysunek 11 Ogólnodostępny kalkulator demo

- Osoba niezalogowana ma dostęp do publicznych informacji, takich jak lista ostatnich anonimowych kalkuleacji oraz możliwość skorzystania z wersji demo kalkulatora PIT.
- Aby zapisać wynik kalkuleacji i móc wrócić do niego później, użytkownik musi się zalogować lub zarejestrować. Dzięki temu rozwiązaniu dane użytkowników są chronione, a nieautoryzowany dostęp do pełnej funkcjonalności systemu – ograniczony.

7.5. Zarządzanie swoimi kalkulacjami przez użytkownika

Kalkulator PIT Strona główna Kalkulator PIT Test User

Edytuj kalkulację PIT

Dochód: 50000

Koszty uzyskania przychodu: 5000

Ulgi i odliczenia: 200

Typ opodatkowania: Skala podatkowa (17%/32%)

Liczba dzieci: 1

Składki społeczne: 5000

Składki zdrowotne: 3000

☐ Wspólne rozliczenie małżeńskie

Zaktualizuj kalkulację

Rysunek 12 Widok edycji kalkulacji

Kalkulator PIT Strona główna Kalkulator PIT Test User

Moje kalkulacje PIT

Kalkulacja #1

Dochód: 50 000,00 zł

Kwota podatku: 2 164,00 zł

Edytuj Usuń

Rysunek 13 Przeglądanie kalkulacji

- Po zalogowaniu użytkownik ma dostęp do listy swoich kalkulacji podatkowych oraz możliwość dodawania nowych.
- Intuicyjny formularz umożliwia szybkie uzupełnienie danych i natychmiastowe przeliczenie podatku.
- Przyjazny interfejs użytkownika sprawia, że cały proces — od wprowadzenia danych po zapis kalkulacji — odbywa się w sposób przejrzysty i wygodny, bez konieczności korzystania ze skomplikowanych narzędzi czy formularzy.

7.6. Zarządzanie swoimi danymi przez użytkownika

Kalkulator PIT Strona główna Kalkulator PIT Test User

Zarządzaj kontem

Imię
Test User
Zapisz zmiany

Adres e-mail
user@example.com
Zapisz zmiany

Aktualne hasło
Nowe hasło
Potwierdzenie hasła
Zapisz zmiany

Rysunek 14 Edycja profilu użytkownika

- Użytkownik ma możliwość edycji oraz usunięcia swoich kalkulacji podatkowych.
- Może również zaktualizować swoje dane profilowe, takie jak imię, adres e-mail, hasło oraz avatar.
- Wszystkie formularze wyposażone są w walidację danych i odpowiednie zabezpieczenia.

Tak zaprojektowany panel ustawień konta łączy przejrzysty interfejs z solidną logiką bezpieczeństwa, zapewniając, że do systemu trafiają wyłącznie poprawnie zweryfikowane i bezpiecznie przetworzone informacje.

7.7. Walidacja danych – przykłady

```
public function update(Request $request, $id)
{
    $request->validate([
        'name' => 'required|string|max:255',
        'email' => 'required|email|max:255|unique:users,email,' . $id,
        'role' => 'required|string',
        'password' => 'nullable|string|min:8|confirmed',
    ]);

    $user = User::findOrFail($id);
    $user->name = $request->name;
    $user->email = $request->email;
    $user->role = $request->role;

    if ($request->filled('password')) {
        $user->password = bcrypt($request->password);
    }

    $user->save();

    return redirect()->route('admin.users.index')->with('success', 'Dane użytkownika zostały zaktualizowane.');
```

Kalkulator PIT Strona główna Kalkulator PIT Test User

Zarządzaj kontem

Imię

Wypełnij to pole

Zapisz zmiany

Adres e-mail

Zapisz zmiany

Aktualne hasło

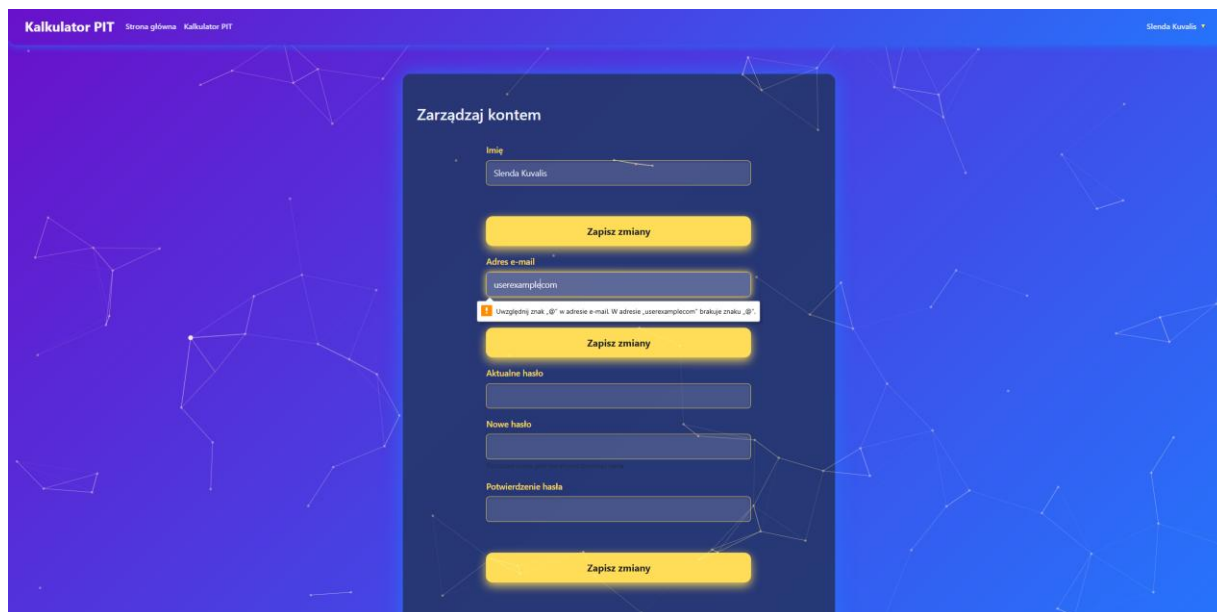
Nowe hasło

Potwierdzenie hasła

Zapisz zmiany

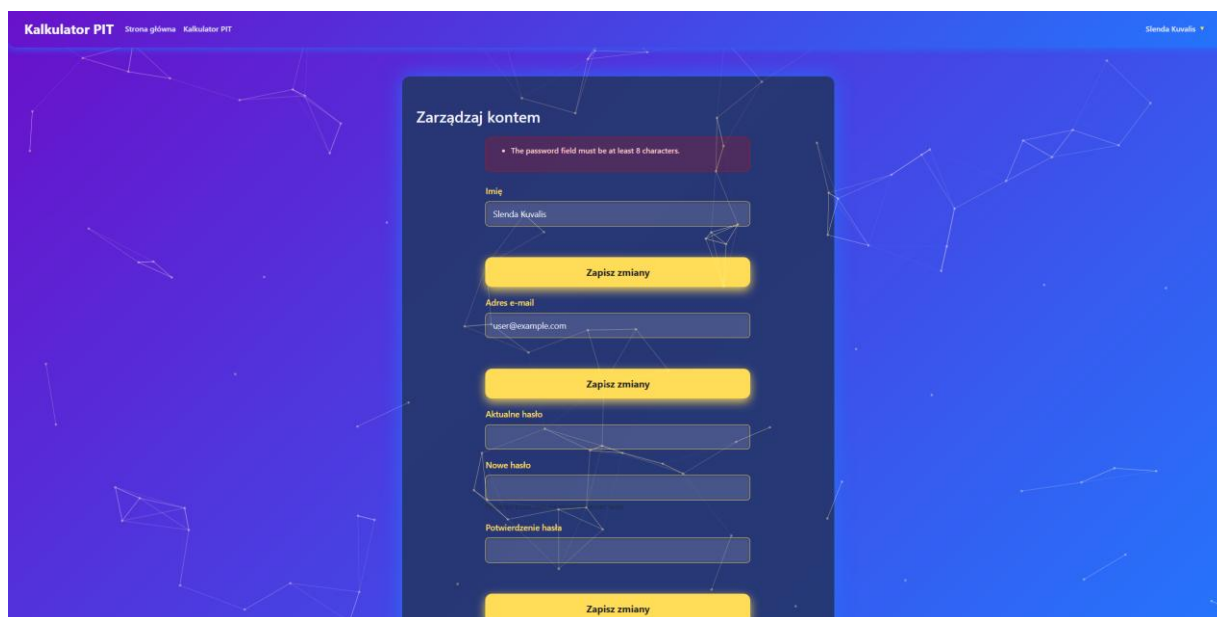
Rysunek 15 Brak danych

Przykład: Brak danych w polu nazwy konta.



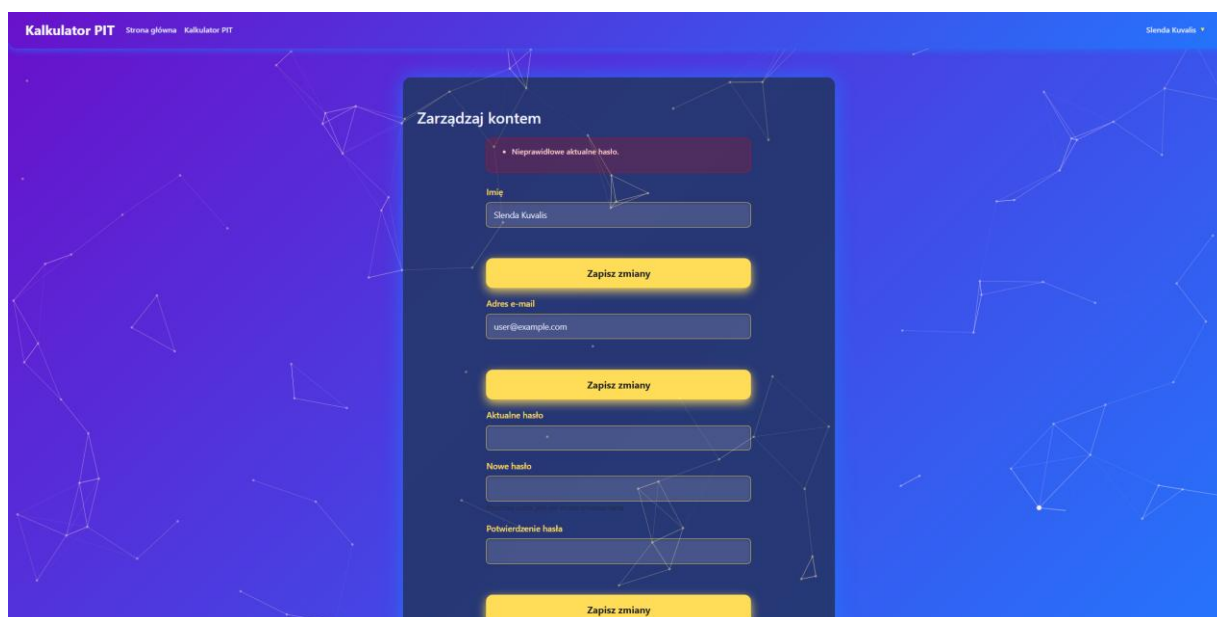
Rysunek 16 Błędny e-mail

Przykład: Błędnie wpisany adres e-mail.



Rysunek 17 Niepoprawna długość hasła

Przykład: Zbyt krótkie hasło.



Rysunek 18 Niepoprawne stare hasło Przykład:

Błędne stare hasło.

- Aplikacja korzysta z wbudowanego mechanizmu walidacji Laravel – reguły walidacyjne są bezpośrednio definiowane w metodach kontrolerów (np. calculate, update, store).
- W każdej metodzie `$request->validate([...])` określa szczegółowe zasady sprawdzania poprawności danych dla formularzy użytkownika i administratora, np.:

✓ pola wymagane (required):
`'income' => 'required|numeric|min:0'`

✓ walidacja typów i zakresów (numeric|min:0, in:scale,flat,ryczałt):
`'tax_type' => 'required|in:scale,flat,ryczałt'`

✓ walidacja unikalności z pominięciem edytowanego ID:
`'email' => 'required|email|unique:users,email,' . $id`

✓ długość hasła (min:8, confirmed):
`'password' => 'nullable|string|min:8|confirmed'`

✓ pola opcjonalne (nullable):
`'social_insurance' => 'nullable|numeric|min:0'`

✓ walidacja logiczna (boolean):
`'is_married' => 'nullable|boolean'`

Po stronie klienta Blade wykorzystuje:

✓ atrybuty HTML5 (required, min, max, type, pattern) do wstępnej weryfikacji danych w przeglądarce,

✓ atrybut novalidate w formularzu – w przypadku chęci pominięcia walidacji HTML5 i testowania tylko reguł backendowych,

✓ dyrektywy @error('pole') ... @enderror do wyświetlania komunikatów walidacyjnych przekazanych z kontrolera.

Dzięki zastosowaniu powyższych zabezpieczeń dane użytkownika są walidowane zarówno po stronie przeglądarki, jak i serwera – co zapewnia spójność, bezpieczeństwo oraz brak możliwości zapisania niepoprawnych informacji w bazie danych.

Przykład działania w kodzie na podstawie widoku resources/views/admin/dashboard.blade.php

```
1 @extends('layouts.app')
2
3 @section('content')
4 <style>
5     html,
6     body {
7         height: 100%;
8         margin: 0;
9         background: linear-gradient(135deg, #6a11cb 0%, #2575fc 100%);
10        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
11        color: #e6e6ff;
12        overflow-x: hidden;
13    }
14
15    .content-wrapper {
16        flex: 1;
17        display: flex;
18        justify-content: center;
19        align-items: center;
20        padding-top: 80px;
21        padding-bottom: 40px;
22        box-sizing: border-box;
23        position: relative;
24        z-index: 10;
25        width: 100%;
26    }
27
28    .container.py-5,
29    .container.py-4 {
30        max-width: 1280px;
31        background: #ffffff;
32        border-radius: 16px;
33        box-shadow: 0 0 25px #373737;
34        padding: 40px;
35        position: relative;
36        z-index: 2;
37    }
38
39    h1.display-4,
40    h1.mb-4 {
41        color: #ffdd59;
42    }
43
44    .btn-primary {
45        background-color: #ffdd59;
46        border: none;
47        color: #222;
48        font-weight: 700;
49        box-shadow: 0 4px 10px #c5c5c5;
50        transition: background-color 0.3s ease, color 0.3s ease;
51    }
52
53    .btn-primary:hover {
54        background-color: #ffc700;
55        color: #111;
56        box-shadow: 0 6px 14px #a5a5a5;
57    }
58
59    .btn-outline-primary {
60        color: #ffdd59;
61        border-color: #ffdd59;
62        font-weight: 600;
63        transition: background-color 0.3s ease, color 0.3s ease;
64    }
65
66    .btn-outline-primary:hover {
67        background-color: #ffdd59;
68        color: #222;
69    }
70
71    .table {
72        width: 100%;
73    }
74
75    .table th,
76    .table td {
77        word-wrap: break-word;
78        font-size: 0.95rem;
79    }
80
81    .table-bordered {
82        border-color: #c5c5c5;
83    }
84
```



```

86 .table-bordered: not(caption)>* {
87     border-color: rgba(255, 255, 255, 0.2) !important;
88 }
89
90 .dataTables_info,
91 .text-muted {
92     display: none !important;
93 }
94
95 .table-custom-bg {
96     background: rgba(38, 55, 112, 0.4);
97     backdrop-filter: blur(8px);
98     -webkit-backdrop-filter: blur(8px);
99     box-shadow: 0 8px 32px rgba(37, 117, 252, 0.4);
100     border-radius: 16px;
101     border: 1px solid rgba(255, 221, 89, 0.3);
102 }
103
104 .table-custom-bg,
105 .table-custom-bg th,
106 .table-custom-bg td {
107     color: #e0e7ff !important;
108 }
109
110 .table-custom-bg thead {
111     background-color: rgba(38, 55, 112, 0.7);
112     color: #ffdd59 !important;
113 }
114
115 .table-responsive {
116     overflow-x: auto;
117 }
118
119 @media (max-width: 768px) {
120
121     .container.py-5,
122     .container.py-4 {
123         padding: 20px;
124     }
125 }

```

```

126 .table th,
127 .table td {
128     font-size: 0.8rem;
129     padding: 0.4rem;
130 }
131
132 .table thead {
133     display: none;
134 }
135
136 .table,
137 .table tbody,
138 .table tr,
139 .table td {
140     display: block;
141     width: 100%;
142 }
143
144 .table tr {
145     margin-bottom: 1rem;
146     border-bottom: 2px solid #ffdd59;
147 }
148
149 .table td {
150     text-align: right;
151     padding-left: 50%;
152     position: relative;
153 }
154
155 .table td::before {
156     content: attr(data-label);
157     position: absolute;
158     left: 1rem;
159     width: 45%;
160     padding-right: 1rem;
161     white-space: nowrap;
162     font-weight: bold;
163     text-align: left;

```

```

167 #background-canvas {
168     position: fixed;
169     top: 0;
170     left: 0;
171     width: 100vw;
172     height: 100vh;
173     pointer-events: none;
174     z-index: 1;
175     mix-blend-mode: screen;
176 }
177 </style>
178
179 <div class="container py-4">
180     <h1 class="mb-4">Panel administratora <span> historia zmian</span></h1>
181
182     @if (session('success'))
183     <div class="alert alert-success alert-dismissible fade show" role="alert">
184         {{ session('success') }}
185         <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Zamknij"></button>
186     </div>
187     @endif
188
189     <div class="mb-3">
190         <form action="{{ route('dashboard.clear-history') }}" method="POST">
191             <input type="submit" value="Czy na pewno chcesz usunąć wszystkie kalkulacje i historię?">
192             @csrf
193             @method('DELETE')
194             <button type="submit" class="btn btn-outline-danger">
195                 <i class="bi bi-trash3"></i> Wyczyść wszystkie kalkulacje i historię
196             </button>
197         </form>
198     </div>
199
200
201     @if ($histories->isEmpty())
202     <div class="alert alert-info">Brak zapisanych zmian.</div>

```

```

203 ~ @else
204 ~ <div class="table-responsive">
205 ~ <table class="table table-bordered table-striped align-middle text-center table-custom-bg">
206 ~ <thead class="table-dark">
207 ~ <tr>
208 ~ <th>Data</th>
209 ~ <th>Użytkownik</th>
210 ~ <th>Typ akcji</th>
211 ~ <th>Kwota podatku</th>
212 ~ <th style="min-width: 500px;">Szczegóły</th>
213 ~ <th>Akcje</th>
214 ~ </tr>
215 ~ </thead>
216 ~ <tbody>
217 ~ @php
218 ~ $labels = [
219 ~ 'income' => 'Przychód',
220 ~ 'expenses' => 'Koszty uzyskania',
221 ~ 'deductions' => 'Odliczenia',
222 ~ 'tax_type' => 'Typ podatku',
223 ~ 'children' => 'Liczba dzieci',
224 ~ 'social_insurance' => 'Składki ZUS',
225 ~ 'health_insurance' => 'Składki zdrowotne',
226 ~ 'is_married' => 'Wspólne rozliczenie',
227 ~ 'taxable_income' => 'Dochód do opodatkowania',
228 ~ 'tax_amount' => 'Kwota podatku',
229 ~ ];
230 ~
231 ~ $actionLabels = [
232 ~ 'created' => 'Utworzono',
233 ~ 'updated' => 'Zaktualizowano',
234 ~ 'deleted' => 'Usunięto',
235 ~ ];
236 ~
237 ~ function displayValue($key, $value)
238 ~ {
239 ~ if ($key === 'is_married') {
240 ~ return $value ? 'Tak' : 'Nie';
241 ~ }
242 ~
243 ~ if ($key === 'tax_type') {
244 ~ $types = ['scale' => 'Skala', 'flat' => 'Liniowy', 'ryczalt' => 'Ryczałt'];
245 ~ return $types[$value] ?? $value;
246 ~ }
247 ~ if ($key === 'children') {
248 ~ return (int) $value;
249 ~ }
250 ~ if (is_numeric($value)) {
251 ~ return number_format($value, 2, ',', ' ') . ' zł';
252 ~ }
253 ~ return $value ?? '-';
254 ~ }
255 ~ @endphp
256 ~
257 ~ @foreach ($histories as $index => $history)
258 ~ @php
259 ~ $details = is_array($history->previous_values)
260 ~ ? $history->previous_values
261 ~ : json_decode($history->previous_values, true) ?? [];
262 ~ $newDetails = is_array($history->new_values)
263 ~ ? $history->new_values
264 ~ : json_decode($history->new_values, true) ?? [];
265 ~
266 ~ $hasPrevious = empty($details);
267 ~ $hasCurrent = empty($newDetails);
268 ~
269 ~ $taxAmountToShow = 0;
270 ~ if ($history->action === 'deleted') {
271 ~ $taxAmountToShow = $details['tax_amount'] ?? 0;
272 ~ } elseif ($hasCurrent) {
273 ~ $taxAmountToShow = $newDetails['tax_amount'] ?? 0;
274 ~ } elseif ($hasPrevious) {
275 ~ $taxAmountToShow = $details['tax_amount'] ?? 0;
276 ~ }
277 ~ @endphp
278 ~ <tr>
279 ~ <td>{{ $history->created_at->format('Y-m-d H:i') }}</td>
280 ~ <td>{{ $history->user->name ?? 'Brak' }}</td>
281 ~ <td>{{ $actionLabels[$history->action] ?? ucfirst($history->action) }}</td>
282 ~ <td>{{ number_format($taxAmountToShow, 2, ',', ' ') }} zł</td>
283 ~ <td class="text-start">
284 ~ <div class="d-flex justify-content-center">
285 ~ <button class="btn btn-sm btn-outline-primary" type="button"
286 ~ data-bs-toggle="collapse" data-bs-target="#detailsCollapse{{ $index }}"
287 ~ aria-expanded="false" aria-controls="detailsCollapse{{ $index }}">
288 ~ Pokaż szczegóły
289 ~ </button>
290 ~ </div>
291 ~
292 ~ <div class="collapse mt-2" id="detailsCollapse{{ $index }}">
293 ~ @if ($history->action === 'created')
294 ~ @php $dataToShow = $newDetails; @endphp
295 ~ @if (empty($dataToShow))
296 ~ <ul class="mb-0 list-unstyled">
297 ~ @foreach ($labels as $key => $label)
298 ~ @if (isset($dataToShow[$key]))
299 ~ <li class="d-flex justify-content-between border-bottom py-1">
300 ~ <strong>{{ $label }}</strong>
301 ~ <span>{{ displayValue($key, $dataToShow[$key]) }}</span>
302 ~ </li>
303 ~ @endif
304 ~ @endforeach
305 ~ </ul>
306 ~ @else
307 ~ <div class="text-muted">Brak danych do wyświetlenia.</div>
308 ~ @endif
309 ~ @elseif ($history->action === 'updated')
310 ~ @if ($hasPrevious || $hasCurrent)
311 ~ <ul class="mb-0 list-unstyled">
312 ~ @foreach ($labels as $key => $label)
313 ~ @php
314 ~ $old = $details[$key] ?? null;

```

```

315         $new = $newDetails[$key] ?? null;
316     @endif
317     @if ($old != null || $new != null)
318         <li class="d-flex justify-content-between border-bottom py-1">
319             <strong>{{ $label }}</strong>
320             <span>
321                 @if ($old != null && $new != null && $old != $new)
322                     <span>
323                         class="text-danger">{{ $displayValue($key, $old) }}</span>
324                         <span class="text-success ms-2">
325                             <i class="bi bi-arrow-right"></i>
326                             {{ $displayValue($key, $new) }}
327                         </span>
328                     @else
329                         {{ $displayValue($key, $new ?? $old) }}
330                     @endif
331                 </span>
332             </li>
333         @endif
334     @foreach
335 </ul>
336 @else
337     <div class="text-muted">Brak danych do wyświetlenia.</div>
338 @endif
339 @elseif ($history->action === 'deleted')
340     @if ($hasPrevious)
341         <ul class="mb-0 list-unstyled">
342             @foreach ($labels as $key => $label)
343                 @if (isset($details[$key]))
344                     <li class="d-flex justify-content-between border-bottom py-1">
345                         <strong>{{ $label }}</strong>
346                         <span>{{ $displayValue($key, $details[$key]) }}</span>
347                     </li>
348                 @endif
349             @foreach
350 </ul>
351 @endif

```

```

352     <div class="text-muted">Brak danych do wyświetlenia.</div>
353 @endif
354 @endif
355 </div>
356 </td>
357 <td>
358     <div class="d-flex justify-content-center gap-2">
359         @if ($history->user)
360             <a href="{{ route('tax-history.edit', $history->id) }}"
361                 class="btn btn-sm btn-outline-primary">
362                 <i class="bi bi-pencil-square"></i> Edytuj
363             </a>
364         @endif
365         <form action="{{ route('tax-history.destroy', $history->id) }}" method="POST"
366             onsubmit="return confirm('Czy na pewno chcesz usunąć tę historię?');">
367             @csrf
368             @method('DELETE')
369             <button type="submit" class="btn btn-sm btn-outline-danger">Usuń</button>
370         </form>
371     </div>
372 </td>
373 </tr>
374 @endforeach
375 </tbody>
376 </table>
377 </div>
378 @endif
379 </div>
380 </div>
381 </div>
382 </div>
383 </div>
384 </div>
385 </div>
386 </div>
387 </div>

```

```

388 <canvas id="background-canvas"></canvas>
389
390 <script>
391     (() => {
392         const canvas = document.getElementById('background-canvas');
393         const ctx = canvas.getContext('2d');
394
395         let width, height, points;
396         const POINT_COUNT = 120;
397         let mouse = {
398             x: null,
399             y: null
400         };
401
402         function resize() {
403             width = window.innerWidth;
404             height = window.innerHeight;
405             canvas.width = width * devicePixelRatio;
406             canvas.height = height * devicePixelRatio;
407             canvas.style.width = width + 'px';
408             canvas.style.height = height + 'px';
409             ctx.setTransform(1, 0, 0, 1, 0, 0);
410             ctx.scale(devicePixelRatio, devicePixelRatio);
411         }
412
413         function distance(p1, p2) {
414             return Math.sqrt((p1.x - p2.x) ** 2 + (p1.y - p2.y) ** 2);
415         }
416
417         function findNearestNeighbors(point, allPoints, n = 4) {
418             return allPoints
419                 .filter(p => p !== point)
420                 .map(p => ({
421                     point: p,
422                     dist: distance(point, p)
423                 }))
424                 .sort((a, b) => a.dist - b.dist)
425                 .slice(0, n)
426                 .map(item => item.point);
427         }
428     })

```

```

429
430
431 function init() {
432   resize();
433   points = [];
434
435   for (let i = 0; i < POINT_COUNT; i++) {
436     points.push({
437       x: Math.random() * width,
438       y: Math.random() * height,
439       originX: 0,
440       originY: 0,
441       vx: (Math.random() - 0.5) * 0.4,
442       vy: (Math.random() - 0.5) * 0.4,
443     });
444
445     points.forEach(p => {
446       p.originX = p.x;
447       p.originY = p.y;
448     });
449   }
450
451   function draw() {
452     ctx.clearRect(0, 0, width, height);
453
454     points.forEach(p => {
455       const neighbors = findNearestNeighbors(p, points, 4);
456
457       neighbors.forEach(nbr => {
458         const dist = distance(p, nbr);
459         if (dist < 180) {
460           let opacity = 1 - dist / 180;
461
462           if (mouse.x !== null) {
463             const midX = (p.x + nbr.x) / 2;
464             const midY = (p.y + nbr.y) / 2;
465             const mouseDist = Math.sqrt((mouse.x - midX) ** 2 + (mouse.y - midY) **

```

```

466               2);
467             if (mouseDist < 130) {
468               opacity = Math.min(1, opacity + (130 - mouseDist) / 130);
469             }
470
471             ctx.strokeStyle = `rgba(255, 221, 89, ${opacity * 0.6})`;
472             ctx.lineWidth = 1;
473             ctx.beginPath();
474             ctx.moveTo(p.x, p.y);
475             ctx.lineTo(nbr.x, nbr.y);
476             ctx.stroke();
477           }
478         });
479
480         if (mouse.x !== null) {
481           const distToMouse = distance(p, mouse);
482           if (distToMouse < 130) {
483             let opacity = 1 - distToMouse / 130;
484             ctx.strokeStyle = `rgba(255, 221, 89, ${opacity})`;
485             ctx.lineWidth = 1.5;
486             ctx.beginPath();
487             ctx.moveTo(p.x, p.y);
488             ctx.lineTo(mouse.x, mouse.y);
489             ctx.stroke();
490           }
491         }
492       });
493     }
494
495     points.forEach(p => {
496       let radius = 2.5;
497       let opacity = 0.4;
498
499       if (mouse.x !== null) {
500         const d = distance(p, mouse);
501         if (d < 100) {
502           radius = 5;
503           opacity = 1;
504         }
505       }
506
507       ctx.beginPath();
508       ctx.arc(p.x, p.y, radius, 0, Math.PI * 2);
509       ctx.fillStyle = `rgba(255, 221, 89, ${opacity})`;
510       ctx.fill();
511     });
512   }
513
514   function animate() {
515     points.forEach(p => {
516       p.vx += (Math.random() - 0.5) * 0.1;
517       p.vy += (Math.random() - 0.5) * 0.1;
518
519       p.x += p.vx;
520       p.y += p.vy;
521
522       p.vx += (p.originX - p.x) * 0.002;
523       p.vy += (p.originY - p.y) * 0.002;
524
525       p.vx *= 0.98;
526       p.vy *= 0.98;
527
528       if (p.x < 0) {
529         p.x = 0;
530         p.vx *= -0.7;
531       }
532       if (p.x > width) {
533         p.x = width;
534         p.vx *= -0.7;
535       }
536       if (p.y < 0) {
537         p.y = 0;
538         p.vy *= -0.7;
539       }

```

```

540 ✓      if (p.y > height) {
541          p.y = height;
542          p.y *= -0.7;
543      }
544  });
545
546  draw();
547  requestAnimationFrame(animate);
548  }
549
550 ✓  window.addEventListener('resize', () => {
551      resize();
552      init();
553  });
554
555 ✓  window.addEventListener('mousemove', (e) => {
556      mouse.x = e.clientX;
557      mouse.y = e.clientY;
558  });
559
560 ✓  window.addEventListener('mouseout', () => {
561      mouse.x = null;
562      mouse.y = null;
563  });
564
565  init();
566  animate();
567  })();
568  </script>
569  #endsection
570

```

Rysunek 19 Widok przykładowego blade z formularzem, stylem i tłem

8. Podsumowanie

Projekt **Kalkulatora PIT** to nowoczesna aplikacja webowa stworzona w oparciu o framework **Laravel 12**, której głównym celem jest umożliwienie użytkownikom szybkiego i intuicyjnego obliczania podatku dochodowego oraz zarządzania historią kalkulacji. System uwzględnia zarówno potrzeby zwykłych użytkowników, jak i administratorów, oferując im dedykowane panele funkcjonalne.

W dokumentacji szczegółowo przedstawiono:

- proces konfiguracji i uruchomienia środowiska lokalnego,
- strukturę aplikacji i organizację tras,
- funkcjonalności dostępne dla różnych ról użytkowników (gość, użytkownik, administrator),
- zasady responsywnego projektowania interfejsu,
- procesy uwierzytelniania i autoryzacji,
- system zarządzania użytkownikami i historią kalkulacji,
- implementację operacji CRUD,
- integrację z systemem walidacji danych oraz ochronę przed atakami CSRF.

Aplikacja wyróżnia się czytelnym i spójnym interfejsem graficznym z dynamicznym tłem oraz trybem ciemnym, zapewniając przyjazne doświadczenie użytkownika również na urządzeniach mobilnych. Dzięki zastosowaniu komponentów Bootstrap, Blade oraz nowoczesnych technik frontendowych, projekt łączy przejrzystość kodu z estetycznym wykonaniem.

Rozwiązanie zostało wzbogacone o **moduł demo**, umożliwiający anonimowe testowanie funkcji kalkulatora bez konieczności logowania, co może być przydatne w przypadku prezentacji lub wprowadzenia do aplikacji.