

Programowanie aplikacji klient – serwer

## Aplikacja do obliczania podatku dochodowego za wybrane lata

Adrian Cybulski

10018

Informatyka

Spis treści

[Aplikacja do obliczania podatku dochodowego za wybrane lata 1](#_Toc10990024)

[Wstęp 2](#_Toc10990025)

[Środowisko dla zaprogramowanej aplikacji 2](#_Toc10990026)

[Środowisko programisty 2](#_Toc10990027)

[Serwer WWW 2](#_Toc10990028)

[Serwer baz danych 2](#_Toc10990029)

[Struktura bazy danych 3](#_Toc10990030)

[Elementy i funkcje aplikacji 6](#_Toc10990031)

[Ekran rejestracji 6](#_Toc10990032)

[Ekran logowania 7](#_Toc10990033)

[Część administracyjna 10](#_Toc10990034)

[Rozwiązania programistyczne 10](#_Toc10990035)

[Połączenie z baza danych 10](#_Toc10990036)

[Ekran rejestracji 11](#_Toc10990037)

[Ekran logowania 12](#_Toc10990038)

[Wyświetlanie danych 14](#_Toc10990039)

[Obliczanie podatku 15](#_Toc10990040)

[Wyświetlanie danych 17](#_Toc10990041)

[Panel administracyjny 19](#_Toc10990042)

[Eksport danych 21](#_Toc10990043)

# Spis rysunków

[Rys 2. 1 Ekran powitalny 8](#_Toc10990726)

[Rys 2. 2 Ekran rejestracji 9](#_Toc10990727)

[Rys 2. 3 Pomyśle zarejestrowano 9](#_Toc10990728)

[Rys 2. 4 Ekran logowania 9](#_Toc10990729)

[Rys 2. 5 Zalogowano 10](#_Toc10990730)

[Rys 2. 6 Strona główna 10](#_Toc10990731)

[Rys 2. 7 Nav-nar 10](file:///C:\Users\Goldi\Documents\!Szkoła\Projekt%20podatki%20Adrian%20Cybulski%2010018.docx#_Toc10990732)

[Rys 2. 8 Brak danych 10](#_Toc10990733)

[Rys 2. 9 Formularz 11](#_Toc10990734)

[Rys 2. 10 Wynik 11](file:///C:\Users\Goldi\Documents\!Szkoła\Projekt%20podatki%20Adrian%20Cybulski%2010018.docx#_Toc10990735)

[Rys 2. 11 Zalogowany nav-bar 11](file:///C:\Users\Goldi\Documents\!Szkoła\Projekt%20podatki%20Adrian%20Cybulski%2010018.docx#_Toc10990736)

[Rys 2. 12 Formularz zalogowanego 12](#_Toc10990737)

[Rys 2. 13 Admin dane 12](file:///C:\Users\Goldi\Documents\!Szkoła\Projekt%20podatki%20Adrian%20Cybulski%2010018.docx#_Toc10990738)

[Rys 2. 14 Admin nav-bar 12](#_Toc10990739)

# Wstęp

Celem projektu było stworzenie aplikacji internetowej umożliwiającej liczenie podatku dochodowego dla osób fizycznych w latach obrachunkowych 2017, 2018 i 2019. Aplikacja posiada również możliwość dalszej rozbudowy o kolejne lata. Całość projektu została wykonana w języku programowania PHP.

Według zaleceń inwestora, aplikacja powinna posiadać funkcje umożliwiajace użytkownikow samodzielne wprowadzanie danych, według których rozliczane będą podatki dochodowe. Elementami wymaganymi przez inwestora są: przychód, koszt uzyskania przychodu, składki na ubezpieczenie społeczne oraz rok obrachunkowy adekwatny do danych. Dodatkowo aplikacja powinna przechowywać dane dodatkowe takie jak: imię, nazwisko, pesel oraz nip. W celu jednoznaczej identyfikacji użytkowników.

Aplikacja powinna również zawierać system logowania oraz rejestracji dla użytkowników ułatwiając tym samym dostęp do danych podatkowych z poprzednich lat. Żeby taki system miał prawo bytu wszystkie powyższe funkcje zostały zachowane dla użytkowników zalogowanych.

# Środowisko dla zaprogramowanej aplikacji

## Środowisko programisty

Całość projektu została wykonana z wykorzystaniem języka programowania PHP. Który jest to imperatywny, obiektowy, funkcyjny, proceduralny, refleksyjny, interpretowany i skryptowy język programowania, którego naturalnym i oryginalnym przeznaczeniem są serwery WWW, aplikacje webowe i strony internetowe. Główną zaletą PHP jest możliwość generowania stron internetowych w czasie rzeczywistym oraz jej integracja z bazą danych. W przeciwieństwie do poleceń napisanych w JavaScript, skrypty PHP nie działają w przeglądarce, są natomiast wykonywane po stronie serwera WWW. Dzięki PHP można pobierać informacje od klientów serwera WWW a także manipulować nimi.

## Serwer WWW

Wykorzystanym serwerem WWW w aplikacji jest Apache. Który jest najszerzej stosowanym serwerem HTTP w Internecie. W sierpniu 2015 jego udział wśród serwerów wynosił ponad 37%. W połączeniu z interpreterem języka skryptowego PHP i bazą danych MySQL, Apache stanowi jedno z najczęściej spotykanych środowisk w firmach oferujących miejsce na serwerach sieciowych

## Serwer baz danych

Serwerem baz danych preferowanym przez inwestora był MySQL był pisany raczej z myślą o szybkości niż kompatybilności ze standardem SQL – przez dłuższy czas MySQL nie obsługiwał nawet transakcji, co było zresztą głównym argumentem przeciwników tego projektu. MySQL obsługuje większą część obecnego standardu ANSI/ISO SQL (tj. SQL:2003). Wprowadza również swoje rozszerzenia i nowe elementy języka

# Struktura bazy danych

W projektach takich jak ten, nie jest wskazane by baza danach była zbyt przestrona. Na potrzeby projektu została utworzona baza danych „Podatki” zawierająca trzy tabele, który pola zostały nazwane adekwatnie do ich zastosowania:

* Użytkownicy – do stworzenia funkcji logowania i rejestracji niezbędne są 2 rzeczy – login i hasło. Tabela użytkownicy została stworzona właśnie do tego celu. Przechowuja dane wrażliwe takie jak login i hasło użytkowników zarejestrowanych w systemie. Tabela opiera się na kluczu głównym PESEL. Zarówno pole *PESEL* jak i *password* są polami tekstowymi. Pole *PESEL*, nie może być dłuższe niż 11 znaków, ponieważ nie powinno przechowywać niczego innego niż numer pesel



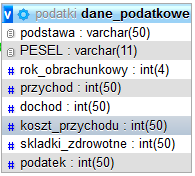
Rys 1. 1 Tabela użytkownicy

* Dane\_podstawowe – tabela zawiera informacje podstawowe użytkowników takie jak imie, nazwisko, pesel oraz nip. Jest połączona w relację z tabelą *użytkownicy* kluczem pesel. Tabela umożliwia jednoznaczne identyfikowanie użytkowników z racji oparcia o klucz obcy PESEL, który w polsce jest jednoznacznym i unikatowym identyfikatorem obywateli. Pola tabeli składają się główie z pól tekstowych, nie wlicząjąc numeru NIP, który jest typem całkowitym ogarniczonym do 10 znaków.



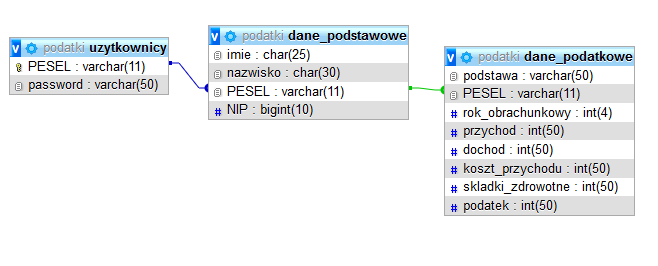
Rys 1. 2 Table dane\_podstawowe

* Dane\_podatkowe – tabela przechowująca informacje na temat danych podatkowych takich jak: podstawa opodatkowania, rok obrachunkowy, przychód, dochód, koszt uzyskania przychodu, składki na ubezpieczenie społeczne oraz wysokość podatku. Dodatkowo podobnie jak w tabeli *dane\_podstawowe*, została ustanowiona relacja z tabelą *użytkownicy* kluczem pesel. Umożliwia to dostęp do danych tylko na podstawie numeru pesel. Pola złożone są z typów liczbowych, ponieważ są to dane podatkowe umożliwiające rozliczanie podatku dochodowego, tym samym powinny być to liczby.



Rys 1. 3 Table dane\_podatkowe

Poniżej przedstawione zostały relacje między tabelami. Kluczem głównym jest tu pole *PESEL* uwzględnione w tabeli *użytkownicy*, natomiast pozostałe pola *PESEL* są kluczami obcymi opartymi właśnie o tabele *użytkownicy*



Rys 1. 4 Baza danych

Na potrzeby stworzenia bazy danych aplikacji został wykorzystany plik .sql, umożliwiający prosty import kodu odpowiedzialnego za stworzenie całej struktury bazy danych włącznie z tabelami, definicjami pól, kluczami. Plik ten został zamieszczony w dostarczonych przez wykonawce plikach źródłowych. Dodatkowo, zostało stworzone konto administratora, umożliwiające dostęp do wszystkich danych przesłanych przez użytkowników. Poniżej znajdują się główne składowe pliku

Tworzenie tabeli *użytkownicy*

|  |
| --- |
| CREATE TABLE uzytkownicy(  PESEL varchar(11) primary key,  password varchar(50) not null  ); |

Tworzenie tabeli dane\_podstawowe

|  |
| --- |
| CREATE TABLE dane\_podstawowe(  imie char(25) not null,  nazwisko char(30) not null,  PESEL varchar(11) not null,  NIP bigint(10) not null  ); |

Tworzenie tabeli dane\_podatkowe

|  |
| --- |
| CREATE TABLE dane\_podatkowe(  podstawa varchar(50) not null,  PESEL varchar(11) not null,  rok\_obrachunkowy int(4) not null,  przychod int(50) not null,  dochod int(50) not null,  koszt\_przychodu int(50) not null,  skladki\_zdrowotne int(50) not null,  podatek int(50) not null  ); |

Tworzenie konta administratora o domyślnym loginie i haśle

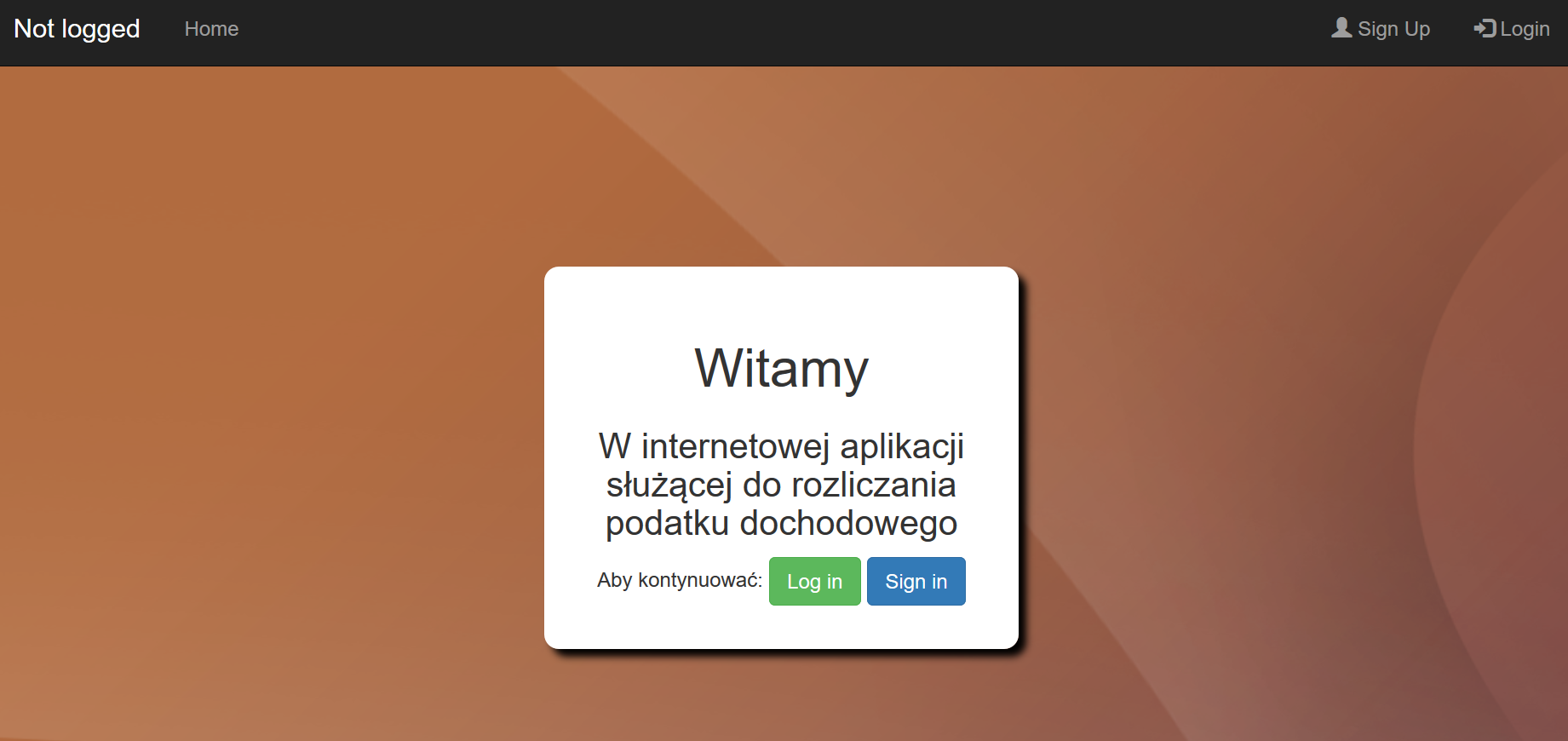
|  |
| --- |
| INSERT INTO uzytkownicy (PESEL, password)  VALUES ('Admin', 'root'); |

Utworzenie kluczy obcych

|  |
| --- |
| Alter Table dane\_podstawowe  add FOREIGN KEY (PESEL) REFERENCES uzytkownicy(PESEL);  Alter Table dane\_podatkowe  add FOREIGN KEY (PESEL) REFERENCES dane\_podstawowe(PESEL); |

# Elementy i funkcje aplikacji

Użytkownik po wejściu na stronę jako pierszwą rzecz widzi ekran powitalny, informujący go o przeznaczeniu aplikacji, a o konieczności logowania, bądź rejestracji



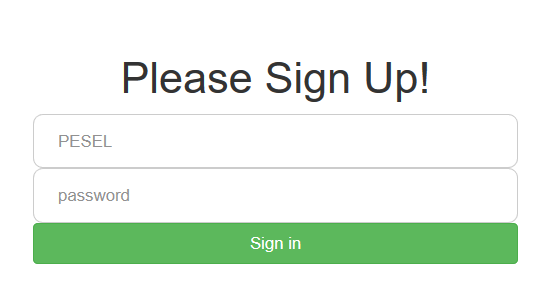
Rys 2. 1 Ekran powitalny

Użytkownik dostrzeże również bardziej subtelne aspekty strony jak pasek nawigacyjny, wysposażony w funkcje takie jak: informacje o zalogowaniu, przycisk Home przenoszący na strone początkową(na której się znajduje) oraz możliwość rejestracji i logowania

Kolejny etap jest zależny od decyzji użytkownika. Dla osób posiadających konto będzie to ekran logowania, natomiast dla osób nie posiadających konta będzie to ekran rejestracji

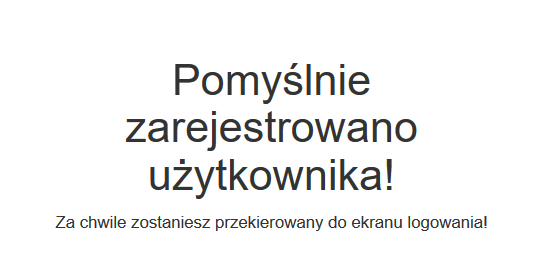
## Ekran rejestracji

Ekran rejestracji został wykonany w stylu minimalistycznym, dzieki zachowanej symetryce jest prosty, a zarazem nie przytłacza użytkownika.



Rys 2. 2 Ekran rejestracji

Gdy wszystkie dane dostarczone przez użytkownika spełniają wymagania i proces rejestracji przebiegnie pomyślnie, użytkownikowi zobaczy:

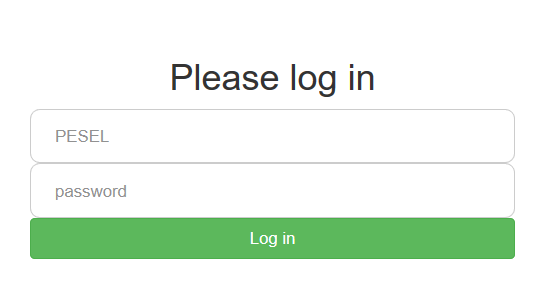


Rys 2. 3 Pomyśle zarejestrowano

I zostanie przekierowany do ekranu logowania

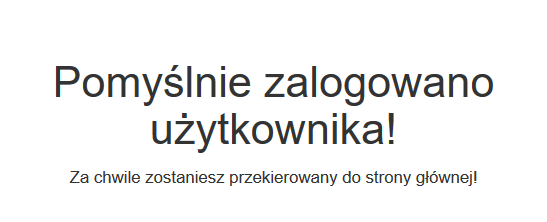
## Ekran logowania

Ekran logowania został wykonany identycznie do ekranu rejestracji



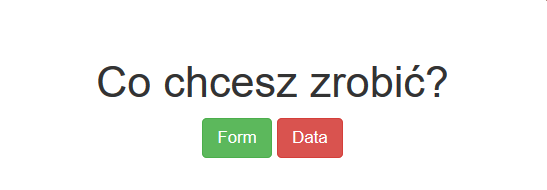
Rys 2. 4 Ekran logowania

Oto komunikat potwierdzający weryfikacje użytkownika

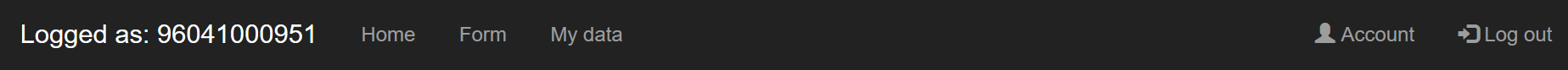


Rys 2. 5 Zalogowano

Nastepnie użytkownik jest przekierowywany do strony głównej, na której może znaleźć możliwość wypełnienia formularza lub dostęp do swoich danych



Rys 2. 6 Strona główna

Dodatkowo zostaną dla niego udostępnione nowe możliwości paska nawigacyjnego dostępne wyłącznie dla zalogowanych użytkowników.

Rys 2. Nav-nar

Użytkownik zalogowany ma podgląd swojego loginu, bezpośrednie przekierowanie do formularza, dostęp do swoich danych, a także możliwosć wglądu w swoje konto i wylogowania.

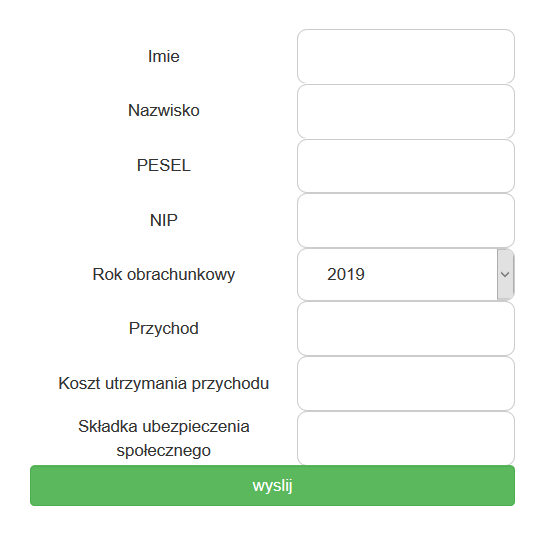
Oczywiście w przypadku dopiero co zarejestrowane użytkownika dostęp do danych będzie niemożliwy z racji ich braku, zostanie przekierowany na strone **my\_data.php** i otrzyma taką informację:



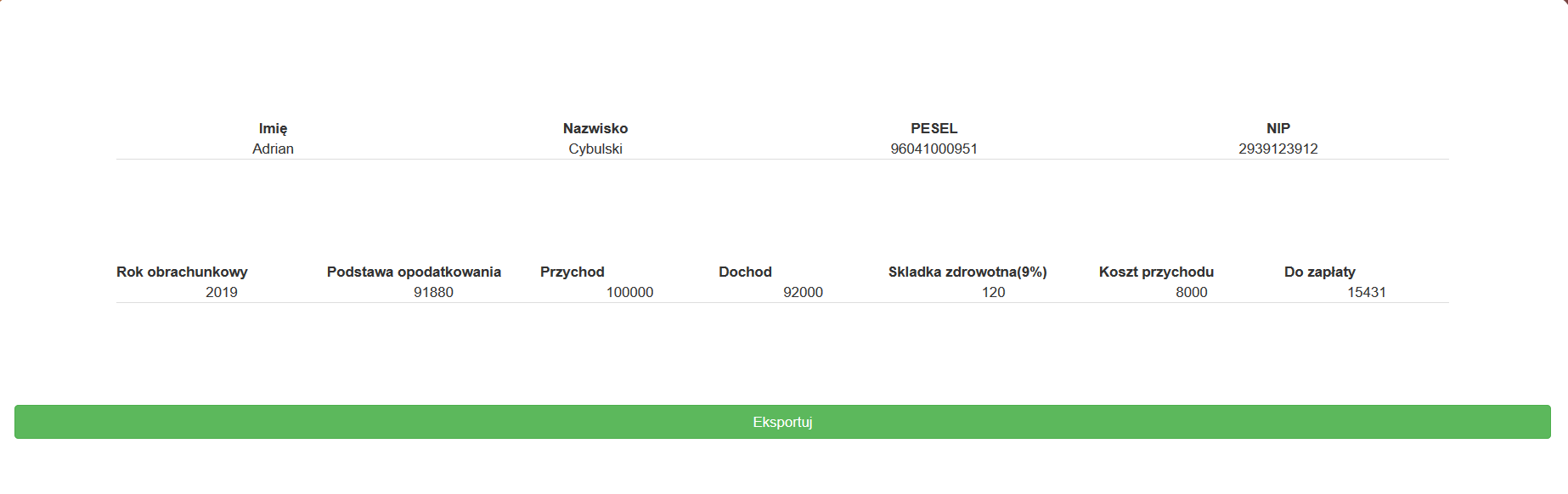
Rys 2. 8 Brak danych

Dlatego nowy użytkownik powinien wybrać opcję *Form*, dzieki które zostanie przekierowany do strony **form.php**, w które może wypełnić informację na swój temat oraz danych z jakich chciałby rozliczyć podatek dochodowy.

Formularz został wykonany w podobym minimalistycznym stylu co poprzednie.



Rys 2. 9 Formularz

 Po poprawnym wypełnieniu formularza i naciśnieciu przycisku *wyślij* użytkownik zostanie przekierowany na strone **summary.php**. Gdzie otrzyma wszystkie informacje dotyczące swojej osoby, a także informacji odnośnie podatku dochodowego

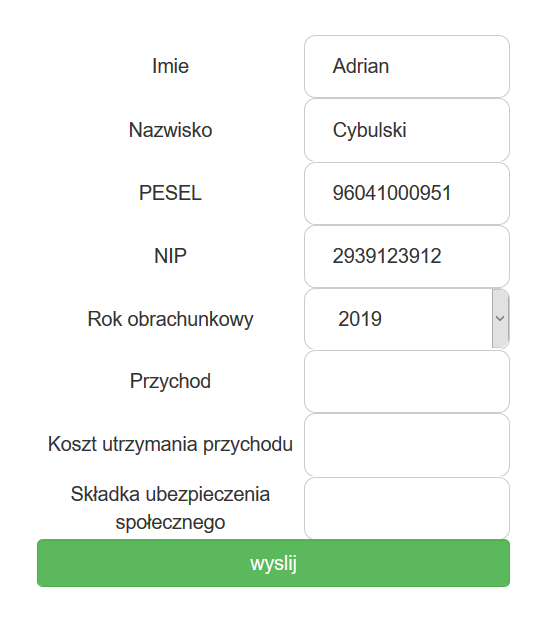
Rys 2. Wynik

Użytkownikowi została również udostępniona możliwość wyeksportowania danych do pliku Excel za pośrednictwem przycisku *Exportuj*.

 Jeśli użytkownik wyraża chęć ponownego wypełnienia formularza dotyczącego innych lat może skorzystać z zakładki *Form* w panelu nawigacyjnym znajdującym się na górze

Rys 2. Zalogowany nav-bar

Następnie jeśli dane na temat użytkownika zostały już raz wprowadzone do systemu, formularz automatycznie je uzupełni. Zostawiając użytkownikowi tylko konieczność wypełnienia dany podatkowych.



Rys 2. 12 Formularz zalogowanego

## Część administracyjna

Projekt przewiduje konto administratora mające dostęp do wszelkich danych przesłanych przez użytkowników. Zostało uwzględnione w części opisującej stuktury bazy danych. Po zalogowaniu na konto administratora, panel nawigacyjny zyskuje nową funcje *Users data*, która przekieruje administratora do strony **control\_panel.php.**



Rys 2. 14 Admin nav-bar

Rys 2. Admin dane

# Rozwiązania programistyczne

## Połączenie z baza danych

Kluczowym elementem takiego projektu jest połączenie z bazą danych. Na potrzeby tego projektu został stworzony plik **connection.php**

Jest to prosty plik konfigurujący połączenie z bazą danych, do którego dostarczane są tylko zmienne odnośnie nazwy serwera, nazwy użytkownika oraz nazwy bazy danych, z którą ma się połączyć

|  |
| --- |
| $servername = "localhost";  $username = "root";  $db\_name = "podatki";    $db\_conn = new mysqli($servername, $username, '' ,$db\_name)  or die("Connection failed: " . $db\_conn->connect\_error); |

## Ekran rejestracji

Jego kod źródłowy przechowuje prosty formularz

|  |
| --- |
| <form action="signup\_check.php" method="POST">    <div id = "form\_div\_login">  <h1>Please Sign Up!</h1>  <table id = "form\_table">  <input type = "text" maxlength="11" name = "login" required="" placeholder="PESEL">  <input type = "password" name = "password" required="" placeholder="password">  </table>  <input type="submit" class="btn btn-success" name="zaloguj" value="Sign in"/>  </div>    </form> |

Całość logiczna funkcji rejestracji przechowywana jest w pliku **signup\_check.php**, która posiada wszelkie możliwe zabezpieczenia dotyczące danych rejestracji. Zaczynając od sprawdzenia czy login dostarczony przez użytkownika jest numerem, następnie czy składa się z 11 cyfr. Dzięki tak zastosowanym zabezpieczeniem aplikacja działa szybciej, ponieważ najbardziej wymagającym czasowo procesem jest proces łączenia aplikacji z bazą danych, dlatatego połączenie występuje tylko w momencie, gdy aplikacja jest pewną, że dane dostarczone przez użytkownika są poprawnie zdefiniowane i ostatnim elementem weryfikacji jest ich porównanie z danymi dostępnymi w bazie danych. Zostaje wysłane zapytanie do bazy danych wybierające numer pesel sprawdzające czy użytkownik o podanym numerze pesel jest już w bazie. Jeśli go nie ma zostaje wysłane zapytanie, zapisujące użytkownika w bazie danych

**INSERT INTO uzytkownicy ( PESEL, password) VALUES ($login, $password);**

|  |
| --- |
| if (is\_numeric($login)) {  if (strlen($login) == 11) {  if ($result = mysqli\_query($db\_conn, $select\_check)) {  if (mysqli\_num\_rows($result) == 0) {  if (mysqli\_query($db\_conn, $insert\_user)) {  echo'<div id="form\_div\_login">';  echo"<h1>Pomyślnie zarejestrowano użytkownika!</h1>";  echo"Za chwile zostaniesz przekierowany do ekranu logowania!";  header("Refresh:3; login.php");  }  } else {  echo '<h1>Login istnieje w bazie!</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; signup.php");  }  } else {  echo '<h1>Błąd połączenia z bazą</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; signup.php");  }  } else {  echo '<h1>Pesel składa się z 11 cyfr !!!</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; signup.php");  }  } else {  echo '<h1>Pesel składa się z cyfr !!!</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; signup.php");  } |

## Ekran logowania

Ponownie został zastosowany prosty formularz, a całość logiczna przeniesiona do osobnego pliku. W tym przypadku jest to plik **login\_check.php**

|  |
| --- |
| <form action="login\_check.php" method="POST">    <div id = "form\_div\_login">  <h2> Please log in </h2>  <table id = "form\_table">  <input type = "text" name = "login" required="" placeholder="PESEL">  <input type = "password" name = "password" required="" placeholder="password">    </table>  <input type="submit" class="btn btn-success" name="zaloguj" value="Log in"/>    </div> |

Całość logiczna funkcji logowania przechowywana jest w pliku **login\_check.php**, która posiada wszelkie możliwe zabezpieczenia dotyczące danych logowania. Zaczynając od sprawdzenia czy login dostarczony przez użytkownika jest numerem, następnie czy składa się z 11 cyfr. Dzięki tak zastosowanym zabezpieczeniem aplikacja działa szybciej, ponieważ najbardziej wymagającym czasowo procesem jest proces łączenia aplikacji z bazą danych, dlatatego połączenie występuje tylko w momencie, gdy aplikacja jest pewną, że dane dostarczone przez użytkownika są poprawnie zdefiniowane i ostatnim elementem weryfikacji jest ich porównanie z danymi dostępnymi w bazie danych. Zostaje wysłane zapytanie do bazy danych wybierające numer pesel oraz hasło

**SELECT PESEL, password FROM uzytkownicy WHERE PESEL = '$login';**

Jeśli warunek *($login == $row['PESEL'] AND $password == $row['password'])* jest poprawny następuje przypisanie loginu użytkownika do zmiennej sesyjnej, informacja o pomyślnej weryfikacji loginu I hasła oraz o przekierowaniu na strone główną

|  |
| --- |
| if (is\_numeric($login)) {  if (strlen($login) == 11) {  if ($result\_check = mysqli\_query($db\_conn, $select\_check)) {  if (mysqli\_num\_rows($result\_check) > 0) {  $row = mysqli\_fetch\_assoc($result\_check);  if ($login == $row['PESEL'] AND $password == $row['password']) {  $\_SESSION['username'] = $login;  echo"<h1>Pomyślnie zalogowano użytkownika!</h1>";  echo"Za chwile zostaniesz przekierowany do strony głównej!";  header("Refresh:3; main\_page.php");  } else {  echo '<h1>Błędne hasło uzytkownika</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; login.php");  }  } else {  echo '<h1>Błędny login</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; login.php");  }  } else {  echo '<h1>Błąd połączenia z bazą</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; login.php");  }  } else {  echo '<h1>Pesel składa się z 11 cyfr !!!</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; login.php");  }  } else {  echo '<h1>Pesel składa się z cyfr</h1>';  echo"Za chwile zostaniesz przekierowany ponownie do ekranu logowania!";  header("Refresh:3; login.php");  } |

## Wyświetlanie danych

Do tego celu zostało zastosowane kilka podstron. Najbardziej obszerną jest strona **summary.php**, do której przekierowuje formularz. Strona odpowiedzialna jest za szereg zabezpieczeń, dodawania rekordów do bazy, a także ich wyświetlania.

Zaczynając od zabezpieczeń. Wysyłane zostaje zapytanie do bazy danych o nazwie *select\_check­*, którego zadaniem jest wywołanie danych z bazy danych, takich jak numer pesel oraz rok\_obrachunkowy danego użytkownika

**SELECT dane\_podstawowe.PESEL, dane\_podatkowe.rok\_obrachunkowy FROM dane\_podstawowe RIGHT JOIN dane\_podatkowe ON dane\_podstawowe.pesel = dane\_podatkowe.pesel WHERE dane\_podatkowe.PESEL = $pesel;**

Następnie przeprowadzane jest sprawdzanie danych formularza z danymi dostępnymi w bazie danych. Jeśli nie ma żadnego rekordu o numerze pesel dostarczonym przez użytkownika dane zostają dodane za pośrednictwem dwóch zapytań

**INSERT INTO dane\_podstawowe (imie, nazwisko, PESEL, NIP) VALUES ( '$imie', '$nazwisko', $pesel, $nip);**

Oraz

**INSERT INTO dane\_podatkowe (podstawa, PESEL, rok\_obrachunkowy, przychod, dochod, koszt\_przychodu, skladki\_zdrowotne, podatek) VALUES ('$podstawa', '$pesel', $rok\_obrachunkowy, $przychod, $dochod, $koszt\_przychodu, $skladki\_zdrowotne, $podatek)**

Natomiast, jeśli dany numer pesel znajduje się już w bazie danych następuje sprawdzanie, czy również znajduje się tam rok, taki jak wprowadzony przez użytkownika. Jeśli rok obrachunkowy dostępny w bazie danych pokrywa się z rokiem rozrachunkowym dostarczonym przez użytkownika zmienna sesyjna *year* zostaje ustawiona na wartość true, blokujac tym samym warunek wysłania kolejnego zapytania.

|  |
| --- |
| if ($result\_check = mysqli\_query($db\_conn, $select\_check)) {  if (mysqli\_num\_rows($result\_check) == 0) {  if (mysqli\_query($db\_conn, $insert\_podstawowe)) {  echo '#podstawowe added <br />';  } else {  echo 'Error: ' . $insert\_podstawowe . '<br />' . mysqli\_error($db\_conn);  }  if (mysqli\_query($db\_conn, $insert\_podatkowe)) {  echo '#podatkowe added <br />';  } else {  echo 'Error: ' . $insert\_podatkowe . '<br />' . mysqli\_error($db\_conn);  }  } else {  while ($row = mysqli\_fetch\_array($result\_check)) {  if ($row['rok\_obrachunkowy'] == $rok\_obrachunkowy) {  $\_SESSION['year'] = true;  echo '#rok istnieje <br />';  }  }  if ($\_SESSION['year'] == false ) {  if (mysqli\_query($db\_conn, $insert\_podatkowe)) {  echo '#podatkowe added <br />';  $\_SESSION['year'] = false;  } else {  echo 'Error: ' . $insert\_podatkowe . '<br />' . mysqli\_error($db\_conn);  }  }  }  } |

## Obliczanie podatku

Do obliczenia podatku służy szereg funkcji dostępnych w pliku **functions.php**. Zaczynając od tego, że dla poszczególnych lat mniej lub bardziej zmieniała się metoda obliczania. Dla zachowania możliwości rozwoju o kolejne lata, aplikacja dzieli się na 3 funkcje, które jako argument przyjmują zmienną *$podstawa* odpowiedzialną za przechowywanie wysokości podstawy opodatkowania

* Podatek\_2019

|  |
| --- |
| function podatek\_2019($podstawa){    if($podstawa <= 8000){  $wartosc\_podatku = 0;  }  else if($podstawa >= 8001 AND $podstawa <= 13000){  $x1 = 1440 - 883.98 \* ($podstawa - 8000) /5000;  $wartosc\_podatku = ($podstawa \* 0.18) - $x1;  }  else if($podstawa >= 13001 AND $podstawa <= 85528){  $x2 = 556.02;  $wartosc\_podatku = ($podstawa \* 0.18) - $x2;  }  else if($podstawa >= 85529 AND $podstawa <= 127000){  $x3 = 556.02 - 556.02 \* ($podstawa - 85528)/41472;  $wartosc\_podatku = 15395.04 + ($podstawa - 85528) \* 0.32 - $x3;  }  else if($podstawa >= 127001){  $wartosc\_podatku = 15395.04 + ($podstawa - 85528) \* 0.32;    }  return round($wartosc\_podatku, 2);  } |

* Podatek\_2018

|  |
| --- |
| function podatek\_2018($podstawa){  if($podstawa <= 8000){  $wartosc\_podatku = 0;  }  else if($podstawa >= 8001 AND $podstawa <= 13000){  $x1 = 1440 - 883.98 \* ($podstawa - 8000) /5000;  $wartosc\_podatku = ($podstawa \* 0.18) - $x1;  }  else if($podstawa >= 13001 AND $podstawa <= 85528){  $x2 = 556.02;  $wartosc\_podatku = ($podstawa \* 0.18) - $x2;  }  else if($podstawa >= 85529 AND $podstawa <= 127000){  $x3 = 556.02 - 556.02 \* ($podstawa - 85528)/41472;  $wartosc\_podatku = 15395.04 + ($podstawa - 85528) \* 0.32 - $x3;  }  else if($podstawa >= 127001){  $wartosc\_podatku = 15395.04 + ($podstawa - 85528) \* 0.32;  }  return round($wartosc\_podatku, 2);  } |

* Podatek\_2017

|  |
| --- |
| function podatek\_2017($podstawa){  if($podstawa <= 6000){  $wartosc\_podatku = 0;  }  else if($podstawa >= 6001 AND $podstawa <= 11000){  $x1 = 1188 - 661.98 \* ($podstawa - 6600) / 4400;  $wartosc\_podatku = ($podstawa \* 0.18) - $x1;  }  else if($podstawa >= 11001 AND $podstawa <= 85528){  $x2 = 556.02;  $wartosc\_podatku = ($podstawa \* 0.18) - $x2;  }  else if($podstawa >= 85529 AND $podstawa <= 127000){  $x3 = 556.02 - 556.02 \* ($podstawa - 85528)/41472;  $wartosc\_podatku = 15395.04 + ($podstawa - 85528) \* 0.32 - $x3;  }  else if($podstawa >= 127001){  $wartosc\_podatku = 15395.04 + ($podstawa - 85528) \* 0.32;    }    return round($wartosc\_podatku, 2);  } |

Kolejną z funkcji odpowiedzialnych za obliczanie wysokości podatki dochodowego jest funkcja *wysokosc\_podatku*, która przyjmuję jako argument zmienną *$dochod.* Natomiast wartość zmiennej dochód otrzymywana jest poprzez różniece kosztu uzyskania przychodu z przychodzem

|  |
| --- |
| function wysokosc\_podatku($dochod){  $kwota\_podatku = $dochod \* 0.91;  return $kwota\_podatku;  } |

Następnie rok obrachunkowy dostarczony przez użytkownika w formularzu filtrowany jest pod kątem odpowiedniej funkcji za pomocą trzech pętli warunkowych

|  |
| --- |
| if ($rok\_obrachunkowy == 2017) {  $podatek = wysokosc\_podatku(podatek\_2017($podstawa));  }  if ($rok\_obrachunkowy == 2018) {  $podatek = wysokosc\_podatku(podatek\_2018($podstawa));  }  if ($rok\_obrachunkowy == 2019) {  $podatek = wysokosc\_podatku(podatek\_2019($podstawa));  } |

## Wyświetlanie danych

Ostatnim elementem jest wyświetlanie danych. W tym celu wykorzystana zostaje baza danych, z której te dane pochodzą

Najpierw pobierane są dane na temat użytkownika. Wysyłane jest zapytanie do bazy danych takich jak imię, nazwisko, pesel oraz nip danego użytkownika na podstawie numeru pesel pobranego z formularza

**SELECT imie, nazwisko, PESEL, NIP FROM dane\_podstawowe WHERE PESEL = $pesel;**

Nastepnie jeśli resultatem zapytania jest co najmniej 1 rząd, tworzona zostaje tabela adekwatna do kolumn bazy danych. Rezultat zapytania przypisywany jest do zmiennej *$row*, a następnie pętla warunkowa while wpisuje wszystkie dane w poszczególne kolumny.

|  |
| --- |
| if ($result = mysqli\_query($db\_conn, $select\_podstawowe)) {  if (mysqli\_num\_rows($result) > 0) {  echo '<table class = "db">';  echo "<tr>";  echo "<th><center>Imię</th>";  echo "<th><center>Nazwisko</th>";  echo "<th><center>PESEL</th>";  echo "<th><center>NIP</th>";  echo "</tr>";  while ($row = mysqli\_fetch\_array($result)) {  echo "<tr>";  echo "<td>" . $row['imie'] . "</td>";  echo "<td>" . $row['nazwisko'] . "</td>";  echo "<td>" . $row['PESEL'] . "</td>";  echo "<td>" . $row['NIP'] . "</td>";  echo "</tr>";  }  } else {  echo "Nie znaleziono rekordow";  }  } else {    } |

Po utworzeniu tabeli przechowującej dane podstawowe użytkownika kolejne zapytanie jest wysyłane do bazy danych – tym razem odnośnie dany podatkowych. Ponownie jak w poprzednim przypadku dostęp do danych podatkowych dostępny jest tylko na podstawie danego przez użytkownika numeru pesel

**SELECT dane\_podatkowe.podstawa, dane\_podstawowe.PESEL, dane\_podatkowe.rok\_obrachunkowy, dane\_podatkowe.przychod, dane\_podatkowe.dochod, dane\_podatkowe.koszt\_przychodu, dane\_podatkowe.skladki\_zdrowotne, dane\_podatkowe.podatek FROM dane\_podstawowe RIGHT JOIN dane\_podatkowe ON dane\_podstawowe.pesel = dane\_podatkowe.pesel WHERE dane\_podatkowe.PESEL = $pesel;**

Następnie jeśli resultatem jest co najmniej jeden rząd tworzona zostaje tabela o kolumnach adekwatnych do danych dostarczonych przez baze danych. Wartość zmiennej *$result* przypisywana jest do zmiennej *$row* i pętla warunkowa while wypisuje wszystkie dane do kolumn i wierszy.

|  |
| --- |
| if ($result = mysqli\_query($db\_conn, $select\_podatkowe)) {  if (mysqli\_num\_rows($result) > 0) {  echo '<table class="db">';  echo "<tr>";  echo "<th><center>Rok obrachunkowy</th>";  echo "<th><center>Podstawa opodatkowania</th>";  echo "<th><center>Przychod</th>";  echo "<th><center>Dochod</th>";  echo "<th><center>Skladka zdrowotna(9%)</th>";  echo "<th><center>Koszt przychodu</th>";  echo "<th><center>Do zapłaty</th>";  echo "</tr>";  } else {    }  while ($row = mysqli\_fetch\_array($result)) {  echo "<tr>";  echo "<td>" . $row['rok\_obrachunkowy'] . "</td>";  echo "<td>" . $row['podstawa'] . "</td>";  echo "<td>" . $row['przychod'] . "</td>";  echo "<td>" . $row['dochod'] . "</td>";  echo "<td>" . $row['skladki\_zdrowotne'] . "</td>";  echo "<td>" . $row['koszt\_przychodu'] . "</td>";  echo "<td>" . $row['podatek'] . "</td>";  echo "</tr>";  }  echo "</table>";  echo "<div>";  mysqli\_free\_result($result);  } |

## Panel administracyjny

Podobnie jak w przypadku wyświetlania informacji dla użytkownika, wysyłane jest zapytanie do bazy danych. Jednak tym razem nie ma ogarniczenia **WHERE** i wszystkie dane są wybierane i wypisywane

SELECT dane\_podstawowe.imie, dane\_podstawowe.nazwisko, dane\_podstawowe.PESEL, dane\_podstawowe.NIP, dane\_podatkowe.podstawa, dane\_podatkowe.rok\_obrachunkowy, dane\_podatkowe.przychod, dane\_podatkowe.dochod, dane\_podatkowe.koszt\_przychodu, dane\_podatkowe.skladki\_zdrowotne FROM dane\_podstawowe JOIN dane\_podatkoweON dane\_podstawowe.pesel = dane\_podatkowe.pesel;

|  |
| --- |
| if($result = mysqli\_query($db\_conn, $select\_admin)){  if(mysqli\_num\_rows($result) > 0){  //echo $result;  echo '<div class = "table\_div">';  echo '<table class="db">';  echo "<tr>";  echo "<th>Imię</th>";  echo "<th>Nazwisko</th>";  echo "<th>PESEL</th>";  echo "<th>NIP</th>";  echo "<th>Podstawa opodatkowania</th>";  echo "<th>Rok obrachunkowy</th>";  echo "<th>Przychod</th>";  echo "<th>Dochod</th>";  echo "<th>Koszt przychodu</th>";  echo "<th>Skladki zdrowotne</th>";  echo "<th>Do zapłaty</th>";  echo "</tr>";    while($row = mysqli\_fetch\_array($result)){  echo "<tr>";  echo "<td>" . $row['imie'] . "</td>";  echo "<td>" . $row['nazwisko'] . "</td>";  echo "<td>" . $row['PESEL'] . "</td>";  echo "<td>" . $row['NIP'] . "</td>";  echo "<td>" . $row['podstawa'] . "</td>";  echo "<td>" . $row['rok\_obrachunkowy'] . "</td>";  echo "<td>" . $row['przychod'] . "</td>";  echo "<td>" . $row['dochod'] . "</td>";  echo "<td>" . $row['koszt\_przychodu'] . "</td>";  echo "<td>" . $row['skladki\_zdrowotne'] . "</td>";  echo "<td>" . $row['podatek'] . "</td>";  echo "</tr>";  }  echo "</table>";  echo "<div>";  mysqli\_free\_result($result);  }  else{  echo "Nie znaleziono rekordow";  echo $result;  }  } |

## Eksport danych

Kolejnym kluczowym elementem wymaganym przez inwestora była możliwość eksportu danych do formatu .xls. Na potrzeby takiego wymagania utworzony został plik o nazwie **export.php**, którego zadaniem jest zapisanie danych w pliku .xml. Pomimo skomplikowanego założenia, funkcja okazała się prosta. Głównym jej elementem jest pobranie danych z bazy danych i stworzenie dla nich tabeli w zmiennej *$output* identycznie jak w przypadku powyższych przykładów. Następnie za pomocą funkcji dostępnych w php:

**header('Content-Disposition: attachment; filename=Twoje\_dane\_podatkowe.xls');**

**echo $output;**

Dane przekierowywane są do załącznika o nazwie **Twoje\_dane\_podatkowe** umożliwiajac ich ściągniecie na swój komputer.

|  |
| --- |
| if(mysqli\_num\_rows($result) > 0)  {  $output .= '  <table class="table" bordered="1">  <tr>  <th>Imie</th>  <th>Nazwisko</th>  <th>PESEL</th>  <th>NIP</th>  <th>rok\_obrachunkowy</th>  <th>podstawa\_opodatkowania</th>  <th>dochod</th>  <th>koszt\_przychodu</th>  <th>skladka\_ubezpieczenia</th>  <th>podatek</th>    </tr>  ';  while($row = mysqli\_fetch\_array($result))  {  $output .= '  <tr>  <td>'.$row["imie"].'</td>  <td>'.$row["nazwisko"].'</td>  <td>'.$row["PESEL"].'</td>  <td>'.$row["NIP"].'</td>  <td>'.$row["rok\_obrachunkowy"].'</td>  <td>'.$row["podstawa"].'</td>  <td>'.$row["przychod"].'</td>  <td>'.$row["dochod"].'</td>  <td>'.$row["koszt\_przychodu"].'</td>  <td>'.$row["skladki\_zdrowotne"].'</td>  </tr>  ';  }  $output .= '</table>';  header('Content-Type: application/xls');  header('Content-Disposition: attachment; filename=Twoje\_dane\_podatkowe.xls');  echo $output;  header("refresh:0; summary.php");  } |

# Podsumowanie

Projekt okazał się edukacyjny pod wieloma względami, zaczynając od części czysto technicznej – dzięki temu, że aplikacja jest tak rozbudowana pozwoliła poznać wiele nieznanych dotąd funkcji jezyka zarówno php jak i mysql. Dzięki nowym założeniom projektowym przybywającym z tygodnia na tydzień zyskałem niezbędne doświadczenie w pracy z inwestorem. Mimo skomplikowania projektu, wszystkie napotkane problemy można było rozwiązać samodzielnie lub z asystą prowadzącego.