CENTRALNA KOMISJA EGZAMINACYJNA

Arkusz zawiera informacje prawnie chronione do momentu rozpoczęcia egzaminu

Nazwa kwalifikacji: **Programowanie, tworzenie i administrowanie stronami internetowymi i bazami danych**
Oznaczenie kwalifikacji: **EE.09**
Numer zadania: **06**
Wersja arkusza: **SG**

Wypełnia zdający

Numer PESEL zdającego*

Miejsce na naklejkę z numerem PESEL i z kodem ośrodka

Czas trwania egzaminu: **150** minut. EE.09-06-21.01-SG

EGZAMIN POTWIERDZAJĄCY KWALIFIKACJE W ZAWODZIE

Rok 2021

CZĘŚĆ PRAKTYCZNA

PODSTAWA PROGRAMOWA 2017

Instrukcja dla zdającego

1. Na pierwszej stronie arkusza egzaminacyjnego wpisz w oznaczonym miejscu swój numer PESEL i naklej naklejkę z numerem PESEL i z kodem ośrodka.
2. Na KARCIE OCENY w oznaczonym miejscu przyklej naklejkę z numerem PESEL oraz wpisz:
   - swój numer PESEL*,
   - oznaczenie kwalifikacji,
   - numer zadania,
   - numer stanowiska.
3. Sprawdź, czy arkusz egzaminacyjny zawiera 6 stron i nie zawiera błędów. Ewentualny brak stron lub inne usterki zgłoś przez podniesienie ręki przewodniczącemu zespołu nadzorującego.
4. Zapoznaj się z treścią zadania oraz stanowiskiem egzaminacyjnym. Masz na to 10 minut. Czas ten nie jest wliczany do czasu trwania egzaminu.
5. Czas rozpoczęcia i zakończenia pracy zapisze w widocznym miejscu przewodniczący zespołu nadzorującego.
6. Wykonaj samodzielnie zadanie egzaminacyjne. Przestrzegaj zasad bezpieczeństwa i organizacji pracy.
7. Po zakończeniu wykonania zadania pozostaw arkusz egzaminacyjny z rezultatami oraz KARTĘ OCENY na swoim stanowisku lub w miejscu wskazanym przez przewodniczącego zespołu nadzorującego.
8. Po uzyskaniu zgody zespołu nadzorującego możesz opuścić salę/miejsce przeprowadzania egzaminu.

Powodzenia!

** w przypadku braku numeru PESEL – seria i numer paszportu lub innego dokumentu potwierdzającego tożsamość*

Zadanie egzaminacyjne

Wykonaj aplikację internetową portu lotniczego, wykorzystując pakiet XAMPP oraz edytor zaznaczający składnię.

Aby wykonać zadanie, zaloguj się na konto **Egzamin** bez hasła. Na pulpicie znajdziesz archiwum ZIP o nazwie *z6.zip* zabezpieczone hasłem: **@SaMolotY9%**

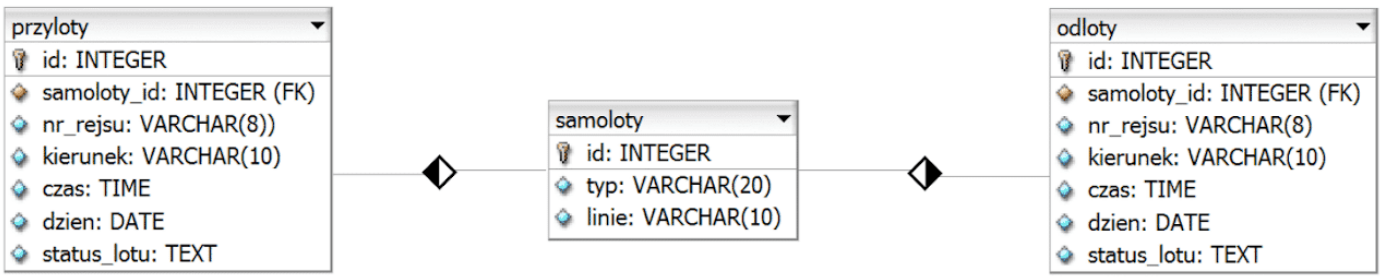
Archiwum należy rozpakować.

Na pulpicie konta **Egzamin** utwórz folder. Jako nazwy folderu użyj swojego numeru PESEL. Umieść w tym folderze rozpakowane pliki.

Po skończonej pracy wyniki zapisz również w tym folderze.

Operacje na bazie danych

Do wykonania zadania należy użyć tabel: *przyloty, samoloty, odloty* przedstawionych na obrazie 1.



Obraz 1. Tabele wykorzystane w zadaniu

Uruchom usługi MySQL i Apache za pomocą XAMPP Control Panel. Za pomocą narzędzia phpMyAdmin wykonaj podane operacje na bazie danych:

- Utwórz bazę danych o nazwie *egzamin*
- Do bazy *egzamin* zaimportuj plik *egzamin.sql* z rozpakowanego archiwum
- Wykonaj zrzut ekranu po imporcie. Zrzut zapisz w folderze z numerem PESEL, w formacie PNG i nazwij *import*. Nie kadruj zrzutu. Powinien on obejmować cały ekran monitora, z widocznym paskiem zadań. Na zrzucie powinny być widoczne elementy wskazujące na poprawnie wykonany import tabel
- Zapisz i wykonaj zapytania SQL działające na bazie *egzamin*. Zapytania zapisz w pliku *kwerendy.txt*, w folderze z numerem PESEL. Wykonaj zrzuty ekranu przedstawiające wyniki działania kwerend. Zrzuty zapisz w formacie JPG i nadaj im nazwy *kw1*, *kw2*, *kw3*, *kw4*. Zrzuty powinny obejmować cały ekran monitora z widocznym paskiem zadań
  - Zapytanie 1: wybierające jedynie pola id, nr_rejsu, czas, kierunek i status_lotu z tabeli *odloty* posortowane malejąco według czasu
  - Zapytanie 2: wyznaczające najwcześniejszy (najmniejszy) czas odlotu dla lotów, których czas odlotu jest pomiędzy 10:10 ÷ 10:19
  - Zapytanie 3: wybierające jedynie pola nr_rejsu z tabeli *odloty* oraz linie z tabeli *samoloty*, dla lotów których kierunkiem jest Malta; zapytanie wykorzystuje relację
  - Zapytanie 4: aktualizujące w tabeli *przyloty* pole status_lotu na: „planowy” dla lotu o numerze rejsu LH9829

Witryna internetowa

LP.	NUMER REJSU	CZAS	KIERUNEK	STATUS
9	LH9331	10:22:00	MONACHIUM	
8	LH9888	10:19:00	HAMBURG	
7	LH9821	10:16:00	BERLIN	
6	LX5622	10:13:00	WIEDEN	
5	LX5673	10:06:00	MALTA	
4	LX5647	10:03:00	LONDYN LT	ODPRAWA
10	W68769	09:56:00	ZURYCH	BOARDING
3	W63425	09:45:00	WARSZAWA	ODPRAWA
1	FR1646	09:20:00	NEAPOL	WYSTARTOWAL
2	FR1327	09:10:00	ALICANTE	OPOZNIONY 10 MIN

Obraz 2. Witryna internetowa

Przygotowanie grafiki:

- Plik *zad6.png*, wypakowany z archiwum, należy przeskalować z zachowaniem proporcji tak, aby jego wysokość wynosiła dokładnie 150 px. Należy zachować przezroczystość obrazu

Cechy witryny:

- Składa się ze strony o nazwie *airport.php*
- Zastosowany właściwy standard kodowania polskich znaków
- Tytuł strony widoczny na karcie przeglądarki: „Odloty samolotów”
- Arkusz stylów w pliku o nazwie *styl6.css* prawidłowo połączony z kodem strony
- Podział strony na bloki: dwa bloki banera, poniżej blok główny, następnie trzy bloki stopki. Podział zrealizowany za pomocą znaczników sekcji, zgodnie z obrazem 2
- Zawartość pierwszego bloku banera: nagłówek drugiego stopnia o treści „Odloty z lotniska”
- Zawartość drugiego bloku banera: obraz *zad6.png* z tekstem alternatywnym „logotyp”
- Zawartość bloku głównego:
  - Nagłówek czwartego stopnia o treści: „tabela odlotów”
  - Tabela o pięciu kolumnach
  - Nagłówki kolumn mają podpisy: „lp.”, „numer rejsu”, „czas”, „kierunek”, „status”
  - Zawartość tabeli jest wypełniana skryptem 1
- Zawartość pierwszego bloku stopki:
  - Odnośnik o treści „Pobierz obraz” otwierający plik *kw1.jpg* w nowym oknie
- Zawartość drugiego bloku stopki: efekt działania skryptu 2
- Zawartość trzeciego bloku stopki: tekst: „Autor: ”, dalej wstawiony numer PESEL zdającego

Styl CSS witryny internetowej

Cechy formatowania CSS:

- Dla pierwszego bloku banera: wyrównanie tekstu do środka, szerokość 75%, wysokość 150 px, rozmiar czcionki 200%
- Dla drugiego bloku banera: szerokość 25%, wysokość 150 px
- Dla bloku głównego: tło koloru RGB 128, 0, 0; marginesy wewnętrzne 50 px, tekst wielkimi literami (niezależnie od tego jak tekst został zapisany w HTML, wymagana transformacja na wielkie litery)

- Dla pierwszego i trzeciego bloku stopki: szerokość 20%, margines wewnętrzny górny: 90 px (tylko górny)
- Dla drugiego bloku stopki: wyrównanie tekstu do środka, szerokość 60%
- Dla selektora body: krój czcionki Arial, kolor tła RGB 244, 164, 96; biały kolor czcionki
- Dla selektora paragrafu (akapitu): rozmiar czcionki 150%, obramowanie 2 px, linią kropkowaną koloru RGB 169, 169, 169
- W momencie najechania kursorem na paragraf, jego kolor tła zmienia się na RGB 169, 169, 169
- Dla selektora tabeli: szerokość 100%, obramowanie tabeli i komórki ma być połączone (collapse)
- Dla selektora komórki danych i komórki nagłówkowej tabeli: obramowanie 1 px, linią ciągłą o kolorze RGB 192, 192, 192; marginesy wewnętrzne 5 px

Skrypt połączenia z bazą

W tabeli 1 podano wybór funkcji PHP do obsługi bazy danych. Wymagania dotyczące skryptów:

- Oba skrypty napisane w języku PHP, w pliku *airport.php*
- Działanie skryptu 1:
  - Skrypt łączy się z serwerem bazodanowym na *localhost*, użytkownik **root** bez hasła, baza danych o nazwie *egzamin*
  - Skrypt wysyła do bazy danych zapytanie 1
  - Dane otrzymane z bazy są umieszczane w kolejnych wierszach tabeli, w odpowiednich komórkach, tak jak na Obrazie 2
  - Po wykonaniu operacji na bazie skrypt zamyka połączenie z serwerem
- Działanie skryptu 2:
  - W oparciu o mechanizm ciasteczek skrypt wypisuje na stronie komunikaty:
    - Gdy odwiedzający wejdzie na stronę pierwszy raz zakładane jest ciasteczko z czasem trwania 1 godzina od utworzenia oraz wyświetlany jest w paragrafie, czcionką pochyloną, komunikat: „Dzień dobry! Sprawdź regulamin naszej strony”
    - Jeżeli odwiedzający w ciągu godziny od poprzedniego wejścia wejdzie ponownie na stronę wyświetlany jest w paragrafie, czcionką pogrubioną, komunikat „Miło nam, że nas znowu odwiedziłeś”

Tabela 1. Wybór funkcji języka PHP do obsługi bazy MySQL i MariaDB

Funkcje biblioteki MySQLi	Zwracana wartość
mysqli_connect(*serwer, użytkownik, hasło, nazwa_bazy*)	id połączenia lub FALSE, gdy niepowodzenie
mysqli_select_db(*id_polaczenia, nazwa_bazy*)	TRUE/FALSE w zależności od stanu operacji
mysqli_error(*id_polaczenia*)	Tekst komunikatu błędu
mysqli_close(*id_polaczenia*)	TRUE/FALSE w zależności od stanu operacji
mysqli_query(*id_polaczenia, zapytanie*)	Wynik zapytania
mysqli_fetch_row(*wynik_zapytania*)	Tablica numeryczna odpowiadająca wierszowi zapytania
mysqli_fetch_array(*wynik_zapytania*)	Tablica zawierająca kolejny wiersz z podanych w wyniku zapytania lub FALSE, jeżeli nie ma więcej wierszy w wyniku zapytania
mysqli_num_rows(*wynik_zapytania*)	Liczba wierszy w podanym zapytaniu
mysqli_num_fields(*wynik_zapytania*)	Liczba kolumn w podanym zapytaniu

bool **setcookie** (string $name , string $value = "" , int $expire = 0 , string $path = "" , string $domain = "" , bool $secure = false , bool $httponly = false)

setcookie() defines a cookie to be sent along with the rest of the HTTP headers. Like other headers, cookies must be sent *before* any output from your script (this is a protocol restriction). This requires that you place calls to this function prior to any output, including *<html>* and *<head>* tags as well as any whitespace.

Once the cookies have been set, they can be accessed on the next page load with the $_COOKIE array. Cookie values may also exist in $_REQUEST.

`name`

The name of the cookie.

`value`

The value of the cookie. This value is stored on the clients computer; do not store sensitive information. Assuming the `name` is *'cookiename'*, this value is retrieved through $_COOKIE['cookiename']

`expire`

The time the cookie expires. This is a Unix timestamp so is in number of seconds since the epoch. In other words, you'll most likely set this with the time() function plus the number of seconds before you want it to expire. Or you might use mktime(). *time()+60*60*24*30* will set the cookie to expire in 30 days. If set to 0, or omitted, the cookie will expire at the end of the session (when the browser closes).

`path`

The path on the server in which the cookie will be available on. If set to *'/'*, the cookie will be available within the entire `domain`. If set to *'/foo/'*, the cookie will only be available within the */foo/* directory and all sub-directories such as */foo/bar/* of `domain`. The default value is the current directory that the cookie is being set in.

`domain`

The (sub)domain that the cookie is available to. Setting this to a subdomain (such as *'www.example.com'*) will make the cookie available to that subdomain and all other sub-domains of it (i.e. w2.www.example.com). To make the cookie available to the whole domain (including all subdomains of it), simply set the value to the domain name (*'example.com'*, in this case).

`secure`

Indicates that the cookie should only be transmitted over a secure HTTPS connection from the client. When set to **TRUE**, the cookie will only be set if a secure connection exists. On the server-side, it's on the programmer to send this kind of cookie only on secure connection (e.g. with respect to $_SERVER["HTTPS"]).

`httponly`

When **TRUE** the cookie will be made accessible only through the HTTP protocol. This means that the cookie won't be accessible by scripting languages, such as JavaScript. It has been suggested that this setting can effectively help to reduce identity theft through XSS attacks (although it is not supported by all browsers), but that claim is often disputed. Added in PHP 5.2.0. **TRUE** or **FALSE**